포팅 매뉴얼

Settings

Version

Name	Detail(Version)
Java	zulu 17.0.10
Spring Boot	3.2.5
Postgresql	16.2
Gradle	8.7
ELK	8.12.2
Spring Data JPA	3.2.5
Redis	2.7

Name	Detail(Version)
Spring Cloud	2023.0.1
Intelij Ultimate	2023.3.4

Port

Service	External Port	Internal Port
Nginx	443(80)	443(80)
Spring Cloud Gateway	-	8000
Eureka	-	8761
postgres	5432	5432
Redis	6379	6379
MongoDB	47017	27017
RabbitMQ - AMQP	4885	5672
RabbitMQ - UI	8672	15672
RabbitMQ - STOMP	61613	61613
jenkins	8080	8080
Sonarqube	8081	8081
chatbot-service	-	8080
openchat-service	-	8080
auth-service	-	8443
ai-service	8085	8085

Docker Containers

Nginx

```
sudo docker exec -it nginx bash
```

```
$ apt-get update

$ apt-get install vim
```

```
$ apt-get install certbot python3-certbot-nginx
```

```
$ vim /etc/nginx/conf.d/default.conf
```

```
server {
    listen       80;

    # 수정
    server_name  forteams.co.kr;

    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;
    }

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

    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }
}
```

```
certbot —-nginx -d forteams.co.kr
```

default.conf (final)

```
server {
    listen 80;
    server_name forteams.co.kr www.forteams.co.kr;
    location / {
```

```
        return 301 https://$host$request_uri;
    }
}

server {
    listen 443 ssl;
    server_name forteams.co.kr www.forteams.co.kr;

    ssl_certificate /etc/letsencrypt/live/forteams.co.kr/fullchain.p
em;
    ssl_certificate_key /etc/letsencrypt/live/forteams.co.kr/privke
y.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    location / {
        proxy_pass http://front-container:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-Proto $scheme;

        add_header 'Access-Control-Allow-Origin' 'https://forteams.c
o.kr' always;
        add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTION
S, PUT, DELETE' always;
        add_header 'Access-Control-Allow-Headers' 'Origin, X-Request
ed-With, Content-Type, Accept, Authorization' always;
        add_header 'Access-Control-Allow-Credentials' 'true' always;

        if ($request_method = 'OPTIONS') {
            add_header 'Access-Control-Allow-Origin' 'https://fortea
ms.co.kr' always;
            add_header 'Access-Control-Allow-Methods' 'GET, POST, OP
TIONS, PUT, DELETE' always;
            add_header 'Access-Control-Allow-Headers' 'Origin, X-Req
uested-With, Content-Type, Accept, Authorization' always;
            add_header 'Access-Control-Allow-Credentials' 'true' alw
ays;
            add_header 'Content-Length' 0;
            add_header 'Content-Type' 'text/plain';
            return 204;
        }
    }
```

```
    location /oauth2/authorization/microsoft {
        proxy_pass http://gateway:8000;
        proxy_http_version 1.1;
        proxy_set_header Connection "";
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /login/oauth2/code/microsoft {
        proxy_pass http://gateway:8000;
        proxy_http_version 1.1;
        proxy_set_header Connection "";
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

location /api/v1 {
    proxy_pass http://gateway:8000;
    proxy_http_version 1.1;
    proxy_set_header Connection "";
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;

    add_header 'Access-Control-Allow-Origin' 'https://forteams.co.k
r' always;
    add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS, P
UT, DELETE' always;
    add_header 'Access-Control-Allow-Headers' 'Origin, X-Requested-W
ith, Content-Type, Accept, Authorization' always;
    add_header 'Access-Control-Allow-Credentials' 'true' always;

    if ($request_method = 'OPTIONS') {
        add_header 'Access-Control-Allow-Origin' 'https://forteams.c
o.kr' always;
        add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTION
S, PUT, DELETE' always;
        add_header 'Access-Control-Allow-Headers' 'Origin, X-Request
```

```
ed-With, Content-Type, Accept, Authorization' always;
        add_header 'Access-Control-Allow-Credentials' 'true' always;
        add_header 'Content-Length' 0;
        add_header 'Content-Type' 'text/plain';
        return 204;
    }
}

location /api/ws {
    proxy_pass http://gateway:8000;
    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_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;

    add_header 'Access-Control-Allow-Origin' 'https://forteams.co.k
r' always;
    add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS, P
UT, DELETE' always;
    add_header 'Access-Control-Allow-Headers' 'Origin, X-Requested-W
ith, Content-Type, Accept, Authorization' always;
    add_header 'Access-Control-Allow-Credentials' 'true' always;

    if ($request_method = 'OPTIONS') {
        add_header 'Access-Control-Allow-Origin' 'https://forteams.c
o.kr' always;
        add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTION
S, PUT, DELETE' always;
        add_header 'Access-Control-Allow-Headers' 'Origin, X-Request
ed-With, Content-Type, Accept, Authorization' always;
        add_header 'Access-Control-Allow-Credentials' 'true' always;
        add_header 'Content-Length' 0;
        add_header 'Content-Type' 'text/plain';
        return 204;
    }
}

}

server {
    if ($host = www.forteams.co.kr) {
```

```
        return 301 https://$host$request_uri;
    } # managed by Certbot

    if ($host = forteams.co.kr) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    server_name forteams.co.kr www.forteams.co.kr;
    return 404; # managed by Certbot
}
```

