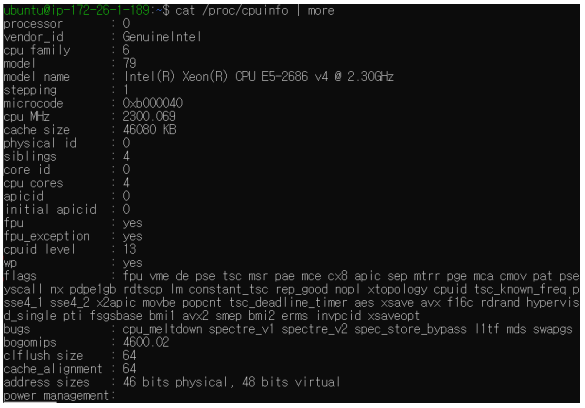
⚙️

포팅메뉴얼(메인서버)

🛠️개발환경

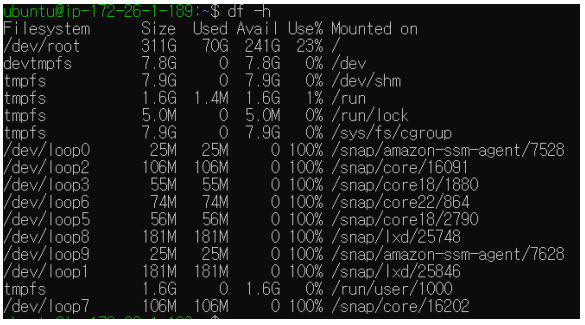
서버 인스턴스 사양

- CPU 정보: Intel Xeon(R) Core 4개



- RAM : 16GB

- Disk : 300GB



- 현재 메모리 사용량

형상 관리

- Gitlab
- Jira

UI / UX

- Figma

OS

- Ubuntu 20.04
- Windows 10

이슈 관리

- Jira

Communication

- Mattermost
- Notion

Front-end

- React : 16.13.0
- Axios : 1.5.0
- bootstrap : 5.3.2
- papago : 1.1.1
- react-kakao-maps-sdk : 1.1.21
- react-slick : 0.29.0
- reduxjs/toolkit : 1.9.5
- slick-carousel : 1.8.1
- react-wordcloud : 1.2.7
- sockjs : 0.3.24
- sockjs-client: 1.6.1
- stomp/stompjs : 7.0.0
- sweetalert2 : 11.7.31
- wowjs : 1.1.3
- testing-library/jest-dom : 5.17.0
- testing-library/react : 12.1.5
- testing-library/user-event : 13.5.0

DB

- MySQL : 8.0.33.0
- Redis : 7.0

Back-end

Spring Boot

- IntelliJ : 2023.1.3
- Java : 17.0.7
- Spring Boot : 3.0.10
- querydsl : 5.0.0 jakarta
- chatGPT : 1.0.4
- spring-boot-starter-data-elasticsearch
- elasticsearch-rest-high-level-client : 7.17.1
- lombok
- spring-boot-starter-data-jpa
- spring-boot-starter-web
- spring-boot-starter-security
- JWT : 0.11.5
- spring-boot-starter-websocket
- spring-boot-starter-mail : 2.7.0
- spring-cloud-gcp-starter : 1.2.5.RELEASE

