<자료: 2021-1-02-07>

#개발자료 모음

Soulmates

김인석

1.Nginx conf파일모음

I.nginx.conf

```
user  nginx;
worker_processes  auto;

error_log  /var/log/nginx/error.log notice;
pid        /var/run/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;

    include /etc/nginx/conf.d/*.conf;


    fastcgi_buffers 8 16k;
    fastcgi_buffer_size 32k;
    fastcgi_connect_timeout 300000;
    fastcgi_send_timeout 300000;
    fastcgi_read_timeout 300000;


}
```

II.vallay.conf

```
map $http_authorization $access_token {
    "~*^bearer (.*)$" $1;
    default $http_authorization;
}


server {

        listen 8282;
        server_name soulmates.onstove.com;


        large_client_header_buffers 4 16k;
        error_page  405     =200 $uri;
        error_page  401     =200 $uri;
        location / {
                #502 bad gateway 에슈
                resolver 172.32.0.2 valid=10s;
                keepalive_requests 10000;
                keepalive_timeout 75000;
                #500 error 에슈i
                proxy_buffers 4 256k;
                proxy_set_header X-SSL-CLIENT-CERT $ssl_client_cert;
                proxy_busy_buffers_size 256k;
                proxy_buffer_size  32k;


                proxy_redirect off;
                proxy_pass_header Server;
                #proxy_set_header Host $host;
                proxy_set_header Host $http_host;
                proxy_set_header X-Forwarded-Host $server_name;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Scheme $scheme;
                proxy_intercept_errors on;

                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "Upgrade";

                proxy_buffering   off;
                proxy_set_header Upgrade websocket;

                #proxy_connect_timeout 100000000;
                #proxy_send_timeout 100000000;
                #proxy_read_timeout 100000000;

                #proxy_pass  http://10.250.93.80:3100/socket.io/;

        }


        location ~* ₩.io {
                add_header 'Access-Control-Allow-Origin' '*';
                proxy_connect_timeout 1000s;
                proxy_read_timeout  3600s;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;
                proxy_set_header X-NginX-Proxy false;
                proxy_ssl_session_reuse   off;
                proxy_redirect off;
```

2. rabbitmq config파일

```
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";
            proxy_pass "http://10.250.93.80:3100";
}
     location /socket.io/ {

            proxy_pass http://10.250.93.80:3100/socket.io/;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";
            proxy_redirect off;

            proxy_buffers 8 32k;
            proxy_buffer_size 64k;

            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header Host $http_host;
            proxy_set_header X-NginX-Proxy true;
    }

    location /socket.io/socket.io.js {
            proxy_pass http://10.250.93.80:3100;
    }

    location /chat/ {
            proxy_pass http://10.250.93.80:3100;
    }


    location /users/ {
            #proxy_set_header HTTP_AUTHORIZATION $http_authorization;
            proxy_pass http://10.250.93.80:3000;
    }

    location /follow/ {
            proxy_pass http://10.250.93.85:8040;
    }

    location /post {
            proxy_pass http://10.250.93.85:8030;
    }

    location /like {
            proxy_pass http://10.250.93.85:8030;
    }

    location /feed/ {
            proxy_pass http://10.250.93.85:8030;
    }

    location /comment {
           proxy_pass http://10.250.93.85:8030;
    }

    location /profile/{
            proxy_pass http://10.250.93.80:3000;
    }

    location /noti/{
            proxy_pass http://10.250.93.80:3001;
    }
```

