♾️

Infra 설정

서버

기본 제공 서버

- **ssh -i K9B103T.pem ubuntu@k9b103.p.ssafy.io**

추가 제공 서버

- **ssh -i K9B103T.pem ubuntu@k9b103a.p.ssafy.io**

배포 관련 정보

▼ **정보**

1. Nginx

- port: 80(SSL), 11111(No SSL)

2. MySQL

- host: cyes.cehwvnokrv9c.ap-northeast-2.rds.amazonaws.com
- port: 39698
- username: cyesadmin
- pw: c-yeswecanescapessafy951961971982011

3. Redis

- port: 6339
- pw : cyesredishackerfuckingmanweusesoket97jmeter!
- 들어가는 방법

```
1. docker exec -it redis /bin/bash
2. redis-cli
3. auth cyesredishackerfuckingmanweusesoket97jmeter!
```

4. MongoDB

- host: k9b103.p.ssafy.io
- port: 31024
- username: yoomongo
- pw: cyesyoomongojofkawoosocket1026

5. Jenkins

- id: rudcnrql103
- pw: rudcnrql103!@#
- GitLab Access Token: f8410a871335fb15717ba3c8ad5762b4
- GitLab Credential:
- SSH Credential:

6. Grafana

- id: admin
- pw: cyesgrafanatrustme103
- port: 10000
- url: http://cyes.site:10000

7. Prometheus

- port: 8080
- url: http://cyes.site:8080

8. Nginx Exporter

- port: 12222
- url: http://cyes.site:12222

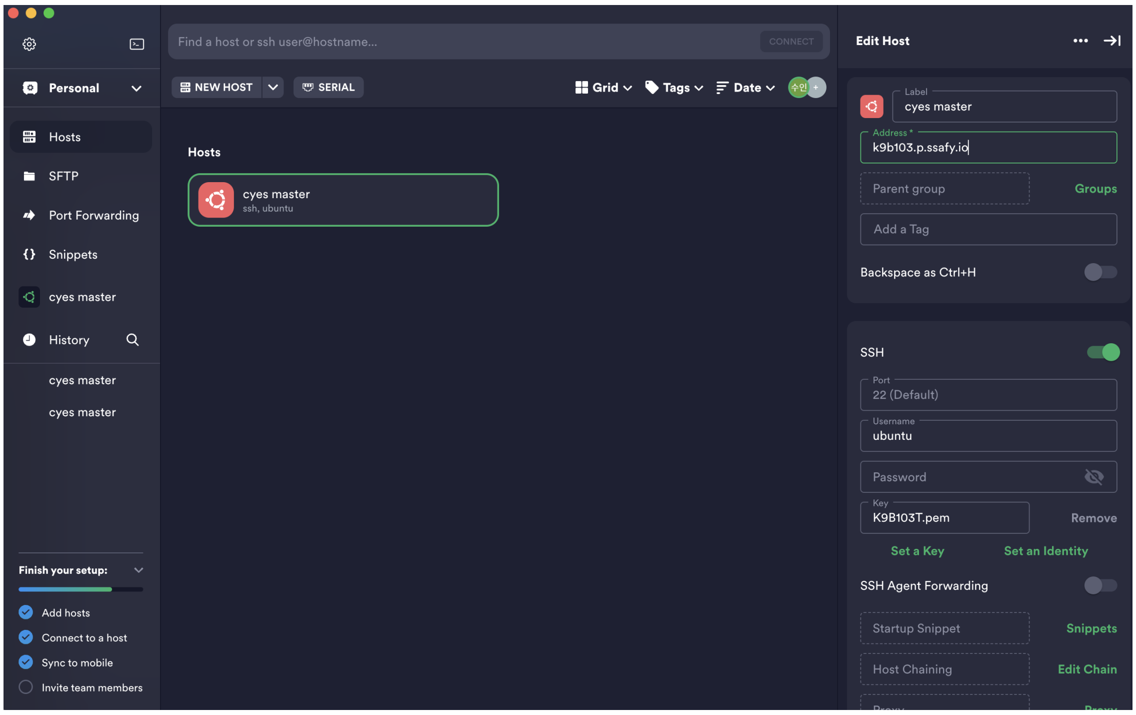
1차 테스트 일정(23.11.01 수)

▼ **수동 배포**

▼ **Termius**

💡 macOS, Windows 및 Linux용 SSH 클라이언트로, 다양한 기능과 팀 관리 옵션을 제공

Termius 설정



Label: cyes master
Address *: k9b103.p.ssafy.io
Parent group
Add a Tag
Backspace as Ctrl+H
SSH
Port: 22 (Default)
Username: ubuntu
Password
Key: K9B103T.pem Remove
Set a Key Set an Identity
SSH Agent Forwarding
Startup Snippet
Host Chaining

1. **설치**
   - MicroSoft Store를 통해 설치
2. **설정**
   - **New host** 선택
   - **Label** 입력
     - Termius에서 내가 식별할 이름
     - 기본 서버: cyes master
     - 추가 서버: cyes master(미정)
   - **Address** 입력
     - 접속할 주소
     - 기본 서버: k9b103.p.ssafy.io
     - 추가 서버: k9b103a.p.ssafy.io
   - **SSH** 입력
     - UserName: ubuntu
     - key: Pem key import

▼ **Docker & Docker Hub**

💡 Docker: 컨테이너 기술을 사용하여 애플리케이션을 개발, 배포 및 실행하기 위한 오픈 소스 플랫폼
Docker Hub: Docker 이미지를 저장, 공유 및 관리하기 위한 클라우드 기반 레지스트리 서비스입니다.

