

Munki and the Machine

Code Assist's Impact on Open Source

Greg Neagle, October 2026

Previously at MacSysAdmin...

Oct 2023

Default (GPT-3.5)

GN

What is the current version of Munki?



As of my last knowledge update in September 2021, the latest stable version of Munki was 5.2.0. However, software versions can change quickly, and new releases may have come out since then. To find the most current version of Munki, I recommend visiting the official Munki GitHub repository or the project's website, or checking with the Munki community for the latest information. Keep in mind that the version number may have changed since my last update.



Munki 6.3.3 Official Release

 gregneagle released this Aug 2, 2023  v6.3.3  d3c9eb4

 Regenerate

Oct 2023

> What is the current version of Munki?

The current version of Munki is Munki 3.

Munki 6.3.3 Official Release

 gregneagle released this Aug 2, 2023 v6.3.3 d3c9eb4

> That's not correct.

Apologies for the incorrect information earlier. The latest version of Munki is Munki 6.0 or later.

> That's still not right.

I apologize for the confusion. It seems that I am unable to provide the correct information about the latest version of Munki at the moment. I recommend visiting the official Munki website or checking with the Munki community for the most up-to-date information on the latest version.

Oct 2023

> What is the current version of Munki?

According to the information provided in the context, the current version of Munki is 6.3.3. You can find the latest version of Munki on their GitHub repository releases page: <<https://github.com/munki/munki/releases>>.

Can we make a custom Munki chatbot available?

Oct 2023



Too expensive

Response quality not good enough yet



Too complex to
release as a local
tool

MENU



In October 1-4

MacSysAdmin

Everyone here has the sense that right now is one of those moments when we are influencing the future.
— Steve Jobs

We have started planning for the nineteenth chapter of

MENU



In the first week of Oct

MacSysAdmin

"We're here to put a dent in the universe.
Otherwise, why else even be here?"
— Steve Jobs

In 2025 we will celebrate the MacSysAdmin Conference's

MENU



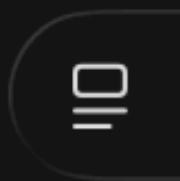
In the first week of October 2026 we proudly present

MacSysAdmin Conference 2026

"If today were the last day of my life, would I
wanna do, what I am about to do today?"
— Steve Jobs

"...it was twenty years ago [today]", to paraphrase The

"...Plus ça change, plus c'est la meme chose", as the
Canadian prog rockers Rush once sung, *The more it
changes, the more it's the same*. With that said, we still feel
that there is a need for people to meet in person and be
able to talk and exchange experiences with their peers in a



What is the current version of Munki?

[Releases](#) / v7.3.0

Munki 7.3 Official Release

Latest



gregneagle released this last month



v7.3.0



28d92fe

This is the official release of Munki 7.3.0, a feature-add and bug-fix release

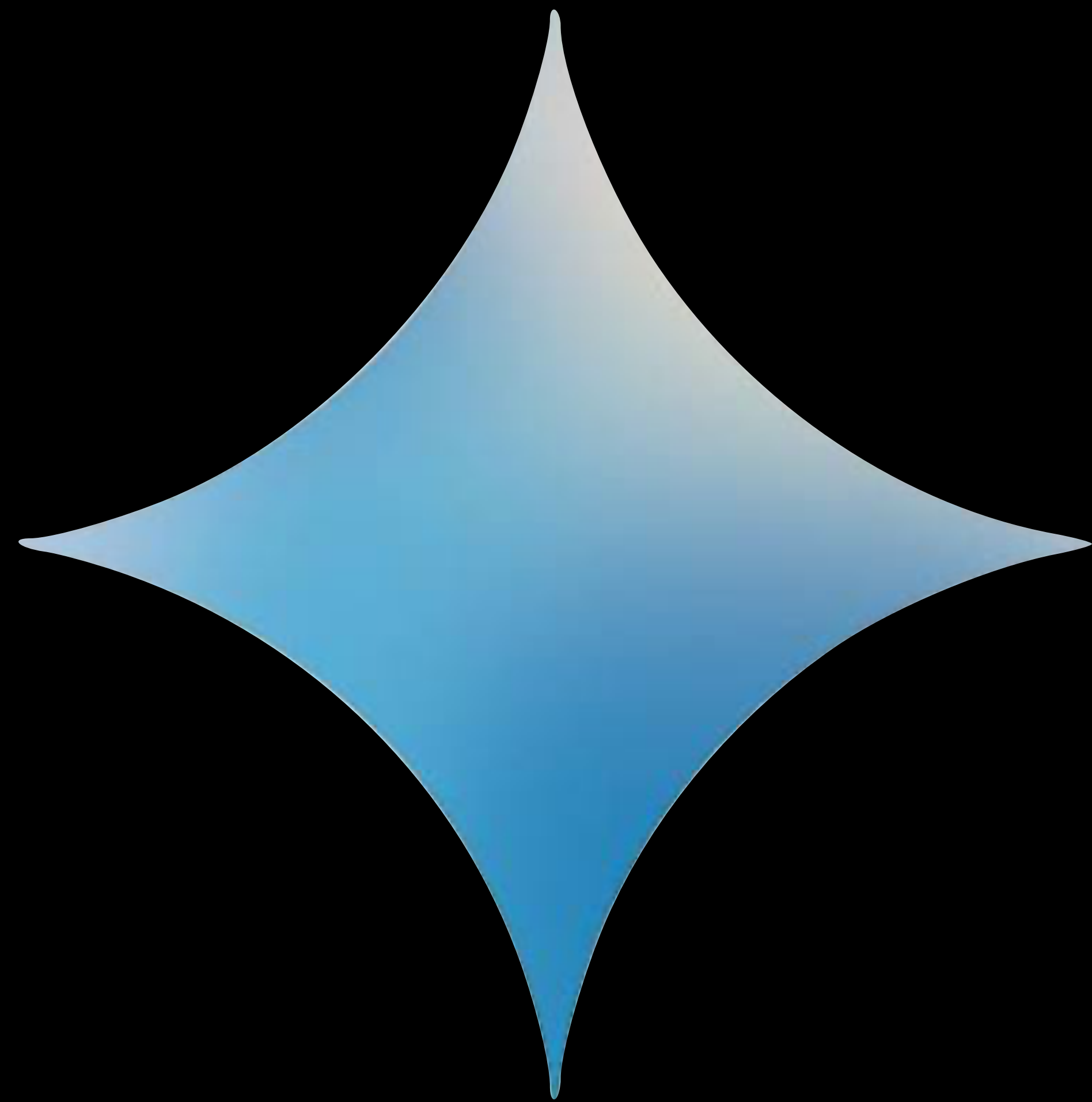
Build info

- This release was built with Xcode 26.6 (17F113) on macOS 26.5.2 (25F8) has been signed and notarized by MacAdmins Open Source.
- Distribution package version: 7.3.0.5808

AI code assist









Concerns



BUSINESS
INSIDER

A top economist says
more layoffs are likely
as the job market nears
a tipping point with AI

ORACLE

ORACLE CUT
21,000 JOBS IN
12 MONTHS

"The adoption and deployment of
AI technologies across our
operations have resulted, and
may continue to result, in
reductions to our workforce."

Source: Oracle regulatory filing

Bloomberg Brief

ORACLE CUT 21,000 JOBS IN 12 MONTHS

“

AI-driven
displacement
could raise the
unemployment rate
slightly in 2026,
with upside risks
from faster
adoption and larger
displacement.

PIERFRANCESCO MEI
GOLDMAN SACHS ECONOMIST

PHOTO: GETTY IMAGES



yahoo! finance

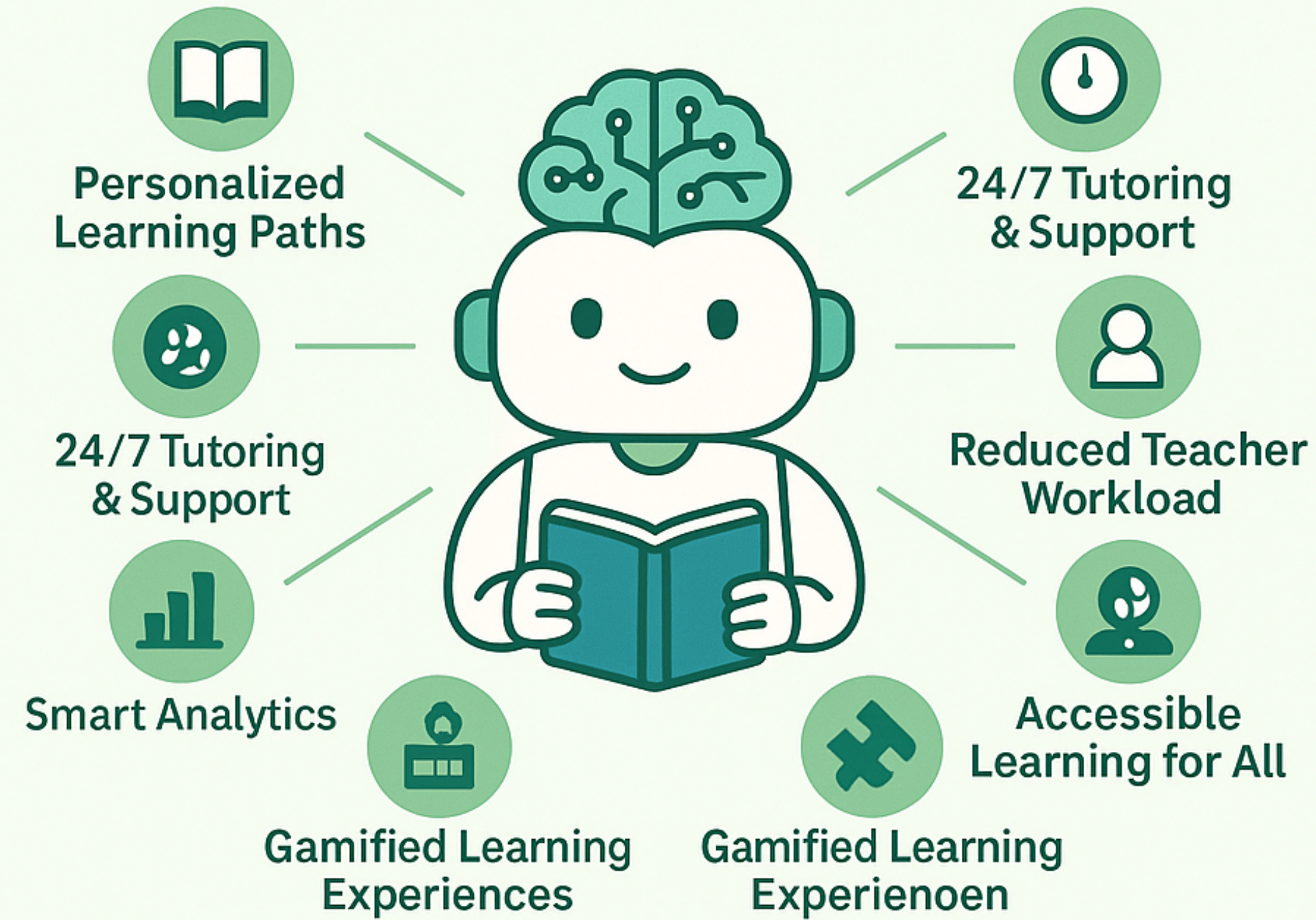


Goodbye, \$165,000 Tech Jobs. Student Coders Seek Work at Chipotle.

As companies like Amazon and Microsoft lay off workers and embrace A.I. coding tools, computer science graduates say they're struggling to land tech jobs.



Positive Effects of AI in Education

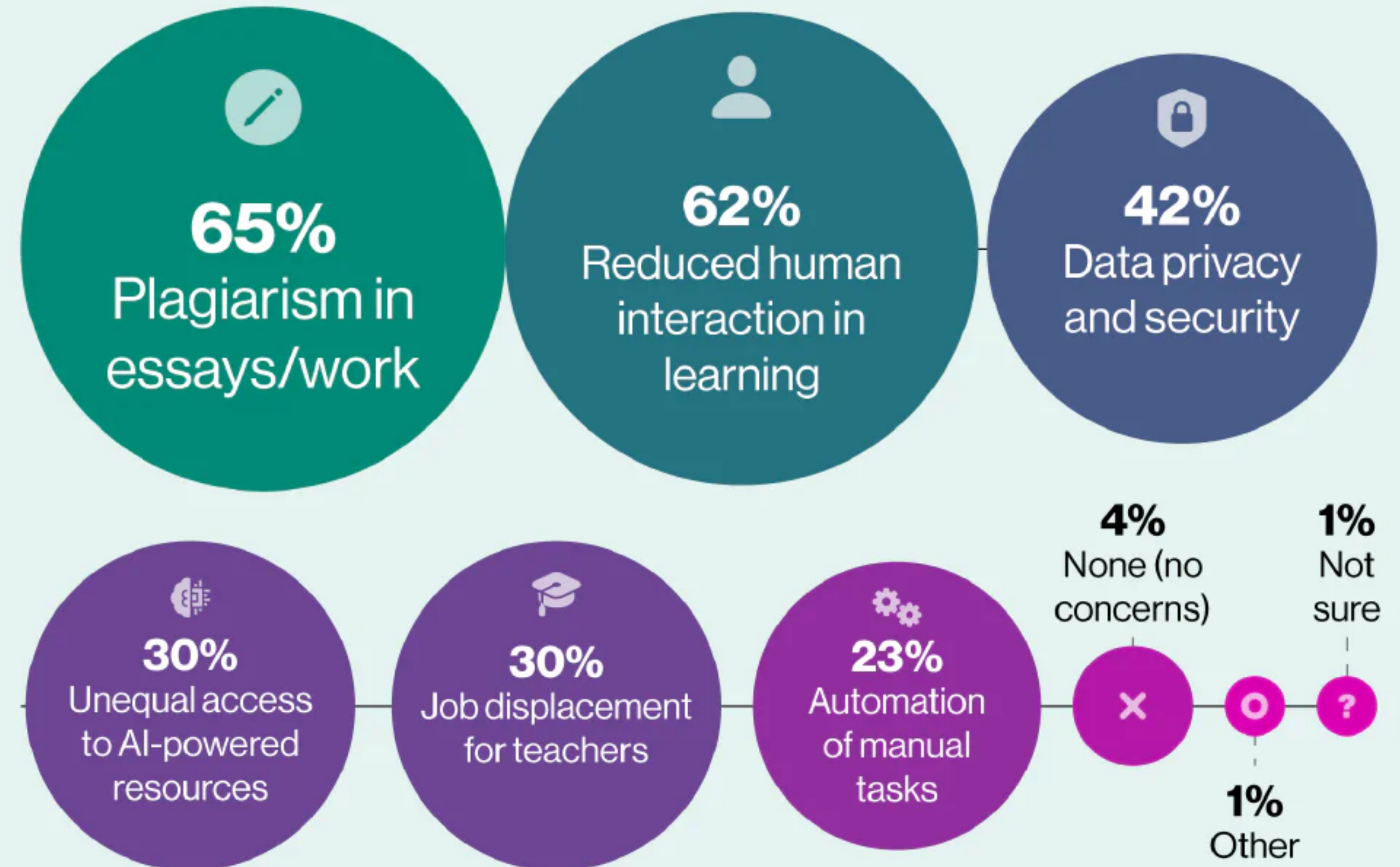


AI isn't replacing teachers. It's empowering them – and students – to learn smarter, not harder.

