Site outages for Adobe Commerce stores: General ways of identifying and measuring their effects

By Adobe Commerce Support

Table of contents:

Introduction

There are multiple ways where a merchant (or anyone interested in the site's health) can identify site-downs, also known as site outages. A site-down is the time period when the merchant's site cannot adequately respond to the customer's requests. The definition of outages and their effects on sites is broad and the authors of this article will not attempt to make a thorough analysis on all possible outage cases or their cause. This article attempts to describe a) the general steps one needs to make to pinpoint an issue that may be an outage and b) identify whether the issue is truly an outage, a small-scale outage or just a temporary impediment. The concepts to identify outages are very general and can be used with any Application & Performance tools. In this article, the described concepts will be applied with the use of the New Relic Application & Performance UI Suit.

The described approaches work for Adobe Commerce 2.3.0-2.3.7-p1, 2.4.0-2.4.3, both on-premises and on our cloud infrastructure.

First steps and preparations

Navigate to New Relic and to the site that requires analysis

The site under investigation needs to have New Relic APM fully installed and functional. Additionally, if New Relic Infrastructure is also installed, it would be beneficial for further analysis. If those two New Relic features are not installed or malfunctioning, analysis related to the site's performance cannot be properly done. To navigate into New Relic Query Builder, please follow the steps from Supplemental Information.

Note: Adobe Commerce on our cloud infrastructure projects include access to certain New Relic services. For details refer to the New Relic services article in our developer documentation.

Verify that CDN is sampled into New Relic Logs

Once the user has successfully navigated into the environment, they need to test whether New Relic samples the sites Content Delivery Network (CDN) appropriately. Depending on the CDN, there are various ways of checking and verifying that the CDN logs are properly defined and integrated into New Relic.

For this article, we will use the CDN maintained by Fastly and integrated into New Relic inside the "Log" Data table. To check Fastly's integration into New Relic Log, the user may use the following query:

```
SELECT * from Log where fb.input IS NULL SINCE 5 hours ago
```

Query 1

The result from Query 1 will look similar to the table below (Figure 1). If the results contain entries that have `"cache_status"`, `"tls_*"` and `"waf_*"` where '*' has various fields and variables, then Fastly logs are properly sampled in New Relic Log. In the above query, the user may use any timeframe of choice, for example "2 days ago", "100 minutes ago", or even specify the exact timeframe by using the syntax (for example 12:30 – 14:00 March 19 UTC):

```
SINCE '2021-03-19 12:30:00 UTC' UNTIL '2021-03-19 14:00:00 UTC'
```