Docker 설정

1. **ec2 서버 접속**
2. **Docker** 설치

```
# apt update
sudo apt update

# install docker
```

```
sudo apt install docker.io

# check docker version
docker -v
# Docker version 24.0.5, build 24.0.5-0ubuntu1~20.04.1
```

3. **Docker Hub** 가입 및 레포지토리 생성
    - 회원가입
    - Create Repository

▼ BE - container

💡 1. Dockerfile 생성
2. IDE 내부에서 Build 후 jar 파일 생성 확인
3. Docker image 생성
4. Docker Hub Repo에 Push
5. ec2 서버에서 Pull 받은 후 실행

1. **Dockerfile** 생성

```
# jdk version
FROM openjdk:11-jdk

# 애플리케이션 위치 지정(환경변수)
ARG JAR_FILE=./build/libs/cyes_master-0.0.1-SNAPSHOT.jar

# JAR_FILE 경로로부터 jar파일 복사 후 /app.jar 경로에 복사
COPY $JAR_FILE app.jar

# Docker 컨테이너가 시작될 때 실행될 명령을 지정
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

2. IDE 내부에서 **Build** 후 jar파일 및 오류 확인
3. **Docker image** 생성

```
# 이미지 생성
docker build --tag <도커계정명>/cyes_master:0.0.1 .

# 이미지 확인
docker images
```

4. **Docker Hub Push**

```
# 로그인
docker login

# push
docker push <이미지 이름>
```

5. ec2 서버에서 **pull** 후 **run**

```
# docker login
docker login

# docker pull
docker pull <이미지 이름>

# run
docker run -d --name <컨테이너명> -p 포트:포트 <도커이미지>
```

▼ FE - container

💡
1. root dir에 Dockerfile 생성 (cyesfront)
2. IDE 내부에서 Build
3. Docker image 생성
4. Docker Hub Repo에 Push
5. ec2 서버에서 Pull 받은 후 실행

1. **Dockerfile** 생성

```
```

2. IDE 내부에서 **Build**

   docker build -t qotnqls1998/cyes .

3. **Docker image** 생성

```
# 이미지 생성
docker build --t qotnqls1998/cyes_master:0.0.1 .

# 이미지 확인
docker images
```

4. **Docker Hub Push**

```
# 로그인
docker login

# push
docker push <이미지 이름>
```

5. ec2 서버에서 **pull** 후 **run**

```
# docker login
docker login

# docker pull
docker pull <이미지 이름>

# run
docker run -d --name <컨테이너명> -p 포트:포트 <도커이미지>
```

▼ **MongoDB**

▼ **Redis**

▼ **Nginx**

- 참고자료

[프로젝트] SpringBoot + React 웹 서비스 Docker(Nginx, SSL/Reverse Proxy, Redis)로 배포하기
기존의 프로젝트에서는 Github Actions를 이용하여 빌드 파일을 압축하여 S3로 전송한 뒤, CodeDeploy를 통하여 EC2 서버 내에서 Nginx를 통해 배포하였다. 하지만, CI/CD가 복잡하고 긴 점이 아쉬워서 도커를 이용하여 여러 개의 컨테이너(Ng
https://velog.io/@kmw10693/프로젝트-SpringBoot-React-웹-서비스-DockerNginx-SSLReverse-Proxy-Redis로-배포하기

1. nginx 설치

```
docker run -d --name nginx-container -p 80:80 -v /path/to/nginx.conf:/etc/nginx/nginx.conf nginx:latest
```

2. nginx.conf

```
events {}

http {
  server {
    listen 80;
    server_name your-domain.com; # 도메인 또는 IP 주소로 변경

    location /springboot {
      proxy_pass http://호스트IP주소:8080; # Spring Boot 서버로 프록시
      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }

    location /react {
      proxy_pass http://호스트IP주소:3000; # React 서버로 프록시
      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
  }
}
```

3. SSL 설정 후 nginx.conf

```
events {}

http {
  server {
    listen 80;
    server_name your-domain.com; # 도메인 또는 IP 주소로 변경

    # HTTP 요청을 HTTPS로 리다이렉션
    location / {
      return 301 https://$host$request_uri;
    }
  }

  server {
    listen 443 ssl;
    server_name your-domain.com; # 도메인 또는 IP 주소로 변경

    ssl_certificate /etc/nginx/ssl/your-certificate.crt; # SSL 인증서 파일 경로
    ssl_certificate_key /etc/nginx/ssl/your-certificate.key; # SSL 인증서 키 파일 경로

    location /springboot {
      proxy_pass http://호스트IP주소:5000; # Spring Boot 서버로 프록시
      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }

    location /react {
      proxy_pass http://호스트IP주소:3897; # React 서버로 프록시
      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
  }
}
```

4. nginx (11.03)

```
# Default server configuration
server
{
        listen 80 default_server;
        listen [::]:80 default_server;

        server_name cyes.site k9b103.p.ssafy.io;
        return 301 https://$server_name$request_uri; # HTTP를 HTTPS로 리다이렉트
}

server
{
        listen 11111 default_server;
        listen [::]:11111 default_server;
        server_name cyes.site k9b103.p.ssafy.io;
        location /metrics {

                stub_status on;

