Calibre-web

- Unable to fetch metadata

Unable to fetch metadata

A permanently spinning “loading” state in the Fetch Metadata dialog usually means the backend request to the metadata provider failed or never returned. The container logs and your browser’s network tab will show which. Here’s how I’d narrow it down.

1. Check the container log while you search

bash

```
docker logs -f --tail 100 calibre-web
```

Run a search in the UI and watch for tracebacks. Typical hits are `ConnectionError`, `Max retries exceeded`, `Temporary failure in name resolution`, or a `ModuleNotFoundError` (a missing Python dependency in the image). If you want more detail, raise the log level under Admin → Edit Basic Configuration → Logfile Configuration → Log Level: Debug.

2. Test outbound connectivity from inside the container

bash

```
docker exec -it calibre-web sh -c 'python3 - <<EOF
import requests
for u in ["https://www.googleapis.com/books/v1/volumes?q=test",
          "https://comicvine.gamespot.com"]:
    try:
        print(u, requests.get(u, timeout=10).status_code)
    except Exception as e:
        print(u, "ERR", e)
EOF'
```

If this fails, the problem is Docker networking or DNS. Fixes include setting DNS in your compose file (`dns: [1.1.1.1, 8.8.8.8]`), checking the host firewall or `iptables`/`ufw` forwarding rules, and checking whether an IPv6-only resolution path is breaking things. An empty `/etc/resolv.conf` or a broken Docker bridge after a firewall change is a common cause on Ubuntu.

3. Check the browser side (F12 → Network)

When you click search, look for the POST to `/metadata/search`:

- **Pending forever or 504:** a reverse proxy timeout or buffering issue. For nginx, make sure the proxy passes `X-Forwarded-*` headers and has a reasonable `proxy_read_timeout`.
- **500:** the backend error will be in the container log.
- **No request at all:** a JavaScript error in the Console tab. Try a hard refresh or a private window, since a cached old JS file after an upgrade can break the dialog.

4. Provider-specific causes

- **Google Scholar** is frequently blocked or rate-limited, because it returns captchas to servers. Untick it in the dialog.
- **Google Books** can fail or be throttled from some IP ranges or without an API key. Try it alone.
- **ComicVine** needs an API key configured.
- Providers are toggled with the checkboxes in the Fetch Metadata dialog itself (admin account required). Try one provider at a time to see which one hangs the whole request.

5. Update the image

Metadata providers change often, and old versions break when upstream sites change.

bash

```
docker compose pull && docker compose up -d
```

If you use the linuxserver image, also check its release notes for known issues with the version you're on.

If you paste the relevant log lines from step 1, plus how Calibre-Web is exposed (direct port or behind a reverse proxy), I can pinpoint it.

Good news: networking and DNS are fine. The container reaches both providers, and the logs show no tracebacks. The two status codes tell you what is going on:

- **Google Books returns 429 (Too Many Requests).** Google rate-limits unauthenticated requests per IP, and server/shared IPs often hit that limit quickly. Calibre-Web's Google provider sends keyless requests, so it can fail constantly from your IP.
- **ComicVine returns 403 on the bare homepage.** That's partly expected, since the site blocks generic bot requests and the real API needs a key. It isn't conclusive on its own, but it's another provider that won't work out of the box.

The endless spinner most likely happens because Calibre-Web queries all enabled providers at once, and a provider that gets blocked or hangs can stall the whole dialog.

What to try, in order:

1. **Untick all providers in the Fetch Metadata dialog except one**, and test each separately. Open Edit Metadata on a book, click Fetch Metadata, and search with only one box checked. Try Amazon, Douban, Lubimyczytac, etc., not just Google. Whichever one answers is your working provider; leave the others off.
2. **Check what the browser sees.** Press F12, open the Network tab, run a search, and look at the `/metadata/search` request. If it shows pending, a 500, or a 504, tell me which. A pending request with a reverse proxy in front (your `lib.meson.in` domain suggests nginx or similar) points to a proxy timeout. If there's no request at all, check the Console tab for a JS error.
3. **Test whether it's your IP or just the endpoint.** From the host (not the container):

bash

```
curl -s -o /dev/null -w "%{http_code}\n" "https://www.googleapis.com/books/v1/volumes?q=test
```

Also wait a few minutes and retry. A 429 that clears means you were briefly throttled; one that persists means your IP is flagged.

