Incident documentation/20190628-EventStreams

< Incident documentation

document status: final

Summary

Two unrelated incidents happened to the same service one after the other on the same day. See https://phabricator.wikimedia.org/T226808 for more info.

1. The migration from kafka200x to new kafka-main200x hosts was in progress, but the plan did not include changing the Kafka broker list for dependent services. All EventStreams daemons on scb2* hosts in codfw were unavailable because of a lack of available Kafka brokers, and the only form of alarm was a UNKOWN Icinga state in the related UI.

2. A single IP address started many connections to https://stream.wikimedia.org/v2/stream/recentchange, eventually filling up the pool of allowed concurrent connections in Varnish (25 * 8 varnish backends == 200 total connections). Since the pool was used, other clients were being denied connections with a HTTP 502 from nginx.

A bandaid patch was deployed to block the offending IP's connections to allow legitimate clients to connect over the weekend until we had time to deploy a more robust fix.

Impact

Clients that attempted to connect to EventStreams during the outage windows got at HTTP 502, meanwhile the ones already holding a long lived HTTP connection were unaffected (except when Luca roll restarted EventStreams on the scb* hosts, because that action forced them to reconnect). If they are good SSE clients, then they should be able to resume from where they left off after the issue was fixed.

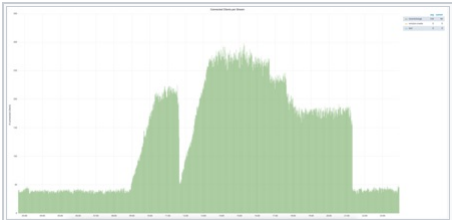
Detection

Luca noticed SERVICE_UNKNOWN Icinga alerts due to the Kafka broker migration. At 10:42 UTC he then saw the 'PROBLEM - Check if active EventStreams endpoint is delivering messages' alarm caused full connection pool. Both notifications were not optimal:

- The UNKNOWN state of the service health checks meant that EventStreams in codfw was not working and it was left in that state for hours.
- The Icinga alarm for the external endpoint's health check is configured for the analytics contact group only. This was ok at the beginning when the service was in its infant state and the Analytics team needed more development cycles to fix bugs and stability issues, but now the service is stable and used by a lot of automated tools in our community, so it would probably need a broader notification audience.

Timeline

- 2019-06-27 14:43 - Keith starts the work to replace kafka2001 with kafka-main2001. The former was the only available broker remaining for EventStreams' daemons on scb2* nodes. **[FIRST OUTAGE BEGINS]**
- 2019-06-28 08:43 - Luca notices by chance that all the EventStreams service health checks for scb2* hosts was in UKNOWN state, and roll restarts all of them. **[FIRST OUTAGE ENDS]**
- 2019-06-28 10:42 - Icinga notifies Analytics that the HTTP service health check for https://stream.wikimedia.org/v2/stream/recentchange is in critical state. **[SECOND OUTAGE BEGINS]**
- 2019-06-29 11:36 - Luca roll restarts all the EventStreams daemons on scb1* hosts **[SECOND OUTAGE ENDS]**
- 2019-06-29 14:16 - Icinga notifies Analytics that the HTTP service health check for https://stream.wikimedia.org/v2/stream/recentchange is in critical state. **[THIRD OUTAGE BEGINS]**
- 2019-06-29 17:44 - Andrew restart EventStreams on scb1001 to test verbose logging (as attempt to gather data about client IPs trying to reconnect). This created more free slots for new connections, and the overall service recovered. **[THIRD OUTAGE ENDS]**
- 2019-06-29 21:16 - Andrew deploys a bandaid patch to blacklist the IPs that were holding too many concurrent HTTP connections to EventStreams. This patch worked cause the party hoarding connections was not a hostile one, rather was taken as many resources as were available.



Backend connections for the EventStream services on scb* hosts.

Cache text nginx's HTTP statuses

Conclusions

The connection pool limits for EventStreams are naive, and we are working on a better way to manage concurrent connections. Ideally we would have connection limits per IP per host. We have also a sub-optimal alarming and documentation for EventStreams that should be fixed as part of the follow ups of this Incident report.

Actionables

- Add per client-IP concurrency limits to EventStreams. Our first try adds IP limits per worker but given that there are as many workers as processors per host we need to be more precise and set per host limits. Ongoing work: (T226808)(https://gerrit.wikimedia.org/r/c/mediawiki/services/eventstreams/+/519713)
- Discuss with the SRE team if the EventStreams external health check should alarm both Analytics and the SRE team (T227065).
- Discuss with the Core Platform team and the SRE team if the UNKNOWN state of a service health check is something that we should alarm on (T227065).

Category: Incident documentation

Main page
Recent changes
Server admin log (Prod)
Server admin log (RelEng)
Deployments
SRE/Operations Help
Incident status

Cloud VPS & Toolforge
Cloud VPS documentation
Toolforge documentation
Request Cloud VPS project
Server admin log (Cloud VPS)

Tools
What links here
Related changes
Special pages
Permanent link
Page information
Cite this page

Print/export
Create a book
Download as PDF
Printable version