Closed (moved) Opened 2 years ago by John Northrup

10.5.0-RC8.ee.0 Deployment Outage

Context

A SQL update in 10.5.0-RC8.ee.0 has caused high database load and made GitLab unresponsive for interactive actions.

Timeline

On date: 2018-02-16

- 20:02 UTC - Deployment of `10.5.0-RC8.ee.0` starts.
- 21:10 UTC - Alert of `Increased Error Rate Across Fleet`
- 21:13 UTC - Alert of `Number of PostgreSQL Databases in prd has changed in the past minute. There are now 1 databases in environment "prd"`
- 21:13 UTC - Internal GitLab users point to failing MRs with 500 level status messages.
- 21:16 UTC - Alert of `Postgres transactions showing high rate of statement timeouts`
- 21:23 UTC - Oncall Production pings DB to look into Postgres alert
- 21:29 UTC - Problem identified as faulty SQL that caused sequential reads on `CI::Build` to ramp out of control.
- 21:32 UTC - Tweeted about 500 errors on MRs.
- 21:33 UTC - Source of bad SQL coming from Unicorn -- i.e. web requests -- identified
- 21:37 UTC - Chef client stopped on web nodes in preparation for corrective action.
- 21:39 UTC - Hot patch proposed to roll back the offending SQL
- 21:41 UTC - Hot patch dry-run started to verify patching will be successful.
- 21:45 UTC - Hot patch applying to web nodes to bring GitLab back online.
- 21:47 UTC - Tweeted about known problem and that resolution was being deployed.
- 21:50 UTC - Hot patch dry-run applying to API nodes to very patching will be successful.
- 21:51 UTC - Hot patch applying to API nodes.
- 21:54 UTC - All nodes have been patched and services restarted, validating stability now.
- 21:55 UTC - Hot patch dry-run applying to git nodes to very patching will be successful.
- 21:58 UTC - Hot patch applying to git nodes for completeness.
- 21:58 UTC - Tweeted that GitLab.com was recovered and online.
- 22:02 UTC - Hot patch dry-run applying to back-end nodes (sidekiq fleet)
- 22:04 UTC - Hot patch applying to back-end nodes (sidekiq fleet)
- 22:08 UTC - All `gitlab-base-{fe,be}` nodes are running `10.5.0-RC8.ee.0-Patch1`

Incident Analysis

- How was the incident detected?
  - Ratio of 5xx class errors increased across the fleet and alerted via Prometheus Monitoring.
  - Internal GitLab users an external users via Twitter started reporting 500 class errors on MRs
  - Database alert for `Number of PostgreSQL Databases in prd has changed in the past minute` fired
- Is there anything that could have been done to improve the time to detection?
  - Frontend alerting worked as designed and quickly.
  - Time to escalate to a DB instead of immediately paging wasted 10 minutes
  - Database alerting did not function well but a precautionary information alert effectively served as a backstop
- How was the root cause discovered?
  - Database load was racing, a deep dive of SQL jobs in flight was performed to identify the faulty query.
- Was this incident triggered by a change?
  - Yes, this was triggered by the `10.5.0-RC8.ee.0` deployment.
- Was there an existing issue that would have either prevented this incident or reduced the impact?
  - No.

Root Cause Analysis

- New SQL deployed to production web controllers overloaded Postgres with full sequential scans
- The queries took longer to complete than the available number of connections (or cpu cores) could sustain so the nearly all the available connections were running the bad sql

- The code was written this way because it was an optimization for a different call site. The developer believed that was the only consumer of this code and the problematic call site was hidden in a Rails has_many relationship between two other classes.

The interesting question is why this wasn't caught in development, staging, or canary.

- It wasn't caught in development because the test database has much smaller tables and aside from overly slow queries we have no systematic way to find problematic queries in the test environment.
- It wasn't caught in staging because it behaved as expected on the known use cases. We do not run rspec tests or any automated tests in staging.
  - Also, staging is quite out of date so there were very few unexpired artifacts in canary (anything older than 30 days is "expired") making plans unrealistically fast
- It wasn't caught in canary because it involved a migration so process skips canary

What went well

- Once the Zoom call was started and joined the problematic query was quickly identified and rolled back.