        }

}

server {
    listen 443 ssl default_server;
    listen [::]:443 ssl default_server;

    server_name cyes.site k9b103.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/cyes.site/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/cyes.site/privkey.pem;

    location / {
        proxy_pass http://localhost:9510;
        add_header 'Access-Control-Allow-Origin' '*';
        add_header 'Access-Control-Expose-Headers' 'Authorization, Authorizationrefresh';
        include /etc/nginx/proxy_params;

        proxy_buffer_size          128k;
        proxy_buffers              4 256k;
        proxy_busy_buffers_size    256k;
    }
    location /api {
        proxy_pass http://localhost:5000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
    location /monitor/ {
        proxy_pass http://localhost:19999/;
        proxy_set_header Host $host;

        # WebSocket 지원을 위한 추가 설정
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }
    location /api-docs {
        proxy_pass http://localhost:5000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

▼ 자동배포

▼ Jenkins

▼ Docker Compose

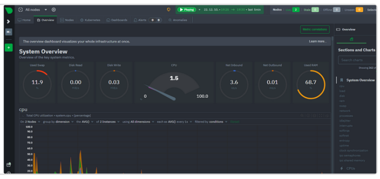
▼ 서버 모니터링

▼ Netdata

서버 모니터링 with docker => 2. netdata

🍺 목차 🍺(보시려면 아래 더보기 를 눌러주세요.) 더보기 서버 모니터링 with docker => 1. vnstat 서버 모니터링 with docker => 2. netdata 서버 모니터링 with docker => 3. grafana, prometheus, nodeexporter, cadvisor, alertmanager 서버 모니터링 with docker => 4. mysqld-exporter, nginx-prometheus-exporter

https://mungkhs1.tistory.com/71



▼ Prometheus & Grafana

1. Node Exporter 설치

- 모니터링 서버에서

```
wget https://github.com/prometheus/node_exporter/releases/download/v1.6.1/node_exporter-1.6.1.linux-386.tar.gz
```

- 압축 해제

```
tar xvfz node_exporter-1.3.1.linux-arm64.tar.gz
```

- 실행

```
nohup ./node_exporter --web.listen-address=:8081 &
```

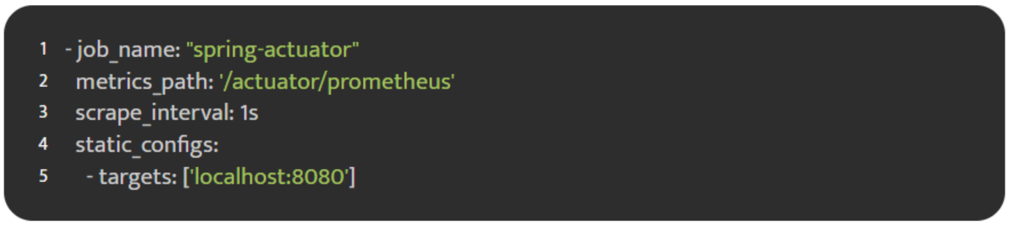
2. 프로메테우스

```
wget https://github.com/prometheus/prometheus/releases/download/v2.47.2/prometheus-2.47.2.linux-386.tar.gz
```

- 압축 해제

```
tar xvfz prometheus-2.47.2.linux-386.tar.gz
```

- 폴더에서 prometheus.yml 수정
- 초 수정도 가능



```
- job_name: "spring-actuator"
  metrics_path: '/actuator/prometheus'
  scrape_interval: 1s
  static_configs:
    - targets: ['localhost:8080']
```

```
# my global config
global:
  scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
          # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["localhost:9090"]
  - job_name: "nginx_monitor"
    metrics_path: "/metrics"
    static_configs:
      - targets: ["wishme.co.kr:8082"]
```

- 실행 - status, targets에 들어가서 연결 확인

```
nohup ./prometheus --config.file=prometheus.yml --web.listen-address=:8080 > prometheus.log 2>&1 &
```

그라파나

- 설치

```
wget https://dl.grafana.com/enterprise/release/grafana-enterprise-9.0.5.linux-amd64.tar.gz
tar -zxvf grafana-enterprise-9.0.5.linux-amd64.tar.gz
```

- 포트 수정 - /conf/default.ini

```
# The http port to use
http_port = 10000
```

- 실행

```
nohup ./grafana-server > grafana.log 2>&1 &
```

Nginx exporter 설치

```
wget https://github.com/nginxinc/nginx-prometheus-exporter/releases/download/v0.11.0/nginx-prometheus-exporter_0.11.0_linux_38
```

```
tar xvfz nginx-prometheus-exporter_0.11.0_linux_386.tar.gz
```

실행

```
nohup ./nginx-prometheus-exporter -nginx.scrape-uri=http://localhost/metrics --web.listen-address=:12222 &
```

▼ Error 정리

- **jar 파일 찾지 못함 → IDE 내부에서 Build (Docker Image Build)**

```
ERROR: failed to solve: failed to compute cache key: failed to calculate checksum of ref d41eb12e-fd2b-4fbd-863e-00293fba72
72::o96i09newfy5m6xhchgfzvy3e: failed to walk /var/lib/docker/tmp/buildkit-mount3533975976/build/libs:
 lstat /var/lib/docker/tmp/buildkit-mount3533975976/build/libs: no such file or directory
```