- Purple : nextjs 라우팅
- Green : MS Social login 라우팅
- Blue : API 라우팅

Jenkins

Service Pipeline

```
pipeline {
    agent any
    environment {
        BRANCH_NAME = 'BE-develop'
        DOCKER_NETWORK = 'api-network'
        registryCredential = 'dockerhub-token'
        DOCKER_REGISTRY = 'bogan123/forteams'  // Docker 레지스트리 설
정
    }
    stages {
        stage('Checkout Code') {
            steps {
                git branch: 'BE-develop',
                    credentialsId: 'gitlab-qhdrnak',
                    url: 'https://lab.ssafy.com/s10-final/S10P31S20
4.git'
            }
        }
        stage('Initialize') {
            steps {
                script {
```

```
                    // 현재 디렉토리 출력
                    sh "pwd"
                    sh "ls -la" // 디렉토리 내용 확인
                    // BUILD_NUMBER를 스크립트 내에서 설정
                    // env.BUILD_NUMBER = "${currentBuild.number}"

                    // 마지막에 쓸 GIT_COMMIT 변수 선언
                    def currentCommit = sh(script: "git rev-parse HE
AD", returnStdout: true).trim()
                    env.GIT_COMMIT = currentCommit
                    echo "Current GIT_COMMIT: ${env.GIT_COMMIT}"
                }
            }
        }
        stage('Prepare') {
            steps {
                script {
                    // Git의 총 커밋 수 확인
                    def commitCount = sh(script: "git rev-list --cou
nt HEAD", returnStdout: true).trim().toInteger()
                    if (commitCount > 1 && fileExists('.last_success
ful_commit')) {
                        env.LAST_SUCCESSFUL_COMMIT = readFile('.last
_successful_commit').trim()
                        echo "Last Commit: ${LAST_SUCCESSFUL_COMMI
T}"
                        // if (env.LAST_SUCCESSFUL_COMMIT == 'null')
{
                        //     env.LAST_SUCCESSFUL_COMMIT = 'HEAD~1'
                        //     echo "Fallback to HEAD~1 because the
file was empty."
                        // }
                    } else if (commitCount > 1) {
                        env.LAST_SUCCESSFUL_COMMIT = 'HEAD~1'
                    } else {
                        // 커밋이 하나만 있는 경우, 아무 것도 비교하지 않음
                        env.LAST_SUCCESSFUL_COMMIT = 'HEAD'
                    }

                }
            }
        }
        stage('Check Changes') {
            steps {
```

```
                script {
                    if (env.LAST_SUCCESSFUL_COMMIT != 'HEAD') {
                        // if (env.LAST_SUCCESSFUL_COMMIT == '' || e
nv.LAST_SUCCESSFUL_COMMIT == null || env.LAST_SUCCESSFUL_COMMIT ==
'null') {
                        //     env.LAST_SUCCESSFUL_COMMIT = 'HEAD~1'
// 기본값 설정
                        //     echo "Invalid LAST_SUCCESSFUL_COMMIT,
reset to 'HEAD~1'"
                        // }
                        // env.LAST_SUCCESSFUL_COMMIT = 'HEAD~1'
                        def changedDirs = sh(script: "git diff --nam
e-only ${LAST_SUCCESSFUL_COMMIT} HEAD | xargs -I {} dirname {} | sor
t | uniq", returnStdout: true).trim()
                        echo "Changed directories: ${changedDirs}"
                        def servicesToBuild = []
                        def uniqueDirs = [:]

                        changedDirs.tokenize('\n').each {
                        // 경로를 /로 분할하여 프로젝트 루트 디렉토리 이름 추
출
                        def pathSegments = it.split('/')
                        // 예를 들어, BE/discovery/src/main/resources
-> BE 또는 discovery 가 될 수 있습니다.
                        if (pathSegments.size() > 1) {
                            def rootDir = pathSegments[1] // 루트 디렉
토리를 가정하는 부분, 상황에 맞게 조정 필요
                            // gradle 권한 설정
                            if (!uniqueDirs.containsKey(rootDir)) {
                                uniqueDirs[rootDir] = true
                                sh "chmod +x BE/${rootDir}/gradlew"
                                switch(rootDir) {
                                    case 'auth':
                                        servicesToBuild.add('auth')
                                        break
                                    case 'openchat':
                                        servicesToBuild.add('opencha
t')
                                        break
                                    case 'chatbot':
                                        servicesToBuild.add('chatbo
t')
                                        break
                                    case 'langchain':
```

```
                                        servicesToBuild.add('langcha
in')
                                        break
                                    case 'notification':
                                        servicesToBuild.add('notific
ation')
                                        break
                                    case 'discovery':
                                        servicesToBuild.add('discove
ry')
                                        break
                                    case 'gateway':
                                        servicesToBuild.add('gatewa
y')
                                        break
                                    case 'config':
                                        servicesToBuild.add('confi
g')
