Nginx shows 2 different statuses in the upstream response log

Asked 3 years, 7 months ago Modified 3 years, 7 months ago Viewed 1k times

1

I am running an nginx-ingress controller in a kubernetes cluster and one of my log statements for the request looks like this:

```
upstream_response_length: 0, 840
upstream_response_time: 60.000, 0.760
upstream_status: 504, 200
```

I cannot quite understand what does that mean? Nginx has a response timeout equal to 60 seconds, and tries to request one more time after that (successfully) and logs both requests?

P.S. Config for log format:

```
log-format-upstream: >-
      {
      ...
      "upstream_status": "$upstream_status",
      "upstream_response_length": "$upstream_response_length",
      "upstream_response_time": "$upstream_response_time",
      ...
      }
```

nginx kubernetes nginx-ingress nginx-log

Share Improve this question

Follow

edited Sep 26, 2019 at 10:18

asked Sep 26, 2019 at 7:33

Ilya Chernomordik

27k ● 26 ● 114 ● 202

Would you please show your configs? – Yasen Sep 26, 2019 at 10:02

1 Answer

Sorted by:

Highest score (default)

2

According to `split_upstream_var` method of ingress-nginx, it splits results of `nginx` health checks.

Since `nginx` can have several upstreams, your log could be interpreted this way:

1. First upstream is dead (504)

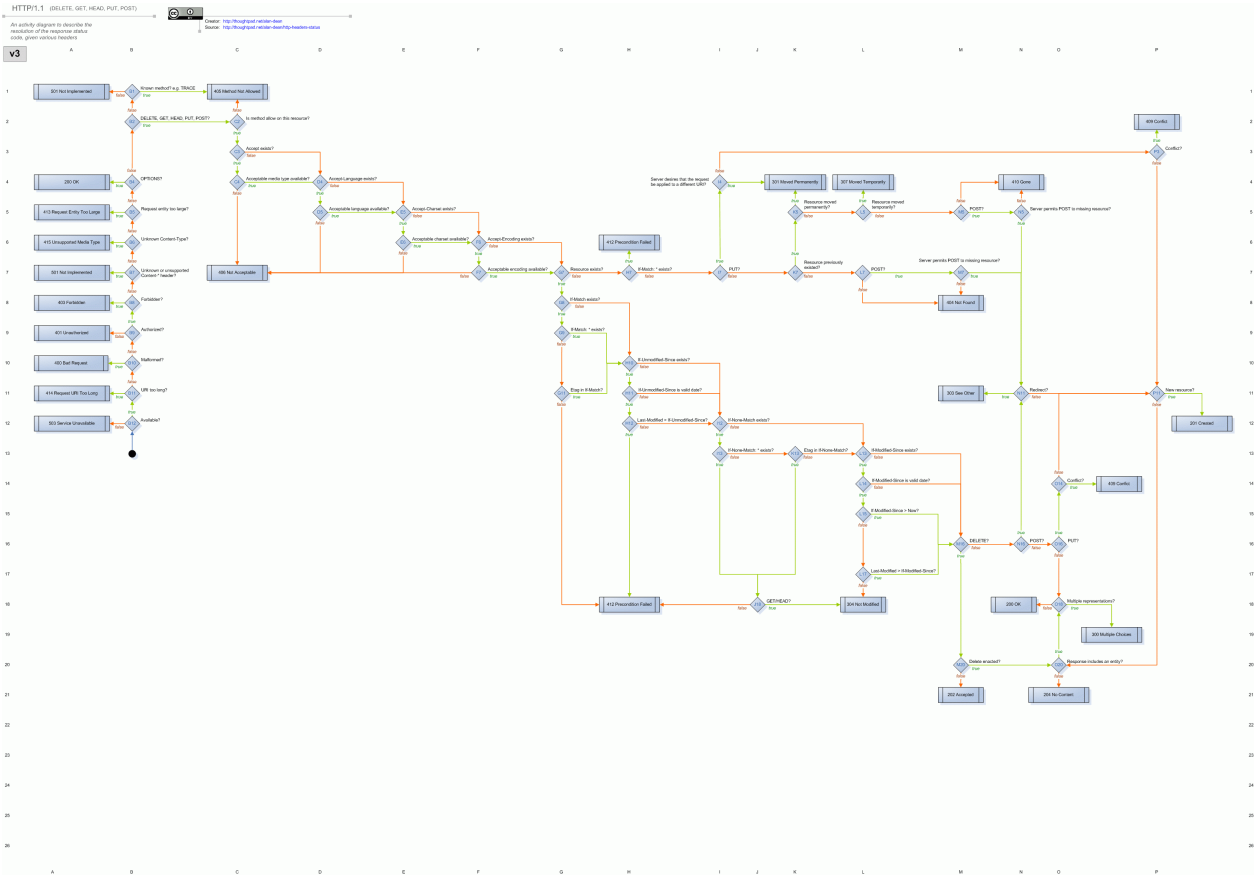
```
upstream_response_length: 0 // responce from dead upstream has zero length
upstream_response_time: 60.000 // nginx dropped connection after 60sec
```

```
upstream_status: 504 // responce code, upstream doesn't answer
```

