



# 포팅 매뉴얼

## 기술 스택

[이슈 관리](#)  
[형상 관리](#)  
[커뮤니케이션](#)  
[개발 환경](#)  
[상세 사용](#)

## AWS EC2 ubuntu 서버 세팅

[Docker](#)  
[NGINX](#)  
[Certbot\(https 설정\)](#)  
[Jenkins](#)  
[Jenkins 설치](#)  
[Jenkins 설정](#)

## MySQL

[프로퍼티 정의](#)  
[프론트엔드 소개 페이지 프로젝트 설정 파일](#)  
[백엔드 프로젝트 설정 파일](#)

## Frontend

[개발자 모드로 확장 프로그램 실행](#)

## DATA

[\[1\] EC2 python 환경 설정](#)  
[python3용 pip 설치](#)  
[java & konlpy & 주요 라이브러리 설치](#)  
[기타 라이브러리 설치](#)  
[\[2\] Airflow 설정](#)  
[Airflow 설치](#)  
[Airflow 세팅](#)  
[Airflow 실행 \(백그라운드 daemon 실행\)](#)

## 기술 스택

### 이슈 관리

Jira

### 형상 관리

Gitlab

### 커뮤니케이션

Mattermost, Notion, Figma

### 개발 환경

- OS
  - Windows 10
- Web server
  - NGINX 1.18.0
- WAS
  - Apache Tomcat 9.0.65
- IDE
  - IntelliJ IDEA 2022.1.4
  - Visual Studio Code 1.70.0
- DB
  - MySQL 8.0.31
- 배포
  - AWS EC2 Ubuntu 20.04 LTS
  - Docker 20.10.20
  - Jenkins 2.374

### 상세 사용

- Backend
  - Java 17.0.4.1
  - Spring Boot 2.7.4

- Spring Data JPA 2.7.4
- Gradle 7.5.1
- Lombok 1.18.24

## 2. Frontend

- React 18.0.0
- TypeScript 4.2.4
- React Redux 8.0.4
- Axios 1.1.3
- Tailwind CSS 3.2.0
- Chart.js 3.9.1

## 3. Data

- ETL (데이터 파이프라인)
  - Python 3.8.10
  - apache-airflow 2.1.2
- 자연어 처리
  - JPype1-py3 0.5.5.4
  - konlpy 0.6.0
  - kss 3.6.4
  - newspaper3k 0.2.8
- 추천 시스템
  - scikit-learn 1.1.3
  - scikit-surprise 1.1.3
- 워드 클라우드
  - wordcloud 1.8.2.2
- MySQL 연결
  - mysql-connector-python 8.0.31
  - SQLAlchemy 1.3.24
  - PyMySQL 1.0.2

# AWS EC2 ubuntu 서버 세팅

- root 계정 로그인

```
sudo su
```

## Docker

- 사전 패키지 다운로드

```
sudo apt-get update
sudo apt-get install \
  ca-certificates \
  curl \
  gnupg \
  lsb-release
```

- gpg 키 다운로드

리눅스 패키지 툴이 프로그램 패키지가 유효한지 확인하기 위해, 프로그램 설치 전에 gpg 키로 검증하는 과정을 거치기 때문에 gpg 키를 받아야 함

```
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

- 도커 설치

```
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

- Storage Driver 변경

```
docker info | grep "Storage Driver"
Storage Driver: aufs
```

위 명령어 입력 시 Storage driver가 aufs가 아니면 변경 필요

```
systemctl stop docker
cp -au /var/lib/docker /var/lib/docker.bk
rm -rf /var/lib/docker
vim /etc/docker/daemon.json
```

```
{
  "storage-driver": "aufs"
}
```

```
systemctl start docker
docker info | grep "Storage Driver"
```

## NGINX

- 설치

```
sudo apt update
sudo apt install nginx
```

- NGINX 상태확인/실행/중지

```
// nginx 상태확인
service nginx status

// nginx 실행
service nginx start

// nginx 재실행
service nginx restart

// nginx 중지
service nginx stop
```

## Certbot(https 설정)

- 사전 준비

https가 적용될 서버 블록이 필요하기 때문에 우선 NGINX를 이용해서 서버 블록을 지정

```
// 도메인 이름으로는 k7d102.p.ssafy.io와 www.easssue.com을 사용
mkdir -p /var/www/{도메인이름}/html
chown -R $USER:$USER /var/www/{도메인이름}/html
chmod -R 755 /var/www/{도메인이름}
```

/etc/nginx/sites-available 에 k7d201.p.ssafy.io.conf 를 만들고 아래 명령어로 /etc/nginx/sites-enabled 에도 심볼릭 링크로 연결

```
ln -s /etc/nginx/sites-available/j7d108.p.ssafy.io.conf /etc/nginx/sites-enabled/
```

- /etc/nginx/sites-available/k7d201.p.ssafy.io.conf

```
upstream backend {
    ip_hash;
    server 172.26.8.127:8080;
}

upstream frontend {
    ip_hash;
    server 172.26.8.127:3000;
}

server {
    listen 80 default_server;
    listen [::]:80 default_server;

    server_name k7d102.p.ssafy.io www.easssue.com;

    if ($host = k7d102.p.ssafy.io) {
        return 308 https://$host$request_uri;
    }

    if ($host = www.easssue.com){
        return 308 https://$host$request_uri;
    }
}

server {
    listen 443 ssl;
    server_name www.easssue.com;

    ssl_certificate /etc/letsencrypt/live/www.easssue.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/www.easssue.com/privkey.pem;
```

```

        location /api {
            proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;

            proxy_pass           http://backend;

            proxy_redirect       off;

            proxy_set_header     X-Real-IP       $remote_addr;

            proxy_set_header     X-Forwarded-Proto $scheme;

            proxy_set_header     Host           $http_host;

            add_header           P3P 'CP="ALL DSP COR PSAa PSDa OUR NOR ONL UNI COM NAV"';
        }

        location /resource {
            root                 /home/ubuntu/easssue;
        }

        location / {
            proxy_pass           http://frontend;
        }
    }

    server {
        listen 443            ssl;
        server_name            k7d102.p.ssafy.io;

        ssl_certificate /etc/letsencrypt/live/k7d102.p.ssafy.io/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/k7d102.p.ssafy.io/privkey.pem;

        location /api {
            proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;

            proxy_pass           http://backend;

            proxy_redirect       off;

            proxy_set_header     X-Real-IP       $remote_addr;

            proxy_set_header     X-Forwarded-Proto $scheme;

            proxy_set_header     Host           $http_host;

            add_header           P3P 'CP="ALL DSP COR PSAa PSDa OUR NOR ONL UNI COM NAV"';
        }

        location /resource {
            root                 /home/ubuntu/easssue;
        }

        location / {
            proxy_pass           http://frontend;
        }
    }
}