                                        break
                                }
                            }
                        }
                    }
                    echo "${servicesToBuild}"
                    env.SERVICES_TO_BUILD = servicesToBuild.join
(',')
                    } else {
                        echo "Only one commit present, skipping chan
ge detection."
                        env.SERVICES_TO_BUILD = ''
                    }
                }
            }
        }
        stage('Build JAR') {
            when {
                expression { env.SERVICES_TO_BUILD != '' }
            }
            steps {
                script {
                    env.SERVICES_TO_BUILD.tokenize(',').each { servi
ce ->
                        dir("BE/${service}") {
                            echo "Building JAR for ${service}..."
```

```
                            sh "./gradlew clean build -x test"
                            sh 'ls -l build/libs/'
                        }
                    }
                }
            }
        }
        // stage('Verify JAR') {
        //     steps {
        //         script {
        //             dir("BE/discovery") {
        //                 sh 'ls -l build/libs/'
        //             }
        //         }
        //     }
        // }
        stage('Build and Push Docker Images') {
            when {
                expression { env.SERVICES_TO_BUILD != '' }
            }
            steps {
                script {
                    env.SERVICES_TO_BUILD.tokenize(',').each { servi
ce ->
                        dir("BE/${service}") {
                            sh 'pwd'
                            sh 'ls -l'
                            docker.withRegistry("", registryCredenti
al) {
                                // Docker 이미지 빌드 및 푸시 (태그: 버전
번호)
                                sh "docker build -t ${DOCKER_REGISTR
Y}:${service}-${GIT_COMMIT} ."
                                sh "docker push ${DOCKER_REGISTR
Y}:${service}-${GIT_COMMIT}"

                                // Docker 이미지 빌드 및 푸시 (태그: lat
est)
                                // sh "docker build -t ${DOCKER_REGI
STRY}:${service}-latest ."
                                // sh "docker push ${DOCKER_REGISTR
Y}:${service}-latest"

                            }
```

```
                        }
                    }
                }
            }
        }
        stage('Deploy Services') {
            when {
                expression { env.SERVICES_TO_BUILD != '' }
            }
            steps {
                script {
                    sh "docker network create ${DOCKER_NETWORK} || t
rue"
                    env.SERVICES_TO_BUILD.tokenize(',').each { servi
ce ->
                        def imageName = "${DOCKER_REGISTRY}:${servic
e}-${GIT_COMMIT}"
                        sh "docker stop ${service} || true"
                        sh "docker rm ${service} || true" //
                        // sh "docker run -d --network ${DOCKER_NETW
ORK} -e 'JAVA_OPTS=-Djasypt.encryptor.password=####' --name ${servic
e} ${imageName}"
                        sh "docker run -d --network ${DOCKER_NETWOR
K} -e 'jasypt.encryptor.password=####' --name ${service} ${imageNam
e}"
                    }
                }
            }
        }
        stage('Health Check') {
            steps {
                script {
                    // 시도 횟수를 위한 변수
                    env.SERVICE_TO_BUILD.each { serviceName ->
                        def retryCount = 20
                        for (int i = 1; i <= retryCount; i++) {
                            // 건강 검사 엔드포인트 호출
                            def response = sh(script: "curl -s 'htt
p://forteams/api/\${serviceName}/health'", returnStdout: true).trim
()
                            if (response.contains('status: health
y')) {
                                sh "curl -d '{\"text\":\":nongdam_da
nce: \${serviceName} Release Complete! :nongdam_dance:\"}' -H 'Conte
```

```
nt-Type: application/json' -X POST https://meeting.ssafy.com/hooks/i
8y5ryrfnpbf8x6ysysyk8rn9o"
                                break
                            }
                            if (i == retryCount) {
                                sh "curl -d '{\"text\":\":keroro1:
\${serviceName} Release Fail OTL... :keroro1:\"}' -H 'Content-Type:
application/json' -X POST https://meeting.ssafy.com/hooks/i8y5ryrfnp
bf8x6ysysyk8rn9o"
                                error("\${serviceName} health check
failed after 20 attempts.")
                            }
                            echo "The server \${serviceName} is not
alive yet. Retry health check in 5 seconds..."
                            sleep(time: 5, unit: 'SECONDS')
                        }
                    }
                }
            }
        }
    }
    post {
        success {
            script {
                // 성공적인 빌드의 커밋 ID를 파일에 기록
                // sh "echo ${env.GIT_COMMIT} > .last_successful_com
mit"
                echo "Recording GIT_COMMIT: ${env.GIT_COMMIT}"
                sh "echo ${env.GIT_COMMIT} > .last_successful_commi
t"
                sh "cat .last_successful_commit" // 파일 내용 출력
            }
        }
    }
}
```