IDE

- IntelliJ : 2022.3.1
- Visual Studio Code : 1.75.0

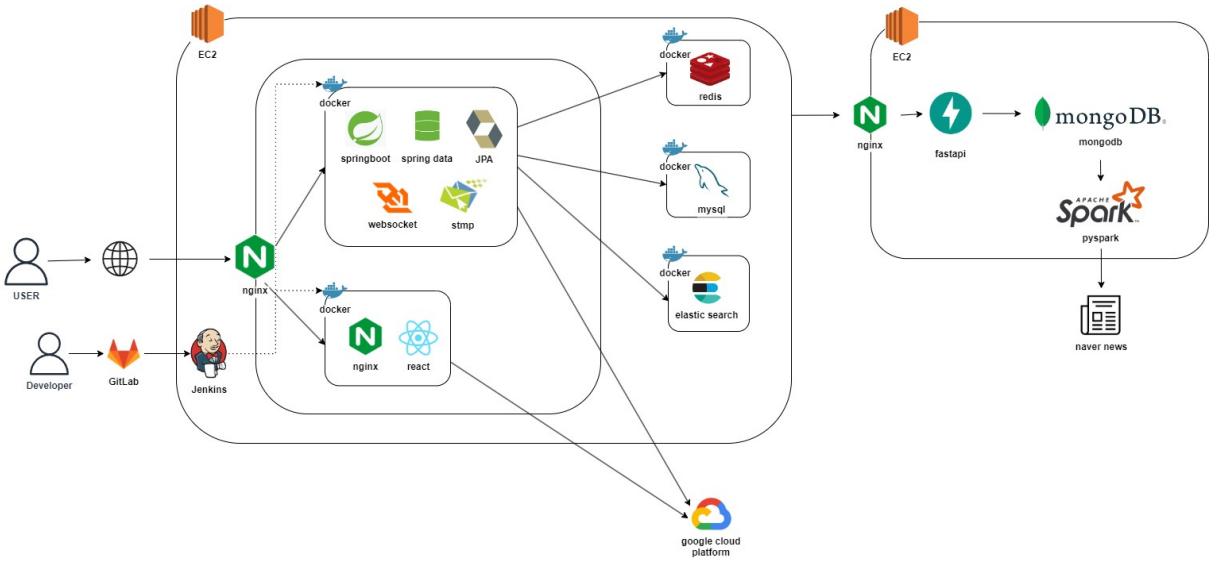
Infra

- Web Server : Nginx
- Jenkins : 2.416
- Docker : 24.0.5

기타 편의 툴

- WSL2
- Postman

시스템 아키텍처

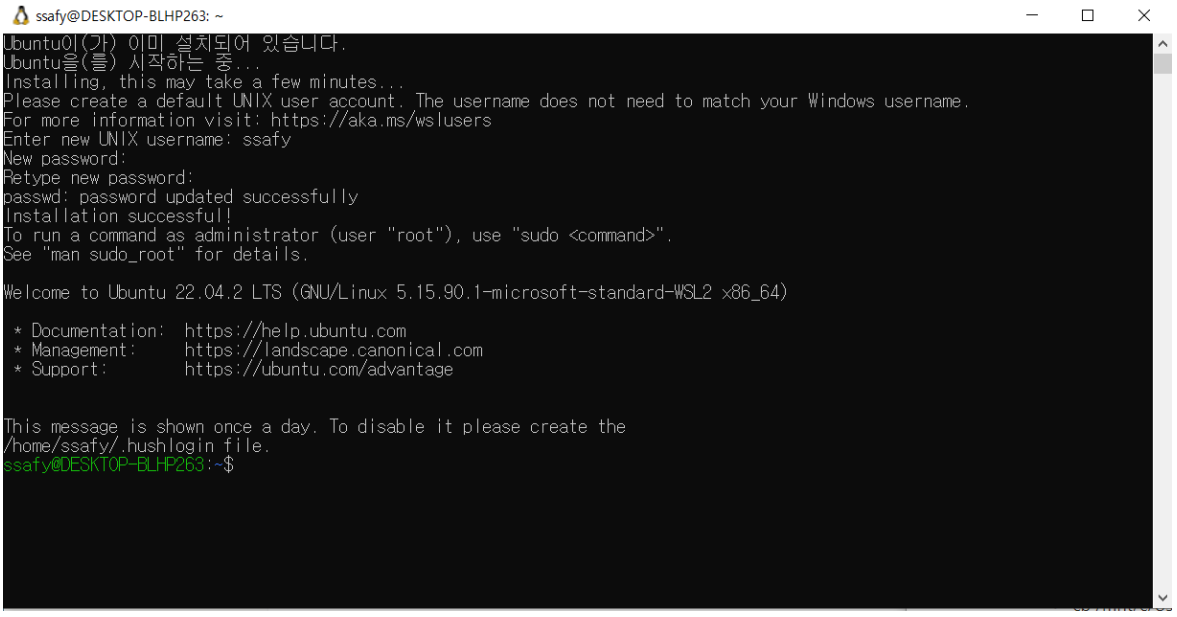


🔨배포 과정

📂빌드 환경설정

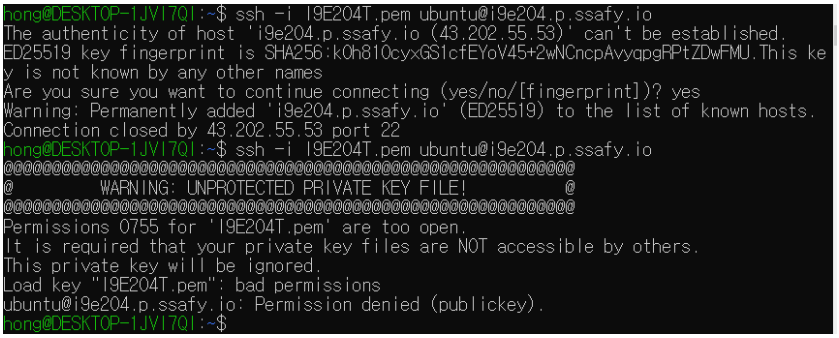
0. WSL2 설치

- powershell 관리자 모드로 실행
- WSL2 설치
  - $ wsl --install
  - $ wsl --set-default-version 2
  - $ dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
  - $ dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
  - 재부팅 → 우분투 설치 및 사용자 이름/비번 입력(둘 다 ssafy로 설정)



1. SSAFY EC2 접속(WSL 이용)

- 바탕화면에 있던 팸키를 우분투 폴더로 복사 붙여넣기
  - cp /mnt/c/Users/SSAFY/Desktop/J9E204T.pem ~/
- ssh 접속
  - cd ~
  - 키 권한 변경



    - 권한 에러
    - 개인키는 권한이 너무 open 되어있어도 경고 문구가 뜨면서 permission deny가 뜨기 때문에 권한을 축소시켜서 (chmod 400) 사용하도록 한다.
    - $ chmod 400 pem키
    - 그래도 안된다면???
      - https://velog.io/@wingnawing/작고-귀여운-AWS-.pem-권한-에러-WARNING-UNPROTECTED-PRIVATE-KEY-FILE
      - 우분투 환경으로 가서 진행해야함!!! cd ~로 가서 pem키 권한을 바꿔주자
  - ssh -i J9E204T.pem ubuntu@j9e204.p.ssafy.io
  - ssh 접속 쉽게 하는법(wsl ubuntu에서 진행)

```
mkdir ~/.ssh
cd ~/.ssh // ssh 폴더 생성 및 이동
cp [로컬 pem 키 위치] ~/.ssh // pem 키 옮기기
vi config  // config 파일 생성
```

  - config

```
Host ssafy
HostName j9e204.p.ssafy.io
User ubuntu
IdentityFile ~/.ssh/J9E204T.pem
```

  - 이후부터 ssh ssafy로 접속

- 방화벽 포트 확인 : sudo ufw status verbose

2. EC2 초기 설정

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt install build-essential

$ sudo ln -sf /usr/share/zoneinfo/Asia/Seoul /etc/localtime # 한국으로 시간 설정
```

자바 설치

```
$ sudo apt install openjdk-17-jdk
```

https://languagestory.tistory.com/154

(필수)Swap 설정

- https://sundries-in-myidea.tistory.com/102
- 이거 안해주면 나중에 메모리 초과돼서 서버 안들어가짐 ㅜㅜ

3. 서버에 도커 설치

```
$ sudo apt update
$ sudo apt install apt-transport-https ca-certificates curl software-properties-common
$ sudo wget -qO- https://get.docker.com/ | sh
```

```
$ sudo usermod -aG docker ${USER}
$ sudo systemctl restart docker
```

- sudo를 사용하지 않고 docker를 사용할 수 있다.
- docker 그룹은 root 권한과 동일하므로 꼭 필요한 계정만 포함
- 현재 계정에서 로그아웃한 뒤 다시 로그인

4. Mysql 설치

