포팅 메뉴얼

I. 기술 스택 & 개발 환경

1. 사용 도구

💡 **사용도구**

- 이슈 관리: JIRA
- 형상 관리: GitLab
- 커뮤니케이션: Mattermost, Notion, Google Docs, Kakao Talk
- 디자인: Figma
- UCC: Movavi
- CI/CD: EC2, Docker, Jenkins

2. 개발도구

💡 **개발도구**

- Visual Studio Code
- IntelliJ IDEA: 2024.1.4 u
- JDK: java 17.0.11 2024-04-16 LTS
-

3. 개발환경

3-1. 개발환경 및 외부서비스(API)

개발환경 및 외부서비스

- Front-end
  - Node.js: 18.20
  - React: 18.2.0
  - Typescript: 4.9.5
  - ESLint: 8.57.0
  - typescript-eslint: 7.16.0
- Back-end
  - JDK: 17.0.10 LTS
  - SpringBoot: 3.2.3
  - SpringSecurity
  - jjwt-api:0.11.5
  - Gradle: 8.5
  - Spring Jpa 3.3
- DB
  - Mysql: 8.0
  - Redis
- Infra
  - AWS EC2: Ubuntu 20.04.6 LTS
  - Docker: 25.0.4
  - Jenkins: 2.448
  - Nginx: 1.18.0 (Ubuntu)
  - Elasticsearch 7.6
  - Kibana 7.6
  - Logstash 9.6
  - Kafka 3.8

- Grafana 9.52
- Prometheus

3-2. 환경변수

```
[Back 환경 변수]

## mysql
spring.datasource.url=jdbc:mysql://??/catchcatch?autoReconnec
spring.datasource.username=root
spring.datasource.password=각자의 비밀번호
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto= create
spring.jpa.properties.hibernate.format_sql=true

## jwt
jwt.secret = sjkinrjekaijglkjgljaghjladfhrerguihaeruihgnlsdzf
jwt.access-token-expire = 144000000

## sms
sms.api-key=NCSZAQEZJNWS2PJW
sms.api-secret=TBCPNJBYQLZ8XDDEDQRM2R5IPCS9NZLE
sms.sms-provider=01029463517
## osiv
spring.jpa.open-in-view=false

## redis
spring.data.redis.host=i11b101.p.ssafy.io
spring.data.redis.port=6379
```

```
# oauth kakao
spring.security.oauth2.client.registration.kakao.client-id=9d
spring.security.oauth2.client.registration.kakao.client-secre
spring.security.oauth2.client.registration.kakao.redirect-uri
spring.security.oauth2.client.registration.kakao.authorizatio
spring.security.oauth2.client.registration.kakao.client-authe
spring.security.oauth2.client.registration.kakao.client-name=
spring.security.oauth2.client.registration.kakao.scope= profi
spring.security.oauth2.client.provider.kakao.authorization-ur
spring.security.oauth2.client.provider.kakao.token-uri = http
spring.security.oauth2.client.provider.kakao.user-info-uri =
spring.security.oauth2.client.provider.kakao.user-name-attrib

# oauth google
spring.security.oauth2.client.registration.google.client-id =
spring.security.oauth2.client.registration.google.client-secr
spring.security.oauth2.client.registration.google.scope = pro
spring.security.oauth2.client.registration.google.redirect-ur

## naver cloud
naver.cloud.id=8sugutuo9v
naver.cloud.secret=H4XFAc6CG5TiJ286KrAIGFNhbkgOSp6c1uHMMk3O

## openai
openai.model=gpt-4o-mini
openai.image.model=dall-e-3
openai.api.key=sk-proj-PkRxsSCYZmJcpXyhZttFT3BlbkFJTDitk1cvuI
openai.api.url= https://api.openai.com/v1/chat/completions
openai.api.create-image-url=https://api.openai.com/v1/images/

## aws
cloud.aws.s3.bucket=everstarbucket
cloud.aws.credentials.access-key=AKIAQGYBQCQ4GYHL3YXF
cloud.aws.credentials.secret-key=wFQKM9szybNnm5a8sFwJCHaQGjRs
cloud.aws.region.static=ap-northeast-2
cloud.aws.region.auto=false
cloud.aws.stack.auto=false
```

```
## file upload
spring.servlet.multipart.max-file-size=10MB
spring.servlet.multipart.max-request-size=10MB

##openvidu
openvidu.url=https://i11b101.p.ssafy.io:4443/
openvidu.secret=pdw06135

#kafka
spring.kafka.producer.bootstrap-servers=http://i11b101.p.ssaf
spring.kafka.consumer.group-id=group_1
spring.kafka.consumer.group=group

## diffusion ai
diffusionai.model=anything-v5
diffusionai.api.key=jeOb8nKM7uO6bxToSpz37MIKwH0uPDCAB3MUTbtoX
diffusionai.api.url=https://modelslab.com/api/v6/images/img2i
```

