

Closed (moved)

Opened 2 years ago by  [John Jarvis](#)

Gitlab system wide outage on 2017-09-11

Context

All Gitlab.com services were unavailable for approximately 40 minutes on September 11th due to lost connectivity to the primary database.

Timeline

On date: 2017-09-11

- 2017-09-11 10:14:27 UTC - Gitlab.com services stopped responding
- 2017-09-11 10:56:27 UTC - All site functionality was restored.

Incident Analysis

- How was the incident detected? - Multiple alerts and user reports.
- Is there anything that could have been done to improve the time to detection? - No, we were notified immediately.
- How was the root cause discovered? - On the production call, we discovered that servers were seeing timeouts when trying to connect to the db load balancer.
- Was this incident triggered by a change? - No.
- Was there an existing issue that would have either prevented this incident or reduced the impact? - No

Root Cause Analysis

- Why was were gitlab services unavailable for 30 minutes? - We lost connectivity to the primary database.
 - Why did we lose connectivity to the database? - The LB that fronts the primary database was not accepting connections.
 - Why did it stop accepting connections? - TBD, submitted ticket to Azure.
- Why did everything go unavailable with a 500 error for this type of failure? - We currently do not have a way to handle database failures gracefully? (TBD)
- Why was status.gitlab.com unavailable? (TBD)

What went well

- Keeping our customers informed of what was going on with our status twitter account.
- Working to quickly work around the issue on the call.



What can be improved

- status.gitlab.com should have remained available for our customers.
- A slightly more graceful way of handing this, other than giving the users 500s and 502s.


Corrective actions


- <https://gitlab.com/gitlab-com/infrastructure/issues/2745>

Edited 2 years ago by [John Jarvis](#)

Linked issues   1

Relates to

 [Routing outage 2017-09-13](#)
#2766

 WoW ending ...



[John Jarvis](#) @jarv mentioned in issue [#2745 \(closed\)](#) 2 years ago



John Jarvis @jarv changed the description 2 years ago



Yorick Peterse @yorickpeterse · 2 years ago

Maintainer

Why did everything go unavailable with a 500 error for this type of failure? - We currently do not have a way to handle database failures gracefully? (TBD)

We sort of do. If primary is unavailable (= throwing errors somehow) we retry operations a few times before giving up. It's possible that if the LB never returned an error (and instead just hung) we'd eventually run into Unicorn timeouts.

Most likely we still contact the primary in requests, even though with recent load balancing changes in EE the number of cases where this happens should be very low.

This brings the question: if the primary is down and we need it, what do we do? There's not much else you can do at that point other than showing an error page.

Edited by **Yorick Peterse** 2 years ago



Jason Tevnan @jtevnan · 2 years ago

Azure issue has been opened: issue nr 11709116313610



George Crawford @georgecrawford · 2 years ago

I'm sorry to hijack this thread if it's not appropriate, but we're seeing very similar issues on a hosted GitHub instance. Could the infrastructure for GitHub be suffering from the same issue?



Jason Tevnan @jtevnan · 2 years ago

@georgecrawford this is an internal loadbalancer for gitlab.com; so there is no connection I can see.



Ilya Frolov @ilyaf · 2 years ago

During the outage there was no connectivity on network level from -fe and -be roles (all our fleet pretty much) to LB IP address. Before pointing to database address directly we've seen that all the new connections was in SYN_SENT state, and all the established connections were just re-transmitting whatever they were sending, w/o any packets back and w/o any of the port/net uncheacheable icmp packets that could have helped our servers to react. So LB never returned any error in that sense -- I'm not seeing any other way to detect that other than timeouts.



John Jarvis @jarv · 2 years ago

Owner

This brings the question: if the primary is down and we need it, what do we do? There's not much else you can do at that point other than showing an error page.

You are right there probably is not much we can do except maybe give the user something that points to our status page which would be (slightly) better than our current 500 page.

This got me thinking though that if the primary is down but the secondaries are up, is it possible for us to be more resilient to failure?



John Jarvis @jarv changed the description 2 years ago



Ghost User @ghost1 · 2 years ago

@jarv possible, flip to read only on the followers. Something for <https://gitlab.com/gitlab-com/infrastructure/issues/2647>



Ghost User @ghost1 mentioned in issue [#2743 \(moved\)](#). 2 years ago



Gabriel Mazetto @brodock · 2 years ago

Developer

This: <https://gitlab.com/gitlab-org/gitlab-ce/issues/37534> could be useful when you have a primary database outage and you want to use a follower to keep a read-only version online (and make it work like a temporary Geo secondary node)