- **400 에러 → include param 삭제(Nginx)**

```
xhr.js:256 GET https://cyes.site/api/problem/all 400 (Bad Request)
dispatchXhrRequest @ xhr.js:256
xhr @ xhr.js:49
dispatchRequest @ dispatchRequest.js:51
request @ Axios.js:146
Axios.<computed> @ Axios.js:172
wrap @ bind.js:5
handleCheck @ Login.tsx:24
callCallback @ react-dom.development.js:4164
invokeGuardedCallbackDev @ react-dom.development.js:4213
invokeGuardedCallback @ react-dom.development.js:4277
invokeGuardedCallbackAndCatchFirstError @ react-dom.development.js:4291
executeDispatch @ react-dom.development.js:9041
processDispatchQueueItemsInOrder @ react-dom.development.js:9073
processDispatchQueue @ react-dom.development.js:9086
dispatchEventsForPlugins @ react-dom.development.js:9097
(anonymous) @ react-dom.development.js:9288
batchedUpdates$1 @ react-dom.development.js:26140
batchedUpdates @ react-dom.development.js:3991
dispatchEventForPluginEventSystem @ react-dom.development.js:9287
dispatchEventWithEnableCapturePhaseSelectiveHydrationWithoutDiscreteEventReplay @ react-dom.development.js:6465
dispatchEvent @ react-dom.development.js:6457
dispatchDiscreteEvent @ react-dom.development.js:6430
Login.tsx:30 Error during login: AxiosError {message: 'Request failed with status code 400', name: 'AxiosError', code: 'ERR
_BAD_REQUEST', config: {…}, request: XMLHttpRequest, …}
```

```
# 👀 C'YES Porting Manual

### 1. **Develop Environment**

> #### 1.1 MICRO SERVICE

    1. 모놀리식 젠킨스 파이프라인

```
> jenkins 파일
       pipeline {
       agent any
       stages {
         //백엔드
               stage('BE build') {
                   steps {
                       dir('Server/webserver'){
                           sh '''
                           pwd
                           echo 'springboot build'
                           chmod +x gradlew
                           ./gradlew clean build -x test
                           '''
                       }
                   }
               }
               stage('BE Dockerimage build') {
                   steps {
                       dir('Server/webserver'){
                           sh '''
                           echo 'Dockerimage build'
                           docker build -t docker-springboot:0.0.1 .
                           '''
                       }
                   }
               }
               stage('BE Deploy') {
                   steps {
                       dir('Server/webserver'){

                           sh '''
                           echo 'Deploy'

                           result=$(docker container ls -a --filter "name=cyes_back" -q)
                           if [-n "$result"]; then
                                   docker stop $result
```
```

```
                                   docker rm $result

                               else
                                   echo "No such containers"
                               fi
                           docker run -d -p 127.0.0.1:5000:5000 -p 1026:5000 --name cyes_back -e JAVA_OPTS="-Duser.timezone=Asia/Seoul'


                           docker images -f "dangling=true" -q | xargs -r docker rmi
                           '''
                       }
                   }
               }


               //프론트 엔드
               stage('FE build') {
                   steps {
                       dir('Front/cyesfront'){
                           sh '''
                               pwd
                               echo 'Frontend build'
                                DEBIAN_FRONTEND=noninteractive apt install -y npm

                               npm install
                               CI=false npm run build
                           '''
                       }
                   }
               }
               stage('FE Dockerimage build') {
                   steps {
                       dir('Front/cyesfront'){
                           sh '''
                               echo 'Dockerimage build'
                               docker build --no-cache -t cyes_front:0.0.1 .
                           '''
                       }
                   }
               }
               stage('FE Deploy') {
                   steps {
                       dir('Front/cyesfront'){

                           sh '''
                               echo 'FE Deploy'

                           result=$( docker container ls -a --filter "name=cyes_front" -q )
                           if [ -n "$result" ]; then
                                   docker stop $result
                                   docker rm $result

                               else
                                   echo "No such containers"
                               fi

                           docker run -d -p 127.0.0.1:9510:80 --name cyes_front cyes_front:0.0.1
                           docker images -f "dangling=true" -q | xargs -r docker rmi
                           '''
                       }
                   }
               }
           }
       }


