Narang 포팅 메뉴얼

서버 배포 가이드

프로젝트 사용 도구

이슈 관리 : JIRA

형상 관리 : GitLlab

커뮤니케이션 : Notion, Mattermost

코드 리뷰 : Gerrit

디자인 : Figma

UCC :

CI/CD : Jenkins

개발 환경

Backend

openjdk version : 17.0.9

SpringBoot : 3.2.1

IntelliJ : 23.3.2

DB : MongoDB, MySQL

Frontend

vscode : 1.85.1

React : 18.2.0

react-router-dom : 6.21.2

@reduxjs/toolkit ：2.0.1

package.json

```
{
  "name": "frontend",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "lint": "eslint . --ext js,jsx --report-unused-disable-di
    "preview": "vite preview"
  },
  "dependencies": {
    "@charlietango/use-native-lazy-loading": "^1.10.0",
    "@emotion/react": "^11.11.3",
    "@emotion/styled": "^11.11.0",
    "@headlessui/react": "^1.7.18",
    "@heroicons/react": "^2.1.1",
    "@react-google-maps/api": "^2.19.2",
    "@reduxjs/toolkit": "^2.0.1",
    "@stomp/stompjs": "^7.0.0",
    "axios": "^1.6.7",
    "date-fns": "^3.3.1",
    "event-source-polyfill": "^1.0.31",
    "framer-motion": "^11.0.3",
    "jsonwebtoken": "^9.0.2",
    "moment": "^2.30.1",
    "react": "^18.2.0",
    "react-beautiful-dnd": "^13.1.1",
    "react-calendar": "^4.8.0",
    "react-datepicker": "^5.0.0",
    "react-dom": "^18.2.0",
    "react-icons": "^5.0.1",
    "react-intersection-observer": "^9.5.3",
    "react-multi-carousel": "^2.8.4",
    "react-redux": "^9.1.0",
    "react-router-dom": "^6.21.2",
    "react-select": "^5.8.0",
    "react-spinners": "^0.13.8",
```

```
    "redux": "^5.0.1",
    "redux-thunk": "^3.1.0",
    "sockjs-client": "^1.6.1",
    "y-webrtc": "^10.3.0",
    "y-websocket": "^1.5.3",
    "yjs": "^13.6.11"
  },
  "devDependencies": {
    "@types/react": "^18.2.43",
    "@types/react-dom": "^18.2.17",
    "@vitejs/plugin-react": "^4.2.1",
    "autoprefixer": "^10.4.17",
    "eslint": "^8.55.0",
    "eslint-plugin-react": "^7.33.2",
    "eslint-plugin-react-hooks": "^4.6.0",
    "eslint-plugin-react-refresh": "^0.4.5",
    "postcss": "^8.4.33",
    "sass": "^1.70.0",
    "tailwindcss": "^3.4.1",
    "vite": "^5.0.8"
  }
}
```

서버사양

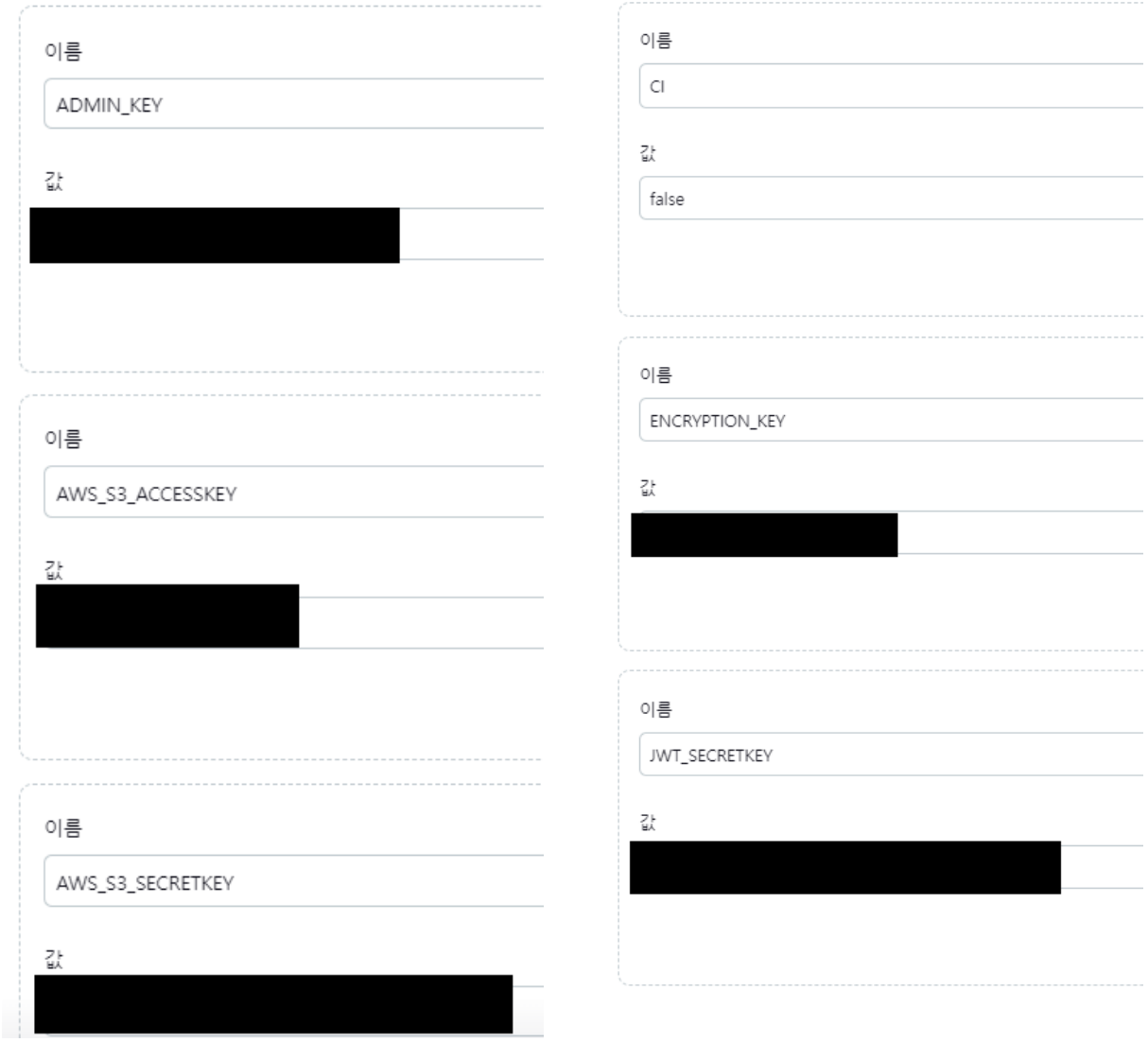
- AWS Lightsail $80 플랜
  - CPU : 쿼드코어
  - RAM: 16GB
  - SSD : 320GB
  - Traffic: 6TB/month (Upload/Download 합산)
- S3