2. Second upstream works (200)

```
upstream_response_length: 840 // healthy upstream returned 840b
upstream_response_time: 0.760 // healthy upstream responced in 0.760
upstream_status: 200 // responce code, upstream is ok
```

P.S. JFYI, here's a cool HTTP headers state diagram



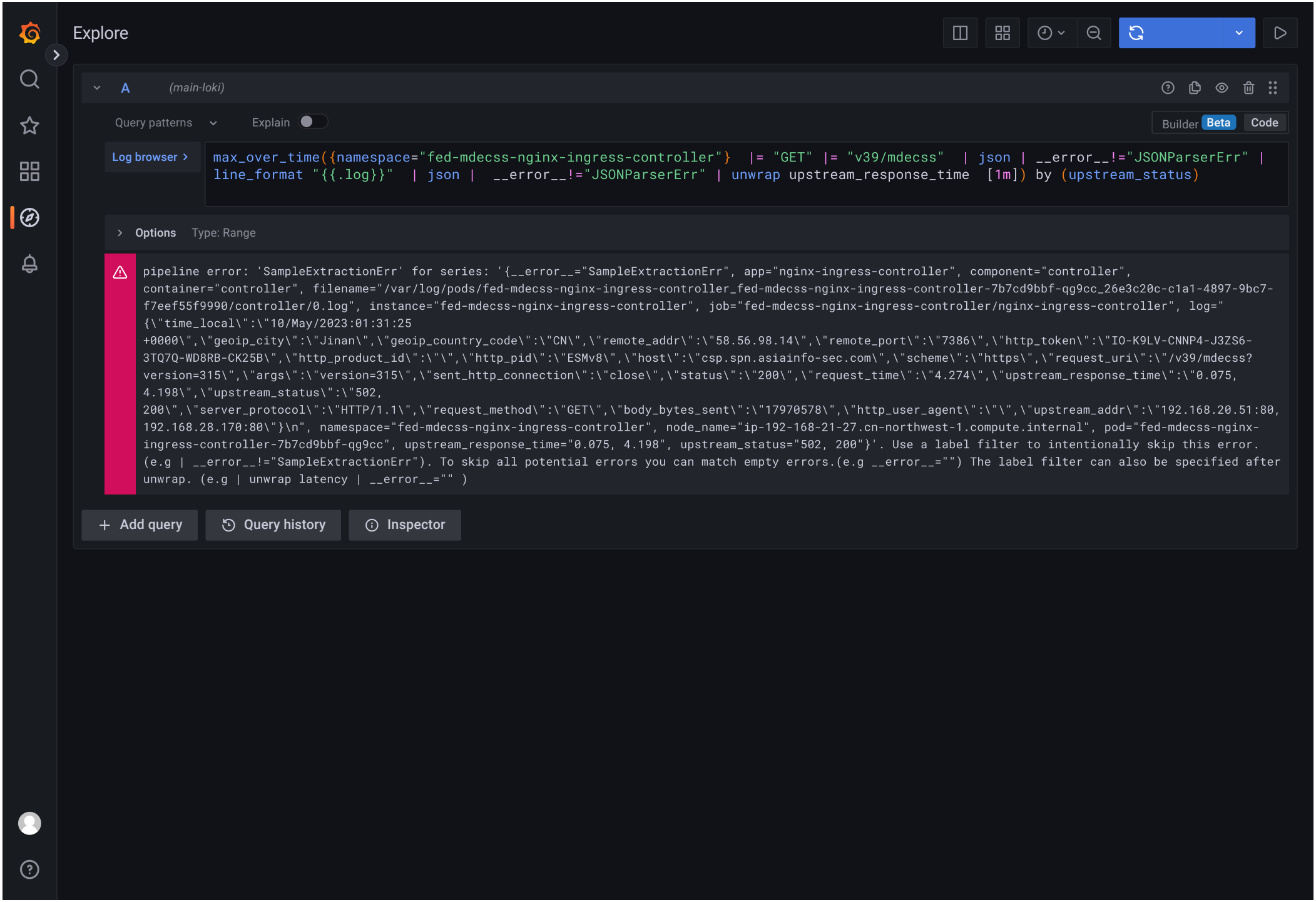
Share Improve this answer Follow edited Sep 26, 2019 at 9:27