I.Valleyserver.conf

```
input {
    rabbitmq {
        host => "10.250.93.126:5672"
        user => "admin"
        password => "1234"
        queue => "postCreated"
        #exchange => "feed.topic"
        #exchange_type => "topic"
        #key => "post.create"
        durable => true
        vhost => "/"
        type => 'feed'

    }

  rabbitmq {
        host => "10.250.93.126:5672"
        user => "admin"
        password => "1234"
        queue => "createdUser"
        #exchange => "feed.topic"
        #exchange_type => "topic"
        #key => "users.create"
        durable => true
        vhost => "/"
        type => 'user'

    }

  rabbitmq {
        host => "10.250.93.126:5672"
        user => "admin"
        password => "1234"
        queue => "collectHashTag"
        #exchange => "feed.topic"
        #exchange_type => "topic"
        #key => "users.create"
        durable => true
        vhost => "/"
        type => 'hashTag'

    }

}
filter{
        mutate{
                remove_field =>["createDt"]
                }
        #date{
         #     match => ["insert_ts","YYYYMMddHHmmss"]
         #     timezone => "Asia/Seoul"
```

```
}
filter{
        mutate{
                remove_field =>["createDt"]
                }
        #date{
         #     match => ["insert_ts","YYYYMMddHHmmss"]
         #     timezone => "Asia/Seoul"
        #
        ruby{
                code => "event.set('@timestamp', LogStash::Timestamp.new(event.get('@timestamp')+(9*60*60)))"
        }

        mutate{
                add_field =>["createDt","%{@timestamp}"]
        }

#       ruby{
#               code => "event.set('kr_time',event.get(@timestamp').time.localtime('+9:00').strftime(%Y%m%d'))"
#           }
}
output {
        #stdout {codec =>json}

        if [type] == 'feed' {
            elasticsearch {
                "hosts" => "localhost:9200"
                "index" => "feed"
                #index => "%{kr_tiime}"
                }
           }

        if [type] == 'user' {
           elasticsearch {
                "hosts" => "localhost:9200"
                "index" => "user"
                }
         }
#데이터가 삭제되었을 때 문제, 데이터 관리적인 측면에서의 문제,
        if [type] == 'hashTag' {
           elasticsearch {
                "hosts" => "localhost:9200"
                "index" => "hashtag"
                }
         }

    stdout { codec => rubydebug }
}
```

3. Elasticsearch mapping

I.user Index

```
{
    "settings":{
        //성능 테스트시 변경될 수 있음
        "number_of_shards":3,
        "number_of_replicas":1
    },
    "mappings":{
        "properties": {
            "id":{
                "type": "long"
            },
            "nickname": {
                "type": "text"
            },
            "profile_link":{
                "type": "text"
            },
            "create_dt": {
                "type": "date"
            },
            "interest":{
                "type": "text"
            }
        }

    }
}
```

II.post Index

```
{
    "settings":{
        //성능 테스트시 변경될 수 있음
        "number_of_shards":3,
        "number_of_replicas":1
    },
    "mappings":{
        "properties": {
            "postId":{
                "type":"integer"
            },
            "postType":{
                "type":"text"
            },
            "content":{
                "type":"text"
            },
            "images":{
                "type":"text"
            },
            "code":{
                "type":"text"
            },
            "codeType":{
                "type":"text"
            },
            "link":{
                "type":"text"
            },
            "createDt":{
                "type":"date",
                "store":true
            },
            "likeCnt":{
```

```
                "type":"integer"
            },
            "commentCnt":{
                "type":"integer"
            },
            "userImage":{
                "type":"text"
            },
            "nickname":{
                "type":"text"
            },
            "userId":{
                "type":"integer"
            },
            "hashTag":{
                "type":"text"
            }
        }
    }
}
```

III.hashtag Index

```
{
    "settings":{
        //성능 테스트시 변경될 수 있음
        "number_of_shards":3,
        "number_of_replicas":1
    },
    "mappings":{
        "properties": {
            "hashTag": {
                "fielddata":"true",
                "type":"keyword"
            }
        }
    }
}
```

4. 팔로우 추천 쿼리

I. 1단계
<알 수도 있는친구>

```
pageRank
CALL gds.graph.create(
'tGraph',
//만들 graph 이름

MATCH (n:User{userId:24})
MATCH (n)-[:FOLLOW*2]->(m)
WHERE NOT (n)<-[:FOLLOW]-(m)
RETURN m

'User',
'FOLLOW'
)

//nickname
MATCH (u1:User{userId:23})
```

```
CALL gds.pageRank.stream('tGraph', {
maxIterations: 20,
dampingFactor: 0.85,
//위 두 인자는 데이터를 다뤄보며 조절해야할 것
sourceNodes:[u1]
})
YIELD nodeId, score
WITH u1,gds.util.asNode(nodeId).nickname AS nick,
gds.util.asNode(nodeId).profileImg AS profileImg,
gds.util.asNode(nodeId).description AS description,
gds.util.asNode(nodeId).isFollowed AS isFollowed

RETURN nick,profileImg,description,isFollowed,interests
ORDER BY score DESC, nick ASC LIMIT 50;
```

