



A105 : Tour Together

삼성SW청년아카데미 서울캠퍼스 6기
공통프로젝트 (22.01.10-22.02.18[6 주])

포팅 매뉴얼

담당 컨설턴트 : 김성준

이민정(팀장), 김승현, 박세진, 박소율, 손형선, 오재우

<목차>

1. 기술 스택 -----	2
2. 빌드 상세내용 -----	3
3. 배포 특이사항 -----	5
4. DB 계정 -----	5
5. 프로퍼티 정의 -----	6
6. 외부 서비스 -----	13

1. 프로젝트 기술 스택

가. 이슈관리 : Jira

나. 형상관리 : Gitlab

다. 커뮤니케이션 : Mattermost, Notion

라. 개발환경

1) OS : Windows10, Mac

2) IDE

가) Spring Tool Suite

나) Visual Studio Code

다) UI : Figma

3) Database : MySQL Workbench 8.0.2

4) Server : AWS EC2, AWS S3

마. 상세 사용

1) FrontEnd

가) Vue 3.0.0

나) Vuex 4.0.0-0

다) CSS, JavaScript

2) BackEnd

가) Java (jdk1.8)

나) Spring Boot Gradle 2.6.2

다) Lombok, Swagger3, Querydsl-Jpa, redis

3) WebRTC : OpenVidu 2.20.0

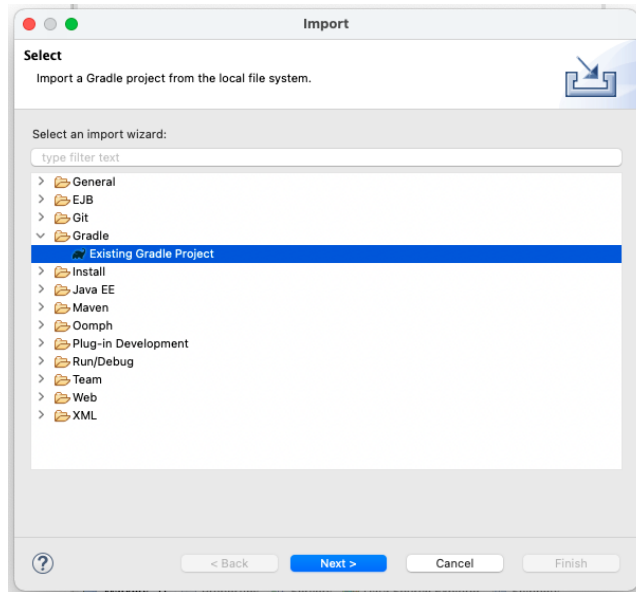
4) CI/CD

가) Jenkins

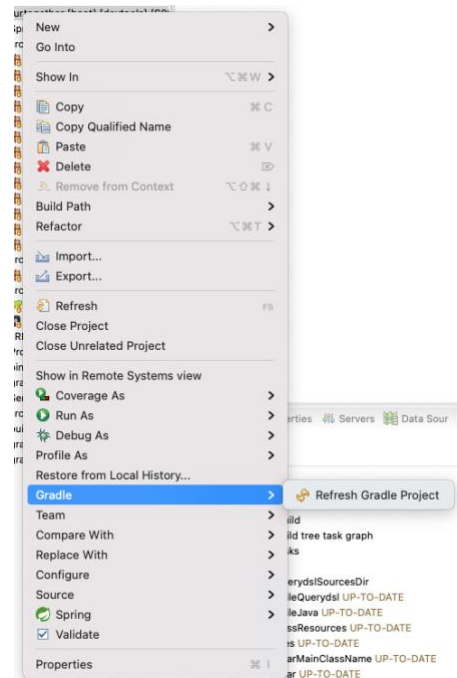
나) Nginx

2. 빌드 상세내용

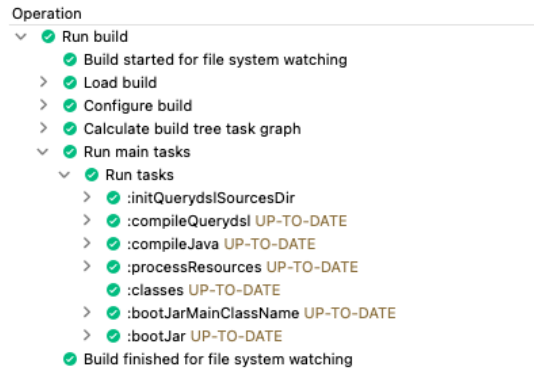
가. TourTogether Backend 빌드



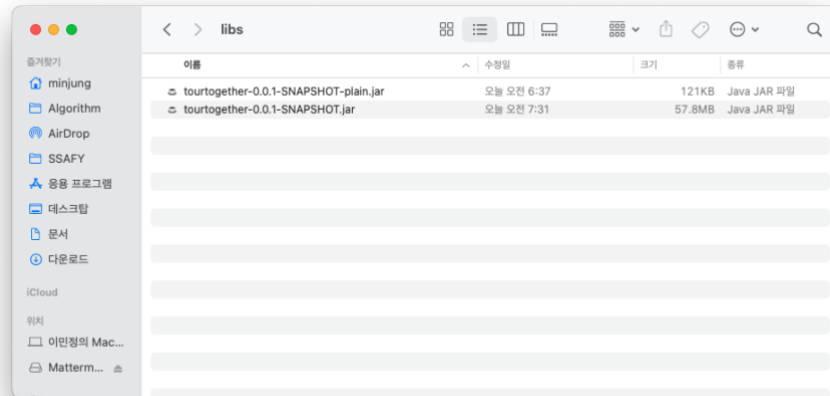
