

프론트

1. s3 - 버킷

Amazon S3 > 버킷 > moim-front

moim-front 정보 퍼블릭 액세스 가능

객체 | 속성 | 권한 | 지표 | 관리 | 액세스 지정

객체 (6) 정보 열기 삭제 작업 폴더 만들기 업로드

객체는 Amazon S3에 저장되어 있는 기본 엔티티입니다. [Amazon S3 인벤토리](#)를 사용하여 버킷에 있는 모든 객체의 목록을 얻을 수 있습니다. 다른 사용자가 객체에 액세스할 수 있게 하려면 명시적으로 권한을 부여해야 합니다. [자세히 알아보기](#)

☐

이름

▲

유형

▼

마지막 수정

▼

크기

▼

스토리지 클래스

▼

<input type="checkbox"/>	css/	폴더	-	-	-
<input type="checkbox"/>	favicon.ico	ico	2024. 5. 3. pm 4:15:27 PM KST	0B	Standard
<input type="checkbox"/>	fonts/	폴더	-	-	-
<input type="checkbox"/>	img/	폴더	-	-	-
<input type="checkbox"/>	index.html	html	2024. 5. 3. pm 4:15:28 PM KST	827.0B	Standard
<input type="checkbox"/>	js/	폴더	-	-	-

프론트 이미지 업로드

2. cloudfront

CloudFront > 배포 > E1MGKVCVKVRN33

E1MGKVCVKVRN33

일반 | 보안 | **원본** | 동작 | 오류 페이지 | 무효화 | 태그

원본

원본 이름

▼

원본 도메인

▼

원본 경로

▼

원본 유형

▼

<input type="radio"/>	moim-front.s3.ap-...	moim-front.s3-we...	S3 static website
-----------------------	----------------------	---------------------	-------------------

https 적용

3. route 53

레코드 세부 정보



레코드 편집

레코드 이름

www.jangeunji.shop

레코드 유형

A

값



도메인 적용 (www.jangeunji.shop)

4. git actions

a. 자동화 스크립트 작성

```
name: Deploy to AWS S3

on:
  push:
    branches:
      - goth/depl
jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2

      - name: setup node.js
        uses: actions/setup-node@v2
        with:
          node-version: '20'

      - name: npm install
        run: npm install

      - name: npm build
        run: npm run build
```

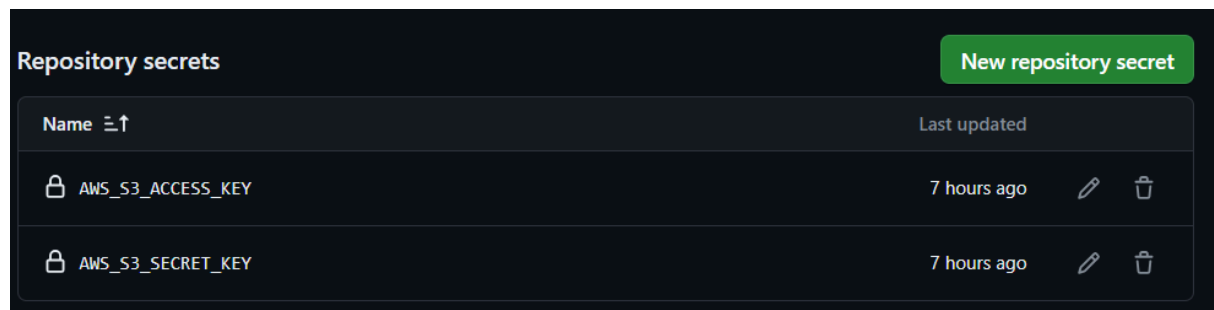
```

- name: setup aws cli v3
  uses: aws-actions/configure-aws-credentials@v3
  with:
    aws-access-key-id: ${secrets.AWS_S3_ACCESS_KEY}
    aws-secret-access-key: ${secrets.AWS_S3_SECRET_KEY}
    aws-region: "ap-northeast-2"
- name: deploy to s3
  run: |
    aws s3 cp ./dist s3://moim-front/ --recursive