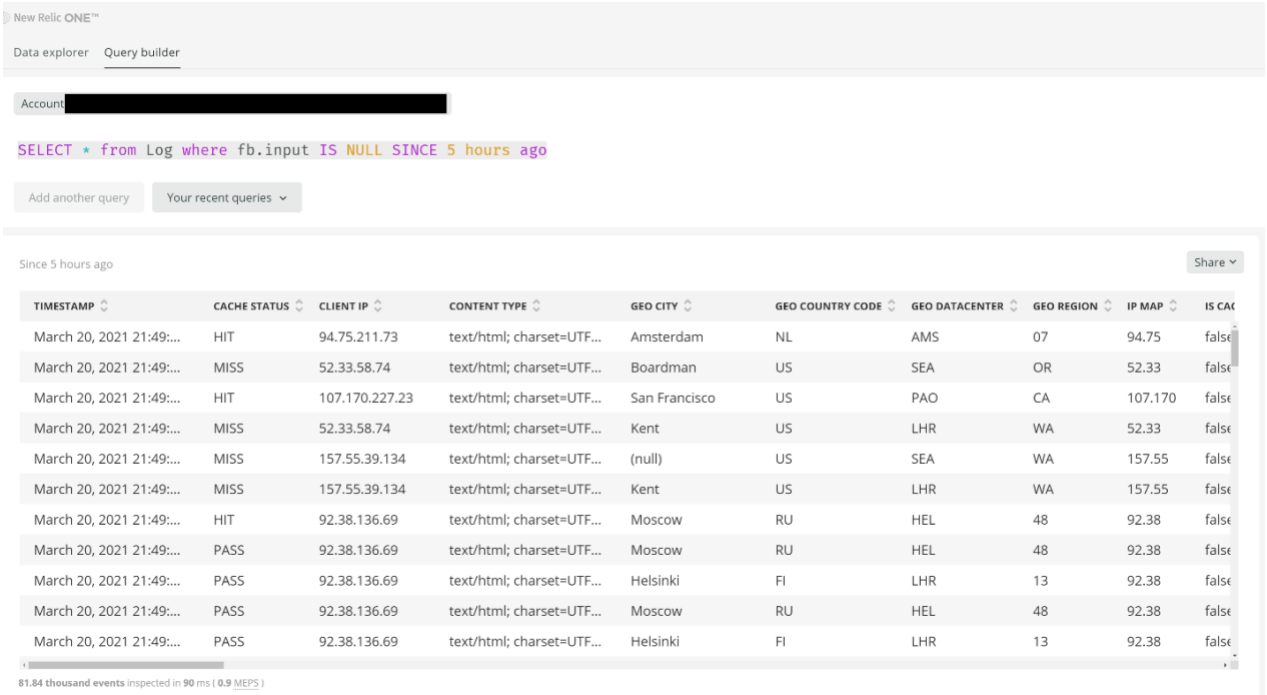


Figure 1: Example of a "Log" table.

In the table in Figure 1, the variable names are the column titles. The user can scroll to see all variable names indicated as the bold column's names of Figure 1. These same variables can be queried in the New Relic Query builder environment. Note that the variable names when queried are slightly differently defined, when compared to the names shown in Figure 1. For example, the "CLIENT IP" variable is queried as "client_ip".

Metrics, variables, and definitions used for the study of outages

For the outages study, the following metrics will be used:

1. Cache Status: Indicated in New Relic as "cache_status". The outcome of a request from the CDN's perspective. If cache status is a "HIT", then the request's content was cached (sometime in the past) and found in the CDN's servers. Consequently, this request is served by the CDN. If cache status is a "MISS", then the CDN sends the request to the site's server and consequently a server's transaction will occur
2. CDN's elapsed time: Indicated in New Relic as "elapsed_time". The duration time in microseconds the CDN requires to serve a request, regardless of its cache status.
3. Based upon the elapsed time metric, the CDN cache latency is defined as the average time of the CDN Misses on a given time period. Cache latency is frequently referred as simply latency.
4. Server transaction status: The outcome of the transaction which takes place in the server. The outcome is reported as a HTTP response code per transaction. For example, "200" for a transaction complete without errors, "503" for a transaction where the Service was not available, etc.
5. Fastly Transaction status: The outcome of a Fastly transaction, recorded as a HTTP response code.

Major Outage cases

Introduction

In this section, we discuss two types of outages which severely affect a site's server. In these two types, the server is affected in such an extent, that its unable to respond to even the most basic set or requests. Key features for each of the outages and examples are presented. Finally, detailed queries and their results which define and measure the magnitude of outages are presented. These queries are used in New Relic Query Builder, but their logic can be applied to any Application Performance Suit.

Type 1 Outage

As a "Type 1" Outage, we refer to the outage where the CDN is unable to communicate, or have major impediments to communicate with the site's server. When this outages occur, the CDN's cache latency significantly exceeds healthy values. A healthy cache latency is considered to be any duration time lower than 3 seconds. To plot a site's cache latency, the user may use the following query:

```
SELECT average(numeric(time_elapsed)) from Log where fb.input IS NULL
and cache_status = 'MISS' Timeseries 1 minutes SINCE '2021-03-19 12:30:00
UTC' UNTIL '2021-03-19 14:00:00 UTC'
```

Query 2

With Query 2 the user requests from NR to retrieve the average of the "time_elapsed", after it converts it into a numeric value (since it is originally a string). The query targets CDN requests which have "cache_status" as misses and plots them with a '1 minute' increment in the timeframe of 2021/3/19 12:30 – 14:00.