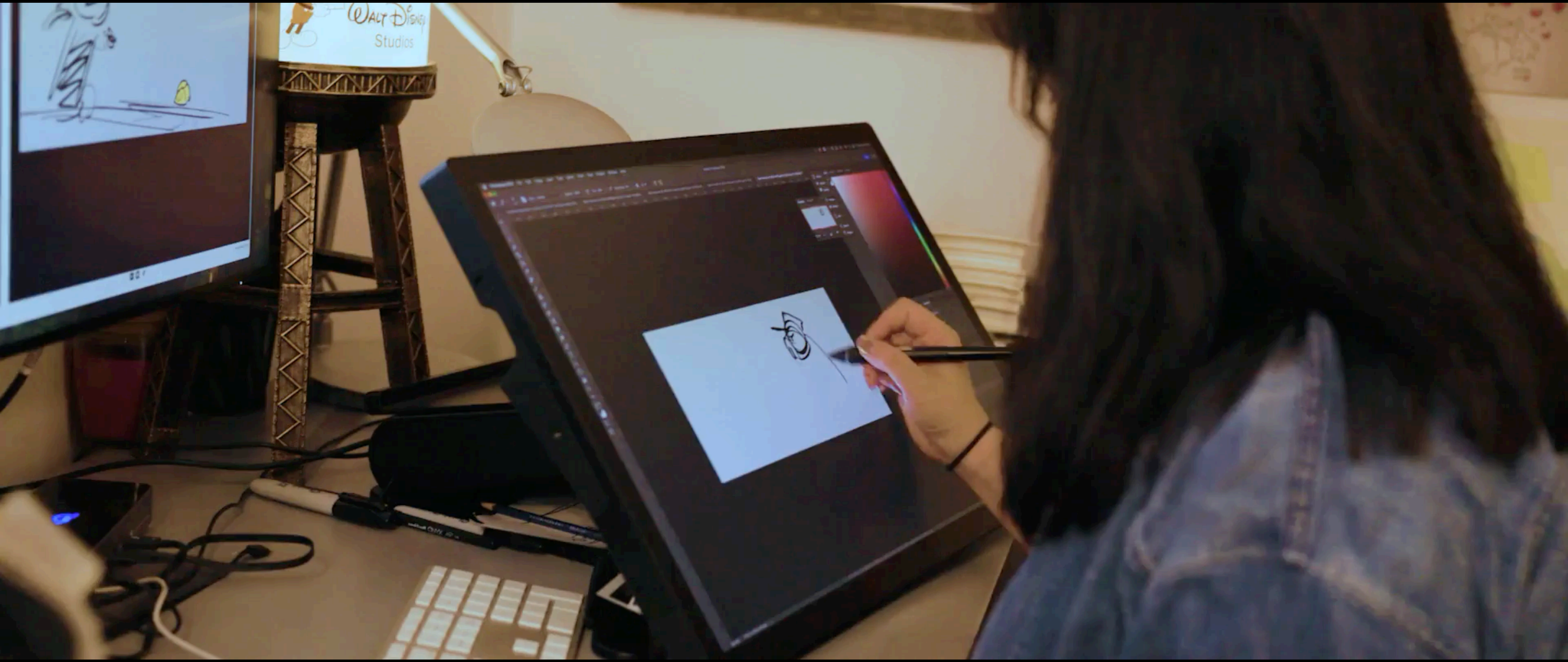
AI Isn't replacing teachers. It's empowering them – and students – to

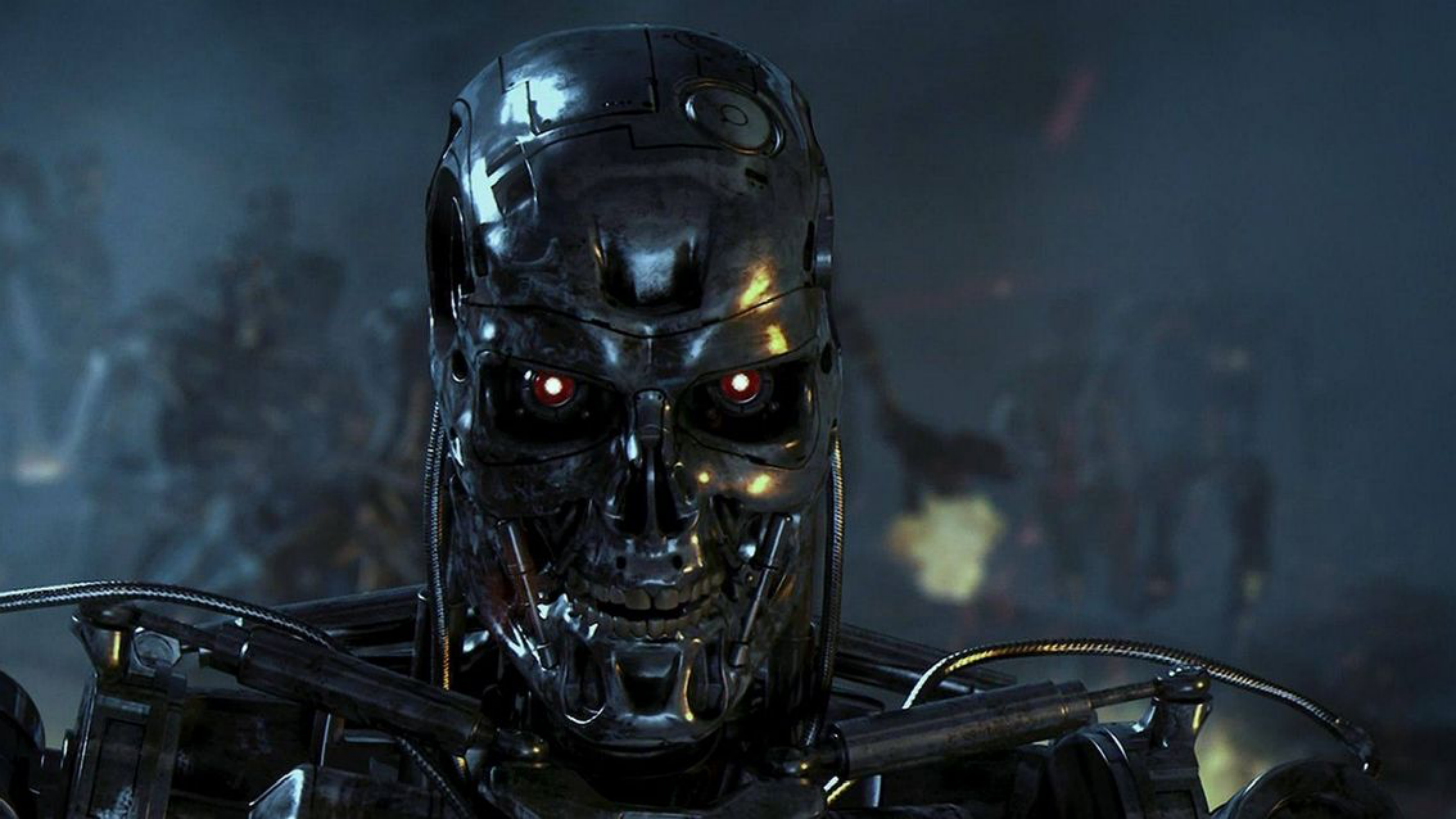
A+

Percentage of teachers (%) Concern



Source: Forbes





But

Generative AI / LLMs







TAEKWONDO SUMMER CAMP 2026



KICK INTO
SUMMER!

JULY 13, 2026 – JULY 17, 2026

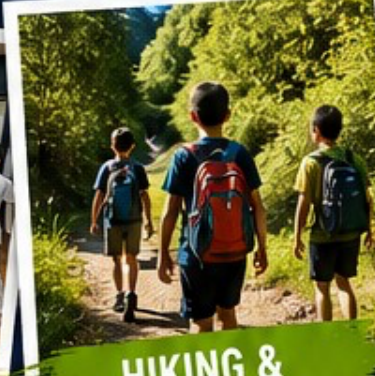
MONDAY – FRIDAY | 9AM – 3PM



SWIMMING



TAEKWONDO



HIKING &
OUTDOOR FUN



LEARNING
ACTIVITIES



NEW FRIENDS

FUN. FITNESS. FRIENDSHIP. GROWTH.
ONE UNFORGETTABLE SUMMER!



FUN
TAEKWONDO
CLASSES



EXCITING
WORKOUTS



CHARACTER
BUILDING



LEARN TO
READ & WRITE
KOREAN



BUILD CONFIDENCE
& LASTING
FRIENDSHIPS



EARLY REGISTRATION
BEFORE JUNE 27, 2026

\$400
PER WEEK



REGULAR REGISTRATION
AFTER JUNE 27, 2026

\$500
PER WEEK



INCLUDES ALL SUMMER CAMP ACTIVITIES
& 1 SUMMER CAMP T-SHIRT!



SPACES ARE LIMITED!
REGISTER TODAY!

SCAN TO
REGISTER
TODAY!



CAMP HIGHLIGHTS

- ✓ TAEKWONDO TRAINING
- ✓ PHYSICAL WORKOUTS
- ✓ SWIMMING
- ✓ HIKING & OUTDOOR ADVENTURES
- ✓ VISIT MUSEUMS
- ✓ LOCAL PARK ACTIVITIES
- ✓ VISIT LOCAL LIBRARY
- ✓ LEARNING ACTIVITIES
- ✓ BUILD SELF RESPECT & RESPECT FOR OTHERS
- ✓ AND SO MUCH MORE!

WONDOO

MMMER

MP



KICK INTO

MMMER!

AI code assist

We shouldn't.

Fortunately, we don't have to.

› Help me write a shell script to return the major macOS version running on the current Mac.

Thought for 2s (ctrl+o to expand)

- Here's a simple shell script for that:

```
#!/bin/bash
sw_vers -productVersion | cut -d. -f1
```

`sw_vers -productVersion` returns something like `15.5.1`, and `cut -d. -f1` extracts the first field (major version), giving you `15`.

If you want it as a variable in a larger script:

```
macos_major=$(sw_vers -productVersion | cut -d. -f1)
```

* Worked for 3s


```
bash-3.2$ sw_vers -productVersion | cut -d. -f1
```

27



MacBook Pro

14-inch, 2023

Chip Apple M2 Pro

Memory 32 GB

Serial number C1W1M3QWTV

macOS Golden Gate 27.0.1

[More Info...](#)

[Regulatory Certification](#)

™ and © 1983-2026 Apple Inc.
All Rights Reserved.

```
bash-3.2$ sw_vers -productVersion | cut -d. -f1
```

10



munki › shared › PosixSpawn.swift › No Selection

```
19
20 // Based on https://stackoverflow.com/a/69143985
21
22 import Darwin
23
24 class PosixSpawnError: MunkiError {}
25
26 /// Uses posix_spawn to run a subprocess. munkiimport uses this function to run a CLI editor
27 /// (like vi) if needed. Running an interactive process using Foundation's NSTask/Process doesn't
28 /// work properly because it doesn't properly connect stdin and stdout to the terminal
29 public func posixSpawn(_ executablePath: String, _ arguments: String...) throws {
30     var pid: pid_t = 0
31     guard let path = executablePath.withCString(strdup) else {
32         throw PosixSpawnError("Could not spawn \(executablePath): could not create cstring from executablePath")
33     }
34     let args = try arguments.map {
35         if let arg = $0.withCString(strdup) {
36             return arg
37         } else {
38             throw PosixSpawnError("Could not spawn \(executablePath): could not create cstrings from arguments")
39         }
40     }
41     defer {
42         ([path] + args).forEach { free($0) }
43     }
44     if posix_spawn(&pid, path, nil, nil, [path] + args + [nil], environ) < 0 {
45         throw PosixSpawnError("Could not spawn \(executablePath): error \(errno)")
46     }
47     if waitpid(pid, nil, 0) < 0 {
48         throw PosixSpawnError("Could not spawn \(executablePath): error \(errno)")
49     }
50 }
```

> Help me write a Swift function for a command-line tool that uses `posix_spawn` to launch another command line tool like a text editor.

Thought for 12s (ctrl+o to expand)

- Here's a clean Swift function using `posix_spawn`:

```
import Foundation
```

```
enum LaunchError: Error {  
    case spawnFailed(Int32)  
}
```

```
/// Launches an executable using posix_spawn and waits for it to exit.
```

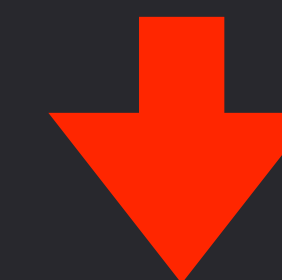
```
/// Returns the child process's exit status.
```

```
@discardableResult
```

```
func launch(_ path: String, args: [String] = []) throws -> Int32 {  
    // argv must be [path] + args + [nil], as C strings we own  
    var argv = ([path] + args).map { strdup($0) }  
    argv.append(nil)  
    defer { argv.forEach { free($0) } }
```



```
9 @discardableResult
10 func launch(_ path: String, args: [String] = []) throws -> Int32 {
11     // argv must be [path] + args + [nil], as C strings we own
12     var argv = ([path] + args).map { strdup($0) }
13     argv.append(nil)
14     defer { argv.forEach { free($0) } }
15
16     var fileActions: posix_spawn_file_actions_t?
17     posix_spawn_file_actions_init(&fileActions)
18     defer { posix_spawn_file_actions_destroy(&fileActions) }
19
20     var attrs: posix_spawnattr_t?
21     posix_spawnattr_init(&attrs)
22     defer { posix_spawnattr_destroy(&attrs) }
23
24     var pid: pid_t = 0
25     let rc = posix_spawn(&pid, path, &fileActions, &attrs, &argv, environ)
26     guard rc == 0 else { throw LaunchError.spawnFailed(rc) }
27
28     var status: Int32 = 0
29     waitpid(pid, &status, 0)
30     return WEXITSTATUS(status)
31 }
```



› When I paste this into Xcode I get an error: "Cannot find 'WEXITSTATUS' in scope"

Thought for 3s (ctrl+o to expand)

- `WEXITSTATUS` is a C macro, and Swift doesn't import C macros. Replace it with the equivalent expression directly:

```
return (status >> 8) & 0xff
```

* Sautéed for 3s

Why?