II. CI / CD 구축

1. 기본설정

1-1. Docker 기본설정

1. Docker 설치

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

2. Docker 공식 GPG 키 추가

```
$ sudo mkdir -m 0755 -p /etc/apt/keyrings
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg
```

3. Docker Repository 설치

```
$ echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stabl
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

4. Docker 설치

```
$ sudo apt-get install docker-ce docker-ce-cli containerd.
```

5. Docker 실행 테스트

```
$ sudo docker run hello-world

# 실행된 도커 컨테이너 확인
$ sudo docker ps

# 이미지 확인
$ sudo docker images
```

6. Docker compose 설치

```
$ sudo curl -L "https://github.com/docker/compose/releases

# 다운로드한 도커 컴포즈 파일을 실행 가능하도록 다운로드한 경로에 권한을
$ sudo chmod +x /usr/local/bin/docker-compose

$ sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker

확인
$ docker-compose -v
```

III. 빌드 및 배포

0. EC2 포트번호

- 80 http
- 443 ssl
- 22 ssh
- 4443 openvidu
- 5061 kibana
- 4000 grafana

1. Local Build

💡 **[Front] 개발 환경에서 직접 빌드**

1. 의존성 설치
   npm install
2. 프로젝트 빌드 (정적 파일 생성)
   npm run build
3. docker 이미지 build

   docker build -t 0/0
4. docker 허브 이미지 올리기

   docker push 0/0

💡 **[Back] 개발 환경에서 직접 빌드**

1. 프로젝트 빌드

   ./gradlew clean build

2. docker 이미지 build

   docker build -t 0/0

3. docker 허브에 이미지 올리기

docker push 0/0

2. Deployment

💡 **배포 시 빌드(jenkins 파이프라인)**

▼ **[Back - server] Jenkins 파이프라인**

```
pipeline {
    agent any

    environment {
        DB_URL = 'jdbc:mysql://j11b106.p.ssafy.io:3306/cat
        DB_USERNAME = 'root' // 자격 증명에서 사용자 이름을 가져
        DB_PASSWORD = 'catchcatch' // 자격 증명에서 비밀번호를
        JWT_SECRET = 'F6FC2FB8246B8C645453F1144A77F64A4411
        JWT_ACCESS_TOKEN_EXPIRE = '60480000000'
        JWT_REFRESH_TOKEN_EXPIRE = '60480000000'

        SPRING_DATA_REDIS_HOST = 'j11b106.p.ssafy.io'
```

```
        SPRING_DATA_REDIS_PORT = '6379'

