



How to Deploy



Use this template to describe the steps engineers should follow to deploy.

1. Local에서 Test
2. Server에서 Test
3. Nginx Deploy
 - NGINX TLS 인증서 적용
4. 백엔드 HA구성
5. Jenkins CI/CD 구성
 - 5.1 (사전작업) GitLab에서 배포토큰 생성
 - 5.2 (사전작업) SSAFY GIT에 배포 토큰 등록 및 Jenkins Item(project) 생성
 - 5.3 (사전작업) Jenkins → AWS 운영서버 접속정보 설정
 - 5.4 Build Stage
 - 5.5 Deploy Stage
 - 5.6 빌드 및 배포 실행 및 확인
6. GitLab CI/CD 구성
 - 6.1 (사전작업) GitLab Runner 설치
 - 6.2 (사전작업) GitLab Runner register(등록)
 - 6.3 (사전작업) GitLab Runner install(설치)
 - 6.4 Build & Deploy Stage 구성
 - 6.5 빌드 및 배포 확인
 - 6.6 프로젝트 배지 달기

1. Local에서 Test

!! 처음 배포할때 로컬에서 테스트 없이 작업하면 많은 시간을 낭비할 수 있음

```
# 소스 받기
# https://lab.ssafy.com/hibuz/s05-webmobile2-sub1.git
git clone https://lab.ssafy.com/s05-webmobile2-sub1/skeleton-project.git
```

yarn test를 위해서는  [Vue.js 단위 테스트](#) 설정 필요

```
# Front
cd frontend
yarn install
# yarn test
yarn serve
```

프론트 테스트: <http://localhost:8080/>

```
# Docker DB 실행
docker run --name mariadb -p 3306:3306 -e MYSQL_ROOT_PASSWORD=mariadb -e MYSQL_DATABASE=ssafy-sk -d mariadb --character-set-server=utf8
# backend
cd backend

# Linux
./mvnw spring-boot:run
# Window
mvnw spring-boot:run
```

Back 테스트: <http://localhost:8080/swagger-ui.html>

2. Server에서 Test

```
# 서버접속
ssh -i cert.pem ubuntu@j5b20x.p.ssafy.io




git clone https://lab.ssafy.com/hibuz/s05-webmobile2-sub1.git skeleton-project

# 프론트 테스트 : http://j5b20x.p.ssafy.io:8080
cd skeleton-project/frontend
yarn install
yarn serve

# Docker DB 실행
docker run --name mariadb -p 3306:3306 -e MYSQL_ROOT_PASSWORD=mariadb -e MYSQL_DATABASE=ssafy-sk -d mariadb --character-set-server=utf8

# 백엔드 테스트: http://j5b20x.p.ssafy.io:8080/swagger-ui.html
cd skeleton-project/backend
./mvnw spring-boot:run
```

▼ 개발 환경 준비

- Docker DB 실행:  [DB](#)
- NVM 설치:  [NodeJS](#)
- SDKMan Java 설치:  [Java](#)

▼ Invalid Host header에러 발생시: skeleton-project/frontend/vue.config.js 추가

```
module.exports = {
  configureWebpack: {
    // other webpack options to merge in ...
  },
  // devServer Options don't belong into `configureWebpack`
  devServer: { host: "0.0.0.0", hot: true, disableHostCheck: true },
};
```

3. Nginx Deploy

프론트엔드 Vue 빌드

```
# 프론트엔드 빌드
yarn build
```

Nginx 설치후 다음 위치에 있는 nginx 설정을 vi 에디터로 수정 (관리자 권한 필요)

```
sudo apt install nginx
sudo vi /etc/nginx/sites-enabled/default
```

웹서버 루트 기존 설정은 주석처리하고 프론트 빌드한 위치로 변경

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

    #root /var/www/html;
    root /home/ubuntu/skeleton-project/frontend/dist;
    ...
}

# 설정 변경 후 syntax 검사 필수
sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

설정을 바꾸면 Nginx 재시작

```
sudo service nginx restart
#or
sudo systemctl restart nginx
#상태확인: Active: active (running)
#sudo systemctl status nginx
```

Nginx 설치: [Nginx](#)