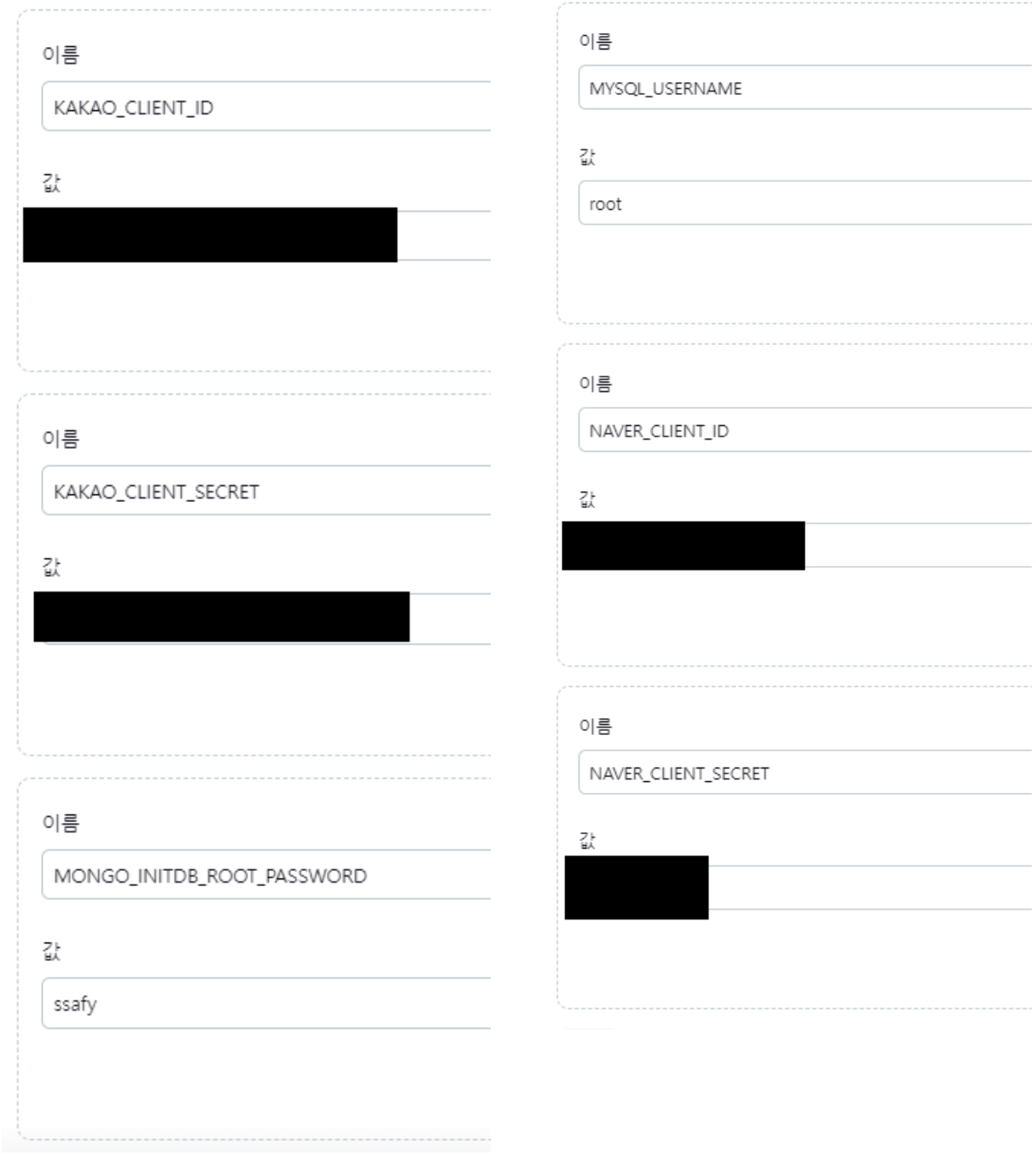
환경변수

.env

```
VITE_GOOGLEMAP_API_KEY=
VITE_PLACES_API_KEY=
VITE_KAKAO_CLIENT_ID=
VITE_KAKAO_REDIRECT_URI=https://i10a701.p.ssafy.io/login/oaut
VITE_NAVER_CLIENT_ID=
VITE_NAVER_REDIRECT_URI=https://i10a701.p.ssafy.io/login/oaut
VITE_NAVER_STATE=

VITE_TRIP_REQUEST_URI=https://i10a701.p.ssafy.io/api/trip
VITE_CHAT_REQUEST_URI=https://i10a701.p.ssafy.io/api/message/
VITE_ALERT_REQUEST_URI=https://i10a701.p.ssafy.io/api/message
VITE_USER_REQUEST_URI=https://i10a701.p.ssafy.io/api/user
VITE_PLAN_REQUEST_URI=https://i10a701.p.ssafy.io/api/plan
VITE_PAYMENT_REQUEST_URI=https://i10a701.p.ssafy.io/api/payme
VITE_REQUEST_API=https://i10a701.p.ssafy.io
```

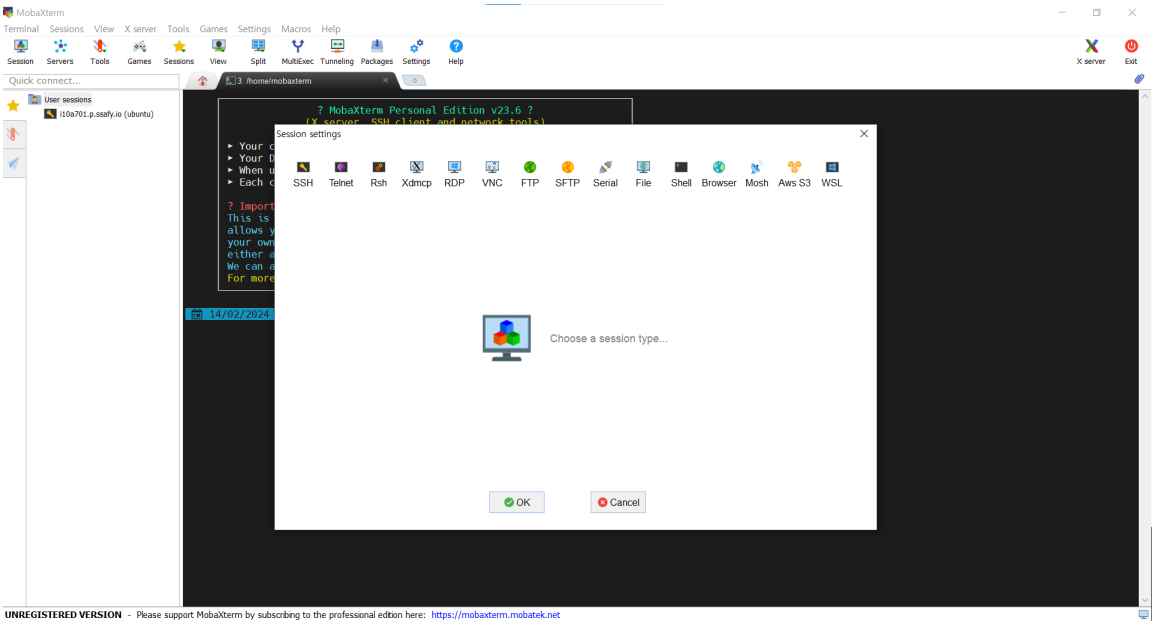
jenkins 환경변수


이름
ADMIN_KEY
값
[redacted]
이름
CI
값
false
이름
AWS_S3_ACCESSKEY
값
[redacted]
이름
ENCRYPTION_KEY
값
[redacted]
이름
AWS_S3_SECRETKEY
값
[redacted]
이름
JWT_SECRETKEY
값
[redacted]