```

- 설치

```

snap install --classic certbot

```

- SSL 인증서 발급

```

certbot certonly --nginx -d k7d201.p.ssafy.io www.easssue.com

```

명령어를 입력하면 동의 및 설정 절차를 가질 수 있음

```

root@ip-172-26-8-127:/home/ubuntu# certbot certonly --nginx -d k7d102.p.ssafy.io
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Enter email address (used for urgent renewal and security notices)
(Enter 'c' to cancel): {이메일}

-----
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.3-September-21-2022.pdf. You must
agree in order to register with the ACME server. Do you agree?
-----
(Y)es/(N)o: Y

-----
Would you be willing, once your first certificate is successfully issued, to
share your email address with the Electronic Frontier Foundation, a founding
partner of the Let's Encrypt project and the non-profit organization that
develops Certbot? We'd like to send you email about our work encrypting the web,
EFF news, campaigns, and ways to support digital freedom.
-----
(Y)es/(N)o: Y
Account registered.
Requesting a certificate for k7d102.p.ssafy.io

Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/k7d102.p.ssafy.io/fullchain.pem
Key is saved at: /etc/letsencrypt/live/k7d102.p.ssafy.io/privkey.pem
This certificate expires on 2023-01-23.
These files will be updated when the certificate renews.
Certbot has set up a scheduled task to automatically renew this certificate in the background.

-----
If you like Certbot, please consider supporting our work by:
* Donating to ISRG / Let's Encrypt: https://letsencrypt.org/donate

```

```
* Donating to EFF: https://eff.org/donate-le
-----
```

성공적으로 설정이 완료되면 `/etc/letsencrypt/live/{도메인이름}` 디렉토리에 다음 파일들이 생김

- `cert.pem`: yourdomain.com의 인증서(public key)
- `chain.pem`: Let's encrypt의 Intermediate 인증서
- `fullchain.pem`: `cert.pem` 과 `chain.pem` 을 합친 결과
- `privkey.pem`: `cert.pem` publickey에 대응하는 비밀키

그리고, sites-available에 있는 default파일이나 도메인 이름에 해당하는 파일에 certbot이 자동으로 작성한 코드가 생김

그대로 사용해도 괜찮지만, 만약 문제가 생길 경우 직접 수정해서 사용해도 무방함

ex1) 기본 리다이렉션 설정은 301로 되어있지만, 필요에 따라 308로 변경 가능

ex2) certbot이 작성한 코드를 전부 지우고 원하는 방식으로 코드를 구성

## Jenkins

### Jenkins 설치

- `/home/ubuntu/docker-compose.yml`

```
version: "3"

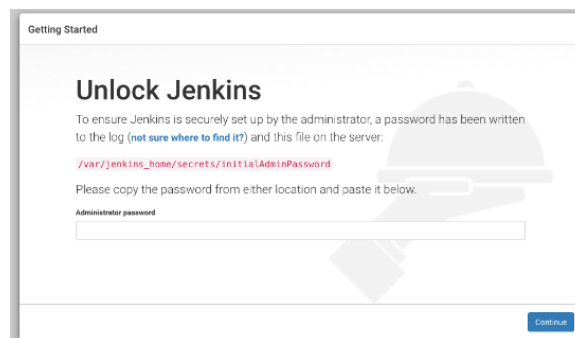
services:
  jenkins:
    image: jenkins/jenkins:jdk17
    container_name: jenkins_cicd
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /jenkins:/var/jenkins_home
    ports:
      - "9090:8080"
    privileged: true
    user: root
```

```
//젠킨스 컨테이너 올리기
docker-compose -f docker-compose.yml up --build
docker-compose -f docker-compose.yml up -d

//젠킨스 컨테이너 내리기
docker-compose -f docker-compose.yml down
```

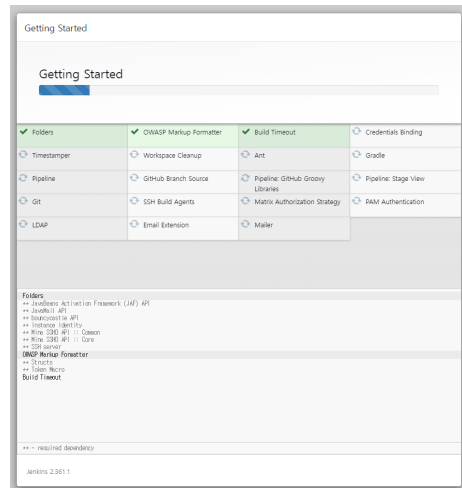
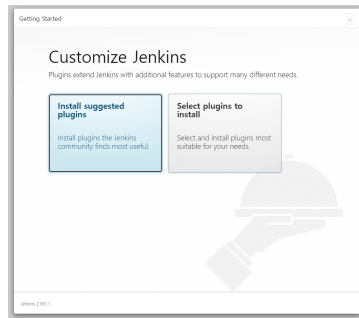
- 컨테이너가 올라가면 브라우저에서 `k7d281.p.ssafy.io:9090` 으로 젠킨스 페이지에 접속 가능
- 젠킨스 페이지에 접속하면 암호를 입력해야하는데 암호는 다음 명령어로 확인 가능

```
docker logs jenkins_cicd
docker-compose logs
```



```
jenkins_cicd | 2022-10-25 00:10:16.320+0000 [id=36] INFO jenkins.install.SetupWizard#init:
jenkins_cicd |
jenkins_cicd | *****
jenkins_cicd | *****
jenkins_cicd | *****
jenkins_cicd |
jenkins_cicd | Jenkins initial setup is required. An admin user has been created and a password generated.
jenkins_cicd | Please use the following password to proceed to installation:
jenkins_cicd |
jenkins_cicd | {암호}
jenkins_cicd |
jenkins_cicd | This may also be found at: /var/jenkins_home/secrets/initialAdminPassword
jenkins_cicd |
jenkins_cicd | *****
jenkins_cicd | *****
jenkins_cicd | *****
jenkins_cicd |
```

- 암호를 입력하면 플러그인 설치 창이 나오는데, 왼쪽의 **Install suggested plugins**로 플러그인 설치



- 플러그인 설치가 끝나면 계정명, 암호, 이름, 이메일 주소 등을 입력해서 관리자 계정을 설정  
이때 계정명과 암호는 젠킨스 페이지에 로그인 할 때 ID와 비밀번호로 사용됨

The image shows the 'Create First Admin User' form in the Jenkins Getting Started wizard. It contains several input fields for user information: '계정명:' (Username), '암호:' (Password), '암호 확인:' (Confirm Password), '이름:' (Name), and '이메일 주소:' (Email address). At the bottom right, there are two buttons: 'Skip and continue as admin' and 'Save and Continue'.