백엔드 Spring Boot 빌드

```
# 백엔드 빌드
./mvnw package

# proxy로 백엔드 테스트: http://j5b20x.p.ssafy.io/api/swagger-ui.html
# 보안상 외부에서 8080 접근 불가하게 실행하려면 --server.address=127.0.0.1 옵션추가
java -jar target/*.jar --server.servlet.context-path=/api
```



터미널에서 백엔드 실행후 빠져나오면 서비스가 종료되므로 백그라운드로 실행해야 합니다. 간단히 명령어 끝에 &를 붙이거나 nohup을 사용하세요
ex) java -jar target/*.jar &

reverse proxy nginx setting을 추가

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

    ##### 여기부터 백엔드 리버스 프록시로 설정 추가
    location /api {
        proxy_pass http://localhost:8080;
        proxy_http_version 1.1;
        proxy_set_header Connection "";

        include /etc/nginx/proxy_params;
    }
    ##### 여기까지
}
```

백엔드 재시작 스크립트 작성

!! pid 파일이 생성되도록 ApplicationPidFileWriter 추가 (서버 실행시 자동 생성되는 application.pid 파일이 커밋되지 않도록 .gitignore에 등록)

```
import org.springframework.boot.context.ApplicationPidFileWriter;

@SpringBootApplication
public class WebCurationApplication {

    public static void main(String[] args) {
        SpringApplication app = new SpringApplication(WebCurationApplication.class);
        app.addListeners(new ApplicationPidFileWriter()); // pid 파일을 생성하는 writer 등록
        app.run(args);
    }
}
```

```
# 운영 서버에서 Backend 재시작 스크립트 생성 및 실행 권한 추가
vi ~/restart_backend_one.sh
```

restart_backend_one.sh

```
#!/bin/bash

# SDKMAN only for java
#source "/home/ubuntu/.sdkman/bin/sdkman-init.sh"

kill $(cat app1.pid)

nohup java -jar app1.jar --server.servlet.context-path=/api --server.address=127.0.0.1 --server.port=8080 \
--spring.pid.file=app1.pid >> app1.log 2>&1 &

# pid 생성 기다리기
sleep 2
echo "complete deploy app1 pid=$(cat app1.pid)"
```

```
# 실행권한 추가 후 테스트
chmod +x ~/restart_backend_one.sh

~/restart_backend_one.sh
```

NGINX TLS 인증서 적용

Docker로 인증서 발급

```
# 팀별 도메인 변경 후 실행
docker run -it --rm --name cert_tmp -p 80:80 -v /home/ubuntu/cert:/etc/letsencrypt certbot/certbot certonly \
--standalone -d j5b20x.p.ssafy.io -m hibuz@hibuz.com
```

https 설정 추가

```
# / 요청을 영구적으로 https로 redirect 처리하는 설정 추가
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    location / {
        return 301 https://$host$request_uri;
    }
}

### 기존 설정을 https(443)용으로 설정하고 TLS 설정을 새롭게 추가
server {
    # 80은 주석 처리
    #listen 80 default_server;
    #listen [::]:80 default_server;

    # 443은 주석 해제
    listen 443 ssl default_server;
    listen [::]:443 ssl default_server;

    ##### 여기부터 발급받은 인증서 위치 및 TLS 설정 추가 (팀별 도메인 변경필요)
    ssl_certificate /home/ubuntu/cert/live/j5b20x.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /home/ubuntu/cert/live/j5b20x.p.ssafy.io/privkey.pem;
    ssl_session_cache shared:le_nginx_SSL:10m;
    ssl_session_timeout 1440m;
    ssl_session_tickets off;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_prefer_server_ciphers off;
    ssl_ciphers "ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384";
    ##### 여기까지

    ...
}
```

4. 백엔드 HA구성

```
# 두번째 서버 실행
java -jar app1.jar \
--server.servlet.context-path=/api \
--server.address=127.0.0.1 \
--server.port=8081 \
--spring.pid.file=app2.pid
```

설정 상단에 reverse proxy 설정 및 upstream 추가 및 proxy_pass 수정

```
# 백엔드 설정 추가
upstream backend {
    server localhost:8080;
    server localhost:8081;
}

server {
    ...
    location /api {
        ##### 이부분 upstream으로 변경
        #proxy_pass http://localhost:8080;
        proxy_pass http://backend;
        ...
    }
}
```