        SPRING_SECURITY_OAUTH2_CLIENT_REGISTRATION_KAKAO_C
        SPRING_SECURITY_OAUTH2_CLIENT_REGISTRATION_KAKAO_C
        SPRING_SECURITY_OAUTH2_CLIENT_REGISTRATION_KAKAO_R
        SPRING_SECURITY_OAUTH2_CLIENT_REGISTRATION_KAKAO_A
        SPRING_SECURITY_OAUTH2_CLIENT_REGISTRATION_KAKAO_C
        SPRING_SECURITY_OAUTH2_CLIENT_REGISTRATION_KAKAO_C
        SPRING_SECURITY_OAUTH2_CLIENT_REGISTRATION_KAKAO_S
        SPRING_SECURITY_OAUTH2_CLIENT_PROVIDER_KAKAO_AUTHO
        SPRING_SECURITY_OAUTH2_CLIENT_PROVIDER_KAKAO_TOKEN_
        SPRING_SECURITY_OAUTH2_CLIENT_PROVIDER_KAKAO_USER_
        SPRING_SECURITY_OAUTH2_CLIENT_PROVIDER_KAKAO_USER_

        SPRING_SECURITY_OAUTH2_CLIENT_REGISTRATION_GOOGLE_
        SPRING_SECURITY_OAUTH2_CLIENT_REGISTRATION_GOOGLE_
        SPRING_SECURITY_OAUTH2_CLIENT_REGISTRATION_GOOGLE_S
        SPRING_SECURITY_OAUTH2_CLIENT_REGISTRATION_GOOGLE_

        SPRING_KAFKA_PRODUCER_BOOTSTRAP_SERVERS = 'http://
        SPRING_KAFKA_CONSUMER_GROUP_ID = 'group_1'
        SPRING_KAFKA_CONSUMER_GROUP = 'group'

        SPRINGDOC_API_DOCS_PATH='/api/auth/v3/api-docs'
        SPRINGDOC_SWAGGER_UI_PATH='/api/auth/swagger-ui.ht

        DOCKERHUB_CREDENTIALS = credentials('dockerhub-jen
    }

    stages {
        stage('Checkout') {
            steps {
                script {
                    // Git 리포지토리에서 특정 브랜치를 체크아웃
                    def gitBranch = 'backend-auth-develop'
                    checkout([$class: 'GitSCM',
                              branches: [[name: "*/${gitBr
                              userRemoteConfigs: [[url: 'h
```

```
                        ])
                }
            }
        }

        stage('Setup') {
            steps {
                script {
                    // application.properties 파일을 환경 변수
                     sh '''
                        ls
                        cd backend-auth
                        ls
                        cd catchcatch-auth
                        mkdir -p src/main/resources
                        echo "spring.datasource.url=${DB_U
                        echo "spring.datasource.username=$
                        echo "spring.datasource.password=$
                        echo "jwt.secret=${JWT_SECRET}" >>
                        echo "jwt.access.token.expire =${J
                        echo "jwt.refresh.token.expire =${

                        echo "spring.data.redis.host=${SPR
                        echo "spring.data.redis.port=${SPR

                        # OAuth2 관련 설정 추가
                         echo "spring.security.oauth2.clie
                         echo "spring.security.oauth2.clie
                         echo "spring.security.oauth2.clie
                         echo "spring.security.oauth2.clie
                         echo "spring.security.oauth2.clie
                         echo "spring.security.oauth2.clie
                         echo "spring.security.oauth2.clie
                         echo "spring.security.oauth2.clie
                         echo "spring.security.oauth2.clie
                         echo "spring.security.oauth2.clie
                         echo "spring.security.oauth2.clie
```

```
                        echo "spring.security.oauth2.clie
                        echo "spring.security.oauth2.clie
                        echo "spring.security.oauth2.clie
                        echo "spring.security.oauth2.clie

                        echo "spring.kafka.producer.boots
                        echo "spring.kafka.consumer.group
                        echo "spring.kafka.consumer.group

                        echo "springdoc.api-docs.path=${S
                        echo "springdoc.swagger-ui.path=$
                    '''
                }
            }
        }

        stage('Build') {
            steps {
                script {
                    // gradlew 파일이 있는지 확인하고 실행 권한
                    sh '''
                        echo "Checking for gradlew file"