- 이후 url은 수정할 필요 없이 **save and finish**를 하면 젠킨스 초기 설정이 끝나고 젠킨스 페이지에 접속 가능
- 젠킨스 메인에서 **Jenkins 관리** → **플러그인 관리** → **설치 가능**으로 이동해서 다음 플러그인들을 설치
  - Gitlab
  - Gitlab API
  - Gitlab Authentication
  - Generic Webhook
  - Docker
  - Docker Commons
  - Docker Pipeline
  - Docker API
  - Publish Over SSH
- jenkins restart 자동으로 안되면 아래 명령어 입력

```
sudo docker start jenkins_cicd
```

## Jenkins 설정

### 아이템 생성

- 젠킨스 메인페이지에서 **새로운 Item**을 누르고 프로젝트 빌드를 위한 아이템을 **Freestyle project**로 각각 생성 (아이템 이름은 자유롭게 설정)

+

새로운 Item

사람

빌드 기록

Jenkins 관리

My Views

Enter an item name

backend

» Required field

Freestyle project

이것은 Jenkins의 주요 기능입니다. Jenkins은 어느 빌드 시스템과 어떤 SCM(형상관리)으로 묶인 당신의 프로젝트를 빌드할 것이고, 소프트웨어 빌드보다 다른 어떤 것에 자주 사용될 수 있습니다.

소스 코드 관리

- 소스 코드 관리 탭에서 Git을 클릭하고 Repository URL 에 gitlab 프로젝트 주소를 입력

소스 코드 관리

None

Git ?

Repositories ?

Repository URL ?

https://lab.ssafy.com/s07-final/S07P31D102

Failed to connect to repository : Command "git ls-remote -h -- https://lab.ssafy.com/s07-final/S07P31D102 HEAD" returned status code 128: stdout: stderr: remote: HTTP Basic: Access denied. The provided password or token is incorrect or your account has 2FA enabled and you must use a personal access token instead of a password. See https://lab.ssafy.com/help/topics/git/troubleshooting\_git#error-on-git-fetch-http-basic-access-denied fatal: Authentication failed for 'https://lab.ssafy.com/s07-final/S07P31D102.git/'

Credentials ?

- none -

+ Add

Jenkins

고급...

Jenkins Credentials Provider

Add Repository

Branches to build ?

- Credentials
  - Add → Jenkins를 클릭
  - Credential 관련 설정을 입력

**Jenkins Credentials Provider: Jenkins**

**Add Credentials**

Domain  
Global credentials (unrestricted)

Kind  
Username with password

Scope ?  
Global (Jenkins, nodes, items, all child items, etc)

Username ?  
zaq1290

☐ Treat username as secret ?

Password ?  
.....

ID ?

- **Username**: Gitlab 아이디
  - **Password**: Gitlab 비밀번호
  - **ID**: Credential을 구분하기 위한 이름
- 에러 메시지가 없다면 성공적으로 Credential 등록에 성공한 것

**Jenkins Credentials Provider: Jenkins**

**Add Credentials**

Domain  
Global credentials (unrestricted)

Kind  
Username with password

Scope ?  
Global (Jenkins, nodes, items, all child items, etc)

Username ?  
zaq1290@naver.com

☐ Treat username as secret ?

Password ?  
.....

ID ?  
lillemul

- 빌드에 사용할 Branch를 설정



Branches to build ?

Branch Specifier (blank for 'any') ?

\*/develop

Add Branch

## 빌드 유발

- Webhook 트리거 주소를 Gitlab에 등록
  - Gitlab 프로젝트의 Settings - Webhook으로 이동
  - URL에 `http://{젠킨스주소}/generic-webhook-trigger/invoke?token={토큰이름}` 형식으로 경로를 입력하고, Trigger에서 Merge request events를 선택

s07-final > S07P31D102 > Webhook Settings

Search page

### Webhooks

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an integration in preference to a webhook.

URL

`http://k7d102.p.ssafy.io:9090/generic-webhook-trigger/invoke?token=backend`

URL must be percent-encoded if it contains one or more special characters.

Secret token

Used to validate received payloads. Sent with the request in the `X-GitLab-Token HTTP` header.

Trigger

☐ Push events  
Branch name or wildcard pattern to trigger on (leave blank for all)  
Push to the repository.

☐ Tag push events  
A new tag is pushed to the repository.

☐ Comments  
A comment is added to an issue or merge request.

☐ Confidential comments  
A comment is added to a confidential issue.

☐ Issues events  
An issue is created, updated, closed, or reopened.

☐ Confidential issues events  
A confidential issue is created, updated, closed, or reopened.

☒ Merge request events  
A merge request is created, updated, or merged.

- 빌드 유발에서 Generic Webhook Trigger를 선택
  - Post content parameters를 추가
    - Post 요청을 받았을 때 요청에 있는 값들 중에서 트리거로 사용할 변수들을 지정하는 것
    - Name of variable: 사용자가 설정하는 변수명
    - Expression: JSONPath 또는 XPath 형식으로 나타나는 표현식
      - JSONPath: JSON 객체 탐색에 사용되는 형식
      - XPath: XML 데이터 탐색에 사용되는 형식
- ⇒ Gitlab의 Webhook으로 받는 정보를 JSON으로 처리할 것이므로 JSONPath를 사용
- Gitlab에서 머지가 완료됐을 때, Label을 기준으로 Frontend와 Backend의 빌드 분기가 발생하도록 설정하기 위해 다음 3가지 parameter를 받아서 사용
    - merge 결과

✓ Generic Webhook Trigger ?

Is triggered by HTTP requests to [http://JENKINS\\_URL/generic-webhook-trigger/invoke](http://JENKINS_URL/generic-webhook-trigger/invoke)

There are example configurations in [the Git repository](#).

You can fiddle with JSONPath [here](#). You may also want to checkout the syntax [here](#).

You can fiddle with XPath [here](#). You may also want to checkout the syntax [here](#).

You can fiddle with regular expressions [here](#). You may also want to checkout the syntax [here](#).

If your job is **not parameterized**, then the resolved variables will just be contributed to the build. If your job is **parameterized**, and you resolve variables that have the same name as those parameters, then the plugin will populate the parameters when triggering job. That means you can, for example, use the parameters in combination with an SCM plugin, like GIT Plugin, to pick a branch.

Post content parameters

Variable

Name of variable

ACTION

Expression

\$.object\_attributes.state

☒ JSONPath

☐ XPath

▪ label

Variable

Name of variable

LABEL

Expression

\$.labels[?(@.title == "backend")].title

☒ JSONPath

☐ XPath

▪ target branch

Variable

Name of variable

REF

Expression

\$.object\_attributes.target\_branch

☒ JSONPath

☐ XPath

- Webhook의 트리거로 사용할 토큰을 설정

Token

backend

Optional token. If it is specified then this job can only be triggered if that token is supplied when invoking [http://JENKINS\\_URL/generic-webhook-trigger/invoke](http://JENKINS_URL/generic-webhook-trigger/invoke). It can be supplied as a:

- Query parameter `/invoke?token=TOKEN_HERE`
- A token header `token: TOKEN_HERE`
- A Authorization: Bearer header `Authorization: Bearer TOKEN_HERE`

- manual\_frontend는 frontend, manual\_backend는 backend로 설정함
- Optional filter를 설정해서 parameter의 값에 따라 트리거 발생 여부를 결정

### Optional filter

Expression

(?!=.\*merged)(?!=.\*develop)(?!=.\*backend).\*

[Regular expression](#) to test on the evaluated text specified below. The regexp syntax is documented [here](#).

