

UP NEXT

## erase-install: A Tool for Upgrading and Reinstalling macOS

GRAHAM PUGH



# Graham Pugh

Senior Client Engineer - Apple Services  
ETH Zürich



@GrahamRPugh



@GrahamRPugh



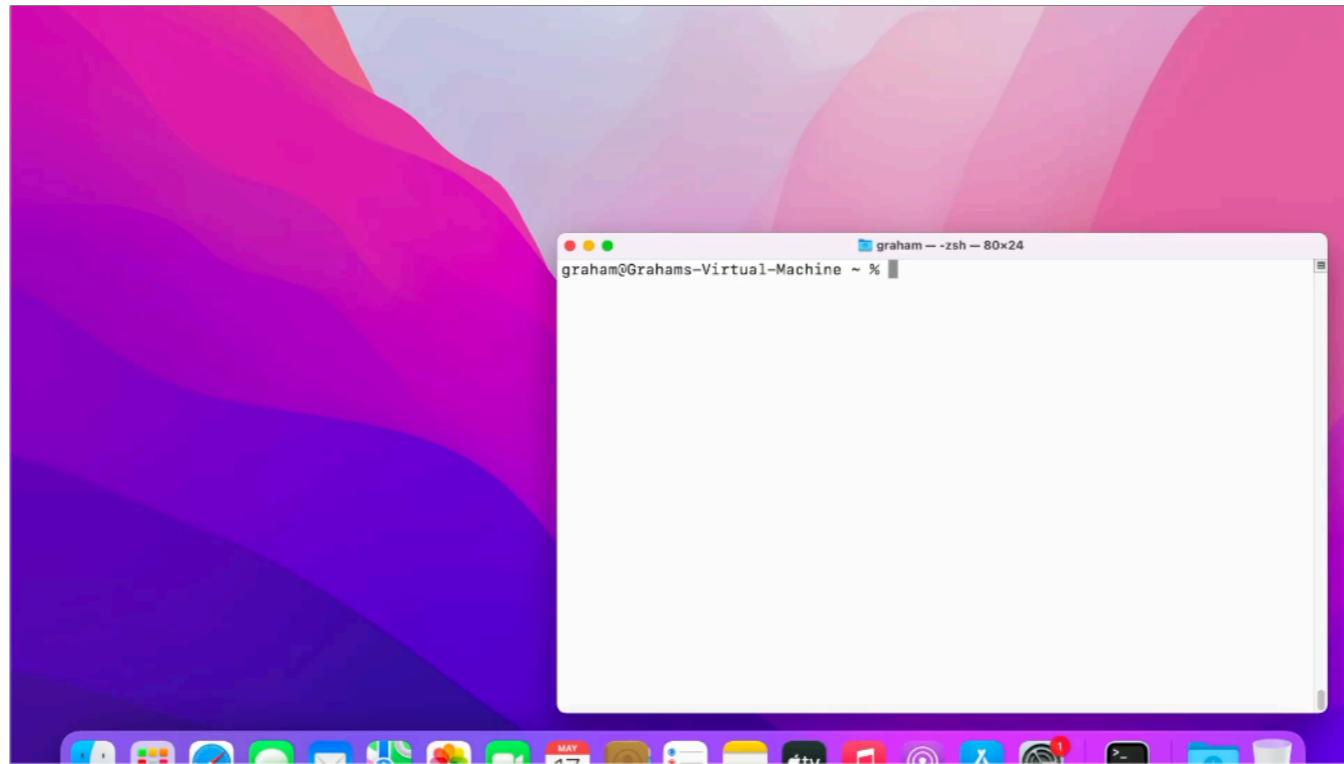
@grahampugh



grahamrpugh.com

My name is Graham, I'm from Lancashire originally, but I've spent the past 6 years in Switzerland working in the central IT Services of the Swiss Federal Technical University in Zürich.

I'm going to talk today about a project I created, called erase-install.

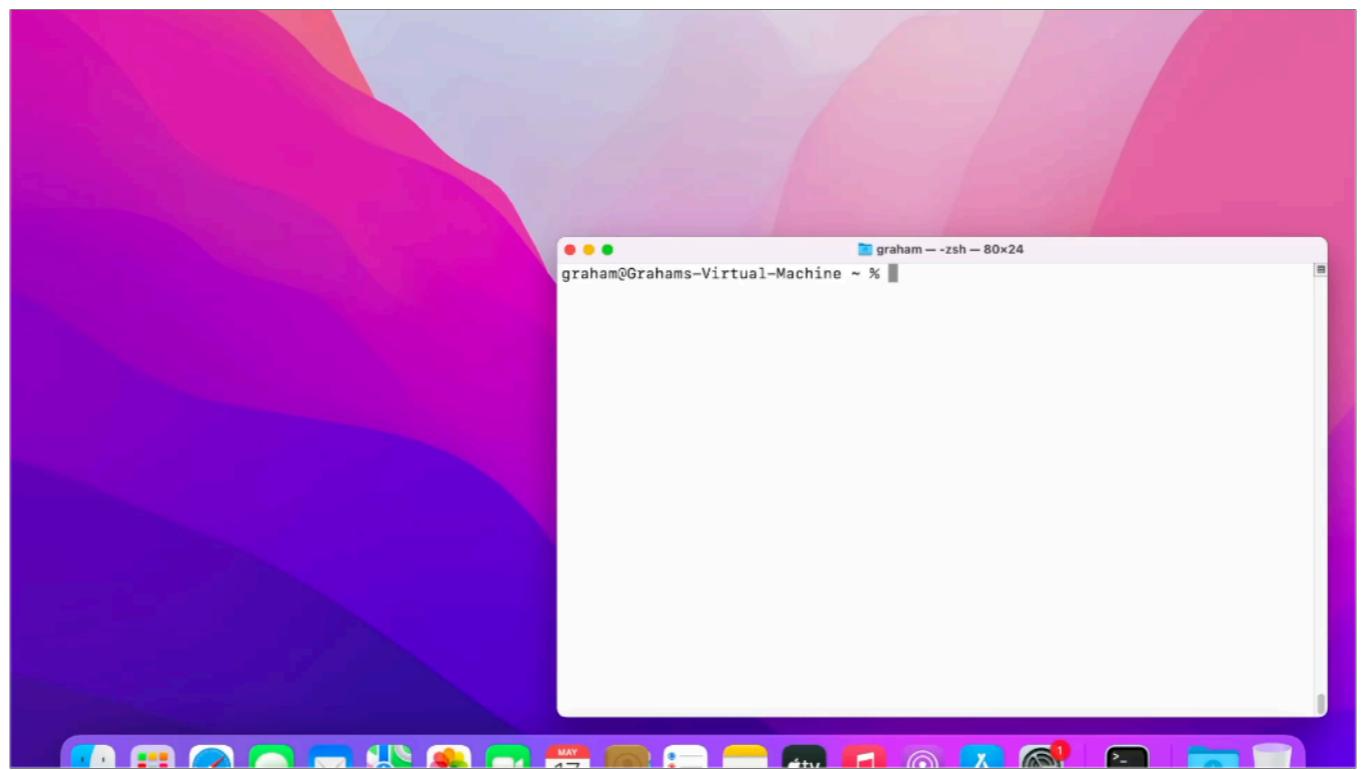


Erase-install is a tool for erasing or reinstalling macOS with a single command or click. It brings together other open-source tools to perform a workflow which will download a macOS installer and use it to perform a silent reinstallation. The only user interaction required is on Apple Silicon Macs, where we need the user's password. That's not required on Intel Macs.

In this screen recording, you're seeing the terminal output behind the user dialog because that's how I invoked the script, but if you're running this from a management tool, your users will only see the dialogs.

The script is agnostic to which management tool you're using, your tool just needs to be able to run a script or install a package to use it. I'm aware of it being used from Jamf Pro, SimpleMDM, Jamf Now, WorkspaceONE, InTune, Munki and Mosyle.

We'll look again at the video later when I've given more context.



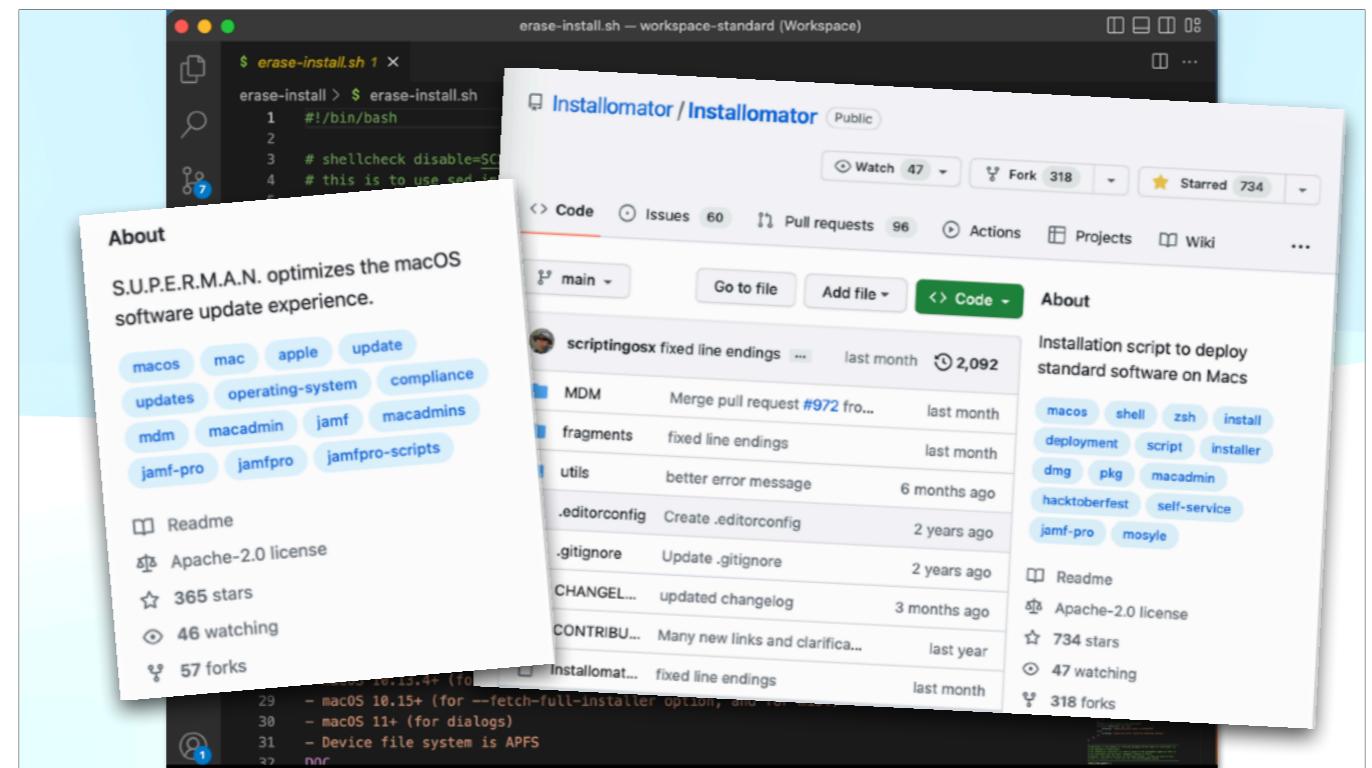


Behind the scenes, erase-install is a big, long bash script. And yes, big long bash scripts are not considered best practice... But for many of us ordinary Mac Admins who aren't software engineers, it's just natural for us to start to approach solving a problem by scripting in bash, and then before we know it, it's 3000, 🍎 5000, 7000 lines long...

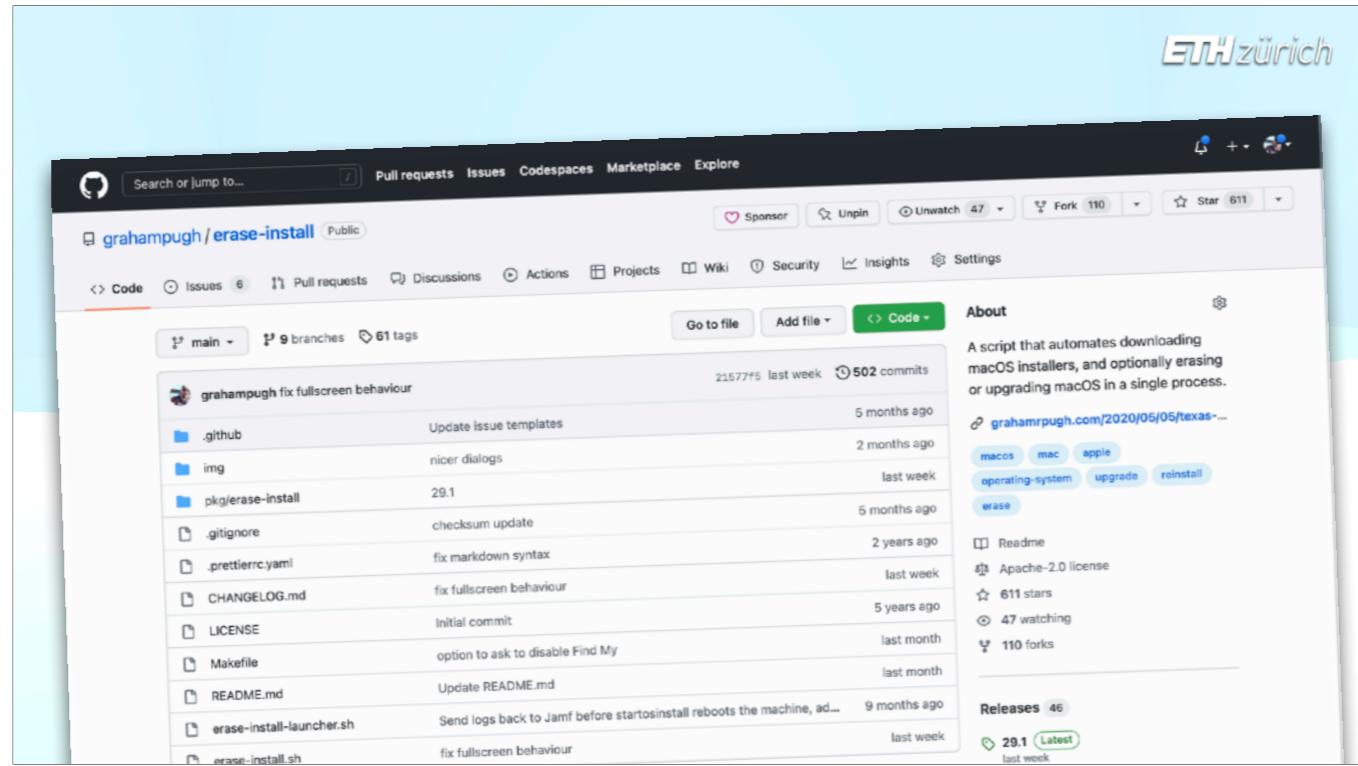
And just like Kevin White's SUPERMAN and Armin Briegel and friends' Installomator...

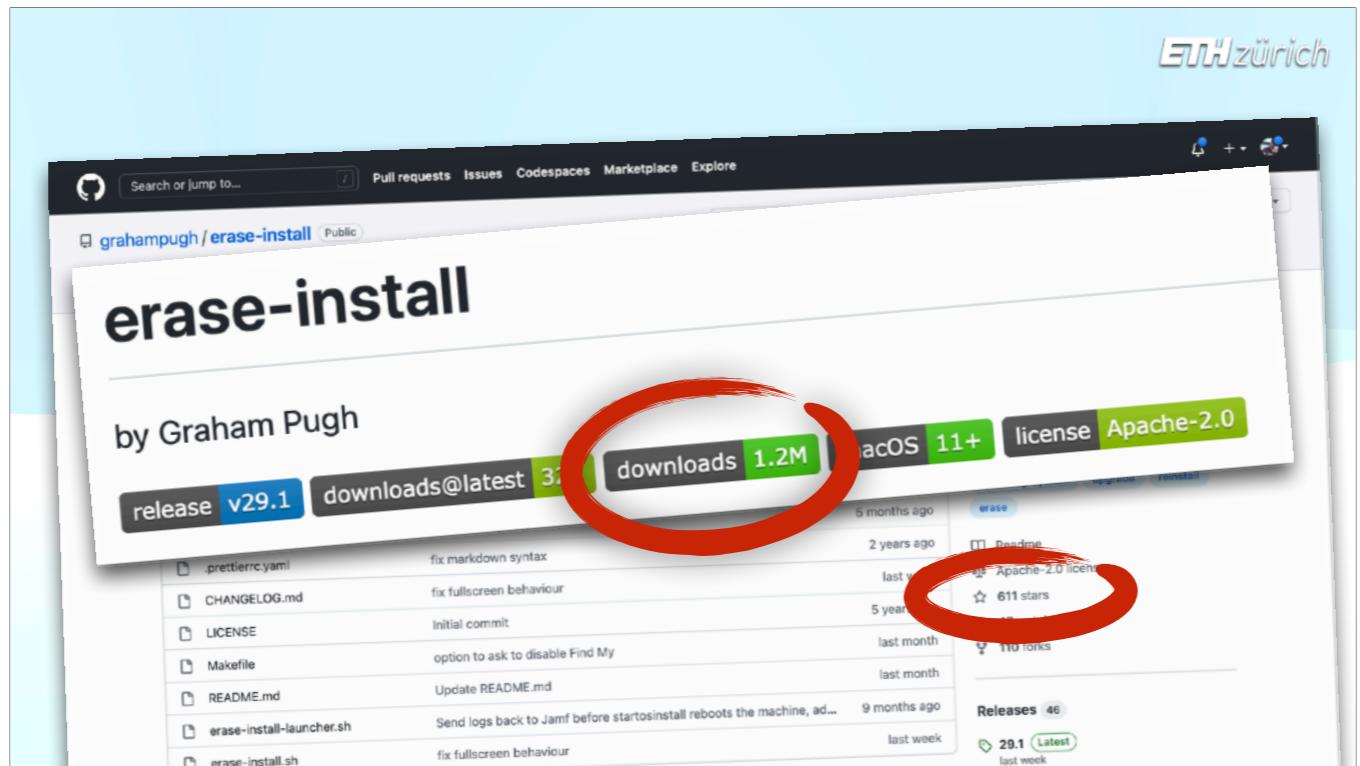
The screenshot shows a terminal window titled "erase-install.sh — workspace-standard (Workspace)". The window contains the following text:

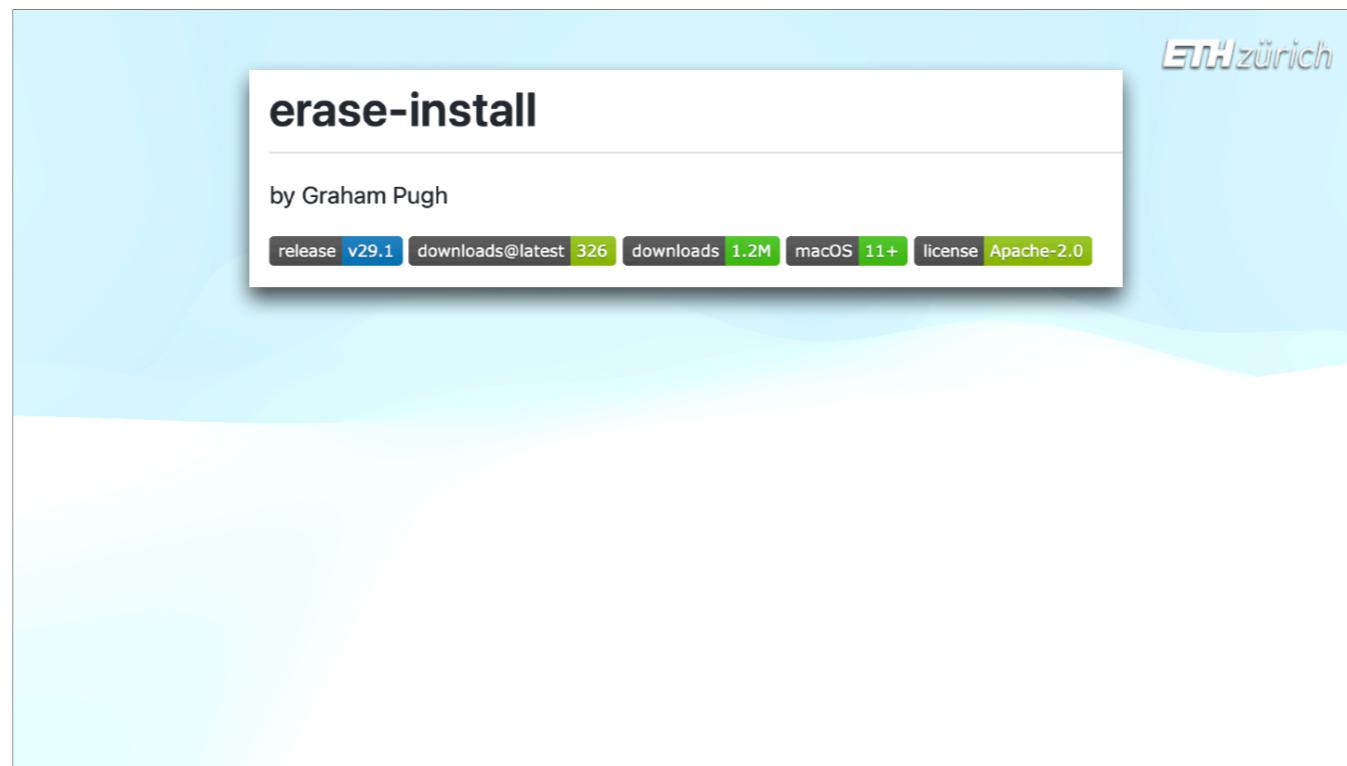
```
$ erase-install.sh 1 ×
erase-install > $ erase-install.sh
1  #!/bin/bash
2
3  # shellcheck disable=SC2001
4  # this is to use sed in the case statements
5  # shellcheck disable=SC2034
6  # this is due to the dynamic variable assignments used in the localization strings
7
8  :<<DOC
9  =====
10 erase-install.sh
11 =====
12 by Graham Pugh
13
14 WARNING. This is a self-destruct script. Do not try it out on your own device!
15
16 See README.md and the GitHub repo's Wiki for details on use.
17
18 It is recommended to use the package installer of this script. It contains
19 swiftDialog and mist, which are required for most of the use-cases of this script.
20
21 This script can, however, also be run standalone.
22 It will download and install swiftDialog if needed and not found.
23 It will also download mist if it is not found.
24 Suppress the downloads with the --no-curl option.
25
26 Requirements:
27 - macOS 12.4+
28 - macOS 10.13.4+ (for --erase option)
29 - macOS 10.15+ (for --fetch-full-installer option, and for mist)
30 - macOS 11+ (for dialogs)
31 - Device file system is APFS
32
33
```



... erase-install has got a lot of use and attention over the past couple of years. 🍎 – much more than I could have anticipated. That's because, regardless of best practices, these are the tools that have been shared publicly and have proven themselves able to solve problems being faced by large numbers of us Mac admins.







So, in this talk I want to achieve three things:

- 🍎 I'll give you all a short synopsis of the erase-install project,
- 🍎 I'll look forward to the time when we don't need it anymore, and,
- 🍎 as a representative of ordinary Mac admins, I'll share what experience I have gained from maintaining an open source project that has built up a relatively large user base.

## erase-install

by Graham Pugh

release v29.1 | downloads@latest 326 | downloads 1.2M | macOS 11+ | license Apache-2.0

- Why erase-install?

## erase-install

by Graham Pugh

release v29.1 | downloads@latest 326 | downloads 1.2M | macOS 11+ | license Apache-2.0

- Why erase-install?
- But, seriously, why *still* erase-install?