                        cd backend-auth
                        cd catchcatch-auth
                        ls
                        if [ -f gradlew ]; then
                            echo "Gradle Wrapper found, se
                            chmod +x gradlew
                            echo "Running Gradle Wrapper"
                            ./gradlew --version
                            echo "Building with Gradle"
                            ./gradlew clean build -i
                        else
                            echo "Gradle Wrapper not found
                            exit 1
                        fi
                    '''
```

```
                }
            }
        }

        stage('Docker Build') {
            steps {
                script {
                    // Docker 이미지를 빌드하고 태그를 붙입니다.
                    sh '''
                         cd backend-auth
                         cd catchcatch-auth

                         docker --version
                         docker login -u jjongbbang2 -p pd
                         docker build -t jjongbbang2/catch
                         docker build -t jjongbbang2/catch
                         docker images -a
                         docker push jjongbbang2/catchcatc
                         docker push jjongbbang2/catchcatc
                         docker rmi jjongbbang2/catchcatch
                         docker rmi jjongbbang2/catchcatch

                    '''
                }
            }
        }

        stage('SSH to Ubuntu') {
            steps {
                script {
                    sshagent(['SSH']) { // SSH 자격 증명을 설
                        sh '''
                            ssh -o StrictHostKeyChecking=n
                            cd auth
                            ./deploy.auth.sh
                        '''
                    }
                }
```

```
            }
        }



    }
}
```

▼ **[Front server] Jenkins 파이프라인**

> jenkins 파이프라인

```
pipeline {
    agent any

    environment {
        DOCKERHUB_CREDENTIALS = credentials('dockerhub-jen
    }
    tools {
        // 사용할 도구들을 정의합니다.
        nodejs "node"
    }

    stages {
        stage('Checkout') {
            steps {
                script {
                    // Git 리포지토리에서 특정 브랜치를 체크아웃
                    def gitBranch = 'frontend-main-develop
                    checkout([$class: 'GitSCM',
                              branches: [[name: "*/${gitBr
                              userRemoteConfigs: [[url: 'h
                             ])
                }
            }
        }
         stage('Setup') {
```

```
            steps {
                script {
                    // application.properties 파일을 환경 변수
                     sh '''
                        ls
                        cd frontend-main
                        echo "GENERATE_SOURCEMAP=false" >
                        ls -a

                    '''
                }
            }
        }

        stage('Build') {
            steps {
                script {
                    // package.json 파일 유무 확인 후 npm으로
                    echo "Checking for package.json file"

                    sh '''
                        cd frontend-main
                        npm --version
                        npm install --force
                        npm run build
                        '''
                }
            }
        }

        stage('Docker Build') {
            steps {
                script {
                    // Docker 이미지 빌드 및 Docker Hub에 업로
                    sh '''
                        cd frontend-main
                        ls
                        docker --version
```

```
                        docker login -u jjongbbang2 -p pdw
                        docker build -t jjongbbang2/catchc
                        docker build -t jjongbbang2/catchc
                        docker images -a
                        docker push jjongbbang2/catchcatch
                        docker push jjongbbang2/catchcatch
                        docker rmi jjongbbang2/catchcatch-
                        docker rmi jjongbbang2/catchcatch-
                    '''
                }
            }
        }

        stage('SSH to Ubuntu') {
            steps {
                script {
                    sshagent(['SSH']) { // SSH 자격 증명을 설
                        sh '''
                            ssh -o StrictHostKeyChecking=n
                            cd front
                            ./deploy.front.sh
                        '''
                    }
                }
            }
        }
    }
}
```

💡 **배포 시 docker 빌드 파일**

▼ **Frontend Dockerfile**

```
FROM node:18 AS build
```

```
WORKDIR /app

COPY package.json package-lock.json ./
RUN npm install

COPY . .