1. `Checkout Code` 단계

- 코드 체크아웃을 위한 단계

2. `Checkout Code` 단계

- 현재 커밋 ID를 가져와 `env.GIT_COMMIT` 에 저장

3. `Prepare` **단계**

- 커밋 수를 확인하고, 마지막 성공 커밋을 설정

4. `Check Changes` **단계**

- 마지막 성공 커밋과 현재 커밋 간의 변경된 디렉토리를 확인
- 변경된 디렉토리에 따라 빌드할 서비스를 결정하고 환경 변수에 저장

5. `Build JAR` **단계**

- `SERVICES_TO_BUILD` 변수가 설정된 경우에만 JAR 파일을 빌드

6. `Build and Push Docker Images` **단계**

- `SERVICES_TO_BUILD` 변수가 설정된 경우에만 Docker 이미지를 빌드하고 푸시
- 각 서비스 디렉토리에서 Docker 이미지를 빌드하고 레지스트리에 푸시

7. `Deploy Services` **단계**

- Docker 네트워크를 생성하고 각 서비스의 컨테이너를 실행

8. `Health Check` **단계**

- 각 서비스에 대해 건강 검사를 수행하여 성공적으로 배포되었는지 확인

9. Post - Success 단계

- 빌드가 성공하면 현재 커밋 ID를 `.last_successful_commit` 파일에 기록

front pipeline

```
pipeline {
    agent any

    environment {
        // 환경 변수 설정
        NODE_VERSION = '20.x'
        HOME = '.'
        DOCKER_REGISTRY = 'bogan123/forteams-front'
        registryCredential = 'dockerhub-token'
        CONTAINER_NAME = 'front-container'
        DOCKER_NETWORK = 'api-network'
```

```
    }

    stages {
        stage('Checkout') {
            steps {
                // Git 저장소에서 소스 코드 가져오기
                git branch: 'FE-develop',
                    url: 'https://lab.ssafy.com/s10-final/S10P31S20
4.git',
                    credentialsId: 'gitlab-qhdrnak'
            }
        }
        stage('Build Docker Image') {
            steps {
                dir('FE/app') {
                    script {
                        // Docker 이미지 빌드
                        docker.withRegistry("", registryCredential)
{
                            sh "docker build -t ${DOCKER_REGISTRY}:F
E-latest ."
                            sh "docker push ${DOCKER_REGISTRY}:FE-la
test"
                        }
                    }
                }
            }
        }

        stage('Deploy to Production') {
            steps {
                script {
                    // SSH를 통해 프로덕션 서버에서 Docker 컨테이너 실행
                    sh "docker stop ${CONTAINER_NAME} || true"
                    sh "docker rm ${CONTAINER_NAME} || true"

                    // 새 컨테이너 실행
                    sh "docker run -d -p 3000:3000 --restart always
--network ${DOCKER_NETWORK} --name ${CONTAINER_NAME} ${DOCKER_REGIST
RY}:FE-latest"
                }
            }
        }
    }
```

```
    post {
        always {
            // 워크스페이스 정리
            cleanWs()
        }
    }
}
```

Spring Cloud Gateway

```
jwt:
  secret: ENC(NqAJbCF2cJjys/fltEU6Xqn7RAQ+IyUkai+h4N69TICiqaAkELN0gJ
lG6EMH8Ejop2OIY8UK2iU=)
server:
  port: 8000

eureka:
  client:
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://discovery:8761/eureka

logging:
  level:
    org:
      springframework:
        cloud:
          gateway: DEBUG
        reactive:
          DispatcherHandler: TRACE
        client:
          RestTemplate: DEBUG
    reactor:
      netty:
        http:
          server: DEBUG
    springframework:
      web:
        reactive:
          function:
            client:
```

```
              WebClient: DEBUG
spring:
  cloud:
    gateway:
      routes:
        - id : ms-auth
          uri : lb://AUTH
          predicates:
            - Path=/api/v1/auth/**
        - id : ms-login
          uri : lb://AUTH
          predicates:
            - Path=/oauth2/authorization/microsoft/**

        - id : ms-login2
          uri : lb://AUTH
          predicates:
            - Path=/login/oauth2/code/**

        - id : user-auth
          uri : lb://AUTH
          predicates:
            - Path=/api/v1/user/**
          filters:
            - JwtTokenFilter

        - id: chatbot-http
          uri: lb://CHATBOT
          predicates:
            - Path=/api/v1/chatbot/**
          filters:
            - JwtTokenFilter

        - id: folder-http
          uri: lb://CHATBOT
          predicates:
            - Path=/api/v1/folder/**
          filters:
            - JwtTokenFilter

        - id: token-test-http
          uri: lb://CHATBOT
          predicates:
            - Path=/api/v1/token-test/**
```

```
          filters:
            - JwtTokenFilter

        - id: openchat-http
          uri: lb://OPENCHAT
          predicates:
            - Path=/api/v1/openchat/**
          filters:
#            - JwtTokenFilter

        - id: chatbot-websocket
          uri: lb://CHATBOT
          predicates:
            - Path=/api/ws/chatbot
          filters:
#            - JwtTokenFilter

        - id: openchat-websocket
          uri: lb://OPENCHAT
          predicates:
            - Path=/api/ws/openchat
          filters:
#            - JwtTokenFilter
#      default-filters:
#        - DedupeResponseHeader=Access-Control-Allow-Origin
  application:
    name: gateway
```

Spring Cloud Eureka

```
server:
  port: 8761

spring:
  application:
    name: discovery
#
eureka:
  client:
    register-with-eureka: false
    fetch-registry: false
```