Open source challenges

- Code complexity over time
- Not enough people familiar with the code
- Lots of ideas, few resources
- Testing

Help!

A New Hope

Feature request - offer to quit and relaunch apps when updating #1157

✓ Closed

 ofirgalcon opened on Aug 31, 2022

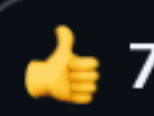
⋮

Managed Software Centre (like the Apple AppStore) should ideally offer users the option to quit, update and relaunch apps automatically.

This is an old request of mine that was discussed in the past, but I never posted it here.

Create sub-issue ▾



 7

Feature request - offer to quit and relaunch apps when updating #1157

✓ Closed



ofirgalcon opened on Aug 31, 2022



Managed Software Centre (like the Apple AppStore) should ideally offer users the option to quit, update and relaunch apps automatically.

This is an old request of mine that was discussed in the past, but I never posted it here.

Create sub-issue



7



gregneagle on Aug 31, 2022

Member



It's not something I'm likely to work on in the near future. If someone else would like to tackle it, I'd be happy to review any work.





Mike Solin <mike@mikesolin.com>

to munki-dev@googlegroups.com

Nov 8, 2016, 10:13:49 AM



It'd be really cool if there was a way to assist the user with quitting these applications, then relaunching them after the update process is finished. The App Store does that now, though I'm not sure how.

Mike

Proposal: Add "Close apps and update" option to MSC 116 views



jordan...@gmail.com <jordancalhoun@gmail.com>

to munki-dev

Dec 4, 2025, 9:17:30 AM



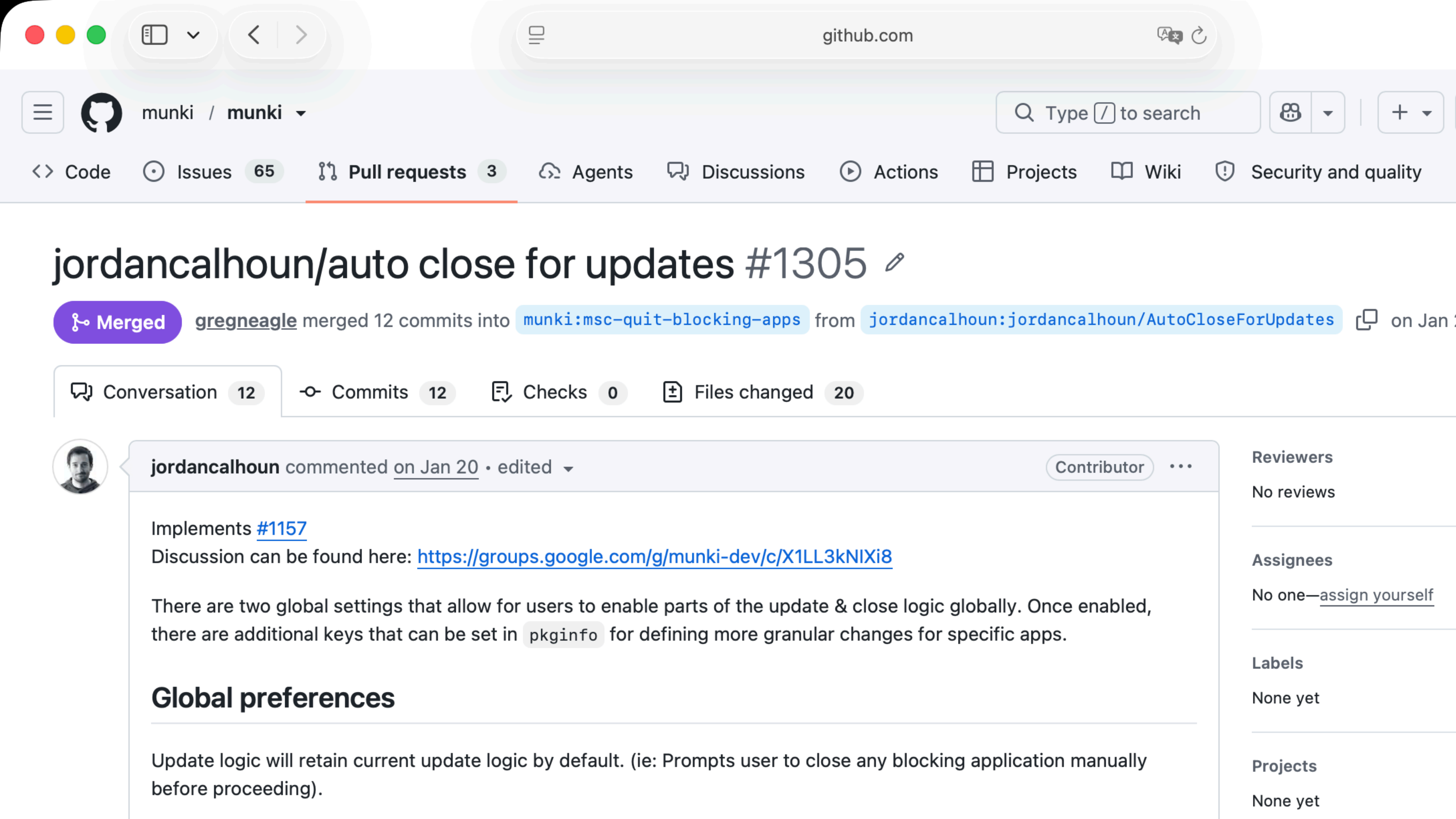
One of the largest friction points our users face is having to manually close apps before MSC can run updates. This is particularly difficult for apps that remain running in the background (looking at your 1Password). We often find some our users aren't sure how to fully quit some apps.

The proposal would be to add a preference to MSC that, when set, will offer to close any running apps for the user before updating. The general flow would be as follows:

- Check for blocking apps that are running, same as MSC does today
- Alert the user of the running apps, offering a button to close the apps
- Attempt to close all apps
- Prompt the user about any apps that fail to close, and offer the choice to manually close the apps and "try again" or force quit the apps (similar to how apps like CleanMyMac might do)
- Run pending updates

I have a proof of concept working and would be happy to submit a PR for this feature if this is something you are open to adding

Thanks!



jordancalhoun/auto close for updates #1305

Merged

gregneagle merged 12 commits into munki:msc-quit-blocking-apps from jordancalhoun:jordancalhoun/AutoCloseForUpdates on Jan 20

Conversation 12

Commits 12

Checks 0

Files changed 20

jordancalhoun commented on Jan 20 • edited

Contributor

Implements [#1157](#)
Discussion can be found here: <https://groups.google.com/g/munki-dev/c/X1LL3kNIXi8>

There are two global settings that allow for users to enable parts of the update & close logic globally. Once enabled, there are additional keys that can be set in pkginfo for defining more granular changes for specific apps.

Global preferences

Update logic will retain current update logic by default. (ie: Prompts user to close any blocking application manually before proceeding).

Reviewers

No reviews

Assignees

No one—[assign yourself](#)

Labels

None yet

Projects

None yet



Search

Software

Categories

My Items

Updates

2

Updates

2 pending updates

UPDATE ALL



Google's

Other a

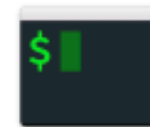


Conflicting applications running

You must quit these applications before proceeding with installation or removal:



Google Chrome



iTerm

☒ Reopen applications after update

Quit Apps and Update

Skip and Update Others

Cancel



iTerm2

George Nachman

Version 3.7.3 • 56.0 MB

CANCEL

iTerm2 is a replacement for Terminal and the successor to iTerm. Its focus is on...

More

Xcode 27

Apple

Version 27.0 • 3.0 GB

Development tools for macOS, iOS, iPadOS, tvOS, watchOS, and visionOS

OK, what does this have to do
with AI code assist?

AI-assisted new Munki features

Shipped

- Assisted quit and relaunch of blocking applications (7.1)
- Allowed notification windows (7.2)
- Client certificate auth with servers that don't advertise acceptable CA names (7.3)
- Add timeout to `version_script` (7.3)
- `munkiimport` improvements:
 - `--rebuild-catalogs` option (7.3)
 - support for disk images containing a folder of applications (7.3)

AI-assisted new Munki features

Shipped

- Assisted quit and relaunch of blocking applications (7.1)
- **Allowed notification windows** (7.2)
- Client certificate auth with servers that don't advertise acceptable CA names (7.3)
- Add timeout to `version_script` (7.3)
- `munkiimport` improvements:
 - `--rebuild-catalogs` option (7.3)
 - support for disk images containing a folder of applications (7.3)

Munki 7.2

Limit notifications to selected time frames

Limit notifications to selected time frames

UI features/requirements

- Select desired hour(s) for notifications
- Must select at least one hour
- No minutes selection
- Hours don't have to be a continuous range
- Admin should be able to restrict possible choices
- Should seem like a native UI for selecting



Settings



Limit notifications to selected hours

Start:

9

:00 AM



End:

6:00 PM





Settings



Limit notifications to selected hours

Start:

8am



End:

6pm





Settings

☒ Limit notifications to selected hours



8a



9a



10a



11a



12a



1p



2p



3p



4p



5p



6p

 Settings

☒ Limit notifications to selected hours

8a	9	10		11		12p		1		2		3	4	5
----	---	----	--	----	--	-----	--	---	--	---	--	---	---	---

Settings

MSCHourSelector.swift

```
1  //
2  // · MSCHourSelector.swift
3  // · Managed Software Center
4  //
5  // · Created by Greg Neagle
6  //
7  ─
8  |
9
```




Claude Code v2.1.162

Sonnet 4.6 · Amazon Bedrock

/Users/Shared/munki-git/munki

› Can you help me create a custom control in Swift to allow users to select a range of hours?

```
import Cocoa

class TimeRangeSelectorView: NSView {

    // MARK: – Properties

    var startTime: Int = 9 {
        didSet { needsDisplay = true }
    }

    var endTime: Int = 17 {
        didSet { needsDisplay = true }
    }

    private var isDragging = false
    private var dragStartTime: Int?

    private let hours = Array(0...23)

    // Colors
    private let gridColor = NSColor.gray.withAlphaComponent(0.3)
    private let selectedColor = NSColor.systemBlue.withAlphaComponent(0.3)
    private let backgroundColor = NSColor.controlBackgroundColor

    // MARK: – Initialization

    override init(frame frameRect: NSRect) {
        super.init(frame: frameRect)
        setup()
    }
}
```


HourSelection

HourSelection

AppDelegate.swift

Assets.xcassets

HourSelector.swift

MainMenu.xib

HourSelectionTests

HourSelectionUITests

HourSelection

HourSelection

HourSelector.swift

TimeRangeSelectorView

HourSelection: Ready | Today at 2:59 PM

```
7
8 import Cocoa
9
10 class TimeRangeSelectorView: NSView {
11
12     // MARK: - Properties
13
14     var startTime: Int = 9 {
15         didSet { needsDisplay = true }
16     }
17
18     var endTime: Int = 17 {
19         didSet { needsDisplay = true }
20     }
21
22     private var isDragging = false
23     private var dragStartTime: Int?
24
25     private let hours = Array(0...23)
26
27     // Colors
28     private let gridColor = NSColor.gray.withAlphaComponent(0.3)
29     private let selectedColor = NSColor.systemBlue.withAlphaComponent(0.3)
30     private let backgroundColor = NSColor.controlBackgroundColor
31
32     // MARK: - Initialization
33
34     override init(frame frameRect: NSRect) {
35         super.init(frame: frameRect)
36         setup()
37     }
38
39     required init?(coder: NSCoder) {
40         super.init(coder: coder)
41         setup()
42     }
43
44     private func setup() {
45         wantsLayer = true
46     }
```

Line: 43 Col: 1

Filter

Filter

Filter

Auto

Filter

Filter

Identity and Type

Name

HourSelector.swift

Type

Default - Swift Source

Location

Relative to Group

HourSelector.swift

Full Path

/Users/gneagle/Desktop/
HourSelection/
HourSelection/
HourSelector.swift

On Demand Resource Tags

Only resources are taggable

Target Membership

HourSelection

Default

Text Settings

Text Encoding

No Explicit Encoding

Line Endings

No Explicit Line Endings

Indent Using

Spaces

Widths

4

4

Tab

Indent



HourSelection

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----



Settings



Restrict update notifications to selected hours

8a

9

10

11

12p

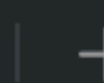
1

2

3

4

5



```
1  //
2  //  HoursSelector.swift
3  //  Managed Software Center
4  //
5  //  Created by Greg Neagle on 6/5/26.
6  //  Copyright © 2026 The Munki Project. All rights reserved.
7  //
8
9  import Cocoa
10
11 class HoursSelector: NSControl {
12     // MARK: - Properties
13
14     /// The text labels to display in cells
15     var cellLabels: [String] = [] {
16         didSet {
17             if selectedIndices.isEmpty {
18                 selectedIndices = Set([0])
19             }
20             // Recreate accessibility children when labels change
21             _accessibilityChildren = createAccessibilityChildren()
22             needsDisplay = true
23         }
24     }
25
26     /// Set of selected cell indices
27     var selectedIndices = Set<Int>() {
28         didSet {
29             needsDisplay = true
30         }
31     }
32 }
```

Demo

AI-assisted new Munki features

Shipped

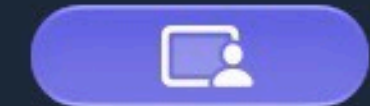
- Assisted quit and relaunch of blocking applications (7.1)
- Allowed notification windows (7.2)
- Client certificate auth with servers that don't advertise acceptable CA names (7.3)
- Add timeout to `version_script` (7.3)
- `munkiimport` improvements:
 - `--rebuild-catalogs` option (7.3)
 - support for disk images containing a folder of applications (7.3)

AI-assisted new Munki features

In Progress

- Additional launch arguments when relaunching blocking apps
- "Awareness" of data-constrained (low data) networks
- `optional_uninstalls`/Suggested removals
- Linux support for `makecatalogs`
- Customizing buttons for On Demand or Optional Installs

"Low data" support



Search

Software

Categories

My Items

Updates

Updates

No pending updates

CHECK AGAIN

Problem updates



Demo Large App
Example Co
Version 5.0.0 • 878.9 MB

MORE INFO

Download paused — low data connection
...