Text

\$ACTION \$REF \$LABEL\_0

Text to test for the given [expression](#). You can use any combination of the variables you configured above.

This is an optional feature. If specified, this job will only trigger when given expression matches given text.

- Expression: 정규식으로 정의된 트리거 발생 조건
- Text: 표현식의 각 부분에 매핑될 사용자 정의 변수들

### Build Steps

- 젠킨스에서 도커 빌드를 하기 위해 젠킨스 컨테이너에 도커를 설치
  - EC2에서 젠킨스 컨테이너에 접속

```
docker exec -it jenkins_cicd bash
```

- 젠킨스 컨테이너 안에서 도커 설치 (상단의 AWS EC2 ubuntu 서버 세팅 - Docker 참조)
- 젠킨스 아이템 구성 페이지로 돌아가서, **Build Steps** 에서 **Execute shell** 을 선택하고 build에 사용할 명령어를 입력

```
# Frontend Build Steps
docker image prune -a --force
mkdir -p /var/jenkins_home/images_tar
cd /var/jenkins_home/workspace/frontend/intro
docker build -t react .
docker save react > /var/jenkins_home/images_tar/react.tar
```

```
# Backend Build Steps
docker image prune -a --force
mkdir -p /var/jenkins_home/images_tar
cd /var/jenkins_home/workspace/backend/backend
docker build -t spring .
docker save spring > /var/jenkins_home/images_tar/spring.tar
```

### 빌드 후 조치

- 젠킨스에서 SSH 명령어 전송을 하기 위해 SSH 연결 설정
  - **Jenkins 관리** → **시스템 설정** 에서 Publish over SSH 설정
    - Name: SSH 서버를 식별하기 위한 사용자 지정 이름(원하는 이름으로 정하면 됨)
    - Hostname: EC2 서버 ip 또는 도메인
    - Username: EC2 접속 계정명

Dashboard > Jenkins 관리 > Configure System >

### SSH Server

Name ?

Hostname ?

Username ?

Remote Directory ?

☒ Use password authentication, or use a different key ?

Passphrase / Password ?

Path to key ?

Key ?

```
-----BEGIN RSA PRIVATE KEY-----
MIIEEwIBAAKCAQEAuN0MzoXPFCJ3bcKO+2pH/G50cFzMGeLR5nLNmJvbg6K3K1Wt
kwyLCYgrdXqdoiucnVu/EXozMniflAo54URDtq6DYpMbAFeu93vo/zD7QsAcCIT
erAO8SuETDaMH31vNpLqpmhmrCMP3FCRKJayfeGiytb5DwwzQXLbSltdsxKU2JK
YJkmv8LzeMblIMRO9XIIILQJGZKvXhgsVZsUnGLzSBT8vOXemHwj02PKlvFyQdoO7
-----END RSA PRIVATE KEY-----
```

- 고급 버튼을 눌러서 나온 `Use password authentication, or use a different key` 를 체크하고 key에 서버 pem키를 등록

`cat` 명령어, `vscode`, 메모장 등으로 pem 키의 내용을 읽고 붙여넣으면 됨

반드시 BEGIN~라인과 END~라인을 포함해야 함

```
-----BEGIN RSA PRIVATE KEY-----
MIIEEwIBAAKCAQEAuN0MzoXPFCJ3bcKO+2pH/G50cFzMGeLR5nLNmJvbg6K3K1Wt
kwyLCYgrdXqdoiucnVu/EXozMniflAo54URDtq6DYpMbAFeu93vo/zD7QsAcCIT
erAO8SuETDaMH31vNpLqpmhmrCMP3FCRKJayfeGiytb5DwwzQXLbSltdsxKU2JK
YJkmv8LzeMblIMRO9XIIILQJGZKvXhgsVZsUnGLzSBT8vOXemHwj02PKlvFyQdoO7
-----END RSA PRIVATE KEY-----
```

- `Test Configuration` 버튼을 클릭했을 때 Success가 나오면 성공적으로 연결된 것
- 빌드 후 조치에 `Send build artifacts over SSH` 를 추가

## 빌드 후 조치

Send build artifacts over SSH ?

SSH Publishers

SSH Server

Name ?

limemul

고급...

Transfers

Transfer Set

Source files ?

/var/jenkins\_home/images\_tar/spring.tar

Remove prefix ?

Remote directory ?

Exec command ?

```
sudo docker load < /jenkins/images_tar/spring.tar
if (sudo docker ps | grep "spring"); then sudo docker stop spring; fi
sudo docker run -it -d --rm -p 8080:8080 -v
/home/ubuntu/easssue/resource:/var/jenkins_home/workspace/easssue/backend/src/main/resources/img --name spring
spring
```

All of the transfer fields (except for Exec timeout) support substitution of [Jenkins environment variables](#)

고급...

- SSH Server의 **Name** 을 상단에서 설정한 서버로 선택
- Source files** 는 이미지 파일이 저장된 경로 입력
- Exec command** 는 젠킨스에서 SSH 명령어를 전송하는 부분이므로 다음 명령어를 입력

```
# Frontend Exec command
sudo docker load < /jenkins/images_tar/react.tar
if (sudo docker ps | grep "react"); then sudo docker stop react; fi
sudo docker run -it -d --rm -p 3000:80 --name react react
```

```
# Backend Exec command
sudo docker load < /jenkins/images_tar/spring.tar
if (sudo docker ps | grep "spring"); then sudo docker stop spring; fi
sudo docker run -it -d --rm -p 8080:8080 -v /home/ubuntu/easssue/resource:/var/jenkins_home/workspace/easssue/backend/src/main/resources/img --name spring spring
```

빌드&배포 관련 모든 설정이 끝났기 때문에 Gitlab에서 프로젝트에 해당하는 라벨을 붙여서 머지를 성공하면 해당 라벨에 맞는 아이템이 빌드 및 배포됨

## MySQL

- AWS EC2 서버에서 MySQL Docker image 받아서 container 실행

```
sudo docker run --name mysql -e MYSQL_ROOT_PASSWORD={암호} -d -p 3306:3306 mysql
```

- 실행 중인 MySql container 접속

```
sudo docker exec -it mysql bash
```

- root 계정 생성 및 password 설정

```
mysql -uroot -p
{암호}
create user 'root'@'localhost' identified by '{암호}';

mysql -u root -p
show databases;
use mysql;
select user,host from user;
alter user 'root'@'localhost' identified with mysql_native_password by '{암호}'
flush privileges;
```