```

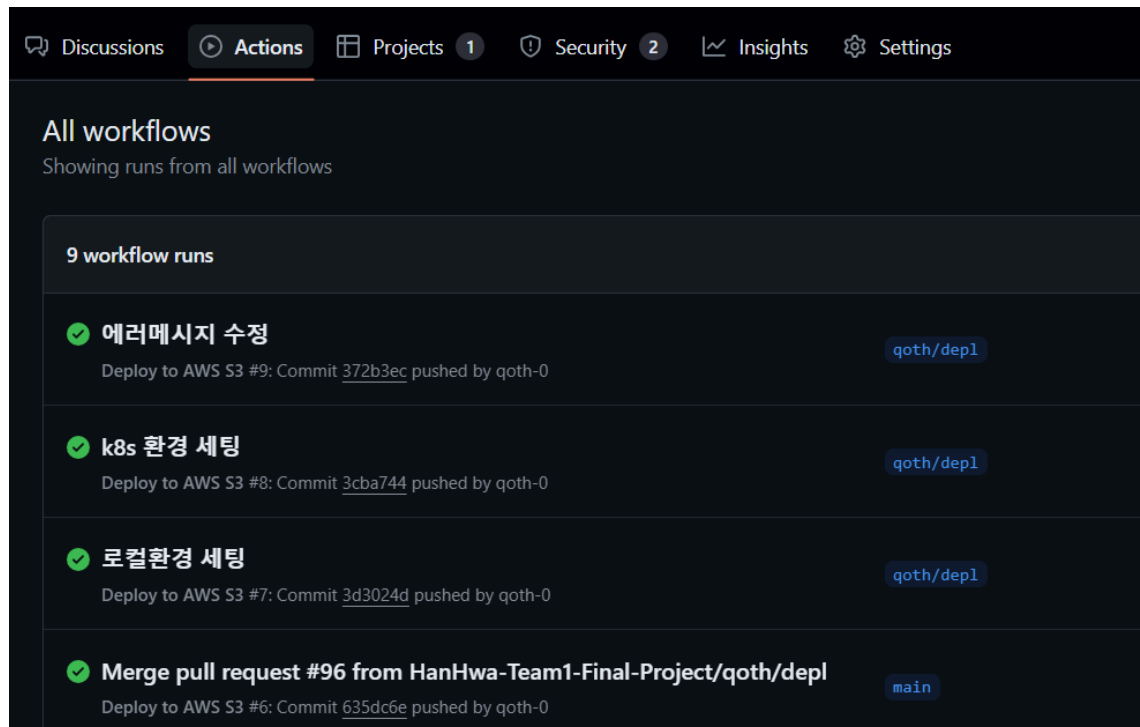
빌드 및 s3 업로드 자동화

b. git secret 설정



주요정보 secret 설정

c. s3 배포 자동화 테스트



소스코드 변경 및 push - s3 이미지 자동 업로드

5. 배포 확인

← → ↺

🔑 ☆ 📁 | 👤

🌐 https://www.jangeunji.shop/main

MOIM

🔍 일정 검색

🔔 0

🚪 로그아웃

👤 user

user@user.com

📅 캘린더

📊 아이젠하워 매트릭스

📅 일정 생성

👤 모임 리스트

👤 채팅 테스트

📁 채팅방 생성

💬 채팅 리스트

📅 오늘의 일정

< >

today

May 2024

month

week

day

Sun	Mon	Tue	Wed	Thu	Fri	Sat
28	29	30	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1

<https://www.jangeunji.shop>

백엔드

1. eks 노드 그룹

EKS > 클러스터 > moim > 노드 그룹 > moim-nodes

moim-nodes

🔄 편집 삭제

노드 그룹 구성 정보

Kubernetes 버전
1.29

AMI 릴리스 버전 정보
1.29.3-20240424

AMI 유형 정보
AL2_x86_64

인스턴스 유형
t3.medium

상태
✔️ 활성화

디스크 크기
20 GiB

세부 정보

노드

상태 문제 0

Kubernetes 레이블

업데이트 구성

Kubernetes 테인트

업데이트 기록

태그

노드 (1) 정보

🔍 속성 또는 값으로 노드 필터링

< 1 >

노드 이름

인스턴스 유형

노드 그룹

생성됨

상태

t3.medium

moim-nodes

생성됨
May 2, 2024, 17:43 (UTC+09:00)