The result for a site, which faces a type 1 outage, may look like the following chart:

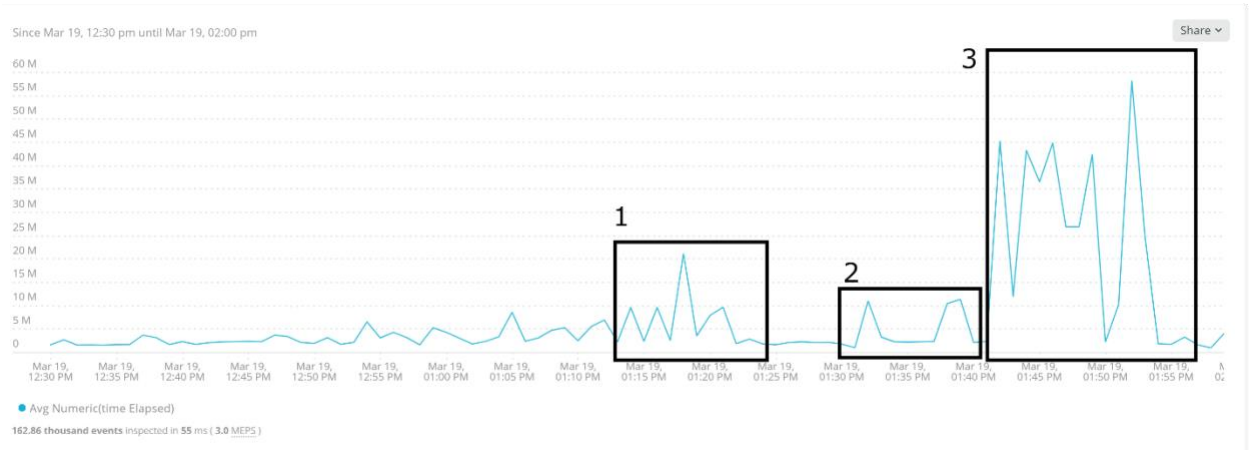


Figure 2: Chart of the example site's cache latency. The chart's total timeframe (lower and upper timestamps) is shown on the upper left corner. The timeframes where issues occur are marked in boxes with a designated number.

On the chart in Figure 2, we observe that from the time period of 2021/3/19 1:10 pm to 1:25 pm -box number 1- the site faces latency that reaches 10 seconds (10 M in the chart means 10 M microseconds, which is 10 seconds) and 20 seconds. Then on the second time period – box number 2 - of 1:30 to 1:40 pm the site tries to recover with only two peaks of 10 seconds each. Finally, on the third time period –

box number 3 – of 1:40 to 1:55 pm, the site clearly faces a 'type 0' outage, since latency reaches the very high values of 45 and nearly 60 seconds.

The user may argue that an outage is indicated in all three individual boxes described in the Figure 1 chart. This argument has a strong point. However, in this article we will not go any further and analyze whether cases described in boxes 1 and 2 are truly outages or not. It is certain beyond doubt that these cases are issues, but the severity of them requires further information and data to be analyzed. An empirical way to differentiate an issue - or even a precursor to issues - from an actual outage is the magnitude of cache latency. As indicated earlier, healthy latency is withing the threshold of 1-3 seconds. Latency values higher than 3 seconds is considered to be an indication that the site's server is making a significant effort to serve the request, thus indicating an issue. Latency higher than 10 seconds is considered to occur when the server cannot respond to the requests been given, and is considered as "site down". Therefore, well established empirical cutoffs and values for cache latency are described and summarized in the below table:

Cases	Threshold – cutoff for cache latency
Healthy site's performance	$t \leq 3\ sec$
Possible issue	$3\ \text{sec} < t \leq 10\ sec$
Outage	$t > 10\ sec$

Table 1: Empirical thresholds of cache latency for outages.