▼ Sonarqube

도커 허브에서 이미지 Pull 받기

```
sudo docker pull sonarqube
```

도커에서 pull 받은 sonarqube 돌리기

```
sudo docker run -d --name sonarqube -p <설정 포트> sonarqube
```

잘 돌아가는지 확인

```
sudo docker ps -a

밑에처럼 나옴
CONTAINER ID   IMAGE                 COMMAND                  CREATE
b8ffbc8df06f   sonarqube             "/opt/sonarqube/dock…"   8 secc
c2be74d38c9a   jenkins/jenkins:lts   "/usr/bin/tini -- /u…"   17 hou
075f19c1fd28   nginx:latest          "/docker-entrypoint.…"   17 hou
```

만약에 위에처럼 했는데 안 된다면, sonar.properties 파일에 있는 default 포트 번호 바꿔줘야 함

host머신으로 설정 파일 docker cp

```
docker cp sonarqube:/opt/sonarqube/conf/sonar.properties .

(컨테이너(sonarqube) 안에 있는(:) 이 파일(/opt/sonarqube/conf/sonar.prope

sonarqube:: 이 부분은 컨테이너의 이름을 지정합니다. 여기서는 sonarqube라는 이름

/opt/sonarqube/conf/sonar.properties: (Copy할 파일의 위치) 이는 컨테이너

.: (Paste할 위치) 이 부분은 파일을 복사할 대상 위치를 나타내며, 여기서는 현재 작
```

host머신에 있는 sonar.properties 수정

```
# TCP port for incoming HTTP connections. Default value is 9000.
sonar.web.port=<원하는 포트 번호> # <- 이 부분 원래 주석임. 풀기!
```

container로 다시 cp

```
docker cp ./sonar.properties sonarqube:/opt/sonarqube/conf/sonar.prc
```

docker에서 sonarqube 중지했다가 다시 시작

```
sudo docker stop sonarqube

sudo docker start sonarqube

# 설정이 바뀌었으므로 꼭 하기https://www.youtube.com/watch?v=nyCDyjx6TuM&
```

▼ OpenChat & Chatbot Infra Setting

Open Chat & Chatbot 인프라 세팅을 위한 docker-compose 작성

- 경로 : /home/ubuntu/compose/
- open-chat-docker-compose.yml

```
version: '3.8'

services:
  redis:
    image: redis:latest
    container_name: redis
    ports:
      - "6379:6379"

  mongodb:
    image: mongo:latest
    container_name: mongodb
    volumes:
      - mongodb_data:/data/db
    ports:
      - "47017:27017"
    environment:
      MONGO_INITDB_ROOT_USERNAME: ####
      MONGO_INITDB_ROOT_PASSWORD: ####
```

```
  adminmongo:
    image: mrvautin/adminmongo
    container_name: adminmongo
    ports:
      - "8674:8674"
    environment:
      ADMIN_MONGO_PORT: 8674
      CONN_NAME: "Localhost"
      DB_HOST: "mongodb"
      DB_PORT: 27017
      DB_USERNAME: ####
      DB_PASSWORD: ####

  rabbitmq:
    image: rabbitmq:3-management
    container_name: rabbitmq
    ports:
      - "5672:5672" # AMQP port
      - "8672:15672" # UI port
      - "61613:61613" #STOMP port
    environment:
      RABBITMQ_DEFAULT_USER: ####
      RABBITMQ_DEFAULT_PASS: ####
      RABBITMQ_STOMP_ENABLED: 'true'
    volumes:
      - ./rabbitmq.conf:/etc/rabbitmq/rabbitmq.conf

volumes:
  mongodb_data:
```

- rabbitmq.conf

```
# Enable the management plugin
management.tcp.port = 15672

# AMQP default listener
listeners.tcp.default = 5672

# STOMP plugin configuration
stomp.listeners.tcp.1 = 61613

# Default user settings
```

```
# default_user = user
# default_pass = password
```