RUN npm run build

FROM nginx:alpine

COPY --from=build /app/dist /usr/share/nginx/html

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

▼ **Backend Dockerfile**

```
FROM openjdk:17-alpine
ARG JAR_FILE=/build/libs/catchcatchMain-0.0.1-SNAPSHOT.jar
COPY ${JAR_FILE} app.jar
COPY src/main/resources/application.properties /app/applic
EXPOSE 8080
ENTRYPOINT ["java","-jar","/app.jar"]
```

💡 **배포시 docker-compose.yml 파일**

▼ **Back-main && frontend blue docker-compose.yml 파일**

```
version: "3"
services:
  back_blue:
    container_name: back_blue
    image: jjongbbang2/catchcatch-main-blue
```

```
    expose:
      - 8080
    ports:
      - 8080:8080
    environment:
      TZ: Asia/Seoul
      SPRING_DATASOURCE_URL: jdbc:mysql://mysql:3306/evers

      SPRING_DATA_REDIS_HOST: redis
      SPRING_DATA_REDIS_PORT: 6379
      SPRING_JPA_HIBERNATE_DDL_AUTO: update
      SERVER_PORT: 8080
    networks:
      - backend

  front_blue:
    container_name: front_blue
    image: jjongbbang2/catchcatch-front-blue
    expose:
      - 3000
    ports:
      - 3000:80
    networks:
      - backend


networks:
  backend:
    external:
      name: backend
```

💡 **nginx, mysql, redis docker-compose.yml 파일**

```
version: "3"
services:
  mysql:
    image: mysql
    container_name: mysql
    environment:
      TZ: Asia/Seoul
    ports:
      - 3306:3306
    networks:
      - backend

  redis:
    container_name: redis
    image: redis
    ports:
      - 6379:6379
    networks:
      - backend

  nginx:
    image: nginx:latest
    container_name: nginx
    volumes:
      - ./nginx/conf.d:/etc/nginx/conf.d
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
    restart: always
    ports:
      - 80:80
      - 443:443
    networks:
      - backend

  certbot:
    container_name: certbot
    image: certbot/certbot
    restart: unless-stopped
```

```
    volumes:
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
    entrypoint: "/bin/sh -c 'trap exit TERM; while :; do cert


networks:
  backend:
    external:
      name: backend
```

💡 nginx 설정 파일

▼ default.blue.conf 파일

```
## default.blue.conf 파일

upstream chat {
    # Load balancing to two backend instances
    server everstar_chat_1:8080;
    server everstar_chat_2:8080;
}

server {
    listen 80;
    server_name i11b101.p.ssafy.io;
    server_tokens off;

    location /.well-known/acme-challenge/ {
        allow all;
        root /var/www/certbot;
    }

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

```
    }
}

server {
    listen 443 ssl;
    server_name i11b101.p.ssafy.io;
    server_tokens off;

    ssl_certificate /etc/letsencrypt/live/i11b101.p.ssafy.
    ssl_certificate_key /etc/letsencrypt/live/i11b101.p.ss
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    location / {
        proxy_pass http://front_blue:3000;
        proxy_set_header Host $host:$server_port;
        proxy_set_header X-Forwarded-Host $server_name;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forw
        proxy_cache off;  # Ensure caching is off
    }

    location /api/auth {
        proxy_pass http://everstar_auth:8080;
        proxy_set_header Host $host:$server_port;
        proxy_set_header X-Forwarded-Host $server_name;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forw
        proxy_cache off;  # Ensure caching is off
    }

    location /api/chat {
        proxy_pass http://chat;
        proxy_set_header Host $host:$server_port;
        proxy_set_header X-Forwarded-Host $server_name;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forw
        proxy_cache off;  # Ensure caching is off
```

```
        # webSocket
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_read_timeout 20m;
    }

    location /api {
        proxy_pass http://everstar_blue:8080;
        proxy_set_header Host $host:$server_port;
        proxy_set_header X-Forwarded-Host $server_name;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forw