```

    1. 모놀리식 젠킨스 파이프라인

```
> 마이크로 서비스로 변경한 후 젠킨스 파일

pipeline {
    agent any

stages {


      //백엔드
            stage('BE build') {

                steps {
```

```
                dir('Server/webserver'){
                    sh '''
                    pwd
                    echo 'springboot build'
                    chmod +x gradlew
                    ./gradlew clean build -x test
                    '''
                }
            }
        }


        stage('BE Dockerimage build') {

            steps {

                dir('Server/webserver'){
                    sh '''
                    echo 'Dockerimage build'
                    docker build -t docker-springboot:0.0.1 .
                    '''
                }
            }
        }


        stage('BE Deploy') {

            steps {

                dir('Server/webserver'){

                    sh '''
                    echo 'Deploy'

                    result=$( docker container ls -a --filter "name=cyes_back" -q )
                    if [ -n "$result" ]; then
                            docker stop $result
                            docker rm $result

                        else
                            echo "No such containers"
                        fi

                            echo "gogo"

                    docker images -f "dangling=true" -q | xargs -r docker rmi
                    '''
                }
            }
        }


        //프론트 엔드
        stage('FE build') {
            steps {
                dir('Front/cyesfront'){
                    sh '''
                        pwd
                        echo 'Frontend build'
                         DEBIAN_FRONTEND=noninteractive apt install -y npm

                        npm install
                        CI=false npm run build
                    '''
                }
            }
        }
        stage('FE Dockerimage build') {
            steps {
                dir('Front/cyesfront'){
                    sh '''
                        echo 'Dockerimage build'
                        docker build --no-cache -t cyes_front:0.0.1 .
                    '''
                }
            }
        }
        stage('FE Deploy') {
            steps {
                dir('Front/cyesfront'){

                    sh '''
                        echo 'FE Deploy'
                    result=$( docker container ls -a --filter "name=cyes_front" -q )
                    if [ -n "$result" ]; then
                            docker stop $result
                            docker rm $result

                        else
                            echo "No such containers"
                        fi



                    echo "gogo"
                    docker images -f "dangling=true" -q | xargs -r docker rmi
                    '''
                }
            }
        }


          stage('MSA Container backend') {
                steps {
                    dir('/'){

                      script {
                def fileName = 'spring-boot.tar'

                // 파일이 존재하는지 확인
                if (fileExists(fileName)) {
                    echo "Deleting ${fileName}"
                    // 파일 삭제
                    sh "rm ${fileName}"
                } else {
                    echo "${fileName} does not exist. Skipping deletion."
                }



            }


            sh '''
                    docker save -o spring-boot.tar docker-springboot:0.0.1

                    result=$(ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker container ls -a --filter 'name=cyes_back' -q")
                    if [ -n "$result" ]; then

                            ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "rm spring-boot.tar"
                            scp -i /jenkins_key /spring-boot.tar ubuntu@k9b103a.p.ssafy.io:/home/ubuntu
                            ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker stop cyes_back"
                            ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker rm cyes_back"
                            ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker rmi docker-springboot:0.0.1"

                        else
                            echo "No such containers"
                        fi

                        ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker load -i spring-boot.tar"
                        ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker run -d -p 127.0.0.1:5000:5000 -p 1026:5000 --name cy
                '''
                    }

                    }
                }

             stage('MSA Container frontend') {
                steps {
                    dir('/'){

            script {
                def fileName = 'cyes_front.tar'

                // 파일이 존재하는지 확인
                if (fileExists(fileName)) {
                    echo "Deleting ${fileName}"
                    // 파일 삭제
                    sh "rm ${fileName}"
                } else {
                    echo "${fileName} does not exist. Skipping deletion."
                }

            }

            sh '''
```

~~~
                        docker save -o cyes_front.tar cyes_front:0.0.1


                            result=$(ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker container ls -a --filter 'name=cyes_front' 
                        if [ -n "$result" ]; then

                                ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "rm cyes_front.tar"
                                ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker stop cyes_front"
                                ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker rm cyes_front"
                                ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker rmi cyes_front:0.0.1"

                            else
                                echo "No such containers"
                            fi
                            scp -i /jenkins_key /cyes_front.tar ubuntu@k9b103a.p.ssafy.io:/home/ubuntu
                            ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker load -i cyes_front.tar"
                            ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker run -d -p 127.0.0.1:9510:80 --name cyes_front cyes_fr
                    '''
                        }

                        }
                    }
        }
    }
```

> ### 1.2 Nginx Setting

    모놀리식 아키텍처 경우 nginx 설정

```
> nginx -> site-enabled -> default

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

        server_name cyes.site k9b103.p.ssafy.io;
        return 301 https://$server_name$request_uri; # HTTP를 HTTPS로 리다이렉트
}

server
{
        listen 11111 default_server;
        listen [::]:11111 default_server;
        server_name cyes.site k9b103.p.ssafy.io;

        location /metrics {

                stub_status on;

        }

}

server {
    listen 443 ssl default_server;
    listen [::]:443 ssl default_server;

    server_name cyes.site k9b103.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/cyes.site/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/cyes.site/privkey.pem;

    location / {
        proxy_pass http://localhost:9510;
        add_header 'Access-Control-Allow-Origin' '*';
        add_header 'Access-Control-Expose-Headers' 'Authorization, Authorizationrefresh';
        include /etc/nginx/proxy_params;

        proxy_buffer_size          128k;
        proxy_buffers              4 256k;
        proxy_busy_buffers_size    256k;
    }
    location /api {
        proxy_pass http://localhost:5000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
~~~

~~~~
    location /monitor/ {
        proxy_pass http://localhost:19999/;
        proxy_set_header Host $host;

        # WebSocket 지원을 위한 추가 설정
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }

    location /api-docs/ {
        proxy_pass_request_headers on;
        proxy_set_header Host $host;
        proxy_http_version 1.1;
        proxy_pass http://localhost:5000/api-docs/;
    }
}
```

> ### work 서버 nginx default 파일

    마이크로 서비스의 경우  nginx deafalt 설정.

