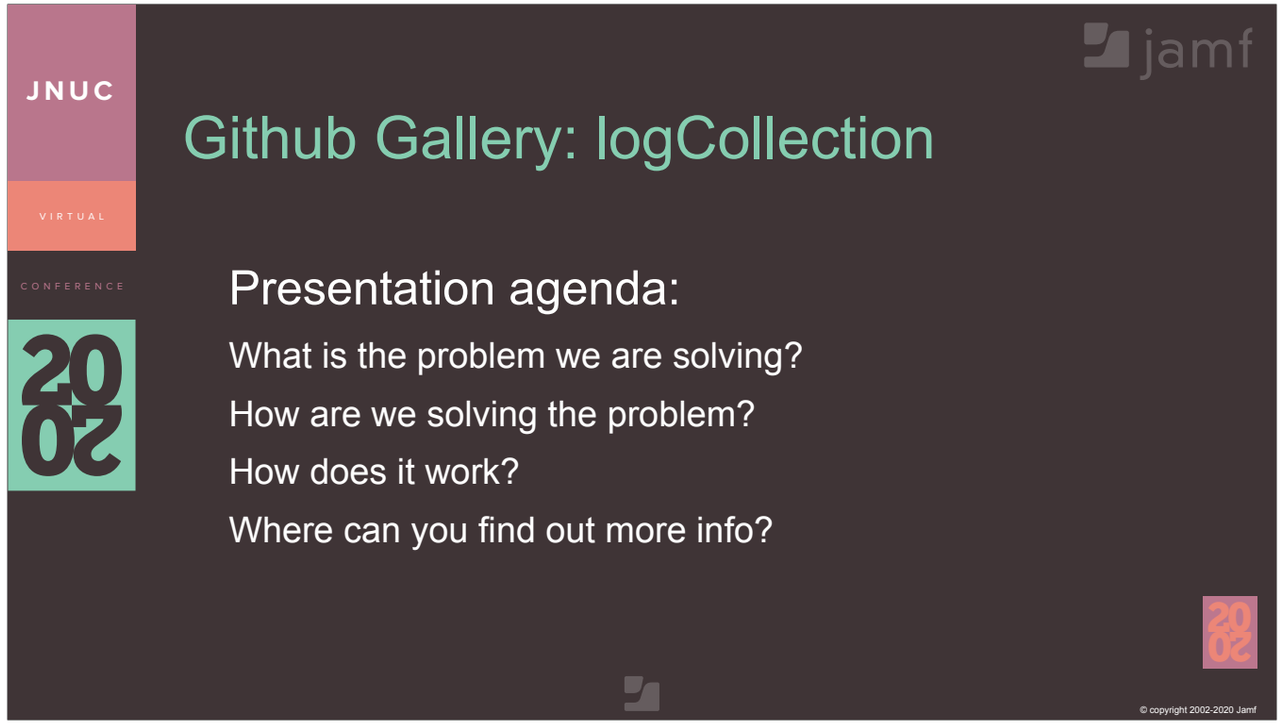
UP NEXT

# Github Gallery: logCollection

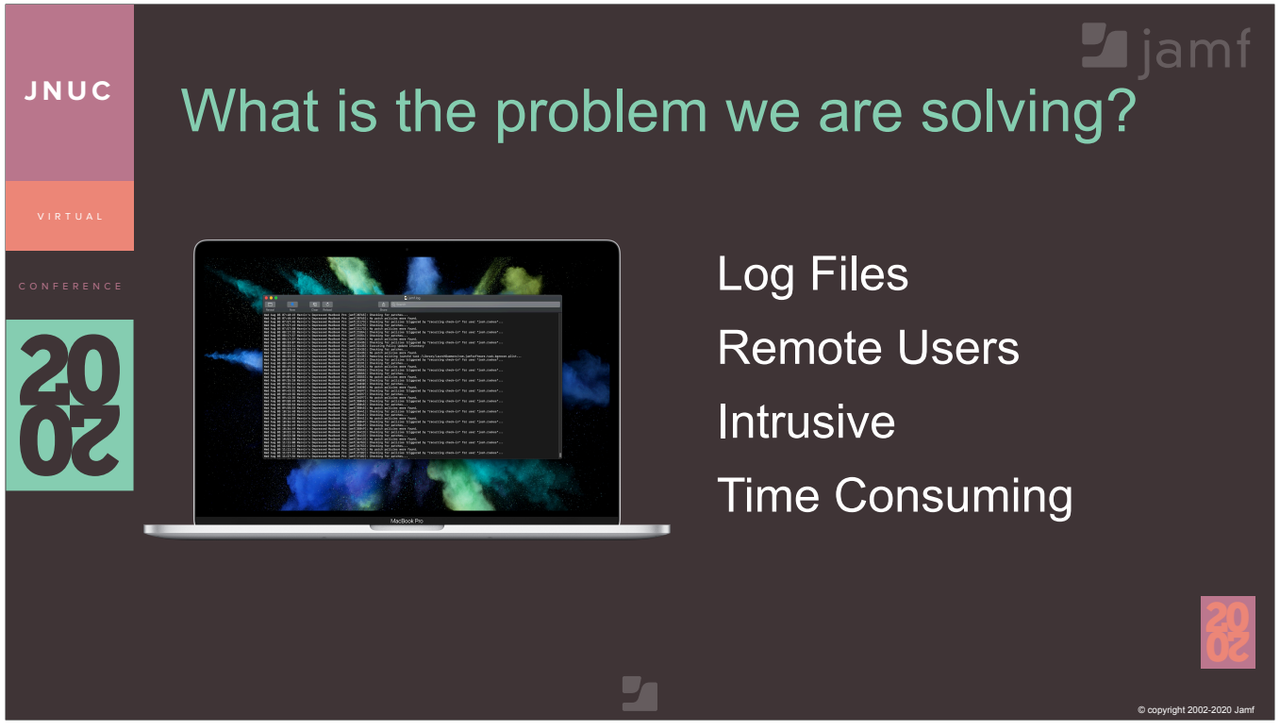

Hello Everyone -

My name is Josh Roskos and I am a Sr. Enterprise Customer Success Manager here at Jamf. You might recognize me from some other popular Github repos, such as macOSUpgrade.