mongoDB 접속

```
docker exec -it mongodb mongosh
```

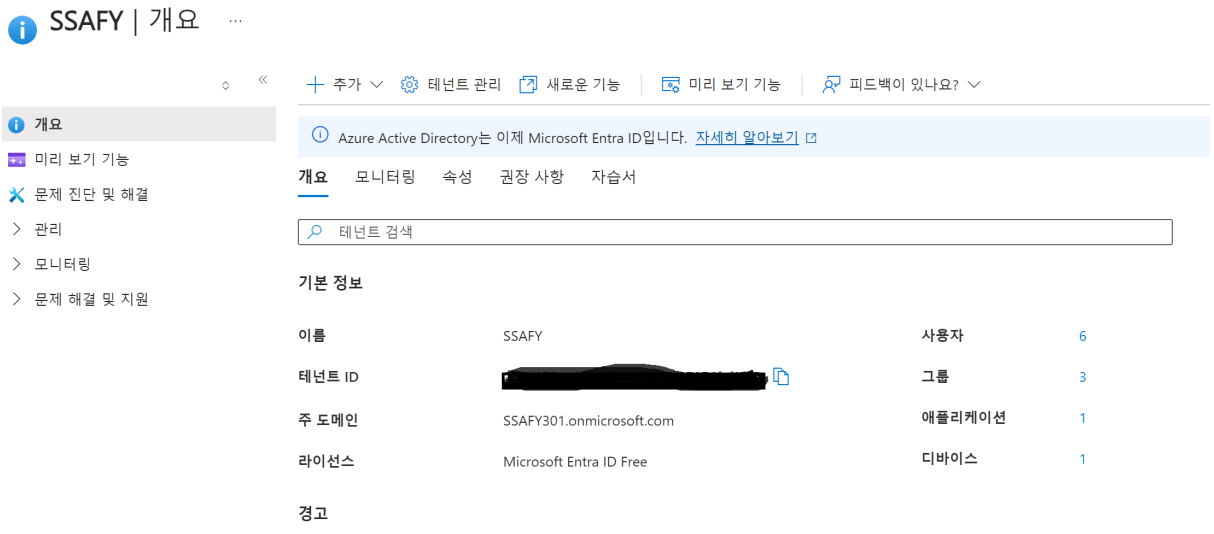
mongoDB compass 연결하기

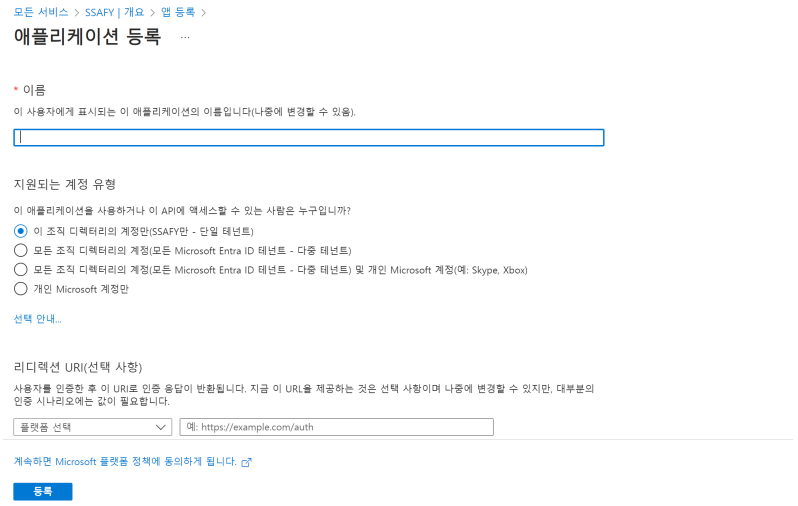
```
mongodb://####:####@forteams.co.kr:47017
```