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

        server_name cyes.site k9b103a.p.ssafy.io;
        return 308 https://$server_name$request_uri; # HTTP를 HTTPS로 리다이렉트
}

server
{
        listen 11111 default_server;
        listen [::]:11111 default_server;
        server_name cyes.site k9b103.p.ssafy.io;

        location /nginx_sub_metrics {

                stub_status on;
        }

}


server {
    listen 443 ssl default_server;
    listen [::]:443 ssl default_server;

    server_name cyes.site k9b103a.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/cyes.site/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/cyes.site/privkey.pem;

    location / {
        proxy_pass http://localhost:9510;
        add_header 'Access-Control-Allow-Origin' '*';
        add_header 'Access-Control-Expose-Headers' 'Authorization, Authorizationrefresh';
        include /etc/nginx/proxy_params;

        proxy_buffer_size          128k;
        proxy_buffers              4 256k;
        proxy_busy_buffers_size    256k;
    }

    location /api {
        proxy_pass http://localhost:5000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }

}
```

---

> 소켓만 떼어놓은 코드

```
~~~~

```
pipeline {
    agent any

stages {

    stage('MSA Container Deploy') {
                    steps {
                        dir('/'){

                          script {
                    def fileName = 'spring-boot.tar'

                    // 파일이 존재하는지 확인
                    if (fileExists(fileName)) {
                        echo "Deleting ${fileName}"
                        // 파일 삭제
                        sh "rm ${fileName}"
                    } else {
                        echo "${fileName} does not exist. Skipping deletion."
                    }



                }
                sh '''
                        docker save -o spring-boot.tar docker-springboot:0.0.1

                        ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "rm spring-boot.tar"

                        scp -i /jenkins_key /spring-boot.tar ubuntu@k9b103a.p.ssafy.io:/home/ubuntu

                        result=$(ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker container ls -a --filter 'name=cyes_back' -q")
                        if [ -n "$result" ]; then
                                ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker stop cyes_back"
                                ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker rm cyes_back"
                                ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker rmi docker-springboot:0.0.1"

                            else
                                echo "No such containers"
                            fi





                            ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker load -i spring-boot.tar"
                            ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker run -d -p 127.0.0.1:5000:5000 -p 1026:5000 --name cye
                    '''
                        }

                        }
                    }


      //백엔드
            stage('BE build') {

                steps {

                    dir('Server/webserver'){
                        sh '''
                        pwd
                        echo 'springboot build'
                        chmod +x gradlew
                        ./gradlew clean build -x test
                        '''
                    }
                }
            }


            stage('BE Dockerimage build') {

                steps {

                    dir('Server/webserver'){
                        sh '''
                        echo 'Dockerimage build'
                        docker build -t docker-springboot:0.0.1 .
                        '''
                    }
                }
            }
```

~~~
            stage('BE Deploy') {

                steps {

                    dir('Server/webserver'){

                        sh '''
                        echo 'Deploy'

                        result=$( docker container ls -a --filter "name=cyes_back" -q )
                        if [ -n "$result" ]; then
                                docker stop $result
                                docker rm $result

                            else
                                echo "No such containers"
                            fi
                        docker run -d -p 127.0.0.1:5000:5000 -p 1026:5000 --name cyes_back -e JAVA_OPTS="-Duser.timezone=Asia/Seoul" do


                        docker images -f "dangling=true" -q | xargs -r docker rmi
                        '''
                    }
                }
            }


            //프론트 엔드
            stage('FE build') {
                steps {
                    dir('Front/cyesfront'){
                        sh '''
                            pwd
                            echo 'Frontend build'
                             DEBIAN_FRONTEND=noninteractive apt install -y npm

                            npm install
                            CI=false npm run build
                        '''
                    }
                }
            }
            stage('FE Dockerimage build') {
                steps {
                    dir('Front/cyesfront'){
                        sh '''
                            echo 'Dockerimage build'
                            docker build --no-cache -t cyes_front:0.0.1 .
                        '''
                    }
                }
            }
            stage('FE Deploy') {
                steps {
                    dir('Front/cyesfront'){

                        sh '''
                            echo 'FE Deploy'

                        result=$( docker container ls -a --filter "name=cyes_front" -q )
                        if [ -n "$result" ]; then
                                docker stop $result
                                docker rm $result

                            else
                                echo "No such containers"
                            fi

                        docker run -d -p 127.0.0.1:9510:80 --name cyes_front cyes_front:0.0.1
                        docker images -f "dangling=true" -q | xargs -r docker rmi
                        '''
                    }
                }
            }
        }
    }

```

---

### 스프링, 레디스, 프론트 옮기기
~~~

~~~~
```
pipeline {
    agent any

stages {


      //백엔드
            stage('BE build') {

                steps {

                    dir('Server/webserver'){
                        sh '''
                        pwd
                        echo 'springboot build'
                        chmod +x gradlew
                        ./gradlew clean build -x test
                        '''
                    }
                }
            }


            stage('BE Dockerimage build') {

                steps {

                    dir('Server/webserver'){
                        sh '''
                        echo 'Dockerimage build'
                        docker build -t docker-springboot:0.0.1 .
                        '''
                    }
                }
            }


            stage('BE Deploy') {

                steps {

                    dir('Server/webserver'){

                        sh '''
                        echo 'Deploy'

                        result=$( docker container ls -a --filter "name=cyes_back" -q )
                        if [ -n "$result" ]; then
                                docker stop $result
                                docker rm $result

                            else
                                echo "No such containers"
                            fi
                        <!-- docker run -d -p 127.0.0.1:5000:5000 -p 1026:5000 --name cyes_back -e JAVA_OPTS="-Duser.timezone=Asia/Seou
                                echo "gogo"