But today, we're here to talk about logCollection!

Will start out by diving into what the problem is that this workflow is trying to solve; how we're solving it, as well as diving into the code that makes it work; and finally, where you can find this solution and how to get it all setup.



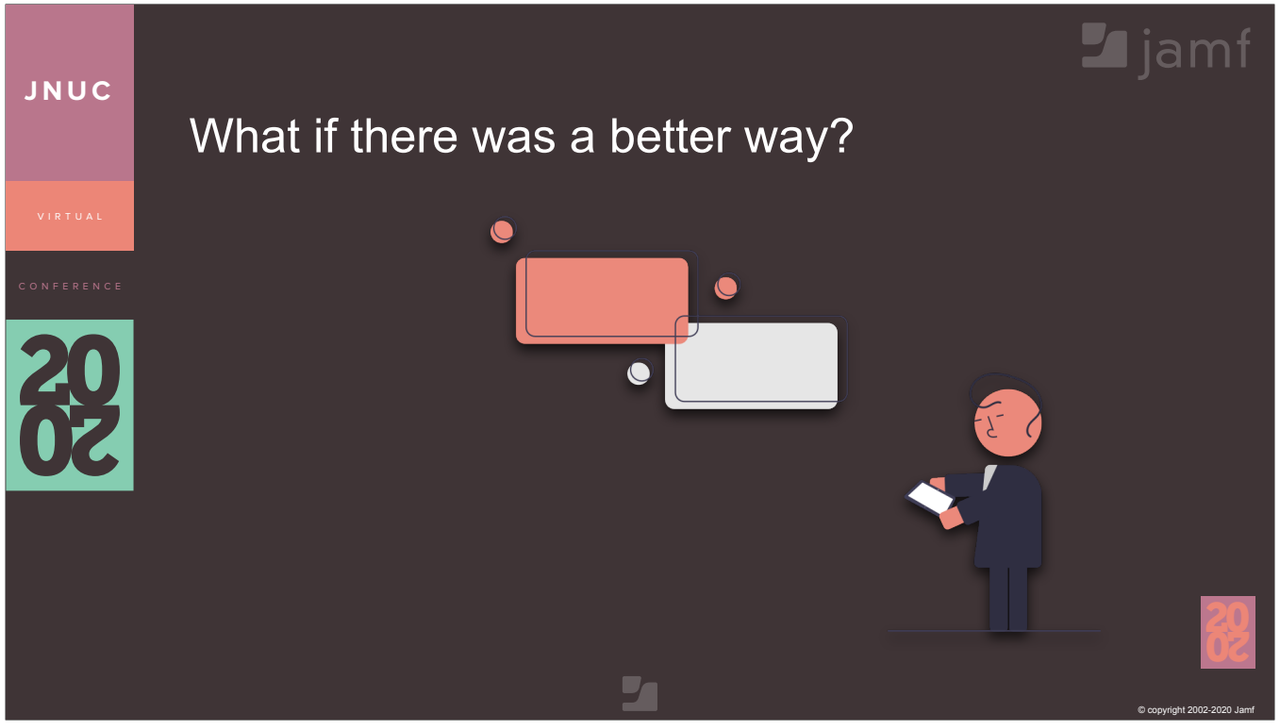

Alright, so why did I create this workflow?

As you're all aware, there are countless log files on every device and many times, these log files can aid us greatly in determining what is happening on a users Mac when they run into problems.

[CLICK] Even before COVID-19, our users were not typically "near" IT. If we needed logs, we'd have to establish a screen sharing session to gather the files, or send a technician out to do a desk-side visit.

[CLICK] This all is very intrusive to the user [CLICK] and extremely time consuming for both them, and your IT staff.

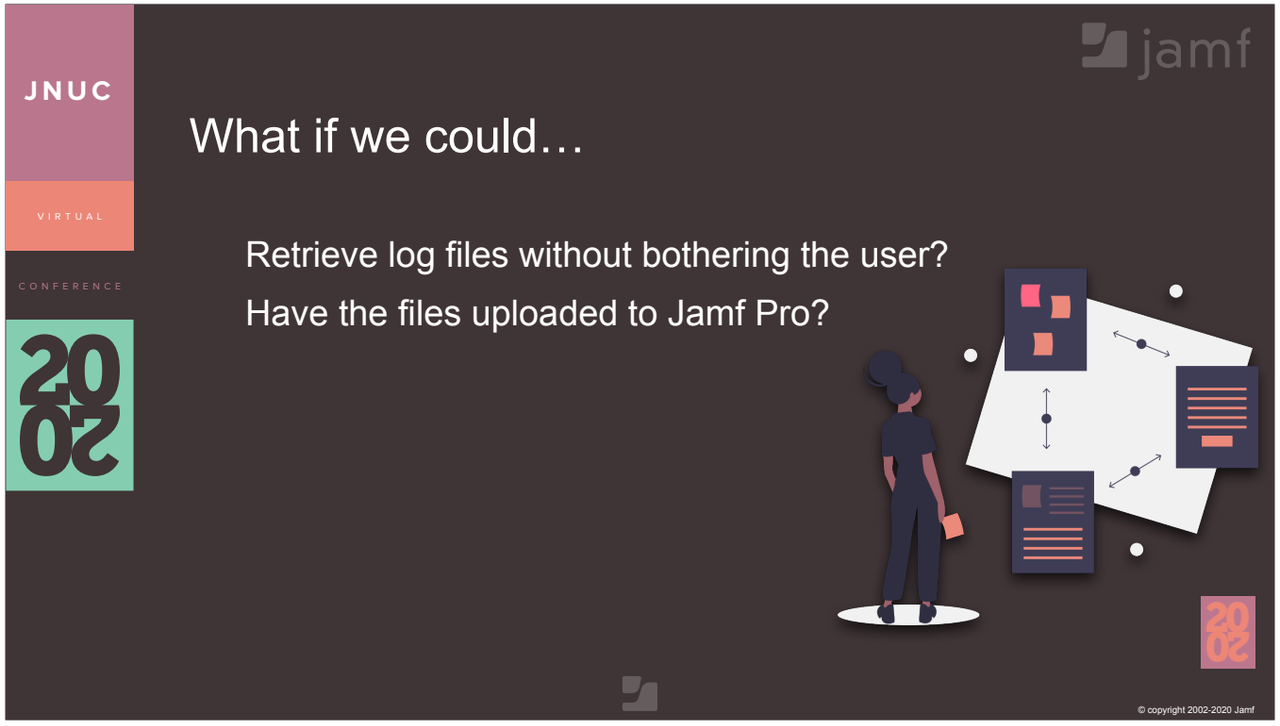
Now, in the new world we are in, where users are all working from their own homes. We can't just send a technician to their house, and depending on what you've used in the past for screen sharing, it may not work for those off the corporate network.