```
$ docker pull mysql:8.0.33
$ docker images
$ docker run --name mysql -e MYSQL_ROOT_PASSWORD=ssafy -d -p 3306:3306 mysql:8.0.33
$ docker ps -a # docker 컨테이너 리스트 출력
```

https://velog.io/@_nine/Docker-MySQL설치-및-접속하기

https://poiemaweb.com/docker-mysql

- mysql 비번 ssafy

```
$ docker exec -it mysql bash
mysql -u root -p
```

- mysql 도커 컨테이너 접속

- curl ifconfig.me ← 퍼블릭 ip주소 알아내기
  - MOA 서버 ip 주소 : 3.38.179.92
- 파일 전송 : `scp -i [pem file] [upload file] [user id]@[ec2 public IP]:~/[transfer address]`
  - Permission denied (public) 에러났을 때
    - https://john-analyst.medium.com/윈도우에서-우분투-서버로-파일-전송하기-57b7076adae9

- scp -i ./J9E204T.pem /mnt/c/Users/SSAFY/Desktop/plugin.tar.gz ubuntu@j9e204.p.ssafy.io:/home/ubuntu/jenkins_home

5. Redis 설치(일단 나중)

```
$ docker pull redis:7.0
# $ docker run -d -p 6379:6379 redis --requirepass "ssafy"
$ docker run --name redis -d -p 6379:6379 redis:7.0 --requirepass "ssafy"
# redis 컨테이너 접속
$ docker exec -it redis redis-cli -a ssafy
```

6. 젠킨스 설치

```
$ docker run -d --name jenkins --restart=on-failure \
-p 9090:8080 \
-v /var/jenkins_home:/var/jenkins_home \
-v /var/run/docker.sock:/var/run/docker.sock \
-e TZ=Asia/Seoul \
-u root \
jenkins/jenkins

# 초기 비밀번호
$ docker exec -u 0 -it jenkins /bin/bash
$ cat /var/jenkins_home/secrets/initialAdminPassword
```

- `-restart` -> on-failure 옵션은 비정상 종료시 컨테이너를 재실행합니다.
- `p` -> 외부 접속을 위해 호스트의 8080 포트를 바인딩 해주었습니다.
- `v` -> 호스트의 var/jenkins 디렉토리를 호스트 볼륨으로 설정하여 jenkins 컨테이너의 home 디렉토리에 마운트시켰습니다.

docker.sock 파일은 도커 데몬과 통신할 수 있는 소켓 파일입니다. docker.sock 파일을 컨테이너에 마운트시켜서 도커 명령을 실행할 수 있게 해줍니다. 이러한 방식을 dood(docker out of docker)라고 합니다.

- `e` -> 젠킨스의 timezone을 KST 기준으로 설정해줍니다.
- `u` -> 추후 권한 문제가 발생할 수 있기 때문에 user 옵션을 root 사용자로 주었습니다.

6-1. 젠킨스 & 깃랩 연동

- credential 생성
  - https://aamoos.tistory.com/365
- 젠킨스에 깃랩 연동
  - https://velog.io/@msung99/CICD-Jenkins-Docker-를-활용한-SpringBoot-배포-자동화-구축#:~:text=CI 구축 %3A GitHub 와 Jenkins 연동
  - https://aamoos.tistory.com/365
  - https://phillip5094.tistory.com/130 → 특정 branch는 여기 참고
- 깃랩에 젠킨스 연동
  - https://velog.io/@suhongkim98/jenkins-gitlab-연동-및-webhook-설정하기#:~:text=간에 연동이 완료되었습니다.-,4. webhook 설정,-파이프라인 프로젝트 생성

6-2. 젠킨스 도커 안에 도커 설치

```
$ apt update
$ apt install apt-transport-https ca-certificates curl software-properties-common
$ sudo wget -qO- https://get.docker.com/ | sh
```

- bash: wget: command not found

- apt-get install wget

6-3. 젠킨스 도커안에 npm 설치

```
$ apt-get install npm
```

- sudo를 사용하지 않고 docker를 사용할 수 있다.
- docker 그룹은 root 권한과 동일하므로 꼭 필요한 계정만 포함
- 현재 계정에서 로그아웃한 뒤 다시 로그인

참고자료

https://seosh817.tistory.com/287

https://boying-blog.tistory.com/18

- 퍼블릭 [ip주소]:9090으로 접속
- 위에서 얻은 비밀번호 입력해서 로그인

docker restarting이 계속 될 때

https://veneziar.tistory.com/52

6. Nginx 설치

```
$ sudo apt-get update
$ sudo apt-get install nginx
$ sudo apt-get install vim
```

https://everydayyy.tistory.com/127

- 처음 설치하면 시작이 안되있으므로 시작

  https://velog.io/@jkijki12/배포-Aws-인스턴스에-Nginx-적용하기

  ```
  $ sudo service nginx start
  $ sudo service nginx status
  ```

7. elastic search 설치

- docker-compose 설치

  ```
  // 최신 docker compose를 해당 링크에서 받을 수 있음
  sudo curl -L https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/d
  // 권한 부여
  sudo chmod +x /usr/local/bin/docker-compose
  // 설치 확인
  docker-compose version
  ```

- elastic search와 kibana 설치
  - https://backtony.github.io/spring/elk/2022-03-02-spring-elasticsearch-2/
  - 도커파일 생성(nori 형태소 추가하기 위해)

```
# dockerfile 생성
vi Dockerfile

# 작성
ARG ELK_VERSION
FROM docker.elastic.co/elasticsearch/elasticsearch:${ELK_VERSION}
RUN elasticsearch-plugin install analysis-nori
```

- compose 파일(`es.yml`) → 버전은 springboot 3.0.12버전에 맞는 8.5.3 사용