## erase-install

by Graham Pugh

release v29.1 | downloads@latest 326 | downloads 1.2M | macOS 11+ | license Apache-2.0

- Why erase-install?
- But, seriously, why *still* erase-install?
- Why open source?

# Once upon a time...

To talk about how this project came about, let's go back to the beginning...

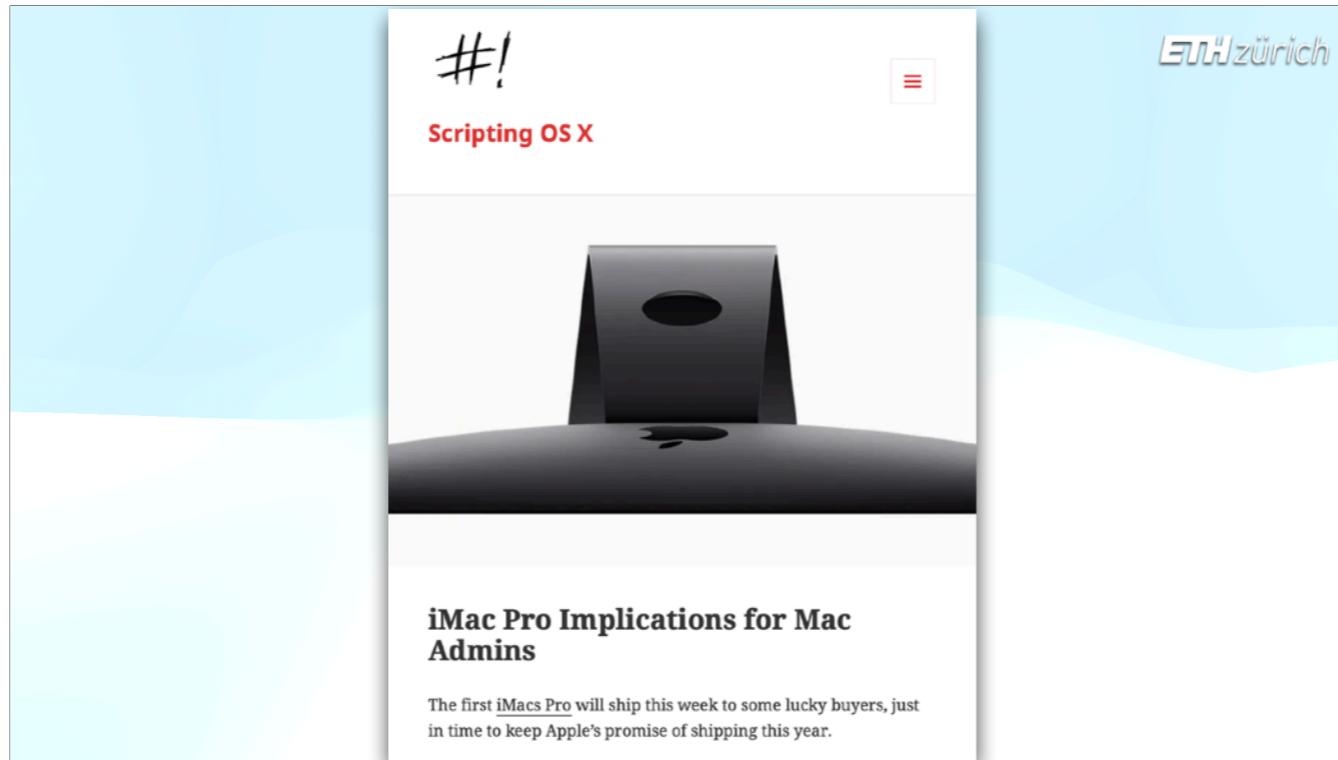
🍎 iOS has had an "Erase All Contents And Settings" option in the System Settings for years. Back in the early 2010s, we could only dream about having this feature on OS 10, 🍏 but it was ok, because we had good redeployment workflows using NetBoot, DeployStudio and imaging.

# Once upon a time...



Once upon a time...





But this started coming undone in 2017 when Apple introduced T2 Macs, starting with the iMac Pro. 🍎 NetBoot was killed, and imaging started to become unstable.

The slide has a light blue header bar with the text '#!' and 'Scripting OS X' in red, and the ETH Zürich logo in the top right.

## NetBoot

Prohibiting External Boot will (probably) also prohibit NetBoot and NetInstall. However, Apple updated their support article "Create a NetBoot, NetInstall, or NetRestore image" with the note:

*iMac Pro computers don't support starting up from network volumes.*

### iMac Pro Implications for Mac Admins

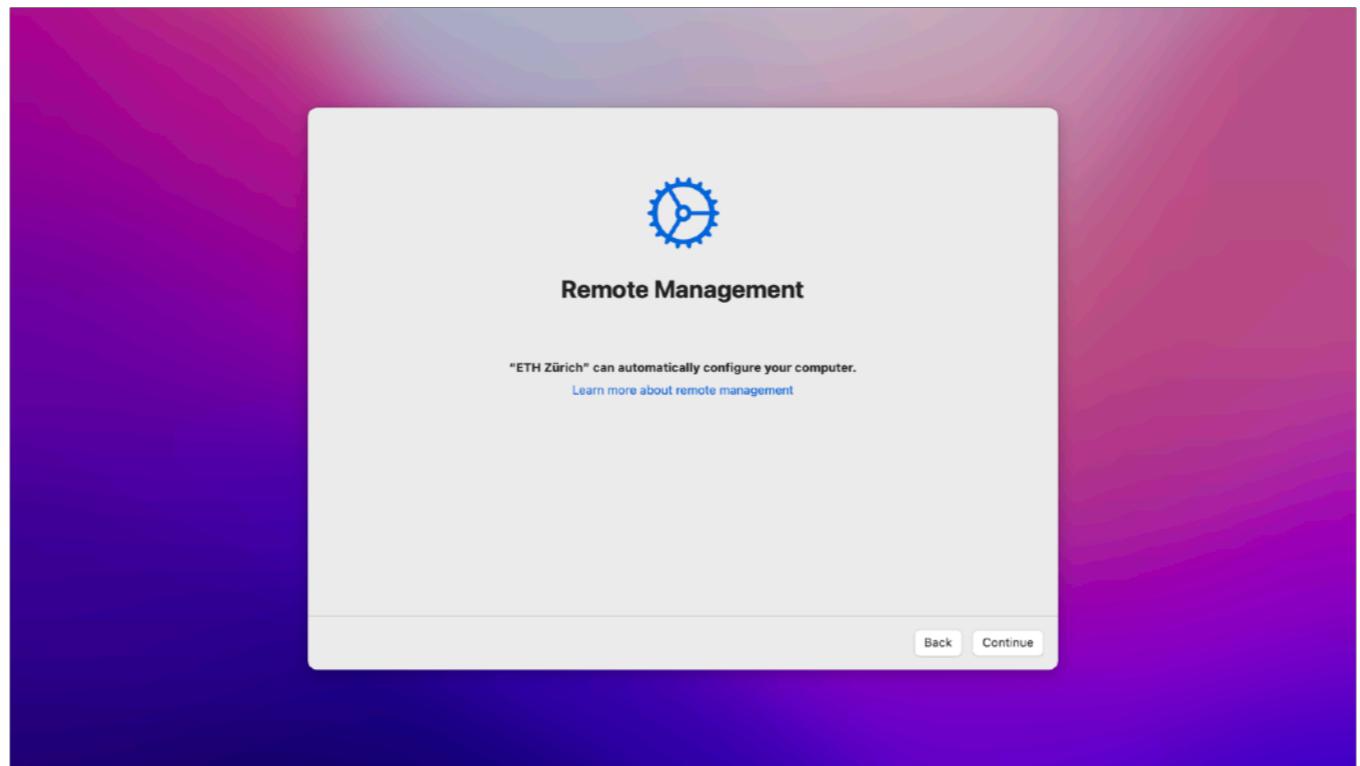
The first iMacs Pro will ship this week to some lucky buyers, just in time to keep Apple's promise of shipping this year.

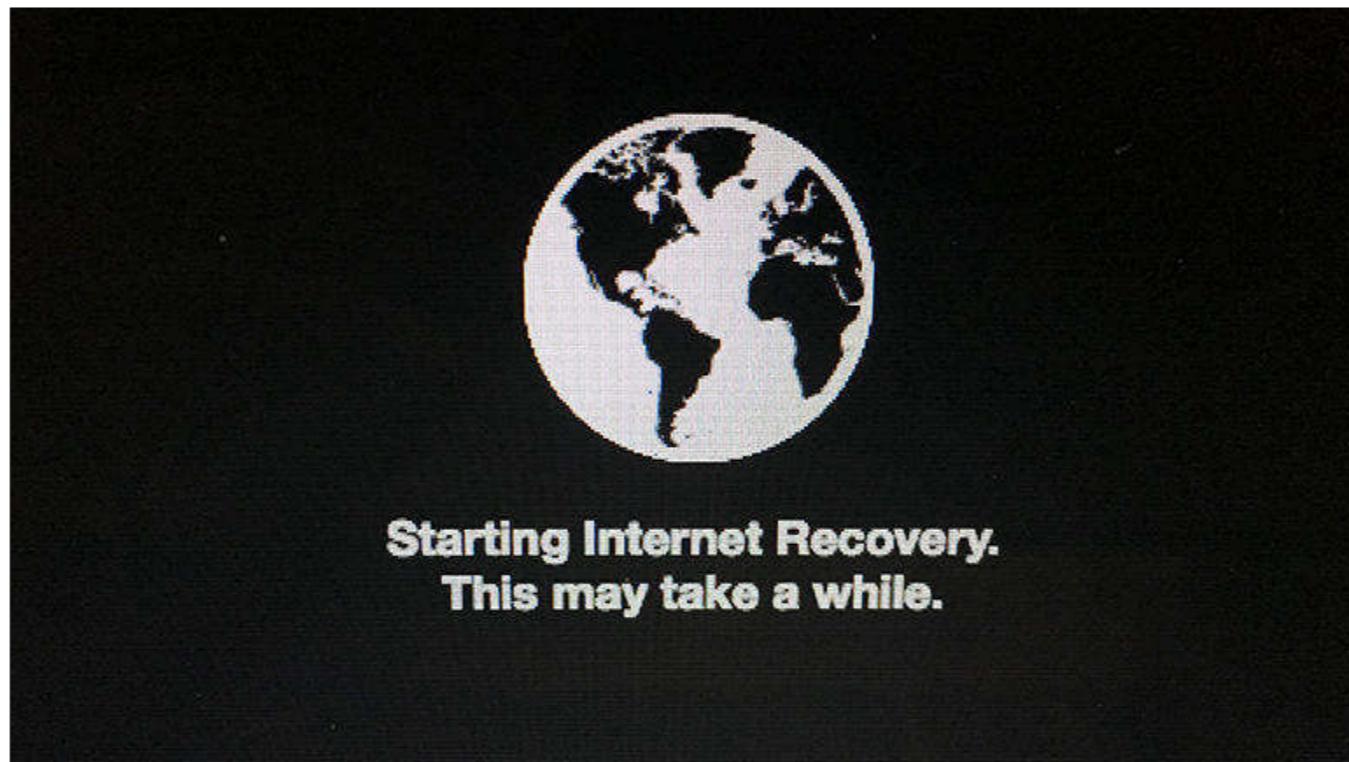


### Device Enrollment Program

Streamline the on boarding of institutionally owned devices. Enroll devices in MDM during activation and skip basic setup steps to get users up and running quickly.

The Device Enrollment Program became our new setting up workflow. But for redeployments, we still had to get a clean OS on the system. Back then, 🍎 getting that magical, elusive Remote Management screen was extremely hit-and-miss. Reinstalling from the Recovery partition would not clear the old DEP record, so enrolment was not attempted afterwards.



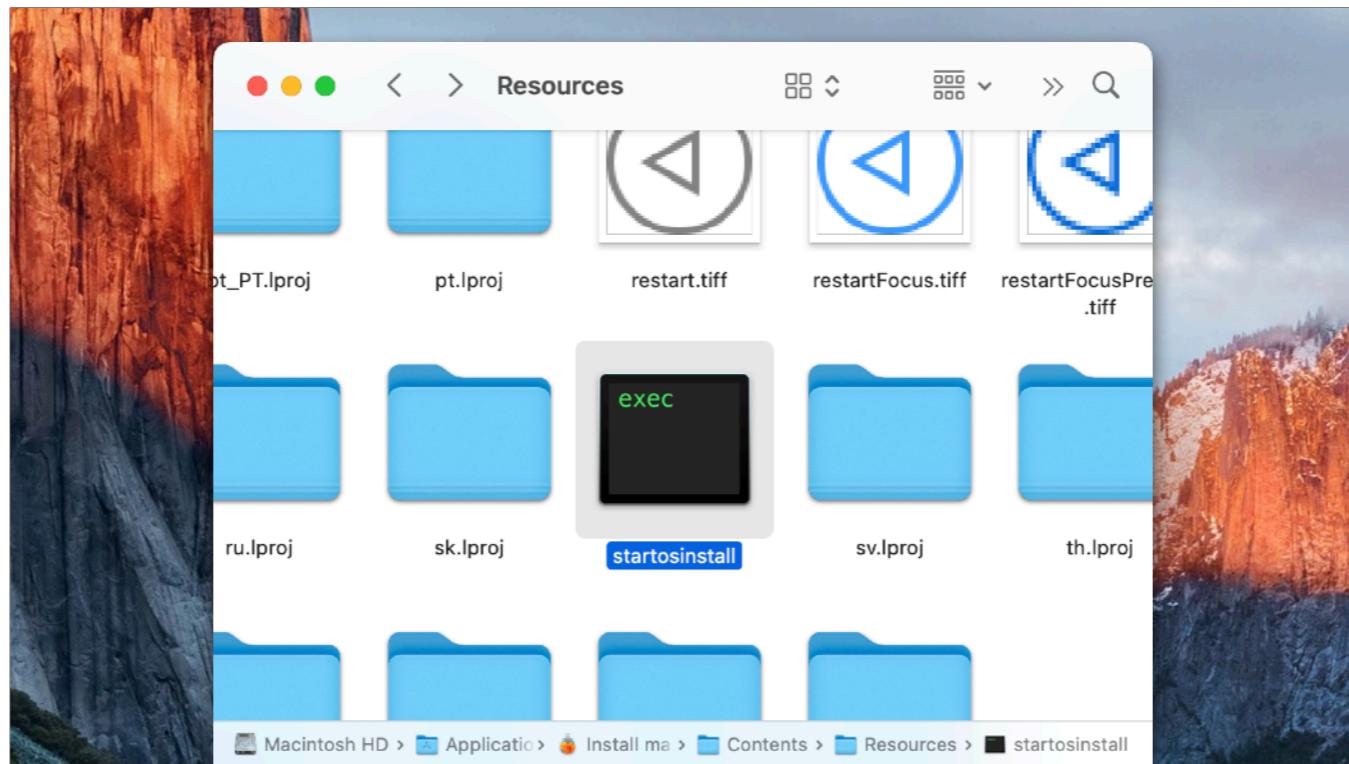


Internet Recovery was the best solution for us, but that meant even more clicks, even slower, and no way to automate it.

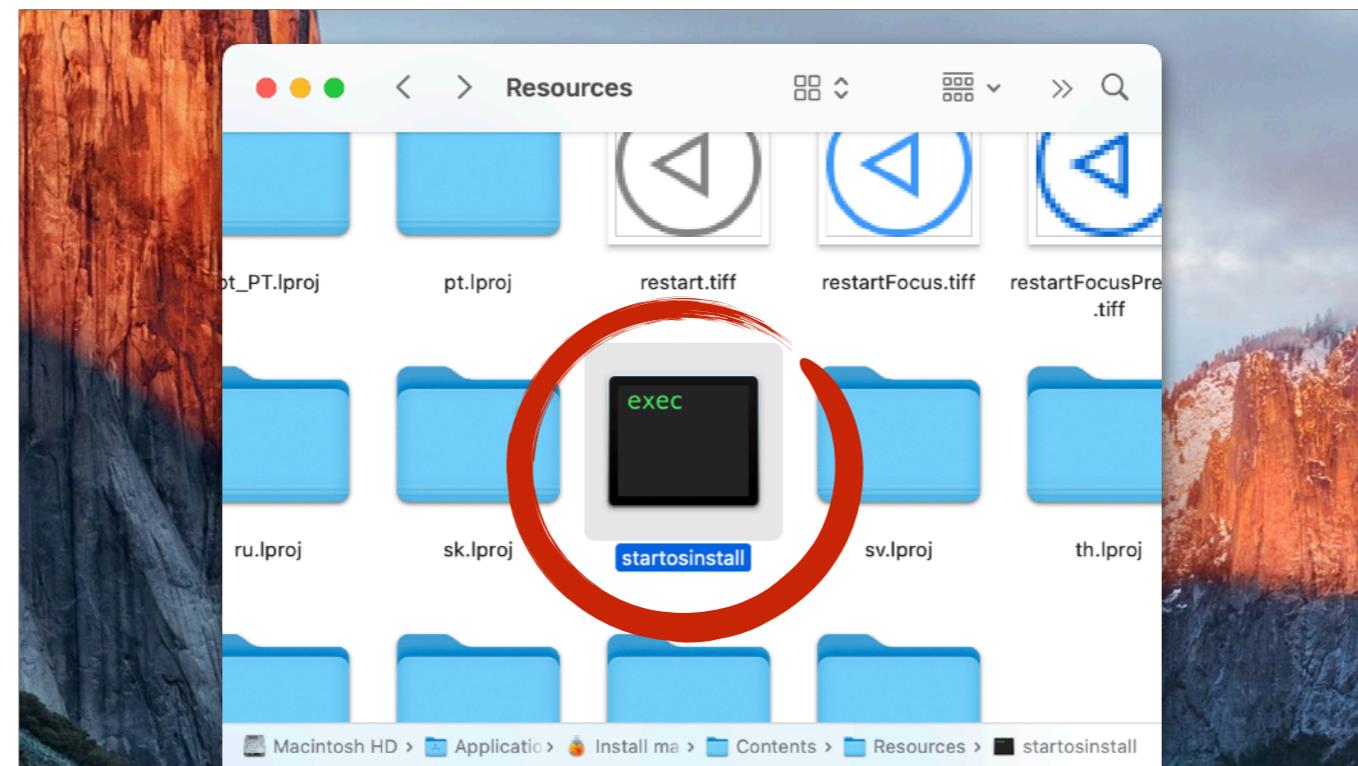


However, when El Capitan was released in 2015, we had already got the first glimpse of what would become a completely new option for us to install OS 10.

The El Capitan installer was distributed as an app...



...And inside the app bundle, in the Resources folder, Apple added a command line tool called **startosinstall**. Every version of macOS since then has included this tool.





At this point, `startosinstall` wasn't massively useful. It couldn't be run completely silently as you needed to agree to the license.

```
sudo '/Applications/Install OS X El Capitan.app/  
Contents/Resources/startosinstall'  
  
--volume /Volumes/Untitled  
--applicationpath "/Applications/Install OS X El  
Capitan.app"  
--license  
--usage
```





But with macOS Sierra, the addition of the **agreetolicense** option, and the undocumented '**nointeraction**' option, meant we could run the command unattended from a script. 🍎 By the way, that latter one isn't needed anymore, but back then it was required.

```
sudo '/Applications/Install macOS Sierra.app/  
Contents/Resources/startosinstall'
```

```
--agreetolicense  
--nointeraction
```



```
sudo '/Applications/Install macOS Sierra.app/  
Contents/Resources/startosinstall'
```

--agreetolicense

~~--nointeraction~~



## Install macOS

You can use these supported methods to erase a volume and install macOS:

- In spring 2018, an update to the macOS Installer will let you erase your boot volume and install macOS to it using `startosinstall` and the new `--eraseinstall` flag. If you specify additional packages with the `--installpackage` flag, they'll be installed after installing macOS. Use the `productbuild(1)` command to build these packages.
- Startup from a bootable installer or macOS Recovery, and use Disk Utility to erase the target volume before installing.
- Connect the Mac to be erased in **target disk mode**, and use Disk Utility or the `asr(8)` command to restore a pre-configured software image. This is also known as *monolithic system imaging*.

`Startosinstall` was still not relevant for redeployments, until February 2018, when Apple unusually published a support article  which announced a new upcoming option for the `startosinstall` tool, called `eraseinstall`.

## Install macOS

You can use these supported methods to erase a volume and install macOS:

In spring 2018, an update to the macOS Installer will let you erase your boot volume and install macOS to it using `startosinstall` and the new `--eraseinstall` flag. If you specify additional packages with the `--installpackage` flag, they'll be installed after installing macOS. Use the `productbuild(1)` command to build these packages.

- Startup from a bootable installer or macOS Recovery, and use Disk Utility to erase the target volume before installing.
- Connect the Mac to be erased in **target disk mode**, and use Disk Utility or the `asr(8)` command to restore a pre-configured software image. This is also known as *monolithic system imaging*.



This new option allowed us to completely wipe the system drive and install a fresh macOS system in a single command. 🍎 And it could also be done silently with the no interaction flag.

```
sudo '/Applications/Install macOS High  
Sierra.app/Contents/Resources/startosinstall'
```

```
--eraseinstall  
--newvolumename "Macintosh HD"
```



```
sudo '/Applications/Install macOS High  
Sierra.app/Contents/Resources/startosinstall'  
  
--eraseinstall  
--newvolumename "Macintosh HD"  
--agreetolicense  
--nointeraction
```





I just want to emphasise that the `startosinstall` binary is only present in the installer app, not on the system itself, and it cannot be run from the recovery partition. So to use `startosinstall` to reinstall the system, first of all you have to download the installer, and  only then can you run `startosinstall`.

During the beta phase for that spring release of macOS High Sierra, 10.13.4, I started trying to write a solution that would use this new feature to provide a one-click, start-to-finish Erase-All-Contents-And-Settings button in our Jamf Pro Self Service.

ETHzürich



The screenshot shows a blog post on the 'Der Flounder' website. The post title is 'Using installinstallmacos.py to download macOS High Sierra installers'. It was published on February 27, 2018, by rtrouton. The post discusses how Apple moved macOS Installer applications from the App Store to Software Update catalogs, making it possible to download them via command-line. It mentions a Python script called `installinstallmacos.py` written by Greg Neagle. The script performs several tasks: parsing a Software Update feed, identifying macOS installers, displaying choices, creating a disk image, mounting it, and downloading packages. The sidebar includes an RSS feed link, a calendar for February 2018 showing the 27th as the 21st, and a 'Recent Comments' section with entries from Joss, JGuy, and Marc.

Der Flounder  
Seldom updated, occasionally insightful.

About Contact

Type text to search here...

Home > Mac administration, macOS, Scripting > Using installinstallmacos.py to download macOS High Sierra installers

## Using installinstallmacos.py to download macOS High Sierra installers

February 27, 2018 rtrouton Go to comments Leave a comment

Starting with macOS Sierra, Apple moved the macOS Installer applications from being exclusively an App Store download to now being included in the regular Software Update catalogs. This means that it's possible to download macOS installers, including those for macOS betas or hardware-specific macOS builds, using the command-line softwareupdate tool.

To assist with this task, [Greg Neagle](#) has written a Python script named `installinstallmacos.py`. `installinstallmacos.py` is designed to do the following:

1. Parse a specified Software Update feed.
2. Identify the listed products which appear to be macOS installers.
3. Display a menu of the available choices.

Once you've selected from the available options, the script does the following:

4. Creates a disk image and names it with the appropriate information for the specified macOS installer.
5. Mounts the disk image.
6. Downloads all the relevant packages from the Software Update feed for the specified macOS installer.