So…

Just how do we capture these logs from a user’s Mac, without disrupting their own business critical functions they provide?

What if there was an entirely new and better way of gathering these log files?



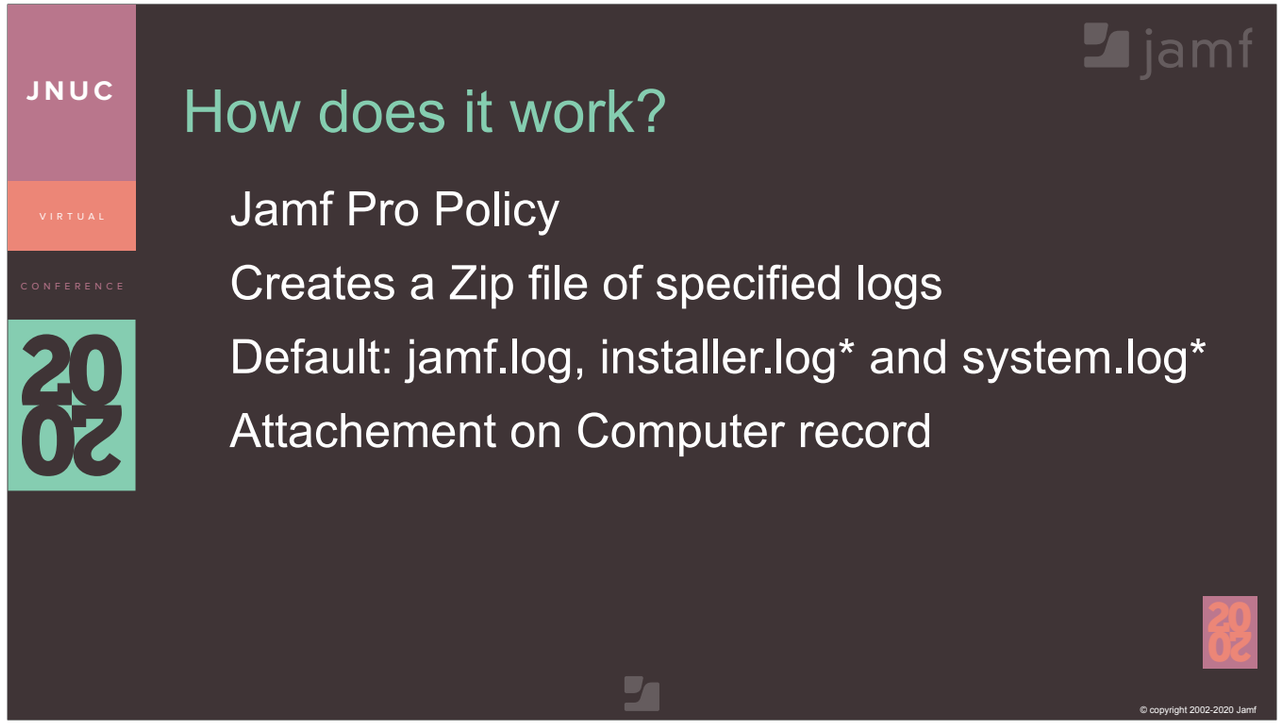

What if we good silently, retrieve the needed log files from the users machine without them really even noticing?

What if those files were uploaded to Jamf Pro, where you the admin or technician could easily retrieve them?

Well, now they can…


…with logCollection

This workflow can be used either via a policy scoped to a specific Mac or via Self Service; quickly gathering the needed logs and uploading them to Jamf Pro where you…can easily retrieve them.




That’s amazing! But really, how does it all work?

We first need to upload the script from the Github repo (which I’ll direct you all too shortly) and use a Policy inside Jamf Pro to distribute it. When this policy is scoped to either a specific machine to run or when called via Self Service, it will create an archive of all the log files you’ve specified in the policy config.

By default, I recommend *jamf.log*; *system.log**; and the *installer.log**
Now, you may wonder what the asterisk is for, well…by adding an asterisks to the end of the log file path, it will also grab any rollover log files.