4. **See the real error with debug logging.** Admin → Edit Basic Configuration → Logfile Configuration → set Log Level to Debug, then search once and read the log (`docker logs --tail 50 lib.meson.in`). The earlier tail only showed startup because no search had run yet, and the `^Z` suspended the command rather than ending it. Run `fg` then Ctrl+C to clean that up, or kill the stopped job with `kill %1`.
5. **If it's a persistent Google block**, a Google Books API key would raise the quota, but stock Calibre-Web has no setting for it. The more practical route is to use providers that don't depend on Google, or to fetch metadata with the Calibre desktop app (`ebook-meta` / the metadata download there) and let Calibre-Web just read the library.

Also make sure the container is current:

bash

```
cd /home/web-data/lib.meson.in
docker compose pull && docker compose up -d
```

Send me what the Network tab shows for `/metadata/search` and the debug log lines from a single search, and I can say whether this is a proxy issue or purely provider blocking.

The 429 from the host itself means this isn't a Docker problem. Google is throttling your server's IP (or IP range), so the container has nothing to fix on its side.

Check whether IPv6 is the culprit:

bash

```bash
curl -4 -s -o /dev/null -w "v4: %{http_code}\n" "https://www.googleapis.com/books/v1/volumes?q=
curl -6 -s -o /dev/null -w "v6: %{http_code}\n" "https://www.googleapis.com/books/v1/volumes?q=
```

Google often throttles whole IPv6 ranges at once. If v4 returns 200 and v6 returns 429, you can force IPv4 for the container. Either disable IPv6 in Docker, or add this to the compose service:

yaml

```yaml
sysctls:
  - net.ipv6.conf.all.disable_ipv6=1
```

Then run `docker compose up -d`.

If both return 429, your IP is flagged or the anonymous quota for it is exhausted. That’s common on hosting and datacenter IPs, and it may clear on its own after some hours, but don’t count on it.

Google also isn’t necessarily the cause of the infinite spinner. A 429 is returned quickly, so Calibre-Web should show an error or an empty result rather than spin forever. Something else is probably hanging, such as another provider or the reverse proxy. So still do this:

1. In the Fetch Metadata dialog, untick Google, then test one other provider at a time.
2. Watch the Network tab (F12) for the `/metadata/search` request. Is it pending, 500, or 504?
3. Set the log level to Debug and read `docker logs --tail 50 lib.meson.in` right after one search.

Longer-term options if Google stays blocked:

- Route Calibre-Web’s outbound traffic through a different IP (a proxy or VPN container). That works, but it’s more setup.
- Use non-Google providers only.
- Use Calibre desktop’s metadata download on another machine, then let Calibre-Web serve the library.
- Patch `cps/metadata_provider/google.py` inside the container to add a Google Books API key (free from Google Cloud Console, which raises the quota a lot). The patch is lost on every image update unless you bind-mount the modified file over the original.

Let me know what the IPv4/IPv6 test shows and what the Network tab says for `/metadata/search`.

That settles the IPv6 question. IPv6 isn’t throttled, it simply doesn’t work from this host (`000` means the connection never completed). Only IPv4 reaches Google, and it gets a 429. So Google is throttling your server’s public IPv4 address, and no container setting will change that. Don’t bother with the IPv6 sysctl change.

Test whether an API key fixes it. Keyed requests get a separate quota that isn't tied to the shared per-IP limit. Create a free key in Google Cloud Console (enable the "Books API", then create an API key) and test:

bash

```
curl -s -o /dev/null -w "%{http_code}\n" "https://www.googleapis.com/books/v1/volumes?q=test&ke
```

If that returns 200, you can patch Calibre-Web to use it. Find the URL in the provider file:

bash

```
docker exec lib.meson.in grep -n "googleapis" /app/calibre-web/cps/metadata_provider/google.py
```