RSS feed

February 2018

M	T	W	T	F	S	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28				

[« Jan](#) [Mar »](#)

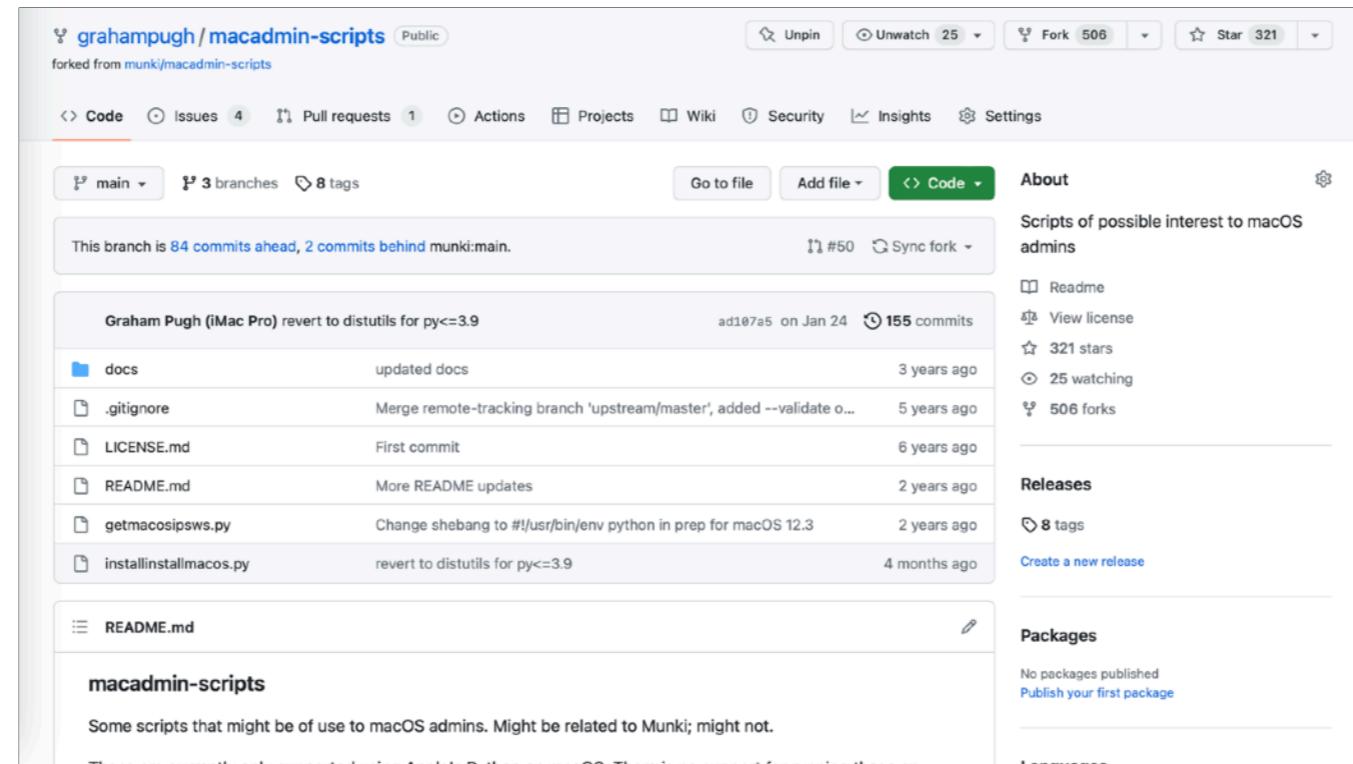
Recent Comments

Joss on [macOS Ventura 13.3 alters expense reports](#)...

JGuy on [Identifying Mac laptops and desktops](#)...

Marc on [macOS Ventura 13.3 alters expense reports](#)...

Two months earlier, Greg Neagle had published a script for downloading macOS installers from Apple's software catalogs, which he called `installinstallmacos.py`. Coincidentally, Rich Trouton blogged about the script at the same time that the first 10.13.4 beta was released.



I adapted this script so that it could automatically give me the latest installer that was compatible with the system that I ran it on, so that I could run it unattended.

Once I got this working, I wrote a quick wrapper script in bash which would grab my version of the installinstallmacos script, run it to grab the latest macOS installer, and then call startosinstall with the eraseinstall option to wipe and reinstall the device.

[grahampugh / macadmin-scripts](#) Public  
forked from [munki/macadmin-scripts](#)

Code Issues 4 Pull requests 1 Actions Projects Wiki Security Insights Settings

main 3 branches 8 tags

This branch is 84 commits ahead of munki:main. #50 Sync fork

Graham Pugh's profile picture (a circular image of a mountain landscape) is highlighted with a large gray arrow pointing to it.

Commits:

- Update for py<=3.9 ad107a5 on Jan 24 155 commits
- Updated docs 3 years ago
- Added remote-tracking branch 'upstream/master', added --validate o... 5 years ago
- mit 6 years ago
- ADME updates 2 years ago
- Change shebang to #!/usr/bin/env python in prep for macOS 12.3 2 years ago
- installins 4 months ago

README.md

**macadmin-scripts**

Some scripts that might be of use to macOS admins. Might be related to Munki; might not.

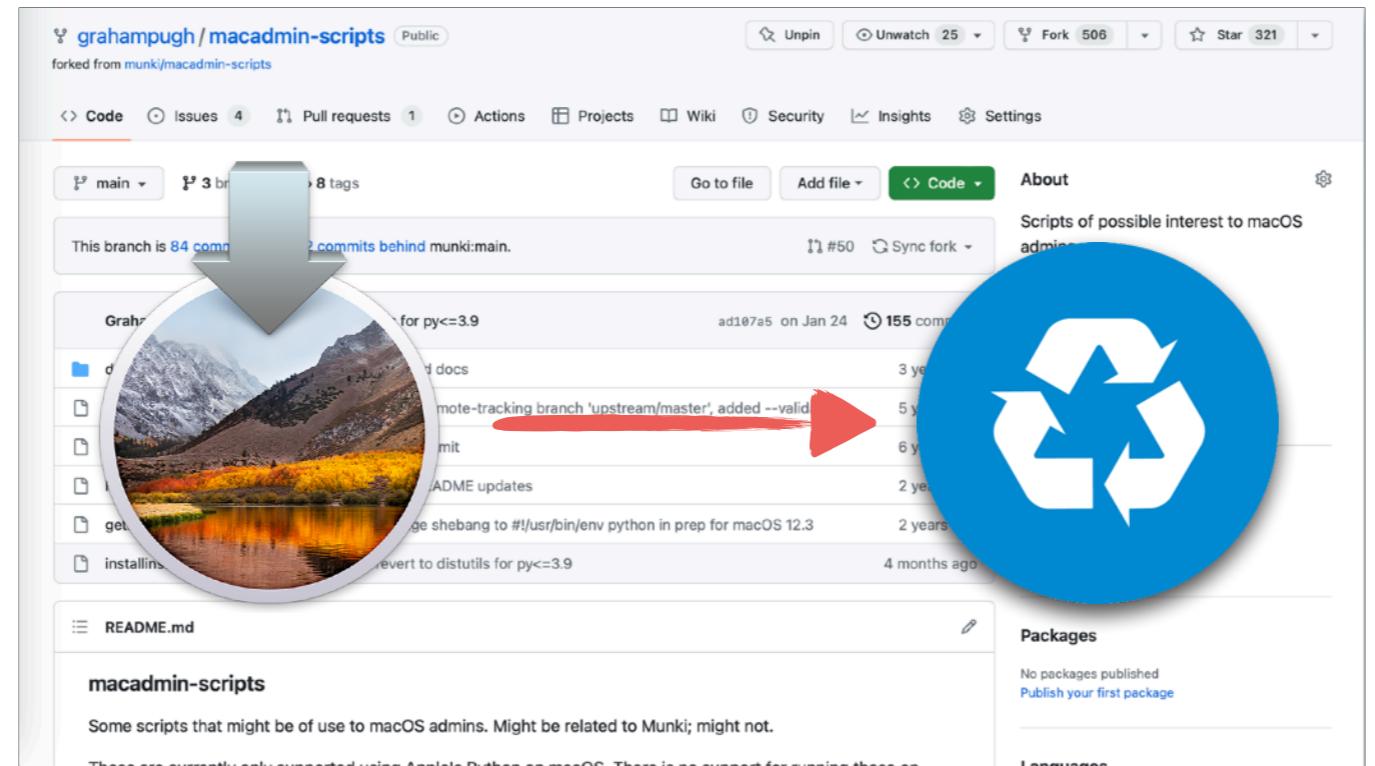
About

Scripts of possible interest to macOS admins

Readme View license 321 stars 25 watching 506 forks

Releases 8 tags Create a new release

Packages No packages published Publish your first package



**grahampugh / macadmin-scripts** Public  
forked from munki/macadmin-scripts

Code Issues 4 Pull requests 1 Actions Projects Wiki Security Insights Settings

main 3 branches 8 tags Go to file Add file Code About

This branch is 84 commits ahead, 2 commits behind munki:main. #50 Sync fork

Graham Pugh (iMac Pro) revert to distutils for py<=3.9 ad107a5 on Jan 24 155 commits

docs	updated docs	3 years ago
.gitignore	Merge remote-tracking branch 'upstream/master', added --validate o...	5 years ago
LICENSE.md	First commit	6 years ago
README.md	More README updates	2 years ago
getmacosipsws.py	Change shebang to #!/usr/bin/env python in prep for macOS 12.3	2 years ago
installinstallmacos.py	revert to distutils for py<=3.9	4 months ago

README.md

**macadmin-scripts**

Some scripts that might be of use to macOS admins. Might be related to Munki; might not.

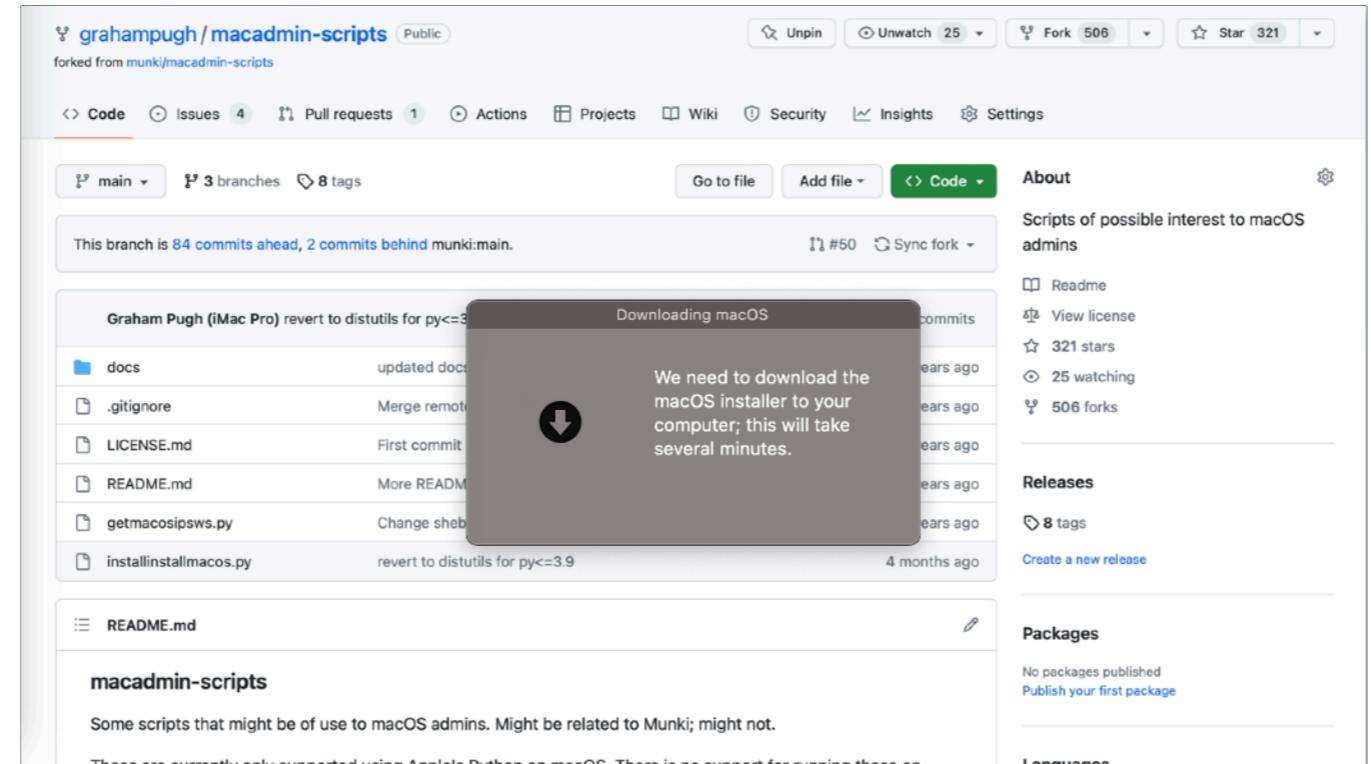
About Scripts of possible interest to macOS admins

Readme View license 321 stars 25 watching 506 forks

Releases 8 tags Create a new release

Packages No packages published Publish your first package

I threw in some jamfHelper user dialogs so the user would first see when the installer was downloading and then block the screen with a full screen dialog when the preparation for erasing began. I uploaded the script to our Jamf instances and created a self-service policy, which in effect became that one-click Erase All Contents And Settings button we'd been wanting for years.

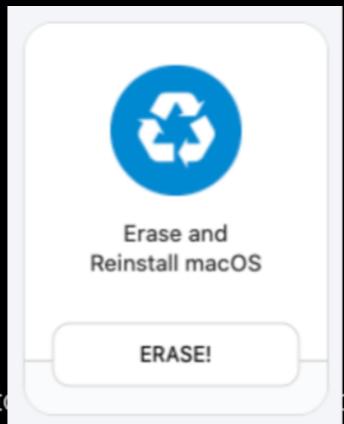


## Erasing macOS



Preparing the installer may take up to 30 minutes. Once completed your computer will reboot and continue the reinstallation.

## Erasing macOS



Preparing the installer may take up to [redacted] minutes. Once completed your computer will reboot and continue the reinstallation.

## Erase All Contents And Settings - erase and reinstall macOS in situ

Mar 26, 2018 •

Many Mac admins have been anticipating an "Erase All Contents And Settings" option for macOS, to emulate that available on iOS. Apple have taken a big step further towards that goal in macOS 10.13.4.

### `startosinstall --eraseinstall`

The `startosinstall` command is included in the `Install macOS High Sierra.app` (and, in fact, since *El Capitan*), which can be downloaded from the Mac App Store. An existing use case for this command is to install macOS High Sierra onto a blank volume, such as in a VM, an external drive, or onto a system volume from a NetInstall image using `Imagr`. A typical command to do so would be:

```
/Applications/Install\ macOS\ High\ Sierra.app/Contents/Resources/startosinstall \
--applicationpath /Applications/Install\ macOS\ High\ Sierra.app \
--agreetolicense --nointeraction --volume /Volumes/External\ Macintosh\ HD
```

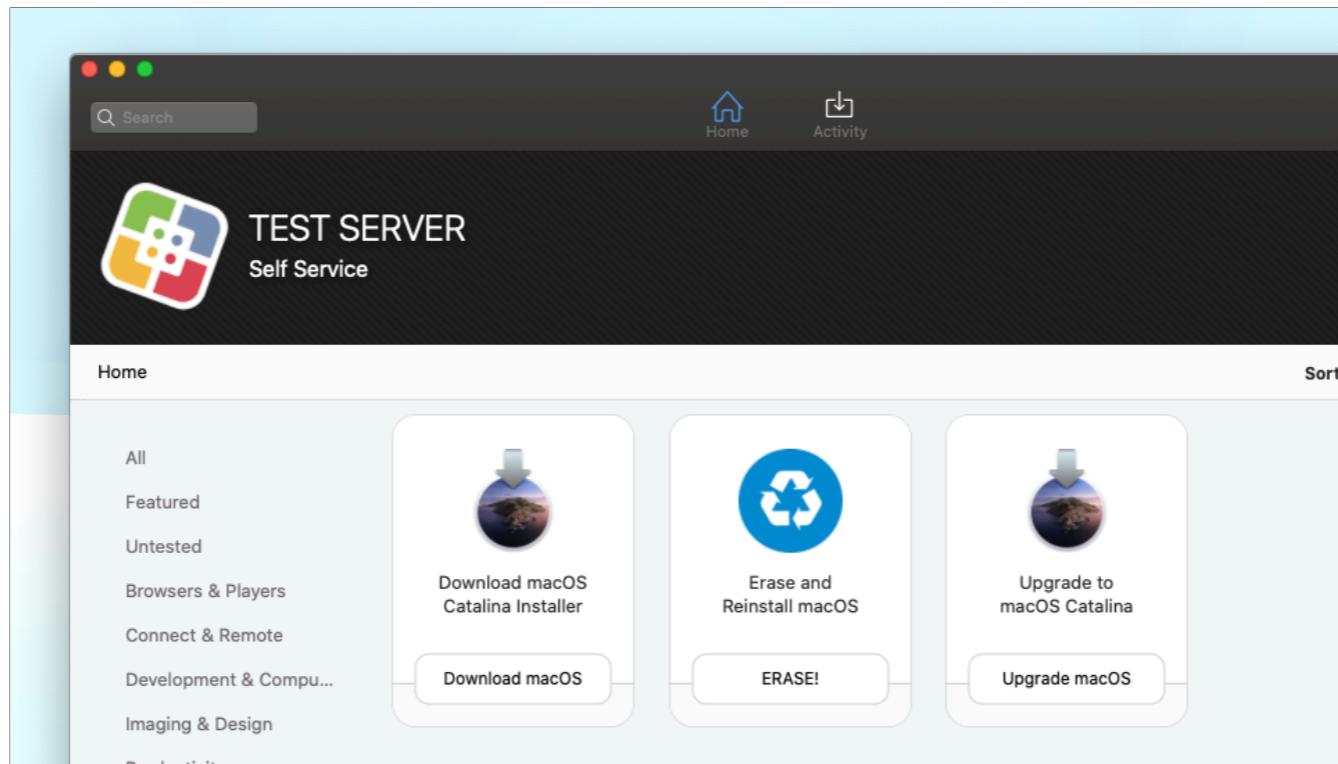
The `--nointeraction` flag is undocumented, as described in Rich Trouton's post, and allows the command to be run without interaction, meaning it can be scripted.

In the Apple Knowledge Base article [Prepare your institution for macOS High Sierra 10.13.4](#), Apple revealed a new flag for the `startosinstall` command, the `--eraseinstall` flag:

The `startosinstall` command inside the macOS Installer app at `Contents/Resources/startosinstall` lets you erase your boot volume and install macOS to it using the new `--eraseinstall` flag. You can use the `--installpackage` flag to specify additional packages to apply after installing macOS.

I blogged about the new feature when 10.13.4 was released, and uploaded my script to git, in case it was useful to anyone else, so had to think of a name for it.

I called the script `erase-install`, because that is all it did at the time.



I hadn't thought of using `startosinstall` for one-click upgrades until that point, but it was obviously useful to be able to offer a method of upgrading without requiring admin rights and without having to manually package up the installer. And it was hardly any additional code to the existing script to build a reinstall option into the erase-install script so that we could have a one-click Upgrade button in our Self Service.

So, calling the script `erase-install` was already a bad idea...

```
sudo softwareupdate -ia
```

So, now we had a good solution for upgrades and redeployments.

For software **updates** we had our tried-and-tested command for forcing updates via a script using the **softwareupdate** command. 🍎 But this also hit a problem with the T2 Mac...

T2 Macs sometimes had to be shut down instead of restarted for some updates. Apple added the 🍎 **-R** option to the **softwareupdate** command, which would figure out whether a shut down was necessary. But that wasn't great, because users could potentially run this at the end of the day, and then arrive at their desk with the computer mysteriously switched off, and have to wait perhaps another 30 minutes for the update to finish.

🍎 On the other hand, installing the new version using the full installer with `startosinstall` didn't need this shutdown.

```
sudo softwareupdate -ia
```



```
sudo softwareupdate -iaR
```



**sudo softwareupdate -iaR**





Furthermore, the software update command started getting, just, unreliable. Updates would fail to show, 🍎 or the update would fail to install at all, or the system would end up in a boot recovery state. 🍎

Greg Neagle, who I always trust to make wise decisions, lost patience with Apple in 2020 and 🍎 removed the softwareupdate command from Munki for upgrades and updates that require a restart, because of this unpredictability.

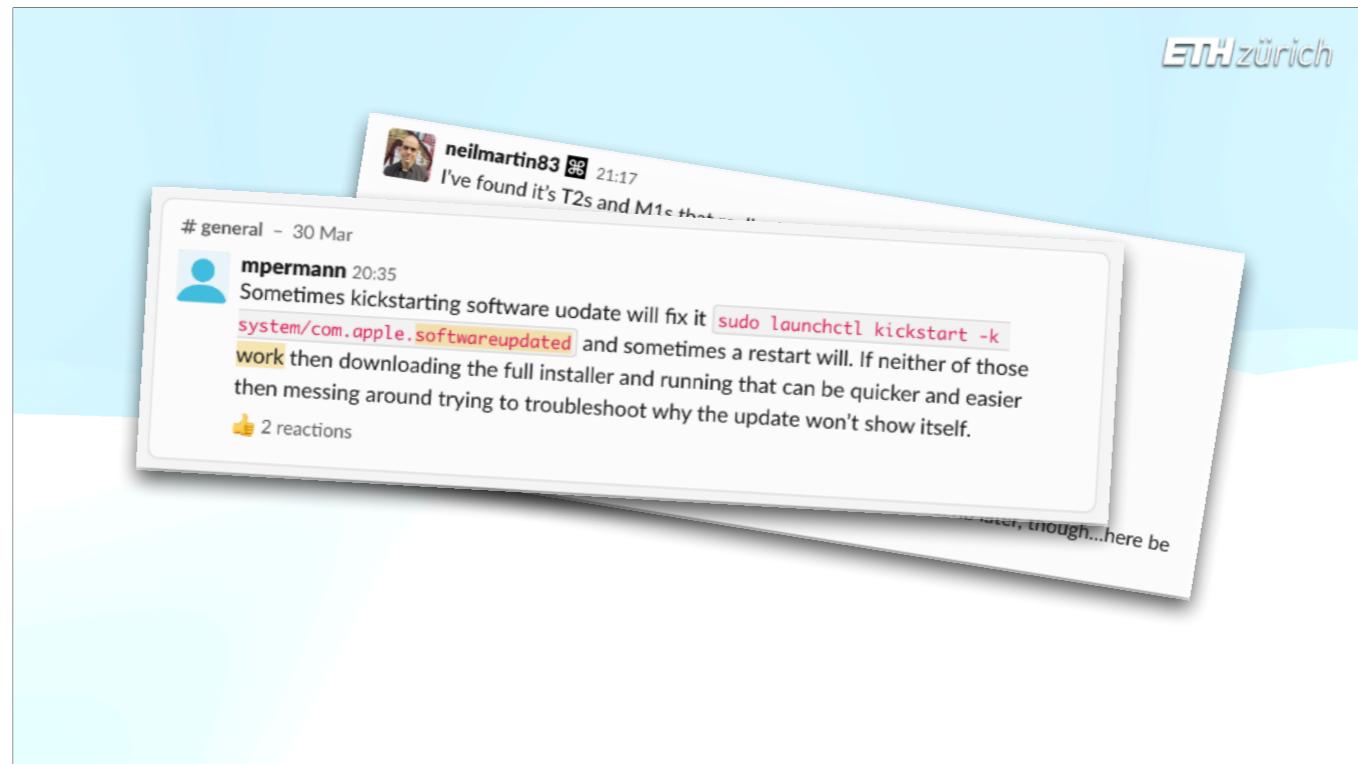
But startosinstall was still reliable, so using erase-install for updates was a predictable workflow, even if it's always seemed like overkill to use a full installer to do a minor update.