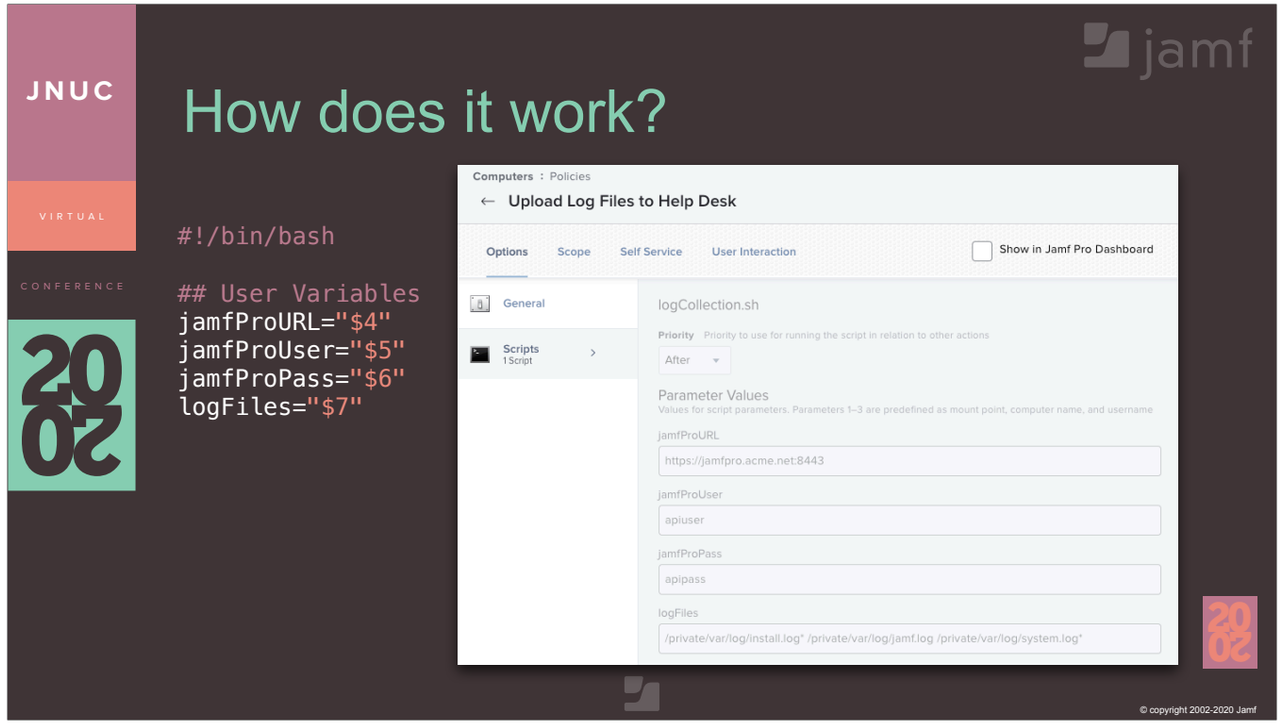
Okay, once we’ve gotten the archive built, we simply upload it to the Jamf Pro device record, under attachments.

Now, let’s take a look behind the scenes to see how the magic of how this works!


Diving in here, this is the entire script, a whopping 20 lines. Sometimes simpler is better

But lets break it down by section to go over what each respective blocks accomplish.

First…




…we'll look at the **User Variable** block.

Here, we are really just specifying some variables that are needed to complete the script.

The **jamfProURL** is the FQDN to your Jamf Pro server, this is the url where you access the admin interface to create policies and deploy applications.

Next up, is the **jamfProUser** and **jamfProPass** variables. These variables are for a local Jamf Pro user, that has the required permissions needed for this script. And if you want to use encryptedStrings to help obfuscate the credentials, there is a version with that already setup and ready to go on the Github repo, along with the specific permissions needed for the local Jamf Pro user account.

Lastly, the **logFiles** variable. This variable is where you can configure the log files you are wanting to collect, such as the previously mentioned jamf.log, installer.log and system.log files.

[CLICK]

All of these, are recommended to be configured on the Policy in Jamf Pro and not in the script itself.