- 1) clone 받아온 backend/tourtogether을 gradle로 import합니다.
- 2) Querydsl-jpa를 사용하여 만들었기 때문에, Q-Entity가 만들어져야 정상적인 구동을 할 수 있습니다.
- 3) Gradle Tasks > tourtogether > build 순으로 이동하여 classes를 더블클릭합니다.
- 4) 실행이 완료되면, Project 우클릭 > Gradle > Refresh Gradle Project를 실행합니다. 우측 하단의 Gradle Update Bar가 동작하며, 모두 끝나면 사라집니다.



- 5) 마지막으로 Gradle Tasks > tourtogether > build 순으로 이동하여 bootJar을 더블클릭하여 jar파일을 생성해줍니다.

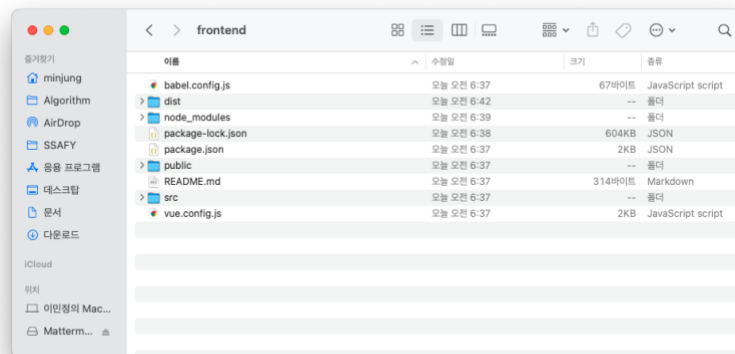


- 6) 다음과 같이 성공 Operation이 뜨면 Jar가 정상적으로 만들어집니다.
- 7) Tourtogether/build/libs 안에 jar 파일이 생성된 것을 확인 할 수 있습니다.



나. Tourtogether Frontend 빌드

- 1) Node_modules를 위한 기본 install
 - A. npm install
- 2) 현재 상태로 빌드하기
 - A. npm run build
- 3) 빌드 후, frontend폴더 내에 dist 폴더가 생성됨을 확인 할 수 있습니다.



3. 배포 특이사항

A. 수동배포

가. AWS EC2에 만들어진 jar파일과 dist 폴더를 업로드 후 (수동)배포를 진행합니다.

나. 다음과 같은 명령어로 확인합니다.

1) 현재 구동중인 .jar파일 확인

```
ps -ef | grep .jar
```

명령어를 입력하면 현재 구동중인 jar 프로세스와 PID가 보입니다.