External Services

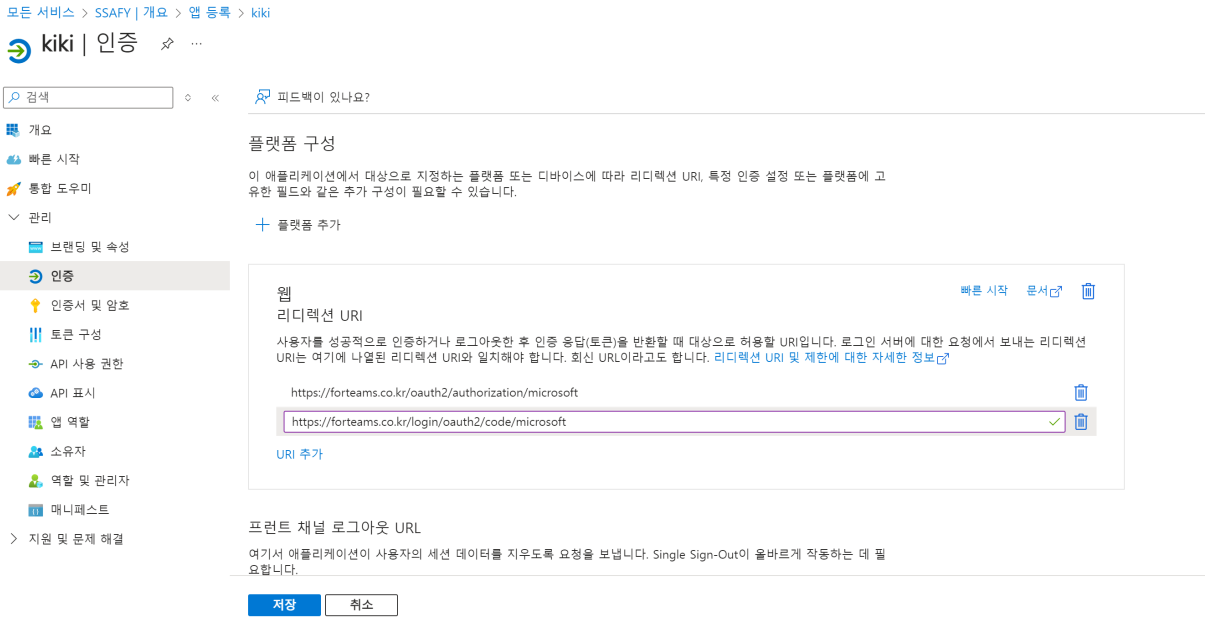
Microsoft Entra ID

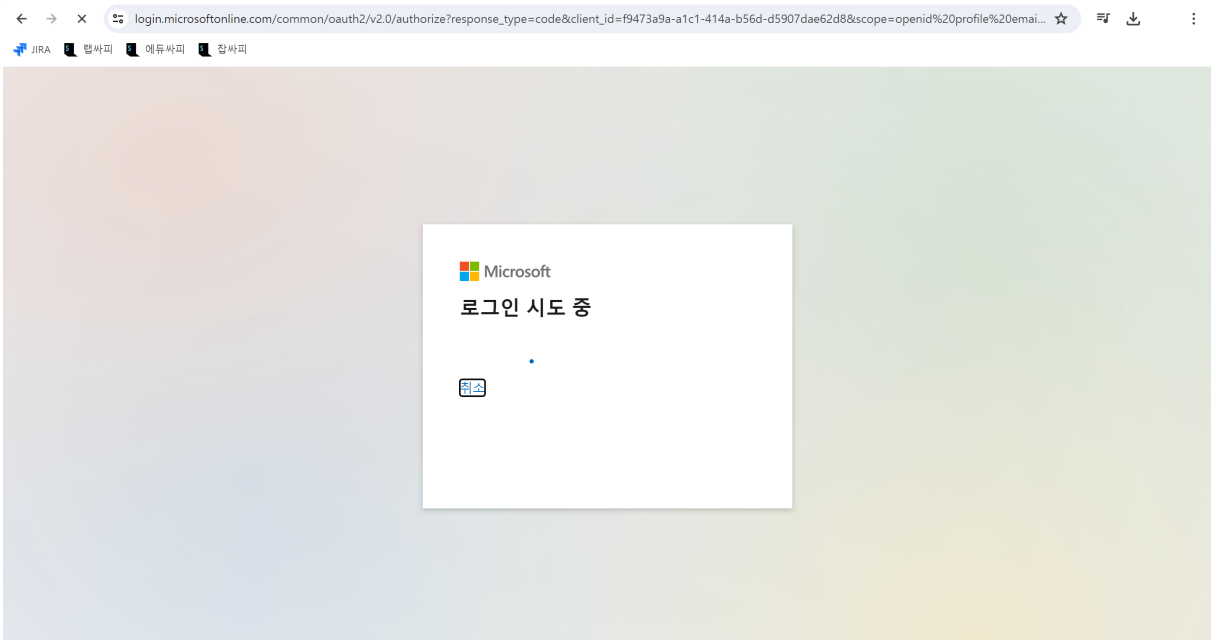
https://portal.azure.com/#view/Microsoft_AAD_IAM/ActiveDirectoryMenuBlade/~/Overview





이후 등록된 애플리케이션 페이지에서 domain명에 따라 url 변경 필요





DB dumpfile

backup.sql

```
docker exec some-postgres pg_dump -U postgres forteams > /backup.sql
```