        # SSE settings
        proxy_buffering off; # Turn off proxy buffering fo
        proxy_cache off;     # Turn off proxy cache for SS
        proxy_http_version 1.1;
        proxy_set_header Connection ''; # Clear the Connec
        chunked_transfer_encoding off; # Disable chunked t
    }

    location /ws {
        proxy_pass http://chat;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forw
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-Host $server_name;
        proxy_set_header X-Forwarded-Proto $scheme;

        # webSocket
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_read_timeout 20m;
    }
```

```
}
```

▼ default.green.conf 파일

```
## default.green.conf 파일

upstream chat {
    # Load balancing to two backend instances
    server everstar_chat_1:8080;
    server everstar_chat_2:8080;
}

server {
    listen 80;
    server_name i11b101.p.ssafy.io;
    server_tokens off;

    location /.well-known/acme-challenge/ {
        allow all;
        root /var/www/certbot;
    }

    location / {
        return 301 https://$host$request_uri;
    }
}
server {
    listen 443 ssl;
    server_name i11b101.p.ssafy.io;
    server_tokens off;

    ssl_certificate /etc/letsencrypt/live/i11b101.p.ssafy.
    ssl_certificate_key /etc/letsencrypt/live/i11b101.p.ss
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
```

```
    location / {
        proxy_pass http://front_green:3000;
        proxy_set_header Host $host:$server_port;
        proxy_set_header X-Forwarded-Host $server_name;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forw

        add_header Cache-Control no-store;
        add_header Pragma no-cache;
    }

    location /api/auth {
        proxy_pass http://everstar_auth:8080;
        proxy_set_header Host $host:$server_port;
        proxy_set_header X-Forwarded-Host $server_name;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forw

        add_header Cache-Control no-store;
        add_header Pragma no-cache;
    }

     location /api/chat {
        proxy_pass http://chat;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forw
        proxy_set_header X-Forwarded-Proto $scheme;

        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_read_timeout 20m;
    }

    location /api {
        proxy_pass http://everstar_green:8080;
```

```
        proxy_set_header Host $host:$server_port;
        proxy_set_header X-Forwarded-Host $server_name;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forw

        # SSE settings
        proxy_buffering off; # Turn off proxy buffering fo
        proxy_cache off;     # Turn off proxy cache for SS
        proxy_http_version 1.1;
        proxy_set_header Connection ''; # Clear the Connec
        chunked_transfer_encoding off; # Disable chunked t

        add_header Cache-Control no-store;
        add_header Pragma no-cache;
    }

    location /ws {
        proxy_pass http://chat;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forw
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-Host $server_name;
        proxy_set_header X-Forwarded-Proto $scheme;

        # webSocket
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_read_timeout 20m;
    }

}
```

💡 **blue green 자동배포를 위한 쉘 스크립트 파일**

deploy.sh 파일

```
EXIST_BLUE=$(docker ps --filter "name=everstar_blue" --filter

if [ -n "$EXIST_BLUE" ]; then
  echo "green up"
  docker-compose -f docker-compose-everstar-green.yml up -d -

  sleep 10

  docker cp ./default.green.conf nginx:/etc/nginx/conf.d/defa
  docker-compose exec -T nginx service nginx reload

  docker stop everstar_blue
  docker rm everstar_blue
  docker stop front_blue
  docker rm front_blue

  echo "rmi image"
  docker rmi jjongbbang2/everstar-back-main-blue
  docker rmi jjongbbang2/everstar-front-blue-u

else
  echo "blue up"
  docker-compose -f docker-compose-everstar-blue.yml up -d --

  sleep 10

  docker cp ./default.blue.conf nginx:/etc/nginx/conf.d/defau
  docker-compose exec -T nginx service nginx reload

  docker stop everstar_green
  docker rm everstar_green
  docker stop front_green
  docker rm front_green

  echo "rmi image"
```

```
  docker rmi jjongbbang2/everstar-back-main-green
  docker rmi jjongbbang2/everstar-front-green-u
fi
```

💡 **Kafka compose.yml 파일**

docker-compose-kafka.yml