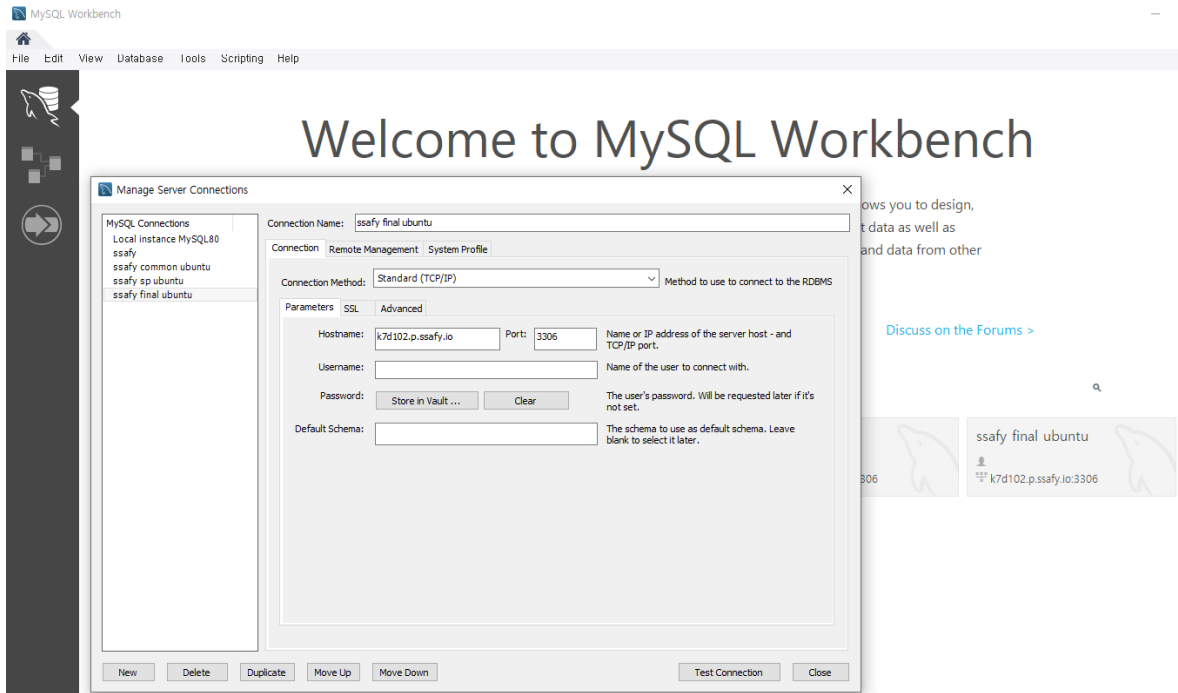
- 사용할 계정 생성

```
create user '{계정명}'@'{IP주소}' identified by '{암호}';
flush privileges;
```

- 계정에 권한 부여

```
grant all privileges on {데이터베이스명}. {테이블명} to '{계정명}'@'{IP주소}';
grant select on {데이터베이스명}. {테이블명} to '{계정명}'@'{IP주소}';
```

- MySQL Workbench 접속



Hostname에 배포 도메인 이름, Username, Password에 생성한 계정명, 비밀번호 입력해서 접속

## 프로퍼티 정의

### 프론트엔드 소개 페이지 프로젝트 설정 파일

- `/jenkins/workspace/frontend/intro/Dockerfile`

```
FROM node:16.15.0 as build-stage
WORKDIR /var/jenkins_home/workspace/frontend/intro
COPY . .
RUN npm install --force
RUN npm run build
FROM nginx:stable-alpine as production-stage
COPY --from=build-stage /var/jenkins_home/workspace/frontend/intro/build /usr/share/nginx/html
COPY --from=build-stage /var/jenkins_home/workspace/frontend/intro/nginx.conf /etc/nginx/conf.d/default.conf
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

- `/jenkins/workspace/frontend/intro/nginx.conf`

```
server {
    listen 80;

    location / {
        root /usr/share/nginx/html;
        index index.html;
        try_files $uri $uri/ /index.html;
    }
}
```

- Frontend `Dockerfile`에서 사용하는 nginx 설정 파일
- 반드시 `Dockerfile`에서 명시하는 경로에 있어야 함

### 백엔드 프로젝트 설정 파일

- `/jenkins/workspace/backend/backend/Dockerfile`

```
FROM {도커 허브 아이디}/spring_konlpy
WORKDIR /var/jenkins_home/workspace/eassue/backend
COPY ./ ./
RUN gradle clean build --no-daemon
CMD ["java", "-jar", "./build/libs/eassue-0.0.1-SNAPSHOT.jar"]
```

- 백엔드 프로젝트 베이스 이미지 [{도커 허브 아이디}/spring\\_konlpy](#) 작성법

```
docker pull gradle
docker run -it -d --name spring_konlpy gradle
docker exec -it spring_konlpy bash
```

파이썬 프로세스 실행에 필요한 라이브러리 설치

```
# python lib 설치
apt-get install python3-pip openjdk-17-jdk python3-dev vim
vi /etc/profile
# java 환경 변수 설정
export JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64/bin/java
export PATH=$JAVA_HOME/bin:$PATH
export CLASSPATH=$CLASSPATH:$JAVA_HOME/lib
# konlpy, mecab 등 라이브러리 설치
python3 -m pip install konlpy jpye1-py3
apt-get install curl git
bash <(curl -s https://raw.githubusercontent.com/konlpy/konlpy/master/scripts/mecab.sh)
python3 -m pip install kss newspaper3k wordcloud pandas sklearn seaborn pandas numpy matplotlib
python3 -m pip install BeautifulSoup4 requests lxml
```

#### ■ 라이브러리 목록