 **neilmartin83** 21:17 I've found it's T2s and M1s that really don't like `softwareupdate`  
Seemed to get worse with Catalina as well on those platforms

 **vmiller** 21:18 I think that is probably right

 **neilmartin83** 21:20 I found it was solid on 10.14.x and pre-T2s e.g the 2013-2017 iMacs

 **flammable** 21:44 yeah, if you're still using 10.14, `softwareupdate` should be fine. no later, though...here be dragons.





Search or jump to... / Pull requests Issues Codespaces Marketplace Explore

munki / munki Public Watch 2

Code Issues 79 Pull requests 5 Actions Projects Wiki Security Insights

## Manual Apple Updates in Munki 5

Eric Holtam edited this page on May 13, 2020 · 4 revisions

The handling of Apple software updates that require a restart is the biggest change in Munki 5. Apple's `softwareupdate` tool, which Munki uses to install Apple software updates, has become increasingly unreliable at the task of installing macOS updates and Security Updates. This mostly affects Macs with T2 chips, though it's not completely clear if the problems are limited to those models. Some of the observed issues include but are not limited to these:

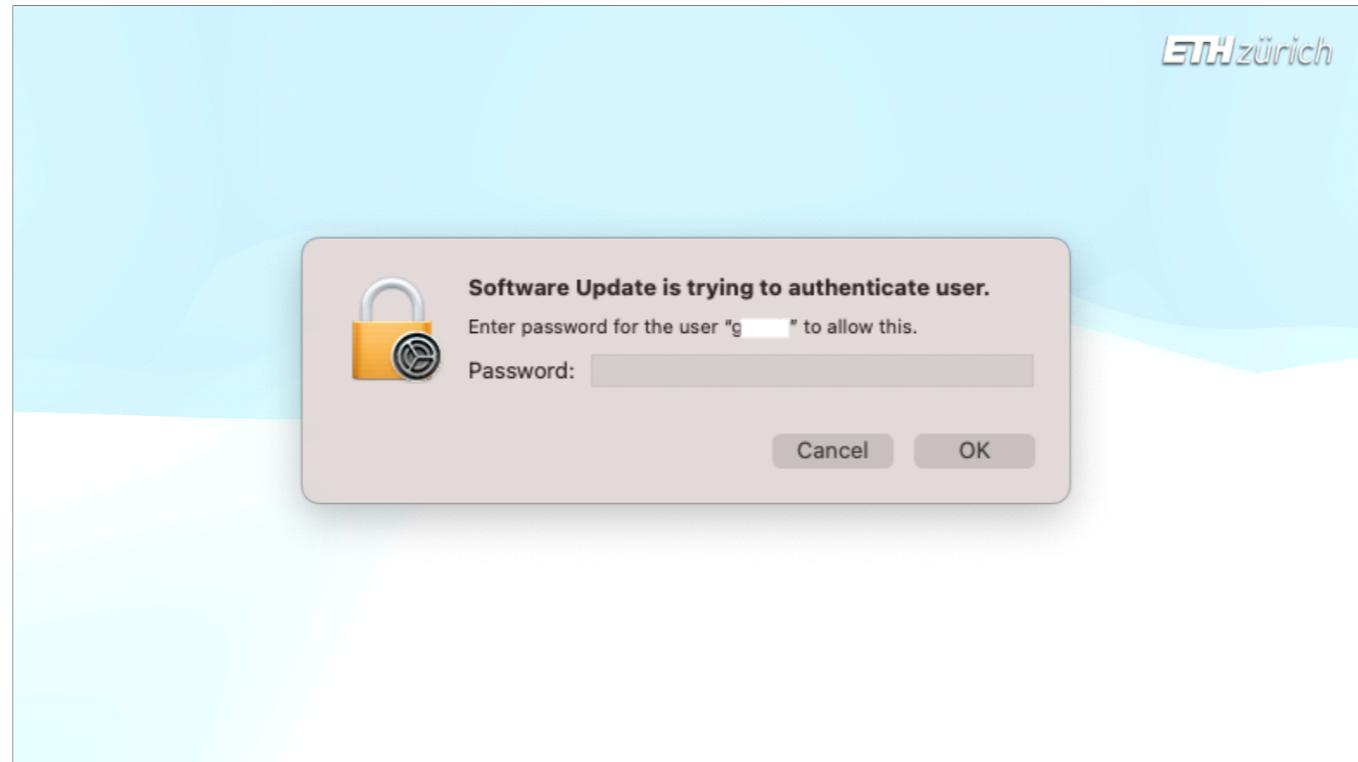
- After an attempted install of a macOS update or Security Update, the Mac boots into "Boot Recovery Utility", requiring manual intervention to get the machine to start up in macOS.

... macOS update or Security Update hangs indefinitely.  
... all these updates.

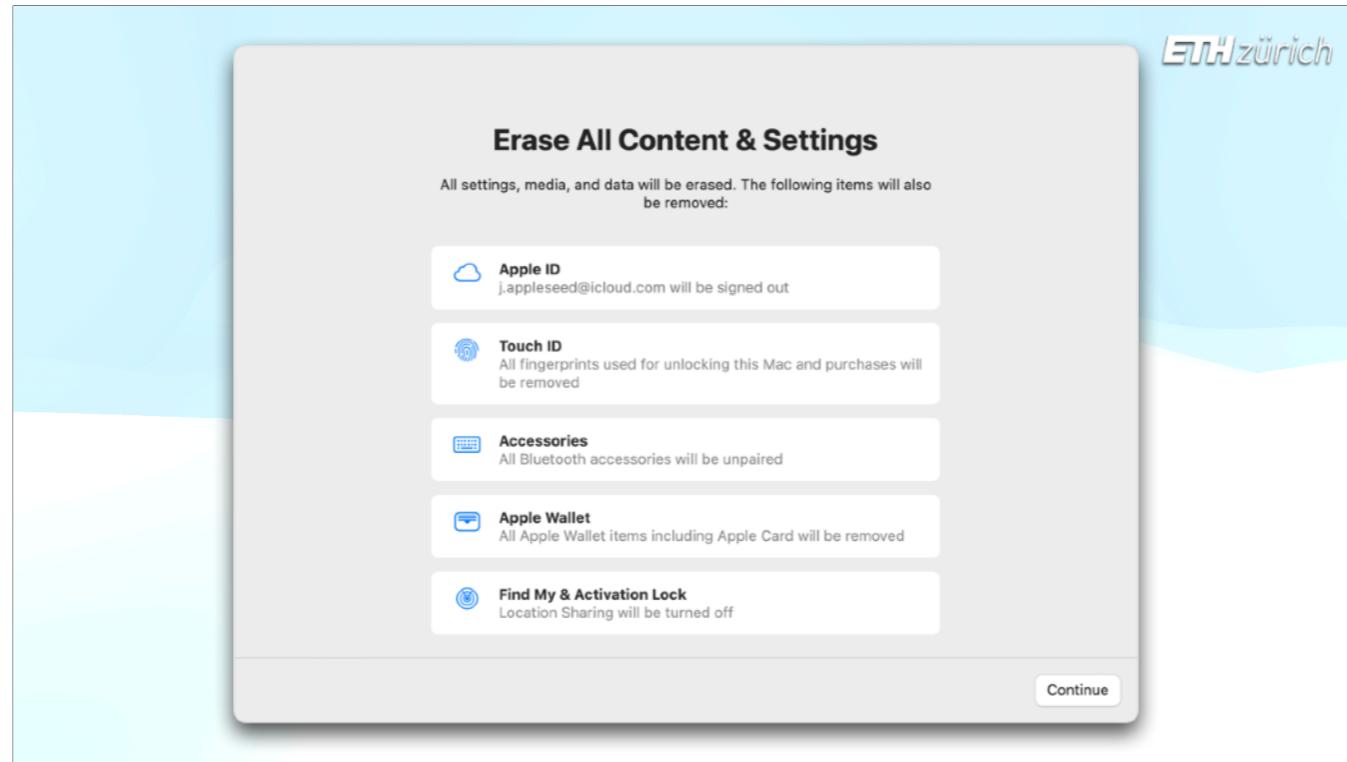


At WWDC 2020, things changed again, as Apple announced Apple Silicon chips. 🍎 The Apple Silicon DTK was released, along with the beta version of Big Sur.





We found out that Apple Silicon computers required a Volume Owner to supply their password via a graphical prompt to unlock the System Volume, making an unattended update impossible.



Soon, Apple finally gave us a built in Erase All Contents & Settings tool! So the original use case of the erase-install script was well and truly Sherlocked!

🍎 when Erase All Contents And Settings works, that is...

Still, with a built in erase feature and with all the new restrictions introduced with Apple Silicon, you'd think use of erase-install would have died away. Quite the opposite proved the case.



**Erase Assistant is not supported  
on this Mac**

Learn More

Quit

Continue



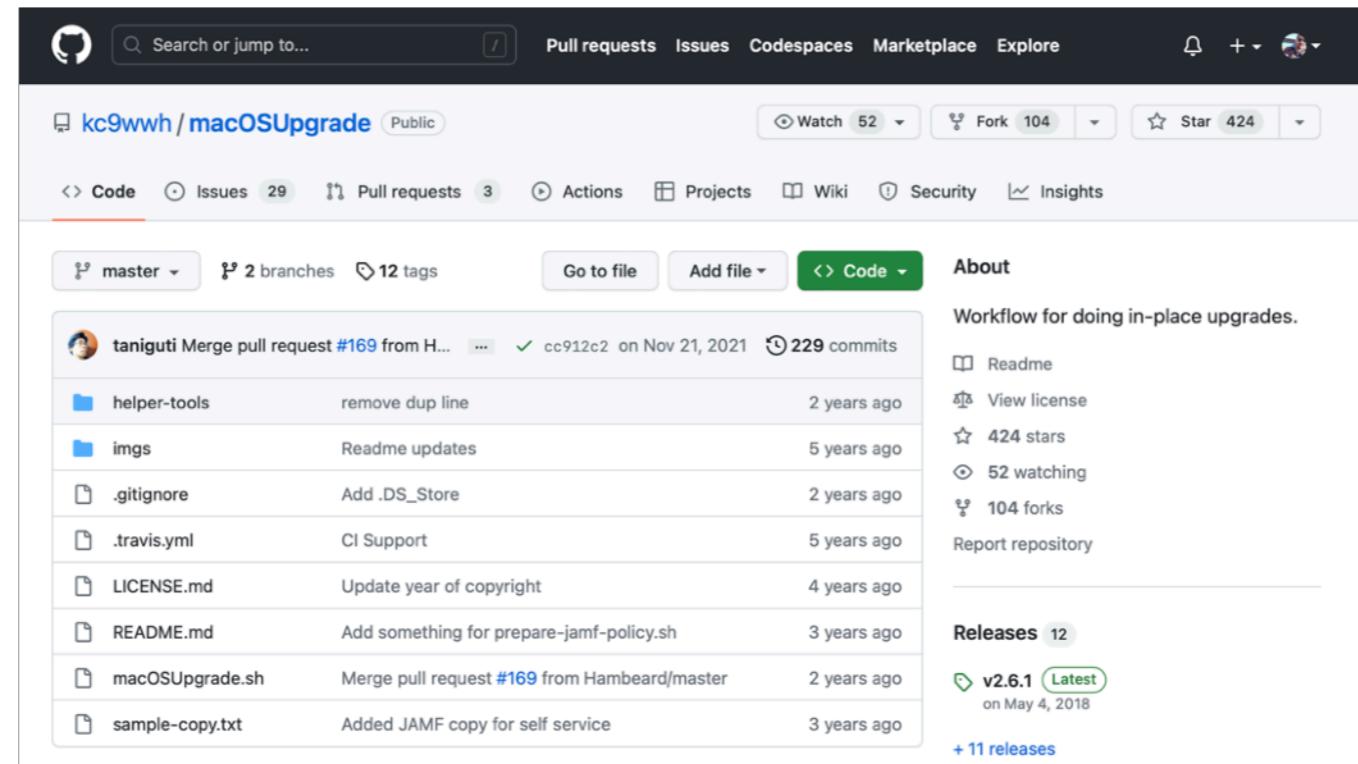
In Big Sur, startosinstall gained new options to allow us to enter the account name and password for Apple Silicon computers – unfortunately in plain text.

But at least it didn't need to be an admin password, just a Volume Owner, and we could control **when** we ask for the user's password in the workflow, and how to present the question to the user. So I added graphical prompt for the user's password to erase-install.

```
sudo '/Applications/Install macOS Big Sur.app/  
Contents/Resources/startosinstall'
```

```
--user fredbloggs  
--stdinpass <<< plaintextpassword
```



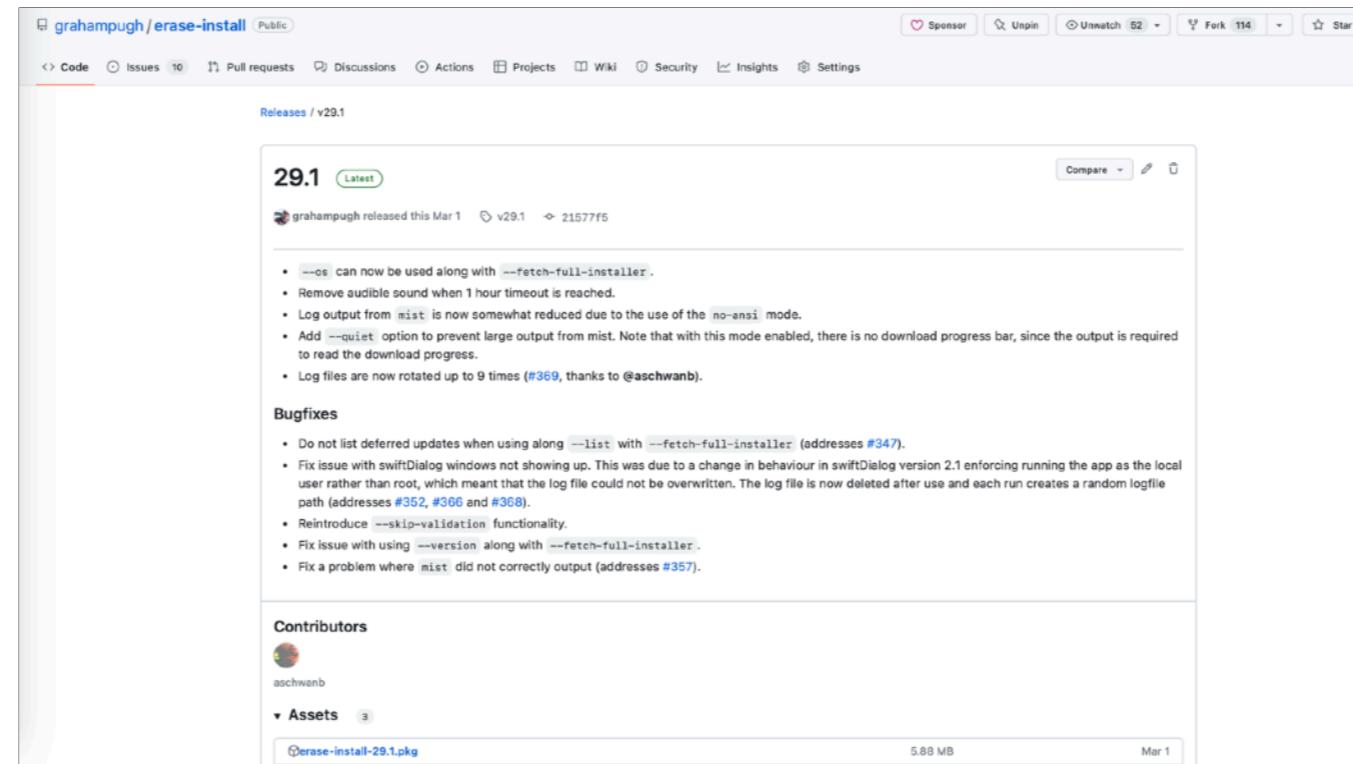


Because `softwareupdate` was getting less and less reliable, and also perhaps because Jamf's previously recommended `macOSUpgrade` script stopped getting developed at this point, use of `erase-install` seems to have been ballooning ever since.



I've also worked on a bunch of improvements and updates in the past two years as the usage has increased, which may have also caused usage to increase. But the functionality has not fundamentally changed between the release of Apple Silicon and today.

So, that's the history of this project. Let's go back and look at what the process looks like today on an Apple Silicon Mac.



If you download the package that I've provided in the GitHub repo, this includes all the dependencies you'll need to use the script, namely Bart Reardon's swiftDialog app and Nindi Gill's mist-cli tool, which has now replaced my version of installinstallmacos.

grahampugh / erase-install Public

Code Issues 10 Pull requests Discussions Actions Projects Wiki Security Insights Settings

Releases / v29.1

**29.1** Latest

grahampugh released this Mar 1 v29.1 · 21577f5

• `--os` can now be used along with `--fetch-full-installer`.  
• Remove audible sound when 1 hour timeout.  
• Log output from `mist` is now somewhat reduced.  
• Add `--quiet` option to prevent large output when using `--fetch-full-installer`.  
• Log files are now rotated up to 9 times (addresses #347).  
• There is no download progress bar, since the output is required to read the download progress.  
• Bugfixes  
• Do not list deferred updates when using along with `--os` (addresses #347).  
• Fix issue with swiftDialog windows not showing correctly when running as root. This was caused by a bug in swiftDialog version 2.1 enforcing running the app as the local user rather than root, which meant that the log file could not be deleted after use. Now the log file is now deleted after use and each run creates a random logfile path (addresses #352, #366 and #368).  
• Reintroduce `--skip-validation` functionality.  
• Fix issue with using `--version` along with `--fetch-full-installer`.  
• Fix a problem where `mist` did not correctly output (addresses #357).

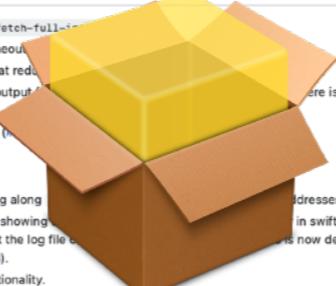
Bugfixes

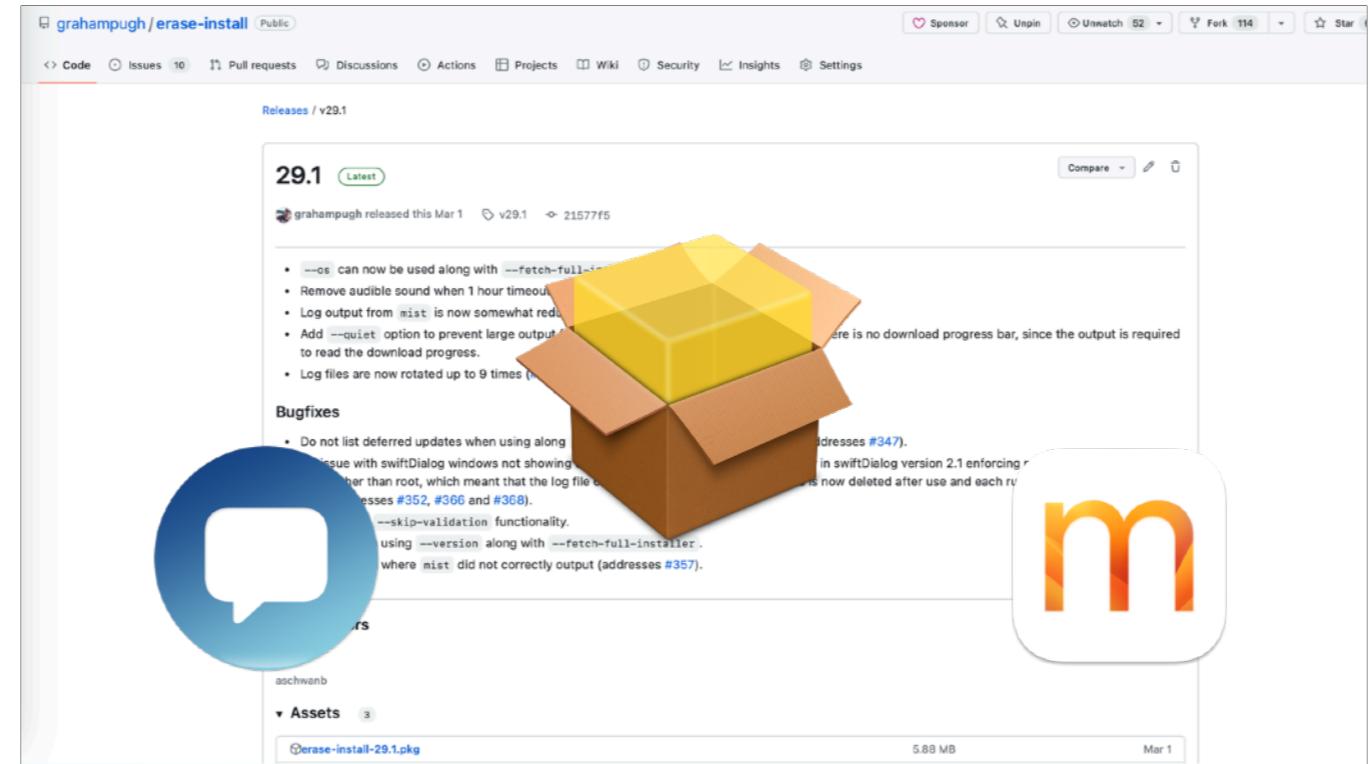
Contributors

aschwanb

Assets 3

erase-install-29.1.pkg 5.89 MB Mar 1

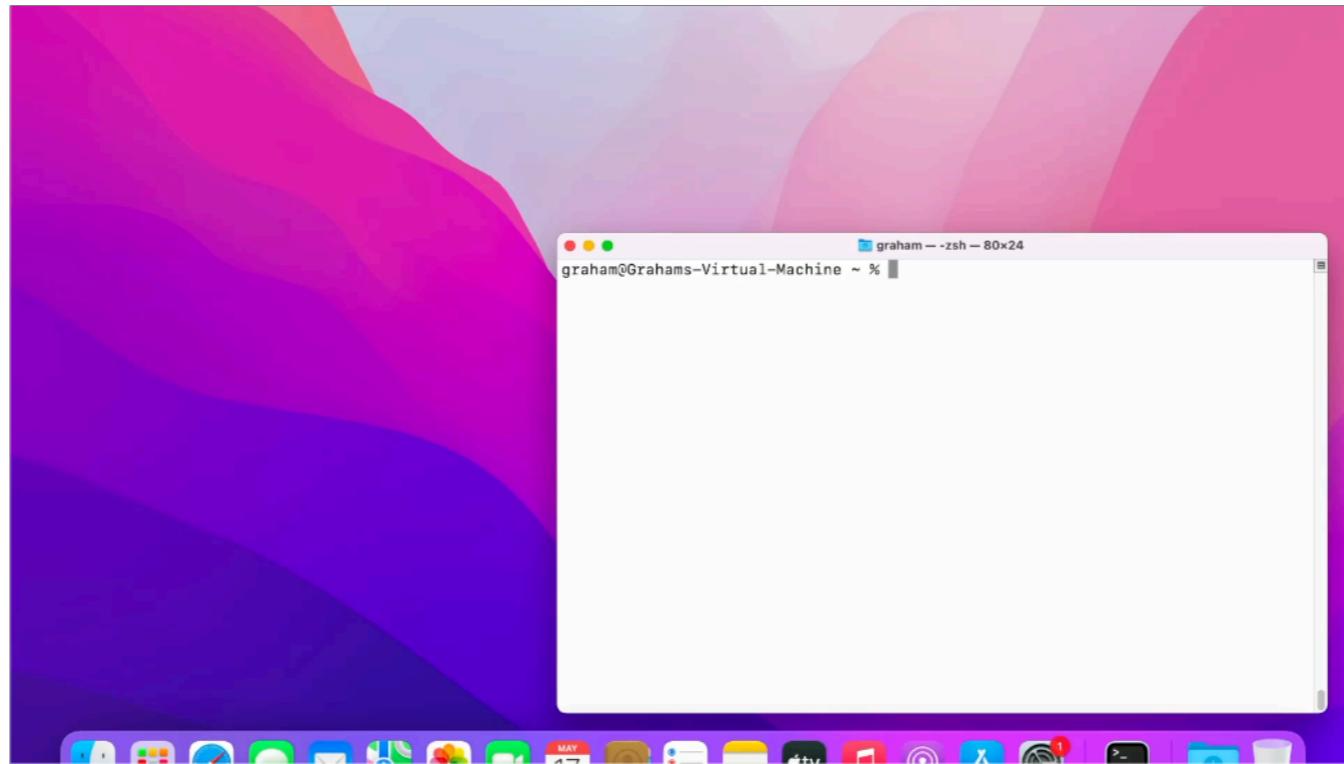






Alternatively you can run the script directly and it will download Mist and SwiftDialog when it needs them. You can even run a curl command to download the script and pipe in the arguments you want to use, which is what I did in the screen recording you saw at the beginning, that I'm going to talk you through now.

```
curl -s https://raw.githubusercontent.com/  
grahampugh/erase-install/main/erase-install.sh  
| sudo bash /dev/stdin <arguments>
```



## VIDEO

We're going to run the reinstall option here to upgrade from Monterey to Ventura. As I said before, Terminal won't be seen when running from your management system, but here it allows you to see more of what's happening in the background.