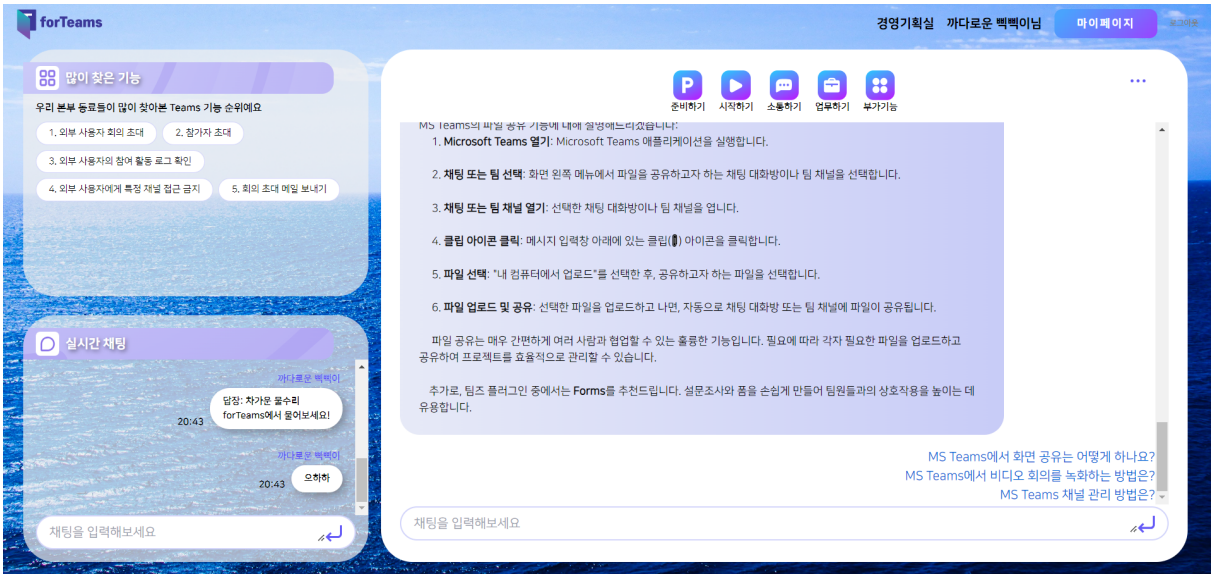
MongoDB dump

chatbot.savedChats.json

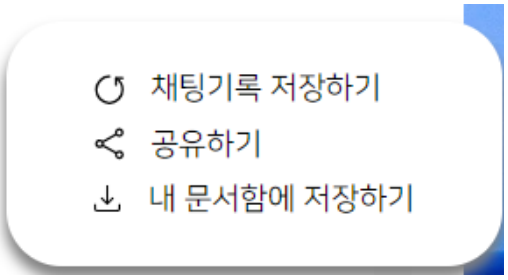
openChat.openChat.json

chatbot.chatLogs.json

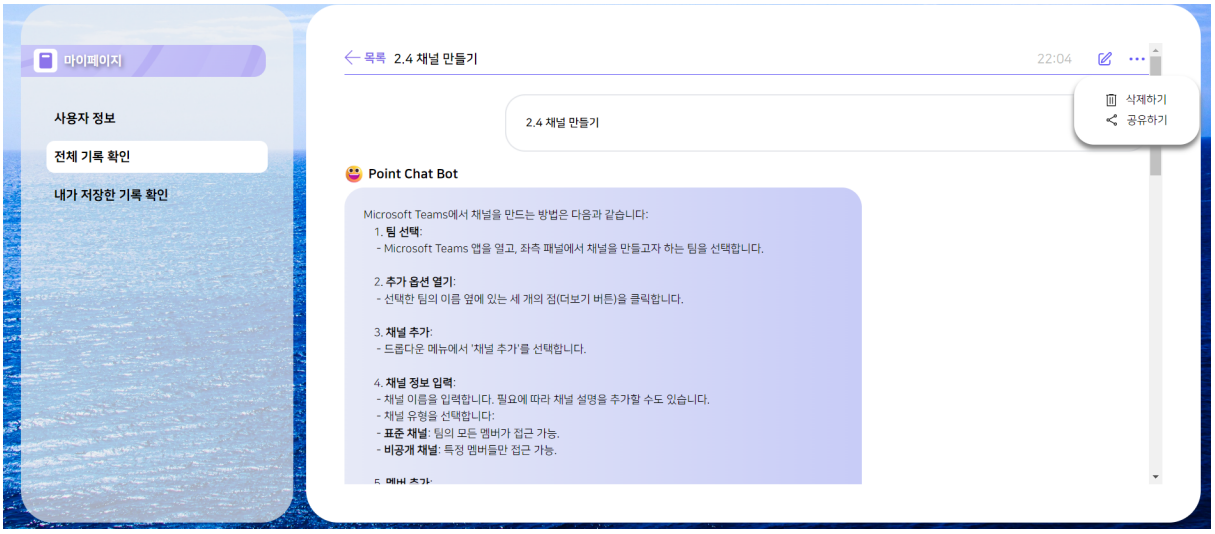
시연 시나리오



- 외부 사용자와 화상회의 어떻게 해? 질문 입력.
  - 관련된 질문이 나옴
  - 추천된 질문 클릭
  - 맥락을 고려하여 연관된 답변이 잘 나오는 것을 확인할 수 있음
  - 또한, 답변 생성 중단도 할 수 있음
- 질문을 뭘 할지 모르겠을 때, 상단의 시작하기 → 2.4 채널만들기 버튼 클릭으로 관련된 답변 내용과 사용법 gif 이미지도 출력
- 많이 찾은 기능에서 내가 속한 부서와 관련된 사람들이 찾아본 키워드 등을 참고하여 질문을 할 수 있음



- 채팅 기록을 저장하고 새로운 채팅 세션을 만들거나 폴더를 만들어 채팅 유형별로 저장할 수 있습니다. 또한 공유 링크를 생성할 수 있어, 저희 서비스에 가입되어있지 않은 외부 사람에게도 Teams 사용법을 공유받을 수 있도록 하였음
- 또한, 익명성 있는 실시간 채팅 화면을 추가하여 자유롭게 질문, 잡담을 나눌 수 있는 대나무숲도 있음
- 저장된 내역에서 버튼 클릭으로 이어쓰기도 가능


마이페이지
사용자 정보
전체 기록 확인
내가 저장한 기록 확인
목록 2.4 채널 만들기
22:04
삭제하기
공유하기
2.4 채널 만들기
Point Chat Bot
Microsoft Teams에서 채널을 만드는 방법은 다음과 같습니다:
1. 팀 선택:
- Microsoft Teams 앱을 열고, 좌측 패널에서 채널을 만들고자 하는 팀을 선택합니다.
2. 추가 옵션 열기:
- 선택한 팀의 이름 옆에 있는 세 개의 점(더보기 버튼)을 클릭합니다.
3. 채널 추가:
- 드롭다운 메뉴에서 '채널 추가'를 선택합니다.
4. 채널 정보 입력:
- 채널 이름을 입력합니다. 필요에 따라 채널 설명을 추가할 수도 있습니다.
- 채널 유형을 선택합니다:
- 표준 채널: 팀의 모든 멤버가 접근 가능.
- 비공개 채널: 특정 멤버들만 접근 가능.