

为什么要做NGINX日志分析看板

Grafana官网的dashboards有NGINX日志采集到ES数据源的展示看板，也有采集到LOKI数据源的展示看板，唯独没有采集到ClickHouse数据源的展示看板。所以这个轮子是必须要造的。

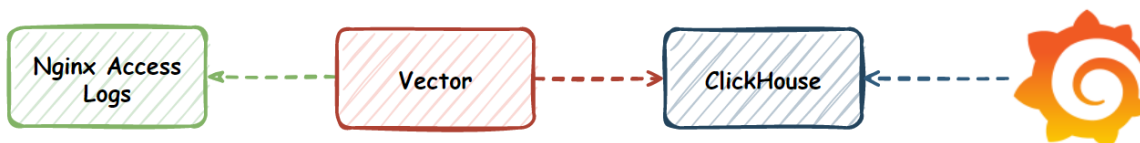
为什么不使用ES存储

ElasticSearch是全文检索引擎的文档数据库，对于业务日志、异常日志、多行日志这类，非结构化、半结构化的日志数据，经常需要做关键字查询，模糊匹配等操作，非常适合使用es，使用倒排索引实现快速全文搜索。

ClickHouse是一个列式存储数据库，尤其擅长处理结构化的大规模的SQL查询和聚合分析操作，所以针对NGINX这类结构化的请求日志，在处理多维分析、聚合查询、分组统计等操作速度极快，并且压缩比极高，存储成本比ES低10倍，CPU、内存的占用也有巨大优势。