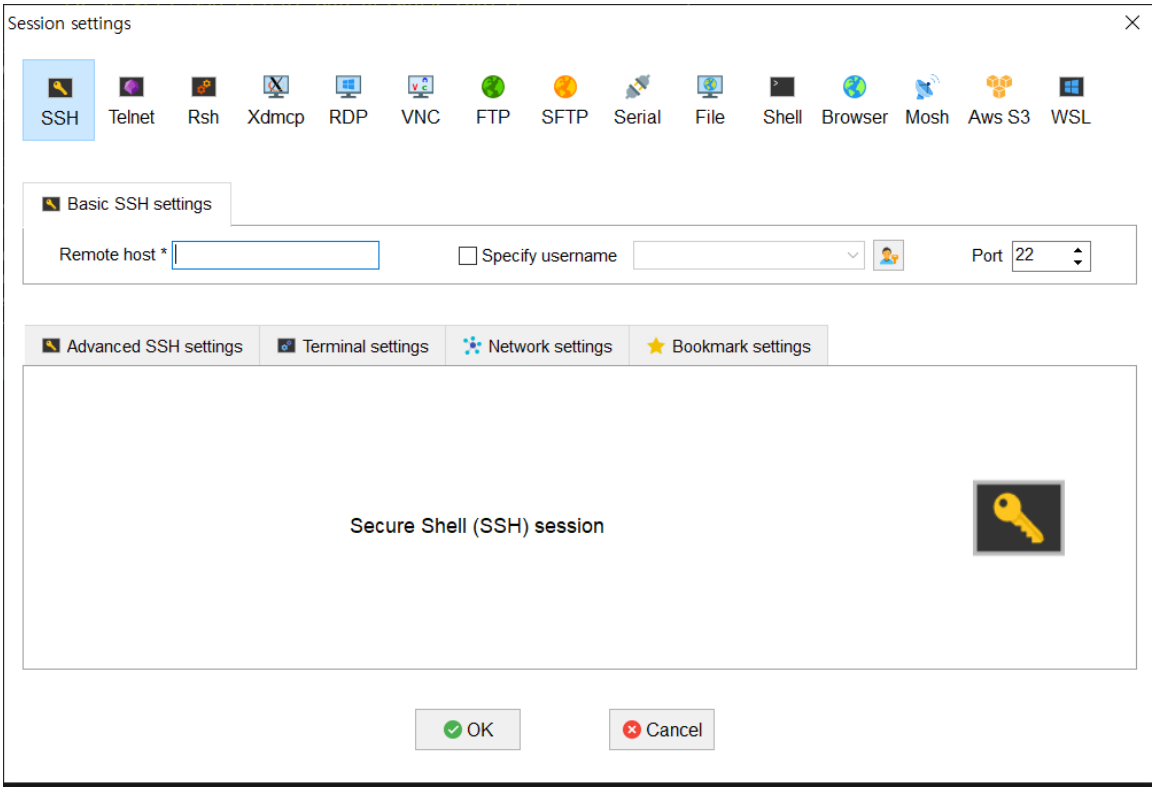

이름
KAKAO_CLIENT_ID
값
[redacted]
이름
KAKAO_CLIENT_SECRET
값
[redacted]
이름
MONGO_INITDB_ROOT_PASSWORD
값
ssafy
이름
MYSQL_USERNAME
값
root
이름
NAVER_CLIENT_ID
값
[redacted]
이름
NAVER_CLIENT_SECRET
값
[redacted]

EC2 서버 접속

- MobaXterm 사용 접속



- SSH 인증서 사용으로 접속 권환 획득



도커 설치

yum으로 도커 설치

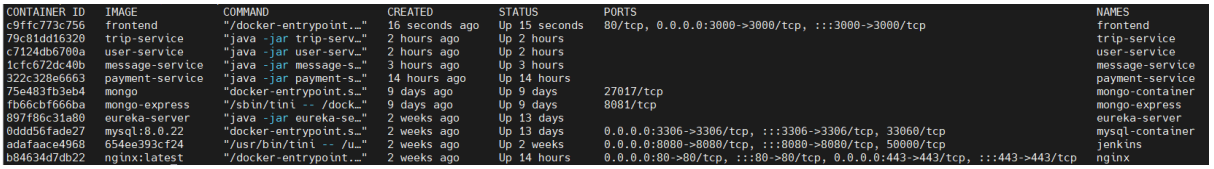
```
sudo yum install docker -y
```

설치한 Docker 버전

```
Docker version 25.0.1, build 29cf629
```

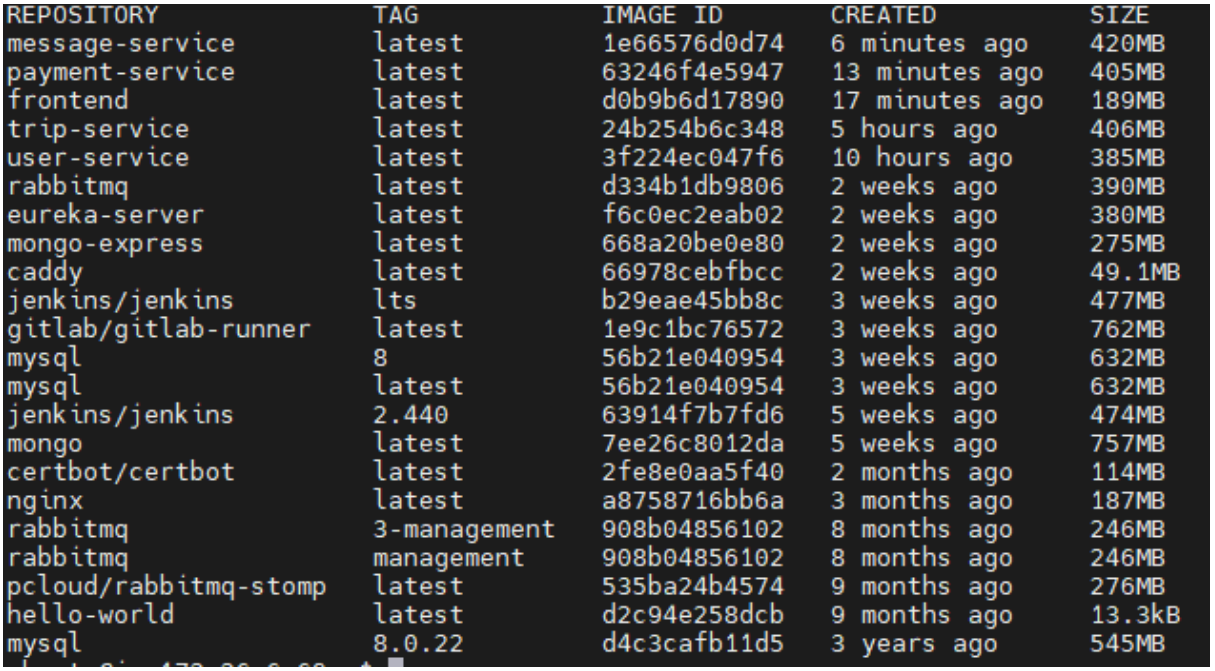
도커 컨테이너들 설치

도커 내부 컨테이너들 상태



```
CONTAINER ID   IMAGE             COMMAND                  CREATED          STATUS          PORTS                                                                                NAMES
c9ffc773c756   frontend          "/docker-entrypoint.…"   16 seconds ago   Up 15 seconds   80/tcp, 0.0.0.0:3000->3000/tcp, :::3000->3000/tcp                                    frontend
79c81dd16320   trip-service      "java -jar trip-serv…"   2 hours ago      Up 2 hours                                                                                           trip-service
c7124db6700a   user-service      "java -jar user-serv…"   2 hours ago      Up 2 hours                                                                                           user-service
1cfc672dc40b   message-service   "java -jar message-s…"   3 hours ago      Up 3 hours                                                                                           message-service
322c328e6663   payment-service   "java -jar payment-s…"   14 hours ago     Up 14 hours                                                                                          payment-service
75e483fb3eb4   mongo             "docker-entrypoint.s…"   9 days ago       Up 9 days       27017/tcp                                                                            mongo-container
fb66cbf666ba   mongo-express     "/sbin/tini -- /dock…"   9 days ago       Up 9 days       8081/tcp                                                                             mongo-express
897f86c31a80   eureka-server     "java -jar eureka-se…"   2 weeks ago      Up 13 days                                                                                           eureka-server
0ddd56fade27   mysql:8.0.22      "docker-entrypoint.s…"   2 weeks ago      Up 13 days      0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp                                 mysql-container
adafaace4968   654ee393cf24      "/usr/bin/tini -- /u…"   2 weeks ago      Up 2 weeks      0.0.0.0:8080->8080/tcp, :::8080->8080/tcp, 50000/tcp                                 jenkins
b84634d7db22   nginx:latest      "/docker-entrypoint.…"   2 weeks ago      Up 14 hours     0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0.0.0:443->443/tcp, :::443->443/tcp             nginx
```

```
docker images | grep -v "<none>" 결과
```



```
REPOSITORY                TAG              IMAGE ID       CREATED          SIZE
message-service           latest           1e66576d0d74   6 minutes ago    420MB
payment-service           latest           63246f4e5947   13 minutes ago   405MB
frontend                  latest           d0b9b6d17890   17 minutes ago   189MB
trip-service              latest           24b254b6c348   5 hours ago      406MB
user-service              latest           3f224ec047f6   10 hours ago     385MB
rabbitmq                  latest           d334b1db9806   2 weeks ago      390MB
eureka-server             latest           f6c0ec2eab02   2 weeks ago      380MB
mongo-express             latest           668a20be0e80   2 weeks ago      275MB
caddy                     latest           66978cebfbcc   2 weeks ago      49.1MB
jenkins/jenkins           lts              b29eae45bb8c   3 weeks ago      477MB
gitlab/gitlab-runner      latest           1e9c1bc76572   3 weeks ago      762MB
mysql                     8                56b21e040954   3 weeks ago      632MB
mysql                     latest           56b21e040954   3 weeks ago      632MB
jenkins/jenkins           2.440            63914f7b7fd6   5 weeks ago      474MB
mongo                     latest           7ee26c8012da   5 weeks ago      757MB
certbot/certbot           latest           2fe8e0aa5f40   2 months ago     114MB
nginx                     latest           a8758716bb6a   3 months ago     187MB
rabbitmq                  3-management     908b04856102   8 months ago     246MB
rabbitmq                  management       908b04856102   8 months ago     246MB
pcloud/rabbitmq-stomp     latest           535ba24b4574   9 months ago     276MB
hello-world               latest           d2c94e258dcb   9 months ago     13.3kB
mysql                     8.0.22           d4c3cafb11d5   3 years ago      545MB
```