Of course this recording has been reduced from a good half hour or more down to 2 minutes.

🍎 The user is asked for their password which is required on Apple Silicon.

It then downloads Mist and swiftDialog if they are not already on the device.

Erase-install tells Mist to look for the latest available macOS installer, because we did not specify a version to download. In this case it's 13.3.1.

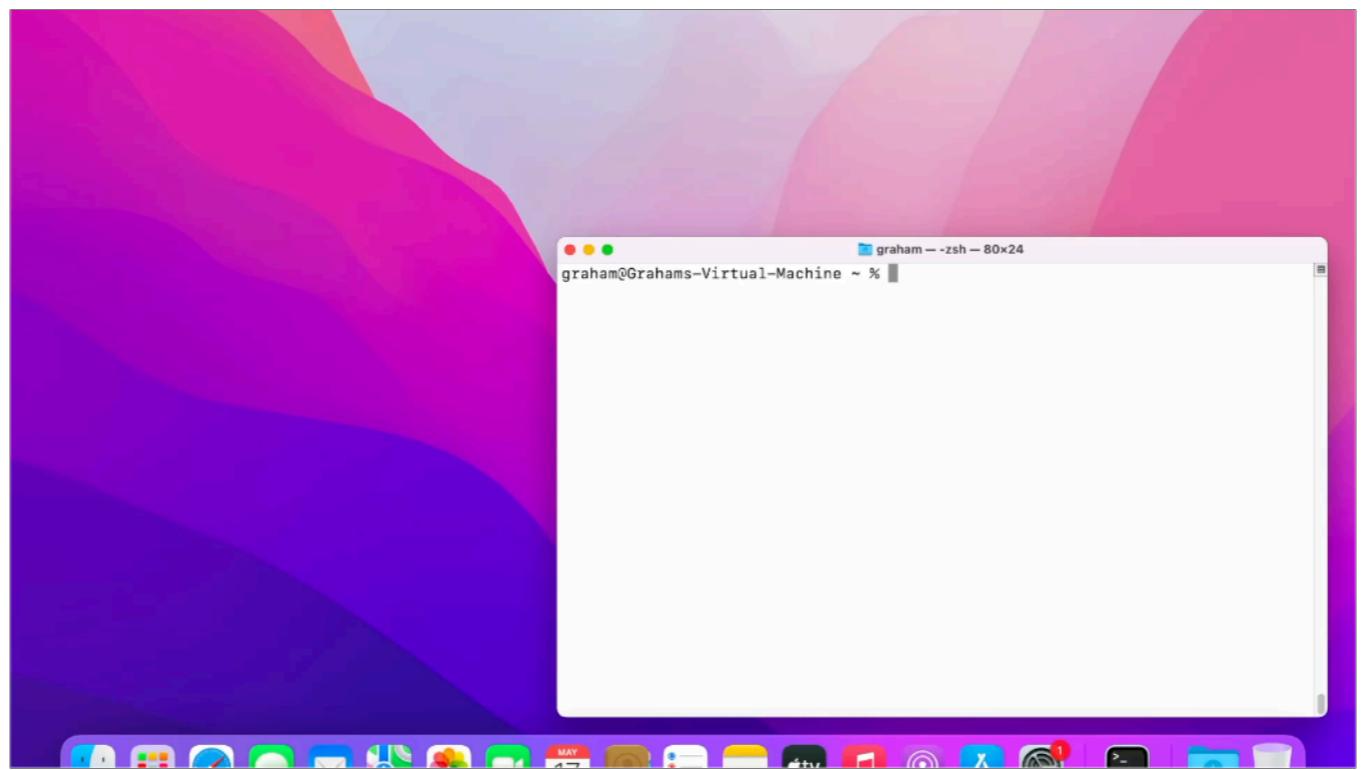
A dialog shows the percentage progress of the download.

Once downloaded, Mist compiles all the components that were downloaded to make the installer app, which creates a DMG, and then the installer is moved into place in the Applications folder.

Once finished, erase-install takes over again and invokes startosinstall to reinstall macOS on the device.

This time a full-screen dialog shows the percentage progress of the OS preparation phase prior to download. This prevents the user from continuing to work.

Once the preparation is complete, the machine automatically reboots and after login we can see that the machine is now on Ventura, although in the VM I recorded the login screen retained the Monterey background during the first login.



```
erase-install.sh --reinstall / --erase
```

As you saw, just specifying the reinstall or erase options is enough. However, there are a whole bunch of additional options you can use, for example 🍎 to specify which installer you download, 🍎 to replace outdated installers,

```
erase-install.sh --reinstall / --erase  
--sameos / --os 13 / --version 13.3.1
```

```
erase-install.sh --reinstall / --erase  
--sameos / --os 13 / --version 13.3.1  
--update / --overwrite /  
--replace-invalid
```

### Waiting for AC Power Connection



Please connect your computer to power using an AC power adapter.

This process will continue if AC power is detected within the specified time.

57

to check for power,

### Waiting for AC Power Connection

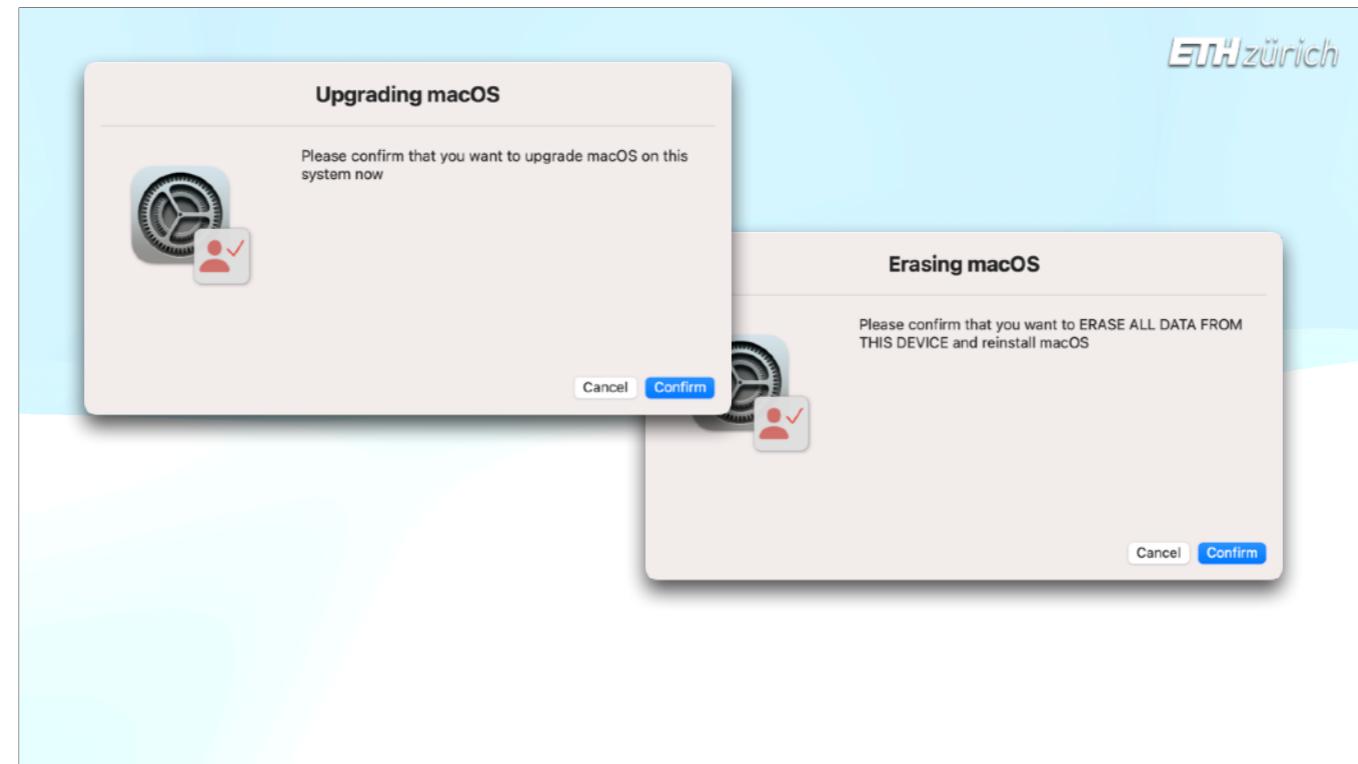


Please connect your computer to power using an AC power adapter.

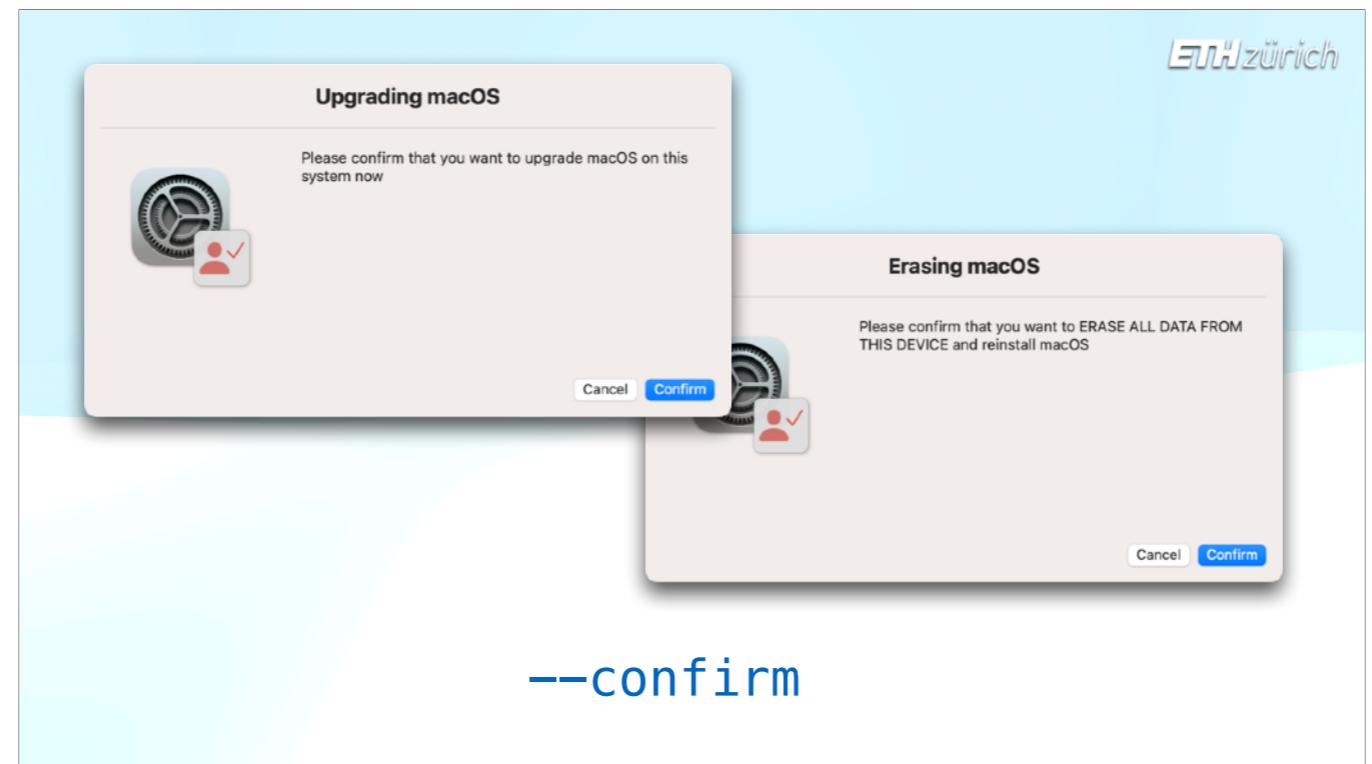
This process will continue if AC power is detected within the specified time.

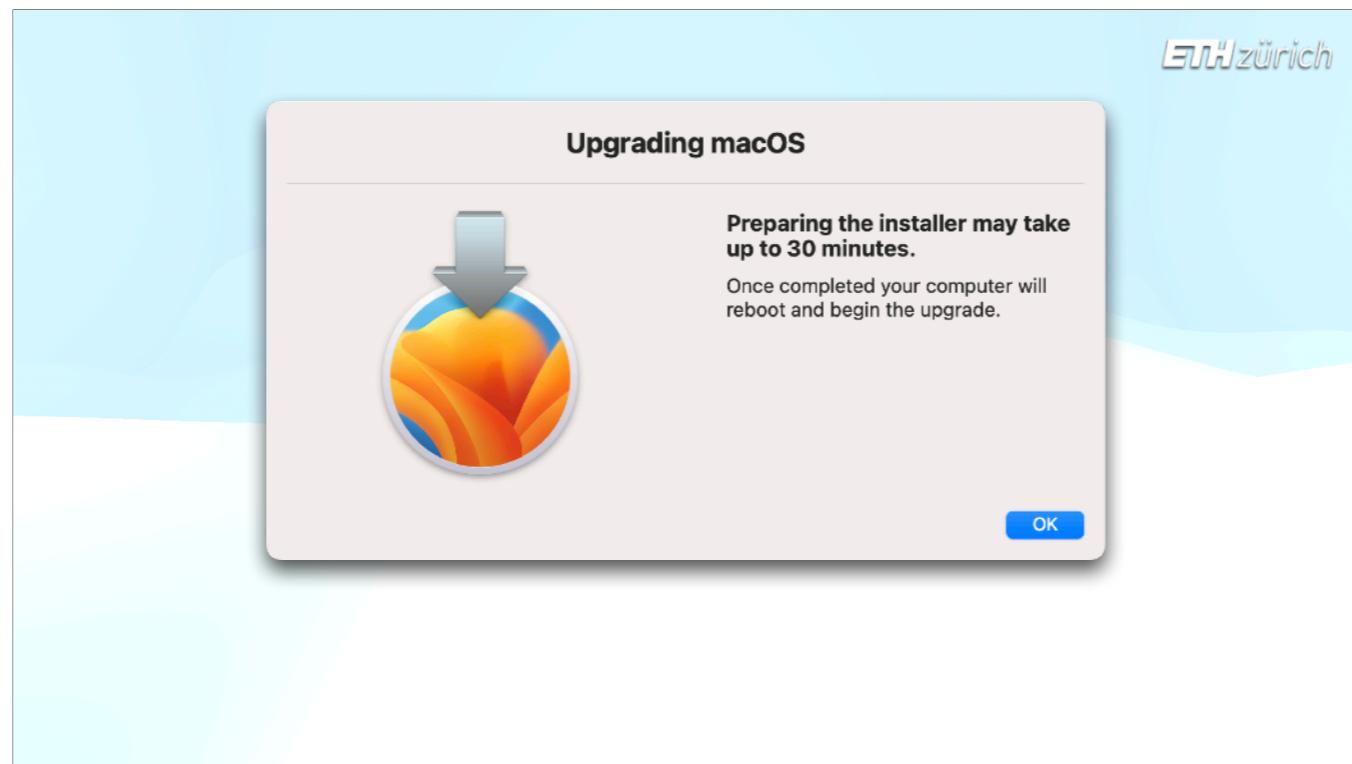
57

--check-power  
--power-wait-limit XX



whether to ask for user confirmation





To add a delay to the reboot to allow users to carry on working during the prep phase, and so on.

Upgrading macOS



Preparing the installer may take  
up to 30 minutes.

Once completed your computer will  
reboot and begin the upgrade.

OK

--rebootdelay 300



I've been showing the dialogs in English, but the text is available in 5 language depending on the language set on the device. 🍎 Fortunately, I have had a lot of help from native speakers, so hopefully these aren't too bad!

*Edu*zürich



*Edu*zürich



To see all possible options, just add --help.

🍎 The wiki also has pretty much everything you could possibly need to know.

🍎 The Slack channel is there if you've read the wiki and still not sure what to do. And of course if there's time for questions at the end, or if you see me during the rest of the conference, 🍎 I'll be happy to answer.

**erase-install.sh --help**

**erase-install.sh --help**

**[github.com/erase-install/wiki](https://github.com/erase-install/wiki)**

**erase-install.sh --help**

**[github.com/erase-install/wiki](https://github.com/erase-install/wiki)**

 **#eraseinstall**

**erase-install.sh --help**

**[github.com/erase-install/wiki](https://github.com/erase-install/wiki)**



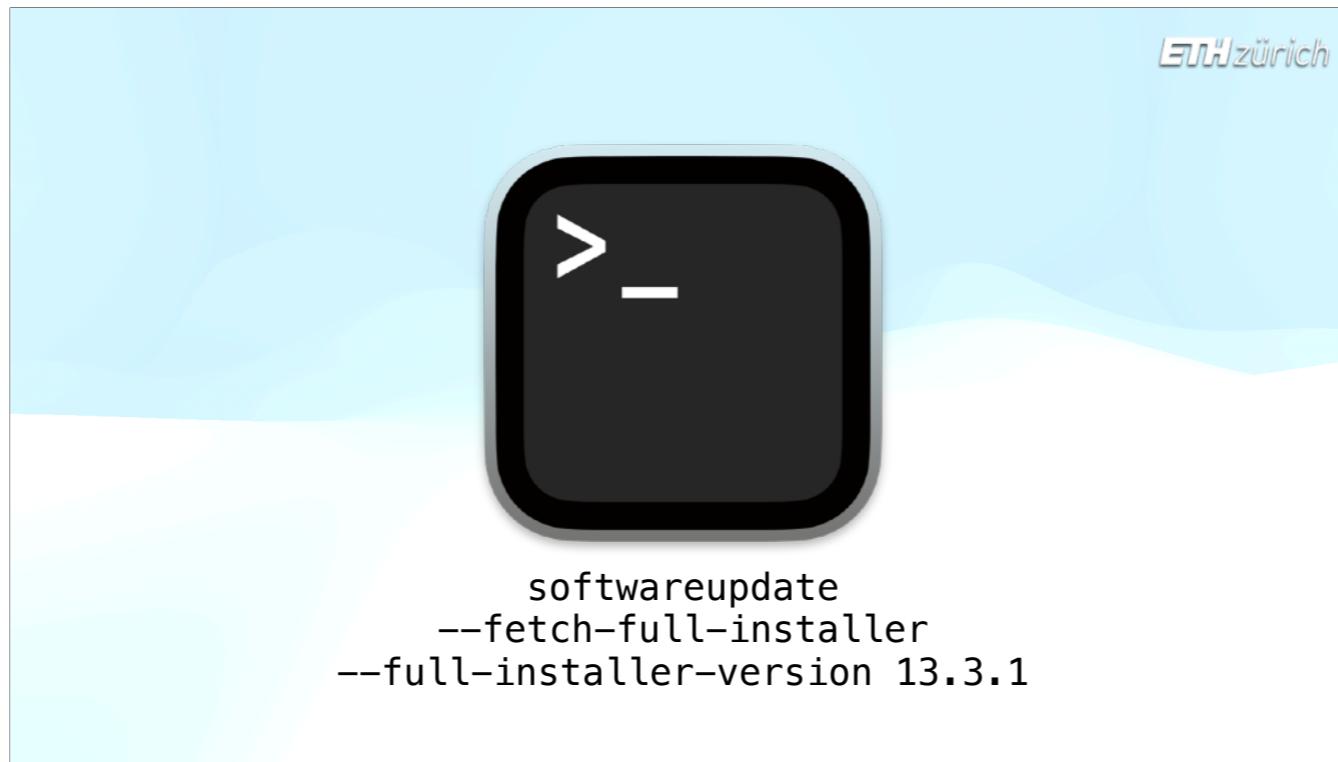
**#eraseinstall**





However, Erase-install shouldn't really need to exist.

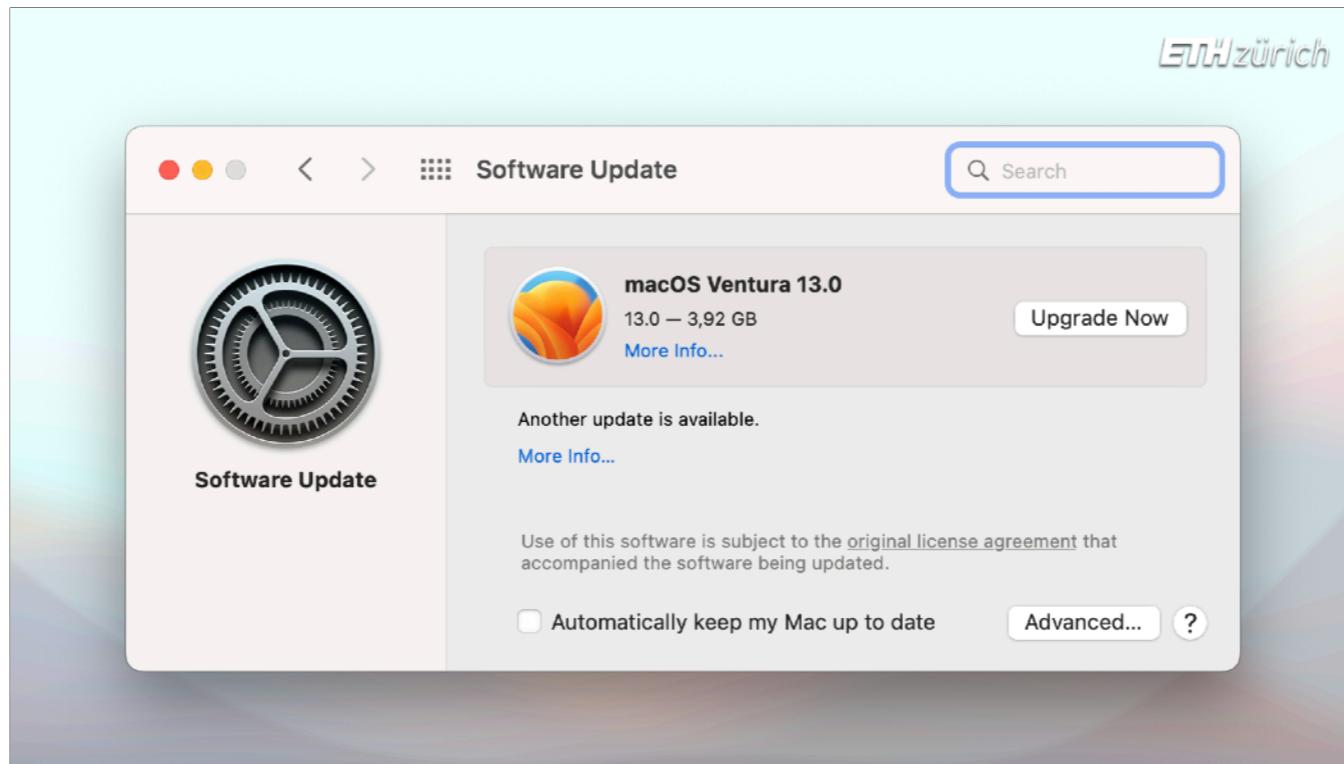
Let's look at what Apple have been doing to try and plug some of the holes that erase-install has been filling.