```
version: "3.7"
services:
  es:
    build:
      # 도커파일의 위치 알려주기
      context: /home/ubuntu
      # 인자 넣어주기
      args:
        ELK_VERSION: 8.5.3
    container_name: es
    environment:
      - node.name=single-node
      - cluster.name=backtony
      - discovery.type=single-node
    ports:
      - 9200:9200
      - 9300:9300
    networks:
      - es-bridge

  kibana:
    container_name: kibana
    image: docker.elastic.co/kibana/kibana:8.5.3
    environment:
      SERVER_NAME: kibana
      ELASTICSEARCH_HOSTS: http://es:9200
    ports:
      - 5601:5601
    depends_on:
      - es
    networks:
      - es-bridge

networks:
  es-bridge:
    driver: bridge
```

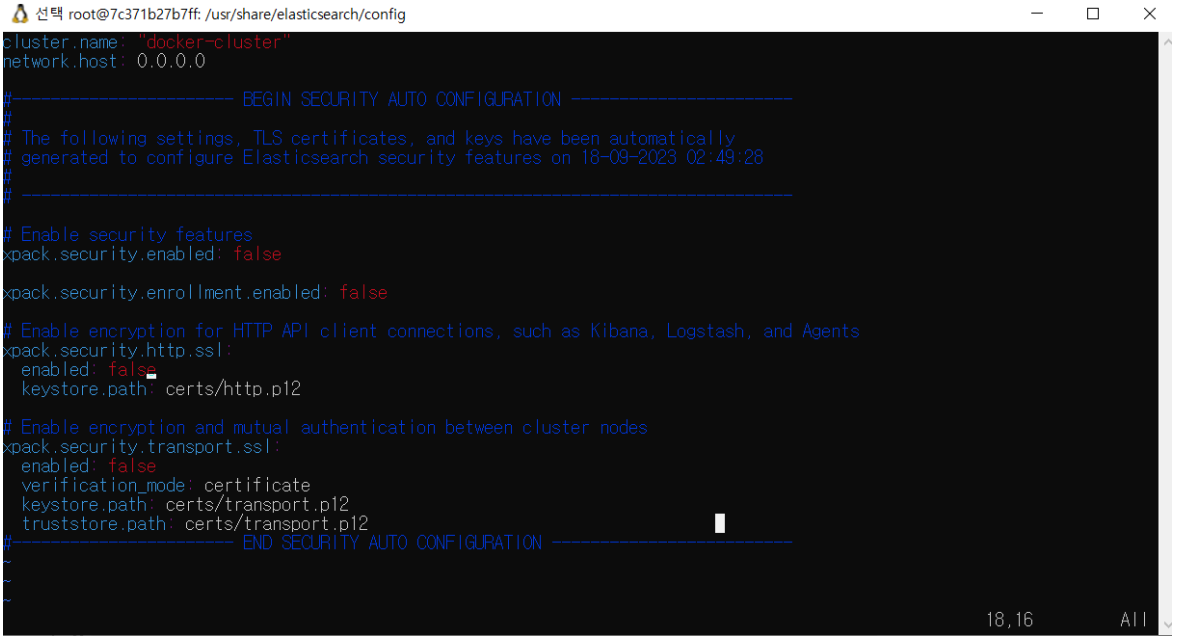
- docker compose 실행

```
# 실행, 데몬으로 띄우려면 맨 뒤에 -d를 붙여준다.
# 기본 실행 도커파일은 docker-compose.yml인데 es.yml로 만들었으므로 지정해주기 위해서 -f 옵션을 사용
docker-compose -f es.yml up -d

# 죽이기
docker-compose -f es.yml down
```

- elastic search의 ssl 설정 해제

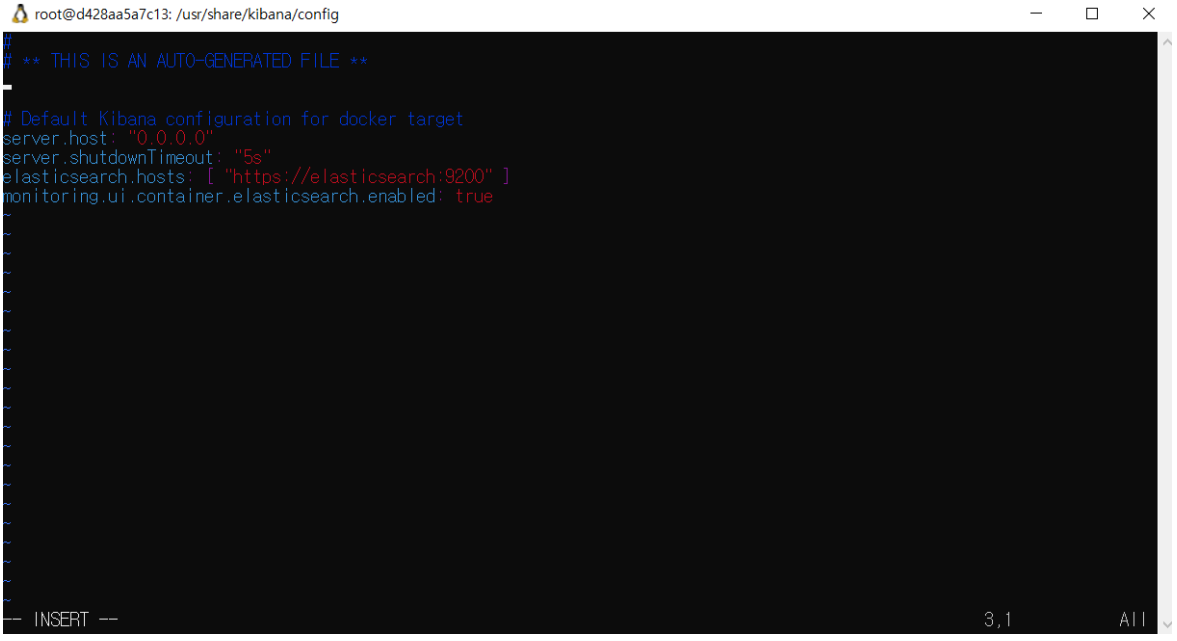
  https://discuss.elastic.co/t/kibana-error-unable-to-retrieve-version-information-from-elasticsearch-nodes-socket-hang-up/318421/2

```
docker exec -u 0 -it es /bin/bash

# vi 다운로드
apt-get update
apt-get upgrade
apt-get install vim

cd config/
vi elasticsearch.yml
```



- kibana의 elastic search 주소를 https로 변경
  - https://discuss.elastic.co/t/unable-to-retrieve-version-information-from-elasticsearch-node/288717
  - 위에서 elastic search의 ssl 설정 해제해서 필요없을 것 같기도??