NGINX日志采集架构

- 基础架构



- 完整架构

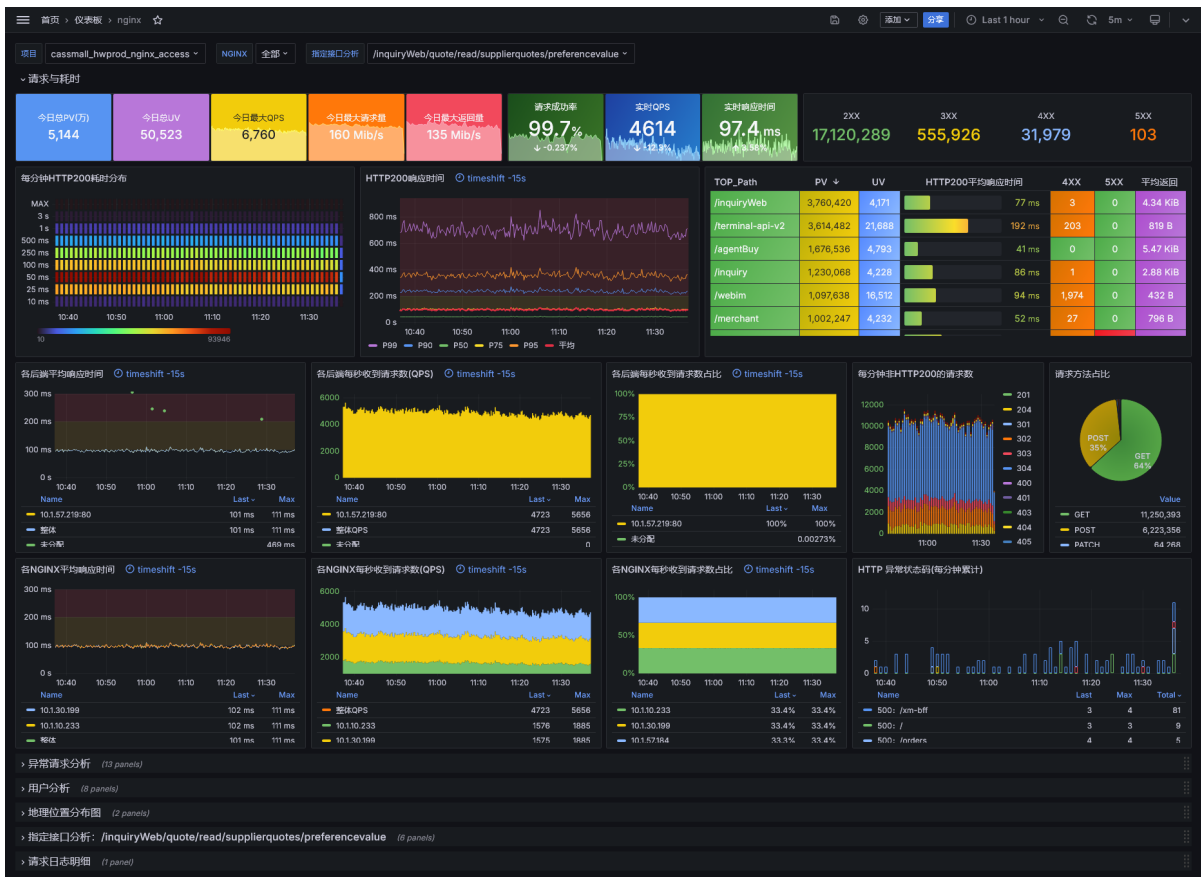


Grafana请求日志分析看板预览

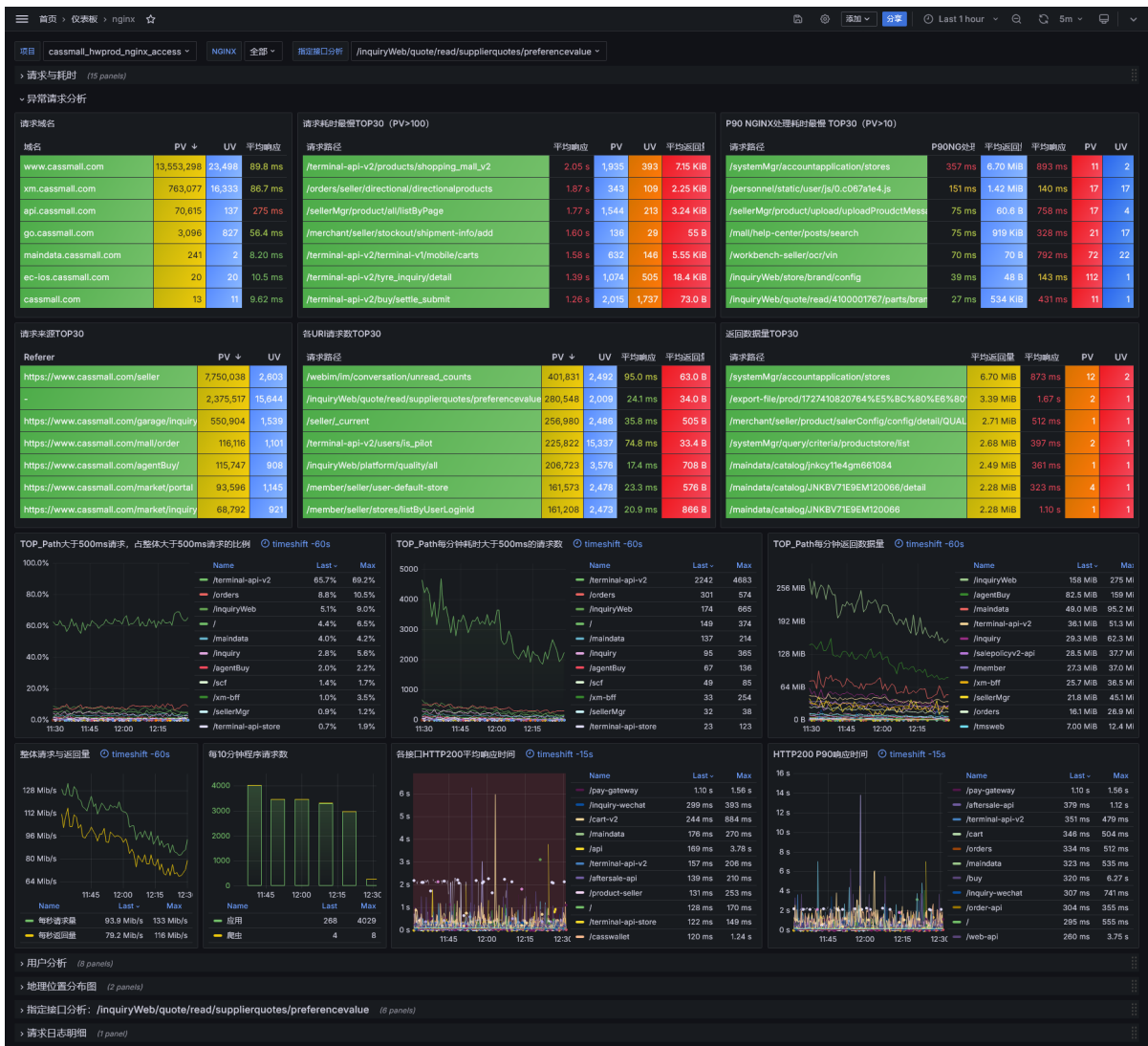
该看板是基于 ClickHouse + Vector 的NGINX请求日志分析看板。包括请求与耗时分析、异常请求分析、用户分析、地理位置分布图、指定接口分析、请求日志明细。