JNUC

VIRTUAL

CONFERENCE

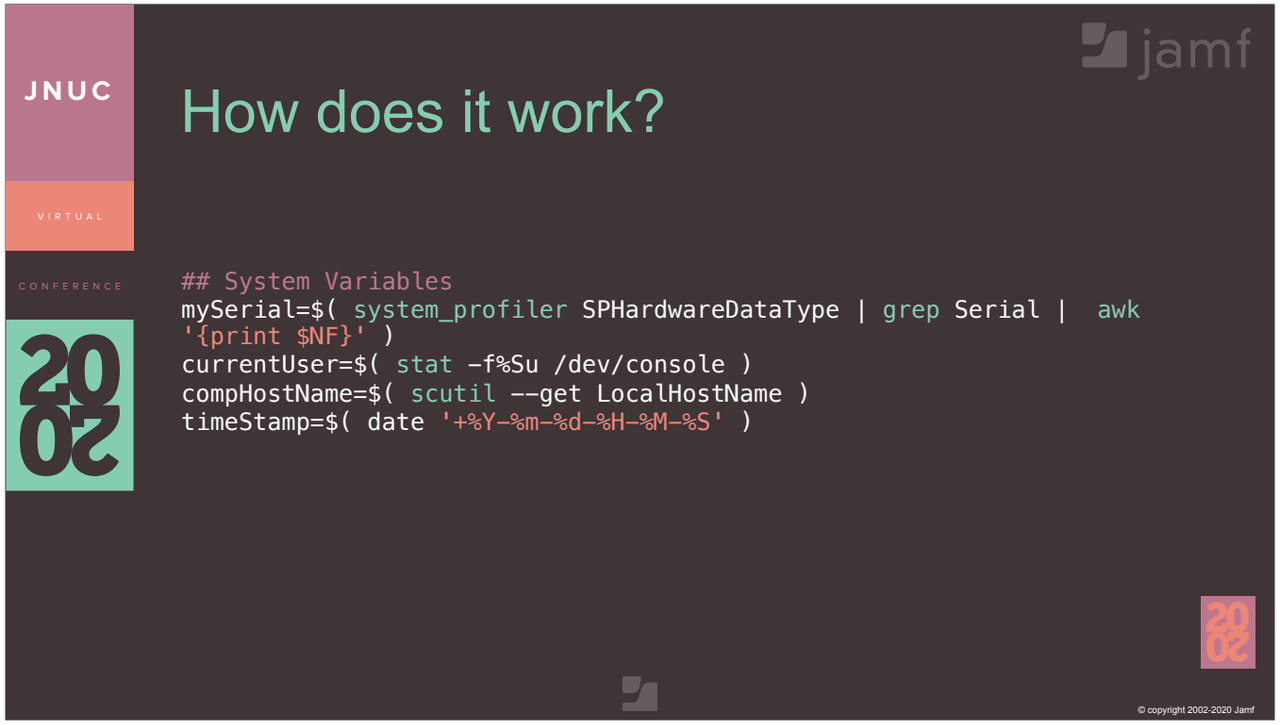
2020

# How does it work?

```
#!/bin/bash

## User Variables
jamfProURL="$4"
jamfProUser="$5"
jamfProPass="$6"
logFiles="$7"

## System Variables
mySerial=$( system_profiler SPHardwareDataType | grep Serial |  awk '{print $NF}' )
currentUser=$( stat -f%Su /dev/console )
compHostName=$( scutil --get LocalHostName )
timeStamp=$( date '+%Y-%m-%d-%H-%M-%S' )

## Log Collection
fileName=$compHostName-$currentUser-$timeStamp.zip
zip /private/tmp/$fileName $logFiles

## Upload Log File
jamfProID=$( curl -k -u $jamfProUser:$jamfProPass $jamfProURL/JSSResource/computers/serialnumber/$mySerial/subset/general
| xpath "//computer/general/id/text()" )
curl -k -u $jamfProUser:$jamfProPass $jamfProURL/JSSResource/fileuploads/computers/id/$jamfProID -F name=@/private/tmp/
$fileName -X POST

## Cleanup
rm /private/tmp/$fileName
exit 0
```



Let’s take a look at the next block…




…system variables.

Here, we are just setting up a few requirements for uploading the archive to Jamf Pro.

First, we need to determine what the serial number of the Mac we are running this on is. We'll use this to query the Classic API to determine the Jamf Pro Computer ID number.

Once we have that, we'll determine who the logged in user is; the current computer host name and the date and time we're running this script. These variables, we'll be used to create the filename of the archive that we'll upload to Jamf Pro.

JNUC

VIRTUAL

CONFERENCE

2020

# How does it work?

```
#!/bin/bash

## User Variables
jamfProURL="$4"
jamfProUser="$5"
jamfProPass="$6"
logFiles="$7"

## System Variables
mySerial=$( system_profiler SPHardwareDataType | grep Serial |  awk '{print $NF}' )
currentUser=$( stat -f%Su /dev/console )
compHostName=$( scutil --get LocalHostName )
timeStamp=$( date '+%Y-%m-%d-%H-%M-%S' )

## Log Collection
fileName=$compHostName-$currentUser-$timeStamp.zip
zip /private/tmp/$fileName $logFiles

## Upload Log File
jamfProID=$( curl -k -u $jamfProUser:$jamfProPass $jamfProURL/JSSResource/computers/serialnumber/$mySerial/subset/general
| xpath "//computer/general/id/text()" )
curl -k -u $jamfProUser:$jamfProPass $jamfProURL/JSSResource/fileuploads/computers/id/$jamfProID -F name=@/private/tmp/
$fileName -X POST

## Cleanup
rm /private/tmp/$fileName
exit 0
```