```
docker exec -u 0 -it kibana /bin/bash

# vi 다운로드
apt-get update
apt-get upgrade
apt-get install vim

cd config/
vi kibana.yml
```



- http://3.38.179.92:5601/ 으로 kibana 접속 확인
  - **Kibana server is not ready yet** 라는 에러가 뜨면 설정이 뭔가 잘못된 것임
    - docker logs <kibana CONTAINER ID>

- 위의 명령어로 뭐가 잘못됐는지 확인하자

8. elastic search 데이터 색인

▼ 검색 index

```
# Click the Variables button, above, to create your own variables.
GET ${exampleVariable1} // _search
{
  "query": {
    "${exampleVariable2}": {} // match_all
  }
}

GET media-info-temp/_search

DELETE media-info-example

PUT media-info-example
{
  "settings": {
    "analysis" : {
      "analyzer": {
        "nori": {
          "type": "custom",
          "tokenizer": "nori_mixed",
          "filter": ["lowercase", "stop"]
        }
      },
      "tokenizer": {
        "nori_mixed": {
          "type": "nori_tokenizer",
          "decompound_mode": "mixed"
        }
      }
    }
  },
  "mappings": {
    "properties": {
          "placeNm": { "type": "text" },
          "placeTy": { "type": "text" },
          "operTime": { "type": "text" },
          "restTime": { "type": "text" },
          "rstdeQuidCn": { "type": "text" },
          "addr": { "type": "text" },
          "latitude": { "type": "float" },
          "longitude": { "type": "float" },
          "telNo": { "type": "long", "null_value": 0  },
          "location": { "type": "geo_point" },
          "seqNo" : {"type": "integer"},
          "mediaTy": { "type": "keyword" },
          "titleNm": { "type": "text", "analyzer" : "nori"},
          "relatePlaceDc": { "type": "text", "analyzer" : "nori" },
          "lastUpdtDe": { "type": "integer" }
    }
  }
}

PUT _ingest/pipeline/media-info-example-pipe
{
  "processors": [
    { "set": { "field": "_id", "value": "{{SEQ_NO}}" } },
    {"rename": { "field": "PLACE_NM", "target_field": "placeNm" } },
    {"rename": { "field": "PLACE_TY", "target_field": "placeTy" } },
    {"rename": { "field": "OPER_TIME", "target_field": "operTime" } },
    {"rename": { "field": "REST_TIME", "target_field": "restTime" } },
    {"rename": { "field": "RSTDE_GUID_CN", "target_field": "rstdeQuidCn" } },
    {"rename": { "field": "ADDR", "target_field": "addr" } },
    {"rename": { "field": "LC_LA", "target_field": "latitude" } },
    {"rename": { "field": "LC_LO", "target_field": "longitude" } },
    {
      "script": {
        "if": "ctx.containsKey('TEL_NO')",
        "source": """
          ctx['telNo'] = ctx['TEL_NO'];
        """
      }
    },
    {"rename": { "field": "SEQ_NO", "target_field": "seqNo" } },
    {"rename": { "field": "MEDIA_TY", "target_field": "mediaTy" } },
```

```
    {"rename": { "field": "TITLE_NM", "target_field": "titleNm" } },
    {"rename": { "field": "RELATE_PLACE_DC", "target_field": "relatePlaceDc" } },
    {"rename": { "field": "LAST_UPDT_DE", "target_field": "lastUpdtDe" } }
  ]
}

POST _reindex
{
  "source": {
    "index": "media-info-temp"
  },
  "dest": {
    "index": "media-info-example",
    "pipeline": "media-info-example-pipe"
  }
}

GET media-info-example/_search
{
  "size": 0,
  "aggs": {
    "mediaTy_terms": {
      "terms": {
        "field": "mediaTy",
        "size": 10
      }
    }
  }
}
```

▼ 자동완성 index

```
# ---------- 자동완성 ----------
GET media-info-auto-complete2/_search
DELETE media-info-auto-complete

PUT media-info-auto-complete
{
  "settings" : {
    "index":{
      "number_of_replicas": "0",
      "max_ngram_diff": 50,
      "analysis":{
        "filter": {
          "suggest_filter": {
            "type": "ngram",
            "min_gram": 1,
            "max_gram": 50
          }
        },
        "analyzer":{
          "my_ngram_analyzer": {
            "tokenizer": "my_ngram_tokenizer",
            "filter": ["lowercase", "stop"]
          },
          "nori": {
          "type": "custom",
          "tokenizer": "nori_mixed",
          "filter": ["lowercase", "stop"]
        }
        },
        "tokenizer": {
          "my_ngram_tokenizer": {
            "type": "ngram",
            "min_gram": "1",
            "max_gram": "10"
          },
          "nori_mixed": {
          "type": "nori_tokenizer",
          "decompound_mode": "mixed"
        }
        }
      }
    }
  },
  "mappings": {
    "properties": {
          "titleNm": {
            "type": "text",
            "fields": {
              "ngram": {
```

```
                "type": "text",
                "analyzer": "my_ngram_analyzer",
                "search_analyzer": "nori"
              },
              "nori": {
                "type": "text",
                "analyzer": "nori"
              }
            }
          }
    }
  }
}

PUT _ingest/pipeline/media-info-auto-complete-pipe
{
  "processors": [
    {"rename": { "field": "column2", "target_field": "titleNm" } }
  ]
}


POST _reindex
{
  "source": {
    "index": "media-type-list"
  },
  "dest": {
    "index": "media-info-auto-complete",
    "pipeline": "media-info-auto-complete-pipe"
  }
}

PUT _ingest/pipeline/media-info-auto-complete-pipe
{
  "processors": [
    {"remove": {
      "field": "REST_TIME"
    }},
    {"remove": {
      "field": "LC_LO"
    }},
    {"remove": {
      "field": "OPER_TIME"
    }},
    {"remove": {
      "field": "PLACE_TY"
    }},
    {"remove": {
      "field": "ADDR"
    }},
    {"remove": {
      "field": "LAST_UPDT_DE"
    }},
  {"remove": {
      "field": "LC_LA"
    }},
  {"remove": {
      "field": "MEDIA_TY"
    }},
  {"remove": {
      "field": "RSTDE_GUID_CN"
    }}
  ]
}

