FIELD NOTES

GitHub API Field Guide

The GitHub API stops working quietly: a list that returns thirty items, a webhook failing for a month, a 403 with no Retry-After. Every script here is read only.

24 fixes, each one a written note with diagrams and a small Python and Node.js script that finds the problem and repairs it. All 24 have a script in the public repo.

Before you run anything. Every script starts in a dry run: it reports what it would change and writes nothing. Read that output first, then set `DRY_RUN=false` when you are satisfied it has understood your data.

Scripts: github.com/allanninal/github-api-fixes
Guides: allanninal.dev/github
All 14 repos: github.com/allanninal

The fixes

1 **a permission error is disguised as 404 Not Found**

The repository is open in a browser tab in front of you. The script asks for the same repository and gets 404 {"message":"Not Found"}. Somebody checks the spelling, then the owner name, then the case of both, then writes a ticket saying the API is broken. The API is not broken. It is refusing to tell you that the repository exists, because telling you would leak the existence of a private repository to a credential that has no business knowing about it.

`github_404_triage.py`

https://www.allanninal.dev/github/404-masking-403/

https://github.com/allanninal/github-api-fixes/tree/main/404-masking-403

---

2 **resource not accessible by integration on one endpoint**

Nineteen endpoints work. The twentieth returns 403 {"message":"Resource not accessible by integration"}, which names no permission, no resource and no level, and reads like a platform bug rather than a configuration one. It is the one failure in this cluster where GitHub actually tells you the answer: it is in the x-accepted-github-permissions response header on that very 403, which almost no HTTP client shows you by default.

`github_app_permission_diff.py`

https://www.allanninal.dev/github/app-permission-missing/

https://github.com/allanninal/github-api-fixes/tree/main/app-permission-missing

---

3 **code search is billed to its own 10 a minute bucket**

The script walks the org, calling GET /search/code once per repository. It gets through nine of them. The tenth returns 403, and GET /rate_limit says you have 4,987 requests left in the hour. Both statements are correct, because the request that was refused was never being counted in the bucket you just looked at.

`github_code_search_budget.py`

https://www.allanninal.dev/github/code-search-bucket-exhausted/

https://github.com/allanninal/github-api-fixes/tree/main/code-search-bucket-exhausted

---

4 **the compare endpoint stops at 250 commits and says nothing**

The release-notes generator diffs v4.2.0...v4.3.0 and produces a changelog that looks entirely plausible. It has 250 entries. The release contains 812 commits, and the ones it dropped are not the boring ones at the end — the shape of what came back is not what the code assumed at all.

`github_compare_truncation.py`

https://www.allanninal.dev/github/compare-250-commit-cap/

https://github.com/allanninal/github-api-fixes/tree/main/compare-250-commit-cap

---

5 **the same webhook URL is registered on the org and the repo**

The bot comments twice on every pull request. Someone checks the receiver for a retry loop, finds none, and blames GitHub for sending duplicates. GitHub is sending exactly one copy of the event to each hook that asked for it — and two hooks asked, one on the repository and one on the organization, both pointing at the same URL.

`github_duplicate_hooks.py`

https://www.allanninal.dev/github/duplicate-webhooks/

https://github.com/allanninal/github-api-fixes/tree/main/duplicate-webhooks

---

6 **rotating the token invalidates every cached ETag at once**

The graph is a sawtooth and it is a very tidy one. Quota consumption sits near zero for fifty-odd minutes, jumps, and settles back, every hour, on the hour. Nothing in the schedule matches. The poller runs every thirty seconds and has done for months. What runs hourly is the installation token.

`github_etag_credential_check.py`

https://www.allanninal.dev/github/etag-invalidated-by-token-rotation/

https://github.com/allanninal/github-api-fixes/tree/main/etag-invalidated-by-token-rotation

---

7 **the installation covers only some repositories, silently**

The scanner reports clean. Every repository it looked at was fine, every check passed, and the summary at the bottom says so. It looked at twelve repositories. The organization has a hundred and forty. Nothing errored, nothing warned, and no line of that report is untrue — the App installation was set to selected repositories at some point in 2023 and the other hundred and twenty-eight have never been inside its field of view.