Paste the output and I'll give you the exact edit. To keep it across image updates, copy the file out, edit it, and bind-mount it back over the original in your compose file:

bash

```
docker cp lib.meson.in:/app/calibre-web/cps/metadata_provider/google.py ./google.py
```

yaml

```
volumes:
  - ./google.py:/app/calibre-web/cps/metadata_provider/google.py:ro
```

The spinner is probably a separate problem. A quick 429 should produce an empty result, not an endless loading state, so I suspect a different provider hangs the combined request, or the reverse proxy. These two checks will settle it:

1. In the Fetch Metadata dialog, untick Google and test one other provider at a time (Amazon, Douban, Lubimyczytac, etc.).
2. Press F12, open the Network tab, run a search, and tell me what `/metadata/search` shows: pending, 500, 504, or nothing at all.

Also set the log level to Debug and run `docker logs --tail 50 lib.meson.in` right after a single search. That will show whether any provider throws an error or times out.

That 400 is expected, because `YOUR_KEY` was a literal placeholder and Google rejected it as an invalid key. The test hasn't happened yet, so create a real key first (Google Cloud Console → enable "Books API" → Credentials → Create API key, then restrict it to the Books API) and rerun it:

bash

```
curl -s -o /dev/null -w "%{http_code}\n" "https://www.googleapis.com/books/v1/volumes?q=test&ke
```

If that returns 200, make the patch. Line 39 only defines the base URL, and the query is appended where the request is made. Check that line first:

bash

```
docker exec lib.meson.in grep -n "requests.get" /app/calibre-web/cps/metadata_provider/google.p
```

I expect something like `results = requests.get(Google.SEARCH_URL + query)`. If it matches, patch it:

bash

```
cd /home/web-data/lib.meson.in
docker cp lib.meson.in:/app/calibre-web/cps/metadata_provider/google.py ./google.py
sed -i 's|requests.get(Google.SEARCH_URL + query)|requests.get(Google.SEARCH_URL + query + "\&k
grep -n "requests.get" google.py   # verify the edit
```

Then add the bind mount to your compose file under the service's `volumes:`:

yaml

```
      - ./google.py:/app/calibre-web/cps/metadata_provider/google.py:ro
```

Apply it with `docker compose up -d`.

The `timeout=10` matters for your original symptom. If the provider call has no timeout, a stalled connection can hang the whole search and leave the dialog spinning forever. You may want to add the same `timeout=10` to the `requests.get` calls in the other provider files you use.

Keep the key restricted to the Books API in the Cloud Console, since it sits in plain text in that file. After an image update, check that the upstream `google.py` hasn't changed much, because the mount will override it.

If the keyed test still returns 429, tell me and we'll go with non-Google providers or routing the container through a different IP. Either way, check the F12 Network tab on `/metadata/search` and the Debug log after one search, because I still think the spinner and the 429 may be separate problems.

The API restriction dropdown only lists APIs that are **enabled in the project**, so the Books API is most likely not enabled yet.

Fix:

1. In Google Cloud Console, make sure the right project is selected (top bar).
2. Go to **APIs & Services → Library**, search for **“Books API”** (the Google Books API), and click **Enable**.
3. Go back to **APIs & Services → Credentials**, open your key, and under **API restrictions** choose **Restrict key**. “Books API” should now appear in the list. If it doesn’t, reload the page, as the list can take a few minutes to refresh.

The key also won’t work for requests until the API is enabled. A key from a project without the Books API enabled usually returns a 403 with a “has not been used in project... or it is disabled” message.

If you want to test first, you can leave the key unrestricted, run the curl test with the real key, and add restrictions afterwards. Don’t leave it unrestricted long-term, since it sits in plain text in `google.py`.