answered Sep 26, 2019 at 9:22

Yasen

4,135 ● 1 ● 16 ● 25

I guess it's really just one upstream for nginx ingress controller, which is a service in kubernetes. Is it possible that nginx just does another request to upstream when the first is dropped after 60 seconds? – Ilya Chernomordik Sep 26, 2019 at 9:40


Explore
A
(main-loki)
Query patterns
Explain
Builder
Beta
Code
Log browser
max_over_time({namespace="fed-mdecss-nginx-ingress-controller"} |= "GET" |= "v39/mdecss" | json | __error__!="JSONParserErr" | line_format "{{.log}}" | json | __error__!="JSONParserErr" | unwrap upstream_response_time [1m]) by (upstream_status)
Options
Type: Range
pipeline error: 'SampleExtractionErr' for series: '{__error__="SampleExtractionErr", app="nginx-ingress-controller", component="controller", container="controller", filename="/var/log/pods/fed-mdecss-nginx-ingress-controller_fed-mdecss-nginx-ingress-controller-7b7cd9bbf-qg9cc_26e3c20c-c1a1-4897-9bc7-f7eef55f9990/controller/0.log", instance="fed-mdecss-nginx-ingress-controller", job="fed-mdecss-nginx-ingress-controller/nginx-ingress-controller", log="{\"time_local\":\"10/May/2023:01:31:25 +0000\",\"geoip_city\":\"Jinan\",\"geoip_country_code\":\"CN\",\"remote_addr\":\"58.56.98.14\",\"remote_port\":\"7386\",\"http_token\":\"IO-K9LV-CNNP4-J3ZS6-3TQ7Q-WD8RB-CK25B\",\"http_product_id\":\"\",\"http_pid\":\"ESMv8\",\"host\":\"csp.spn.asiainfo-sec.com\",\"scheme\":\"https\",\"request_uri\":\"/v39/mdecss?version=315\",\"args\":\"version=315\",\"sent_http_connection\":\"close\",\"status\":\"200\",\"request_time\":\"4.274\",\"upstream_response_time\":\"0.075, 4.198\",\"upstream_status\":\"502, 200\",\"server_protocol\":\"HTTP/1.1\",\"request_method\":\"GET\",\"body_bytes_sent\":\"17970578\",\"http_user_agent\":\"\",\"upstream_addr\":\"192.168.20.51:80, 192.168.28.170:80\"}\n", namespace="fed-mdecss-nginx-ingress-controller", node_name="ip-192-168-21-27.cn-northwest-1.compute.internal", pod="fed-mdecss-nginx-ingress-controller-7b7cd9bbf-qg9cc", upstream_response_time="0.075, 4.198", upstream_status="502, 200"}'. Use a label filter to intentionally skip this error. (e.g | __error__!="SampleExtractionErr"). To skip all potential errors you can match empty errors.(e.g __error__="") The label filter can also be specified after unwrap. (e.g | unwrap latency | __error__="")
Add query
Query history
Inspector