DELETE media-info-auto-complete

GET media-info-auto-complete/_search

POST _reindex
{
  "source": {
    "index": "media-info-auto-complete"
  },
  "dest": {
    "index": "media-info-auto-complete2",
    "pipeline": "media-info-auto-complete-pipe"
  }
}

GET media-info-auto-complete/_search
```

```
GET media-info-auto-complete/_search
{
  "query": {
    "match": {
      "titleNm.ngram": {
        "query": "방",
        "analyzer": "my_ngram_analyzer"
      }
    }
  }
}

GET media-info-auto-complete/_search
{
  "query": {
    "match": {
      "titleNm.ngram": "방탄소년"
    }
  },
  "sort":
    {
      "_score": "desc"
    }

}

GET media-info-auto-complete/_search
{
  "query": {
    "bool": {
      "should": [
        {
          "match": {
            "titleNm.ngram": "nc"
          }
        },
        {
          "match": {
            "titleNm.nori": "nc"
          }
        }
      ]
    }
  }
}

GET media-info-auto-complete/_search
{
   "query": {
    "bool": {
      "must": [
        {
          "match": {
            "titleNm.ngram": "nct"
          }
        }
      ],
      "should": [
        {
          "match": {
            "titleNm.nori": "nct"
          }
        }
      ]
    }
  }
  ,
  "sort":
    {
      "_score": "desc"
    }
}



GET media-info-auto-complete/_search
{
  "query": {
    "bool": {
      "should": [
        {
          "match": {
      "titleNm.ngram": {
        "query": "백",
```

```
        "analyzer": "my_ngram_analyzer"
      }
    }
      },
      {
        "match": {
      "relatePlaceDc.ngram": {
        "query": "백",
        "analyzer": "my_ngram_analyzer"
      }
    }
      }
    ]
  }
 }
}
```

📂빌드 및 배포

도메인 구매

웹을 넘어 클라우드로. 가비아

그룹웨어부터 멀티클라우드까지 하나의 클라우드 허브

https://www.gabia.com/

- 원하는 도메인 구매 및 결제
- DNS 설정

  https://velog.io/@woals4815/도메인-적용하기-가비아에서-구입하는-방법

  - My 가비아 → DNS 관리툴 → DNS 관리에서 호스트 / 값 설정
  - 호스트에는 `@`, 값/위치에는 `domain 주소`를 작성합니다.

  ⇒ 30분??정도 기다려야 제대로 적용됨

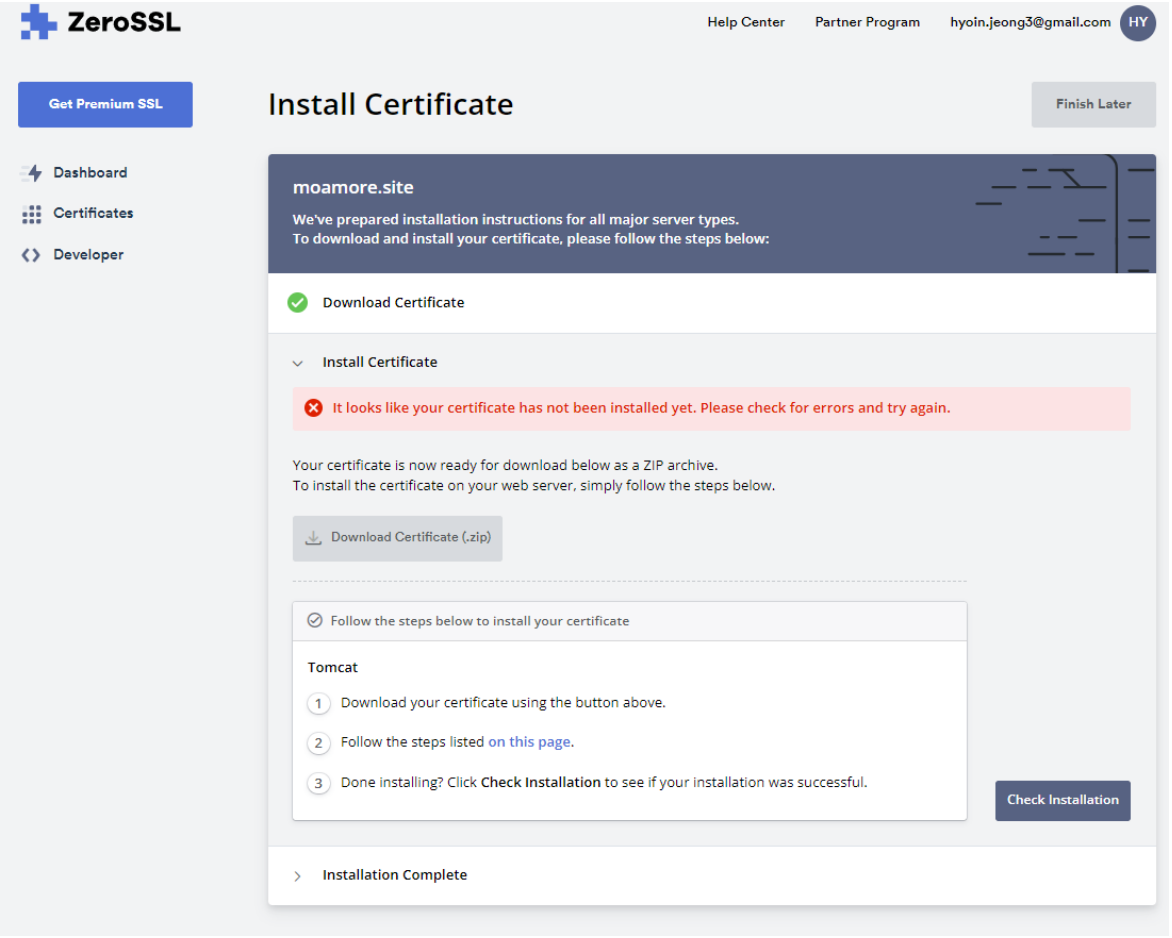
SSL 인증서 다운

SSL For Free - Free SSL Certificates in Minutes

Free SSL certificates issued in less than a minute, for one or multiple domains, supporting wildcards and ACME with tutorials.

https://www.sslforfree.com/

- 도메인 입력 후 로그인
- 그다음 단계는 블로그 참고 → https://foxydog.tistory.com/39
  - step4는 “두 번째. [DNS (CNAME)] 인증”으로 진행
  - 난 아래와같은 에러가 뜨는데,,, 그전에 인증키들 받을 수 있어서 괜찮은듯??



⇒ private.key, certificate.crt, ca_bundle.crt 를 얻을 수 있음!

SSL AWS에 적용

https://cl8d.tistory.com/97

- 에서 다운받은 certificate.crt를 pem키로 변환(wsl ubuntu에서 진행)