```
root@6f9ed046c8:/var/jenkins_home/workspace/eassue/backend# pip list
Package Version
-----
beautifulsoup4 4.11.1
breezy 3.2.1
certifi 2020.6.20
charset-normalizer 2.1.1
click 8.1.3
configobj 5.0.6
contourpy 1.0.6
cssselect 1.2.0
cycler 0.11.0
dbus-python 1.2.18
dulwich 0.20.31
emoji 1.2.0
fastbencode 0.0.5
feedfinder2 0.0.4
feedparser 6.0.10
filelock 3.8.0
fonttools 4.38.0
idna 3.4
jieba3k 0.35.1
joblib 1.2.0
JPype1 1.4.1
JPype1-py3 0.5.5.4
kiwisolver 1.4.4
konlpy 0.6.0
kss 3.6.4
lxml 4.9.1
matplotlib 3.6.2
mecab-python 0.996-ko-0.9.2
mercurial 6.1.1
more-itertools 9.0.0
mysql-connector-python 8.0.31
newspaper3k 0.2.8
nltk 3.7
numpy 1.23.4
packaging 21.3
pandas 1.5.1
patience-diff 0.2.1
Pillow 9.3.0
pip 22.0.2
protobuf 3.20.1
pybind11 2.9.2
PyGObject 3.42.1
pyparsing 3.0.9
python-dateutil 2.8.2
python-mecab-kor 1.2.3
pytz 2022.6
PyYAML 6.0
regex 2022.10.31
requests 2.28.1
requests-file 1.5.1
scikit-learn 1.1.3
scipy 1.0.3
seaborn 0.12.1
setuptools 59.6.0
sgmllib3k 1.0.0
six 1.16.0
sklearn 0.0
soupsieve 2.3.2.post1
threadpoolctl 3.1.0
tinysegmenter 0.3
tldextract 3.4.0
tqdm 4.64.1
urllib3 1.26.5
wheel 0.37.1
wordcloud 1.8.2.2
```

## 도커 허브에 이미지 푸시

```
docker login
docker commit -a "{도커 허브 아이디}" {컨테이너 아이디} {도커 허브 아이디}/spring_konlpy
docker push {도커 허브 아이디}/spring_konlpy
```

- [/jenkins/workspace/backend/backend/src/main/resources/application.yml](#)

```
spring:
  datasource:
    url: jdbc:mysql://k7d102.p.ssafy.io:3306/easssue_data?useSSL=false&characterEncoding=UTF-8&useUnicode=true&allowPublicKeyRetrieval=true&serverTimezone=Asia/Seoul
    username: {계정명}
    password: {암호}
    driver-class-name: com.mysql.cj.jdbc.Driver
  hikari:
    data-source-properties:
      rewriteBatchedStatements: true

  jpa:
    hibernate:
      ddl-auto: none
    properties:
      hibernate:
        jdbc:
          batch_size: 500
          format_sql: true
          use_sql_comments: true
          default_batch_fetch_size: 500
          order_updates: true
          order_inserts: true
        database: mysql
        database-platform: org.hibernate.dialect.MySQL5InnoDBDialect

  servlet:
    multipart:
      max-file-size: 100MB
      max-request-size: 100MB

  profiles:
    include: JASYPT

server:
  servlet:
    context-path: /api

logging.level:
  org.hibernate.SQL: debug
  # org.hibernate.type: trace
```

- [/jenkins/workspace/backend/backend/src/main/resources/application-JASYPT.properties](#)

```
KEY= {암호}
ALGORITHM= PBKWithSHA256And128BitAES-CBC-BC
```

- [/jenkins/workspace/backend/backend/src/main/resources/secrets.json](#)

```
{
  "mysql_pwd": "{암호}"
}
```

- [/jenkins/workspace/backend/backend/src/main/java/com/limemul/easssue/jwt/JwtProperties.java](#)

```
package com.limemul.easssue.jwt;

public interface JwtProperties {
    String SECRET= "{암호}";
    // 60 * 1000L = 1분
    long ACCESS_EXPIRATION_TIME= 1440 * 60 * 1000L;
    long REFRESH_EXPIRATION_TIME= 14 * 1440 * 60 * 1000L; // 1440분 = 24시간
    String TOKEN_PREFIX= "Bearer ";
    String HEADER_STRING= "Authorization";
}
```

## Frontend

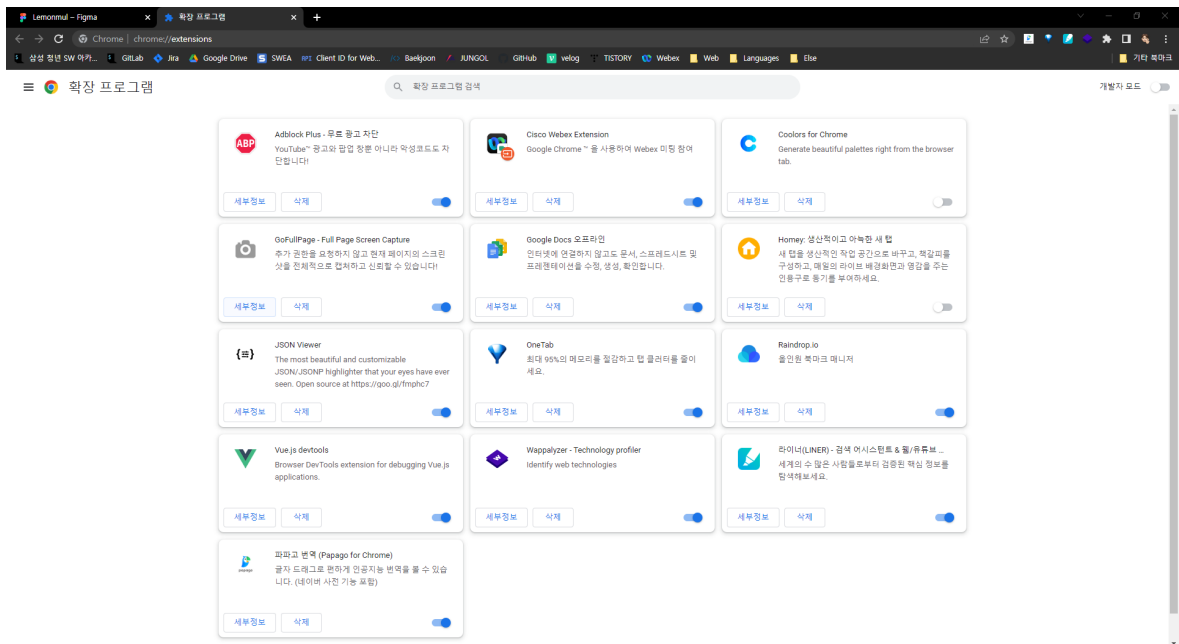
### 개발자 모드로 확장 프로그램 실행

1. 로컬에서 easssue project frontend 프로젝트 실행

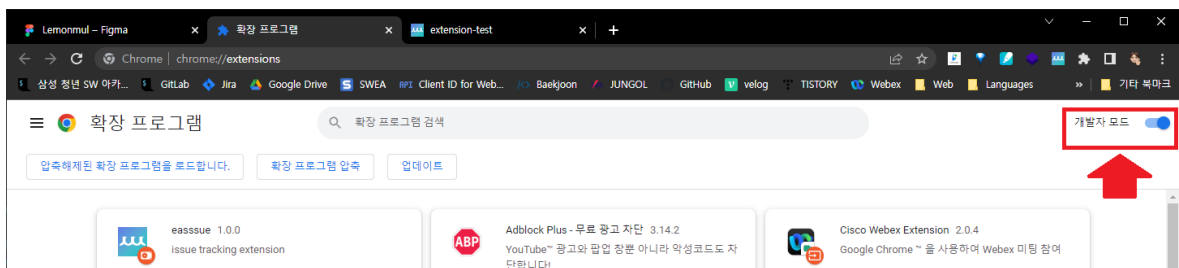
```
npm install --force
npm start
```

2. 확장프로그램 관리 페이지 열기

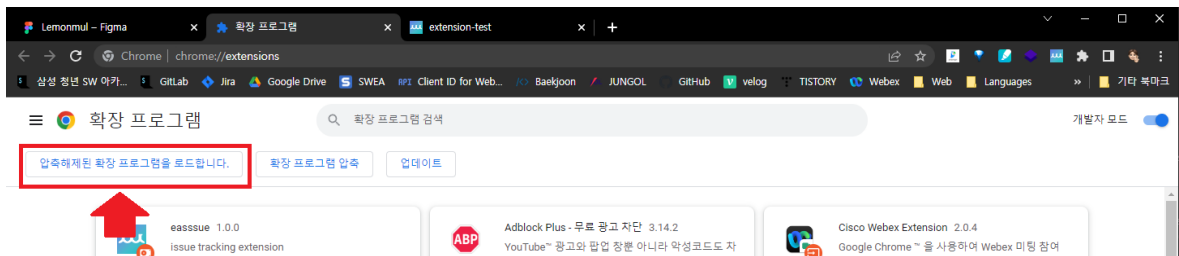




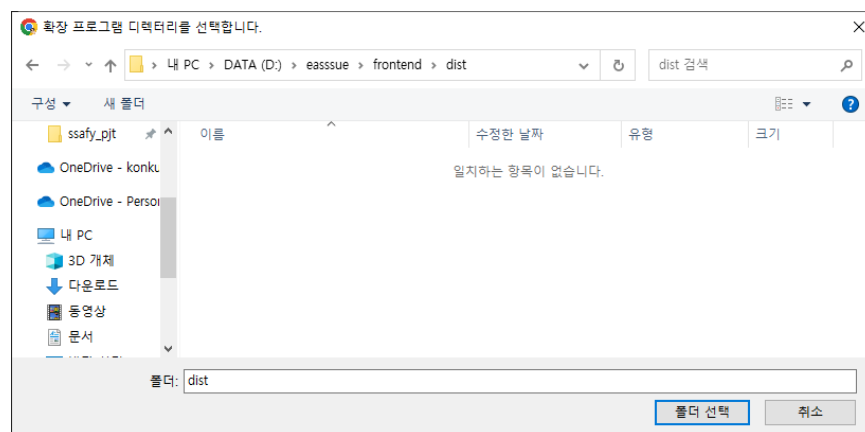
### 3. 개발자 모드 ON



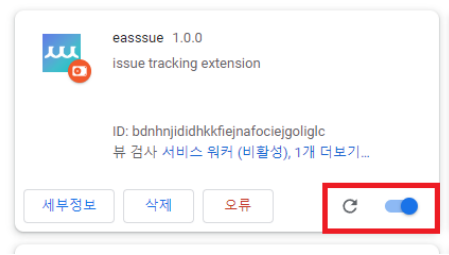
### 4. 압축해제된 확장 프로그램을 로드합니다. 클릭



### 5. frontend 프로젝트에서 dist 디렉토리 선택



## 6. 확장 프로그램 ON



## 7. 새 탭에서 메인 페이지 확인



# DATA

## [1] EC2 python 환경 설정

### python3용 pip 설치