✔️ 준비 완료

워커노드 1개 생성

2. RDS - mariaDB

RDS > 데이터베이스 > moim

moim

요약

DB 식별자
moim

CPU
2.63%

상태
✔️ 사용 가능

클래스
db.t3.small

역할
인스턴스

현재 활동
10 연결

엔진
MariaDB

리전 및 AZ
ap-northeast-2b

주요 데이터 저장

3. ElastiCache - Redis

ElastiCache > Redis 캐시 > moim-redis

moim-redis 정보

🔄 마이그레이션 시작 수정 백업

▼ 클러스터 세부 정보

클러스터 이름
moim-redis

엔진
Redis

업데이트 상태
최신 상태

설명
-

엔진 버전
7.1.0

클러스터 모드
비활성화됨

노드 유형
cache.m4.large

글로벌 데이터 스토어
-

샤드
1

상태
✔️ Available

글로벌 데이터 스토어 역할
-

노드 수
1

이메일 인증코드, 알림 내역, 모임 추천 일정 저장

4. kubectl secret 설정

```
PS C:\Users\Playdata> kubectl get secrets
```

NAME	TYPE	DATA	AGE
api-secret	Opaque	1	8h
docker-secret	kubernetes.io/dockerconfigjson	1	21h
email-secret	Opaque	2	8h
file-secret	Opaque	2	8h
google-secret	Opaque	2	8h
jwt-secret	Opaque	1	8h
kakao-secret	Opaque	2	8h
moim-com-tls	kubernetes.io/tls	2	22h
moim-db-secret	Opaque	3	24h
redis	Opaque	1	21h

주요정보 secret 설정

5. deployment, service 스크립트 작성

```
# depl
apiVersion: apps/v1
kind: Deployment
metadata:
  name: moim-depl
spec:
  # pod 개수
  replicas: 1
  selector:
    matchLabels:
      app: moim
  template:
    metadata:
      labels:
        app: moim
    spec:
      containers:
        - name: k8s-moim
          image: bny1324/moim:v1
          imagePullPolicy: Always
          ports:
            - containerPort: 8080
          resources:
            limits:
              cpu: "1"
              memory: "500Mi"
```

```
  requests:
    cpu: "0.5"
    memory: "250Mi"
env:
- name: REDIS_HOST
  valueFrom:
    secretKeyRef:
      name: redis
      key: REDIS_HOST
- name: DB_HOST
  valueFrom:
    secretKeyRef:
      name: moim-db-secret
      key: DB_HOST
- name: DB_USERNAME
  valueFrom:
    secretKeyRef:
      name: moim-db-secret
      key: DB_USERNAME
- name: DB_PASSWORD
  valueFrom:
    secretKeyRef:
      name: moim-db-secret
      key: DB_PASSWORD
- name: USERNAME
  valueFrom:
    secretKeyRef:
      name: email-secret
      key: USERNAME
- name: PASSWORD
  valueFrom:
    secretKeyRef:
      name: email-secret
      key: PASSWORD
- name: KAKAO_ID
  valueFrom:
    secretKeyRef:
      name: kakao-secret
      key: KAKAO_ID
- name: KAKAO_SECRET
  valueFrom:
    secretKeyRef:
      name: kakao-secret
```

```
        key: KAKAO_SECRET
- name: GOOGLE_ID
  valueFrom:
    secretKeyRef:
      name: google-secret
      key: GOOGLE_ID
- name: GOOGLE_SECRET
  valueFrom:
    secretKeyRef:
      name: google-secret
      key: GOOGLE_SECRET
- name: JWT
  valueFrom:
    secretKeyRef:
      name: jwt-secret
      key: JWT
- name: ACCESS-KEY
  valueFrom:
    secretKeyRef:
      name: file-secret
      key: ACCESS-KEY
- name: SECRET-KEY
  valueFrom:
    secretKeyRef:
      name: file-secret
      key: SECRET-KEY
- name: SECRET
  valueFrom:
    secretKeyRef:
      name: api-secret
      key: SECRET
# 컨테이너 상태 확인
readinessProbe:
  httpGet:
    # healthcheck 경로
    path: /api/auth
    port: 8080
  # 컨테이너 시작 후 지연
  initialDelaySeconds: 10
  # 확인 반복 주기
  periodSeconds: 10
  # 요청이 완료되어야 하는 시간
  timeoutSeconds: 1
```



```

        # 연속 성공 횟수
        successThreshold: 1
        # 연속 실패 횟수
        failureThreshold: 3
    imagePullSecrets:
        - name: docker-secret
---
# service yml 추가
apiVersion: v1
kind: Service
metadata:
    # ingress와 연결될 서비스 명
    name: moim-service
spec:
    # ClusterIP는 클러스터 내부에서만 접근가능한 Service를 생성
    type: ClusterIP
    ports:
        - name: http
          # pod의 외부포트
          port: 8080
          # 서비스와 연결해줄 pod의 내부 포트
          targetPort: 8080
          # depl의 labels과 맞추기
    selector:
        app: moim