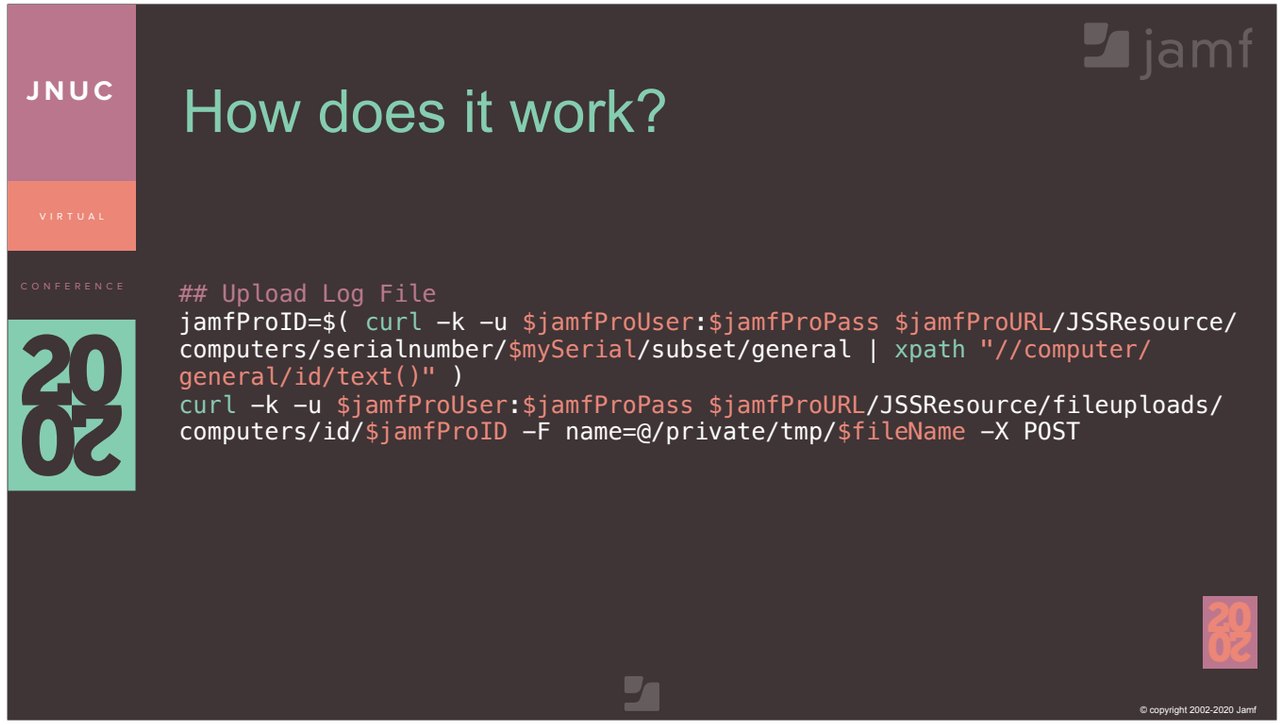


Alright, lets take a look at the collection of the the log files.


Here, you can see we are specifying the variables from the previous block to create the archive filename which is computer host name dash current user dash the current date and time dot zip.

We'll then use that along with the *zip* binary to create an archived zip file in the */private/tmp/* directory with all the log files we specified inside the Policy.


Alright, great! Now that we have the log files compressed in an archive, let’s look at how we upload them.




First thing's first, we need to use that **mySerial** variable from earlier to query the Classic API and determine what the Jamf Pro Computer ID is.

Once we have that, we can go ahead and upload the archive as an attachment to that device record in Jamf Pro.

JNUC

VIRTUAL

CONFERENCE

2020

# How does it work?

```
#!/bin/bash

## User Variables
jamfProURL="$4"
jamfProUser="$5"
jamfProPass="$6"
logFiles="$7"

## System Variables
mySerial=$( system_profiler SPHardwareDataType | grep Serial |  awk '{print $NF}' )
currentUser=$( stat -f%Su /dev/console )
compHostName=$( scutil --get LocalHostName )
timeStamp=$( date '+%Y-%m-%d-%H-%M-%S' )

## Log Collection
fileName=$compHostName-$currentUser-$timeStamp.zip
zip /private/tmp/$fileName $logFiles

## Upload Log File
jamfProID=$( curl -k -u $jamfProUser:$jamfProPass $jamfProURL/JSSResource/computers/serialnumber/$mySerial/subset/general
| xpath "//computer/general/id/text()" )
curl -k -u $jamfProUser:$jamfProPass $jamfProURL/JSSResource/fileuploads/computers/id/$jamfProID -F name=@/private/tmp/
$fileName -X POST

## Cleanup
rm /private/tmp/$fileName
exit 0
```



And for that last block…


…we'll just clean things up by deleting the local copy of the archive and setting a clean exit.

It's alway best to be good stewards.

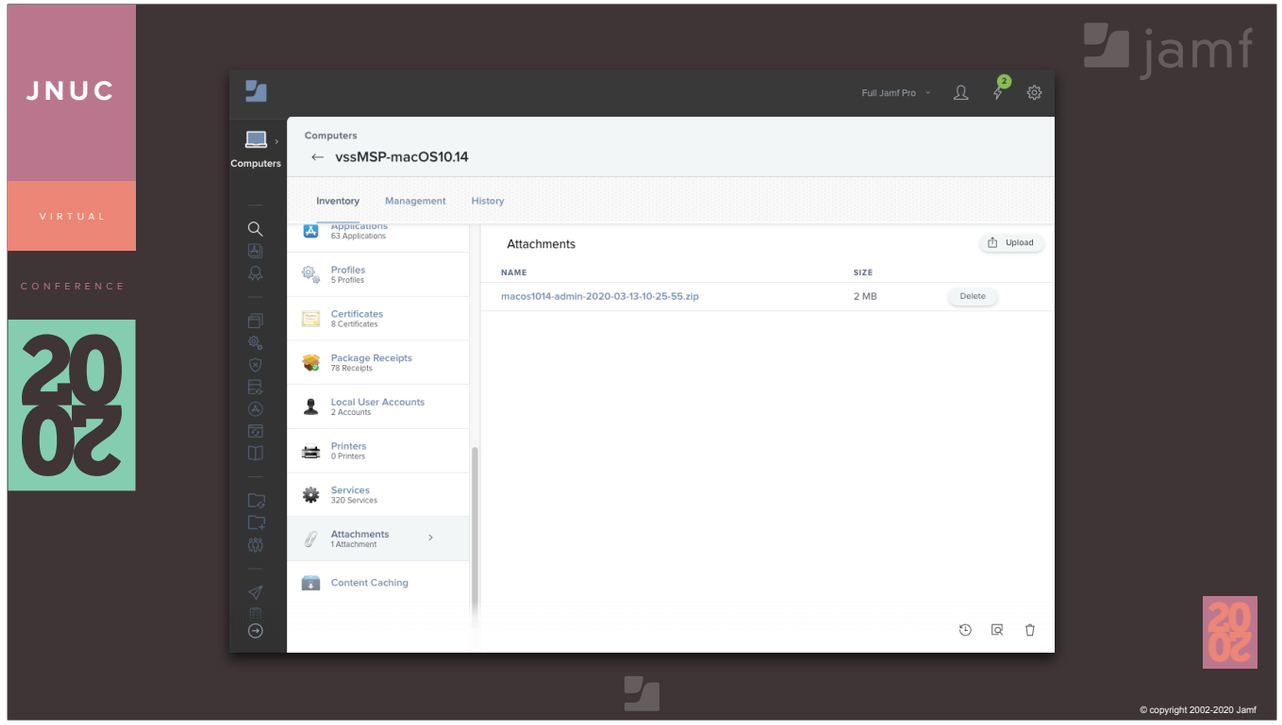

Really, that's it!