```
# 기존 스크립트 복사
cp ~/restart_backend_one.sh ~/restart_backend_ha.sh

# 무중단 배포 스크립트 설정
vi ~/restart_backend_ha.sh
```

restart_backend_ha.sh

```
#!/bin/bash

# SDKMAN only for java
#source "/home/ubuntu/.sdkman/bin/sdkman-init.sh"

kill $(cat app1.pid)

nohup java -jar app1.jar --server.servlet.context-path=/api --server.address=127.0.0.1 --server.port=8080 \
--spring.pid.file=app1.pid >> app1.log 2>&1 &

# 서버가 완전히 시작할때 까지 대기
sleep 2
echo "complate deploy app1 pid=$(cat app1.pid)"

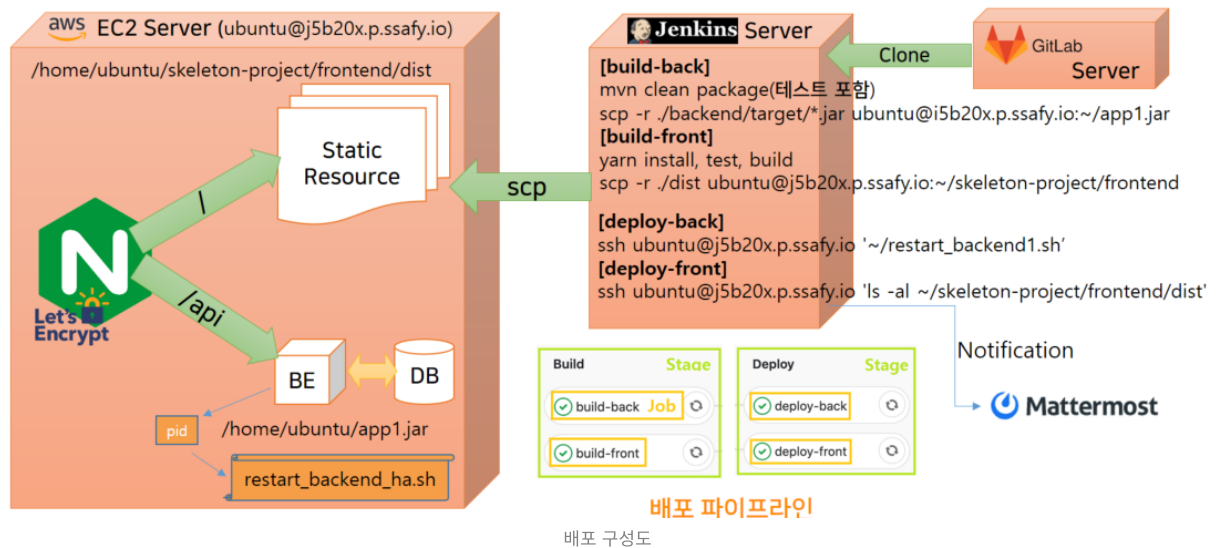
##### 여기서 부터 추가
# 두번째 서버 내리고 jar 복사
kill $(cat app2.pid)
cp app2.jar app2.jar

nohup java -jar app2.jar --server.servlet.context-path=/api --server.address=127.0.0.1 --server.port=8081 \
--spring.pid.file=app2.pid >> app2.log 2>&1 &