```
version: '3.7'
services:
  zk1:
    container_name: zookeeper1
    image: wurstmeister/zookeeper:latest
    restart: always
    hostname: zk1
    ports:
      - "2181:2181"
    environment:
      ZOO_MY_ID: 1
      ZOO_SERVERS: server.1=zk1:2888:3888;2181 server.2=zk2:2
    volumes:
      - "~/zk-cluster/zk1/data:/data"

  kafka1:
    container_name: kafka1
    image: wurstmeister/kafka:latest
    restart: on-failure
    depends_on:
      - zk1
    ports:
      - "9092:9092"  # 이 포트를 원하는 포트로 조정 가능
    environment:
      KAFKA_BROKER_ID: 1
```

```
      KAFKA_ADVERTISED_HOST_NAME: 43.203.128.74  # 이 부분도 필
      BOOTSTRAP_SERVERS: 43.203.128.74:9092
      KAFKA_ZOOKEEPER_CONNECT: "zk1:2181"
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1  # 하나의 브로커
      KAFKA_CREATE_TOPICS: "ek-log:1:1"

  kafka-ui:
    image: provectuslabs/kafka-ui
    container_name: kafka-ui
    ports:
      - "10000:8080"
    restart: always
    environment:
      - KAFKA_CLUSTERS_0_NAME=local
      - KAFKA_CLUSTERS_0_BOOTSTRAPSERVERS=43.203.128.74:9092
      - KAFKA_CLUSTERS_0_ZOOKEEPER=zk1:2181
```

💡 **elk 설치**

elk 설치

```
git clone https://github.com/paullee714/Flask-Vue-ELK-Mongo-D
```

elasticsearch.yml

```
cluster.name: "docker-cluster"
network.host: 0.0.0.0

## X-Pack settings
## see https://www.elastic.co/guide/en/elasticsearch/referenc
#
```

```
xpack.license.self_generated.type: trial
xpack.security.enabled: true
xpack.monitoring.collection.enabled: true
```

kibana.yml

```
server.name: kibana
server.host: "0"
elasticsearch.hosts: [ "http://elasticsearch:9200" ]
xpack.monitoring.ui.container.elasticsearch.enabled: true

## X-Pack security credentials
#
elasticsearch.username: elastic
elasticsearch.password: changeme
```

logstash.yml

```
http.host: "0.0.0.0"
xpack.monitoring.elasticsearch.hosts: [ "http://elasticsearch

## X-Pack security credentials
#
xpack.monitoring.enabled: true
xpack.monitoring.elasticsearch.username: elastic
xpack.monitoring.elasticsearch.password: changeme
```

logstash.conf

```
input {
    tcp {
        port => 5001
    }
}
```

```
## Add your filters / logstash plugins configuration here

output {
    elasticsearch {
        hosts => "elasticsearch:9200"
        user => "elastic"
        password => "changeme"
        index => "elk-logger"
    }
}
```

docker-compose-elk.yml

```
version: '3.7'

services:

  setup:
    build:
      context: setup/
      args:
        ELASTIC_VERSION: ${ELASTIC_VERSION}
    volumes:
      - ./setup/entrypoint.sh:/entrypoint.sh:ro
      - ./setup/lib.sh:/lib.sh:ro
      - ./setup/roles:/roles:ro
    environment:
      ELASTIC_PASSWORD: ${ELASTIC_PASSWORD:-}
      LOGSTASH_INTERNAL_PASSWORD: ${LOGSTASH_INTERNAL_PASSWOR
      KIBANA_SYSTEM_PASSWORD: ${KIBANA_SYSTEM_PASSWORD:-}
      METRICBEAT_INTERNAL_PASSWORD: ${METRICBEAT_INTERNAL_PAS
      FILEBEAT_INTERNAL_PASSWORD: ${FILEBEAT_INTERNAL_PASSWOR
      HEARTBEAT_INTERNAL_PASSWORD: ${HEARTBEAT_INTERNAL_PASSW
      MONITORING_INTERNAL_PASSWORD: ${MONITORING_INTERNAL_PAS
      BEATS_SYSTEM_PASSWORD: ${BEATS_SYSTEM_PASSWORD:-}
    networks:
```