```
cd /mnt/c/Users/SSAFY/Desktop/ # 바탕화면으로 이동
openssl x509 -inform PEM -in certificate.crt -out certificate.pem
```

- 바탕화면의 인증키들(certificate.pem, private.key)을 ec2로 옮기기

```
cd ~ # 무조건 여기에 있는 pem키를 사용해야함!!!
scp -i ./J9E204T.pem /mnt/c/Users/SSAFY/Desktop/certificate.pem ubuntu@j9e204.p.ssafy.io:~
scp -i ./J9E204T.pem /mnt/c/Users/SSAFY/Desktop/private.key ubuntu@j9e204.p.ssafy.io:~
```

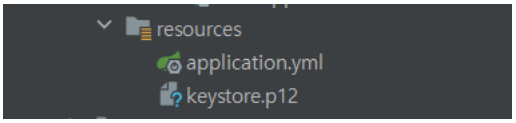
SSL SpringBoot에 적용

https://shanepark.tistory.com/442

- 에서 다운받은 private.key, certificate.crt, ca_bundle.crt(위치-바탕화면)들을 keystore.p12로 변환(wsl ubuntu에서 진행)

```
cd /mnt/c/Users/SSAFY/Desktop/ # 바탕화면으로 이동
openssl pkcs12 -export -out keystore.p12 -inkey private.key -in certificate.crt -certfile ca_bundle.crt
```

- springboot의 resource 폴더에 keystore.p12 붙여넣기



- application.yml파일에 아래 코드 추가

```
server:
  ssl:
    key-store: classpath:keystore.p12
    key-store-password: ssafy(위에서 설정한 비밀번호)
    key-store-type: PKCS12
```

Nginx

- ssl 인증키를 보관할 폴더 생성

```
cd /etc/nginx/
sudo mkdir ssl
```

- 에서 가져온 인증키들의 위치 이동

```
mv ~/certificate.pem /etc/nginx/ssl/
mv ~/private.key /etc/nginx/ssl/
```

- nginx 설정 파일 이동

```
cd /etc/nginx/sites-enabled/
sudo vi default
```

- proxy 설정 수정

```
# /etc/nginx/sites-enabled/default 파일
server {
        listen 80;
        server_name moamore.site;
        return 301 https://moamore.site$request_uri; # https로 redirect하는 것
}

server {
        listen 443 ssl;
        ssl on;
        server_name moamore.site;

        ssl_certificate /etc/nginx/ssl/certificate.pem;
        ssl_certificate_key /etc/nginx/ssl/private.key;

        location / {
                proxy_pass http://localhost:3000;
proxy_http_version 1.1;
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection "upgrade";
      proxy_set_header Host $host;
        }

        location /api {
                proxy_pass https://localhost:8589;
proxy_http_version 1.1;
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection "upgrade";
      proxy_set_header Host $host;
        }

}
```

- nginx 재가동(필수!!)

```
sudo service nginx restart
```

⇒ https://moamore.site를 입력하면 aws의 3000포트를 가리켜서 프론트가 보임

⇒http://moamore.site를 입력해도 https://moamore.site로 이동됨

Front End

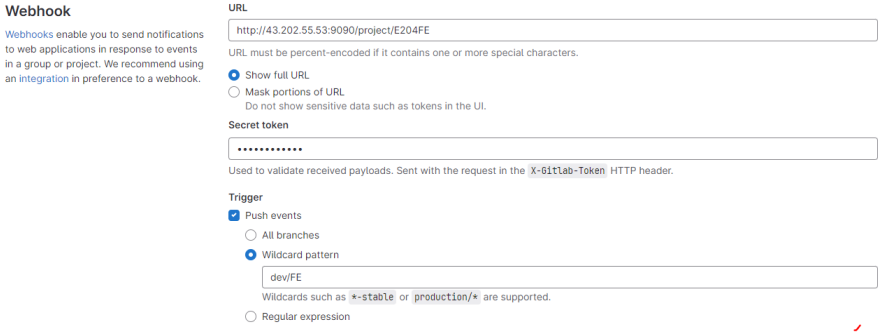
```
// nginx.conf
server {
    listen 80;

    location / {
        root    /app/build;
        index   index.html;
        try_files $uri $uri/ /index.html;
    }
}
```

```
// Dockerfile

# nginx 이미지를 사용합니다. 뒤에 tag가 없으면 latest 를 사용.
FROM nginx

# root 에 app 폴더를 생성
RUN mkdir /app

# work dir 고정
WORKDIR /app

# work dir 에 build 폴더 생성 /app/build
RUN mkdir ./build

# host pc의 현재경로의 build 폴더를 workdir 의 build 폴더로 복사
ADD ./build ./build

# nginx 의 default.conf 를 삭제
RUN rm /etc/nginx/conf.d/default.conf

# host pc 의 nginx.conf 를 아래 경로에 복사
COPY ./nginx.conf /etc/nginx/conf.d

# host pc 의 env를 아래 경로에 복사
COPY ./ /home/ubuntu/env/front/.env

# 3000 포트 오픈
EXPOSE 3000

# container 실행 시 자동으로 실행할 command. nginx 시작함
CMD ["nginx", "-g", "daemon off;"]
```

Jenkins Execute shell

```
cd /var/jenkins_home/workspace/E204FE/Project/Front-end

npm install

CI=false npm run build

docker ps -f name=frontend -q | xargs --no-run-if-empty docker container stop
docker container ls -a -f name=frontend -q | xargs -r docker container rm
docker build -t frontend .