尤其在异常请求分析方面，总结多年异常请求分析经验，从各个角度设计大量异常请求的分析图表。

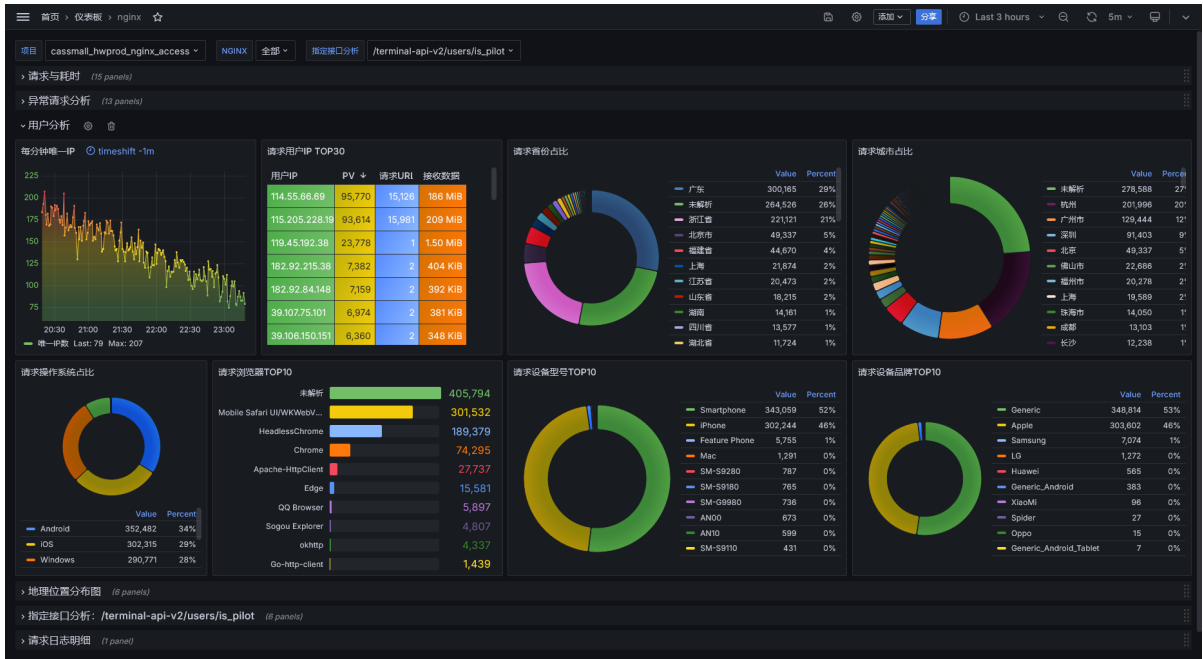
- 整体请求与耗时分析



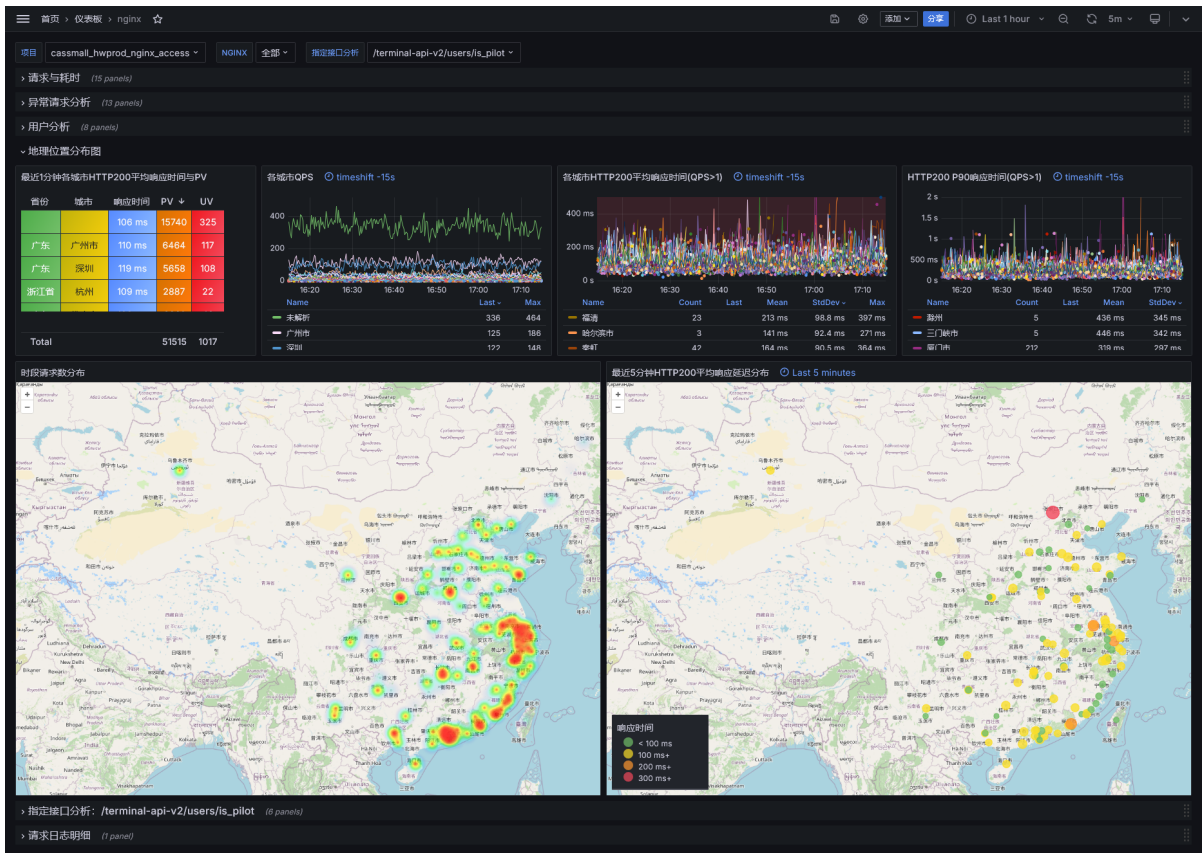
● NGINX异常请求分析



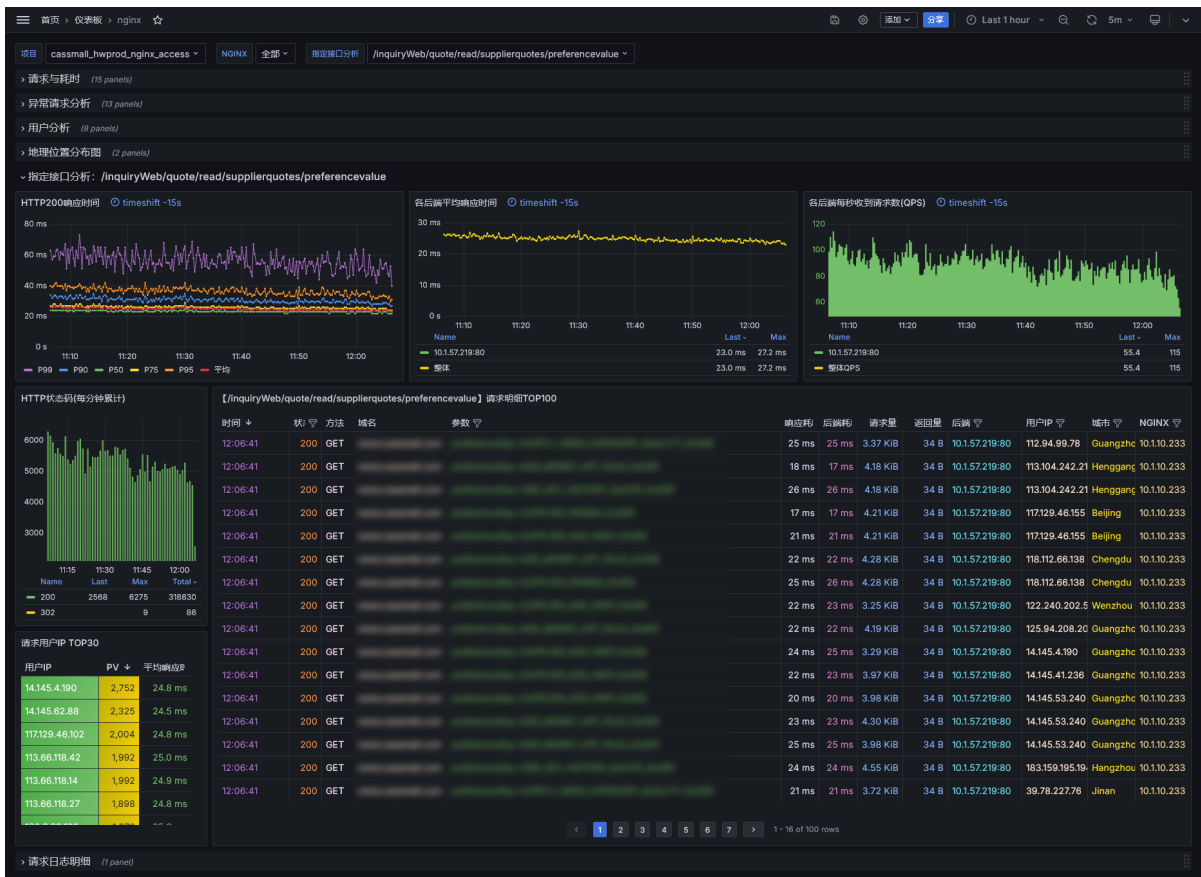
• 用户请求数据分析



• 地理位置数据分析



• 指定接口明细分析



- 请求日志详情分析

时间	状态	方法	域名	参数	响应码	后端	请求量	返回量	后端	用户IP	城市	NGINX
12:07:50	200	GET			10 ms	9 ms	3.41 KIB	50 B	10.157.219:80	14.145.56.203	Guangzhou	10.130.199
12:07:50	200	GET			9 ms	9 ms	3.83 KIB	28 B	10.157.219:80	183.135.98.221	Ningbo	10.130.199
12:07:50	200	GET			12 ms	12 ms	3.91 KIB	50 B	10.157.219:80	118.113.218.24	Chengdu	10.130.199
12:07:50	200	POST			18 ms	19 ms	4.24 KIB	62 B	10.157.219:80	223.73.5710	Guangzhou	10.157.184