Alternative restriction: if the API list still won’t show the Books API, use **Application restrictions → IP addresses** and enter your server’s public IPv4 address. Since the key lives only on your server, this is arguably the stronger restriction anyway. You can find your public IP with:

bash

```bash
curl -4 -s https://ifconfig.me
```

Then rerun the test with the real key:

bash

```bash
curl -s -o /dev/null -w "%{http_code}\n" "https://www.googleapis.com/books/v1/volumes?q=test&ke
```

A 200 means you can go ahead with the patch from my last message. If you get 403, paste the response body (drop the `-o /dev/null -w ...` options) and I’ll tell you which setting is

The key works, so now patch it in. I haven’t seen the exact `requests.get` line yet, so check it first:

bash

```bash
cd /home/web-data/lib.meson.in
docker cp lib.meson.in:/app/calibre-web/cps/metadata_provider/google.py ./google.py
grep -n "requests.get\|SEARCH_URL" google.py
```

If the output includes `requests.get(Google.SEARCH_URL + query)` (or very close to it), run:

bash

```
sed -i 's|requests.get(Google.SEARCH_URL + query)|requests.get(Google.SEARCH_URL + query + "\&k
grep -n "requests.get" google.py   # confirm the edit landed
```

Replace `REAL_KEY` with your key. If the line looks different, paste the grep output and I'll adjust the `sed`.

Then add the bind mount under the service's `volumes:` in your compose file:

yaml

```
      - ./google.py:/app/calibre-web/cps/metadata_provider/google.py:ro
```

Apply and test:

bash

```
docker compose up -d
docker logs --tail 30 lib.meson.in
```

Then open Edit Metadata on a book, click Fetch Metadata, and search with only Google ticked.

Two things to keep in mind:

- Protect the key. Keep `google.py` out of any web-served directory or git repo, and restrict the key to the Books API or to your server's public IPv4 in the Cloud Console.
- If Google works but the spinner still appears with other providers ticked, one of those providers is hanging. In that case, send me the F12 Network result for `/metadata/search` and the Debug log, and I'll help find which one.

Your `google.py` already supports an API key natively, so the `sed` had nothing to match and no patch is needed. The line `API_KEY = "&key=" + config.config_googlebooks_api_key` reads the key from Calibre-Web's own settings.

Set the key in the UI:

1. Go to Admin → **Edit Basic Configuration** → **Feature Configuration**.
2. Paste your key into the **Google Books API Key** field and save.
3. Restart the container, because `API_KEY` is evaluated once when the module loads, so a new key isn't picked up until then:

bash

```
cd /home/web-data/lib.meson.in
docker compose restart
```

If you don’t see the field under Feature Configuration, your image is older than this code, so run `docker compose pull && docker compose up -d`.

Undo the bind mount, if you added it. Remove the `./google.py:/app/...:ro` line from your compose file and delete the local `google.py` copy. Otherwise an old copy will override future image updates for no benefit. Then run `docker compose up -d`.

One real problem remains in this file: `requests.get(...)` has no timeout. If Google, or the connection to it, stalls, the request can hang indefinitely, and a hang like that fits your endless spinner. Everything else in the file handles failures properly: a 429 just logs a warning and returns an empty list. To add a timeout you do need the bind mount, since the file lives inside the image:

bash

```
sed -i 's|requests.get(Google.SEARCH_URL + query + Google.API_KEY)|requests.get(Google.SEARCH_U
grep -n "requests.get" google.py
```

Then keep the `./google.py:/app/calibre-web/cps/metadata_provider/google.py:ro` mount and run `docker compose up -d`. Since the key now comes from the UI, `google.py` no longer contains your secret.

Test: open Fetch Metadata on a book with only Google ticked. If results appear, the key works. If the spinner still shows up when other providers are ticked, one of them is hanging, so send me the F12 → Network status for `/metadata/search` and the Debug log lines from one search.

That error is unrelated to Google. Calibre-Web validates every field on the Basic Configuration page when you save, so a stale **Kepubify path** blocks the whole save, including your API key.

Fix it in the UI:

1. On the same page, find **External binaries**, usually just below Feature Configuration. Look for “Path to Kepubify E-Book Converter”.
2. Either clear that field, or set it to a path that exists in the container (see below).
3. Save again. The Google Books API key should save now.

To find the correct path (if you want Kepubify for Kobo conversion):

bash

```
docker exec lib.meson.in sh -c 'which kepubify; ls -l /usr/bin/kepubify /usr/local/bin/kepubify
```

If one of those exists, enter that path in the field. If none do, your image doesn't include it, so leave the field empty. You only need it if you sync books to Kobo devices.

If the field is hidden or won't clear, check the same page for other binary paths that also fail validation, such as the Calibre `ebook-convert` path or the unrar path, since any one of them can produce a similar error. Clear or correct whichever one the message names.

