Canned RCAs

Last updated by | Marlon Jin | Nov 16, 2022 at 6:41 AM PST

We noticed recently multiple **Sev 3 ICMs** opened asking for RCA related to availability issues , but unfortunately these ICMs are opened without taking into considerations that you can handle it , and should be able to find the root cause , these ICMs were talking about the below categories :

1. Server restarted due to maintenance deployment.
2. DB node where the server is hosted was disabled/restarted
3. Connection failed because of GW restarts.
4. Infra Deployment.
5. Customer Issue a restart / compute scale up or down.

We are expecting for any connectivity issues the customer is complaining that you **must** check the above points before filing an ICM , so please make sure to check these points and you can find all the necessary information related in our ASC / TSGs , if you can’t collect these information please reach out to your TA/SME/Orcas team channel or you can reach out to the EEE team and we will help you.

Starting from that point I would like to share with you some canned RCAs for such issues , and change based on your service if necessary :

1. **Planned Maintenance deployment** :

Root cause:

After investigation of your server instance listed above, we learnt that the unavailability was caused by an Azure Platform Maintenance deployment on the node where the server was running. This caused the server instance to fail over to other node in our infrastructure by design. Currently 50 percentile of all servers WW failover within 30 seconds, we are working aggressively to optimize the failover time further for all our customers. We suggest that our customers implement some sort of retry logic in their application for database connectivity and consider connection pooling, thereby offering resiliency against such planned infrastructure failover.

We follow a deployment schedule of about once a month, where we refresh database binaries and take care of other azure and security updates. Batched updates are planned and customers are able to receive notifications in advance. Today, Platform Maintenance, are not able to be batched with database binary and security updates.

Mitigations & solution:

- In most of the cases, errors like the above are transient in nature and your database server will come back online automatically. Please try to reconnect to your server to see if the server is back online.
- You can also look at the Resource Health Check to see the current state of your server

Going forward if you want to be notified before we run deployments , you can create a notification that will notify you before this deployment , for more details , you can check our documentation :

https://docs.microsoft.com/en-us/azure/mysql/concepts-planned-maintenance-notification //MySQL

https://docs.microsoft.com/en-us/azure/mariadb/concepts-planned-maintenance-notification //MariaDB

https://docs.microsoft.com/en-us/azure/Postgresql/concepts-planned-maintenance-notification // PostgreSQL

2. **DB node restarted**:

Root cause:

The Azure database for MySQL/MariaDB/PostgreSQL is running as a service within Azure ecosystem , each consisting of multiple VMs and is continuously being monitored by our Azure Infrastructure and whenever unhealthy are automatically taken out of usage for repair. When this occurs databases hosted on that VM are failed-over to their secondaries to continue processing requests. the downtime is from xxxx-xx-xx xx:xx:xx to xxxx-xx-xx xx:xx:xx. The node can be down due to various reasons viz hardware failure, OS crashes or bugs, kernel driver issues, transient race conditions etc,, and as a PaaS service, it is more important for us to handle the node down promptly. In this case downtime is <x seconds or minutes> and failover completed in <x seconds or minutes> , we would suggest you to add the retry logic to reduce the impaction to the application or connection resiliency to handle such transient failure when the server fails over.

Mitigation
No manual mitigation performed. The hardware maintenance is outside the realms of Azure PostgreSQL/MySQL/MariaDB Team and done at the core Azure platform level. Our built-in HA fails over the application to another node when the existing node is under maintenance.

3. **GW restart :**

Underlying gateway node(s) was restarted because of a repair job. There is no ongoing issue with the server. We have multiple gateway nodes that serves the connections for all the servers in a region. Whenever service fabric detects an issue with a particular node it takes it out of the pool to run a repair job/ platform maintenance and get the node healthy again. During this time as there are other nodes that still continue to serve the connections, there would be a minor transient blip in the connectivity for some servers and existing connections that are served from this node gets reset, this is because our connections are always proxied via gateway. So if there are long running jobs from connections served from this node, that might get affected. From a new connection standpoint it should still go through.

Mitigation :

No manual mitigation performed , node defect triggers Azure platform maintenance à DB minor transient blip on existing connections; none on new connections , we recommend to implement resilient application code (retries) to cope with this kind of situation.

4. **Infra Deployment** :

The Azure database for MySQL/PostgreSQL/MariaDB is running as a service within Azure ecosystem , each consisting of multiple VMs and is continuously being monitored by our Azure Infrastructure and whenever unhealthy are automatically taken out of usage for repair. When this occurs, databases hosted on that VM are failed-over to their secondaries to continue processing requests. the downtime is from xxxx-xx-xx xx:xx:xx to xxxx-xx-xx xx:xx:xx. The node can be down due to various reasons viz hardware failure, OS crashes or bugs, kernel driver issues, maintenance, etc.,, and as a PaaS service, it is more important for us to handle the node down promptly. In this case downtime is xx seconds and failover completed in xx seconds. The node where your server was hosted was down for maintenance where we apply hardware/OS/kernel/driver/security updates or fixes. We would suggest you add the retry logic to reduce the impaction to the application or alternatively leverage connection pooling like ProxySQL(for MySQL/MariaDB) or PgBouncer(PostgreSQL) which has built-in retry logic and connection resiliency to handle such failure when the server fails over.

5. **You can see it in the ASC operation tab , and for sure no RCA here.**

Notes:

- ASC should be able to detect these failures for you , so you should be able to troubleshoot these type of cases and share the RCA that is **surfaced by ASC in the case insights , or you can use the above RCAs** .
- All the above can be checked through ASC tabs,
- If the restart took longer time than expected , I suggest to check the customer resources before the restart and check the long restart TSGs we have , for MySQL and PostgreSQL .
- If nothing can be found , or you found unexpected results , at that point you can consult your TA/SME or ask in the Orcas team channel , and if nothing can help here , at that point you can ***file an ICM.***
- Maintain the CRI quality while filing these ICMs , showing all the troubleshooting steps you already did.

So broadly speaking , you are allowed to file an ICM under this topic if you faced the below :

1. Availability issues that are not falling under the above categories , for example a crash dump.
2. Availability issues that are falling under the above categories , but the availability was for an unexpected time that impacted the customer SLA, and you did not find the reason in the long restart TSG mentioned above , on this scenario you can file an ICM after checking with your TA/SME and ask in the Orcas team channel.