The user may couple the above findings, by plotting the CDN's HTTP error responses. This can be done with the below query:

```
SELECT count(*) from Log where fb.input IS NULL and status like '5%'
Timeseries 1 minutes SINCE '2021-03-19 12:30:00 UTC' UNTIL '2021-03-19
14:00:00 UTC'
```

Query 3

Notice that in the above query (Query 3), the user keeps the same timeframe of 2021/3/19 12:30 – 14:00. The errors targeted by the query are all the 5XX errors. The chart generated by this query is the below:

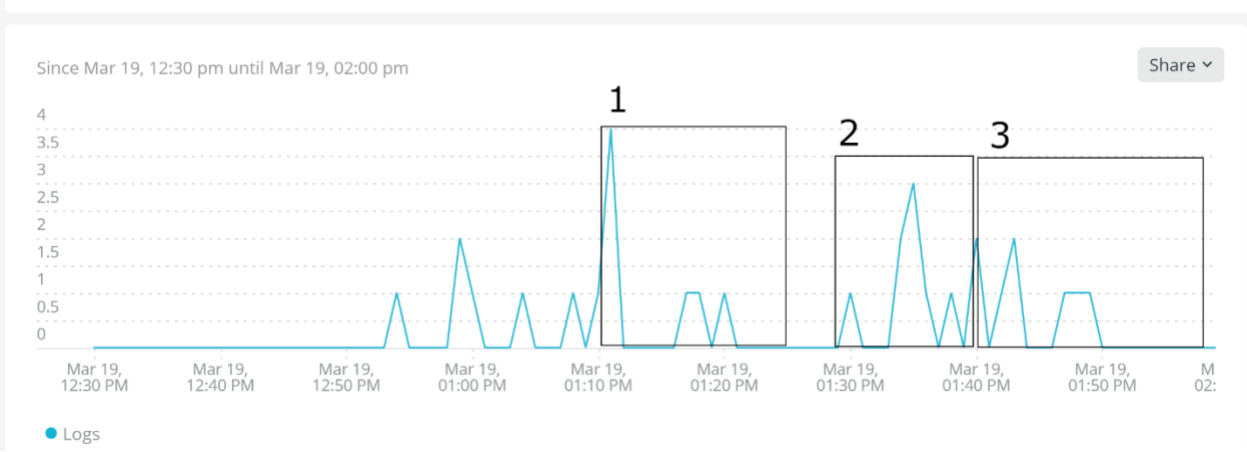


Figure 3: Chart of the example site's 5XX errors. The chart has the same timeframe of Figure 1.

From the above chart (Figure 3), we observe that a number of 5XX errors occur throughout all timeframes where either an issue or an outage occurs. Though the existence of these errors is not indicative of an outage, the occurrence of these in conjunction with the high latency for the same time periods bolster the verdict that these timeframes contain issues and at least one major outage. The magnitude of the 5XX errors is irrelevant to the characterization of an issue as an outage. Simply the occurrence, in a consistent repeated way, is sufficient for identification. The site's example described in Figure 2 has 5XX errors which occur steadily though a time period indicated by the boxes. The magnitude of the errors does not pass 4 errors per minute. But simply because the errors occur consistently every minute during the timeframes indicated in the boxes, bolster the identification of a serious issue in boxes 1 and 2, and an outage in box 3.

Type 2 Outage

The "Type 2" Outage is very similar in behavor when compared to the "Type 1". Again, the CDN is unable to communicate, or has major impediments to communicate with the site's server. However in this case the cache latency has values which are close to zero, or actual zero. Again, the exact same query to caclulate cache latency used for the Type 1 outage (Query 2) can be used. A chart for a site which we can use as an example of a Type 2 outage may look like the one depicted below:

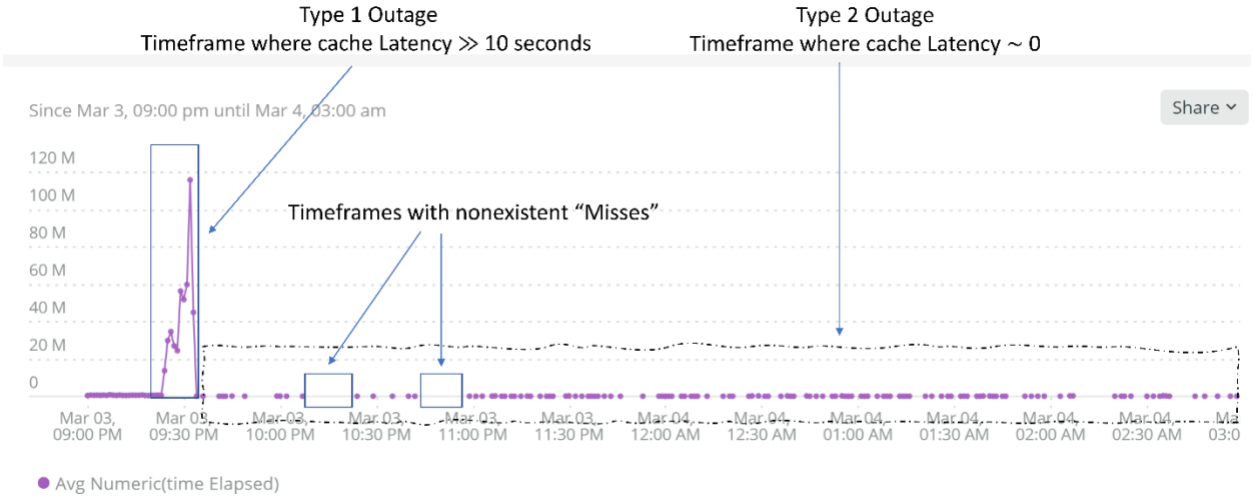


Figure 4: Chart of a site's cache latency which undergoes Type 2 outage. In this specific example, the Type 1 outage is preceding the Type 2.

From the Type 2 case study chart (Figure 4) above we observe the following:

- A major issue, in this case study a Type 1 outage, occurred prior to the Type 2 outage. There is no actual connectivity between those two outages. That means that the occurrence of one does not necessarily "prepare the road" for the other. Still, the occurrence of a major event signifies the start of the outage.
- There are individual timeframes in the Type 2 outage where request classified as "MISS" do not exist at all. In this case, the cache latency measured has the actual value of zero.
- The remaining timeframes in the Type 2 outage include "MISS" requests with extremely low (we emphasize the "extremely") cache latency.

The Type 2 Outage is most frequently accompanied by a vast number of HTTP 503 Errors at the CDN transaction level. To study the 503 errors the following query can be used:

```
SELECT count(*) FROM Log where fb.input IS NULL and status LIKE '503'
Timeseries 1 minutes SINCE '2021-03-03 21:00:00 UTC' UNTIL '2021-03-04
03:00:00 UTC'
```

Query 4

The following chart is obtained:

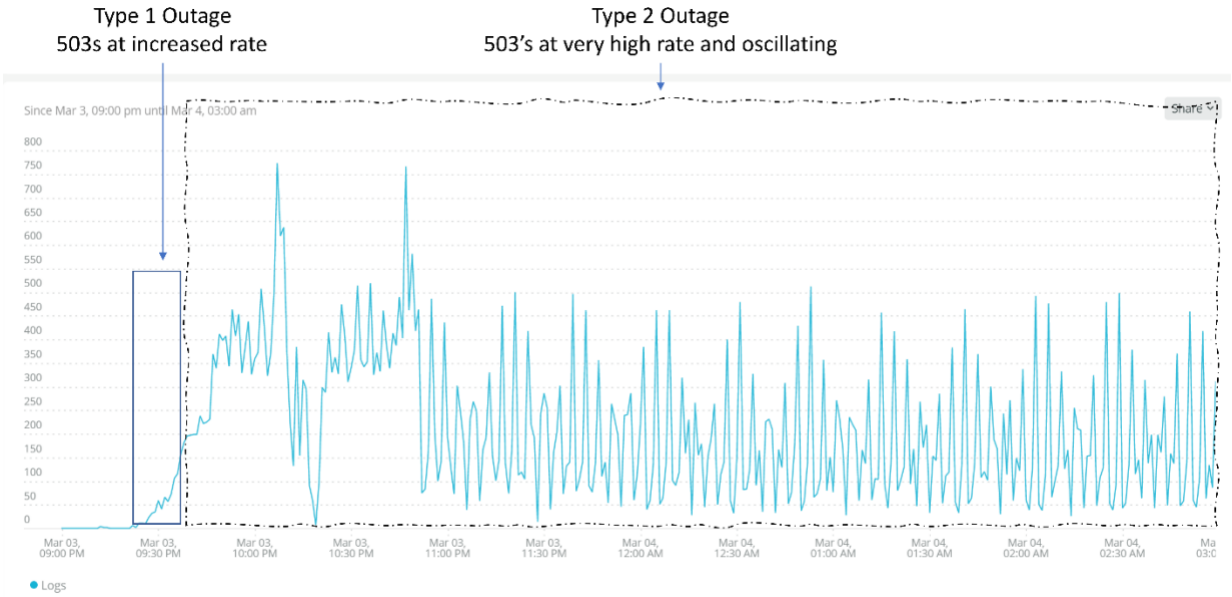


Figure 5: Chart of the CDN HTTP 503 errors for the same timeframe as Figure 5.

From the chart, we observe the following:

- 503 Errors occur on the Type 1 outage, with increased rate as time passes. As mentioned earlier, 503 errors and any 5XX errors in general are indicative, but do not define outages.
- 503 Errors occur throughout the entire Type 2 outage period and oscillate between very high values. These high error values, combined with the zero or almost zero values of cache latency, define the Type 2 outage.

The intensity of the 503 Errors can be viewed more clearly if the user calculates and plots the rate of 503 Errors relative to the total number of requests which reach the CDN. To calculate this rate (and any rate) the user may use the following query:

```
SELECT average(error/TOTAL) from ( Select filter(count(*), WHERE fb.input IS NULL and status LIKE '503') as 'error', filter(count(*), WHERE fb.input IS NULL ) as 'TOTAL' FROM Log Timeseries 1 minutes ) Timeseries 1 minutes SINCE '2021-03-03 21:00:00 UTC' UNTIL '2021-03-04 03:00:00 UTC'
```

Query 5

The notable parts of the above query are the following:

- The query consists of two individual "subqueries". The "subquery" which is the workhorse of the entire query is the one within the marked red parentheses. This subquery acts as the backbone of the entire query
- In the workhorse subquery, the number of 503 errors (indicated with yellow highlight) and the total number of requests (highlighted as orange) are calculated and named (labeled) as "error" and "TOTAL" respectively. The sampling for the "error" and "TOTAL" variables is done per 1 minute (indicated in light blue highlight). The names (labels) given to query results should not start with a number.

- Once “error” and “TOTAL” are calculated, the second “wrapping subquery” takes these values and calculates its average ratio, average(error/TOTAL), again on the same sampling rate of 1 minute (indicated in the dark blue highlight). The variables average(error/TOTAL) from the “wrapping subquery”, ‘error’ and ‘TOTAL’ from the workhorse subquery are highlighted in gray.
- The user may create queries similar to the above where different average ratios are calculated. For example, instead of a 503 error, the user is able to calculate all “5XX” errors (where XX could be any integer number) by replacing ‘503’ with ‘5%’

The result of the above query is described in the below chart:

Figure 6: Ratio of CDN HTTP 503 errors relative to the total amount of requests, in a 1-minute time increment. The timeframe is the same with the ones from Figure 4 and 5.

From the above chart (Figure 6) we observe that the ratio of requests with 503 error compared to the total number of requests is very high. Average is around 0.4, and sometimes even reaching values of > 0.80.

Low Intensity Outage cases

Introduction

In this section we describe an additional two sets of outages, which have a mild negative impact on a site’s server. These types of outages occur when the ratio of the HTTP requests with errors -mostly 5XX errors- relative to those with “healthy” response is significantly higher than normal, while the server still responds to a certain number of requests. A brief description and queries to study these outages are presented. As with the previous chapter, queries are used in New Relic Query Builder, but their logic can be applied to any Application Performance Suit.

Type 3 Outage

“Type 3” outages are cases where the CDN requests errors are significantly higher than what is considered as normal. In all site’s servers it is natural for certain errors to occur. But if a large portion of the requests end up with errors, we can conclude that the server is unable to respond to certain type – or certain traffic level – of requests, thus facing a lower magnitude outage.