# pid 생성 기다리기
sleep 2
echo "complate deploy app2 pid=$(cat app2.pid)"
```

5. Jenkins CI/CD 구성

배포자동화는 Jenkins, Gitlab 중에 선택적으로 적용해 봅니다.



5.1 (사전작업) GitLab에서 배포토큰 생성

GitLab > 설정 > 저장소 > 배포토큰: 이름: SSAFY_GIT(아무거나), 기한설정, read_repository 체크

배포 토큰

Deploy tokens allow read-only access to your repository and registry images.

배포 토큰 추가

Pick a name for the application, and we'll give you a unique deploy token.

이름

SSAFY_GIT

Expires at

2020-10-31

Scopes

☒ read_repository

Allows read-only access to the repository

배포 토큰 만들기

5.2 (사전작업) SSAFY GIT에 배포 토큰 등록 및 Jenkins Item(project) 생성

생성된 사용자 이름과 암호 토큰을 SSAFY GIT > 프로젝트 개발 > 팀정보 > GitLab 수정 버튼을 클릭하여 설정

GITLAB 수정

Gitlab

https://lab.ssafy.com/bootcamp-test/s3p1162001

Deploy Info

UserName

gitlab+deploy-token-478

:

Token

입력해 주세요.

GitLab 프로젝트의 Settings에서 생성한 Deploy Token 정보 입력

Deploytoken 생성 방법보기

취소

저장

프로젝트 설정 정보

JIRA	<input type="text"/>	<input type="button" value="설정"/>
Gitlab	<input type="text"/>	<input type="button" value="설정"/>
Jenkins	Deploy Info User Name <input type="text"/> : Token <input type="text"/>	<input type="button" value="Jenkins Item 생성"/> <input type="button" value="설정"/>
AWS		<input type="button" value="설정"/>

Jenkins 에서 Git 설정 확인 후 Checkout 받아서 빌드할 브랜치 설정

소스 코드 관리

☐ None
☒ Git

Repositories

Repository URL

Credentials

Branches to build

Branch Specifier (blank for 'any')

5.3 (사전작업) Jenkins → AWS 운영서버 접속정보 설정

!! authorized_keys 파일 수정시 기존에 등록된 cert.pem 의 공개키 정보가 잘못되면 cert.pem 파일로 접속이 불가하므로 주의!!

```
# AWS 운영서버 접속
ssh -i cert.pem ubuntu@j5b20x.p.ssafty.io

# 파일에 아래 jenkins 서버 계정 공개키를 추가하기
vi ~/.ssh/authorized_keys
```

ssh-rsa

AAAAB3NzaC1yc2EAAAADAQABAAQDMUDx1HbfoH2ftJquO59glzLnUwP185amhrq+jdxO0829/NyFfhbafm7NioJ4vQC0cjenkins@prod-yswa-cicd-kr-vm1

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDMUDx1HbfoH2ftJquO59glzLnUwP185amhrq+jdxO0829/NyFfhbafm7NioJ4vQC0c5Pg47RNLgA2vYUnEdSyHUY02Ntwl
dY6rHFRgomf87ZnCFzqWfHmQ1dshDxzYSwubLN7LxERqsdPhAtmzd+DxhMa/NNDC4zkz297nMB5zxH3xFEYwUm+wb58r7cxz7kBNAS5SRXZGamOPYW4urQV005aofGa0jum9E
GQfui14pMLqIyS9uEBCv/Gz9D8mdag2SIMnZU6uALp3dHSpabAKqRQ4FLN6gHvNGIdPbm+dL/uBL/8mDp8Jdk7yKkZku3XTLQeM40Wjrji8cSLmWIXIj_jenkins@prod-ys
wa-cicd-kr-vm1
```

5.4 Build Stage

백엔드: Build > Add build step > **Invoke top-level Maven targets** 추가 후 Maven 빌드 설정

고급... 버튼을 클릭해서 pom.xml 위치도 수정

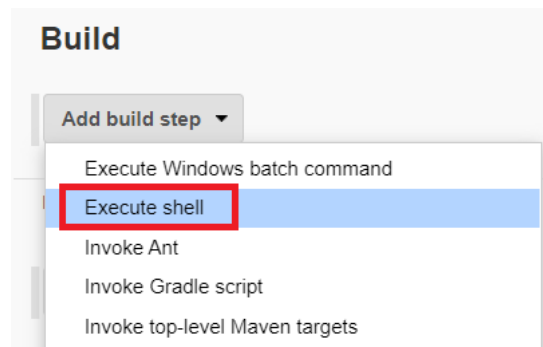
Invoke top-level Maven targets

Maven Version:

Goals:

POM:

프론트엔드: Build > Add build step > **Execute shell** 추가 후 테스트 및 빌드 스크립트 설정



```
##### Backend 배포 서버로 jar 복사 #####
scp ./backend/target/*.jar ubuntu@j5b20x.p.ssafy.io:~/app1.jar

##### Frontend Test & Build #####
cd frontend
yarn install
yarn test
yarn build

# 팀별로 /etc/nginx/sites-enabled/default 에 설정된 배포 파일을 삭제 후 복사 복사
ssh ubuntu@j5b20x.p.ssafy.io 'rm -rf ~/skeleton-project/frontend/dist'
scp -r ./dist ubuntu@j5b20x.p.ssafy.io:/home/ubuntu/skeleton-project/frontend
```