All the way back in the release of macOS Catalina, a new flag for the softwareupdate command was added named "fetch-full-installer". This allowed the download of the full installer directly from Apple's softwareupdate catalog without need for a script like installinstallmacos.

Unfortunately, despite a few improvements every year, it has never been as reliable as the methods used in installinstallmacos or Mist. For some organisations it can be useful though – for one thing, it honours MDM update deferrals, and for another, it can make use of a caching server.

I've built this into the erase-install script as an alternative option for those that want to use it but still wish to make use of the dialogs that erase-install offers.

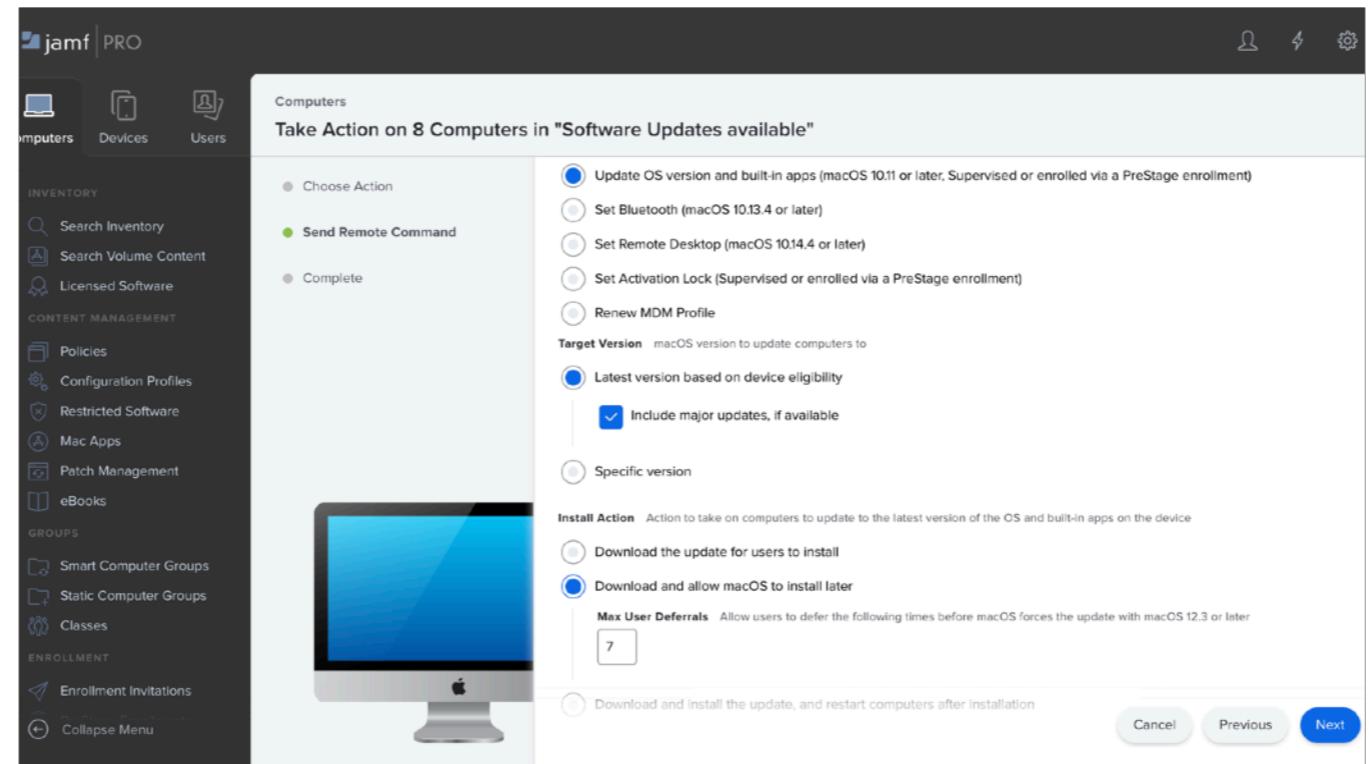


Another very useful development that came with the release of macOS Ventura is the ability to upgrade from an older major macOS using a delta update from the Software Update pane without needing admin rights. This should make upgrades easier.

## Resolved Issues in macOS 13.3 Beta

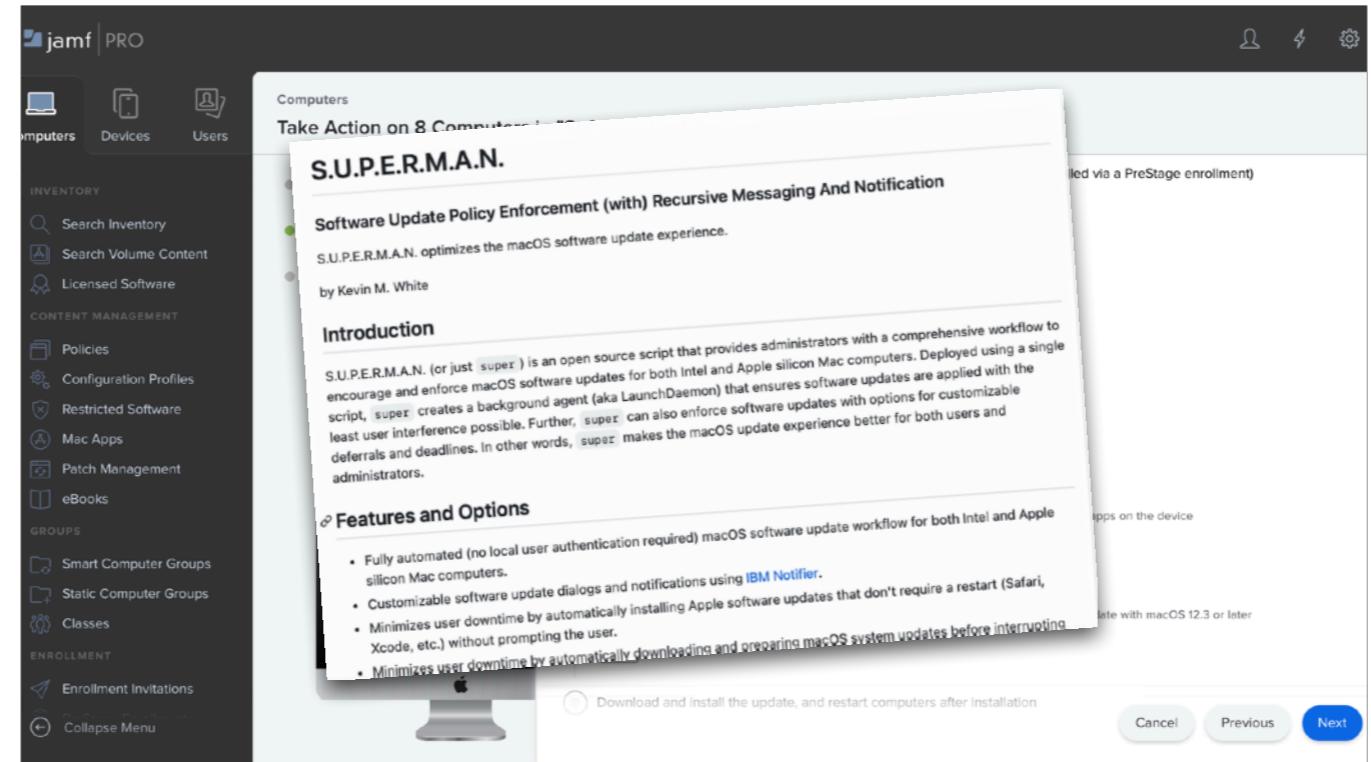
- Software update scans no longer fail to return results when concurrent scans are initiated. **Note:** Programmatic usage of `launchctl kickstart` to periodically restart `softwareupdated` may cause update failures and is not recommended.
- Reliability is improved when installing software updates overnight.
- Resolved an issue where the user deferral count for `Install Later` software updates reset after a restart.

Of course, that's only useful if Software Update gets reliable again. Some of the problems we've had with the `softwareupdate` command hanging over the last 2 or 3 major macOS versions are supposedly fixed in macOS 13.3. We still need more data on that one, but hopefully it's going to start to be reliable enough again to risk using it in scripts once more.



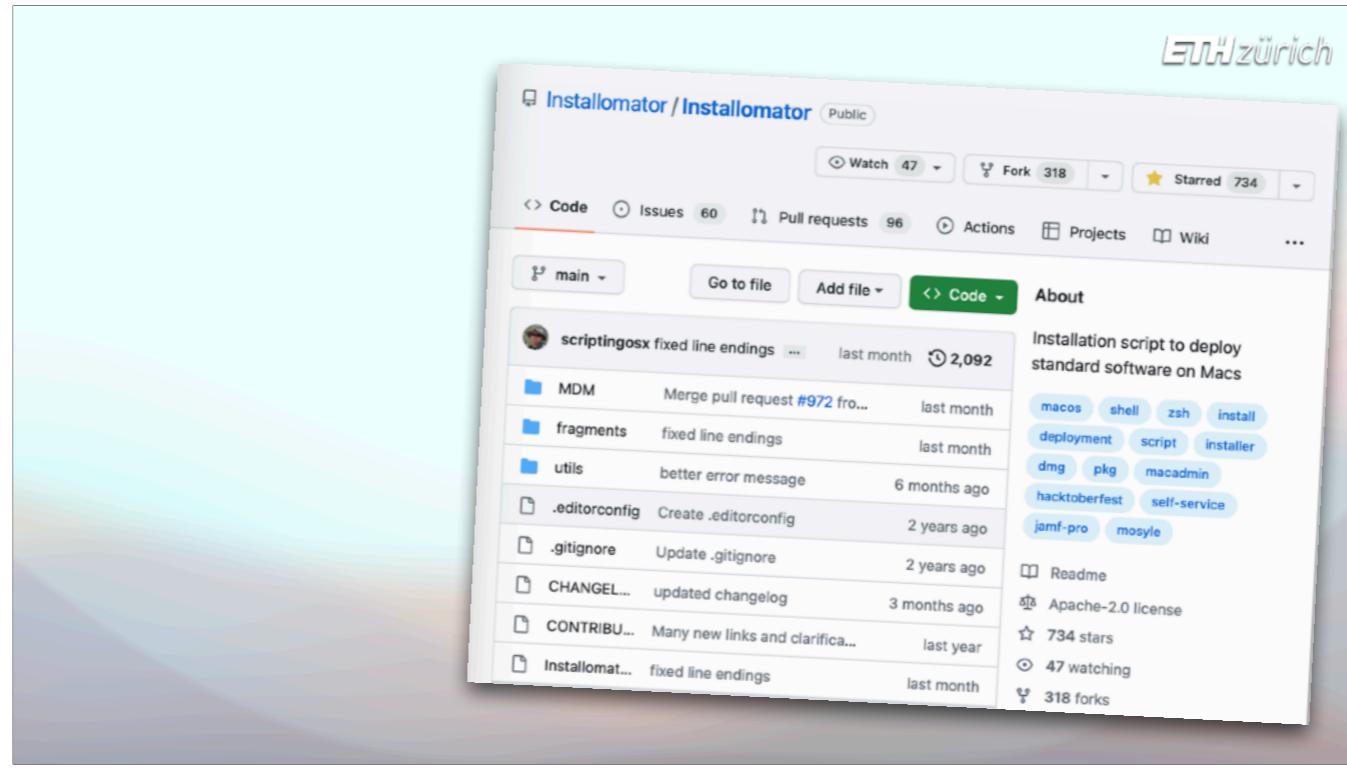
MDM commands are also now available to enforce software updates, including an option to allow the user to defer the update a set number of times.

I don't think these commands are robust enough to rely on them yet, and the user experience of a full enforcement is not yet acceptable. 🍎 Kevin's SUPERMAN script is a great attempt at improving that user experience, but the fact that it takes over 7500 lines of bash code to achieve it tells you how far away from a satisfactory built-in solution we are currently.

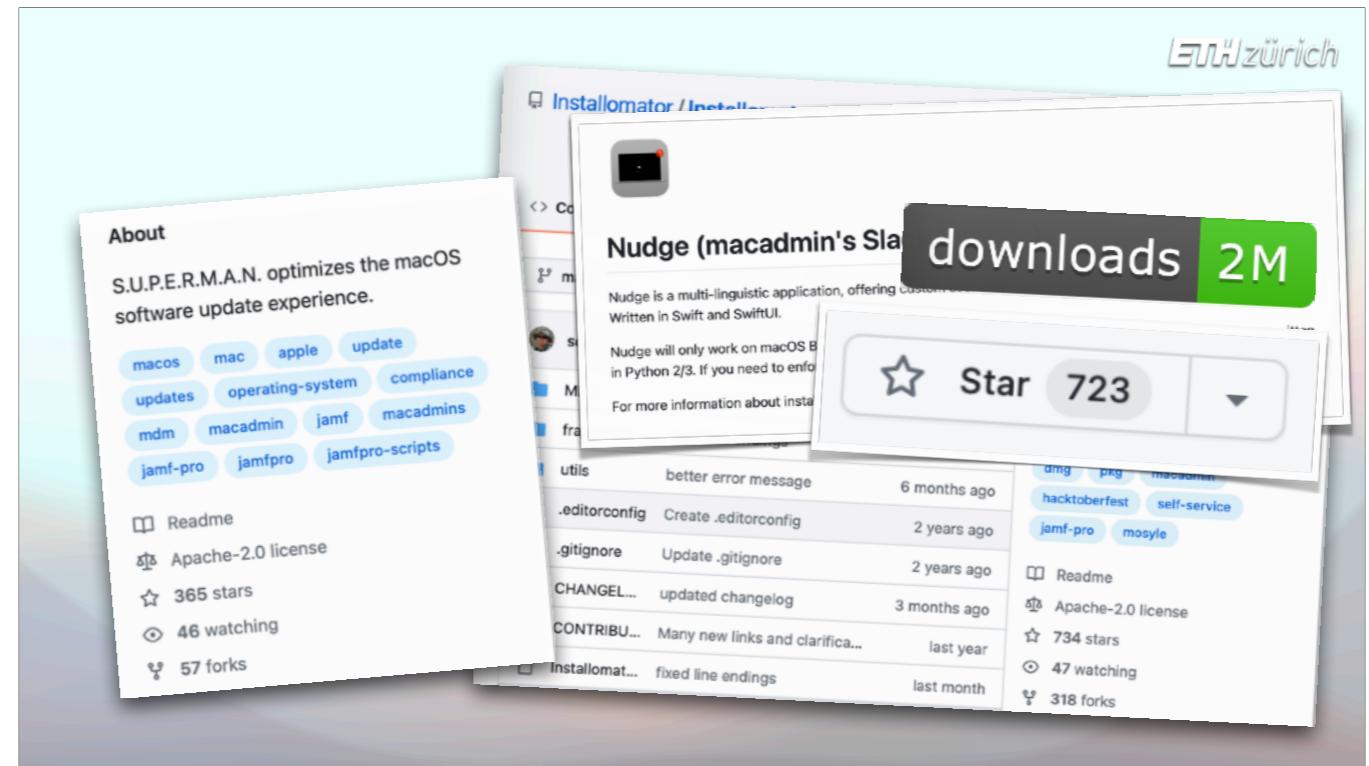


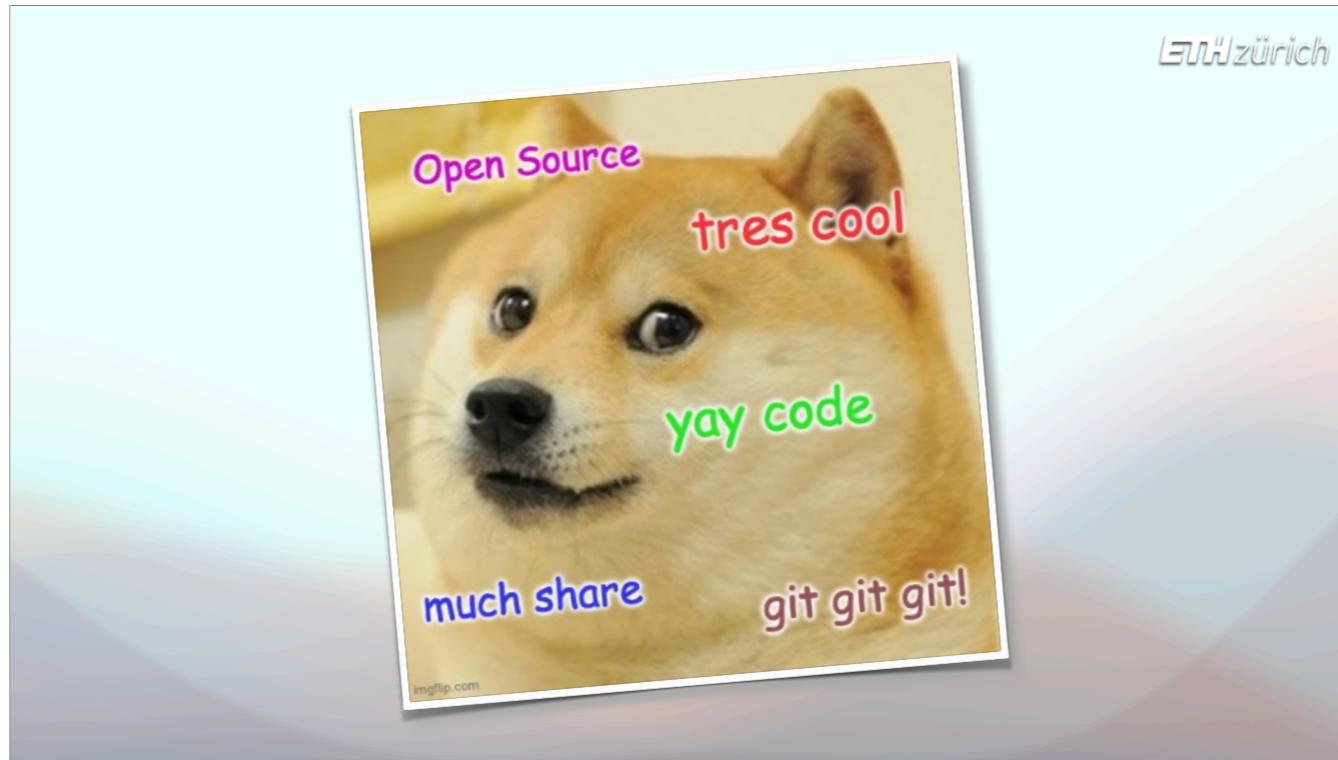


So, we are sadly not yet at a stage where Nudge, erase-install and Superman are becoming obsolete. We can only hope that changes announced in WWDC '23, mean we won't need these tools anymore... watch this space, I guess!



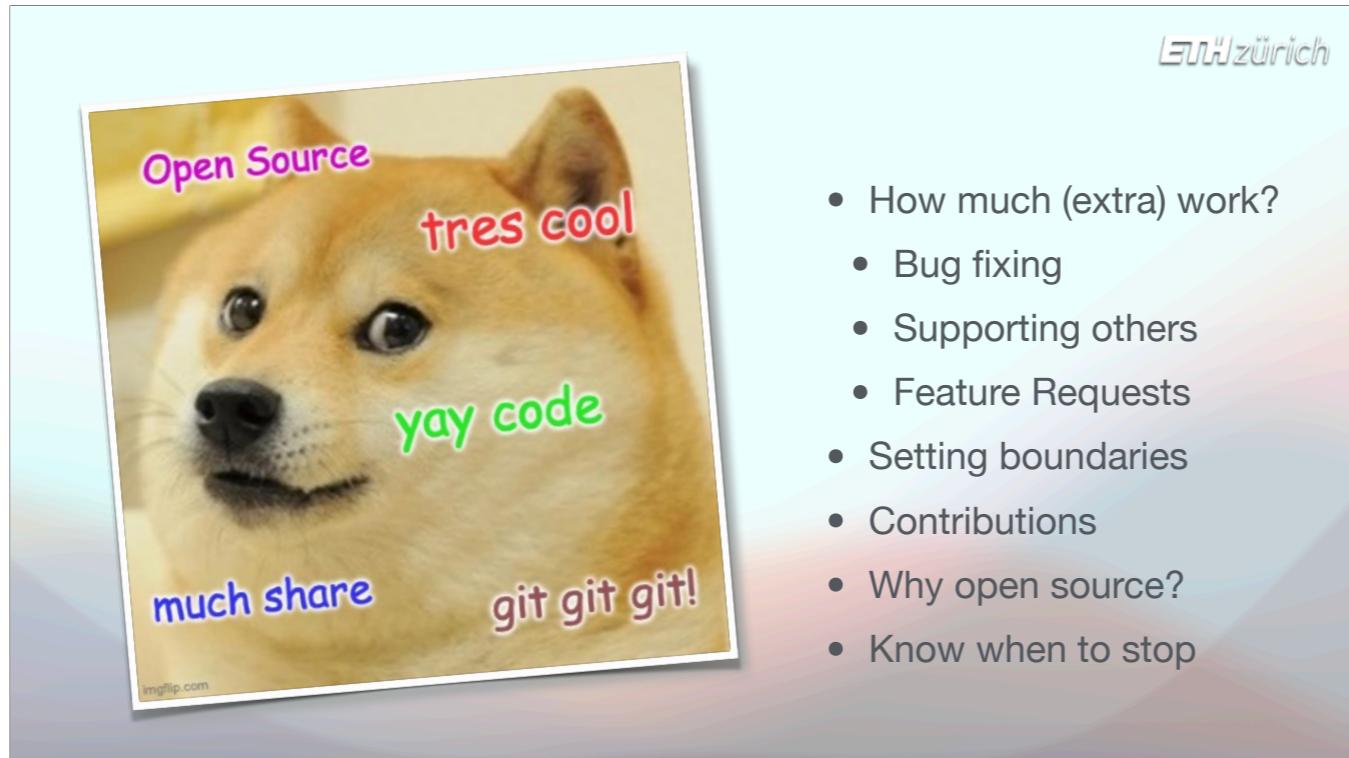
Alright! On to the last part of this talk, which is aimed at MacAdmins who perhaps have not yet embarked on sharing their own code. I'm going to use erase-install as an example of what you might face if you make a script available for use by others, and it starts to get used a lot. I'm sure my experiences are similar to many of our colleagues.





So you've written a cool thing and you think others might find it useful. Your company says it's OK to share it.

But what are the consequences? 🍎 Is it going to generate extra work? Will other people help? Will it grow into something more than originally planned? And what happens if you don't want to continue maintaining it?