A very simple workflow, to accomplish a task we all have to deal with. Sure, we could add some additional error handling, but the end result is the same. If any part fails, it all fails and doesn't create any extra work or issues.

However, it is open source under the MIT license, if you do find something, please submit an **Issue** or a **Pull Request** on Github and I'll be happy to take a look :)

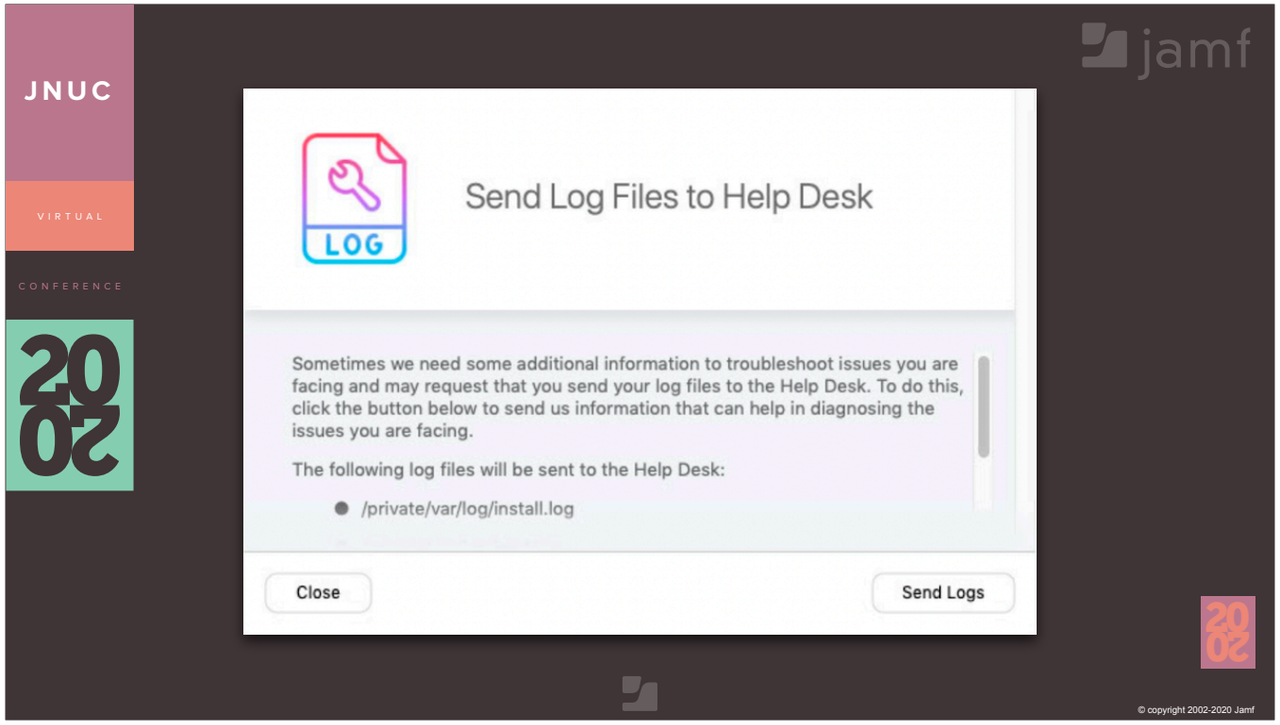
…

Alright, so we've uploaded the log files, lets take a look inside Jamf Pro, and see how that looks!

