sentry消息队列方案

快速入门:

Sentry 环境搭建

Sentry 架构

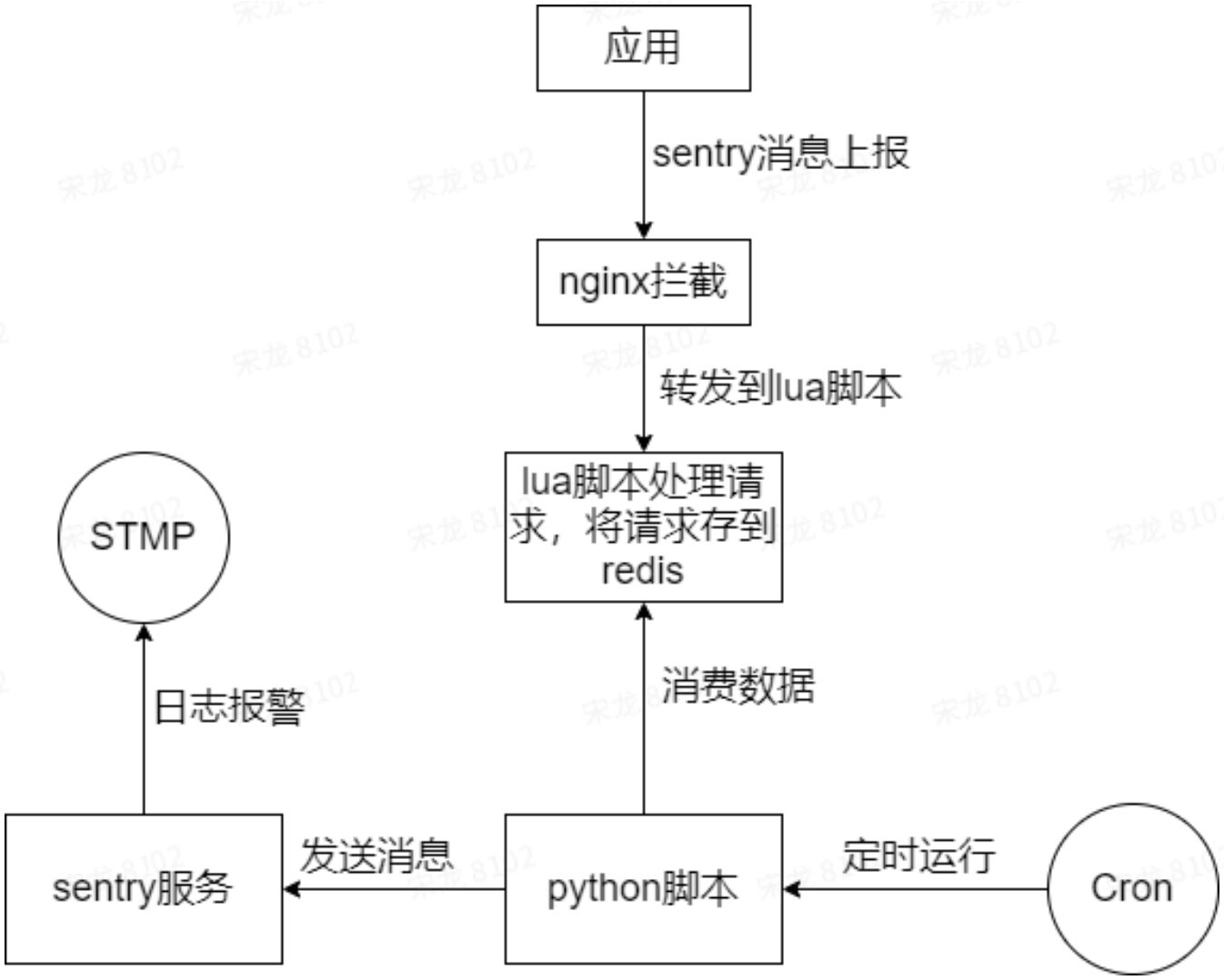
目的:

1. 将现有的sentry告警模式改为：应用-->openresty-->redis<--python脚本-->sentry服务

方案:

- nginx将应用发出的sentry告警消息转发到lua脚本，lua脚本获取该请求的uri、header、body存到redis里，python脚本从redis里获取数据，requests.post()发送到sentry服务；

运行流程:



如何使用：

- 在项目中使用只需将sentry的配置url改为统一形式即可，以前的sentry配置为

```
https://sentry_key:sentry_secret@wow.sentry.ktvsky.com/project_num
//例如vod_be
http://a6f431e198734105aef464f6e702351f:563b78f7bc1941828df0e5ebd79cc17a@wow.sentry.ktvsky.com/20
```

- 采用此方案后:修改后的sentry_key为指定的sentry服务如wow、ktv、ad，project_name为项目名称，域名固定的sentry.quene.com，project_name可以写任意值

```
https://wow:projectname@sentry.quene.com/project_num
//例如
http://wow:vod_be@sentry.quene.com/1
```

搭建过程：

step1:

首先需修改自己机器的hosts文件，将指定域名的请求映射到指定域名

```
106.75.34.22 sentry.quene.com
```

step2:

nginx将该域名的请求转发到lua脚本处理

nginx配置文件:

```
server {
 listen 80;
 server_name sentry.quene.com;

 location / {
 charset utf-8;
 lua_need_request_body on;
 rewrite_by_lua_file /home/work/nginx/conf/lua/sentry.lua;
 }
}
```

step3:

lua脚本获取该消息的请求头，uri和body，并将其存储到redis

注意：由于sentry告警消息的body为加密信息，所以需要将消息的body与请求头、uri分开存储。也是因为这个原因无法获得sentry body中的具体信息，原方案由python raven包发送告警消息到

sentry，改为了用requests.post()方式，body不做任何处理存入redis，取出后自己构造消息的组成，将body原封不动的发送到sentry服务

lua脚本:

```lua
local redis = require "resty.redis"
local cjson = require "cjson"

local r = redis:new()
local ok, err = r:connect("10.10.52.29", 6379)
if not ok then
 return
end

--local uri_args = ngx.req.get_uri_args()
local body = ngx.req.get_body_data()
local header = ngx.req.get_headers()
local request_uri = ngx.var.request_uri

now = os.time()
local obj = {
 uri = request_uri,
 timestamp = now,
 header = header
}
--时间戳作为obj的一个参数，并作为body的key
jsonData = cjson.encode(obj)
ok,err = r:lpush("sentry_header_uri",jsonData)
if not ok then
 ngx.say("set data error",err)
 return
end

body_key = "sentry:body:"..now
ok,err = r:set(body_key,body)
if not ok then
 ngx.say("set body error")
end
```

step4:

python脚本从redis取得数据，根据header中的信息从配置文件中找到sentry_key、sentry_secret、project_num，再与body构造成requests.post()请求并发送到sentry服务，如消息发送失败则将数据重新写入redis，下一次脚本执行再做处理

python脚本:

```
import time
import requests
import redis
import json

config = {
 "wow": {
 "url": "http://10.10.153.224:9001",
 "vod_be": {
 "key": "a6f431e198734105aef464f6e702351f",
 "secret": "563b78f7bc1941828df0e5ebd79cc17a",
 "num": "20"
 },
 "mcms": {
 "key": "35836757370643db9cf51fe2df425bc9",
 "secret": "8ca3a8825fd24fada3a47420e55e40f0",
 "num": "10"
 },
 "vod_li": {
 "key": "aead908e4b4e4480a700f85021d4849e",
 "secret": "cb28f0b991d64c31989bb4822a0310bf",
 "num": "23"
 },
 "wow_user": {
 "key": "3a598aef625243a49800294315600cee",
 "secret": "cac677ca2c014c969869570f60c9711d",
 "num": "2"
 }
 },
 "ktv": {
 "url": "http://10.10.153.224:9002",
 "vadd_stb1": {
 "key": "7ca45e28b66f4e86a13cdd9ff28638ac",
 "secret": "c2604aa199744814a1b84b595cbb9067",
 "num": "6"
 },
 "lscms": {
```

```
 "key": "ede4aa042e6e4da4bb650421eb44bd11",
 "secret": "497ade054d96464bbc84aaba96f1dbef",
 "num": "11"
 },
 "value_added": {
 "key": "1d631feadae24c808d2a4ba87010d005",
 "secret": "2597e93c26a046a49065183079748552",
 "num": "2"
 }
 },
 "default": {
 "url": "http://10.10.153.224:9001",
 "default": {
 "key": "a6f431e198734105aef464f6e702351f",
 "secret": "563b78f7bc1941828df0e5ebd79cc17a",
 "num": "20"
 }
 }

}

r = redis.StrictRedis(host="10.10.52.29", port="6379")


class Redis():
 def get_info(self):
 a = r.rpop("sentry_header_uri")
 if a is None:
 return None, None, None
 data = json.loads(a.decode())
 body_key = "sentry:body:{}".format(data["timestamp"])
 body = r.get(body_key)
 return data, body, body_key


class Sentry(object):
 def run(self, data, body, body_key):
 headers, u = self.create_sentry(data["header"], data["uri"])
 res = self.sentry_post(u, headers, body)
 if res.status_code == 200:
 r.delete(body_key)
 else:
 r.lpush("sentry_header_uri", data["header"])
 print(res.text)

 def create_sentry(self, header, uri):
```

```
    k_index = -1
    s_index = -1
    sentryInfo = header['x-sentry-auth'].split(' ')
    for i in range(len(sentryInfo)):
    if "key" in sentryInfo[i]:
    k_index = i
    flag = sentryInfo[i][sentryInfo[i].index("=") + 1:].replace(",", "", -1)
    elif "secret" in sentryInfo[i]:
    s_index = i
    project = sentryInfo[i][sentryInfo[i].index("=") + 1:].replace(",", "", -1)
    if flag not in config.keys():
    flag = "default"
    project = "default"

    sentry_key = "sentry_key=" + config[flag][project]["key"] + ","
    sentry_secret = "sentry_secret=" + config[flag][project]["secret"]

    if k_index == -1:
    sentryInfo.append(sentry_key)
    elif 0 < k_index < len(sentryInfo):
    sentryInfo[k_index] = sentry_key
    if s_index == -1:
    sentryInfo.append(sentry_secret)
    elif 0 < s_index < len(sentryInfo):
    sentryInfo[s_index] = sentry_secret
    header['x-sentry-auth'] = " ".join(sentryInfo)

    url = config[flag]["url"]
    uri_arr = uri.split("/")
    for i in range(len(uri_arr)):
    if uri_arr[i].isdigit():
    uri_arr[i] = config[flag][project]["num"]
    uri = "/".join(uri_arr)
    u = url + uri
    return header, u

    def sentry_post(self, u, headers, body):
    return requests.post(url=u, headers=headers, data=body)


if __name__ == '__main__':
    while True:
    data, body, body_key = Redis().get_info()
    if data is None:
    time.sleep(10)
    else:
```

```
130  Sentry().run(data, body, body_key)
```