`github_app_coverage_audit.py`

https://www.allanninal.dev/github/installation-repository-selection-partial/

https://github.com/allanninal/github-api-fixes/tree/main/installation-repository-selection-partial

---

8 **only the first page is read because the Link header is ignored**

The request returned 200. The JSON is a well-formed array of pull requests. Your audit read it, counted 30, and reported that the repository is tidy. There are 340 open pull requests. Nothing failed, nothing was logged, and the number you are now acting on is wrong by an order of magnitude — the answer was complete for one page and the rest was advertised in a header nobody read.

`github_link_header_audit.py`

https://www.allanninal.dev/github/link-header-not-followed/

https://github.com/allanninal/github-api-fixes/tree/main/link-header-not-followed

---

9 **polling without ETags spends full quota on unchanged data**

The dashboard polls eight endpoints every thirty seconds and burns through 5,000 requests before lunch. Almost nothing it fetches has changed. GitHub has been sending an etag on every one of those responses, and every response that comes back 304 Not Modified is free — it does not count against the rate limit at all. This is the one problem in this section where the fix pays for itself in a number you can print.

`github_etag_saving.py`

https://www.allanninal.dev/github/no-conditional-requests/

https://github.com/allanninal/github-api-fixes/tree/main/no-conditional-requests

---

10 **per_page is unset so every list costs 3.3x more requests**

The job is correct. It follows rel="next" to the end, it reads every issue, and it burns through the hourly quota by lunchtime. Nothing is broken and nothing needs debugging — it is simply making three and a third times as many requests as it needs to, because per_page was never set and the default is 30.

`github_per_page_audit.py`

https://www.allanninal.dev/github/per-page-default-30/

https://github.com/allanninal/github-api-fixes/tree/main/per-page-default-30

---

11 **the x-poll-interval header is ignored on events endpoints**

The events consumer polls every five seconds because five seconds felt responsive. It gets the same page back seven hundred times an hour. On every one of those responses GitHub has been returning a header that says how long to wait before the next one, and the client has never read it.

`github_poll_interval_check.py`

https://www.allanninal.dev/github/poll-interval-header-ignored/

https://github.com/allanninal/github-api-fixes/tree/main/poll-interval-header-ignored

---

12 **the integration polls for events a webhook would push**

The integration works. It notices new pull requests, it picks up comments, it has never lost anything. It also makes 4,320 requests an hour to do it, notices each of those events an average of thirty seconds after it happened, and would notice nothing at all if the poll ran once a day instead. There is no bug here. There is a design that was never revisited.

`github_webhook_vs_poll.py`

https://www.allanninal.dev/github/polling-instead-of-webhooks/

https://github.com/allanninal/github-api-fixes/tree/main/polling-instead-of-webhooks

---

13 **core REST quota is exhausted and every call returns 403**

Every endpoint fails at once, with the same status, in the same second. That pattern says outage or bad credentials, and it is neither: the hourly bucket is empty, and an empty bucket refuses everything equally. The awkward part is that by the time you are reading the 403 the interesting question has already gone past. Not are we out, but at what rate did we spend it, and would that rate have fitted.

`github_quota_forecast.py`

https://www.allanninal.dev/github/rate-limit-core-exhausted/

https://github.com/allanninal/github-api-fixes/tree/main/rate-limit-core-exhausted

---

14 **requests go out anonymous and are capped at 60 an hour**

It works on the first run and dies on the fourth. On a laptop it survives a couple of minutes; in CI, behind a shared NAT address, it is refused almost immediately and the run before yours gets the blame. Nothing in the code is wrong. The token simply is not arriving, and GitHub does not treat that as an error. It serves you anyway, as a stranger, from a bucket sixty deep.

`github_auth_tier_check.py`

https://www.allanninal.dev/github/rate-limit-unauthenticated/

https://github.com/allanninal/github-api-fixes/tree/main/rate-limit-unauthenticated

---

15 **the client ignores retry-after and keeps hammering the API**

The log shows four hundred consecutive 403s, one second apart, for eleven minutes. The retry logic is working perfectly: it catches the error, waits, tries again, never gives up. GitHub said retry-after: 120 on the very first one, and the client threw that header away along with the rest of the response.

`github_backoff_plan.py`

https://www.allanninal.dev/github/retry-after-ignored/

https://github.com/allanninal/github-api-fixes/tree/main/retry-after-ignored

---

16 **org lists silently omit SSO-enforced organizations**

The user belongs to six organizations. GET /user/orgs returns four. Status 200, valid JSON, no errors array, nothing in the body that hints anything is absent. The two organizations that enforce SAML single sign-on for a token that was never authorized against them are simply not in the list, and the only place GitHub mentions it is a response header called X-GitHub-SSO.