                        docker images -f "dangling=true" -q | xargs -r docker rmi
                        '''
                    }
                }
            }


            //프론트 엔드
            stage('FE build') {
                steps {
                    dir('Front/cyesfront'){
                        sh '''
                            pwd
                            echo 'Frontend build'
                             DEBIAN_FRONTEND=noninteractive apt install -y npm

                            npm install
                            CI=false npm run build
                        '''
                    }
                }
            }
            stage('FE Dockerimage build') {
                steps {
                    dir('Front/cyesfront'){
                        sh '''
~~~~

```
                    echo 'Dockerimage build'
                    docker build --no-cache -t cyes_front:0.0.1 .
                '''
            }
        }
    }
    stage('FE Deploy') {
        steps {
            dir('Front/cyesfront'){

                sh '''
                    echo 'FE Deploy'

                result=$( docker container ls -a --filter "name=cyes_front" -q )
                if [ -n "$result" ]; then
                        docker stop $result
                        docker rm $result

                    else
                        echo "No such containers"
                    fi

                <!-- docker run -d -p 127.0.0.1:9510:80 --name cyes_front cyes_front:0.0.1 -->

                echo "gogo"
                docker images -f "dangling=true" -q | xargs -r docker rmi
                '''
            }
        }
    }


      stage('MSA Container backend') {
            steps {
                dir('/'){

                  script {
            def fileName = 'spring-boot.tar'

            // 파일이 존재하는지 확인
            if (fileExists(fileName)) {
                echo "Deleting ${fileName}"
                // 파일 삭제
                sh "rm ${fileName}"
            } else {
                echo "${fileName} does not exist. Skipping deletion."
            }



        }


        sh '''
                docker save -o spring-boot.tar docker-springboot:0.0.1

                result=$(ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker container ls -a --filter 'name=cyes_back' -q")
                if [ -n "$result" ]; then

                        ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "rm spring-boot.tar"
                        scp -i /jenkins_key /spring-boot.tar ubuntu@k9b103a.p.ssafy.io:/home/ubuntu
                        ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker stop cyes_back"
                        ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker rm cyes_back"
                        ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker rmi docker-springboot:0.0.1"

                    else
                        echo "No such containers"
                    fi

                    ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker load -i spring-boot.tar"
                    ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker run -d -p 127.0.0.1:5000:5000 -p 1026:5000 --name cy
            '''
                }

                }
            }

       stage('MSA Container frontend') {
            steps {
                dir('/'){

        script {
            def fileName = 'cyes_front.tar'
```

~~~
                    // 파일이 존재하는지 확인
                    if (fileExists(fileName)) {
                        echo "Deleting ${fileName}"
                        // 파일 삭제
                        sh "rm ${fileName}"
                    } else {
                        echo "${fileName} does not exist. Skipping deletion."
                    }

                }

                sh '''

                        docker save -o cyes_front.tar cyes_front:0.0.1


                            result=$(ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker container ls -a --filter 'name=cyes_front' -
                        if [ -n "$result" ]; then

                                ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "rm cyes_front.tar"
                                scp -i /jenkins_key /cyes_front.tar ubuntu@k9b103a.p.ssafy.io:/home/ubuntu
                                ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker stop cyes_front"
                                ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker rm cyes_front"
                                ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker rmi cyes_front:0.0.1"

                            else
                                echo "No such containers"
                            fi

                            ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker load -i cyes_front.tar"
                            docker run -d -p 127.0.0.1:9510:80 --name cyes_front cyes_front:0.0.1
                    '''
                        }

                        }
                    }
        }
    }

```

---

### 스프링, 소켓 스프링, 레디스, 프론트 옮기기

```
pipeline {
    agent any

stages {


      // webserver 백엔드
            stage('webserver BE build') {

                steps {

                    dir('Server/webserver'){
                        sh '''
                        pwd
                        echo 'springboot build'
                        chmod +x gradlew
                        ./gradlew clean build -x test
                        echo 'Dockerimage build'
                        docker build -t docker-springboot:0.0.1 .

                        echo 'Deploy'

                        result=$( docker container ls -a --filter "name=cyes_back" -q )
                        if [ -n "$result" ]; then
                                docker stop $result
                                docker rm $result

                            else
                                echo "No such containers"
                            fi

                                echo "gogo"

                        docker images -f "dangling=true" -q | xargs -r docker rmi
                        '''
                    }
                }
            }
~~~

```
        // socketserver 백엔드
        stage('socketserver BE build') {

            steps {

                dir('Server/socketserver'){
                    sh '''
                    pwd
                    echo 'springboot build'
                    chmod +x gradlew
                    ./gradlew clean build -x test
                    '''
                }
            }
        }


        stage('socketserver Dockerimage build') {

            steps {

                dir('Server/socketserver'){
                    sh '''
                    echo 'Dockerimage build'
                    docker build -t socket-springboot:0.0.1 .
                    '''
                }
            }
        }


        stage('socketserver BE Deploy') {

            steps {

                dir('Server/socketserver'){

                    sh '''
                    echo 'Deploy'

                    result=$( docker container ls -a --filter "name=cyes_socket" -q )
                    if [ -n "$result" ]; then
                            docker stop $result
                            docker rm $result

                        else
                            echo "No such containers"
                        fi

                            echo "gogo"

                    docker images -f "dangling=true" -q | xargs -r docker rmi
                    '''
                }
            }
        }


        //프론트 엔드
        stage('FE build') {
            steps {
                dir('Front/cyesfront'){
                    sh '''
                        pwd
                        echo 'Frontend build'
                         DEBIAN_FRONTEND=noninteractive apt install -y npm

                        npm install
                        CI=false npm run build

                        echo 'Dockerimage build'
                        docker build --no-cache -t cyes_front:0.0.1 .