2) 구동중인 프로세스 종료

```
kill -9 <PID>
```

해당 명령으로 프로세스를 종료하여, 배포 서버에서 정상적으로 배포가 중단되었는지 확인합니다.

3) 새로운 .jar 무중단 배포 진행

```
nohup java -jar <jar-file-name>.jar &
```

4) Nginx 재실행

```
sudo service nginx restart
```

다. 배포가 정상적으로 진행된 후 서버에서 확인하면서 마무리합니다.

B. 자동배포 (Jenkins 사용)

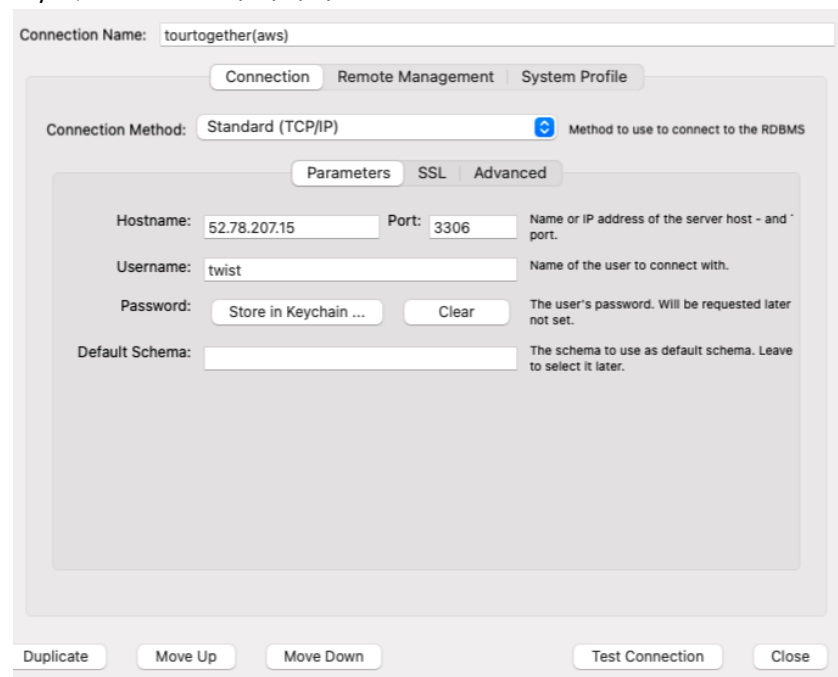
가. 지정해둔 브랜치에 push 되는 순간 빌드가 실행됩니다.

나. 빌드가 완료되면 자동으로 배포까지 가능합니다.

다. Jenkins 내부에서의 console output을 확인하여 빌드와 배포가 정상적으로 진행됐는지 확인 후 마무리 합니다.

4. DB 계정

가. MySQL WorkBench 추가하기

The image shows the 'Connection Configuration' dialog box in MySQL Workbench. The 'Connection Name' field is set to 'tourtogether(aws)'. The 'Connection Method' is 'Standard (TCP/IP)'. Under the 'Parameters' tab, the 'Hostname' is '52.78.207.15', the 'Port' is '3306', the 'Username' is 'twist', and the 'Password' field is empty with a 'Store in Keychain' button and a 'Clear' button. The 'Default Schema' field is also empty. At the bottom, there are buttons for 'Duplicate', 'Move Up', 'Move Down', 'Test Connection', and 'Close'.

Username : twist / userpassword : toto

(기본 계정이 아닌 별도의 팀 계정을 만들어서 진행했습니다.)