When errors occur on the CDN level and it is certain that the server is still responding to a portion of the requests (the user can verify that by checking the cache latency), the "Type 3" outage occurs. To identify a "Type 3" outage, the user may employ Query 5, described in the previous chapter. If the ratio of errors surpasses a certain threshold for a continuous amount of time, the assessment could point to the outage. Thresholds for HTTP 503 errors are described in Table 2

Cases	Threshold – cutoff for errors/total requests
Healthy site's performance	$0 \leq r \leq 0.1$
Possible issue	$0.1 \leq r \leq 0.2$
Outage	$r > 0.2$

Table 2: Empirical thresholds of error ratio for outages.

Type 4 Outage

"Type 4" outages are cases very similar to the "Type 3", with only one main difference. The HTTP errors - again mostly 5XX – occur in the actual site's server, after they are sent by the CDN as "Misses". In such cases the CDN may not report any errors or high cache latency. The server errors however are significantly increased. To identify a "Type 4" outage, the user may employ the following query:

```
SELECT average(error/TOTAL) from ( Select filter(count(*), WHERE httpResponseCode Like '5%') as 'error', count(*) as 'TOTAL' FROM Transaction Timeseries 1 minutes) Timeseries 1 minutes SINCE '2021-03-03 21:00:00 UTC' UNTIL '2021-03-04 03:00:00 UTC'
```

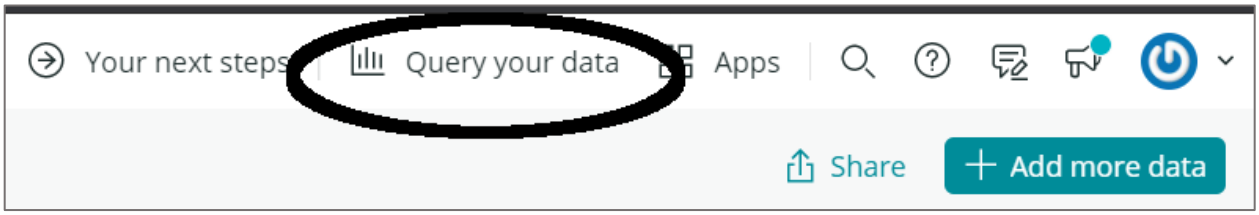
Query 6

For Query 6 we used the same timeframe as the one from Query 5 for simplicity. Note that for Query 6, we sample from the New Relic Table of "Transaction" (highlighted in yellow) which stores the transactions of the server. To identify an outage of this kind, the same thresholds described on Table 3 may be used.

Supplementary information

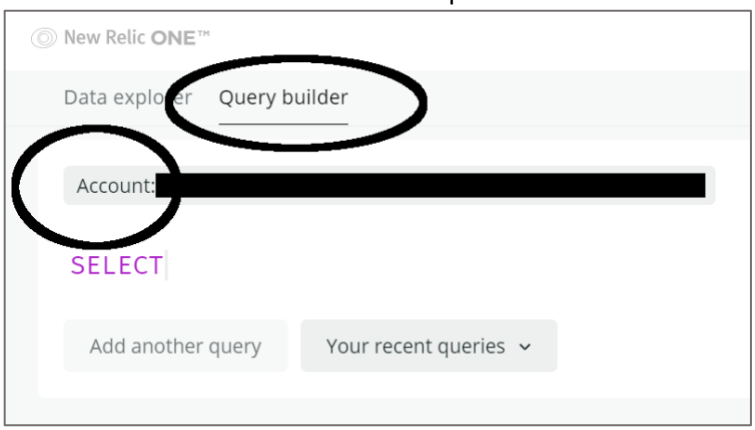
User's first steps to navigate into NR Queries:

1. When logged into New Relic One, click on **Query your data** on the upper right corner:



Depending on the zoom used on your browser only the **Query your data icon** may display:

2. In the upper left corner, if not selected, click on **Query builder**. Check that the account name and information shown in the drop-down menu is the one you wish to study:



You can now use the query line (starts with the SELECT command).