What can be improved

- The alert for high rate of web errors was swamped by the continuing alerts for a high rate of https_git errors.
  - We should silence alerts like this that are firing for a long time.
- The Database alerts that fired were initially confusing. They alerted that the number of Postgres databases had changed because the overloaded databases simply stopped reporting metrics in Prometheus.
  - We can investigate how to make the postgres_exporter more reliable on an overloaded database. Using a persistent database connection or some way to reserve database connections for it might be necessary.
  - We should look into why the alerts for missing prometheus scrapes didn't fire.
  - We should consider paging on "number of databases changed". We can build silences into the manual failover process and HA failovers are uncommon enough not to create too much noise.
- The oncall DB wasn't paged by the alerts and the escalation process wasn't very clear. It took 10 minutes from the first database alert to contacting the DB and it was only luck that they were online at the time.
- It took a few minutes before the impact was apparent and a Zoom call started and joined by developers.
- Better tooling to catch bad SQL earlier in the development cycle more reliably.
- Better tooling to reliably find all changed SQL when models are changed.
- Refreshing Staging regularly
  - Ideally staging should be refreshed before the just first RC is deployed so that the deploy process can test on realistic data
- Some automated tests on staging to check for unexpected side effects on web pages
  - Implementing rspec tests to run on staging would be challenging
  - We could possibly run the profiler against staging after a deploy but actually evaluating the results could be hard
  - We could do something simple like direct pingdom checks at staging so we detect when a page stops responding or takes much longer. But having a large enough set of pages might be expensive
- Allowing testing on Canary

Corrective actions

See "Related issues"

Guidelines

- Blameless Postmortems Guideline
- 5 whys

Edited 2 years ago by Gregory Stark

Linked issues 10

Relates to

- Detect bad SQL in CI tests
  gitlab-org/gitlab-foss#43457
- Consider paging DB specialists and/or production on database alerts
  #3741
- Change release process to make better use of Canary
  #3704
- Refresh staging DB regularly

#3752

Automated baseline testing for staging
#3764

Investigate alerting in case where postgres_exporter cannot make database conjections
#3765

Add instructions to runbooks to page DB specialist earlier
#3766

Reduce alert noise
#3784

Rewrite error rate alerts to be less prone to false positives
#3785

Detect bad SQL in CI tests
gitlab-org/gitlab#21237

Ian Baum @ibaum mentioned in issue gitlab-org/release/tasks#96 (closed) 2 years ago

John Northrup @northrup changed the description 2 years ago

John Northrup @northrup changed title from **10.5.0-RC8.ee.0 Deployment** to **10.5.0-RC8.ee.0 Deployment Outage** 2 years ago

John Northrup @northrup changed the description 2 years ago

John Northrup @northrup changed the description 2 years ago

Andrew Newdigate @andrewn assigned to @_stark 2 years ago

Andrew Newdigate @andrewn · 2 years ago Maintainer

@_stark will look to do a root cause analysis on this

Gregory Stark @0stark changed the description 2 years ago

Gregory Stark @0stark changed the description 2 years ago

Gregory Stark @0stark changed the description 2 years ago

Gregory Stark @0stark changed the description 2 years ago

Gregory Stark @0stark mentioned in issue #3741 (moved) 2 years ago

Gregory Stark @0stark changed the description 2 years ago

Gregory Stark @0stark changed the description 2 years ago

Yorick Peterse @yorickpeterse mentioned in issue #3743 (closed) 2 years ago

Yorick Peterse @yorickpeterse · 2 years ago Maintainer

With the corrective actions being discussed can we close this issue?

Gregory Stark @0stark · 2 years ago

There are 9 points in "What can be improved" and only 3 (or 4 including the merged one) issues opened. So there's still a few more issues to open before this can be closed.

Gregory Stark @0stark marked this issue as related to #3741 (moved) 2 years ago

Gregory Stark @0stark marked this issue as related to #3704 2 years ago

Gregory Stark @0stark marked this issue as related to #3752 (closed) 2 years ago

Gregory Stark @0stark marked this issue as related to #3764 2 years ago

Gregory Stark @0stark marked this issue as related to #3765 (moved) 2 years ago

Gregory Stark @0stark marked this issue as related to #3766 2 years ago

John Northrup @northrup · 2 years ago

🤔 @_stark I tend to agree with @yorickpeterse, so long as the corrective actions all have issues assigned to them I see no reason to keep this outage issue open.

Gregory Stark @0stark · 2 years ago

The only one remaining now is the one about silencing the alert noise for increased error rates. I'm not sure what to put in an issue for that. Perhaps to update the handbook with instructions to create a silence whenever an alert will take longer than one oncall shift to fix?

Yorick Peterse @yorickpeterse · 2 years ago Maintainer

@_stark I think we can adjust any existing alerts if necessary, but for the others we may want to treat them on a per case basis?

Gregory Stark @0stark mentioned in issue #3784 2 years ago

Gregory Stark @0stark marked this issue as related to #3784 2 years ago

Gregory Stark @0stark marked this issue as related to #3785 2 years ago

Gregory Stark @0stark · 2 years ago

Ok, I went ahead and made two new issues for alerts. One for silencing and one for changing `irate` to `rate` to reduce the problem with false positives.

I think that's everything now.

Gregory Stark @0stark changed the description 2 years ago

Gregory Stark @0stark closed 2 years ago

Andrew Newdigate @andrewn moved to production#294 (closed) 1 year ago

Andreas Brandl 🔴 @abrandl mentioned in issue #4962 (closed) 1 year ago

Please register or sign in to reply