Restart after saving, since `google.py` reads the key at import time:

bash

```
cd /home/web-data/lib.meson.in
docker compose restart
```

Then test Fetch Metadata with only Google ticked. If another error appears when saving, send me the exact text.

Kepubify is installed at `/usr/bin/kepubify` and is executable, so the error comes from the path stored in the settings. It is probably empty, outdated, or points somewhere else.

1. Open Admin → **Edit Basic Configuration** → **External binaries**.
2. Set **Path to Kepubify E-Book Converter** to exactly this, with no trailing space:

```
/usr/bin/kepubify
```

3. Fill in your Google Books API key in the same form and save.

If the error persists with that path, clear the Kepubify field completely and save. You only need it for Kobo sync, and you can add it back later.

To confirm the path works from the container's side, you can run:

bash

```
docker exec lib.meson.in /usr/bin/kepubify --version
```

After a successful save, restart so `google.py` picks up the key:

bash

```
cd /home/web-data/lib.meson.in
docker compose restart
```

Then test Fetch Metadata with only Google ticked. If the save shows a different error, send me the exact text.

The binary exists and is executable, so the error is coming from what's saved in the settings. These steps check what's stored and, if needed, bypass the form entirely.

1. Check what Calibre-Web has stored (read-only):

bash

```
docker exec lib.meson.in python3 -c "
import sqlite3
c = sqlite3.connect('/config/app.db')
cur = c.execute('select * from settings')
cols = [d[0] for d in cur.description]
row = cur.fetchone()
for k, v in zip(cols, row):
    if any(s in k for s in ('kepub', 'binar', 'convert', 'rar', 'googlebooks')):
        print(k, '=', repr(v))
"
```

If the container's config is mounted somewhere other than `/config`, adjust the path. If `config_kepubifypath` or a binaries directory setting holds a stale path, that's your answer.

2. Test that the app user can run it (the earlier `ls` only proved the file is executable, not that it works for that user):

bash

```
docker exec -u abc lib.meson.in /usr/bin/kepubify --version
```

If it prints a version, the path is good.

3. Fix it in the UI. If a "Binaries directory" or "Path to Calibre Binaries" field is filled in, clear it or correct it, since a wrong value there can also trigger this error. Set the Kepubify path to `/usr/bin/kepubify`, or leave it empty, and save with the API key.

4. If the form still refuses to save, write the key directly to the database. Replace `YOUR_KEY`:

bash

```bash
docker compose stop
docker run --rm -v "$(pwd)/config:/config" python:3-alpine python3 -c "
import sqlite3
c = sqlite3.connect('/config/app.db')
c.execute(\"update settings set config_googlebooks_api_key='YOUR_KEY', config_kepubifypath=''\"
c.commit()
"
docker compose start
```

This assumes your compose maps `./config` to `/config`. Check the `volumes:` line in your compose file and adjust the host path if it differs. Back up `app.db` first (`cp config/app.db config/app.db.bak`), because this edits the live database. It also clears the Kepubify path, which you only need for Kobo sync.

Send me the output of step 1 and step 2 if this doesn’t resolve it. Then I can tell you exactly which value is tripping the check.

The stored Kepubify path is already correct (`/usr/bin/kepubify`), so the error comes from the form submission rather than the saved config, probably a stray value or whitespace in that field. You don’t need the form to save the key, so write it straight to the database and leave everything else alone.

bash

```bash
cd /home/web-data/lib.meson.in
docker exec lib.meson.in cp /config/app.db /config/app.db.bak
docker exec -e GKEY='PASTE_YOUR_KEY_HERE' lib.meson.in python3 -c "
import sqlite3, os
c = sqlite3.connect('/config/app.db')
c.execute('update settings set config_googlebooks_api_key=?', (os.environ['GKEY'],))
c.commit()
print(c.execute('select length(config_googlebooks_api_key) from settings').fetchone())
"
docker compose restart
```

The printed number is the key’s length. It should be about 39 characters for a Google API key, which confirms it was written. Don’t blank out `config_kepubifypath` this time, since it’s fine as it is. Restart right after the write, because Calibre-Web caches settings in memory and could overwrite the change on its next save.