More



Search

Software

Categories

My Items

Updates

Updates

No pending updates

CHECK AGAIN

Problem updating



Low data connection detected

This item was not downloaded because this Mac is on a low data connection.

OK

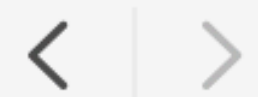
Download anyway

Download paused — low data connection

...

More

optional_uninstalls
(Recommended Removals)



github.com



munki / munki

Type / to search



<> Code

Issues 65

Pull requests 3

Agents

Discussions

Actions

Projects

Wiki

Security and quality

Proposal: new "installable" key #842

Closed



gregneagle opened on Mar 8, 2018

Member



Pkginfo items currently support an "uninstallable" key which Munki can use to determine if it can uninstall an item.

I propose a new "installable" key, which would default to true, but could be set to false. Use case: you want to keep an item in your repo (or want to add a new item) solely for the purpose of being available for uninstalls. (Perhaps you want no new installs of this item, or you've created a new pkginfo item *only* to uninstall unwanted software)

If such an item were added to managed_installs or managed_updates it would behave much as an item with uninstallable=false does when added to managed_uninstalls -- it causes a warning or error, but essentially is a no-op.

For optional_installs, such an item (with installable=false and uninstallable=true) would *not* be displayed if no version was installed; if some version was installed, it would show up with a "Remove" button to allow the user to uninstall the item.

Items with installable=True would not require any of the following fields:

Assignees

No one - A

Labels

Feature

Type

No type

Fields

Priority

➤ Help me add a feature. Munki should support a new manifest key named "optional_uninstalls". Items listed there are only available for _uninstall_ and only appear in Managed Software Center if they are currently installed. This allows admins to "phase out" software items -- they can move items from "optional_installs" to "optional_uninstalls" to prevent any new installations of an item, but also allowing users to remove the items from their Macs if and when they want.

Plan: Add `optional_uninstalls` manifest key to Munki (Swift only)

Context

Admins need a way to "phase out" software: prevent new installs while still letting users self-remove copies already on their Macs. The new `optional_uninstalls` manifest key fills that gap. Items listed there:

- Only appear in Managed Software Center if they are **currently installed**
- Only offer **Uninstall** (never Install)
- Behave like `optional_installs` in every other way (self-serve via SelfServeManifest, catalogs, etc.)

Approach

1. **Separate key in `InstallInfo.plist`** – `optional_uninstalls` stored as its own top-level array. Keeps semantics clear and doesn't touch the existing flow.
2. **New `processOptionalUninstall()` function** – mirrors `processOptionalInstall()` but only adds an item if it is currently installed, and skips install-only paths (precache, disk-space note for not-installed, etc.).
3. **MSC merges both lists** – `getOptionalInstallItems()` reads both arrays from `InstallInfo.plist`, tagging `optional_uninstalls` items with `uninstall_only = true`. Items tagged `uninstall_only` that are not installed are dropped at load time.
4. **Status / button logic** – for `uninstall_only` items the `not-installed` → `install-requested` transition in `update_status()` is blocked.
5. **`mutuallyExclusiveSections`** – `optional_uninstalls` added so an item can't simultaneously be in `optional_installs`.

Files to change

`code/cli/munki/shared/updatecheck/analyze.swift`

- New async function `processOptionalUninstall(_:catalogList:installInfo:)`:
 - Mirrors `processOptionalInstall()` structure.
 - After `alreadyProcessed()` check, also checks `optional_uninstalls` list for a duplicate name.
 - Calls `someVersionInstalled()` – if **not** installed, logs and returns without adding anything.
 - Skips the `unusedSoftwareShouldBeRemoved` / `precache` / disk-space-for-not-installed blocks.
 - Builds the same `processedItem` dict (name, display_name, description, version_to_install, installed=true, uninstallable, etc.).
 - Appends to `installInfo["optional_uninstalls"]` instead of `optional_installs`.

`code/cli/munki/shared/updatecheck/manifests swift`

Commits on Aug 21, 2026

Managed Software Center support for optional_uninstalls

 gregneagle committed 3 weeks ago

56404a1



repoclean support for optional_uninstalls

 gregneagle committed 3 weeks ago

b1f4b8f



manifestutil support for optional_uninstalls

 gregneagle committed 3 weeks ago

d209816



managedsoftwareupdate support for optional_uninstalls

 gregneagle committed 3 weeks ago

35022bd





🔍 Search

📦 Software

📁 Categories

🔬 My Items

⬇️ Updates

Updates

No pending updates

CHECK AGAIN

Suggested removals ⓘ



GrandPerspective

Erwin Bonsma

Version 3.8.0 • 4.0 MB

REMOVE

Note: GrandPerspective is no longer supported at WDAS. Consider using [Radix...](#) [More](#)



Search

 Software

 Categories

 My Items

 Updates

Updates

No pending updates

CHECK AGAIN

Suggested removals



GrandPerspective
Erwin Bonson
Version 3.8.1

Note: GrandPerspective is no longer supported at WDAS. Consider using [Radix...](#) [More](#)



Note: GrandPerspective is no longer supported at WDAS. Consider using [Radix](#) instead. GrandPerspective is a utility application for macOS that graphically displays the disk usage of a file system.

Version 3.8.1 • 4.0 MB

AI-assisted new Munki features

Proposed

- YAML support for manifests and pkginfo files

AI-assisted new Munki features

- Assisted quit and relaunch of blocking applications (**Jordan Calhoun**)
- Allowed notification windows (Greg Neagle)
- Client certificate auth with servers that don't advertise acceptable CA names (**Nathaniel Strauss**)
- Add timeout to `version_script` (**Kevin M. Cox**)
- `munkiimport` improvements:
 - `--rebuild-catalogs` option (**Brandon Kurtz**)
 - support for disk images containing a folder of applications (Greg Neagle)
- Additional launch arguments when relaunching blocking apps (**Nathaniel Strauss**)
- "Awareness" of data-constrained (low data) networks (**Graham Gilbert**)
- `optional_uninstalls`/Suggested removals (Greg Neagle)
- Linux support for `makecatalogs` (Greg Neagle)
- Customizing buttons for On Demand or Optional Installs (**Allister Banks**)

“It has allowed me to **feel more confident** opening a PR that’d hold up to review; **last year I’d have just filed an issue** and hoped someone else had time...”

- Kevin M. Cox, on using AI code assist

Debugging

AI-assisted new Munki bug fixes

Shipped

- A fix for clearing bootstrap mode after a managedsoftwareupdate run (AI found a typo)
- Eliminate a fractional-second delay when building/displaying the Updates page in Managed Software Center
- Eliminate typos in command-line tool help messages

munki



elios Aug 4th at 9:01 AM

Munki 7.3 beta 1: <https://github.com/munki/munki/releases/tag/v7.3.0b1>



kevinmcox Aug 4th at 9:51 AM

Sidebar issue looks good here The Updates tab takes a fraction of a second longer to load, but otherwise looking good.

+1 1



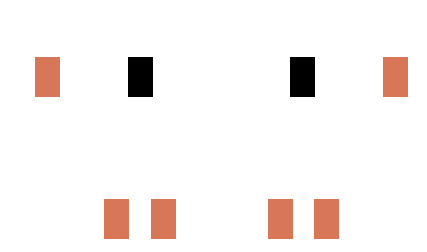
elios Today at 1:49 PM

The Updates page may well be a bit slower to display; that might not be 7.3b specific



kevinmcox Today at 1:49 PM

You are right doesn't look to be (7.3 specific). (edited)



Claude Code v2.1.162

Sonnet 4.6 · Amazon Bedrock

/Users/Shared/munki-git/munki/code/apps/Managed Software Center

- › Can you look at the code for Managed Software Center and tell me why it takes 10x longer to generate the Updates page compared to other pages?

Thought for **34s**, searched for **2** patterns, read **1** file (ctrl+o to expand)

- Now I have the full picture. Let me check how many times `getAppleUpdates()` is called during a single page build:

Searched for **1** pattern, read **1** file (ctrl+o to expand)

- Now I have the complete picture. Here's a clear explanation of what's happening:

Why the Updates page is 10x slower

The root cause is that `getAppleUpdates()` is not cached, while every other data-fetching function used by other pages is.

The caching pattern (used by every other page)

All other data functions check `Cache.shared` before doing work:

```
// MunkiItems.swift:1157 – getOptionalInstallItems()
func getOptionalInstallItems() -> [OptionalItem] {
    if !Cache.shared.keys.contains("optional_install_items") {
        // ... expensive work only done ONCE ...
        Cache.shared["optional_install_items"] = optional_install_items
    }
    return Cache.shared["optional_install_items"] as? [OptionalItem] ?? [OptionalItem]()
}
```

Same pattern for `getUpdateList()` (line 1263), `getProblemItems()` (line 1186), `cachedInstallInfo()` (line 1068) – all cached.

`getAppleUpdates()` not cached at all

For Apple update items, `getIcon()` (MunkiItems.swift:219–237) calls `findSoftwareUpdateIconPath()` and potentially `convertIconToPNG()` (image extraction/conversion). This happens in the `UpdateItem` init via `GenericItem.init(_:)` at line 141. However, there's already a short-circuit at line 227 (`FileManager.default.fileExists(atPath: icon_path)`) that avoids re-conversion once the PNG exists – so on subsequent loads this is cheap. The plist reads are the persistent bottleneck.

Fix

Add caching to `getAppleUpdates()` exactly as the other functions do:

```
func getAppleUpdates() -> [PlistDict] {
    if !Cache.shared.keys.contains("apple_updates") {
        Cache.shared["apple_updates"] = _getAppleUpdates()
    }
    return Cache.shared["apple_updates"] as? [PlistDict] ?? [PlistDict]()
}
```

This would bring the Updates page in line with all the other pages that only do their heavy work once per cache lifetime.

* Sautéed for 3m 11s

※ **recap:** *We investigated why the Updates page in Managed Software Center renders 10x slower than other pages. The cause is that `getAppleUpdates()` lacks caching and is called 4 times per render. Next: add caching to `getAppleUpdates()` in `munki.swift`. (disable recaps in `/config`)*

- ※ **recap:** *We investigated why the Updates page in Managed Software Center renders 10x slower than other pages. The cause is that `getAppleUpdates()` lacks caching and is called 4 times per render. Next: add caching to `getAppleUpdates()` in `munki.swift`. (disable recaps in `/config`)*
- › Why is Managed Software Center, when built with the `code/tools/make_swift_munki_pkg.sh` script, slower than Managed Software Center when built directly from the Xcode app?

Managed Software Center.xcscheme:33 – buildConfiguration = "Debug"

When you press Run (Cmd+R) in Xcode, it uses the **Debug** configuration. When the script runs xcodebuild (make_swift_munki_pkg.sh:365), it explicitly uses **Release**.

The two configurations have opposite optimization settings:

Setting	Debug (Xcode Run)	Release (script)
SWIFT_OPTIMIZATION_LEVEL	-Onone	-O
SWIFT_COMPILATION_MODE	(not set / incremental)	wholmodule
GCC_OPTIMIZATION_LEVEL	0	(default, i.e. -O2)
ONLY_ACTIVE_ARCH	YES	(not set / NO)
DEBUG=1 preprocessor	YES	NO

-Onone means zero optimization – every function call is a real call, values are kept in memory for the debugger, nothing is inlined or eliminated. The Release build with -O + wholmodule should be **faster** at runtime, not slower. Whole-module optimization in particular allows the compiler to inline across file boundaries, which matters a lot in a codebase like this where GenericItem, Cache.shared, and the template machinery are called in tight loops.

So the premise is reversed: the script-built binary should run faster. If you're measuring the opposite, the likely explanation is that you're comparing a Debug run from Xcode (fast to compile, slow to run) against the installed package (slower to build, faster to run), and attributing the difference to the script rather than the configuration. The Xcode Profile action (ProfileAction buildConfiguration = "Release" at line 54) and Archiving action (line 74) both use Release, and those

› OK, I have the trigger, but don't understand (yet) why it affects the speed the Updates page is built. I realized that the version of MSC.app that's built by the script and installed via pkg contains two helper apps: munki-notifier and MunkiStatus, both in Managed Software Center.app/Contents/Helpers. I used Product->Build to build the app in Xcode, verified it built the Updates page quickly, then copied those helper apps into the app bundle. Tested again, and now the Updates page builds 10x slower. I don't understand why the existence of the helper apps in Managed Software Center.app/Contents/Helpers would affect how fast the Updates page is built.

Searched for 1 pattern, read 2 files (ctrl+o to expand)

- That's the cause.

Every time `updates.html` is navigated to (MainWindowController.swift:1208–1211), `removeAllDeliveredNotifications()` is called. That function:

1. Checks `FileManager.default.fileExists(atPath: munkiNotifierPath)` – which is `false` when the helpers aren't present, so it immediately returns
2. When the helpers **are** present, it falls through to `getRunningProcessesWithUsers()`, which calls `/bin/ps -axo user=,comm=` as a subprocess via `exec()`

Spawning a `/bin/ps` process is expensive – it has to enumerate the entire process table. And `exec()` is a synchronous call that runs on the main thread, blocking the UI. This happens every single time the Updates page is loaded or reloaded.

The fix is simple: the `FileManager.default.fileExists` guard at line 1246 is correct and does short-circuit when the helpers aren't there. But when they **are** there, `getRunningProcessesWithUsers()` runs `ps` every page load. It should check whether `munki-notifier` is actually running with something cheaper – or simply launch it unconditionally with `open -a` and let the OS handle the "already running" case, since `open` is idempotent by default. Alternatively, cache the result of `getRunningProcessesWithUsers()` for the duration of the page load rather than re-running `ps` on every navigation to `updates.html`.

* Baked for 3m 19s

* **recap:** *Investigating why Managed Software Center's Updates page loads 10x slower when built via the package script. The cause is that every visit to the Updates page spawns a ps subprocess to check if munki-notifier is running, but only when the helper apps are present in Contents/Helpers. Next step is to fix removeAllDeliveredNotifications() in MainWindowController.swift. (disable recaps in /config)*

Thread.detachNewThread – lowest level, no frameworks beyond Foundation:

```
Thread.detachNewThread {  
    self.removeAllDeliveredNotifications()  
}
```

Task.detached – what's already in place, Swift 5.5+:

```
Task.detached(priority: .background) { self.removeAllDeliveredNotifications() }
```

For this use case the GCD approach is probably the most idiomatic for a Cocoa app – it's immediately readable to anyone familiar with macOS development, doesn't require thinking about Swift concurrency's actor isolation rules, and `removeAllDeliveredNotifications` is a plain synchronous method so there's no benefit to using structured concurrency here. The `Task.detached` version is fine but slightly misleading since it implies `async/await` is involved when it isn't.