nginx 컨테이너 설치

/etc/nginx/nginx.conf

```
user  nginx;
worker_processes  auto;
```

```
error_log  /var/log/nginx/error.log debug;
pid        /var/run/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_loc
                      '$status $body_bytes_sent "$http_refere
                      '"$http_user_agent" "$http_x_forwarded_

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;

    include /etc/nginx/conf.d/*.conf;
}
```

/etc/nginx/conf.d/default.conf

```
map $http_upgrade $connection_upgrade {
        default upgrade;
        '' close;
}
server {
    #server_name localhost;
```

```
    server_name  i10a701.p.ssafy.io;

    #access_log  /var/log/nginx/host.access.log  main;
    error_log /var/log/nginx/error.log;

    include /etc/nginx/conf.d/service-url.inc;

    location / {
        proxy_pass http://frontend:3000;
        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 /mongodb {
         proxy_pass http://mongo-express:8081;
         proxy_set_header Host $host;
         proxy_set_header X-Real-IP $remote_addr;
         proxy_set_header X-Forwarded-For $proxy_add_x_forwar
         proxy_set_header X-Forwarded-Proto $scheme;
    }


    location ~ ^/(swagger|webjars|configuration|swagger-resou
         proxy_pass $service_url;
         proxy_set_header Host $host;
         proxy_set_header X-Real-IP $remote_addr;
         proxy_set_header X-Forwarded-For $proxy_add_x_forwar
         proxy_set_header X-Forwarded-Proto $scheme;
    }

    #error_page  404              /404.html;

    # redirect server error pages to the static page /50x.htm
    #
    error_page   500 502 503 504  /50x.html;
```

```
    location = /50x.html {
        root   /usr/share/nginx/html;
    }

    # proxy the PHP scripts to Apache listening on 127.0.0.1:
    #
    #location ~ \.php$ {
    #    proxy_pass   http://127.0.0.1;
    #}

    # pass the PHP scripts to FastCGI server listening on 127
    #
    #location ~ \.php$ {
    #    root           html;
    #    fastcgi_pass   127.0.0.1:9000;
    #    fastcgi_index  index.php;
    #    fastcgi_param  SCRIPT_FILENAME  /scripts$fastcgi_scr
    #    include        fastcgi_params;
    #}

    # deny access to .htaccess files, if Apache's document ro
    # concurs with nginx's one
    #
    #location ~ /\.ht {
    #    deny  all;
    #}

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/i10a701.p.ssafy.io/
    ssl_certificate_key /etc/letsencrypt/live/i10a701.p.ssafy
    include /etc/letsencrypt/options-ssl-nginx.conf; # manage
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed

    location /eureka {
        proxy_pass http://eureka-server:8761;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
```

```
        proxy_set_header X-Forwarded-For $proxy_add_x_forward
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /jenkins {
        proxy_pass http://jenkins:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forward
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location ~ ^/(api/trip|api/plan) {
        proxy_pass http://trip-service:8081;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forward
        proxy_set_header X-Forwarded-Proto $scheme;
    }

     location /api/message {
        proxy_pass http://message-service:8084;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forward
        proxy_set_header X-Forwarded-Proto $scheme;

    }

     location /api/message/chat {
        proxy_pass http://message-service:8084;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forward
        proxy_set_header X-Forwarded-Proto $scheme;
```

```
      }

      location /api/message/alert/subscribe {
         proxy_pass http://message-service:8084;
         proxy_http_version 1.1;
         proxy_set_header Host $host;
         proxy_set_header X-Real-IP $remote_addr;
         proxy_set_header X-Forwarded-For $proxy_add_x_forward
         proxy_set_header X-Forwarded-Proto $scheme;
         proxy_set_header Connection '';
         proxy_buffering off;
         proxy_cache off;
         chunked_transfer_encoding off;
         proxy_read_timeout 24h;
      }

      location /api/user {
         proxy_pass http://user-service:8081;
         proxy_set_header Host $host;
         proxy_set_header X-Real-IP $remote_addr;
         proxy_set_header X-Forwarded-For $proxy_add_x_forward
         proxy_set_header X-Forwarded-Proto $scheme;
      }

      location /api/payment {
         proxy_pass http://payment-service:8082;
         proxy_set_header Host $host;
         proxy_set_header X-Real-IP $remote_addr;
         proxy_set_header X-Forwarded-For $proxy_add_x_forward
         proxy_set_header X-Forwarded-Proto $scheme;
      }
}
server {
    if ($host = i10a701.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot
```

```
    listen       80;
    listen  [::]:80;
    server_name  i10a701.p.ssafy.io;
    return 404; # managed by Certbot
}
```

젠킨스 컨테이너 설치

도커 이미지 받아오기

```
docker pull jenkins/jenkins:lts
```

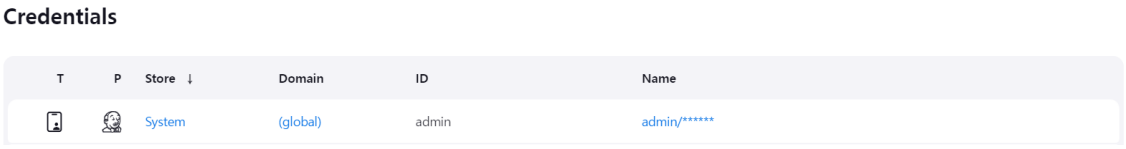
젠킨스 컨테이너 설치

```
docker run -d -p 8080:8080 -v /var/jenkins_home:/var/jenkins_
```

admin password 얻어와서 마저 젠킨스 설치

```
docker logs jenkins
```

젠킨스 credentials 발급



젠킨스 관리 → plugins → SSH Build Agent plugin 설치 → credentials

젠킨스 파이프라인 설정

narang-payment

jenkins pipeline

```
pipeline {
    agent any

    environment {
        service = "payment-service"
```