12:07:50	200	POST			18 ms	19 ms	3.58 KIB	33 B	10.157.219:80	14.145.61.77	Guangzhou	10.157.184
12:07:50	200	POST			21 ms	22 ms	1.12 KIB	56 B	10.157.219:80	182.92.84.148	Beijing	10.130.199
12:07:50	200	POST			23 ms	24 ms	1.58 KIB	25 B	10.157.219:80	221.237.24.21	Chengdu	10.130.199
12:07:50	200	GET			25 ms	25 ms	3.91 KIB	190 B	10.157.219:80	118.113.218.24	Chengdu	10.157.184
12:07:50	200	GET			24 ms	25 ms	3.32 KIB	818 B	10.157.219:80	218.16.62.226		10.130.199
12:07:50	200	GET			30 ms	29 ms	2.05 KIB	27 B	10.157.219:80	120.231.13.116		10.130.199
12:07:50	200	GET			31 ms	30 ms	1.51 KIB	632 B	10.157.219:80	14.26.170.247	Jiangmen	10.130.199
12:07:50	200	GET			29 ms	30 ms	1.47 KIB	206 B	10.157.219:80	118.114.5715	Chengdu	10.130.199
12:07:50	200	GET			31 ms	31 ms	3.71 KIB	369 B	10.157.219:80	39.78.149.69	Jinan	10.157.184
12:07:50	200	POST			31 ms	31 ms	1.60 KIB	25 B	10.157.219:80	171.124.16.107		10.130.199
12:07:50	200	GET			32 ms	31 ms	2.05 KIB	27 B	10.157.219:80	120.231.13.116		10.130.199
12:07:50	200	GET			32 ms	32 ms	3.13 KIB	21 B	10.157.219:80	125.85.80.215	Chongqing	10.157.184
12:07:50	200	PUT			32 ms	32 ms	1.55 KIB	16 B	10.157.219:80	218.91.136.183		10.130.199
12:07:50	200	GET			33 ms	33 ms	4.12 KIB	335 B	10.157.219:80	113.66.239.75	Guangzhou	10.157.184
12:07:50	200	POST			32 ms	33 ms	3.37 KIB	53 B	10.157.219:80	180.139.166.10		10.130.199
12:07:50	200	GET			38 ms	38 ms	1.51 KIB	92 B	10.157.219:80	120.235.71.8		10.130.199