One thing worth noting with both the `Task.detached` and GCD versions: `self` is captured strongly. That's harmless for a window controller that lives for the duration of the app, but if you wanted to be tidy you could use `[weak self]` with GCD:

```
DispatchQueue.global(qos: .background).async { [weak self] in  
    self?.removeAllDeliveredNotifications()  
}
```

* Crunched for 25s

> /recap

└ We're investigating why the Updates page in Managed Software Center builds 10x slower when installed via the build script versus run from Xcode. We just fixed the cause by dispatching `removeAllDeliveredNotifications()` to a background thread at its call site; the next step is to verify the fix works as expected.

※ **recap:** *We found that the Updates page was slow because `removeAllDeliveredNotifications()` runs a blocking `ps` subprocess on the main thread before page generation, and fixed it by dispatching that call to a background thread. The next action is to choose a threading approach (GCD is recommended) and update the call site accordingly.*



```
18 class MainWindowController: NSWindowController {
1182 func load_page(_ name: String) {
1209     if name == "updates.html" {
1210         if !_update_in_progress && NSApp.isActive {
1211             // clear all earlier update notifications
1212             // we'll do this in a background queue so as not to delay
1213             // the loading of the Updates view
1214             DispatchQueue.global(qos: .background).async {
1215                 self.removeAllDeliveredNotifications()
1216             }
1217         }
1218         // record that the user has been presented pending updates
1219         if !_update_in_progress && !shouldAggressivelyNotifyAboutMunkiUpdates() &&
            !thereAreUpdatesToBeForcedSoon() {
1220             _alertedUserToOutstandingUpdates = true
1221         }
1222     }
1223 }
```

More open source

autopkg

Code ▾

 Closed

+155 -21



What

This changes the header handling to best-effort: record whichever validators the server actually sent, warn about the

Development

Code ▾

 Merged

+761 -315



Member ...



-

Verified 73bb0df

Verified efdce92

Verified 31bff98

Verified f00e365

Verified b2dc0d0

Reviewers 

No reviews

Assignees

No one—assign yourself

Labels 

None yet

Projects

None yet

Milestone

No milestone

Development

Successfully merging this pull request may close these

Open

Ensure that `CACHE_DIR` is an absolute path #982

`dtheyer` wants to merge 1 commit into `autopkg:master` from `dtheyer:master`

homebysix requested a review from **Copilot** 5 months ago

Copilot started reviewing on behalf of **homebysix** 5 months ago

View session

Copilot AI reviewed on Apr 21

View reviewed changes

Copilot AI left a comment

Pull request overview

Adjusts `autopkg run` to normalize `CACHE_DIR` to an absolute path to avoid issues when processors change the working directory and to improve reliability when callers provide relative cache paths.

Changes:

- Resolves `cache_dir` to an absolute path and stores it as `CACHE_DIR` via `set_pref` during `run_recipes`.

AutoPkg 3.0.0 Latest

[Compare](#) ▾

homebysix released this yesterday v3.0.0 c36e58f

3.0.0 (September 28, 2026)

Recipe map

AutoPkg now uses a "recipe map," an on-disk JSON cache (`~/Library/AutoPkg/recipe_map.json` by default) of every recipe and override on the local AutoPkg setup, indexed by identifier and shortname. This makes recipe resolution much faster on systems with many configured recipe repos.

The first use after install or `repo-add` / `repo-update` may be briefly slower while the map is built. The cache rebuilds automatically when missing or invalid, stays in sync through `repo-add`, `repo-delete`, `repo-update`, `make-override`, and `new-recipe`, and can be manually rebuilt with `autopkg generate-recipe-map` (useful in CI). CLI `--search-dir` / `--override-dir` values take precedence over preferences.

Troubleshooting recipe map issues:

- Run `autopkg generate-recipe-map` to force a clean rebuild.
- If you edit an override's `Identifier` (common when copying an override for a multi-arch setup), AutoPkg reports that the map is stale and names the new identifier. Run `autopkg generate-recipe-map` to rebuild.
- Set `RECIPE_MAP_PATH` preference or `AUTOPKG_RECIPE_MAP_PATH` env var to redirect the cache to a writable location (e.g. CI workspaces that don't use `~/Library/AutoPkg`).
- Set `DISABLE_RECIPE_MAP` preference or `AUTOPKG_DISABLE_RECIPE_MAP=1` env var to bypass the cache entirely and fall back to the legacy on-disk scanners.

Note

When running AutoPkg with `sudo` (which is still not recommended outside of special use cases like CI), `AUTOPKG_RECIPE_MAP_PATH` is ignored and a warning

Problematic AI Pull Requests

feat: Launch Services support for file extensions

Code

Closed

wants to merge 1 commit into scriptingosx:main from utiluti-extensions

Conversation 2

Commits 1

Checks 0

Files changed 13

+894 -136

commented on Apr 10

Summary

This PR fixes how utiluti handles filename extensions.

Before this change, commands like `type -e ...` and `manage` entries like `extension:xcodeproj` would turn the extension into a UTI right away. That worked for common types, but it broke when macOS returned a dynamic `dyn.*` UTI. In those cases, `utiluti` could accept an extension as input but could not reliably set the handler for that extension.

With this change, `utiluti` keeps track of whether the target is a UTI or a file extension. If an extension maps cleanly to a UTI, it still uses that path. If not, it can fall back to the extension-specific Launch Services entry instead.

What Changed

- `type` and `manage` now keep the original target kind instead of flattening everything into a UTI
- handler lookup and updates now go through one shared path for UTIs and extensions
- extension-specific Launch Services overrides are used when the UTI path is not enough

Please note that some parts of this code have been implemented using Codex, as I am not that strong in Swift

Reviewers

No reviews

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Successfully merging this pull request may close these issues.

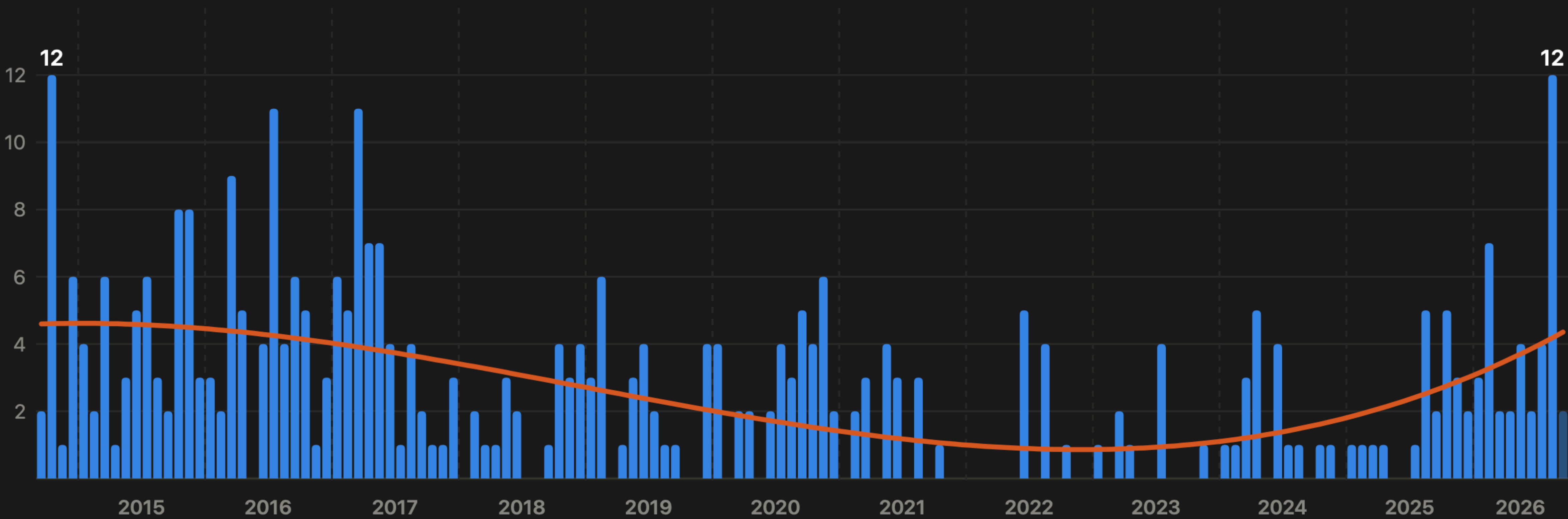
None yet

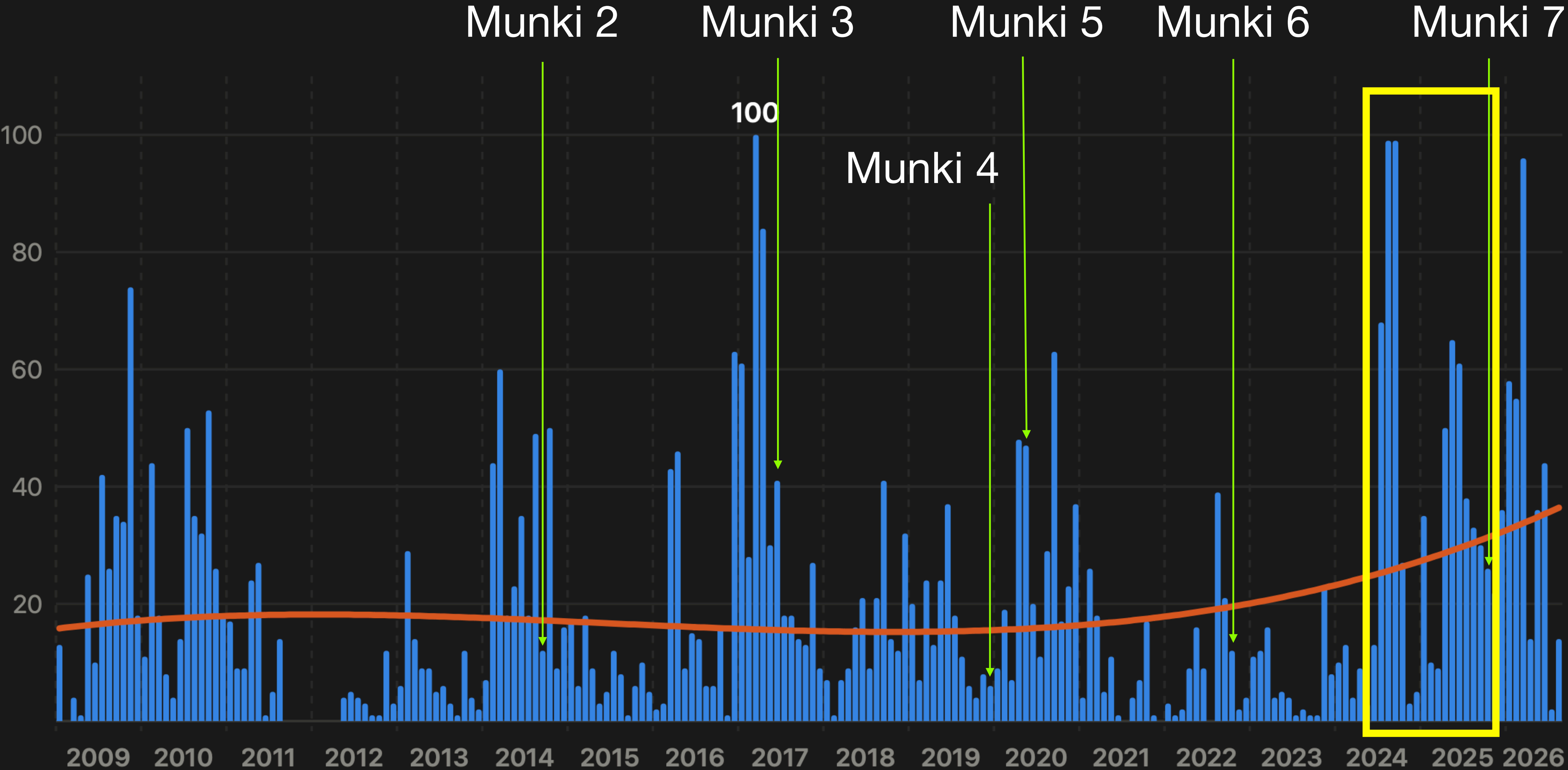
Pace of change

munki/munki · Pull Requests per month

Sep 2014 – Sep 2026 · GitHub · Excluding translation & localization PRs (56 filtered)

PRs / month Trend (cubic)

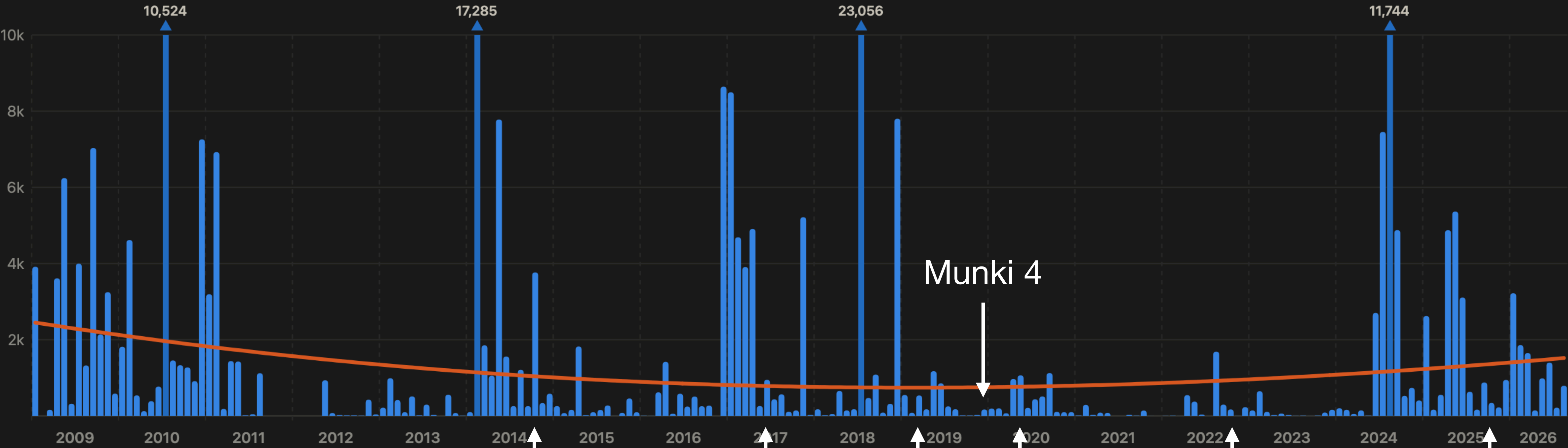




munki · Lines added by gregneagle@mac.com

Jan 2009 – Aug 2026 · local git history · Y-axis capped at 10,000 — lighter bars exceed cap (actual value in label)