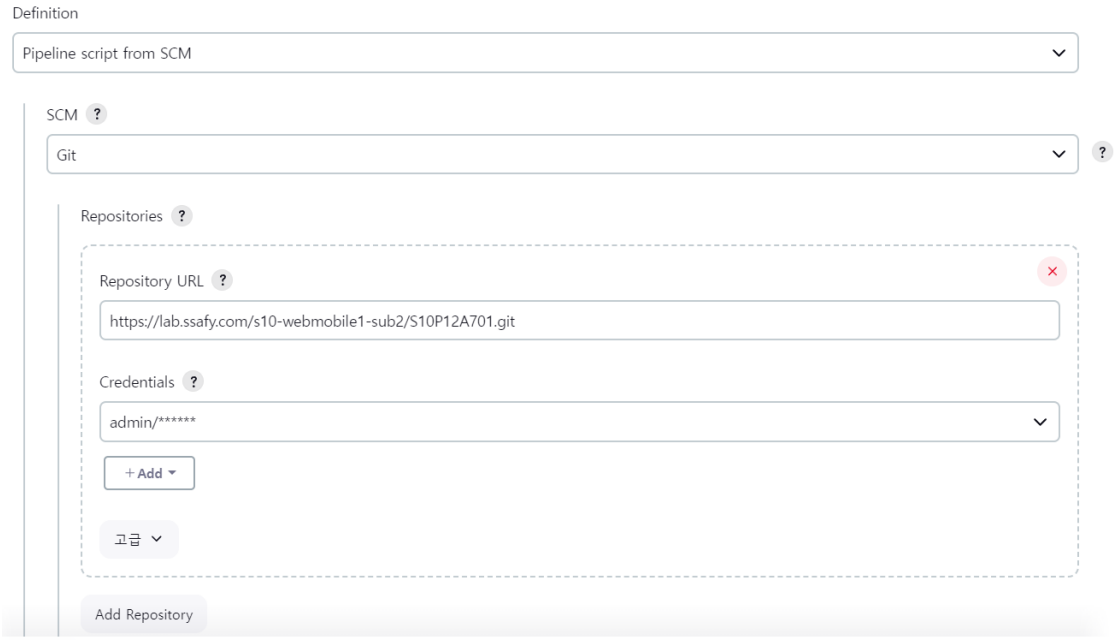
```
    }

    tools {
        // 여기서 뒤 이름을 우리가 젠킨스에서 설정하는 것.
        gradle 'Gradle'
    }
    stages {
        stage('Checkout') {
            steps {
                // 소스 코드 체크아웃
                checkout scm
            }
        }
        stage('Jar Build') {
            steps {
                // Gradle을 사용하여 ${service} 프로젝트 빌드
                sh 'cd backend/${service} && chmod +x gradlew
                // sh 'cd backend/${service} && chmod +x grad
            }
        }
        stage('Docker Image Build') {
            steps {
                // 빌드된 JAR 파일 도커 이미지로 생성
                // sh('docker ps -a | grep '${service}' | awk

                // message (Port 8084)
                // sh('(docker ps -a | grep '8084' | awk \'{p
                // sh('chmod +x ./jenkins/port-refresh-messag

                // 이미지 생성 중 . . .
                echo 'No ${service} running . . . About to Bu
                sh 'docker build -t ${service} -f ./backend/$

                // 메세지 서비스 네트워크 연결 끊고 종료
                echo 'Removing Existing Containers . . . Wait
                sh '(docker network disconnect Narang ${servi
                sh 'docker rm ${service} | true'
```

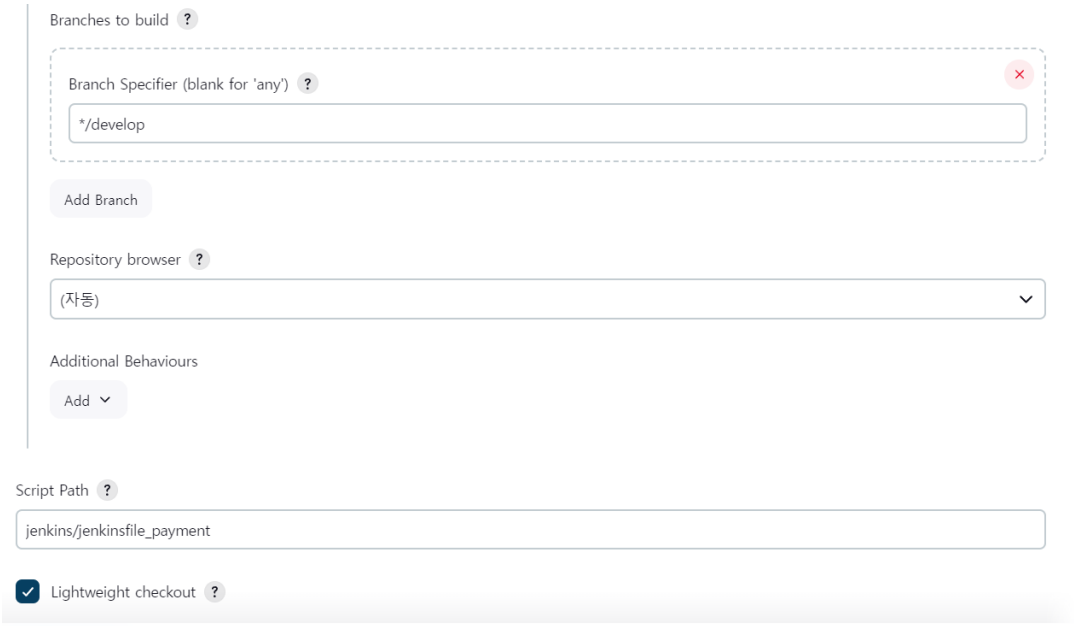
```
                // 해당 이미지로 컨테이너 생성 중 . . .
                sh 'docker run -it --name ${service} --net Na
            }
        }
    }
}
```

Dockerfile

```
# Dockerfile

# 사용할 기본 이미지를 선택합니다.
FROM openjdk:17-alpine

# 작업 디렉토리를 지정합니다.
ARG JAR_FILE=./backend/payment-service/build/libs/payment-ser

# 호스트의 JAR 파일을 컨테이너의 작업 디렉토리로 복사합니다.
COPY ${JAR_FILE} ./payment-service.jar

# 컨테이너에서 수행될 명령어를 지정합니다.
ENTRYPOINT ["java","-jar", "payment-service.jar"]
```

젠킨스 설정





narang-user

jenkins pipeline

```
pipeline {
    agent any

    environment {
        service = "user-service"
    }

    tools {
        // 여기서 뒤 이름을 우리가 젠킨스에서 설정하는 것.
        gradle 'Gradle'
    }
    stages {
        stage('Checkout') {
            steps {
                // 소스 코드 체크아웃
                checkout scm
            }
        }
        stage('Jar Build') {
            steps {
                // Gradle을 사용하여 ${service} 프로젝트 빌드
```

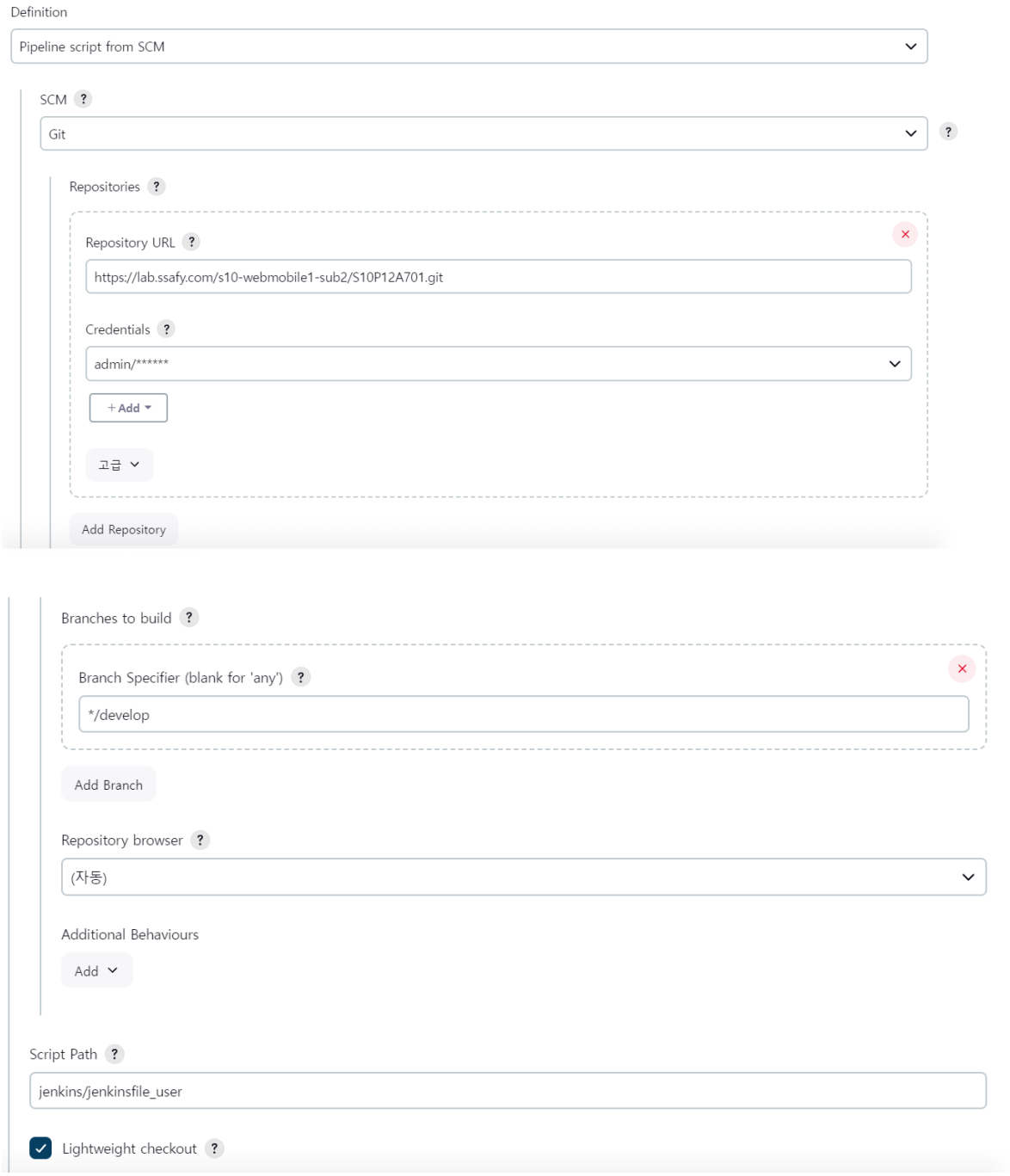
```
                sh 'cd backend/${service} && chmod +x gradlew
                // sh 'cd backend/${service} && chmod +x grad
            }
        }
        stage('Docker Image Build') {
            steps {
                // 빌드된 JAR 파일 도커 이미지로 생성
                // sh('docker ps -a | grep '${service}' | awk

                // message (Port 8084)
                // sh('(docker ps -a | grep '8084' | awk \'{p
                // sh('chmod +x ./jenkins/port-refresh-messag

                // 이미지 생성 중 . . .
                echo 'No ${service} running . . . About to Bu
                sh 'docker build -t ${service} -f ./backend/$

                // 메세지 서비스 네트워크 연결 끊고 종료
                echo 'Removing Existing Containers . . . Wait
                sh '(docker network disconnect Narang ${servi
                sh 'docker rm ${service} | true'

                // 해당 이미지로 컨테이너 생성 중 . . .
                sh 'docker run -it --name ${service} --net Na
            }
        }
    }
}
```