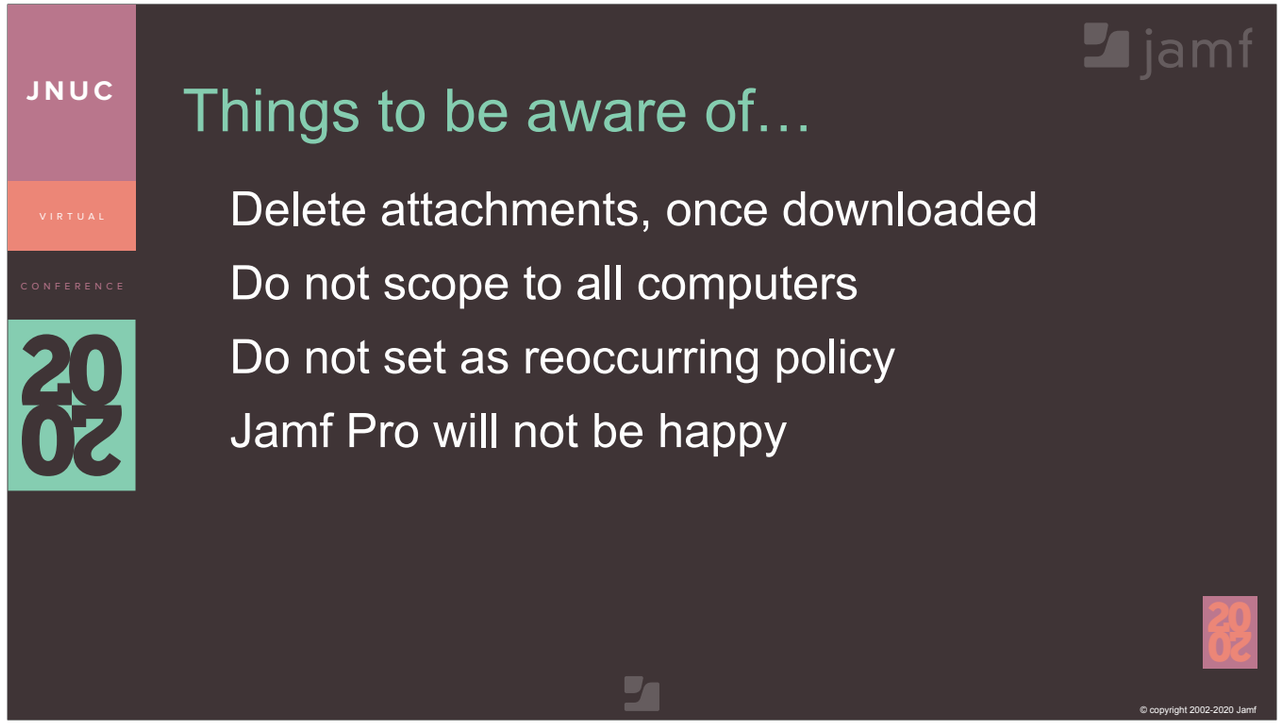



Here you can see we are viewing a computer record for a VM I have in my test Jamf Pro instance. Under attachments, we have our compressed archive here containing the log files I specified in my policy.

In this case, you can see the name of the archive is the host name, dash admin, dash date and time from when it was created.




As I mentioned earlier, you can also make this available via Self Service in a **Help Desk** or **First Aid** type category and direct your users there as needed.




Okay, so you have this workflow all setup and it seems to be working great, but as always…whats the fine print?

First, the Jamf Pro database is not designed to house a large number of files. That is why we have both File Share and Cloud Distribution Points available in Jamf Pro.

So…
- [CLICK] Be sure to delete the attachment, once you've downloaded it
- [CLICK] Do not scope this policy to **All Computers**, unless you are creating as a Self Service policy
- [CLICK] Do not set the policy to reoccurring check-in

[CLICK] In the end, we want to be good stewards to our database. Keep it neat and tidy. If you don't…


…you could end up with an upset Jamf Pro environment. And then, you'll end up having to call support.

On top of that [CLICK]…this could lead to some very upset staff.

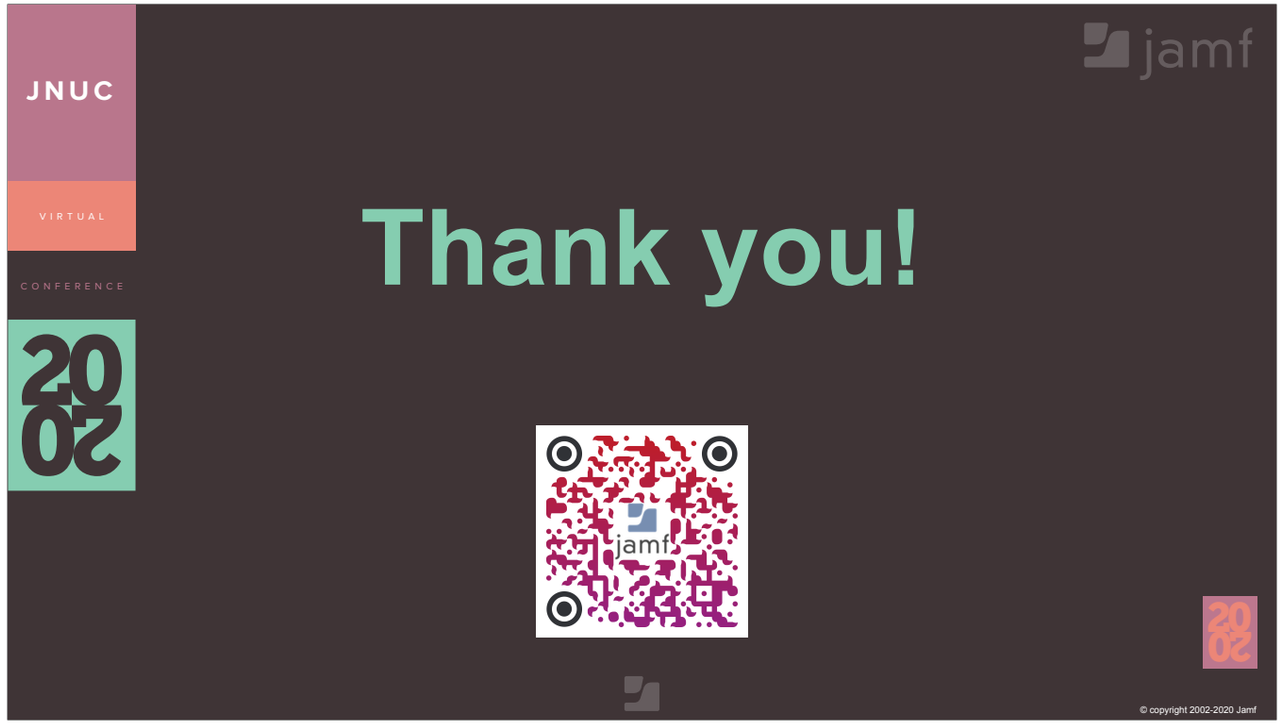
So, to help with this, I've also put together an Extension Attribute to be able to run reports and create Smart Groups based on the number of attachments on a device record, which you can also find on the Github repo.

And with that…


…you can find the entire workflow and all the instructions for getting logCollection setup on Github using the QR code on the screen as well as a copy of this presentation.

I'll be happy to answer any questions now in the chat, otherwise feel free to open up an issue on Github.

[Click to next slide]


JNUC
VIRTUAL
CONFERENCE
2020
jamf
Thank you!
jamf
2020
© copyright 2002-2020 Jamf

Thank you!

# Thank you for listening!

Give us feedback by completing the survey using the widget below

JNUC2020
VIRTUAL CONFERENCE