<좋아할만한 친구>

```
MATCH (u1:User{userId:2})-[:INTEREST_IN]->(i:Interest)
WITH u1, collect(id(i)) AS u1interest
MATCH (u2:User)-[:INTEREST_IN]->(i:Interest) WHERE u1 <>
u2
WITH u1, u1interest, u2, collect(id(i)) AS u2interest
WHERE not (u1:User{userId:2})-[:FOLLOW]->(u2:User)
MATCH (u2)-[:INTEREST_IN]->(i:Interest)
WITH u1, u1interest, u2, u2interest, i.content AS
interest, u2.profileImg AS profileImg
RETURN u1.nickname AS me,
u2.nickname AS recommendedNick,
u2.isFollowed AS isFollowed
interest,
profileImg,
gds.alpha.similarity.jaccard(u1interest, u2interest) AS
similarity

ORDER BY recommendedNick, similarity DESC LIMIT 50;
```

II. 2단계

<알 수도 있는친구>

```
MATCH (u1:User{userId:213})-[:INTERESTED_IN]-
>(i:Interest)
WITH u1, collect(id(i)) AS u1interest


MATCH (u2:User)-[:INTERESTED_IN]->(i:Interest) WHERE u1
<> u2
WITH u1, u1interest, u2, collect(id(i)) AS u2interest
WHERE not (u1)-[:FOLLOW]->(u2)
WITH u1, u1interest, u2,
u2interest,gds.alpha.similarity.jaccard(u1interest,
u2interest) AS similarity, u2.lastPostDt AS lastPostDt,0
as score
WHERE similarity <>0

MATCH (User{userId:u2.userId})
WHERE
duration.inDays(lastPostDt,localdatetime.realtime()).days
<= 7
WITH score + 2 as score,u2,similarity
RETURN u2.userId AS ID,
       u2.nickname AS nick,
       u2.profileImg,
       u2.description,
       u2.interest AS interests,
       similarity+score AS score,
       False as isFollowed

ORDER BY similarity DESC LIMIT 50;
```

<좋아할만한 친구>

```
MATCH (u1:User{userId:29})
MATCH (u1)-[:FOLLOW*1..3]-(u2)
WHERE NOT (u1)-[:FOLLOW]-(u2)
WITH DISTINCT 5 as fScore, u2.nickname as nick,u1


CALL gds.pageRank.stream('G', {
maxIterations: 20,
dampingFactor: 0.85,
//위 두 인자는 데이터를 다뤄보며 조절해야할 것
sourceNodes:[u1]
})

YIELD nodeId, score
WHERE gds.util.asNode(nodeId).nickname = nick
WITH fScore + score * 50 as score,
gds.util.asNode(nodeId) as u2
WHERE u1.nickname<>u2.nickname and NOT (u1)-[:FOLLOW]-
(u2)
RETURN u2.nickname,score,u2.userId

ORDER BY score DESC LIMIT 50
```

5.소요시간이 상당히 많이 걸린 문제들에 대해 정리

① **.npm** 설치문제**error:**
https://stackoverflow.com/questions/52231289/gyp-err-stack-error-eacces-permission-denied-mkdir-var-www-project-name-no

소요시간:약 2시간

② . **logstash input, output** 설치관련 정리

- gem 에 추가하려는 plugin 명칭을 작성해야함

- gem install bundler
  rbenv rehash

-gem package 다루는 순서
gem install->gem file ->bundle update

소요시간:약 하루(10시간)

③. **logstash input, output** 적용 **issue**

1. logstash input, output plugin을 설치하기 위해 $logstash_home에 gem install함
(https://github.com/logstash-plugins/logstash-output-neo4j 참조)

2. 하지만 버전이 맞지않는다는 애러가 발생

3. 외부에서 플러그인 JAR 파일을 가져온 뒤 jruby로 설치 이후 gem install 실행

4. install 실패이후 logstash 삭제 후 재시도

5. 마찬가지로 실패, 이후 서치로 ruby package 설치하는 방법 서치

6. gem install bundler 를 $logstash_home에 해야함을 알게됨, 해당 명령어 실행
   rbenv rehash

7. 더하여 gem file을 먼저 install로 설치하고 gemfile에 plugin 할 것을 추가한다 이후 bundle install을 통해 패키지를 관리한다

8. 성공

소요시간:약 하루(8시간)

④.엘라스틱문제 관련 인프라팀 문의

안녕하세요 다름이 아니라,

소울메이츠 10.250.93.115(es서버) 3000번 포트에 문제가 있는거 같아 문의드립니다.