Dockerfile

```
# Dockerfile

# 사용할 기본 이미지를 선택합니다.
FROM openjdk:17-alpine

# 작업 디렉토리를 지정합니다.
ARG JAR_FILE=./backend/user-service/build/libs/user-service-0
```

```
# 호스트의 JAR 파일을 컨테이너의 작업 디렉토리로 복사합니다.
COPY ${JAR_FILE} ./user-service.jar

# 컨테이너에서 수행될 명령어를 지정합니다.
ENTRYPOINT ["java","-jar", "user-service.jar"]
```

젠킨스 설정



narang_frontend

jenkins pipeline

```
pipeline {
    agent any
    tools {nodejs "nodejs-21.6.1"}

    environment {
        releaseServerAccount = 'ubuntu'
        releaseServerUri = 'i10a701.p.ssafy.io'
        VITE_GOOGLEMAP_API_KEY='AIzaSyA_FWc_9IricQuFYYctKYV-c
        VITE_PLACES_API_KEY='AIzaSyAyxRcWvLITB6iDmq_PaSrZcLTt
        VITE_KAKAO_CLIENT_ID='ac9bb9c5ad229907446e706fe6cc47b
        VITE_KAKAO_REDIRECT_URI='https://i10a701.p.ssafy.io/l
        VITE_NAVER_CLIENT_ID='_cFgoA5jCXN10jJ6pAoo'
        VITE_NAVER_REDIRECT_URI='https://i10a701.p.ssafy.io/l
        VITE_NAVER_STATE='74r0jJN6uX'

        VITE_TRIP_REQUEST_URI='https://i10a701.p.ssafy.io/api
        VITE_CHAT_REQUEST_URI='https://i10a701.p.ssafy.io/api
        VITE_ALERT_REQUEST_URI='https://i10a701.p.ssafy.io/ap
        VITE_USER_REQUEST_URI='https://i10a701.p.ssafy.io/api
        VITE_PLAN_REQUEST_URI='https://i10a701.p.ssafy.io/api
        VITE_PAYMENT_REQUEST_URI='https://i10a701.p.ssafy.io/
        VITE_REQUEST_API='https://i10a701.p.ssafy.io'

    }
    stages {
        stage('Checkout') {
            steps {
                // 소스 코드 체크아웃
                checkout scm
            }
        }
        stage('node Build') {
            steps {
                script {
                    // npm 이용 frontend 프로젝트 빌드
```

```
                    dir("frontend/") {
                        sh "npm install"
                        // package.json 기반 필요 모듈 설치
                        sh "npm run build"
                        // 빌드
                    }

                }
            }
        }
        stage('Docker Image Build') {
            steps {
                // 도커 컨테이너에 프론트앤드 있으면 중지
                sh 'docker stop frontend || true'
                // 도커 컨테이너에 프론트앤드 있으면 삭제
                sh 'docker rm frontend || true'
                //
                sh 'docker build -t frontend -f ./frontend/Do
                // 배포 스크립트 실행 (예: Docker 컨테이너에 배포)
                sh 'docker run --name frontend -p 3000:3000 -


            }
        }
        // stage('Restart Nginx') {
        //     steps {
        //          // 도커 nginx 재시작
        //         sh 'docker restart nginx'
        //     }
        // }
    }
}
```

Dockerfile

```
FROM nginx:latest

# root에 /app 폴더 생성
```

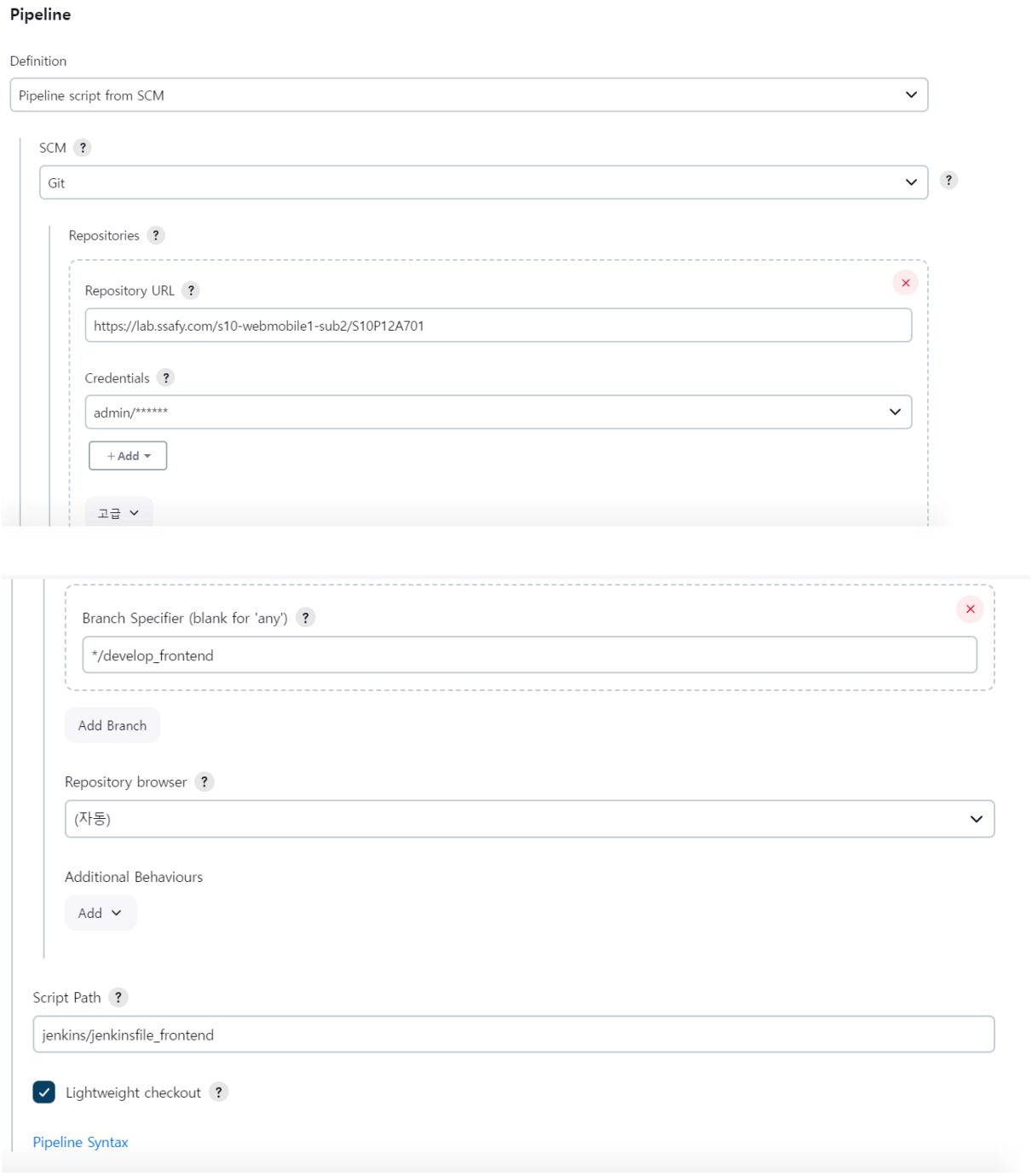
```
RUN mkdir /app

# work dir 고정
WORKDIR /app

# work dir에 build 폴더 생성
RUN mkdir ./build

# host pc의 현재 경로의 build 폴더를 work dir의 build 폴더로 복사
ADD ./frontend/dist ./build

# nginx 설정 파일 복사
COPY frontend/nginx.conf /etc/nginx/conf.d/default.conf

# 3000 포트 개방
EXPOSE 3000

# # container 실행 시 자동으로 실행할 command. nginx 시작함
CMD ["nginx", "-g", "daemon off;"]
```