```
      - elk
    depends_on:
      - elasticsearch

  elasticsearch:
    build:
      context: elasticsearch/
      args:
        ELASTIC_VERSION: ${ELASTIC_VERSION}
    volumes:
      - ./elasticsearch/config/elasticsearch.yml:/usr/share/e
      - elasticsearch:/usr/share/elasticsearch/data:Z
    ports:
      - 9200:9200
      - 9300:9300
    environment:
      node.name: elasticsearch
      ES_JAVA_OPTS: -Xms512m -Xmx512m
      ELASTIC_PASSWORD: ${ELASTIC_PASSWORD:-}
      discovery.type: single-node
    networks:
      - elk
    restart: unless-stopped

  logstash:
    build:
      context: logstash/
      args:
        ELASTIC_VERSION: ${ELASTIC_VERSION}
    volumes:
      - ./logstash/config/logstash.yml:/usr/share/logstash/co
      - ./logstash/pipeline:/usr/share/logstash/pipeline:ro,Z
    ports:
      - 5044:5044
      - 50000:50000/tcp
      - 50000:50000/udp
      - 9600:9600
    environment:
```

```
      LS_JAVA_OPTS: -Xms256m -Xmx256m
      LOGSTASH_INTERNAL_PASSWORD: ${LOGSTASH_INTERNAL_PASSWOR
    networks:
      - elk
    depends_on:
      - elasticsearch
    restart: unless-stopped

  kibana:
    build:
      context: kibana/
      args:
        ELASTIC_VERSION: ${ELASTIC_VERSION}
    volumes:
      - ./kibana/config/kibana.yml:/usr/share/kibana/config/k
    ports:
      - 5601:5601
    environment:
      KIBANA_SYSTEM_PASSWORD: ${KIBANA_SYSTEM_PASSWORD:-}
    networks:
      - elk
    depends_on:
      - elasticsearch
    restart: unless-stopped

networks:
  elk:
    driver: bridge

volumes:
  elasticsearch:
```

💡 **prometheus grafana docker-compose.yml**

docker-compose-grafana.yml

```
version: "3"

networks:
  t4y:
    driver: bridge

services:
  prometheus:
    image: prom/prometheus
    container_name: prometheus
    volumes:
      - ./prometheus/config:/etc/prometheus
      - prometheus-data:/prometheus
    ports:
      - 9090:9090
    command:
      - '--storage.tsdb.path=/prometheus'
      - '--config.file=/etc/prometheus/prometheus.yml'
    restart: always
    networks:
      - t4y

  grafana:
    image: grafana/grafana
    container_name: grafana
    ports:
      - 4000:3000
    volumes:
      - grafana-data:/var/lib/grafana
      - ./grafana/provisioning/:/etc/grafana/provisioning/
    restart: always
    depends_on:
      - prometheus
    networks:
      - t4y

  node_exporter:
    image: prom/node-exporter
```

```
    volumes:
      - /proc:/host/proc:ro
      - /sys:/host/sys:ro
      - /:/rootfs:ro
    command:
      - '--path.procfs=/host/proc'
      - '--path.rootfs=/rootfs'
      - '--path.sysfs=/host/sys'
      - '--collector.filesystem.mount-points-exclude=^/(sys|p
    ports:
      - "9100:9100"
    networks:
      - t4y

volumes:
  grafana-data:
  prometheus-data:
```