5.5 Deploy Stage

Build > Add build step > **Execute shell** 추가 후 Backend 배포 및 알림 스크립트 설정

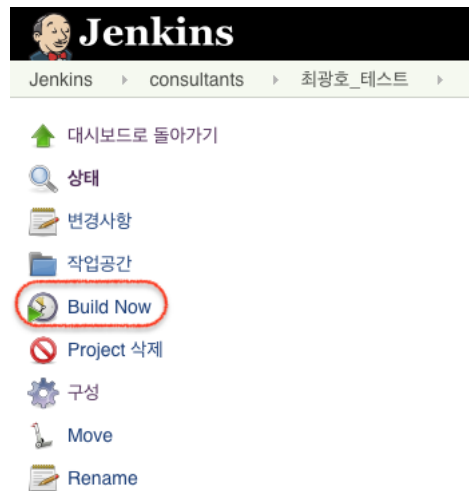
```
##### Backend Deploy #####
# backend1 재시작
ssh ubuntu@j5b20x.p.ssafy.io '~/restart_backend_ha.sh'

##### Frontend Deploy #####
# nginx 재시작이 불필요하므로 단순히 복사여부만 확인
ssh ubuntu@j5b20x.p.ssafy.io 'ls -al ~/skeleton-project/frontend/dist'
```

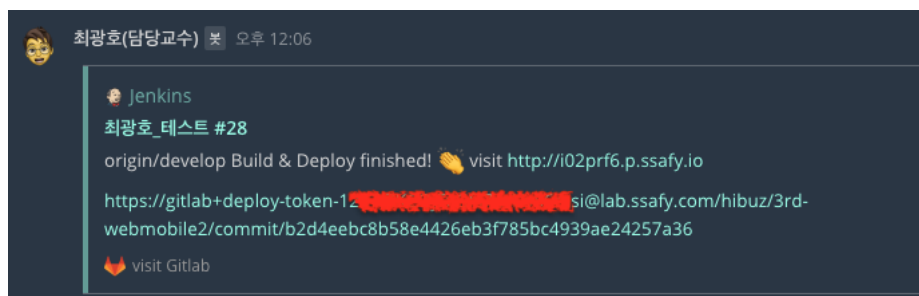
5.6 빌드 및 배포 실행 및 확인

홈	빌드 현황	산출물	팀 정보
<div> <div>빌드 현황</div> <div> <div>설정</div> <div>빌드</div> </div> </div>			
마지막 빌드	마지막 성공 빌드	마지막 실패 빌드	마지막 변경

SSAFY GIT에서 빌드 클릭



Jenkins 에서 직접 **Build Now** 클릭



빌드 후 매터모스트 알림 확인

▼ Build > Add build step > **Execute shell** 추가 후 Mattermost Notification스크립트 설정

```
##### Mattermost Notification 설정 #####
#See https://www.bluexml.com/2019/06/12/bot-mattermost-via-curl-depuis-jenkins/
COMMIT_MSG=$(git log --oneline --format=%B -n 1 HEAD | head -n 1)
REQUEST="curl -i \
-X POST \
-H 'Content-Type: application/json' \
-d '{ \
  \"icon_url\": \"https://www.mattermost.org/wp-content/uploads/2016/04/icon.png\", \
  \"attachments\": [{ \
    \"text\": \"\${GIT_BRANCH} Build & Deploy finished! :clap: visit http://i5c10x.p.ssafy.io\", \
    \"author_name\": \"Jenkins\", \
    \"author_icon\": \"http://jenkins-1.ssafy.com/favicon.ico\", \
    \"author_link\": \"\${BUILD_URL}\", \
    \"title\": \"\${JOB_NAME} \${BUILD_DISPLAY_NAME} - \${COMMIT_MSG}\", \
    \"title_link\": \"\${BUILD_URL}/console\", \
    \"fields\": [{ \
      \"value\": \"\${GIT_URL::-4}/commit/\${GIT_COMMIT}\" \
    }], \
    \"footer\": \"visit Gitlab\", \
    \"footer_icon\": \"https://lab.ssafy.com/slash-command-logo.png\" \
  }] \
}' \
https://meeting.ssafy.com/hooks/{각자 생성한 webhook 정보입력}\"

eval $REQUEST
```

6. GitLab CI/CD 구성

Please enter the executor: custom, parallels, shell, ssh, virtualbox, docker+machine, docker-ssh+machine, kubernetes, docker, docker-ssh, shell
Runner registered successfully. Feel free to start it, but if it's running already the config should be automatically reloaded!