修改NGINX日志格式

编辑NGINX配置文件 nginx.conf (已配置为毫秒级别的时间格式)

```
1 map "$time_iso8601 # $msec" $time_iso8601_ms { "~(^[^+]+)(\+[0-9:]+) #  
  \d+\.\.(d+)$" $1.$3$2; }  
2 log_format main  
3 '{"timestamp": "$time_iso8601_ms",'
```

```

4     "server_ip": "$server_addr", '
5     "remote_ip": "$remote_addr", '
6     "xff": "$http_x_forwarded_for", '
7     "remote_user": "$remote_user", '
8     "domain": "$host", '
9     "url": "$request_uri", '
10    "referer": "$http_referer", '
11    "upstreamtime": "$upstream_response_time", '
12    "responsetime": "$request_time", '
13    "request_method": "$request_method", '
14    "status": "$status", '
15    "response_length": "$bytes_sent", '
16    "request_length": "$request_length", '
17    "protocol": "$server_protocol", '
18    "upstreamhost": "$upstream_addr", '
19    "http_user_agent": "$http_user_agent"
20    '};

```

```

1 # 配置完成后, 重载NGINX
2 nginx -s reload

```

部署ClickHouse

```

1 # 创建部署目录和docker-compose.yaml
2 mkdir -p /opt/clickhouse/etc/clickhouse-server/{config.d,users.d}
3 cd /opt/clickhouse
4 cat <<-EOF > docker-compose.yaml
5 services:
6   clickhouse:
7     image: registry.cn-shenzhen.aliyuncs.com/stars1/clickhouse-server:23.4
8     container_name: clickhouse
9     hostname: clickhouse
10    volumes:
11      - /opt/clickhouse/logs:/var/log/clickhouse-server
12      - /opt/clickhouse/data:/var/lib/clickhouse
13      - /opt/clickhouse/etc/clickhouse-
server/config.d/config.xml:/etc/clickhouse-server/config.d/config.xml
14      - /opt/clickhouse/etc/clickhouse-
server/users.d/users.xml:/etc/clickhouse-server/users.d/users.xml
15      - /usr/share/zoneinfo/PRC:/etc/localtime
16    ports:
17      - 8123:8123
18      - 9000:9000
19 EOF

```

```

1 # vi /opt/clickhouse/etc/clickhouse-server/config.d/config.xml
2 <clickhouse replace="true">
3   <logger>
4     <level>debug</level>
5     <log>/var/log/clickhouse-server/clickhouse-server.log</log>
6     <errorlog>/var/log/clickhouse-server/clickhouse-
server.err.log</errorlog>
7     <size>1000M</size>

```

```

8         <count>3</count>
9     </logger>
10    <display_name>ch_accesslog</display_name>
11    <listen_host>0.0.0.0</listen_host>
12    <http_port>8123</http_port>
13    <tcp_port>9000</tcp_port>
14    <user_directories>
15        <users_xml>
16            <path>users.xml</path>
17        </users_xml>
18        <local_directory>
19            <path>/var/lib/clickhouse/access/</path>
20        </local_directory>
21    </user_directories>
22 </clickhouse>