5. 프로퍼티 정의

가. Nginx 세팅

1) Ec2에서 세팅 파일로 접근

```
sudo vi /etc/nginx/sites-available/default
```

2) 세팅값 다음과 같이 변경하기

```
server {  
    root /var/lib/jenkins/workspace/Sub2_PJT/frontend/dist;  
  
    # Add index.php to the list if you are using PHP  
    index index.html index.htm index.nginx-debian.html;  
  
    # server_name _;  
    server_name i6a105.p.ssafy.io;  
  
    proxy_set_header X-Forwarded-Proto $scheme;  
  
    location / {  
        root /var/lib/jenkins/workspace/Sub2_PJT/frontend/dist;  
        index index.html;  
        try_files $uri $uri/ index.html /index.html;  
    }  
    location /api{  
        allow all;  
        proxy_pass https://localhost:8080;  
        proxy_redirect off;  
        charset utf-8;  
  
        proxy_set_header Host $host;  
        proxy_set_header X-Real-IP $remote_addr;  
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
        proxy_set_header X-Forwarded-Host $server_name;  
    }  
    location /openvidu{  
        proxy_pass http://localhost:5443;  
        proxy_http_version 1.1;
```

```

proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "upgrade";
proxy_set_header Host $host;
}

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/i6a105.p.ssafy.io/fullchain.pem; # managed by
Certbot
    ssl_certificate_key /etc/letsencrypt/live/i6a105.p.ssafy.io/privkey.pem; # managed
by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

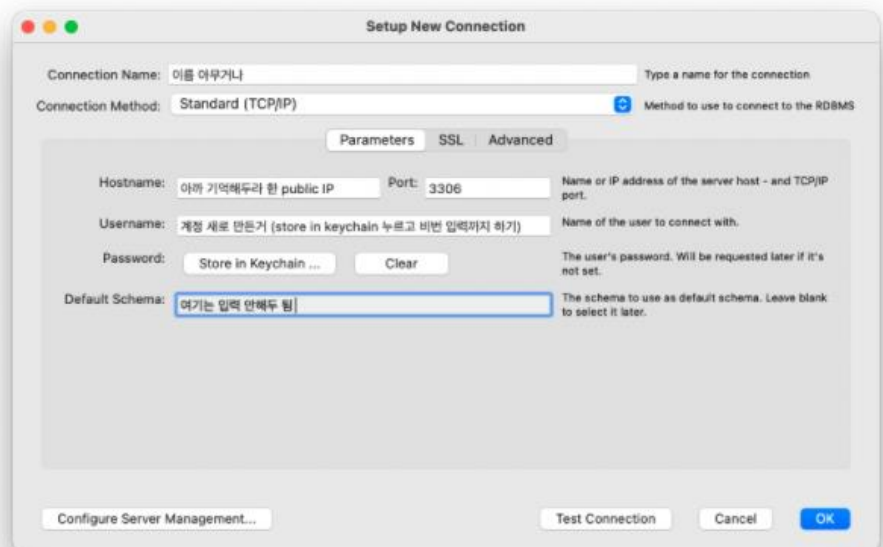
}
server {
    if ($host = i6a105.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80 default_server;
    listen [::]:80 default_server;
    server_name i6a105.p.ssafy.io;
    return 404; # managed by Certbot
}

```

나. AWS EC2 DB 세팅

- `sudo apt update`
- `sudo apt install mysql-server`
- `sudo systemctl start mysql.service`
- `cd /etc/mysql/mysql.conf.d/`
- `sudo vi mysqld.cnf`
→ `bind-address = 0.0.0.0`으로 변경
- `mysql -u root -p`
→ root 계정으로 접속 후
→ `create user '계정이름'@'%' identified by '비밀번호';`
→ `grant all privileges on . to '계정이름'@'%' with grant option;`
→ `flush privileges;`
- `curl http://169.254.169.254/latest/meta-data/public-ipv4`
→ aws서버 접속 상태에서 위 명령어 입력 후 나오는 public IP 값 기억해두기!!!!
- mysql workbench로 가서