- How much (extra) work?
  - Bug fixing
  - Supporting others
  - Feature Requests
  - Setting boundaries
  - Contributions
- Why open source?
- Know when to stop

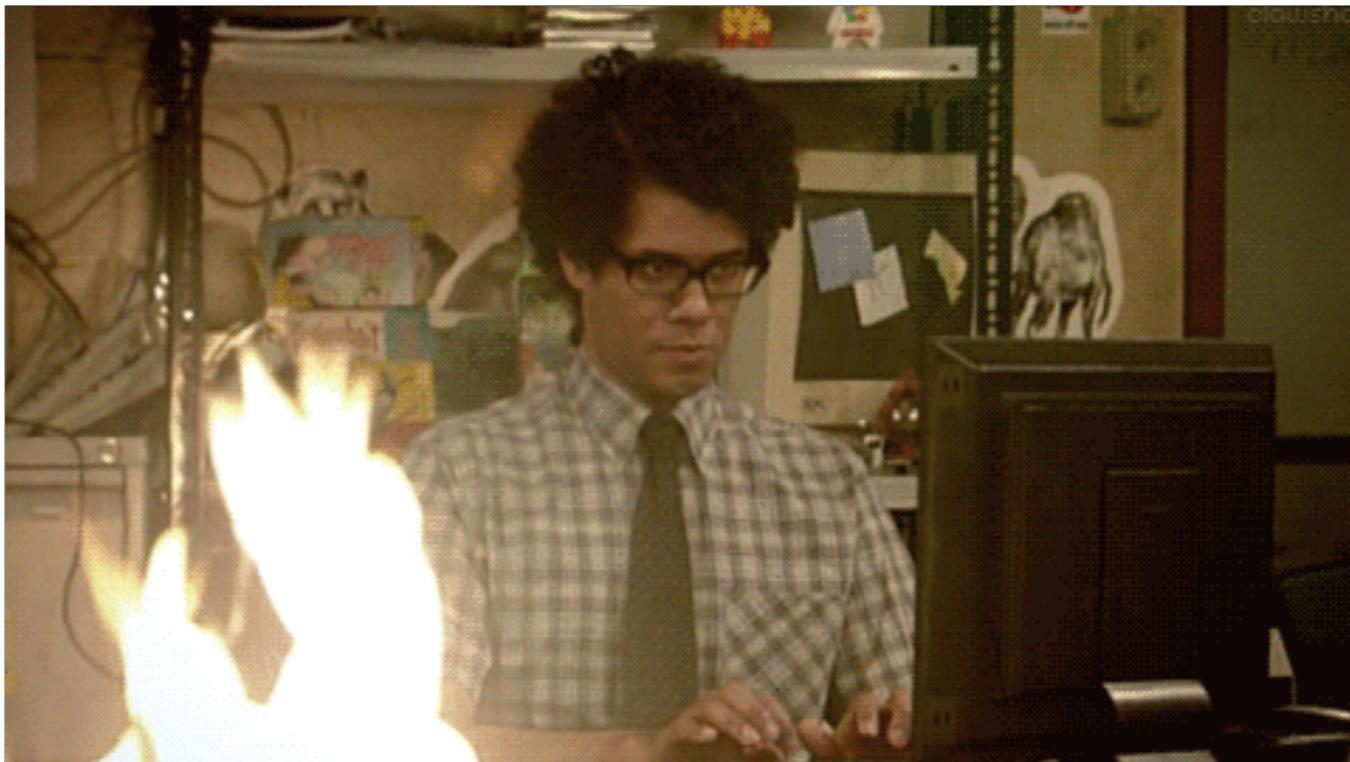


Just putting a script on GitHub doesn't mean any extra work. I have lots of scripts on GitHub that probably nobody else is using.

Others might get used but they're simple and just work, so need little documentation or support.

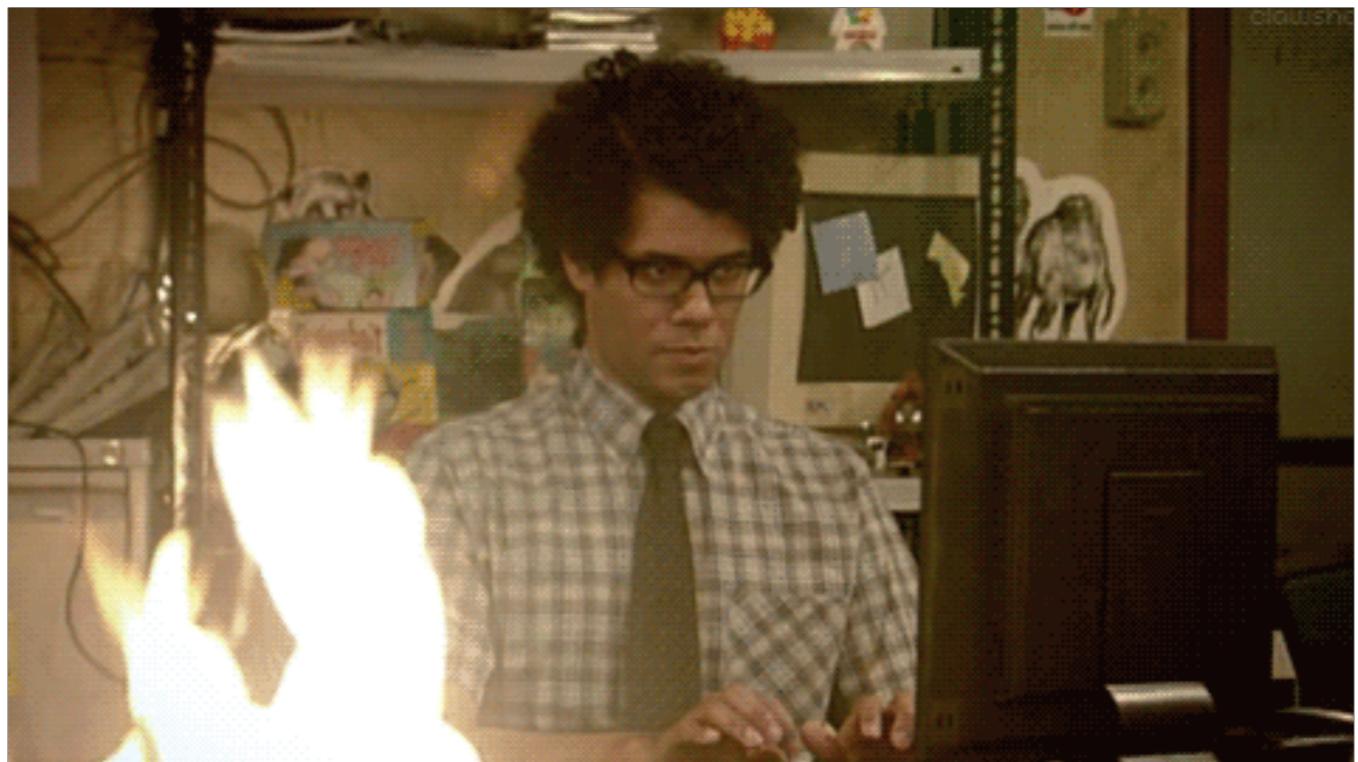
However, erase-install is a bigger, more complex thing that has had to keep evolving. It was clumped together as a quick idea and probably not particularly well written or documented. Nonetheless, it started to get found and used.



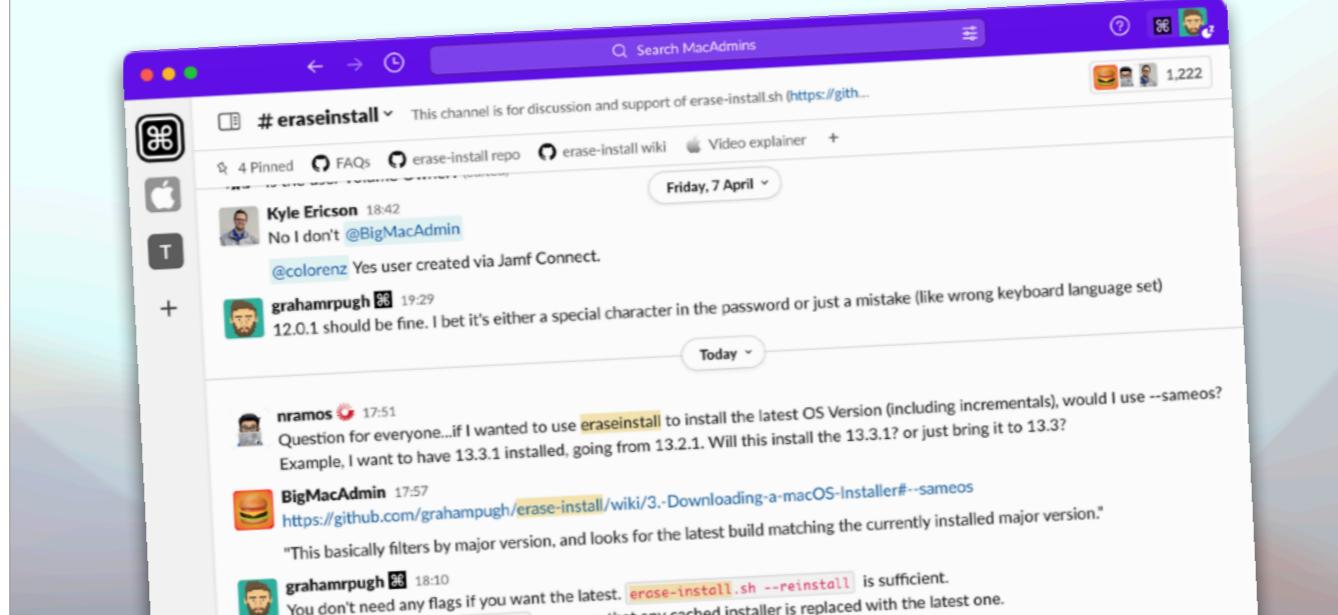


And when people started using it, they started asking about it in the Mac Admins Slack, Jamf Nation, and on the GitHub page.

And of course, just like at work, if people are asking for help with something I built or designed, or reporting things as broken, I can't ignore it.



## Slack channel #eraseinstall



It has been easier to help, once I adopted a channel in the MacAdmins Slack. Monitoring for keywords that relate to the project gave me notifications whenever somebody mentioned it. And I could just point them to the channel, from where there are an increasing number of people who have been helpful in answering questions by newcomers.

The `erase-install` channel has grown to over 1200 members, and averages 120 messages a week. Just to say, it can be good to turn off the notifications in the evening, especially when you live in Europe, since a lot of the traffic comes from North America...



Keeping on top of the questions can be challenging at times. Sometimes it can feel like I need a Helpdesk. [click]

Don't be surprised that people don't always look for documentation before they ask for help. And never underestimate how complete and explicit the documentation needs to be.

Making an FAQ page has been very helpful for me. It's far quicker to just give somebody a link than to type the same answer out over and over.



No, recon won't  
work here.

Security software  
blocking it?

No you cannot use it  
to downgrade!

## GitHub Issues / Discussions

The screenshot shows a list of GitHub issues or pull requests. At the top, there are filters for 'Author', 'Label', 'Projects', 'Milestones', 'Assignee', and 'Sort'. The count shows 11 Open and 168 Closed. Below the filters, there is a list of six items:

- launcher script for jamf ignoring the required version bug
- Error obtaining valid installer. Cannot continue. bug
- [FR] - Set default input field to the Password field when --current-user/-- user is provided enhancement external dependency
- overwrite option not working question
- during reinstall preinstall and post install command fail bug
- ... during the Upgrading portion enhancement

Each item has a small profile picture, a comment icon, and a number indicating the count of comments.

I've also found it to be a good idea to ask people to open a GitHub issue for any sort of bug or question that can't be answered immediately, otherwise it can be too hard to keep on top of it all.

🍎 GitHub Discussions are also useful, because not everyone is in Slack.

# GitHub Issues / Discussions

The screenshot shows the GitHub Issues and Discussions interface. At the top left, there are filters for '11 Open' and '168 Closed' issues. On the right, there are dropdown menus for 'Assignee' and 'Sort'. Below these are two main sections: 'Categories' and 'Discussions'.

**Categories:**

- View all discussions
- General
- Ideas
- Polls
- Q&A
- Show and tell

**Discussions:**

- Need help with timeout error** (Arto254, Mar 9, General, 13 comments)
- Disable OK Button on Utility Window** (JasonJrSmith, last month, Q&A - Unanswered, 2 comments)
- Ignoring Parameters in Jamf Pro** (bfridman-od, Nov 4, 2022, Q&A - Unanswered, 23 comments)
- Where is the GUI** (ooshnoo, Mar 6, General, 5 comments)
- Running 29.0 - Should I be seeing some prompts with the following command?** (Todait, Mar 3, Q&A - Answered, 2 comments)
- Failing for bad password...but not actually** (grahamrpugh.com, 2020/05/05, 1 comment)

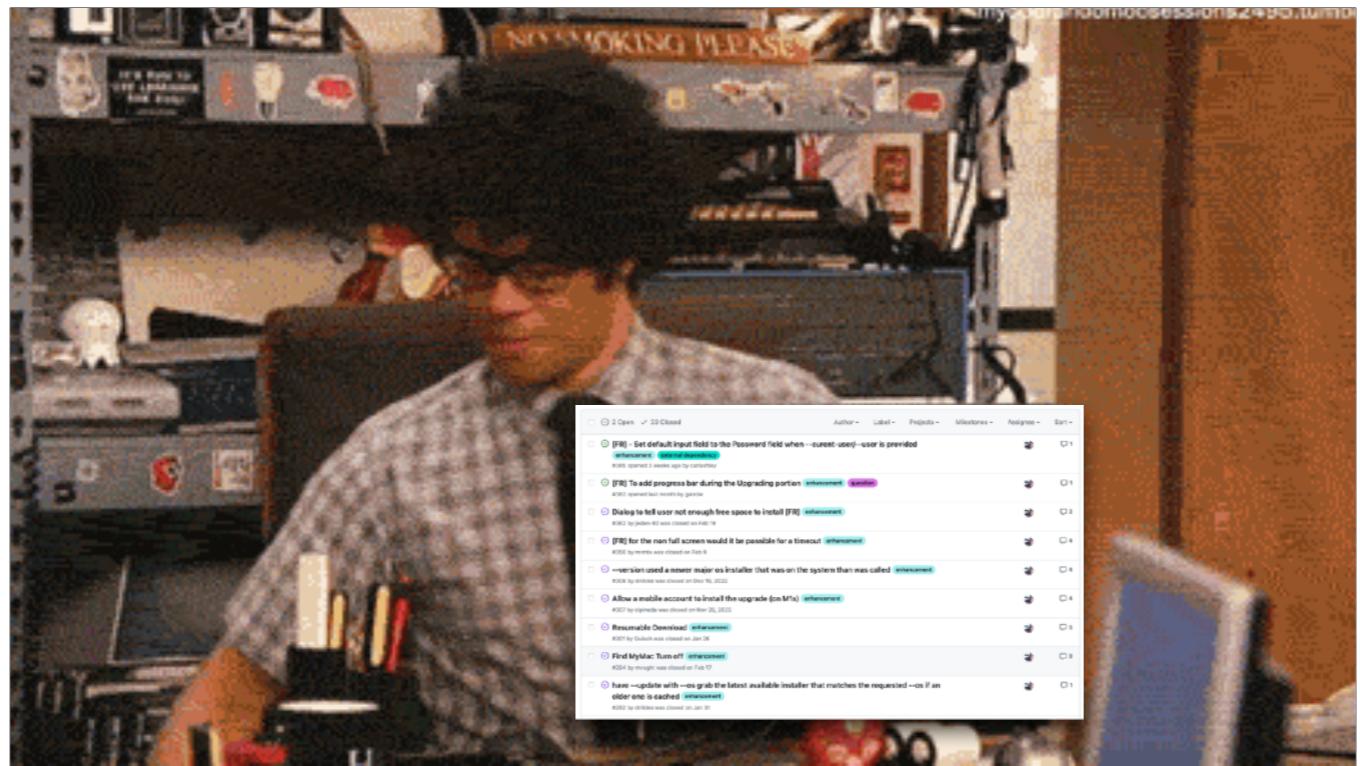
A callout box titled 'Most helpful' provides instructions: 'Be sure to mark someone's comment as an answer if it helps you resolve your question — they deserve the credit!' with a small heart icon.

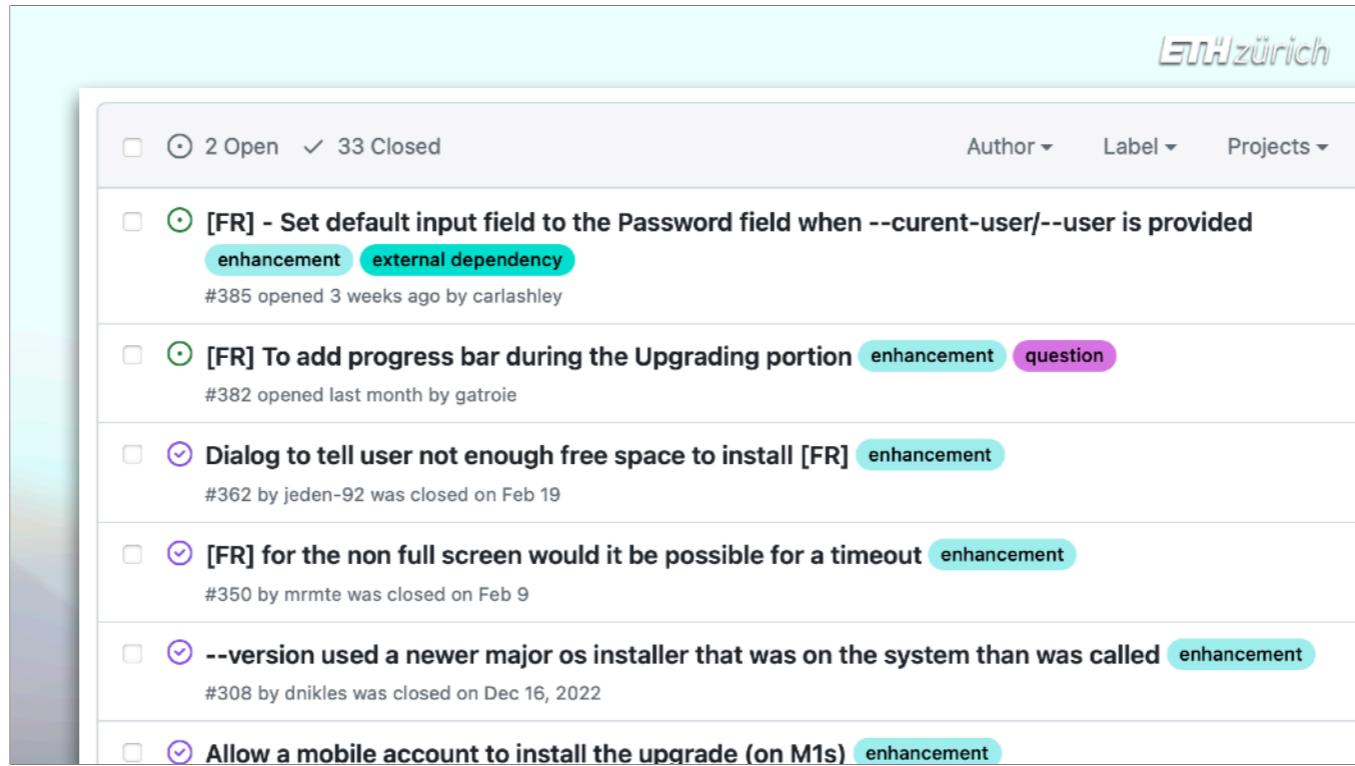


An inevitability about sharing a project is that it might not work in different environments to your own. People also have their own ideas about how the thing should work. Both these aspects become more significant as the usage grows.

If you choose to accept feature requests, they can be the most time-consuming form of support.

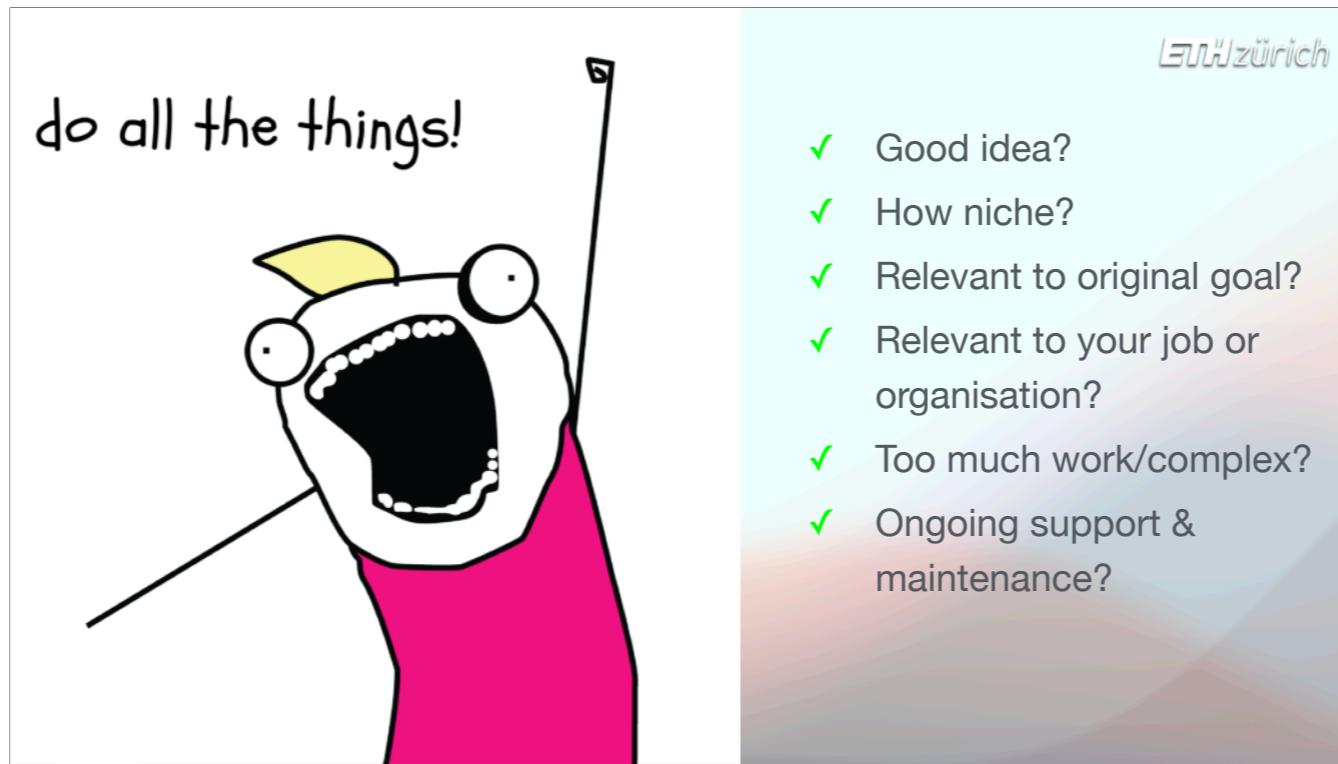






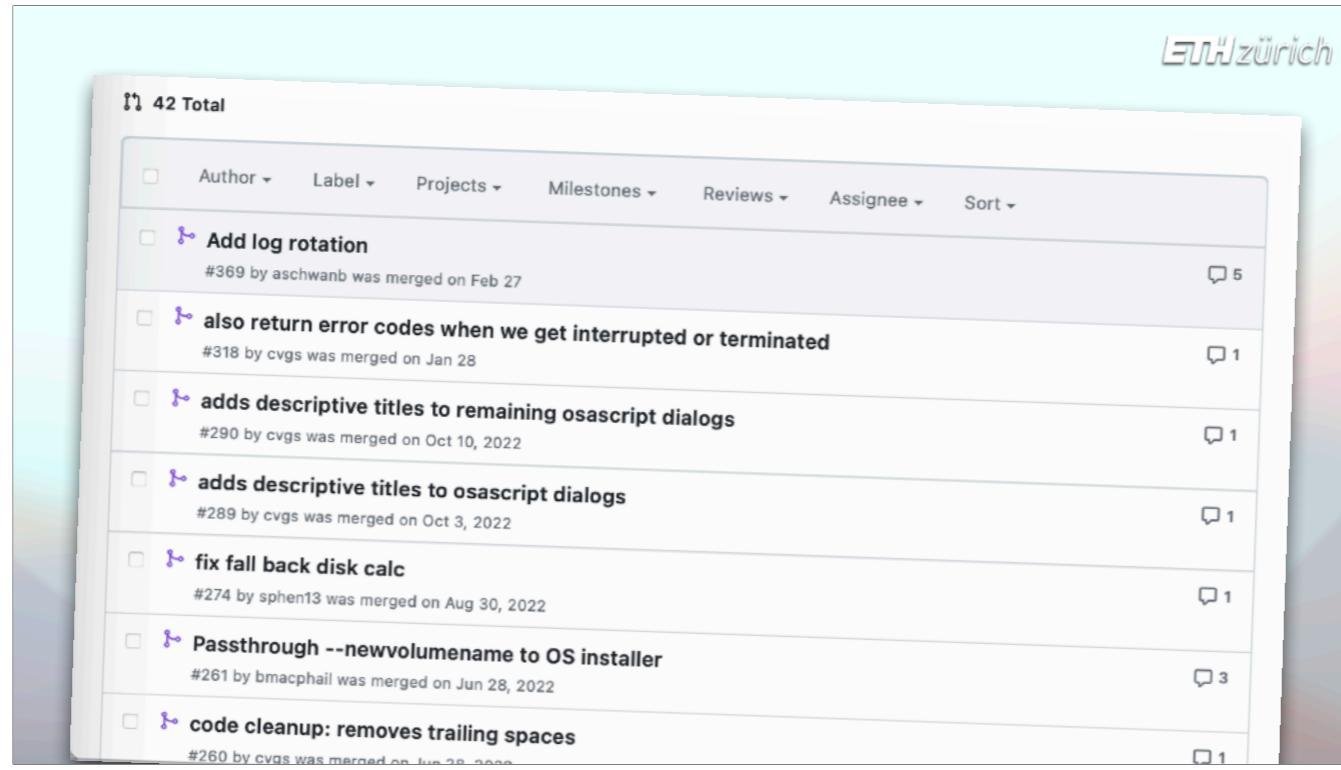
I've found it essential to ask people to record their Feature Request in the GitHub repo so that I could keep a good overview of the things to consider when I next have time to work on the project.