증상으로는, postman으로 10.250.93.115/3000/endpoint에 요청을 보낼 시 응답이 오지 않습니다.

(하지만 해당 서버의 9200번 포트는 정상적으로 요청,응답이 되고 있습니다.)

해당 문제는 2/1 오후 9시즈음 발견했고, 정상작동은 당일 오후 5시즈음까지 확인되었습니다.

해당 문제를 해결하기 위해 시도한 내용으로는,

1.해당 서버 sudo reboot 3회시도

2.해당 서버 로컬에서 curl http://localhost:3000/endpoint로 접근시 정상 응답 요청

3.해당 서버 내 node서버 삭제 후 재설치

4.node서버 포트 변경 3000->9200(정상적으로 외부에서 요청/응답 가능)

*인프라 답변: 이상없음
==>
 하지만, 이상이 분명히 있음을 감지하고 시간이 촉박하여 다른 방법을 간구함,
 다른 was에 애플리케이션 서버를 설치하여 해결함

소요시간:약 5시간

⑤**.rabbitmq** 문제

(여기서도 시간이 엄청많이 걸림, gem, ruby문제때문에)
1.logstash-plugin update를 통해 플러그인 다운로드

2.mongodb,, id값 문제 발생 (objectID)를 가져와야하는데 int를 줌..

3.rabbitmq 연결이 자꾸 에러가남

4.문제 해결을 위해 태완님과 함께 원인을 찾아보다가, vhost 문제가 있음을 확인

5.rabbitmq에 vhost를 추가하고, 유저를 추가해야했다

6.그다음 권한을 부여

7.해결

8.인줄 알았지만, topic방식으로 logstash에서 받아오게되면, elasticsearch의 타겟인덱스로 들어가는 것 뿐아니라
 다른 인덱스에도 들어가는 문제가 발생함

```
==>input {
   rabbitmq {
      host => "10.250.93.126:5672"
      user => "admin"
      password => "1234"
```

```
        queue => "postCreated"
        exchange => "feed.topic"
        exchange_type => "topic"
        key => "post.create"
        durable => true
        vhost => "/"

    }
....
```

9. 아래와 같이 type을 추가해 해결하려했지만 해결이 안됨

```
input {
    rabbitmq {
        host => "10.250.93.126:5672"
        user => "admin"
        password => "1234"
        queue => "postCreated"
        #exchange => "feed.topic"
        #exchange_type => "topic"
        #key => "post.create"
        durable => true
        vhost => "/"
        type => 'feed'

    }
..


        if [type] == 'feed' {
            elasticsearch {
                "hosts" => "localhost:9200"
                "index" => "feed"
                }
            }

        if [type] == 'user' {
            elasticsearch {
```

```
        "hosts" => "localhost:9200"
        "index" => "user"
        }
```

10.exchange, type, key를 삭제해서 direct로 받아오는 방법을 선택, 성공

```
input {
   rabbitmq {
      host => "10.250.93.126:5672"
      user => "admin"
      password => "1234"
      queue => "postCreated"
      #exchange => "feed.topic"
      #exchange_type => "topic"
      #key => "post.create"
      durable => true
      vhost => "/"
      type => 'feed'

  }
```

소요시간 약 2일

⑥**nginx**문제

1. conf.d 임포트 문제, 어디 경로를 임포트 해야할지 모름

2.임포트 경로를 찾은 뒤, conf파일 구조를 이해하여 nginx.conf/ valleyserver.conf 파일을 분리함

3.분리한 뒤, reverse proxy를 위해서 ip, api endpoint별로 나눔

4. 400문제 405문제가 발생=> buffer size, header 문제임을 알게되어 해결

소요시간 약 5시간

⑦**neo4j** 모듈 설치문제

1. neo4j설치후 gds 모듈, algorithm 모듈 설치 시도

2. algorithm 설치하려고 시도했으나, 설치하고 config 파일에 등록, 하지만 안됨

3. 알고보니 버전문제,, 현재 neo4j 4버전, 하지만 algorithm은 3버전까지 지원

4. 그래서 gds를 설치하려고 시도, 하지만 동일한 문제발생

5.알고보니 이것도 버전문제, 4.x대 내에서도 다른 버전으로 다운받아야함

6.해결

소요시간 약 1.5일