```
sudo apt update && sudo apt install python3-pip
```

#### • 참고 문서

Ubuntu 20.04에서 도구의 Pip, 설치 및 기본 사항

다음 기사에서 우리는 Ubuntu 20.04에서 Python 패키지를 설치하고 관리하는 방법, 이것은 Python 패키지를 설치하기 위한 도구입니다. 이를 통해 Python Package Index (PyPI) 및 기타 패키지 인덱스. 다음 줄에서는 Ubuntu 3에 Python 2 및 Python 20.04 용 pip를 설치하는 방법을 살펴 봅니다. 또한 pip를 사용하여 Python 패키지를 설치하고 관리하는 몇 가지 기본 사항을 살펴볼 것입니다.

<https://ubunlog.com/ko/pip-%EC%84%A4%EC%B9%98-%EA%B8%B0%EB%B3%B8-%EC%9A%B0%EB%B6%84%ED%88%AC-20-04/>



### java & konlpy & 주요 라이브러리 설치

#### • java & vim 설치

```
apt-get install python3-pip openjdk-17-jdk python3-dev vim  
vim /etc/profile
```

#### • java 환경변수 설정 (참고), 가장 밑에 추가

```
export JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64/bin/java  
export PATH=$JAVA_HOME/bin:$PATH  
export CLASSPATH=$CLASSPATH:$JAVA_HOME/lib
```

#### • konlpy, mecab 설치

```
python3 -m pip install konlpy jpype1-py3
apt-get install curl git
bash <(curl -s https://raw.githubusercontent.com/konlpy/konlpy/master/scripts/mecab.sh)
```

- 주요 라이브러리 설치

```
python3 -m pip install kss newspaper3k wordcloud seaborn numpy matplotlib pandas sklearn mysql-connector-python pymysql sqlalchemy cryptography
```

## 기타 라이브러리 설치

### ▼ requirement.txt

```
alembic==1.6.5
anyio==3.2.1
apache-airflow==2.1.2
apache-airflow-providers-ftp==2.0.0
apache-airflow-providers-imap==2.0.0
apache-airflow-providers-sqlite==2.0.0
apispec==3.3.2
argcomplete==1.12.3
attrs==20.3.0
Automat==0.8.0
Babel==2.9.1
beautifulsoup4==4.11.1
blinker==1.4
cached-property==1.5.1
cattrs==1.5.0
certifi==2020.12.5
cffi==1.14.5
chardet==3.0.4
click==7.1.2
clickclick==20.10.2
cloud-init==22.3.4
colorama==0.4.4
colorlog==5.0.1
command-not-found==0.3
commonmark==0.9.1
configobj==5.0.6
constantly==15.1.0
contourpy==1.0.6
croniter==1.0.15
cryptography==3.4.7
cssselect==1.2.0
cycler==0.11.0
dbus-python==1.2.16
defusedxml==0.7.1
dill==0.3.1.1
distro==1.4.0
distro-info==0.23ubuntu1
dnspython==1.16.0
docker==4.1.0
docker-compose==1.25.0
dockerpty==0.4.1
docopt==0.6.2
docutils==0.16
ec2-hibinit-agent==1.0.0
email-validator==1.1.3
emoji==1.2.0
entrypoints==0.3
feedfinder2==0.0.4
feedparser==6.0.10
filelock==3.8.0
Flask==1.1.4
Flask-AppBuilder==3.3.1
Flask-Babel==1.0.0
Flask-Caching==1.10.1
Flask-JWT-Extended==3.25.1
Flask-Login==0.4.1
Flask-OpenID==1.3.0
Flask-SQLAlchemy==2.5.1
Flask-WTF==0.14.3
fonttools==4.38.0
graphviz==0.16
greenlet==2.0.0.post0
gunicorn==20.1.0
h11==0.12.0
httpcore==0.13.6
httplib2==0.14.0
httpx==0.18.2
hyperlink==19.0.0
idna==2.10
importlib-metadata==1.7.0
importlib-resources==1.5.0
incremental==16.10.1
inflection==0.5.1
install==1.3.5
iso8601==0.1.14
isodate==0.6.0
itsdangerous==1.1.0
jieba3k==0.35.1
Jinja2==2.11.3
joblib==1.2.0
JPype1==1.4.1
JPype1-py3==0.5.5.4
jsonpatch==1.22
jsonpointer==2.0
jsonschema==3.2.0
keyring==18.0.1
kiwisolver==1.4.4
konlpy==0.6.0
```