Lines added Exceeds cap Trend (cubic)



Munki 2
(MSC.app)

Munki 3

Munki 3.6
(Swift MSC.app)

Munki 5

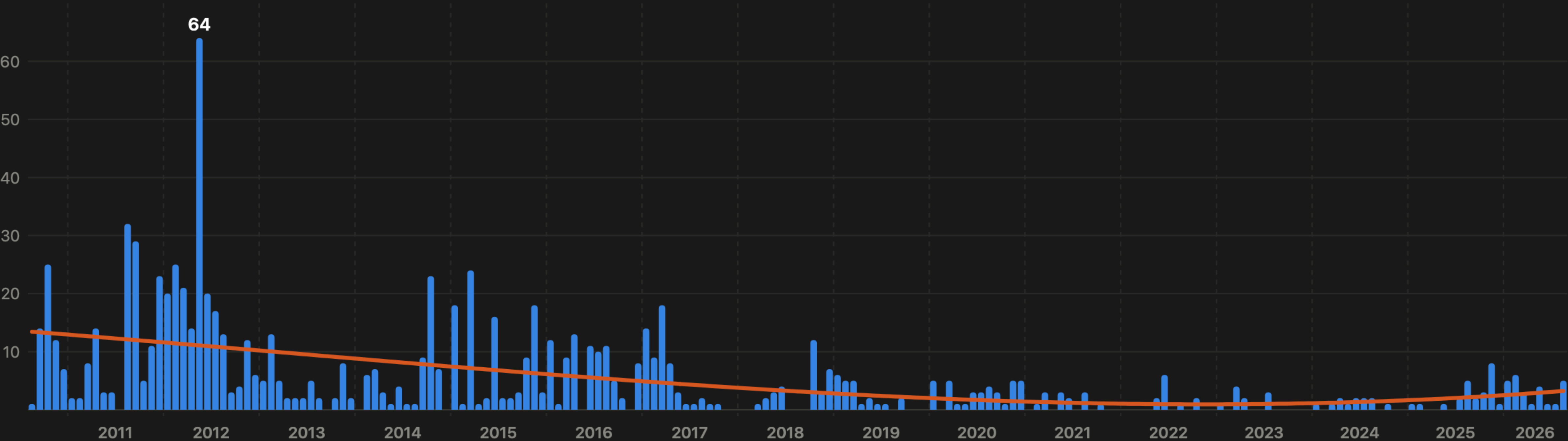
Munki 6

Munki 7
(Swift everything)

munki · Commits by other contributors

Aug 2010 – Aug 2026 · Excludes gregneagle@mac.com · Excludes lproj-only commits (142 filtered) · local git history

Commits / month Trend (cubic)

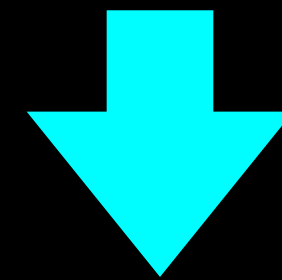


Where might this all lead?

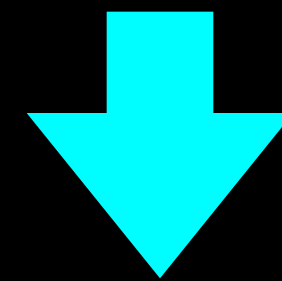
Better tools



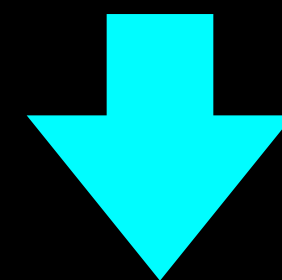
Customer



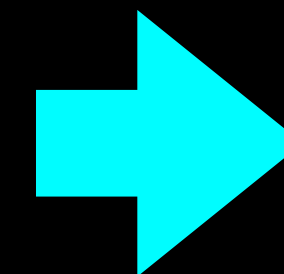
Sales people/sales engineers



Product manager(s)



Developers

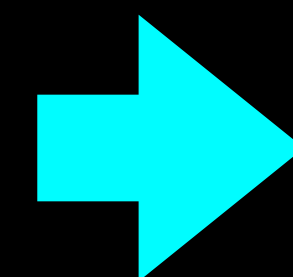


New feature!





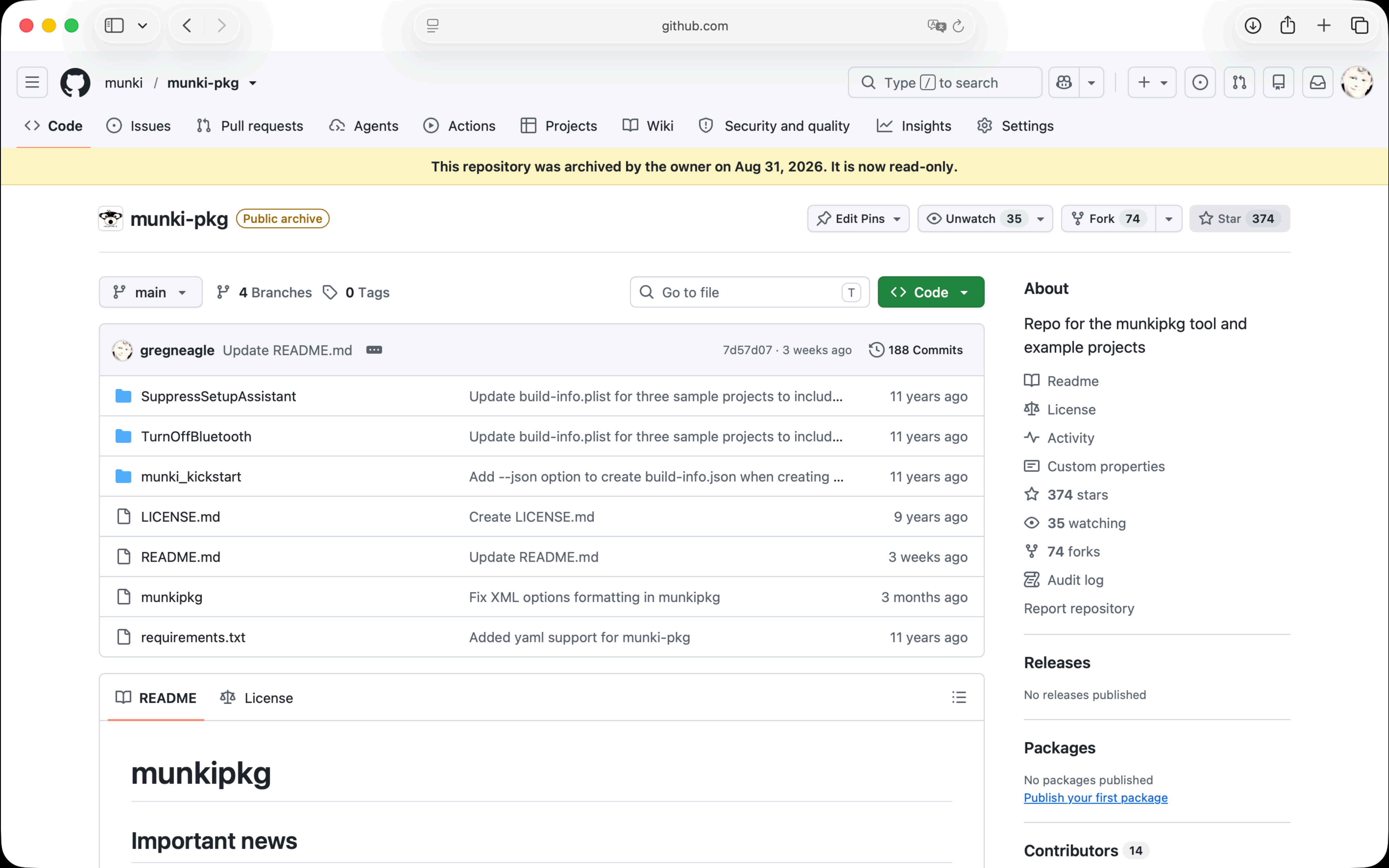
Developers



New feature!



More tools



munki-pkg

Public archive

Edit Pins

Unwatch 35

Fork 74

Star 374

main

4 Branches

0 Tags

Go to file

Code



gregneagle

Update README.md



7d57d07 · 3 weeks ago

188 Commits



SuppressSetupAssistant

Update build-info.plist for three sample projects to includ...

11 years ago



TurnOffBluetooth

Update build-info.plist for three sample projects to includ...

11 years ago



munki_kickstart

Add --json option to create build-info.json when creating ...

11 years ago



LICENSE.md

Create LICENSE.md

9 years ago



README.md

Update README.md

3 weeks ago



munkipkg

Fix XML options formatting in munkipkg

3 months ago



requirements.txt

Added yaml support for munki-pkg

11 years ago



README



License



munkipkg

Important news

About

Repo for the munkipkg tool and example projects

Readme

License

Activity

Custom properties

374 stars

35 watching

74 forks

Audit log

Report repository

Releases

No releases published

Packages

No packages published

[Publish your first package](#)

Contributors 14

github.com

rodchristiansen / **munki-pkg**

Q

Type / to search

<> Code

Pull requests

1

Agents

Actions

Projects

Security and quality

Insights

Files

main

+

Q

Q

Go to file

T

>

SuppressSetupAssistant

>

TurnOffBluetooth

>

munki_kickstart

>

swift

.gitignore

LICENSE.md

README.md

Swift Migration.md

munkipkg

requirements.txt

munki-pkg / munkipkg

Add file

rodchristiansen

Add --verify-json to write verify results to a JSON file

41db944 · 2 months ago

History

This branch is 50 commits ahead of and 6 commits behind munki/munki-pkg:main .

Name	Last commit message	Last commit date
..		
MPKconvert.swift	(refactor) Changed flag --migrate to --convert to match the subco...	9 months ago
buildinfo.swift	Add dynamic versioning support with placeholders for timestamp, date,...	9 months ago
cliutils.swift	Update project to Swift 6.2 with concurrency and Sendable conformance	last year
envloader.swift	Address Copilot review feedback	4 months ago
errors.swift	Make munkipkg CI-safe: hard-fail notarization, JSON result, clean str...	4 months ago
munkipkg.swift	Add --verify-json to write verify results to a JSON file	2 months ago
munkipkgoptions.swift	Add --verify-json to write verify results to a JSON file	2 months ago

swiftpkg

The munki-pkg workflow,
rebuilt in Swift.



swiftpkg

CI passing release swiftpkg 0.4.1 license Apache-2.0 Apple CLI macOS 13+ Apple Swiftpkgr macOS 15+

`swiftpkg` is a macOS command-line tool for building Apple installer packages from version-control-friendly project directories. It is a Swift implementation of [munki-pkg](#). `Swiftpkggr` is its native macOS desktop app. Both frontends use the same `SwiftPkgCore` package engine and open the same portable projects.

Quick start

Install the CLI and desktop app independently from the official [codecarton/homebrew-tap](#) :
Open "https://github.com/codecarton/swiftpkg/blob/main/site/assets/og.png" in a new tab

main 1 Branch 4 Tags Go to file T Add file <> Code

 README



No description, website, or topics provided.

- Report repository

 munkipkg v0.2.2 Latest
8 months ago

No packages published

headmin Henry Stamerjohann

munki / munki-pkg ▾

Type / to search

<> Code

⌚ Issues

🔗 Pull requests

☁ Agents

▶ Actions

📁 Projects

📖 Wiki

🛡 Security and quality

📈 Insights

⚙ Settings

Rewrite munkipkg in Perl so it runs on stock macOS (no Python required) #82

❌ Closed

keeleysam wants to merge 1 commit into `munki:main` from `keeleysam:perl-port`

💬 Conversation 1

🔗 Commits 1

📄 Checks 0

📄 Files changed 11

+1,959 -1

keeleysam commented on Jul 9

Member ⋮

Why

Recent macOS releases no longer ship a Python interpreter. Today `munkipkg` requires each user to install and maintain their own Python 3 (via python.org, Homebrew, Munki's bundled Python, etc.) before they can build a package. This rewrites `munkipkg` as a single Perl script that runs against the Perl that ships with **stock macOS** (currently 5.34), with **no dependencies to install**.

It's a drop-in replacement — nobody should notice the change

The only difference anyone should experience is that **they no longer need to install Python**. Everything else is unchanged:

- Same command name, options, and `--help` / `--version` / usage text.
- Same `munkipkg: ...` stdout messages, the same `ERROR:` / `WARNING:` stderr text, and the same exit codes.

Reviewers

No reviews

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

ADMIN-PROVIDED CONDITIONS · PERL 5.8.6

munki-perls

Typed Munki perls for Macs from Tiger onward.

FROM G3 TO M3, FROM G5 TO M5, WITH INTEL
HOLDING THE MIDDLE: IT'S ALL SUPPORTED.