Then test Fetch Metadata with only Google ticked.

Side note on the form error: the other suspicious value in your output is `config_rarfile_location = '/usr/bin'`. That field expects the path to the `unrar` executable, not a directory, so it’s worth checking:

bash

```
docker exec lib.meson.in which unrar
```

If it prints a path, enter that path in the field. If it prints nothing, clear the field. That likely won’t fix the Kepubify message, but a wrong value there can break saving. If you want to dig into the form error later, open F12 → Network, save the form, and look at the request payload for `config_kepubifypath` to see exactly what the browser sends.

If Google still returns nothing after the restart, run one search and send me `docker logs --tail 30 lib.meson.in`.

That length of 19 means the placeholder text itself was saved: `PASTE_YOUR_KEY_HERE` is exactly 19 characters. A real Google API key is about 39 characters, usually starting with `AIza`. Run the command again with your real key in place of the placeholder:

bash

```
docker exec -e GKEY='AIzaSy...your_real_key...' lib.meson.in python3 -c "
import sqlite3, os
c = sqlite3.connect('/config/app.db')
c.execute('update settings set config_googlebooks_api_key=?', (os.environ['GKEY'],))
c.commit()
print(c.execute('select length(config_googlebooks_api_key) from settings').fetchone())
"
```

The printed length should now be about 39. Then restart the container:

bash

```
docker compose restart
```

If you don’t want the key in your shell history, put a space before the command (works if `HISTCONTROL=ignorespace`, the Ubuntu default), or run `history -d` on that entry afterwards.

Once restarted, open Fetch Metadata on a book with only Google ticked. If nothing comes back, run one search and send me `docker logs --tail 30 lib.meson.in`.

That means the database write worked and Calibre-Web reads the key. You don't need the form to save anything else, so you can ignore the Kepubify error unless you want to change a setting later. Saving fails before anything is written, so it hasn't undone the key.

Test it now:

1. Open a book, then Edit Metadata → Fetch Metadata.
2. Untick every provider except Google Books, then search with a title.

If results show up, the 429 problem is solved. If you get nothing, run one search and send me:

bash

```
docker logs --tail 30 lib.meson.in
```

A warning line from `google.py` (such as a 403 or 429) would show what Google is returning.

If Google works but the spinner returns with all providers ticked, one of the other providers is hanging. Untick them all, tick one at a time, and note which one makes it spin. Then disable that one. Since you're an admin, the toggle in the dialog is saved, so that fixes it permanently.

Fixing the Kepubify save error (optional, for later): run this check, since the earlier `ls` only showed the file is executable, not that the app user can run it:

bash

```
docker exec -u abc lib.meson.in /usr/bin/kepubify --version
```

If that fails, the path works for root but not for the app user, and that's what Calibre-Web is complaining about. If it prints a version, open F12 → Network, save the form, and check the request payload for `config_kepubifypath` to see what the browser actually sends. Browser autofill sometimes inserts a different value. Paste what you find, and I'll look at it with you.

The log shows nothing from a search. `google.py` logs a warning on any Google failure, so there's no Google error. Either the request never reaches Calibre-Web, or it reaches it and something hangs without logging. Kepubify is confirmed fine for the app user, so that's closed.

1. Check that the key works from inside the container, with timing:

bash

```
docker exec lib.meson.in python3 -c "
import sqlite3, requests, time
k = sqlite3.connect('/config/app.db').execute('select config_googlebooks_api_key from settings'
print('key length:', len(k))
t = time.time()
r = requests.get('https://www.googleapis.com/books/v1/volumes?q=test&key=' + k, timeout=15)
print(r.status_code, round(time.time() - t, 2), 's')
"
```

You want `200` and a short time.

2. Find which provider host hangs. This tests each provider's site from the container with a hard timeout:

bash

```
for u in https://www.amazon.com https://book.douban.com https://lubimyczytac.pl https://scholar
  docker exec lib.meson.in curl -s -o /dev/null -m 10 -w "$u  %{http_code}  %{time_total}s\n" "
done
```

A `TIMEOUT/FAIL` or a ~10s result marks a provider that stalls. Scholar and Amazon are the usual suspects, because they tarpit or captcha server IPs. Untick that provider in the dialog, since the dialog's checkboxes are saved per admin.