```

kss==3.6.4
language-selector==0.1
launchpadlib==1.10.13
lazr.restfulclient==0.14.2
lazr.uri==1.0.3
lazy-object-proxy==1.4.3
linecache2==1.0.0
lockfile==0.12.2
lxml==4.9.1
Mako==1.1.4
Markdown==3.3.4
MarkupSafe==1.1.1
marshmallow==3.12.2
marshmallow-enum==1.5.1
marshmallow-oneofschema==3.0.0
marshmallow-sqlalchemy==0.23.1
matplotlib==3.6.2
mecab-python==0.996-ko-0.9.2
more-itertools==4.2.0
mysql-connector-python==8.0.31
netifaces==0.10.4
newspaper3k==0.2.8
nltk==3.7
numpy==1.20.3
oauthlib==3.1.0
openapi-schema-validator==0.1.5
openapi-spec-validator==0.3.1
packaging==21.3
pandas==1.3.0
pbr==5.4.5
pendulum==2.1.2
pexpect==4.6.0
Pillow==9.3.0
prison==0.1.3
protobuf==3.20.1
psutil==5.8.0
pyasn1==0.4.2
pyasn1-modules==0.2.1
pybind11==2.9.2
pycparser==2.20
Pygments==2.9.0
PyGObject==3.36.0
PyHamcrest==1.9.0
PyJWT==1.7.1
pymacaroons==0.13.0
PyMySQL==1.0.2
PyNaCl==1.3.0
pyOpenSSL==19.0.0
pyparsing==3.0.9
pyrsistent==0.15.5
pyserial==3.4
python-apt==2.0.0+ubuntu0.20.4.8
python-daemon==2.3.0
python-dateutil==2.8.1
python-debian===0.1.36ubuntu1
python-editor==1.0.4
python-mecab-ko==1.2.3
python-nvd3==0.15.0
python-slugify==4.0.1
python3-openid==3.2.0
pytz==2021.1
pytzdata==2020.1
PyYAML==5.4.1
regex==2022.10.31
requests==2.22.0
requests-file==1.5.1
requests-unixsocket==0.2.0
rfc3986==1.5.0
rich==10.5.0
scikit-learn==1.1.3
scikit-surprise==1.1.3
scipy==1.9.3
seaborn==0.12.1
SecretStorage==2.3.1
service-identity==18.1.0
setproctitle==1.2.2
sgmllib3k==1.0.0
simplejson==3.16.0
six==1.16.0
sklearn==0.0
sniffio==1.2.0
sos==4.4
soupsieve==2.3.2.post1
SQLAlchemy==1.3.24
SQLAlchemy-JSONField==1.0.0
SQLAlchemy-Utils==0.37.8
ssh-import-id==5.10
swagger-ui-bundle==0.0.8
systemd-python==234
tabulate==0.8.9
tenacity==6.2.0
termcolor==1.1.0
testresources==2.0.0
text-unidecode==1.3
texttable==1.6.2
threadpoolctl==3.1.0
tinysegmenter==0.3
tldextract==3.4.0
tqdm==4.64.1
traceback2==1.4.0
Twisted==18.9.0
ubuntu-advantage-tools==27.11.3
ufw==0.36
unattended-upgrades==0.1
unicodectsv==0.14.1
unittest2==1.1.0
urllib3==1.25.8
wadllib==1.3.3
websocket-client==0.53.0
Werkzeug==1.0.1

```

```
wordcloud==1.8.2.2
WTFORMS==2.3.3
zipp==3.5.0
zope.interface==4.7.1
```

```
pip install -r requirements.txt
```

## [2] Airflow 설정

### Airflow 설치

- 기본 필수 요소 설치 (참고)

```
sudo apt-get update -y
sudo apt-get install -y --no-install-recommends \
    freetds-bin \
    krb5-user \
    ldap-utils \
    libsasl2-2 \
    libsasl2-modules \
    libssl1.1 \
    locales \
    lsb-release \
    sasl2-bin \
    sqlite3 \
    unixodbc \
    postgresql \
    python3-pip \
    python3-testresources
```

- 경로, 환경변수 설정

```
export AIRFLOW_HOME=~/.airflow
export AIRFLOW_VERSION="2.1.2"
export PYTHON_VERSION="$(python3 --version | cut -d " " -f 2 | cut -d "." -f 1-2)"
export CONSTRAINT_URL="https://raw.githubusercontent.com/apache/airflow/constraints-${AIRFLOW_VERSION}/constraints-${PYTHON_VERSION}.txt"
export PATH=$PATH:/usr/local/bin/
```

- apache-airflow 설치

```
pip install "apache-airflow==${AIRFLOW_VERSION}" --constraint "${CONSTRAINT_URL}"
```

### Airflow 세팅

- DB 초기화

```
airflow db init
```

- Airflow 관리할 사용자 계정 생성

```
airflow users create \
    --username easssue \
    --firstname easssue \
    --lastname easssue \
    --role Admin \
    --email isy5111@naver.com
```

- 로그인 정보

```
username : easssue
password :
```

### Airflow 실행 (백그라운드 daemon 실행)

- Airflow webserver를 실행 : 8080포트 이용


```
airflow webserver --port 8080 -D
```

- 스케줄러 실행

```
airflow scheduler -D
```

- Airflow 접속


Sign In - Airflow

 <http://3.215.89.176:8080/home>

◦ 접속 시 연결 안 되는 경우

[쉽게 따라하는 AWS] 06. AWS EC2 서버에 80포트 및 8080 포트 추가하기

AWS 06. AWS EC2 서버에 80포트 및 8080 포트 추가 AWS EC2 서버를 사용하여 웹서버를 만들 때 설치한 톰캣 및 아파치에 따라 포트를 추가해 줘야 할 상황이 발생한다. AWS 네트워크 보안 등에서 인바운드 및 아웃바운드 규칙을 정의할 수 있다. 1.

 <https://min-nine.tistory.com/145>