frontend 컨테이너 내부 nginx 설정

nginx.conf

```
server {
    listen 3000;
    location / {
        root /app/build;
        index index.html;
        try_files $uri $uri/ /index.html;
    }
}
# 도커 fe 컨테이너 내부 nginx 구동
# 컨테이너 내 3000번 포트 사용
# /app/build/index 혹은 /app/build/index.html 실행
```

젠킨스 설정



narang_message

jenkins pipeline

```
pipeline {
    agent any

    environment {
        service = "message-service"
```

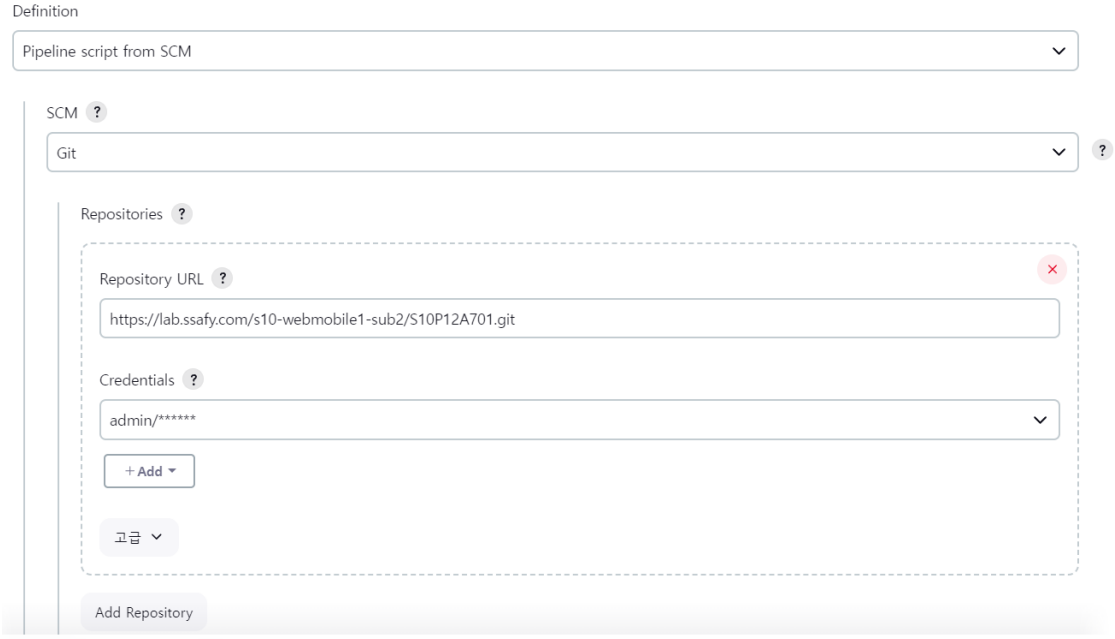
```
    }

    tools {
        // 여기서 뒤 이름을 우리가 젠킨스에서 설정하는 것.
        gradle 'Gradle'
    }
    stages {
        stage('Checkout') {
            steps {
                // 소스 코드 체크아웃
                checkout scm
            }
        }
        stage('Jar Build') {
            steps {
                // Gradle을 사용하여 ${service} 프로젝트 빌드
                sh 'cd backend/${service} && chmod +x gradlew
                // sh 'cd backend/${service} && chmod +x grad
            }
        }
        stage('Docker Image Build') {
            steps {
                // 빌드된 JAR 파일 도커 이미지로 생성
                // sh('docker ps -a | grep '${service}' | awk

                // message (Port 8084)
                // sh('(docker ps -a | grep '8084' | awk \'{p
                // sh('chmod +x ./jenkins/port-refresh-messag

                // 이미지 생성 중 . . .
                echo 'No ${service} running . . . About to Bu
                sh 'docker build -t ${service} -f ./backend/$

                // 메세지 서비스 네트워크 연결 끊고 종료
                echo 'Removing Existing Containers . . . Wait
                sh '(docker network disconnect Narang ${servi
                sh 'docker rm ${service} | true'
```

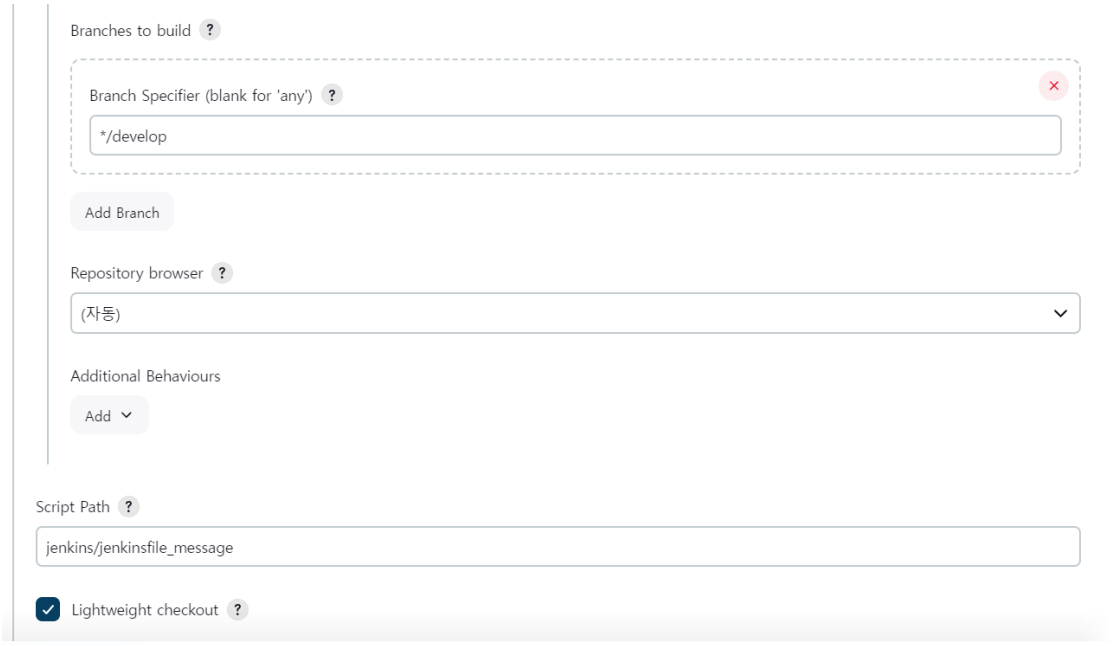
```
                // 해당 이미지로 컨테이너 생성 중 . . .
                sh 'docker run -it --name ${service} --net Na
            }
        }

    }
}
```