3. See what the browser does. This is the key piece, since the server log is silent:

1. Press F12 and open the **Network** tab. Tick **Preserve log**.
2. Run the search with only Google ticked.
3. Click the `metadata/search` request and tell me its **status** (pending, 200, 500, 504, or "(failed)") and its **time**.
4. Also check the **Console** tab for red errors.

4. Rule out the reverse proxy. Your site sits behind a proxy, and the proxy may be what hangs. Check its error log right after a search (the path depends on your setup, commonly):

bash

```
sudo tail -n 20 /var/log/nginx/error.log
```

If the request in the Network tab stays pending and nginx shows `upstream timed out`, add `proxy_read_timeout 120s;` to the Calibre-Web location block. If it's pending and nginx shows

nothing, the request is stuck in the app. In that case the Debug log level matters: set it under Admin → Basic Configuration → Logfile Configuration. If that save is blocked by the Kepubify error too, tell me, and I'll give you the database command for it.

Send me the results of steps 1 and 2 and the Network tab status from step 3, and I can name the culprit.

Everything on the network side checks out, so Google and the other providers aren't the cause.

- **Google key:** 39 characters, returns `200` in 1.55 s from inside the container.
- **Other providers:** Amazon, Douban, Lubimyczytac and Scholar all respond quickly. ComicVine's 403 on the homepage is normal, since it needs its own API key anyway.

That points at the app or the proxy, not the providers.

1. Read the real application log. `docker logs` only shows the startup scripts. Calibre-Web writes its own log to a file, which is why your searches never appeared there:

bash

```bash
docker exec lib.meson.in tail -n 50 /config/calibre-web.log
```

Run one search first, then run this. You're looking for warnings or tracebacks from the metadata code. If the file is quiet, set the log level to Debug in the database, since the form won't save:

bash

```bash
docker exec lib.meson.in python3 -c "
import sqlite3
c = sqlite3.connect('/config/app.db')
c.execute('update settings set config_log_level=10')
c.commit()
"
docker compose restart
```

Then search once and tail the file again. To go back later, set the level to `20` (Info).

2. Bypass the reverse proxy. If your compose file publishes port 8083 (check with `docker port lib.meson.in`), open `http://SERVER_IP:8083` directly, or over Tailscale if you use it, and try Fetch Metadata there.

- **Works there but not on `lib.meson.in`:** the proxy is the problem. Add this to the Calibre-Web location block in nginx, then reload:

nginx

```
proxy_read_timeout 120s;
proxy_buffering off;
proxy_set_header X-Forwarded-Proto $scheme;
proxy_set_header Host $host;
```

- **Spins in both:** it’s the app or the browser.

3. Check the browser (F12):

- **Network tab → `metadata/search`:** is the status pending, 200, 500 or 504, and is the response empty? A 200 with an empty list can still leave some versions showing “loading”.
- **Console tab:** any red errors? A JavaScript error such as a failed script load can freeze the dialog. Also try a hard refresh (Ctrl+Shift+R) and a private window, because the UI scripts can be cached from before an upgrade.

Send me the tail of `calibre-web.log` after one search and the Network status, and I should be able to name the cause.

Your nginx config is the likely culprit, and it fits everything you’ve seen so far. Google, the other providers and the app log all look healthy, so the search request probably succeeds, and the browser then fails to render the results.

The suspect is this line in your CSP:

```
script-src 'self' 'unsafe-inline';
```

with the comment “No unsafe-eval required”. As far as I remember, Calibre-Web’s Fetch Metadata dialog (`get_meta.js`) builds its result rows with Underscore’s `_.template`, which compiles templates using `new Function()`. Browsers block that unless the CSP allows `'unsafe-eval'`. The script then throws an error, and the “loading” message is never replaced. I’m going from memory here, so confirm it first.

1. Confirm in the browser

Open F12 → **Console**, run a search, and look for a red error like:

```
Refused to evaluate a string as JavaScript because 'unsafe-eval' is not an allowed source of
the following Content Security Policy directive...
```

In the **Network** tab, `metadata/search` should show a plain `200` with JSON results. If both are true, this is the cause.