```

pod 1개, secret 사용, dockerhub 이미지 pull, 버전이슈 해결 및 무중단 배포 설정

6. ingress-cert 스크립트 작성

```

# https 인증서 적용 절차

# 1. cert-manager 생성
# cert-manager 생성을 위한 cert-manager namespace 생성
# 1-1) kubectl create namespace cert-manager
# 1-2) Helm 설치 -
# 1-3) cert-manager를 설치하기 위한 Jetstack Helm repository 추가
# 명령어 : helm repo add jetstack https://charts.jetstack.io
# 1-4) Helm repository 업데이트
# 명령어 : helm repo update
# 1-5) cert-manager 차트 설치
# 명령어 : helm install cert-manager jetstack/cert-manager --namespace
cert-manager --version v1.5.0 --create-namespace --set installCRDs=true

```

```

# 2. ClusterIssuer 생성
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: letsencrypt-prod
spec:
  acme:
    # 인증서 서버 주소. 해당 서버의 리소스를 통해 인증서 발행
    server: https://acme-v02.api.letsencrypt.org/directory
    # 인증서 만료 또는 갱신 필요시 알람 email
    email: bny1324@naver.com
    privateKeySecretRef:
      name: letsencrypt-prod
    solvers:
      - http01:
          ingress:
            class: nginx
---
# 3. ClusterIssue를 사용하여 Certificate 리소스 생성 -> Certificate 리소스
# 생성 시에 인증서 발급됨
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: moim-com-tls
  namespace: default
spec:
  secretName: moim-com-tls
  duration: 2160h # 90일
  renewBefore: 360h # 15일 전에
  issuerRef:
    name: letsencrypt-prod
    kind: ClusterIssuer
  commonName: server.jangeunji.shop
  dnsNames:
    - server.jangeunji.shop

```

https 인증서발행

7. ingress 스크립트 작성

```

apiVersion: networking.k8s.io/v1
kind: Ingress

```

```

metadata:
  name: moim-ingress
  annotations:
    kubernetes.io/ingress.class: nginx # ingress-controller가 nginx
    nginx.ingress.kubernetes.io/rewrite-target: /
    cert-manager.io/cluster-issuer: letsencrypt-prod
spec:
  tls: # https 인증서
    # 해당 host로 인증서를 발급
  - hosts:
    - "server.jangeunji.shop"
    secretName: moim-com-tls
  rules:
    - host: server.jangeunji.shop
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                # ingress와 연결될 서비스명 - depl-serv에서 지정
                name: moim-service
                port:
                  number: 8080

```

로드밸런싱 자동화, https 적용

8. route 53

레코드 세부 정보



레코드 편집

레코드 이름

server.jangeunji.shop

레코드 유형

CNAME

값

|

도메인 적용(server.jangeunji.shop)