Adding labels to them also helps break them down into things you can go ahead with versus those things that require further input from the requester.



There are important questions I've had to ask myself when getting feature requests:

- Do I agree that the feature request is actually a useful improvement?
- Is it going to help anyone apart from the one person asking for it?
- Is it relevant to my original reason for creating the project?
- As I'm maintaining the project at work, is the request relevant to my organisation's requirements?
- How difficult and complicated is it going to be to design the change, and is it going to make the overall project more difficult to understand?
- Am I interested enough in the feature to support it in the future if it stops working for some reason?



Similar considerations apply to **Pull Requests**. That's where somebody has discovered a bug or has a feature request, but instead of just asking you to fix it, they update the code themselves and only ask you to merge in their changes. Luckily, Git is very clever at figuring out how to safely merge their code into yours.

Pull Requests tend to happen more with shell script code, because far more people understand it than other scripting languages.

I've found it really important to make sure I understood the code changes before accepting and merging the pull request. 🍎. If you don't understand the code, you're just building problems into your project that may come to haunt you later...

42 Total

	Author	Label	Projects	Milestones	Reviews	Assignee	Sort
<input type="checkbox"/>	Add log rotation	#369 by aschwanb was merged on Feb 27					
<input type="checkbox"/>	also return error codes when we get interrupted or terminated	#318 by cvgs was merged on Jan 28					
<input type="checkbox"/>	adds descriptive titles to remaining osascript dialogs	#290 by cvgs was merged on Oct 10, 2022					
<input type="checkbox"/>	adds descriptive titles to osascript dialogs	#289 by cvgs was merged on Oct 3, 2022					
<input type="checkbox"/>	fix fall back disk calc	#274 by sphen13 was merged on Aug 30, 2022					
<input type="checkbox"/>	Passthrough --newvolumename to OS ins	#261 by bmacphail was merged on Jun 28, 2022					
<input type="checkbox"/>	code cleanup: removes trailing spaces	#260 by cvgs was merged on Jun 28, 2022					



I'll just put this over here with the rest of the fire

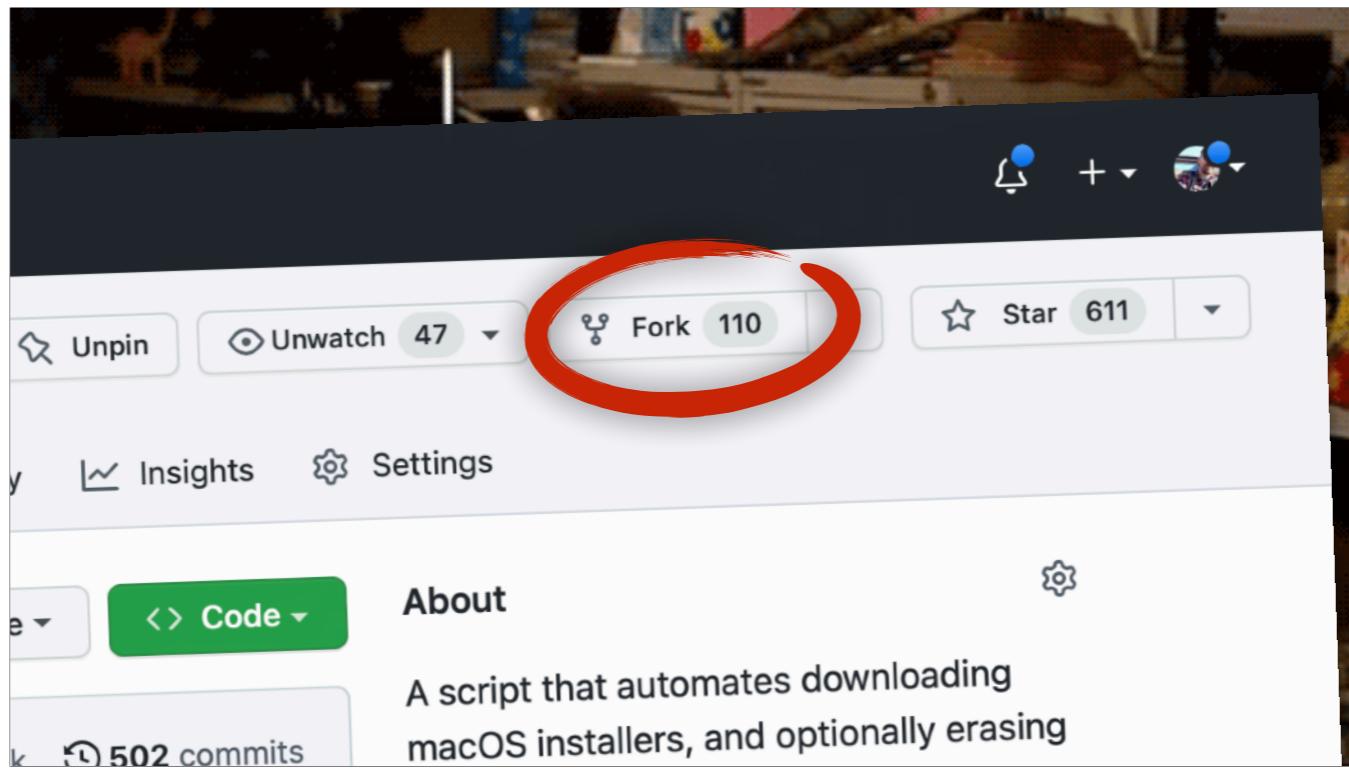


Also perhaps it's obvious, but – you don't have to accept pull requests! If you don't want to accept it, that person can still maintain a fork of the project with their changes in it – they're still benefitting from your work, and they will be able to merge any changes you make in the future into their own fork.

apple Plenty of people are using their own fork of erase-install.



**Just WALK AWAY**



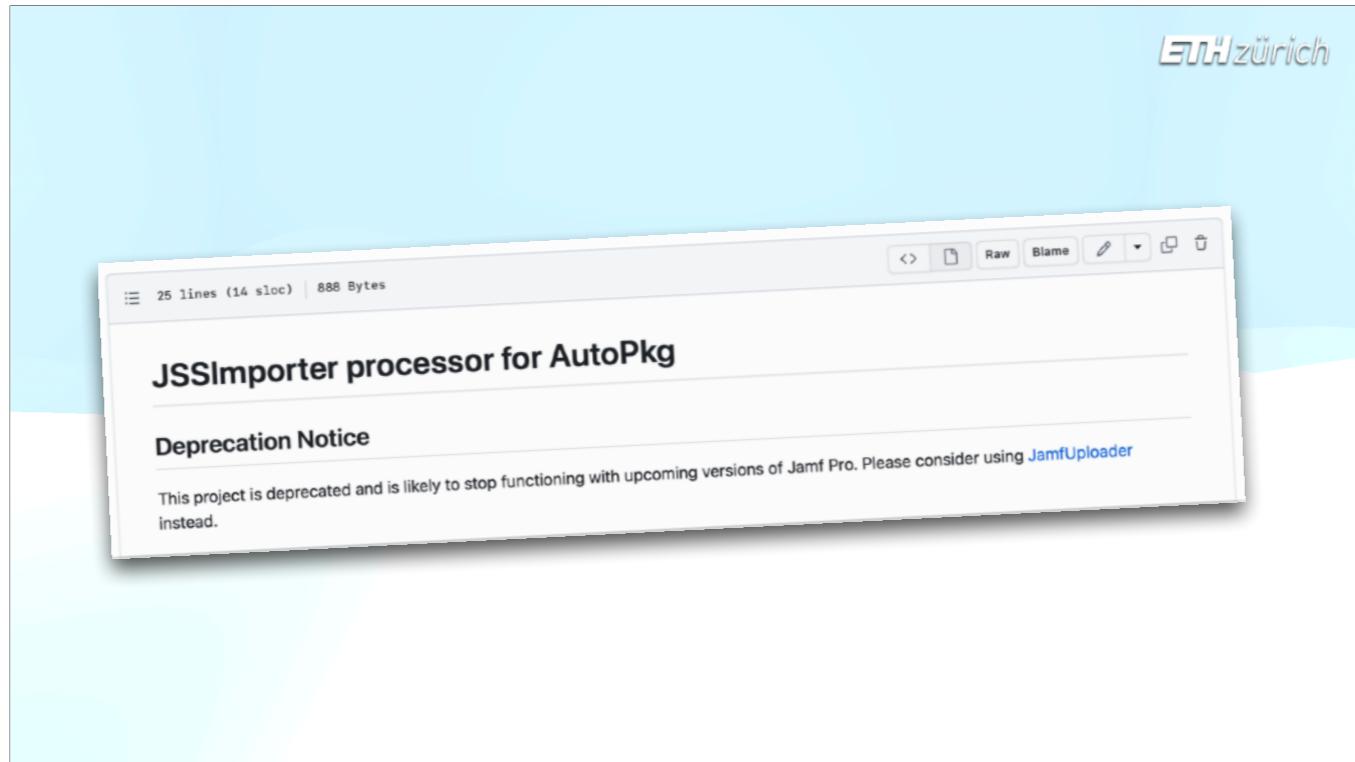


The same is true if you decide that you cannot or don't want to continue maintaining the project.

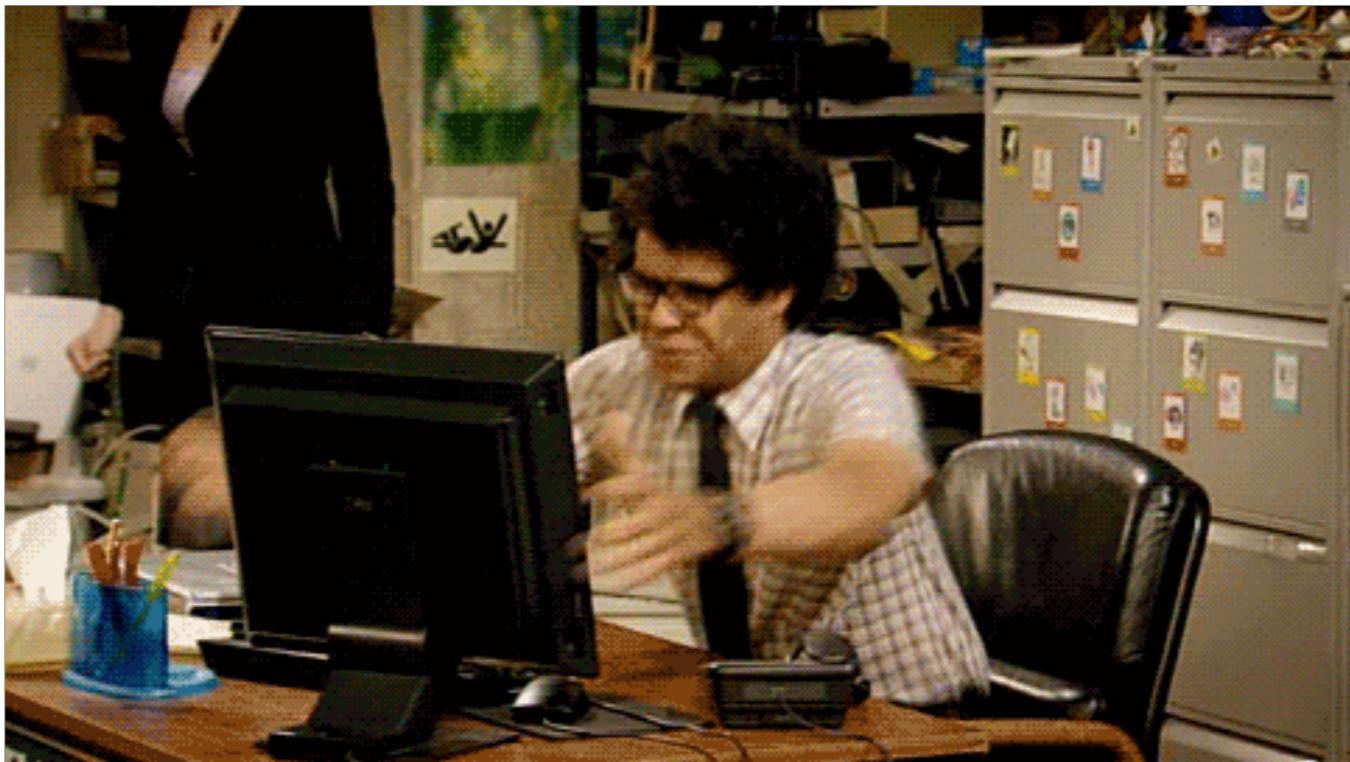
🍎 There are all sorts of legitimate reasons why you might want to stop maintaining some open source project.

Again, people can make their own fork and maintain it themselves, if they still have a use for it when you don't. Somebody can take the project over if they think it's still useful.

- No longer useful in your job or organisation
- You change jobs
- You just don't have time
- You can't fix it
- Somebody wrote a better tool for the job
- Sherlocked by Apple



Just make it clear in the README of the repo that your version is no longer maintained, 🍎 and people will slowly move on.



## If you open source your code...

To summarise:

- expect more work than if you had not published the thing
- You can mitigate that a bit by taking the time from the beginning to create good documentation
- Make a Slack channel if you're starting to get questions about it.
- Create FAQs as you answer questions – this can save you repeating yourself over and over.
- Keep on top of any problems by encouraging people to record it in your GitHub repo.
- Keep the code simple and readable, even if it might seem less efficient. This increases the chances that people will send you pull requests and answer their own questions.
- This is a really helpful community in my experience. You can expect a lot of thanks and a lot of help. So don't be shy in sharing your code!

## If you open source your code...

- Expect more work

## If you open source your code...

- Expect more work
- Document early and in detail

## If you open source your code...

- Expect more work
- Document early and in detail
- Make a Slack channel

## If you open source your code...

- Expect more work
- Document early and in detail
- Make a Slack channel
- Create FAQs

## If you open source your code...

- Expect more work
- Document early and in detail
- Make a Slack channel
- Create FAQs
- Encourage Issues, FRs, PRs in your repo

## If you open source your code...

- Expect more work
- Document early and in detail
- Make a Slack channel
- Create FAQs
- Encourage Issues, FRs, PRs in your repo
- Make your code readable (if you want contributions)

## If you open source your code...

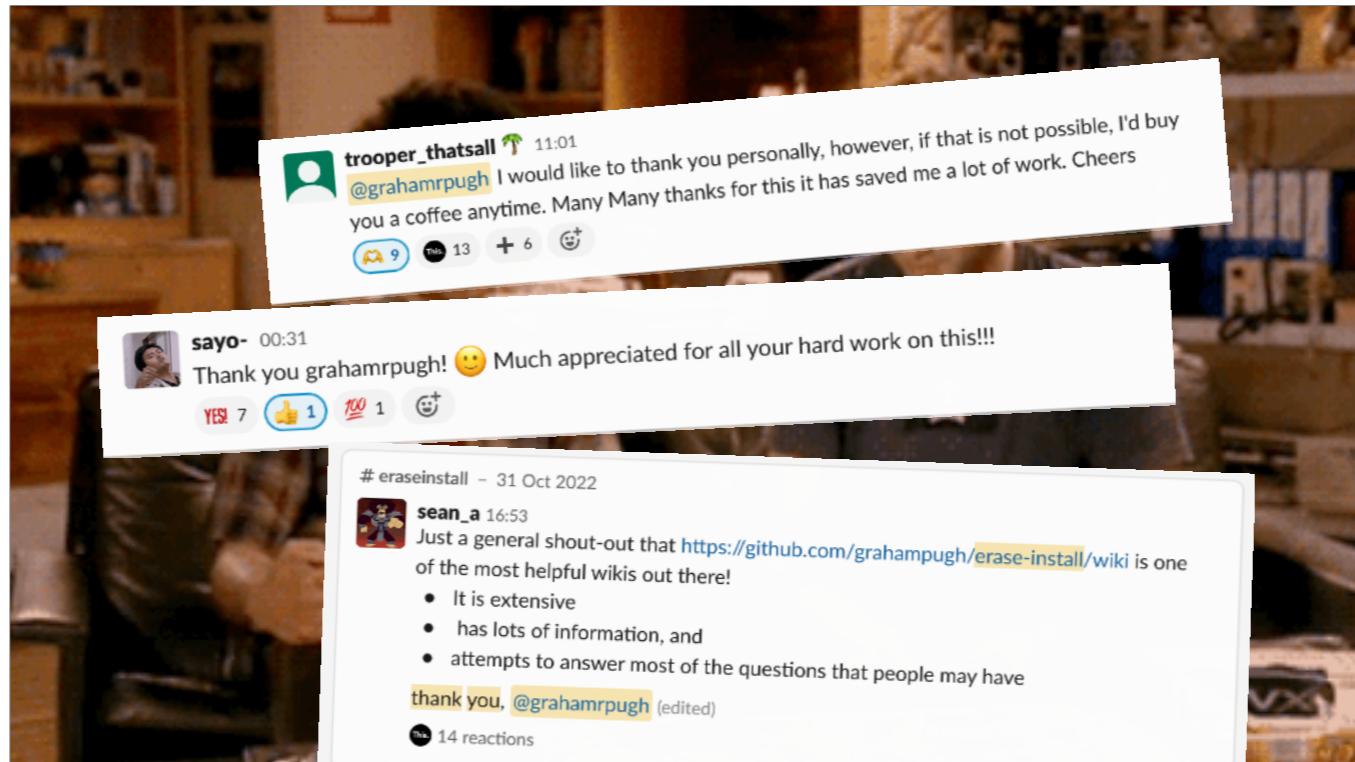
- Expect more work
- Document early and in detail
- Make a Slack channel
- Create FAQs
- Encourage Issues, FRs, PRs in your repo
- Make your code readable (if you want contributions)
- **Expect great feedback and support**

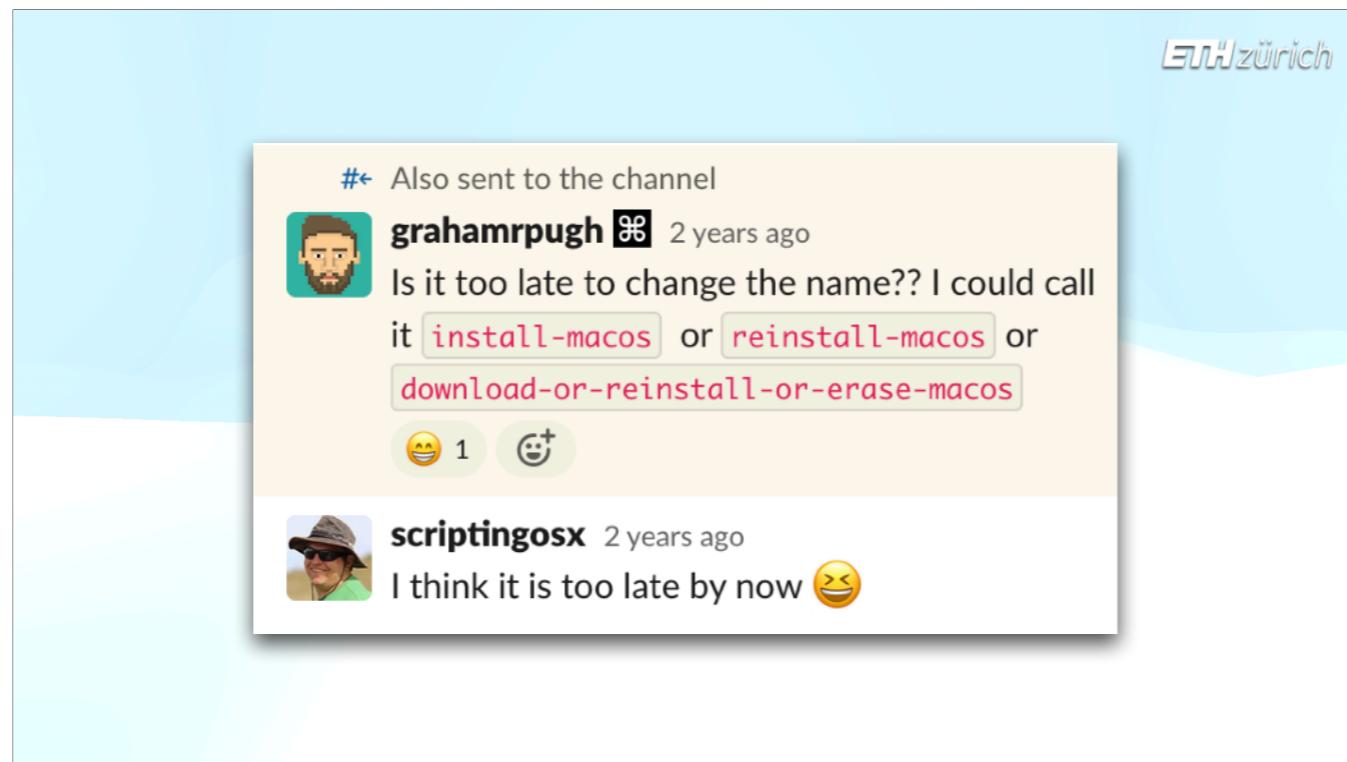


That last point is the main reason I recommend open sourcing your code.

🍎 I could never have anticipated the amount of thanks I've received from people using erase-install.

But also, because I've received so much help from this community, it's only fair to give a little back, whether that's in the form of sharing code or answering questions.





One last thing. Unlike me, take some time to think of a good name for your project. Preferably one that doesn't mean anything. Something like Munki, Kmart, Mist, Superman, Sal. A descriptive name can quickly become a bad idea if you add things!

download-and-or-erase-and-or-reinstall-  
and-or-upgrade-macos.sh



That's it from me, thank you for listening!

*Edu*zürich





Thank You!