Dockerfile

```
# Dockerfile

# 사용할 기본 이미지를 선택합니다.
FROM openjdk:17-alpine

# 작업 디렉토리를 지정합니다.
ARG JAR_FILE=./backend/message-service/build/libs/message-ser

# 호스트의 JAR 파일을 컨테이너의 작업 디렉토리로 복사합니다.
COPY ${JAR_FILE} ./message-service.jar

# 컨테이너에서 수행될 명령어를 지정합니다.
ENTRYPOINT ["java","-jar", "message-service.jar"]
```

젠킨스 설정





narang_trip

jenkins pipeline

```
pipeline {
    agent any

    environment {
        service = "trip-service"
    }
```

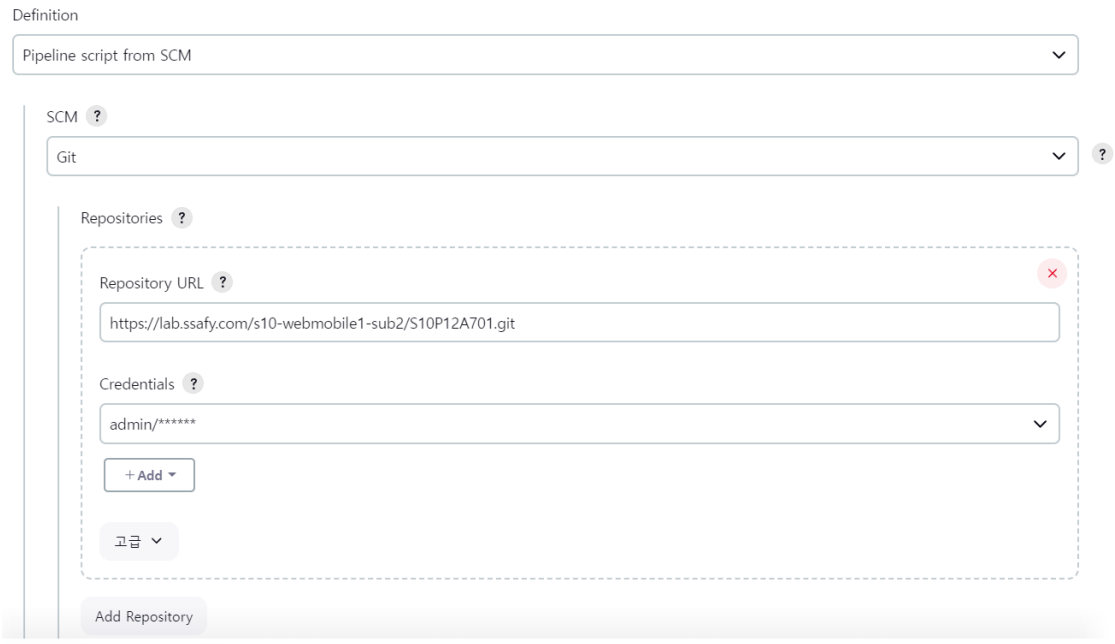
```
    tools {
        // 여기서 뒤 이름을 우리가 젠킨스에서 설정하는 것.
        gradle 'Gradle'
    }
    stages {
        stage('Checkout') {
            steps {
                // 소스 코드 체크아웃
                checkout scm
            }
        }
        stage('Jar Build') {
            steps {
                // Gradle을 사용하여 ${service} 프로젝트 빌드
                sh 'cd backend/${service} && chmod +x gradlew
                // sh 'cd backend/${service} && chmod +x grad
            }
        }
        stage('Docker Image Build') {
            steps {
                // 빌드된 JAR 파일 도커 이미지로 생성
                // sh('docker ps -a | grep '${service}' | awk

                // message (Port 8084)
                // sh('(docker ps -a | grep '8084' | awk \'{p
                // sh('chmod +x ./jenkins/port-refresh-messag

                // 이미지 생성 중 . . .
                echo 'No ${service} running . . . About to Bu
                sh 'docker build -t ${service} -f ./backend/$

                // 메세지 서비스 네트워크 연결 끊고 종료
                echo 'Removing Existing Containers . . . Wait
                sh '(docker network disconnect Narang ${servi
                sh 'docker rm ${service} | true'

                // 해당 이미지로 컨테이너 생성 중 . . .
```

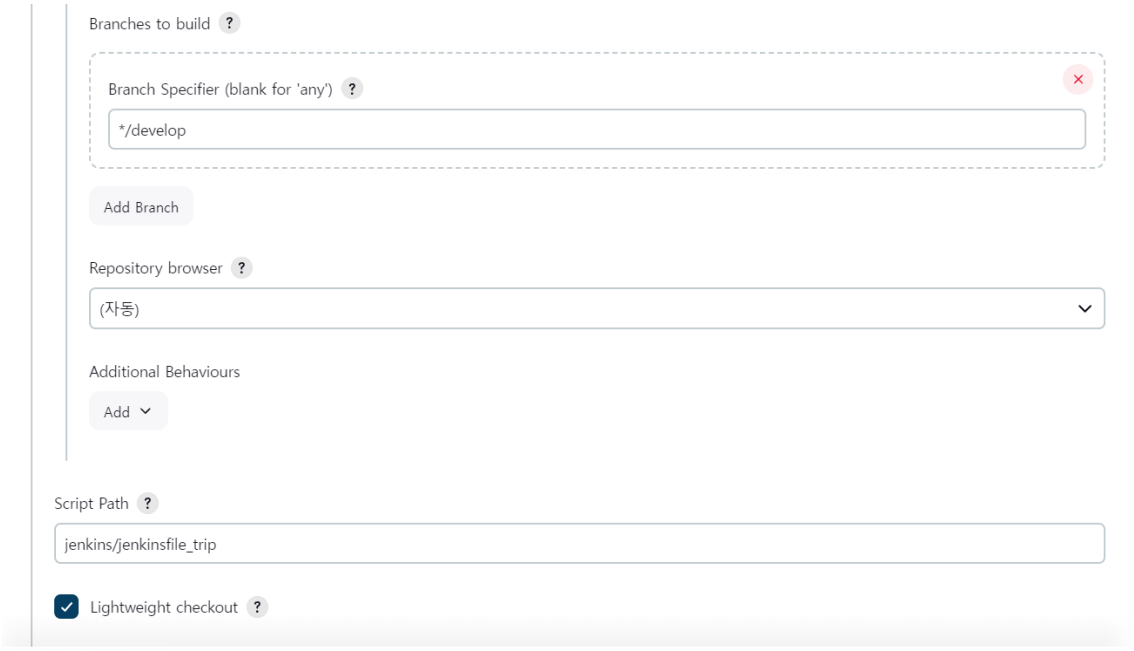
```
                    sh 'docker run -it --name ${service} --net Na
                }
            }
        }
    }
```

Dockerfile

```
# Dockerfile

# 사용할 기본 이미지를 선택합니다.
FROM openjdk:17-alpine

# 작업 디렉토리를 지정합니다.
ARG JAR_FILE=./backend/trip-service/build/libs/trip-service-0

# 호스트의 JAR 파일을 컨테이너의 작업 디렉토리로 복사합니다.
COPY ${JAR_FILE} ./trip-service.jar

# 컨테이너에서 수행될 명령어를 지정합니다.
ENTRYPOINT ["java","-jar", "trip-service.jar"]
```

젠킨스 설정





프로젝트 외부 서비스

- 소셜 로그인
  - 카카오 로그인 API
    - https://developers.kakao.com/docs/latest/ko/kakaologin/js
  - 네이버 로그인 API
    - https://developers.naver.com/products/login/api/api.md
  - JWT
    - https://jwt.io/introduction
- 카카오페이
  - https://developers.kakaopay.com/
- 구글 맵 API
  - https://developers.google.com/maps?hl=ko