6.3 (사전작업) GitLab Runner install(설치)

```
# Runner용 작업 디렉토리 생성
mkdir /home/ubuntu/.gitlab-runner

# Runner 삭제 후 ubuntu 사용자로 재설치
sudo gitlab-runner stop
sudo gitlab-runner uninstall
sudo gitlab-runner install --working-directory /home/ubuntu/.gitlab-runner --config /etc/gitlab-runner/config.toml \
--syslog --user ubuntu

# Runner 실행
sudo gitlab-runner start
```

▼ Runner 실행여부 확인법 2가지

```
sudo gitlab-runner status
Runtime platform                                arch=amd64 os=linux pid=13525 revision=a998cacd version=13.2.2
gitlab-runner: Service is running!
```

```
sudo ps -ef |grep runner
root      4002      1  0 15:18 ?        00:00:00 /usr/lib/gitlab-runner/gitlab-runner run
--working-directory /home/ubuntu/.gitlab-runner --config /etc/gitlab-runner/config.toml
--service gitlab-runner --syslog --user ubuntu
```

▼ Runner 재등록 방법

Gitlab 설정 > CI / CD > Runners > Runner 제거

runner 등록 토큰 초기화 실행 후 새로운 토큰 정보로 6.2, 6.3 재수행

Runners activated for this project

 L71Fbg4K  

중지 Runner 제거

aws production

#78

prd

Runner 활성화 확인




6.4 Build & Deploy Stage 구성

빌드 결과 확인을 위해 MM 웹훅 연동설정 확인

프로젝트 최상위 폴더에 아래와 같이 .gitlab-ci.yml 파일 생성

 Kubernetes 클러스터 추가

 CI/CD 구성

이름	최근 커밋	최근 업데이트
 backend	Add PidFileWriter	1일 전
 frontend	Add header test	1주 전
 .gitlab-ci.yml	Update .gitlab-ci.yml	25 분 전

```
stages:
  - build
  - deploy

build-back:
  stage: build
  only :
```

```

- master
- develop
script :
# Java를 sdkman 으로 설치한 경우 JAVA_HOME 설정을 위해서만 필요
- source "/home/ubuntu/.sdkman/bin/sdkman-init.sh"
- cd $CI_PROJECT_DIR/backend
- chmod +x mvnw
- ./mvnw clean package
- cp target/*.jar ~/app1.jar
tags :
# !! tag 값이 동일하게 설정된 Runner만 파이프라인을 동작시킴
# 여러 서버에 배포할때 서버별 runner 태그로도 활용(dev, stg, prd)
- prd

build-front:
stage: build
cache:
key: ${CI_COMMIT_REF_SLUG}
paths:
- frontend/node_modules
only :
- master
- develop
script :
# Node.js를 nvm 으로 설치한 경우만 필요
- source "/home/ubuntu/.nvm/nvm.sh"
- cd $CI_PROJECT_DIR/frontend
- yarn install
- yarn test
- yarn build
- rm -rf ~/skeleton-project/frontend/dist
- cp -rf dist ~/skeleton-project/frontend
tags :
- prd

deploy-back:
stage: deploy
only :
- master
- develop
script :
# 백엔드 서버 재시작
- /home/ubuntu/restart_backend_ha.sh
tags :
- prd

deploy-front:
stage: deploy
only :
- master
- develop
script :
# 배포는 이미 끝나서 단순 확인하기 위한 job으로만 구성
- ls -al ~/skeleton-project/frontend/dist
tags :
- prd

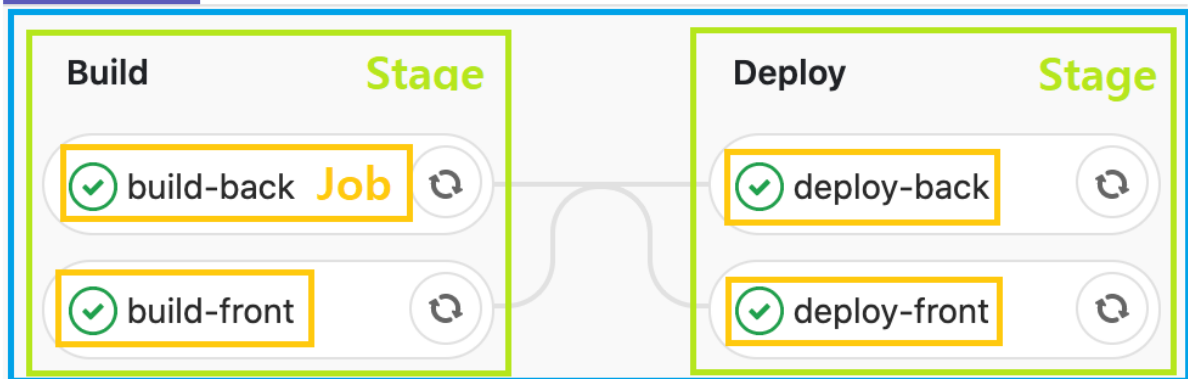
```