                        echo 'FE Deploy'

                    result=$( docker container ls -a --filter "name=cyes_front" -q )
                    if [ -n "$result" ]; then
                            docker stop $result
                            docker rm $result

                        else
                            echo "No such containers"
                        fi



                    echo "gogo"
                    docker images -f "dangling=true" -q | xargs -r docker rmi
                    '''
                }
            }
        }


          stage('MSA webserver backend') {
                steps {
                    dir('/'){

                // webserver 파일이 존재하는 지 확인

                      script {
                def fileName = 'spring-boot.tar'

                // 파일이 존재하는지 확인
                if (fileExists(fileName)) {
                    echo "Deleting ${fileName}"
                    // 파일 삭제
                    sh "rm ${fileName}"
                } else {
                    echo "${fileName} does not exist. Skipping deletion."
                }

            }

            sh '''

                    docker save -o spring-boot.tar docker-springboot:0.0.1

                    result=$(ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker container ls -a --filter 'name=cyes_back' -q")
                    if [ -n "$result" ]; then

                            ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "rm spring-boot.tar"
                            ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker stop cyes_back"
                            ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker rm cyes_back"
                            ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker rmi docker-springboot:0.0.1"

                        else
                            echo "No such containers"
                        fi
                        scp -i /jenkins_key /spring-boot.tar ubuntu@k9b103a.p.ssafy.io:/home/ubuntu
                        ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker load -i spring-boot.tar"
                        ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker run -d -p 127.0.0.1:5000:5000 -p 1026:5000 --name cy
                '''
                    }

                    }
                }


            stage('MSA socketserver backend') {
                steps {
                    dir('/'){

                // websocket  파일이 존재하는 지 확인

                    script {
                def fileName = 'socket-boot.tar'

                // 파일이 존재하는지 확인
                if (fileExists(fileName)) {
                    echo "Deleting ${fileName}"
                    // 파일 삭제
                    sh "rm ${fileName}"
                } else {
                    echo "${fileName} does not exist. Skipping deletion."
                }

            }


            sh '''

                    docker save -o socket-boot.tar socket-springboot:0.0.1

                    ls -a
```

~~~
                        result=$(ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker container ls -a --filter 'name=cyes_socket' -q"
                        if [ -n "$result" ]; then

                                ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "rm socket-boot.tar"
                                ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker stop cyes_socket"
                                ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker rm cyes_socket"
                                ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker rmi socket-springboot:0.0.1"

                            else
                                echo "No such containers"
                            fi

                            scp -i /jenkins_key /socket-boot.tar ubuntu@k9b103a.p.ssafy.io:/home/ubuntu
                            ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker load -i socket-boot.tar"
                            ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker run -d -p 127.0.0.1:5001:5001 -p 1027:5001 --name cye
                    '''
                        }

                        }
                    }

                 stage('MSA Container frontend') {
                    steps {
                        dir('/'){

                script {
                    def fileName = 'cyes_front.tar'

                    // 파일이 존재하는지 확인
                    if (fileExists(fileName)) {
                        echo "Deleting ${fileName}"
                        // 파일 삭제
                        sh "rm ${fileName}"
                    } else {
                        echo "${fileName} does not exist. Skipping deletion."
                    }

                }

                sh '''

                        docker save -o cyes_front.tar cyes_front:0.0.1


                            result=$(ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker container ls -a --filter 'name=cyes_front'
                        if [ -n "$result" ]; then

                                ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "rm cyes_front.tar"
                                ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker stop cyes_front"
                                ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker rm cyes_front"
                                ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker rmi cyes_front:0.0.1"

                            else
                                echo "No such containers"
                            fi
                            scp -i /jenkins_key /cyes_front.tar ubuntu@k9b103a.p.ssafy.io:/home/ubuntu
                            ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker load -i cyes_front.tar"
                            ssh -i /jenkins_key ubuntu@k9b103a.p.ssafy.io "docker run -d -p 127.0.0.1:9510:80 --name cyes_front cyes_fr
                    '''
                        }

                        }
                    }
        }
    }

```

---

> ### 모놀리식 마스터 nginx default

```
# Default server configuration
server
{
        listen 80 default_server;
        listen [::]:80 default_server;

        server_name k9b103.p.ssafy.io;
        return 308 https://$server_name$request_uri; # HTTP를 HTTPS로 리다이렉트
}

server
~~~

```
{
        listen 11111 default_server;
        listen [::]:11111 default_server;
        server_name k9b103.p.ssafy.io;

        location /metrics {

                stub_status on;

        }

}

server {
    listen 443 ssl default_server;
    listen [::]:443 ssl default_server;

    server_name k9b103.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/k9b103.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/k9b103.p.ssafy.io/privkey.pem;

    location /monitor/ {
        proxy_pass http://localhost:19999/;
        proxy_set_header Host $host;

        # WebSocket 지원을 위한 추가 설정
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
    }
}



` ` `
```