[DOWNLOAD LATEST](#)[VIEW ON GITHUB](#)

SIX COLLECTORS
ONE NATIVE PLIST

The screenshot shows a web browser window displaying the GitHub repository page for 'weswhet / munki-perls'. The browser's address bar shows 'weswhet.github.io'. The repository page has a header with the repository name and navigation links: 'Code', 'Issues 1', 'Pull requests 1', 'Agents', 'Actions', 'Projects', 'Security and quality', and 'Insights'. The 'Files' tab is selected, showing a file tree with folders '.github', 'assets', 'conditions', 'lib', and 'perls'. The 'perls' folder is expanded, showing files including 'admin_users.pl', 'backtomymac_configured.pl', 'client_id.pl', 'console_user.pl', 'crashplan_username.pl', and 'filevault_status.pl'. The file 'admin_users.pl' is selected, and its content is displayed in the main area. The file content is a Perl script with 15 lines, including 'use' statements for '5.008006', 'strict', and 'warnings', and a sub routine 'admin_users' that returns a sorted list of members. The file size is 343 Bytes. A commit message by 'keeleysam' is visible: 'Lower minimum Perl version requirement to 5.8.6'.

github.com

weswhet / munki-perls

Code Issues 1 Pull requests 1 Agents Actions Projects Security and quality Insights

Files

main + 🔍

Go to file T

- > .github
- > assets
- ✓ conditions
 - > lib
 - ✓ perls
 - admin_users.pl
 - backtomymac_configured.pl
 - client_id.pl
 - console_user.pl
 - crashplan_username.pl
 - filevault_status.pl

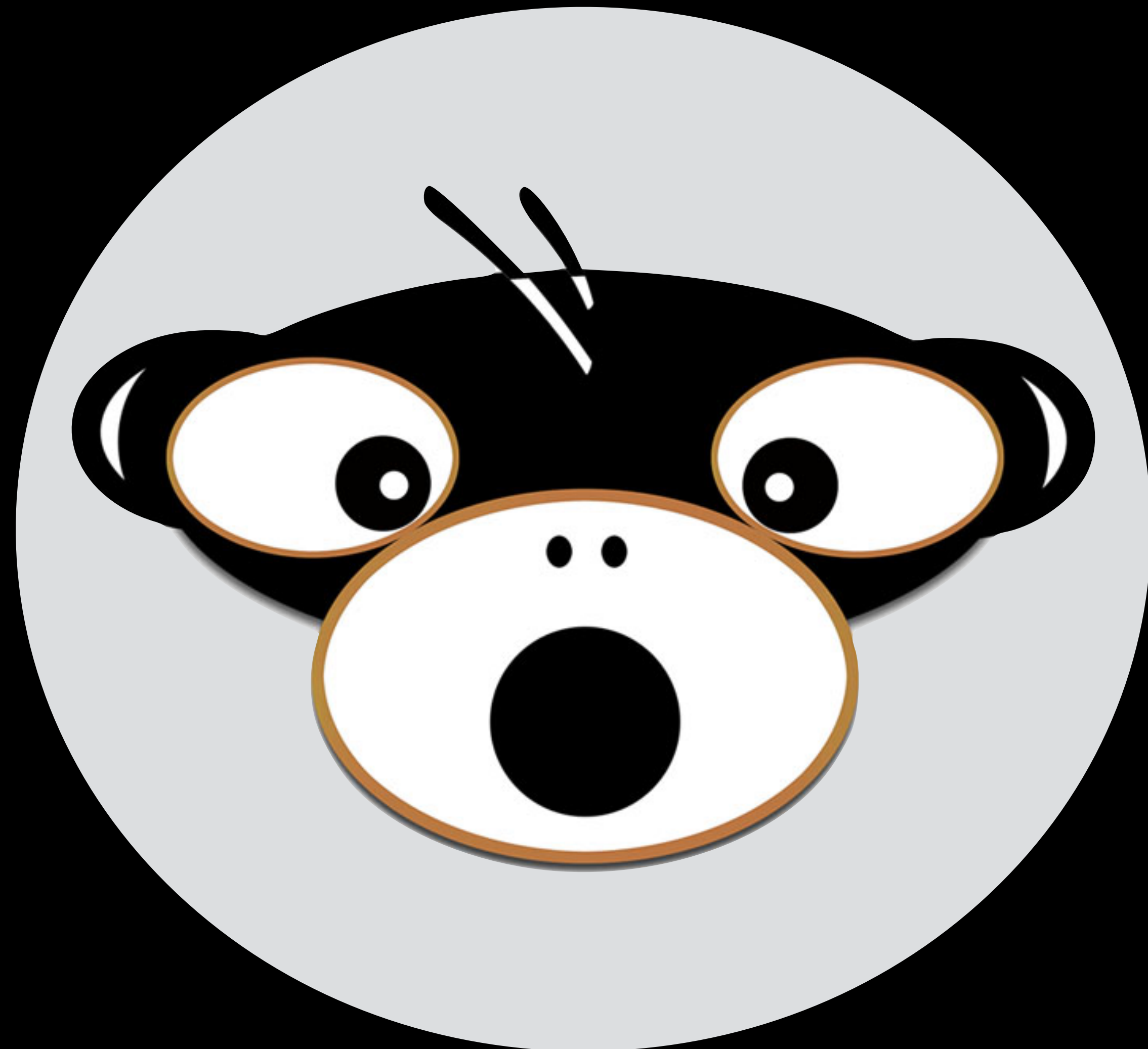
munki-perls / conditions / perls / admin_users.pl

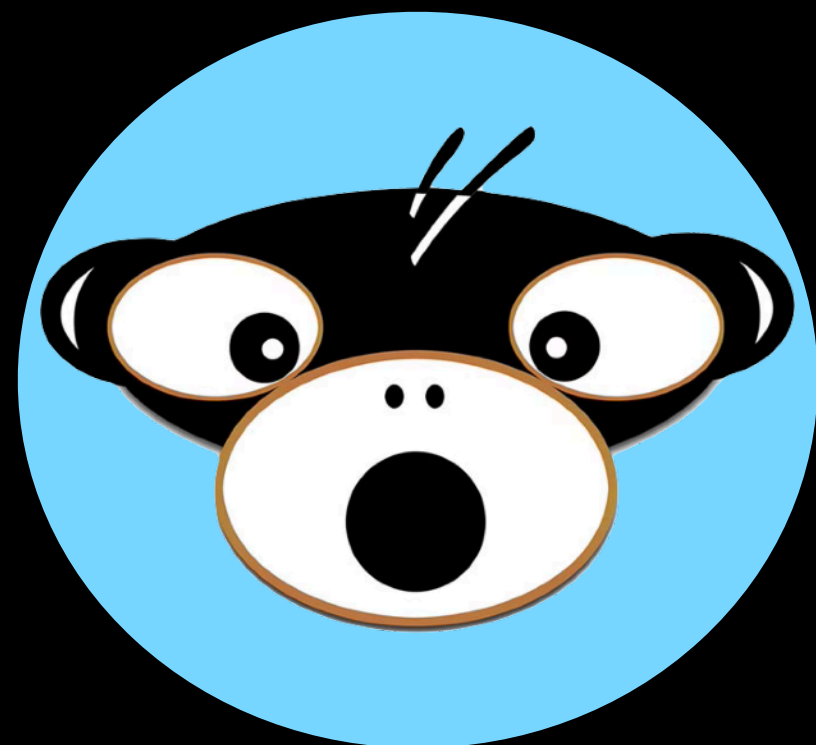
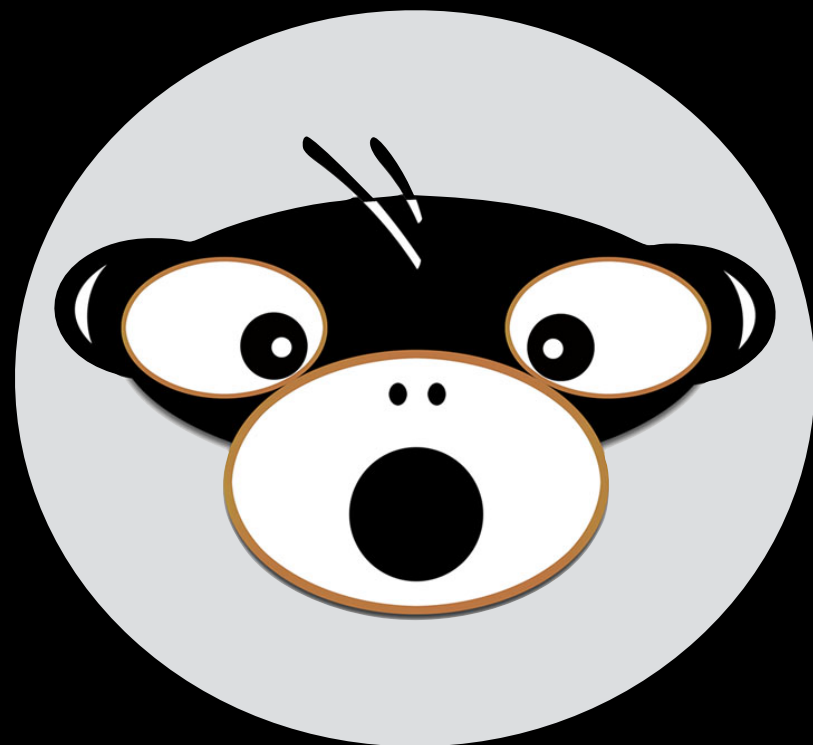
keeleysam Lower minimum Perl version requirement to 5.8.6

Code Blame 15 lines (13 loc) · 343 Bytes

```
1 use 5.008006;
2 use strict;
3 use warnings;
4 use MunkiPerls qw(perl_array);
5
6 sub admin_users {
7     my (undef, undef, undef, $members) = getgrnam('admin');
8     return () unless defined $members && length $members;
9     return sort grep { length $_ } split /\s+/, $members;
10 }
11
12 sub perls {
13     return { admin_users => perl_array(admin_users()) };
14 }
15 1;
```

**Custom forks/
Bespoke tools**





Endgame?

 Good afternoon, Greg

Help me create a tool to manage software and configuration on macOS



Sonnet 5 Medium

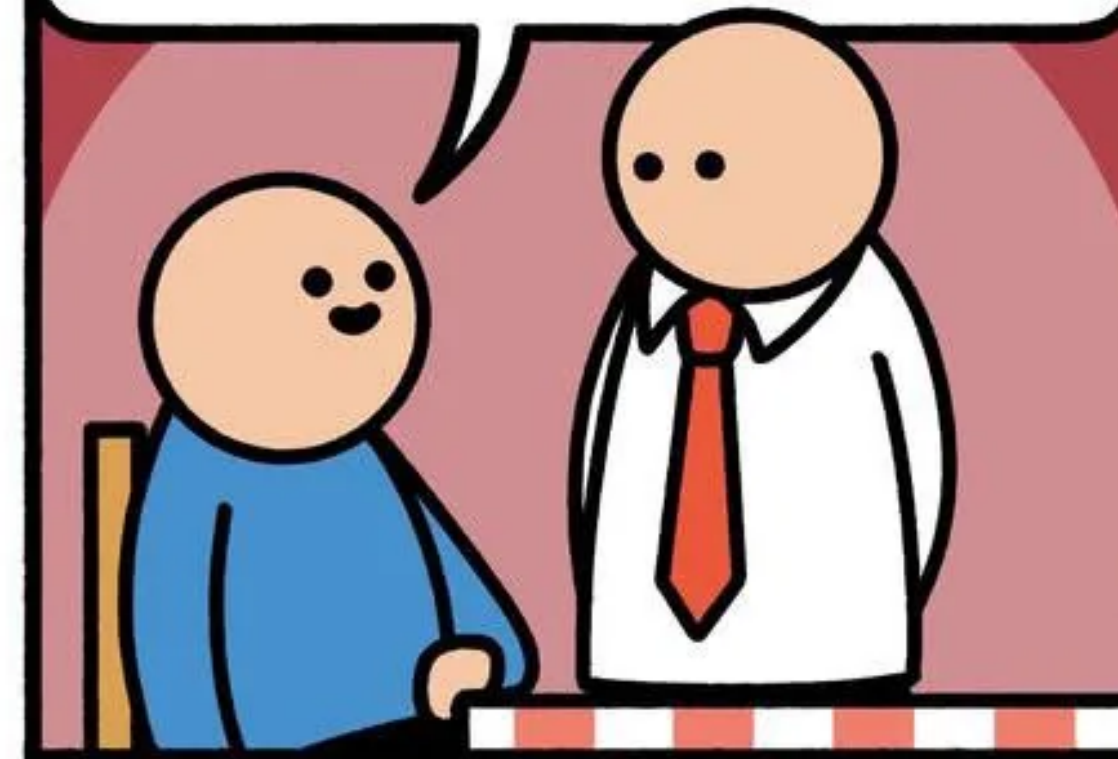


More concerns?