docker run -v /home/ubuntu/env/front/.env:/app/.env -it -d --name frontend -p 3000:80 frontend

echo "y" | docker image prune
```

GitLab Webhook



Back End

```
// Dockerfile
FROM openjdk:17

EXPOSE 8080

ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

```
// application.yml
spring:
  datasource:
    url: jdbc:mariadb://43.202.55.53:3306/drug
    driver-class-name: org.mariadb.jdbc.Driver
    username: ssafyE204
    password: 1793
#  datasource:
#    url: jdbc:mysql://localhost:3307/drug
#    driver-class-name: com.mysql.cj.jdbc.Driver
#    username: root
#    password: ssafy
  jpa:
    open-in-view: false
    generate-ddl: true
    show-sql: true
    hibernate:
      ddl-auto: update

#springdoc:
#  swagger-ui:
#    path: /swagger-ui.html
#    groups-order: DESC
#    operationsSorter: method
#    disable-swagger-default-url: true
#    display-request-duration: true
#  api-docs:
#    path: /api-docs
#  show-actuator: true
#  default-consumes-media-type: application/json
#  default-produces-media-type: application/json
#  paths-to-match:
#    - /v1/**

springdoc:
  packages-to-scan: com
  default-consumes-media-type: application/json;charset=UTF-8
  default-produces-media-type: application/json;charset=UTF-8
  swagger-ui:
    path: /swagger-ui.html
    disable-swagger-default-url: true
    display-request-duration: true
    operations-sorter: alpha

server:
```

```
  ssl:
    key-store: classpath:keystore.p12
    key-store-password: 1793
    key-store-type: PKCS12
```

Jenkins Execute shell

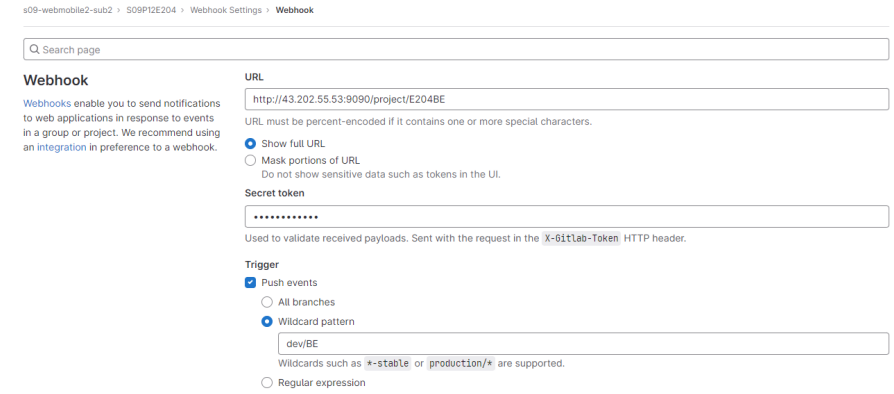
```
cd /var/jenkins_home/workspace/E204BE/Project/Back-end

chmod +x ./gradlew
./gradlew clean build -x test
docker ps -f name=backend -q | xargs --no-run-if-empty docker container stop
docker container ls -a -f name=backend -q | xargs -r docker container rm
docker build -t backend .
docker run -d -it --rm -p 8589:8080 --name=backend backend -h bserver -e TZ=Asia/Seoul
docker rmi -f $(docker images -f "dangling=true" -q) || true
```

[에러]

```
> Could not resolve all files for configuration ':classpath'.
   > Could not resolve org.springframework.boot:spring-boot-gradle-plugin:3.0.0.
     Required by:
         project : > org.springframework.boot:org.springframework.boot.gradle.plugin:3.0.0
```

- ./gradlew clean build -x test 에서 could not resolve org.springframework.boot:spring-boot-gradle-plugin:3.1.3. 에러가 날 때
  - 도커안의 젠킨스의 자바버전이 springboot의 자바버전과 안맞기 때문

    https://sh970901.tistory.com/70




```
ubuntu@ip-172-26-1-169:~$ docker ps -a
CONTAINER ID   IMAGE                                  COMMAND                  CREATED        STATUS                      PORTS                                                                                  NAMES
8f4ef50c4be9   backend                                "java -jar /app.jar …"   7 hours ago    Exited (255) 59 minutes ago   0.0.0.0:8589->8080/tcp, :::8589->8080/tcp                                              backend
6b72c40b8894   frontend                               "/docker-entrypoint.…"   18 hours ago   Exited (255) 59 minutes ago   3000/tcp, 0.0.0.0:3000->80/tcp, :::3000->80/tcp                                        frontend
d428aa5a7c13   docker.elastic.co/kibana/kibana:8.5.3  "/bin/tini -- /usr/l…"   9 days ago     Exited (255) 59 minutes ago   0.0.0.0:5601->5601/tcp, :::5601->5601/tcp                                              kibana
7c371b27b7ff   ubuntu-es                              "/bin/tini -- /usr/l…"   9 days ago     Exited (255) 59 minutes ago   0.0.0.0:9200->9200/tcp, :::9200->9200/tcp, 0.0.0.0:9300->9300/tcp, :::9300->9300/tcp   es
756972395f4e   redis:7.0                              "docker-entrypoint.s…"   2 weeks ago    Exited (255) 59 minutes ago   0.0.0.0:6379->6379/tcp, :::6379->6379/tcp                                              redis
be267af58db4   mysql:8.0.33                           "docker-entrypoint.s…"   2 weeks ago    Exited (255) 59 minutes ago   0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp                                   mysql
020b0006226e   jenkins/jenkins                        "/usr/bin/tini -- /u…"   2 weeks ago    Up 59 minutes               50000/tcp, 0.0.0.0:9090->8080/tcp, :::9090->8080/tcp                                   jenkins
```

UFW(방화벽) 포트 허용 설정

- **ufw allow [port] [protocol]**
  - ufw allow 9090
  - ufw allow 8589
  - ufw allow 3000