```

```

1  # 生成密码(返回的第一行是明文，第二行是密文)
2  PASSWORD=$(base64 < /dev/urandom | head -c8); echo "$PASSWORD"; echo -n
   "$PASSWORD" | sha256sum | tr -d '-'
3
4  # vi /opt/clickhouse/etc/clickhouse-server/users.d/users.xml
5  <?xml version="1.0"?>
6  <clickhouse replace="true">
7      <profiles>
8          <default>
9              <max_memory_usage>10000000000</max_memory_usage>
10             <use_uncompressed_cache>0</use_uncompressed_cache>
11             <load_balancing>in_order</load_balancing>
12             <log_queries>1</log_queries>
13         </default>
14     </profiles>
15     <users>
16         <default>
17             <password remove='1' />
18             <password_sha256_hex>填写生成的密码密文</password_sha256_hex>
19             <access_management>1</access_management>
20             <profile>default</profile>
21             <networks>
22                 <ip>::/0</ip>
23             </networks>
24             <quota>default</quota>
25             <access_management>1</access_management>
26             <named_collection_control>1</named_collection_control>
27             <show_named_collections>1</show_named_collections>
28
29             <show_named_collections_secrets>1</show_named_collections_secrets>
30         </default>
31     </users>
32     <quotas>
33         <default>
34             <interval>
35                 <duration>3600</duration>
36                 <queries>0</queries>
37                 <errors>0</errors>
38                 <result_rows>0</result_rows>

```



```

38         <read_rows>0</read_rows>
39         <execution_time>0</execution_time>
40     </interval>
41 </default>
42 </quotas>
43 </clickhouse>

```

```

1 # 启动
2 docker compose up -d

```

创建数据库与表

```

1 CREATE DATABASE IF NOT EXISTS nginxlogs ENGINE=Atomic;
2
3 CREATE TABLE nginxlogs.nginx_access
4 (
5     `timestamp` DateTime64(3, 'Asia/Shanghai'),
6     `server_ip` String,
7     `domain` String,
8     `request_method` String,
9     `status` Int32,
10    `top_path` String,
11    `path` String,
12    `query` String,
13    `protocol` String,
14    `referer` String,
15    `upstreamhost` String,
16    `responsetime` Float32,
17    `upstreamtime` Float32,
18    `duration` Float32,
19    `request_length` Int32,
20    `response_length` Int32,
21    `client_ip` String,
22    `client_latitude` Float32,
23    `client_longitude` Float32,
24    `remote_user` String,
25    `remote_ip` String,
26    `xff` String,
27    `client_city` String,
28    `client_region` String,
29    `client_country` String,
30    `http_user_agent` String,
31    `client_browser_family` String,
32    `client_browser_major` String,
33    `client_os_family` String,
34    `client_os_major` String,
35    `client_device_brand` String,
36    `client_device_model` String,
37    `createdtime` DateTime64(3, 'Asia/Shanghai'),
38 )
39 ENGINE = MergeTree
40 PARTITION BY toYYYYMMDD(timestamp)
41 PRIMARY KEY (timestamp,
42     server_ip,

```

```

43     status,
44     top_path,
45     domain,
46     upstreamhost,
47     client_ip,
48     remote_user,
49     request_method,
50     protocol,
51     responsetime,
52     upstreamtime,
53     duration,
54     request_length,
55     response_length,
56     path,
57     referer,
58     client_city,
59     client_region,
60     client_country,
61     client_browser_family,
62     client_browser_major,
63     client_os_family,
64     client_os_major,
65     client_device_brand,
66     client_device_model
67 )
68 TTL toDateTime(timestamp) + toIntervalDay(90)
69 SETTINGS index_granularity = 8192;

```

部署Vector采集日志

```

1  # 创建部署目录和docker-compose.yml
2  mkdir -p /opt/vector/conf
3  cd /opt/vector
4  touch access_vector_error.log
5  wget GeoLite2-City.mmdb
6  cat <<-EOF > docker-compose.yml
7  services:
8    vector:
9      image: registry.cn-shenzhen.aliyuncs.com/stars1/vector:0.41.1-alpine
10     container_name: vector
11     hostname: vector
12     restart: always
13     entrypoint: vector --config-dir /etc/vector/conf
14     ports:
15       - 8686:8686
16     volumes:
17       - /usr/local/openresty/nginx/logs:/nginx_logs # 这是需要采集的日志的路径需
要挂载到容器内
18       - /opt/vector/access_vector_error.log:/tmp/access_vector_error.log
19       - /opt/vector/GeoLite2-City.mmdb:/etc/vector/GeoLite2-City.mmdb
20       - /opt/vector/conf:/etc/vector/conf
21       - /usr/share/zoneinfo/PRC:/etc/localtime
22 EOF