다. Jenkins 환경 구축

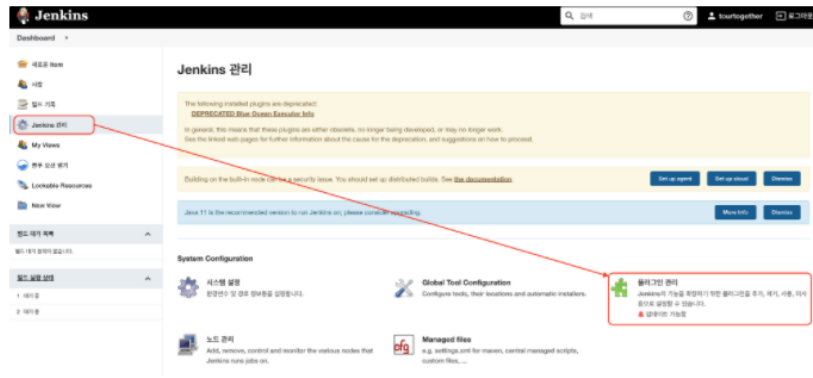
1. 기본적으로 있어야 할 것

ec2 서버에 java, mysql, npm, jenkins, nginx 설치

- <https://kitty-geno.tistory.com/25> - ec2 java 설치
- <https://mirae-kim.tistory.com/73> - ec2 mysql 설치
- <https://choseongho93.tistory.com/246> - ec2 npm 설치
- <https://withhamit.tistory.com/33> - ec2 jenkins 설치하기
- <https://msyu1207.tistory.com/entry/AWS-EC2에-NGINX-설치-및-사용하기> - ec2 nginx 설치

2. jenkins 플러그인 설치

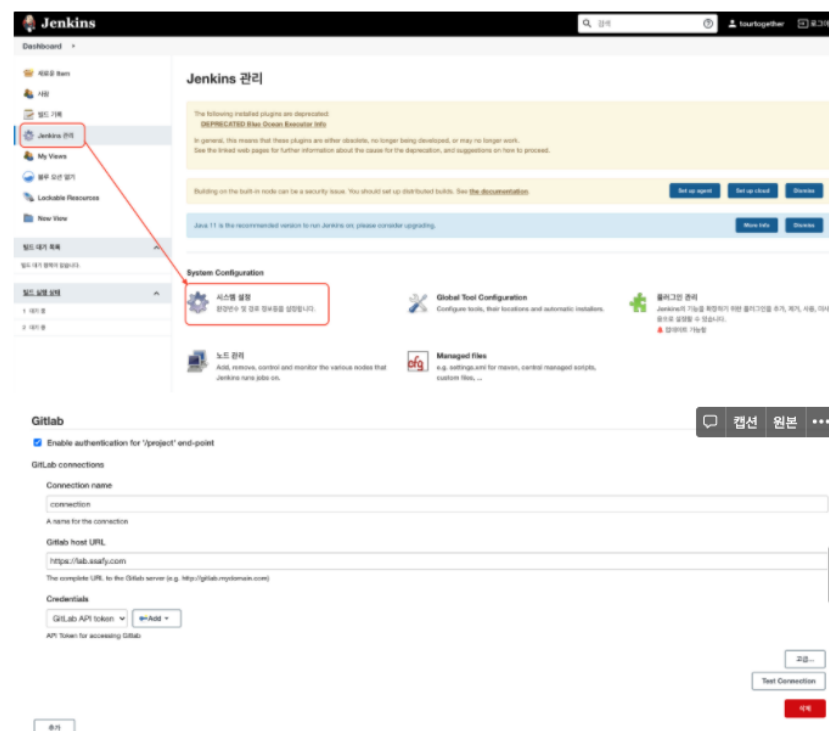
jenkins 관리 → 플러그인 관리에서 필요한 플러그인 설치 (설치가능 탭에서 Search 가능)



- Blue Ocean을 검색 후 검색된 **모든** 플러그인
- gitlab을 검색하고 검색된 **모든** 플러그인
- Node Js 플러그인

3. jenkins에 gitlab 연결

jenkins 관리 → 시스템 설정



- 시스템 설정 후 내리면 GitLab 부분 나옴
- Connection name → 원하는 아무 이름이나 작성해줌
- Gitlab host URL에 gitlab 주소를 적어줌 → <https://xxx.xxxxxx.com> 까지만 적어주면 됨

- Credentials → Add → jenkins를 눌러 새로 생성해줌

- 위 화면과 같은 화면이 나오면 Kind → GitLab API token으로 변경
- API tokens는 gitlab에서 발급받은 personal token 입력
 - personal token 발급법
 - gitlab으로 가서 User Settings → Access Tokens