2. Fix it, only on the edit page

Your CSP applies to the whole site, so allow `unsafe-eval` only where the dialog lives (`/admin/book/<id>`). Add this `map` next to your existing `$loggable` map in the `http` context:

nginx

```
map $uri $csp_script_src {
    ~^/admin/book/  "'self' 'unsafe-inline' 'unsafe-eval'";
    default         "'self' 'unsafe-inline'";
}
```

Then change `script-src` in your CSP line to use the variable:

nginx

```
add_header Content-Security-Policy "default-src 'self'; script-src $csp_script_src; style-src '
```

Update the comment above it as well, since the “no unsafe-eval required” note is no longer accurate. Apply it:

bash

```
sudo nginx -t && sudo systemctl reload nginx
```

Hard-refresh the edit page (Ctrl+Shift+R) and retry Fetch Metadata.

3. Quick test without editing anything

To check the theory first, load the Calibre-Web port directly (`http://SERVER_IP:8083`, or over Tailscale if you use it). That bypasses nginx and its CSP. If the dialog works there, the CSP is confirmed.

Nothing else in the config looks wrong. The 300 s `proxy_read_timeout` is generous, so a proxy timeout isn’t a plausible cause. The `static/` and `cover/` locations define their own `add_header` lines, which drops the CSP and the other headers for those responses, but that doesn’t affect this problem.

If the Console shows a different error, paste it and I'll take it from there.

That's the confirmation. `get_meta.js` calls Underscore's `_.template`, which uses `new Function()`, and your CSP blocks it because `script-src` has no `'unsafe-eval'`. The search itself is succeeding. The browser just refuses to render the results, so the dialog stays on "loading".

1. Add the map in the `http` context, next to your existing `$loggable` map:

nginx

```
# Calibre-Web's Fetch Metadata dialog needs 'unsafe-eval' (Underscore templates)
map $uri $csp_script_src {
    ~^/admin/book/  "'self' 'unsafe-inline' 'unsafe-eval'";
    default         "'self' 'unsafe-inline'";
}
```

2. Use it in the CSP in the HTTPS server block, replacing the existing `Content-Security-Policy` line:

nginx

```
add_header Content-Security-Policy "default-src 'self'; script-src $csp_script_src; style-src '
```

Also change the comment above it, which still says "No unsafe-eval required".

3. Test and reload:

bash

```
sudo nginx -t && sudo systemctl reload nginx
```

Hard-refresh the book edit page (Ctrl+Shift+R) and run Fetch Metadata again. The CSP is delivered with the HTML page, so it's the edit page's URL (`/admin/book/<id>`) that has to match the map.

Notes

- The relaxation applies only to the book edit pages, which only admins or users with edit rights can reach, so the rest of the site keeps the stricter policy.
- If you'd rather not use a map, you can simply add `'unsafe-eval'` to `script-src` site-wide. That works, but it weakens the policy everywhere.
- Your Google Books key is still useful: the 429s were real, and the key fixes them. You don't need the timeout patch to `google.py` any more, so remove that bind mount if you

added it.
- If you want a different level of security, you could also remove the `unsafe-inline` from the policy later, but Calibre-Web depends on inline scripts, so I'd leave it.

If the dialog still hangs after the reload, check the Console for a new error and send it over.

Glad it's working. A few cleanup items, if they apply to you:

- **Log level:** if you set `config_log_level=10` in the database, set it back to `20` (Info) so the log doesn't grow with debug output.
- **`google.py` bind mount:** if you added it, remove it from the compose file and delete the local copy, then run `docker compose up -d`. The key now lives in the settings, so the patch isn't needed.
- **Shell history:** your Google API key is in your bash history from the `docker exec -e GKEY=...` command. Delete it with `history | grep GKEY`, then `history -d <number>`.
- **Backup:** `/config/app.db.bak` still exists inside the container's config volume. It contains the old settings, so delete it when you no longer need it.
- **Kepubify save error:** it only affects saving the Basic Configuration form and isn't blocking anything now. If you ever need to change settings through the UI, send me the request payload and I'll help you track it down.