`github_sso_partial_results.py`

https://www.allanninal.dev/github/saml-partial-results/

https://github.com/allanninal/github-api-fixes/tree/main/saml-partial-results

---

17 **search returns at most 1,000 results whatever total_count says**

The response says "total_count": 24831. You page through it at 100 per request, and on page 11 the API returns 422 Validation Failed with "Only the first 1000 search results are available". The count was true. The results were never there to fetch — total_count describes the match set, not the part of it you are allowed to page through.

`github_search_cap_audit.py`

https://www.allanninal.dev/github/search-1000-result-cap/

https://github.com/allanninal/github-api-fixes/tree/main/search-1000-result-cap

---

18 **search has its own 30-per-minute bucket and drains separately**

The job loops over four hundred repositories and runs one search in each. Around the thirtieth it starts returning 403, and the part that makes no sense is that everything else keeps working: the repository reads in the same loop, on the same token, in the same second, are fine. Two buckets, and only one of them is empty. The error message does not mention which.

`github_search_budget.py`

https://www.allanninal.dev/github/search-bucket-exhausted/

https://github.com/allanninal/github-api-fixes/tree/main/search-bucket-exhausted

---

19 **over 100 concurrent requests trips a secondary rate limit**

Someone replaces a for loop with a Promise.all and the job goes from nine minutes to forty seconds, once. The next run returns 403 on two thirds of the requests. You check the quota, because a 403 from GitHub means the quota, and the quota says four thousand eight hundred requests left. Both numbers are true. They are about different limits.

`github_concurrency_probe.py`

https://www.allanninal.dev/github/secondary-limit-concurrency/

https://github.com/allanninal/github-api-fixes/tree/main/secondary-limit-concurrency

---

20 **bulk issue or comment creation exceeds 80 requests a minute**

The migration script imports 2,400 issues from the old tracker. It gets through about eighty of them, then every remaining call comes back 403. You check the quota and there are 4,900 requests left in it. The limit that stopped you is not counting requests. It is counting the things you created.

`github_content_burst_audit.py`

https://www.allanninal.dev/github/secondary-limit-content-creation/

https://github.com/allanninal/github-api-fixes/tree/main/secondary-limit-content-creation

---

21 **a hot endpoint burns 900 points a minute and gets throttled**

One endpoint in the job keeps failing and the rest are fine. It fails for about a minute, recovers, and fails again twenty minutes later. The hourly quota is barely touched and the concurrency is one, because someone already serialised it after the last incident. What is left is the cap nobody budgets for: not how many requests you make, but how much work you asked one path to do inside sixty seconds.

`github_endpoint_cost_audit.py`

https://www.allanninal.dev/github/secondary-limit-points-per-minute/

https://github.com/allanninal/github-api-fixes/tree/main/secondary-limit-points-per-minute

---

22 **webhook deliveries are failing and nobody reads the log**

Someone opens a pull request and the bot that should comment on it says nothing. You grep your receiver's logs for the last hour and find no request at all, which points the finger at GitHub. It is not GitHub. The delivery happened, your server answered 502, and GitHub wrote that down in a log you have never opened.

`github_hook_delivery_audit.py`

https://www.allanninal.dev/github/webhook-deliveries-failing/

https://github.com/allanninal/github-api-fixes/tree/main/webhook-deliveries-failing

---

23 **the hook is not subscribed to the event you are waiting for**

The handler was written, reviewed, unit tested against a saved payload, and deployed. In production it has never executed once. There is no error anywhere because there is no failure anywhere: the hook was created years ago with push and pull_request, and the event your handler waits for has never been sent to it.

`github_hook_event_coverage.py`

https://www.allanninal.dev/github/webhook-event-not-subscribed/

https://github.com/allanninal/github-api-fixes/tree/main/webhook-event-not-subscribed

---

24 **a webhook with no secret sends no signature to verify**

The receiver has a signature check. It reads X-Hub-Signature-256, computes an HMAC over the body, compares in constant time, and returns 401 when they differ. It has never returned 401, because the hook it serves has no secret, so GitHub does not send the header, and the check quietly skips itself on every request.

`github_hook_secret_audit.py`

https://www.allanninal.dev/github/webhook-no-secret/

https://github.com/allanninal/github-api-fixes/tree/main/webhook-no-secret

---