- 원하는 token name과 만료기간을 정해주고 api에 체크한 다음 Create personal access token을 누름 → 생성 완료 → 화면 상단에 token 발급된거 확인 가능
- API tokens에 발급한 token 값 입력해주고
- ID → gitlab에서 사용하는 아이디 적어줌 (@뒤를 제외한 아이디만 적어주기)
- Add하면 추가 끝
- Credentials에 발급 만든걸로 설정해주면 완료 → Apply 및 저장도 해주기

4. jenkins에 프로젝트 연결 후 기본 설정하기

- dashboard → 새로운 Item
- Enter an item name에 프로젝트 이름 작성해줌 (아무거나 상관없음)
- Freestyle project 선택 후 완료

4-1. 프로젝트 git 연결하기

- 만든 프로젝트로 들어가서 구성 → 소스코드 관리 탭에서 Git을 체크

- Repository URL에 git project clone 주소 적어줌
 - Failed to connect to repository 블라블라 ERROR 나는 경우
 - 주소를 위의 사진 처럼 `https://gitlabID:PersonalAccessToken@Repository주소` 로 작성
 - PersonalAccessToken 아까 위에서 발급받은거 그거 쓰면 됨
- Credentials → Add → jenkins

- kind → SSH Username with private key
- ID → gitlab ID (@뒤 제외)
- Username → gitlab에 로그인하는 email 형식

- Private key → Enter directly → Add
 - 입력하는 곳에 ssh key를 입력해 주어야 함
 - gitlab으로 가서 User Settings → SSH keys에서 key를 입력해 주어야 함
 - 이때 입력하는 키는 배포할 서버에서 생성 가능
 - 서버 접속 후
 - `ssh-keygen -t rsa -C "gitlabID(이메일형식)"`
 - `cat /home/ubuntu/.ssh/id_rsa.pub` 열고 (기본 위치가 보통 /home/ubuntu이므로 cat .ssh/id_rsa.pub만 해도 됨) 출력된 key 값 전체 복사
 - gitlab으로 돌아와서 ssh key 입력해 주고 타이틀(아무거나 작성해도 상관 없음)이랑 유효기간 설정해주면 키 생성 완료 됨 → 이 key를 이제 jenkins에 입력
- Branches to build에 배포될 브랜치를 정해줌

4-2. 프로젝트 빌드 기본설정

- 구성 → 빌드 유발
- Build when a change is pushed to GitLab~~~ 체크 → 고급 → 맨 아래 generate 클릭
- generate까지 생성되면 key가 하나 생성되는데 복사해서 잘 가지고 있기!
- jenkins 설정 apply → 저장
- gitlab으로 가서 배포하고자 하는 프로젝트에서 Settings → Webhooks

- URL에 jenkins 프로젝트 url을 입력해줌 (빌드유발 탭에서 확인 가능)

빌드 유발

- ☐ 빌드를 원격으로 유발 (예: 스크립트 사용)
 - ☐ Build after other projects are built
 - ☐ Build periodically
 - ☒ Build when a change is pushed to GitLab. GitLab webhook URL:
http://i6a105.p.ssafy.io:8080/project/Sub2_PJT
- Enabled GitLab triggers
- ☒ Push Events

- 그리고 secret token에다가 아까 복사해두라고 한 key를 입력 후 push events에는 푸시가 일어날 브랜치를 입력해줌
- 이제 아래의 Add webhook을 클릭
- hook이 생성되면 테스트에서 push events 설정하고 test 해줌
 - 상단에 Hook executed successfully : HTTP 200 뜨면 성공

☐ Feature flag events
URL is triggered when a feature flag is turned on or off

☐ Releases events
URL is triggered when a release is created or updated

SSL verification
☒ Enable SSL verification

Add webhook

Project Hooks (1)

Hook Name	URL	SSL Verification	Actions
Push Events	http://i6a105.p.ssafy.io:8080/project/Sub2_PJT	enabled	Test Edit Delete

Push events
Tag push events
Issues events
Confidential issues events
Note events
Confidential note events
Merge requests events
Job events
Pipeline events

- jenkins로 가면 test build가 진행중이고 성공 여부를 확인 할 수 있음 (이때 실패하면 위에 설정 다 제대로 따라했는지 확인해보기)