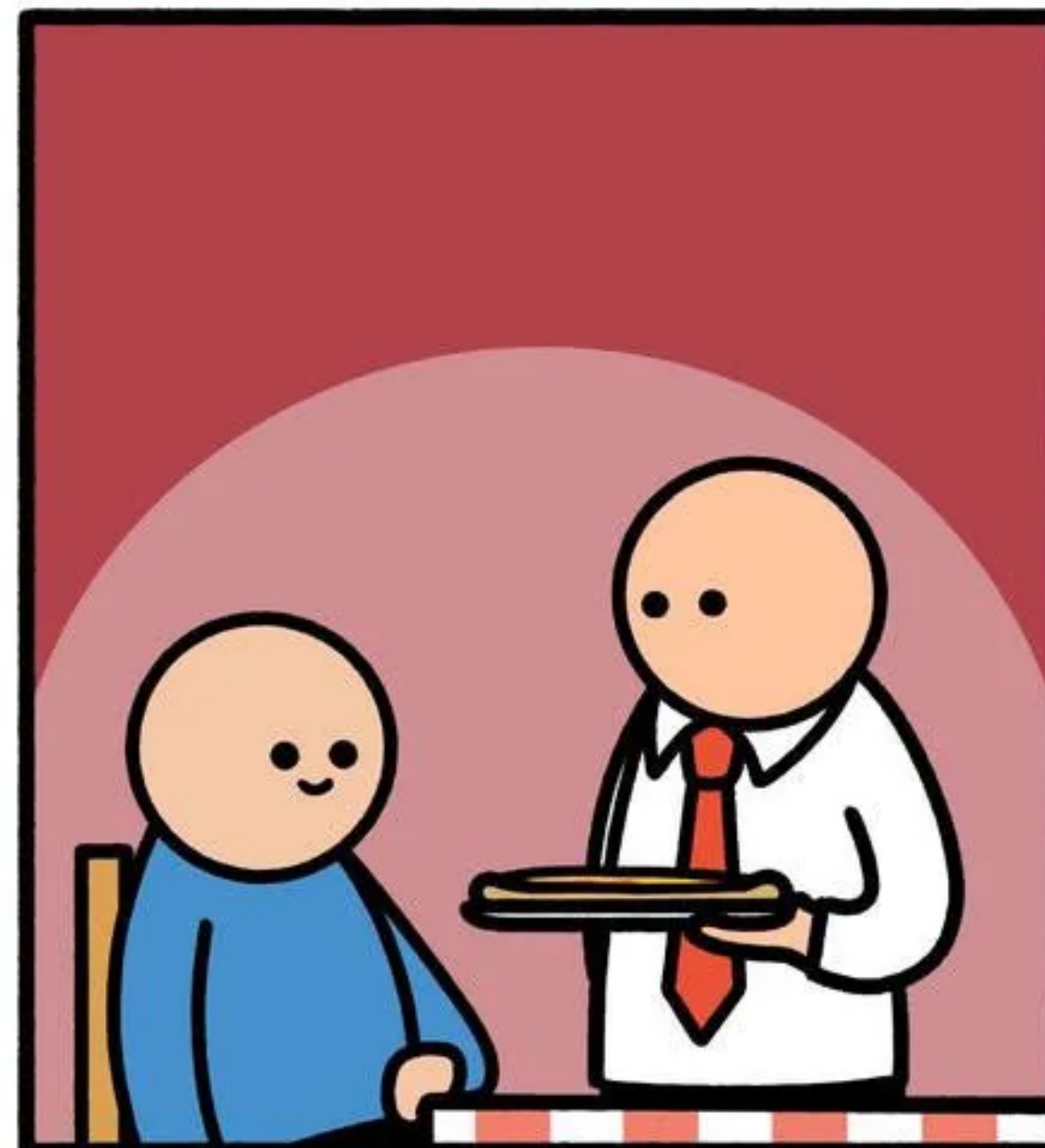
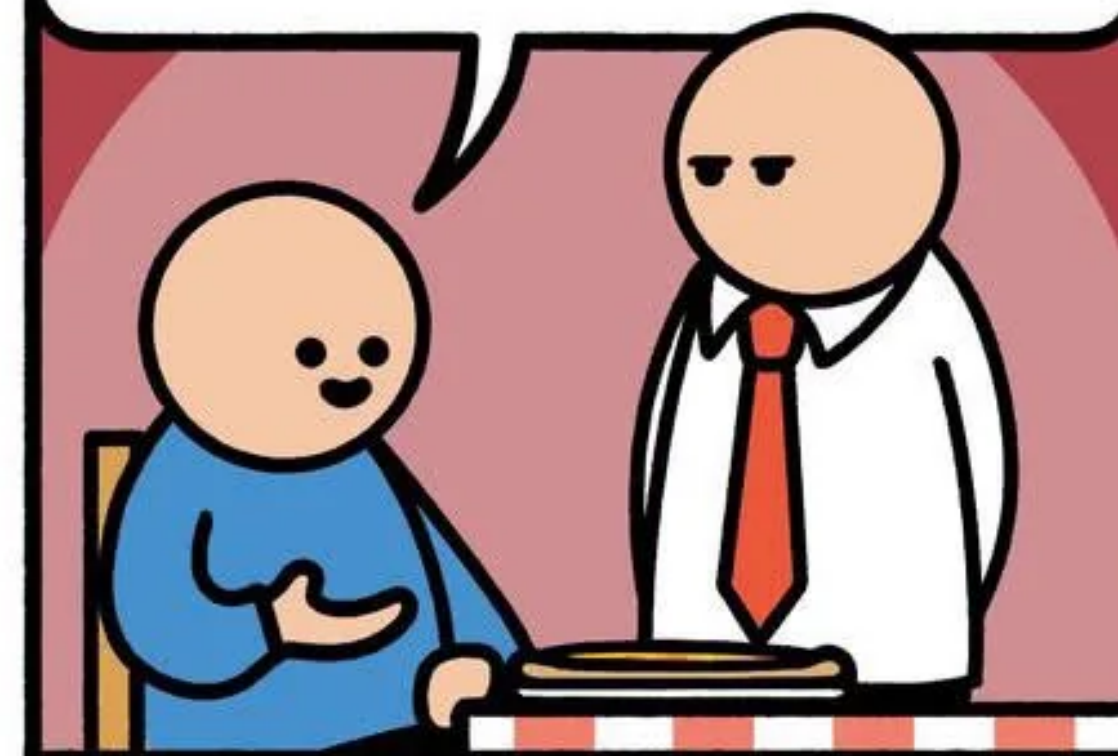
Ownership

AI DESIGN LOGIC

GET ME A
CHEESE PIZZA



WOW, LOOK
WHAT I MADE!



Skills erosion

Trust the black box(?)

~~Complacency~~

Other open source maintainers

“I ... worry about how much money and resources all these tokens are consuming”

“best practices are still immature”

“the rate of change is exhausting”

“I worry about human skill atrophy”

“community access is uneven”

“I certainly see that PRs and issues have gotten much more "wordy"”

“AI tool use has ... directly enable[d] most of the improvements coming to AutoPkg 3.0.”

“I wouldn't have had the time/focus to do [some open source work] without agentic assistance.”

“AI tool use has been more positive than negative for me”

“much less tedious now”

technical debt
cognitive debt
community debt



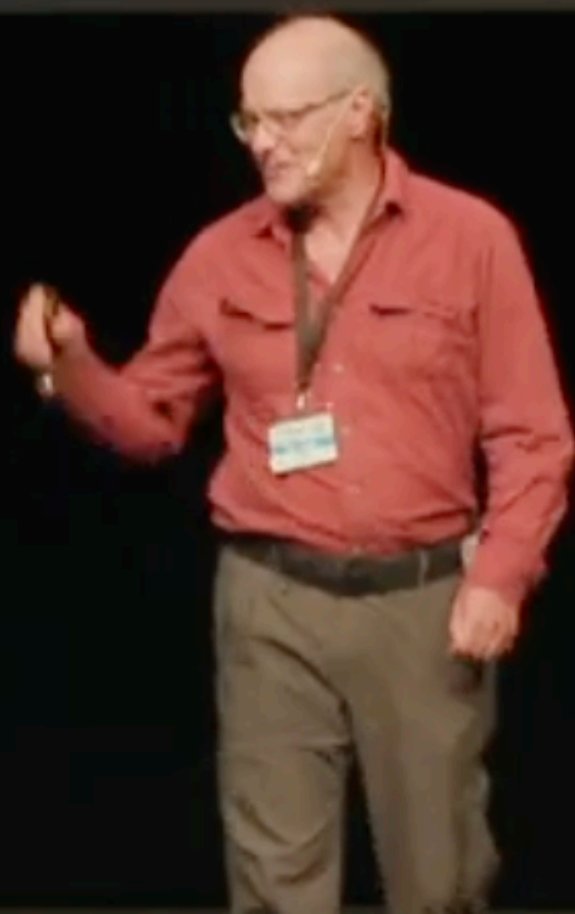
What can/should you do?



Is AI going to take over your job?

Tim Standing

`var tim = Position(company: nil title: nil)`



Discuss your idea

Keep it small

Use plan mode



Claude Code v2.1.222

Sonnet 4.6 · Amazon Bedrock
/Users/Shared/munki-git/munki

```
> /plan
```

```
└─ Enabled plan mode
```

```
> █
```

```
▮ plan mode on (shift+tab to cycle) · ← for agents
```

One more example



Search

Software

Categories

My Items

Updates



Help | Support | Change your passwords | All Categories | What's new in macOS Tahoe | Apple
| Disney Animation Web

All items



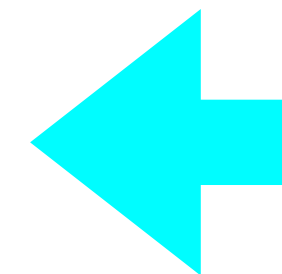
1Password
Utilities - AgileBits Inc.
Installed

REMOVE



1Password for Safari
Utilities - AgileBits Inc.

INSTALL



Adobe Reader
Productivity - Adobe

INSTALL



Amazon WorkSpaces
Remote Access - Amazon

INSTALL



Search

Software

Categories

My Items

Updates

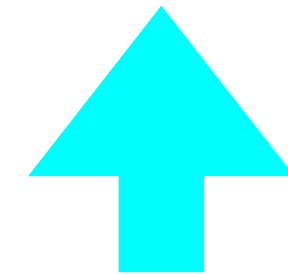


1Password for Safari

AgileBits Inc.

1Password extension for Safari.

INSTALL



Information

Developer
AgileBits Inc.

Size
120.7 MB

Category
Utilities

Version
8.12.37

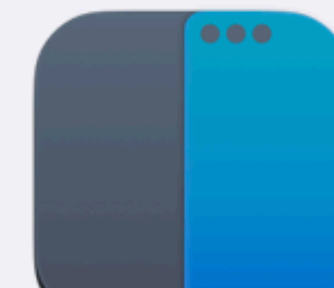
Status
Not installed

More in Utilities

See All



Epic Games Launcher
Utilities - Epic Games, Inc.



Rectangle
Utilities - Knollsoft



Claude Code v2.1.222

Sonnet 4.6 · Amazon Bedrock
/Users/Shared/munki-git/munki

> /plan

└ Enabled plan mode

> Help me change the button appearance in Managed Software Center so that if it says "Install" the button is green, and if it says "Remove", the button is red

■ plan mode on (shift+tab to cycle)


```
Center/Controllers/MainWindowController+WKScriptMessageHandler.swift)
Read(code/apps/MunkiStatus/MunkiStatus/MunkiStatusViewController.swift)
... +24 tool uses (ctrl+o to expand)
(ctrl+b to run in background)
```

* **Fluttering...** (36s • ↓ 2.1k tokens)

└ Tip: Running multiple Claude sessions? Use /color and /rename to tell them apart at a glance.

> █

▮ plan mode on (shift+tab to cycle) • esc to interrupt • ← for agents • ↓ to manage
● high • /effort

● **main**

○ Explore Explore MSC button UI code

1m 4s • ↓ 65.7k tokens

Here is Claude's plan:

Plan: Color-code Install/Remove buttons in MSC

Context

The Managed Software Center action buttons currently use a single blue color for all enabled states. The goal is to make the "Install" button green and the "Remove" button red to give users a clearer visual signal about the action they're about to take.

Architecture

MSC's UI is web-based (WKWebView). The action buttons are HTML/CSS elements. Every button inner `div` already receives the item's `${status}` string as a CSS class — e.g. `not-installed` when the button shows "Install", and `installed` when it shows "Remove". There is no Swift code change needed; this is a pure CSS change.

Button types:

- **Large** (detail page): `div.msc-button-inner large not-installed` – solid filled pill, white text on blue
- **Small** (list views, my-items): `div.msc-button-inner not-installed` – light gray pill with blue text

Verification

1. Build and run MSC in Xcode (or use the existing binary).
 2. Navigate to Software — find an uninstalled optional item → large detail button and small list button should be green.
 3. Navigate to My Items — find an installed removable item → buttons should be red.
 4. Navigate to Updates — Update buttons should remain blue.
 5. Verify hover/active states respond with appropriate darker shades.
 6. Toggle dark mode (System Preferences → Appearance) and confirm colors remain visible and accessible in both modes.
-

Claude has written up a plan and is ready to execute. Would you like to proceed?

- › 1. Yes, auto-accept edits
- 2. Yes, manually approve edits
- 3. Tell Claude what to change
 - shift+tab to approve with this feedback

ctrl+g to edit in Vim · ~/.claude/plans/help-me-change-the-calm-puddle.md

File modified: `code/apps/Managed Software Center/Managed Software Center/Resources/WebResources/base.css`

1. **CSS variables added** — six new variables in both the light mode (`html {}`) and dark mode (`html[data-theme='dark'] {}`) blocks for green install and red remove colors.
2. **New CSS rules added** after the existing large button block:
 - `div.msc-button-inner.large.not-installed` → green background (overrides the default blue)
 - `div.msc-button-inner.large.installed` → red background
 - `div.msc-button-inner.not-installed:not(.large)` → green text on the small pill buttons
 - `div.msc-button-inner.installed:not(.large)` → red text on the small pill buttons
 - Active/hover variants for each

The `status` class is already injected into every button's `div` by the HTML templates (`${status}`), so no Swift or template changes are needed. Update buttons (`update-available`) keep their existing blue color untouched.

* Cooked for 3m 1s

* **recap:** Added CSS to `base.css` to make Install buttons green and Remove buttons red in Managed Software Center, covering both large detail-page buttons and small list-view buttons in light and dark mode. Next step is to build and test in Xcode. (disable recaps



Search

Software

Categories

My Items

Updates



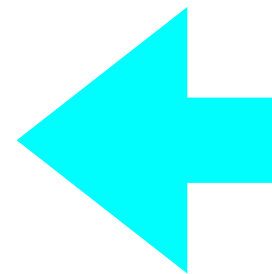
Help | Support | Change your passwords | All Categories | What's new in macOS Tahoe | Apple
| Disney Animation Web

All items



1Password
Utilities - AgileBits Inc.
Installed

REMOVE



1Password for Safari
Utilities - AgileBits Inc.

INSTALL



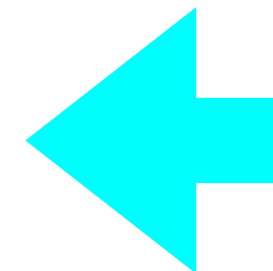
Adobe Reader
Productivity - Adobe

INSTALL



Amazon WorkSpaces
Remote Access - Amazon

INSTALL





Search

Software

Categories

My Items

Updates



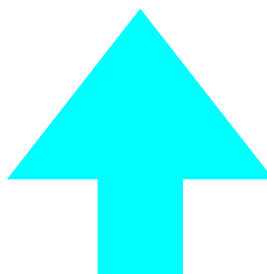
1Password

AgileBits Inc.

The world's most-loved password manager

1Password is the easiest way to store and use strong passwords. Log in to sites and fill forms securely with a single click.

REMOVE



Information

Developer
AgileBits Inc.

Size
375.2 MB

Category
Utilities

Version
8.12.36

Status
Installed

More in Utilities

See All



Epic Games Launcher
Utilities - Epic Games, Inc.



Spectacle
Utilities - Eric Czarny



Search

 Software

 Categories

 My Items

 Updates

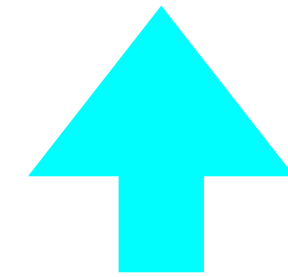


1Password for Safari

[AgileBits Inc.](#)

1Password extension for Safari.

INSTALL



Information

Developer
AgileBits Inc.

Size
120.7 MB

Category
Utilities

Version
8.12.37

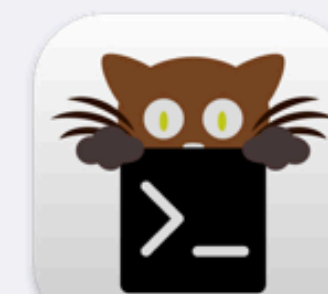
Status
Not installed

More in Utilities

[See All](#)



Aspera Connect
Utilities - IBM Corp



Kitty
Utilities - Kovid Goyal

Don't be afraid

Summary

AI code assist

Potential positives

- Help for maintainers/contributors
 - Code generation
 - Code review
 - Rapid experimentation
 - Debugging assistance
 - Test writing
- Encouraging new contributors
 - Including **you!**

Potential negatives

- Too-rapid pace of change
- Too many Pull Requests
- Skill atrophy
- Implementing code you don't actually understand
- General concerns about AI:
 - Impact on environment
 - Impact on employment
 - Long-term costs

Tack!