Ilya Frolov @ilyaf · 2 years ago

Last time we had probes from that Azure probe IP (168.63.129.16) in pgbouncer logs is few minutes before the outage:

```
ilya [10:07]
```2017-09-11 10:12:26.727 35387 LOG C-0x7f0f67b9c3a8: (nodb)/(nouser)@168.63.129.16
2017-09-11 10:12:31.728 35387 LOG C-0x7f0f67b9c3a8: (nodb)/(nouser)@168.63.129.16
2017-09-11 10:12:36.728 35387 LOG C-0x7f0f67b9c3a8: (nodb)/(nouser)@168.63.129.16
2017-09-11 10:12:41.729 35387 LOG C-0x7f0f67b9c3a8: (nodb)/(nouser)@168.63.129.16
2017-09-11 10:12:46.729 35387 LOG C-0x7f0f67b9c3a8: (nodb)/(nouser)@168.63.129.16
2017-09-11 10:12:51.729 35387 LOG C-0x7f0f67b9c3a8: (nodb)/(nouser)@168.63.129.16
2017-09-11 10:12:56.730 35387 LOG C-0x7f0f67b9c3a8: (nodb)/(nouser)@168.63.129.16
2017-09-11 10:13:01.731 35387 LOG C-0x7f0f67b9c3a8: (nodb)/(nouser)@168.63.129.16
2017-09-11 10:13:06.731 35387 LOG C-0x7f0f67b9c3a8: (nodb)/(nouser)@168.63.129.16
2017-09-11 10:13:11.732 35387 LOG C-0x7f0f67b9c3a8: (nodb)/(nouser)@168.63.129.16
2017-09-11 10:13:16.732 35387 LOG C-0x7f0f67b9c3a8: (nodb)/(nouser)@168.63.129.16
2017-09-11 10:13:21.733 35387 LOG C-0x7f0f67b9c3a8: (nodb)/(nouser)@168.63.129.16
2017-09-11 10:13:26.733 35387 LOG C-0x7f0f67b9c3a8: (nodb)/(nouser)@168.63.129.16
2017-09-11 10:13:31.734 35387 LOG C-0x7f0f67b9c3a8: (nodb)/(nouser)@168.63.129.16
2017-09-11 10:13:36.734 35387 LOG C-0x7f0f67b9c3a8: (nodb)/(nouser)@168.63.129.16
2017-09-11 10:13:41.735 35387 LOG C-0x7f0f67b9c3a8: (nodb)/(nouser)@168.63.129.16
2017-09-11 10:13:46.735 35387 LOG C-0x7f0f67b9c3a8: (nodb)/(nouser)@168.63.129.16
2017-09-11 10:13:51.736 35387 LOG C-0x7f0f67b9c3a8: (nodb)/(nouser)@168.63.129.16
```

Now we do receive SYNs with those weird E and W flags set, but we don't respond to them:

```
tcpdump -i eth0 -vvn ip and host 168.63.129.16 and port not 80 and port not 53
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
10:13:25.046282 IP (tos 0x2,ECT(0), ttl 128, id 226, offset 0, flags [DF], protocol TCP)
 168.63.129.16.54463 > 10.46.2.8.6432: Flags [SEW], cksum 0x93c7 (correct), seq 168.63.129.16.54463
10:13:27.045750 IP (tos 0x2,ECT(0), ttl 128, id 254, offset 0, flags [DF], protocol TCP)
 168.63.129.16.54499 > 10.46.2.8.6432: Flags [SEW], cksum 0x98e6 (correct), seq 168.63.129.16.54499
10:13:30.053751 IP (tos 0x2,ECT(0), ttl 128, id 294, offset 0, flags [DF], protocol TCP)
 168.63.129.16.54499 > 10.46.2.8.6432: Flags [SEW], cksum 0x98e6 (correct), seq 168.63.129.16.54499
^C
```

But, we don't block neither this IP nor 6432 port too.



**Alex Hanselka** @ahanselka mentioned in issue [#2603 \(closed\)](#) 2 years ago



**John Jarvis** @jarv added [moved 1](#) label 2 years ago



**Daniele Valeriani** [GitLab] @omame-gitlab · 2 years ago

I'm creating a new load balancer in [https://gitlab.com/gitlab-com/gitlab-com-infrastructure/merge\\_requests/143](https://gitlab.com/gitlab-com/gitlab-com-infrastructure/merge_requests/143) so we can restore the previous state simply by updating the db\_host in Chef.



**Daniele Valeriani** [GitLab] @omame-gitlab · 2 years ago

As per [conversation in Slack](#) I deleted the new lb and updated the db\_host setting in Chef to point to db3 .



**Daniele Valeriani** [GitLab] @omame-gitlab assigned to [@omame](#) 2 years ago



**Daniele Valeriani** [GitLab] @omame-gitlab · 2 years ago

I have a feeling that Microsoft got bit by [their own bug](#) in the WALinuxAgent. This outage behaved exactly the same way as <https://gitlab.com/gitlab-com/infrastructure/issues/2766>: the load balancer suddenly lost routes to the rest of the network and the only way to recover is to reboot the instance, which you cannot do on an Azure load balancer unless you rebuild it.

I think we can safely close this issue since 1) we don't use that load balancer anymore 2) the root cause is very likely to have been found.



**Daniele Valeriani [GitLab]** [@omame-gitlab](#) marked this issue as related to [#2766 \(moved\)](#).  
[2 years ago](#)



**Daniele Valeriani [GitLab]** [@omame-gitlab](#) closed [2 years ago](#)



**Pablo Carranza [GitLab]** [@pcarranza-gitlab](#) mentioned in issue [#2694 \(closed\)](#), [2 years ago](#)



**Andrew Newdigate** [@andrewn](#) moved to [production#240 \(closed\)](#), [1 year ago](#)

Please [register](#) or [sign in](#) to reply