#23

2022. 1. 31 오전 10:57

Started by GitLab push by 이민정

5. 생성한 프로젝트 빌드와 배포 설정해주기

General	소스 코드 관리	빌드 유발	빌드 환경	Build	빌드 후 조치
---------	----------	-------	-------	-------	---------

- 구성 → Build
- Execute shell에 명령어 입력
 - 아래 박스 안에 있는 것들이 명령어임, 무작정 넣고 실행하지 말고 그 밑에 설명부터 따라하기!!

```
cd frontend
npm install
npm run build
cd ../backend/tourtogether
chmod +x gradlew
./gradlew clean
./gradlew build
nohup java -jar /var/lib/jenkins/workspace/Sub2_PJT/backend/tourtogether/bui
```

- vue 빌드 → run build 명령어 실행 시 dist 폴더가 생성됨
- ec2 서버에 접속해 cd /etc/nginx/sites-available
 - 첫번째 네모에 front 빌드하면서 생기는 dist 폴더 경로 입력해줌
 - 두번째 네모부분도 입력해줌 (라우터 이동할 때, 새로고침할때 404에러가 뜨는걸 막아줌)

```

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

    # SSL configuration
    #
    # listen 443 ssl default_server;
    # listen [::]:443 ssl default_server;
    #
    # Note: You should disable gzip for SSL traffic.
    # See: https://bugs.debian.org/773332
    #
    # Read up on ssl_ciphers to ensure a secure configuration.
    # See: https://bugs.debian.org/765782
    #
    # Self signed certs generated by the ssl-cert package
    # Don't use them in a production server!
    #
    # include snippets/snakeoil.conf;

    root /var/lib/jenkins/workspace/Sub2_PJT/frontend/dist;

    # Add index.php to the list if you are using PHP
    index index.html index.htm index.nginx-debian.html;

    server_name _;

    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ /index.html;
    }
}

```

- 다음은 springboot 배포 → springboot 프로젝트 폴더 안에 gradlew 파일이 있는지 확인
- 없다면 프로젝트 폴더로 들어가서 `gradle init` → `gradle wrapper` 실행
(gradle command not found ERROR가 나면 gradle 설치부터 하고 위의 명령어 재 실행)
- 여기까지 하면 backend 프로젝트 폴더 내에 gradlew 파일이 생성됨
- 이제 저장하고 Build Now를 클릭하면 build가 생성되고 자동으로 배포 진행됨

6. 외부 서비스

가. 카카오 : 서비스 회원가입/로그인을 카카오프로 진행하였습니다. 회원가입/로그인의 다양한 절차를 생략할 수 있어서 이용자의 편의성을 제공합니다.

1) 어플리케이션 추가 후 도메인 등록

Web

사이트 도메인	http://localhost:8080 https://localhost:8080 http://i6a105.p.ssafy.io https://i6a105.p.ssafy.io http://localhost:8081
---------	---

- 카카오 로그인 사용 시 Redirect URI를 등록해야 합니다. [등록하러 가기](#)

2) Redirect URI 설정

Redirect URI

Redirect URI	http://localhost:8080/kakao-login-callback https://localhost:8080/kakao-login-callback/ http://i6a105.p.ssafy.io/kakao-login-callback/ https://i6a105.p.ssafy.io/kakao-login-callback/ http://localhost:8081/kakao-login-callback/ http://localhost:8080/dashboard
--------------	---

- 카카오 로그인에서 사용할 OAuth Redirect URI를 설정합니다. (최대 10개)
- REST API로 개발하는 경우 필수로 설정해야 합니다.

3) 로그인 활성화 후 카카오 인가 코드 수신

앱 키

네이티브 앱 키	1f6e2a4e4a0e8588d26747c7a4ec7a35
REST API 키	a8b7ed9387ad25646ee6577925ce17db
JavaScript 키	63678d67fd1c12adc5465cd1f06e3d6a
Admin 키	f0d1d59398b540f547ccf0abbdaeeac6

4) Kakao accessToken 수신 후 accessToken으로 정보 요청

나. AWS S3 : 서비스 내의 사진을 저장하여, 관련 url을 사용할 수 있는 클라우드입니다.

1) 버킷 만들기

