[Incident] No image results in Production API queries 2023-05-03 - 5 Whys? Postmortem

Background

- Trying to be intentionally blameless, as always with postmortems (see this talk)
- Figuring out why what happened, happened, so we can address root causes
- Start with a "why" question, but then iterate more "why"s on top of that to identify root cause
- May not have a *single* root cause for this issue, but hopefully we can identify a primary root cause
- Might want to focus more on a postmortem of our *response* rather than the root cause

5 Whys? Question process

1. Why was the frontend of Openverse not providing images for seven hours?

- Aggressive caching makes it hard to know if the problem is universal or specific.
- We didn't know if if was only on Francisco's browser or for all users xx
- **We have no insight into results being returned from /v1/images at any given time xxxx**
- The ES alias swap *should* be immediate but we may not be certain that it is
- (To my recollection) we don't have clear guidelines on how to handle an ongoing incident (e.g. runbook, where communication should go, when to escalate an issue, etc.) x
- The people who are more familiar with how ES works were offline (it was night time for them).
- There is no "healthy" old ES instance that we can easily point the API to in case of a failure. x
- Because results caching for "cat" worked for me, and I didn't notice the problem for a long time.
- ES health check wasn't capturing the image index hiccup for some reason.
- Accessing ES in production and checking if everything works is not very well documented. x
- Does the index switch step from the data refresh check that the new index is available? I'd say yes but then why was it returning zero images? X

2. Why don't we have insight into results being returned from /v1/images at any given time?

- We do not have monitoring which checks the media endpoints for *results*, just a healthy status code for the homepage x

- We are missing a “Lucene document count” big number on our existing dashboards for primary aliases, and we could have an alarm for when it is below a threshold depending on the expected index size (e.g. <100m for images, <500k for audio) x
- Elasticsearch as a whole still feels like such a massive unknown for many folks on the team
- The index size returned by ES API was correct (both in GB and in document count), but we were still unable to get any images out of the ES into the API. xxx
- When “testing” the results on the frontend, we should use an unpopular query to prevent it from going to cache.
- The lack of ES expertise has resulted in a lack of monitoring. We don’t know much about how to debug these incidents either. xxx
- We may need better tools as well, like (maybe) Kibana, for monitoring the cluster.
- We don’t have a dashboard for health of the results, like we have for the API/Frontend, and we don’t have Slack alerts

3. Why do we have a lack of ES expertise, which has resulted in a lack of monitoring? We don’t know much about how to debug these incidents either.

- We don’t have a great way to test significant/structural changes to data refresh processes. Perhaps we should always do a “full” (whatever that might mean) data refresh cycle in staging before deploying the production services?
- It has not been a high priority so far, we know it’s important for very specific requirements but other tasks have taken precedence
- What ES learnings we *do* have are not documented anywhere. There’s no good entrypoint for learning more about ES.
- Elasticsearch has largely been “stable” and we haven’t had any large projects (yet!) working on changing significant aspects of it
- None (to my knowledge) of the team has prior experience with Elasticsearch at scale, and we were all thrust into a situation of managing this massive dataset
- If You Don’t Look At The Big Scary Beast In The Room Maybe It Will Go Away 😃
- The ingestion server and data refresh are important, complicated, and scary. It feels like just touching them can break them, which disincentivizes people to learn them well
- We are spread thin on all other layers of the stack.
- (I, Olga, think) This was a pretty unusual error, so we might still have had it even if we had a lot of experience and expertise in ES.
- We haven’t reached out to the Automattic ES community to analyze our ES set up and suggest improvements. We (Sara) did reach out for setting up the filtered index, so we probably have a good start
- ES deployment and the link between ES and API are different from ES expertise in general (how to make the search results relevant)

Solutions (3 votes each)

- Add *visual* monitoring to the frontend, which takes a screenshot of happy paths (should we periodically run the e2e/visual regression tests against production, with real data)?

- Add a monitor for the API which checks that results are returned for each media type (with cache busting)
- Actually write the “Outage Protocol” in a way that everyone can understand it and make the first steps after the incident has been first reported. Xxx Zack Krida
- Test the data refresh in staging to see if we can reproduce the issue we saw in production
- Add a “index document count” big number metric to our ES dashboards
- Create a guide for troubleshooting Elasticsearch in production, with sample queries to try and ways to check health xx
- Set up an “Elasticsearch consult” session with the Automattic ES community to see if there are quick & clear wins/changes we could make for more robusticity & monitoring x
- Audit the setup of the Elasticsearch and its connection to the API together with ES experts in Automattic? - this is the same as the above
- Could something throw an alert when a majority of user searches aren’t returning results?
- Setup an alarm for image/audio results count xxxx
- Create a project to update the Elasticsearch version so we can also learn about the setup and how to migrate it
- Maybe we can keep one old index for a while in case the new ones get corrupted. Is there a way to backup ES?
- Add a check that the new ES index is returning results (not simply exists) to data refresh before it deletes the old index. Would be perfect if we could also check that the API can connect to the new index and get the results xxxxx

- **Action items**
  - **Block image refresh until “**Add a check that the new ES index is returning results (not simply exists) to data refresh before it deletes the old index. Would be perfect if we could also check that the API can connect to the new index and get the results”
  - ✓ Create Milestone Zack Krida
    - Tracking issue: https://github.com/WordPress/openverse/issues/2060
    - Milestone: https://github.com/WordPress/openverse/milestone/10
  - Create an issue requesting an “Elasticsearch troubleshooting” runbook Zack Krida
  - ✓ Allow choosing the preferred index for search results in the API via a query param Dhruv Bhanushali
    - https://github.com/WordPress/openverse/issues/2070
  - ✓ Make the check before promotion of index more comprehensive Dhruv Bhanushali
    - https://github.com/WordPress/openverse/issues/2071
  - ~~Create an Uptime Robot alarm for when images/ or audio/ endpoint are returning 0 results~~ Krystle Salazar

        - Uptime Robot does not have the ability to check field values, only presence (or not) of “keywords”
    - Create an “Outage Protocol” runbook Madison Swain-Bowden
    - Set up an “Elasticsearch consult” session with Automattic Krystle Salazar