```



```
1 # conf目录采集配置
2 cd /opt/vector/conf
3 cat <<-EOF > vector.yaml
4 timezone: "Asia/Shanghai"
5 api:
6   enabled: true
7   address: "0.0.0.0:8686"
8 EOF
```

```
1 # vi nginx-access.yaml
2 sources:
3   01_file_nginx_access:
4     type: file
5     include:
6       - /nginx_logs/access.log #nginx请求日志路径(注意是挂载到容器内的路径)
7 transforms:
8   02_parse_nginx_access:
9     drop_on_error: true
10    reroute_dropped: true
11    type: remap
12    inputs:
13      - 01_file_nginx_access
14    source: |
15      .message = string!(.message)
16      if contains(.message,"\\x") { .message = replace(.message, "\\x",
17      "\\\"\\x") }
18      . = parse_json!(.message)
19      .createdtime = to_unix_timestamp(now(), unit: "milliseconds")
20      .timestamp = to_unix_timestamp(parse_timestamp!(.timestamp , format:
21      "%+"), unit: "milliseconds")
22      .url_list = split!(.url, "?", 2)
23      .path = .url_list[0]
24      .query = .url_list[1]
25      .path_list = split!(.path, "/", 3)
26      if length(.path_list) > 2 {.top_path = join!(["/", .path_list[1]])}
27      else {.top_path = "/" }
28      .duration = round(((to_float(.responsetime) ?? 0) -
29      (to_float(.upstreamtime) ?? 0)) ?? 0,3)
30      if .xff == "-" { .xff = .remote_ip }
31      .client_ip = split!(.xff, ",", 2)[0]
32      .ua = parse_user_agent!(.http_user_agent , mode: "enriched")
33      .client_browser_family = .ua.browser.family
34      .client_browser_major = .ua.browser.major
35      .client_os_family = .ua.os.family
36      .client_os_major = .ua.os.major
37      .client_device_brand = .ua.device.brand
38      .client_device_model = .ua.device.model
39      .geoip = get_enrichment_table_record("geoip_table", {"ip": .client_ip})
40      ?? {"city_name":"unknown","region_name":"unknown","country_name":"unknown"}
41      .client_city = .geoip.city_name
42      .client_region = .geoip.region_name
43      .client_country = .geoip.country_name
44      .client_latitude = .geoip.latitude
45      .client_longitude = .geoip.longitude
```

```

41     del(.path_list)
42     del(.url_list)
43     del(.ua)
44     del(.geoiip)
45     del(.url)
46 sinks:
47     03_ck_nginx_access:
48         type: clickhouse
49         inputs:
50             - 02_parse_nginx_access
51         endpoint: http://10.7.0.26:8123 #clickhouse http接口
52         database: nginxlogs #clickhouse 库
53         table: nginx_access #clickhouse 表
54         auth:
55             strategy: basic
56             user: default #clickhouse 库
57             password: GlwszBQp #clickhouse 密码
58         compression: gzip
59     04_out_nginx_dropped:
60         type: file
61         inputs:
62             - 02_parse_nginx_access.dropped
63         path: /tmp/access_vector_error.log #解析异常的日志
64         encoding:
65             codec: json
66     enrichment_tables:
67         geoiip_table:
68             path: "/etc/vector/GeoLite2-City.mmdb"
69             type: geoiip
70             locale: "zh-CN"

```

```

1 # 启动
2 cd /opt/vector
3 docker compose up -d

```

Grafana新增ClickHouse数据源

在Grafana中增加ClickHouse数据源时，注意点开 Additional settings 右边的箭头，配置 Default database 为存放日志的默认库，如上的：nginxlogs。

导入NGINX请求日志分析的Grafana看板

Grafana看板ID：22037

下载地址：

<https://grafana.com/grafana/dashboards/22037>

注意：如果你保存日志的表名不是 access 结尾的，项目 菜单会没有数据，需要点击看板右上角的设置 - 变量 - project，在下方的 Regex 项，输入你需要展示的日志表的正则，或者留空，展示默认库的所有表。