6.5 빌드 및 배포 확인

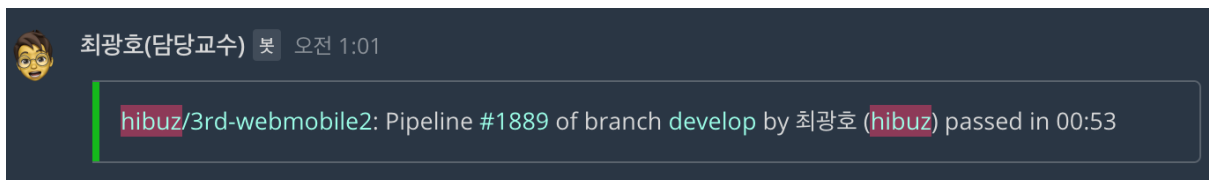
<div> <div>전체 12</div> <div>대기중 0</div> <div>실행중 0</div> <div>완료 12</div> <div>브랜치</div> <div>태그</div> </div> <div>파이프라인 실행</div>				
Status	Pipeline	Commit	Stages	
<div> <div>성공</div> <div>latest</div> </div>	<div>#1889 by </div>	<div> <div>deveLop</div> <div>427d7c8a</div> <div>Update .gitlab-ci.yml</div> </div>	<div> <div>✓</div> <div>✓</div> </div>	<div> <div>00:00:53</div> <div>1 minute ago</div> </div>

파이프라인

Jobs 4



파이프라인 구조



빌드 후 매터모스트 알림 확인

6.6 프로젝트 배지 달기

GitLab 설정 > CI/CD > General pipelines 에 있는 내용을 참고하여 GitLab 설정 > 일반 > 배지에 설정

배지

프로젝트 배지 커스텀마이징 배지에 대해 자세히 알아보세요.

링크

GitLab이 지원하는 변수: `%(project_path)`, `%(project_id)`, `%(default_branch)`, `%(commit_sha)`

`https://lab.ssafy.com/hibuz/3rd-webmobile2/commits/develop`

e.g. `https://example.gitlab.com/%(project_path)`

배지 이미지 URL

GitLab이 지원하는 변수: `%(project_path)`, `%(project_id)`, `%(default_branch)`, `%(commit_sha)`

`https://lab.ssafy.com/hibuz/3rd-webmobile2/badges/develop/pipeline.svg`

e.g. `https://example.gitlab.com/%(project_path)/badges/%(default_branch)/badge.svg`

배지 이미지 미리 보기

pipeline passed

변경 사항 저장

프로젝트 배지 설정 화면

Apache License 2.0 31 Commit 6 Branch 0 Tag 891킬로바이트 Files

pipeline running

develop 3rd-webmobile2 / 이력 파일 찾기 웹 IDE

LICENSE 추가
28초 전 에 최광호 님이 커밋하였습니다.

f2965eaf

- Add README
- Add CHANGELOG
- Add CONTRIBUTING
- Kubernetes 클러스터 추가
- CI/CD 구성

이름	최근 커밋	최근 업데이트
backend	Add PidFileWriter	1일 전
frontend	Update Login.vue	11 분 전
.gitlab-ci.yml	Update .gitlab-ci.yml	17 분 전
LICENSE	LICENSE 추가	28초 전

벤티가 적용된 화면