9. dockerfile 작성

```
FROM openjdk:17-jdk-alpine as stage1

WORKDIR /app

# .dockerignore에 불필요 파일 추가
COPY . .

# /app/build/libs/*.jar 파일을 아래 명령어를 통해 생성
RUN chmod +x ./gradlew
RUN ./gradlew bootJar

# 새로운 work stage 시작, 기존 스테이지는 자동으로 사라진다.
FROM openjdk:17-jdk-alpine

WORKDIR /app

# 왼쪽이 stage1, 오른쪽이 도커 두번째 stage
COPY --from=stage1 /app/build/libs/*.jar app.jar

# CMD 또는 ENTRYPOINT를 통해 컨테이너 실행
ENTRYPOINT [ "java", "-jar", "app.jar" ]
```

이미지 빌드 및 dockerhub 업로드 자동화

10. application.yml secret 적용

```
# 예외 처리를 커스텀하게 진행하기 위해 Whitelabel Error Page 옵션 OFF
server:
  servlet:
    encoding:
      charset: UTF-8
      force: true
  error:
    whitelabel:
      enabled: false
spring:
  servlet:
    multipart:
      max-file-size: 10MB
      max-request-size: 10MB
  datasource:
    driver-class-name: org.mariadb.jdbc.Driver
    # url: jdbc:mariadb://localhost:3306/moim // root, 1234
    url: jdbc:mariadb://${DB_HOST}:3306/moim
    username: ${DB_USERNAME}
    password: ${DB_PASSWORD}
  jpa:
    database: mysql
```

```

database-platform: org.hibernate.dialect.MariaDBDialect
generate-ddl: true
hibernate:
  ddl-auto: update
  show_sql: true
data:
  redis:
    # Spring data JPA repository 와 Spring data redis Repository가 모두
인식되어
    # redis가 이게 내 리포지토리가 맞아? 그럼 명시해줘 라는 로그 해결
    repositories:
      enabled: false
#      host: localhost
    host: ${REDIS_HOST}
    port: 6379
mail:
  host: smtp.gmail.com
  port: 587
  username: ${USERNAME}
  password: ${PASSWORD}
  properties:
    mail:
      debug: true
      smtp.auth: true
      smtp.timeout: 50000 # SMTP 서버에 연결을 시도하고 응답을 기다리는 최대 시간이
50,000ms
      smtp.starttls.enable: true
security:
  oauth2:
    client:
      # OAuth 로그인 시 설정한 Application의 정보를 사용하여
      # AccessToken을 Authorization Server에게 발급받을 때 사용
    registration:
      google:
        client-id: ${GOOGLE_ID}
        client-secret: ${GOOGLE_SECRET}
        redirect-uri:
https://server.jangeunji.shop/login/oauth2/code/google
        scope: profile, email

      kakao:
        client-id: ${KAKAO_ID}
        client-secret: ${KAKAO_SECRET}
        redirect-uri:
https://server.jangeunji.shop/login/oauth2/code/kakao
        client-authentication-method: client_secret_post
        authorization-grant-type: authorization_code
        scope: account_email, profile_nickname, profile_image
        client-name: Kakao

    # Spring에서 카카오의 provider 정보는 제공하지 않으므로, 직접 설정 필요
    # AccessToken을 Authorization Server에게 발급 받은 후,
    # 해당 AccessToken으로 Resource Server의 API를 사용할 때

```

```

    # provider 부분의 정보를 사용하여 API 호출
    provider:
      kakao:
        authorization-uri: https://kauth.kakao.com/oauth/authorize
        token-uri: https://kauth.kakao.com/oauth/token
        user-info-uri: https://kapi.kakao.com/v2/user/me
        user-name-attribute: id

jwt:
  secretKey: ${JWT}

  access:
    # access token 만료시간 / 30분 (1000L(ms -> s) * 60L(s -> m) * 30L(m ->
h))
    # expiration: 1800000
    expiration: 60000000000
    header: Authorization

  refresh:
    # refresh token 만료시간 / 2주 (1000L(ms -> s) * 60L(s -> m) * 60L(m -> h)
* 24L(h -> 하루) * 14)
    expiration: 1209600000
    header: Authorization_Refresh













cloud:
  aws:
    credentials:
      access-key: ${ACCESS-KEY}
      secret-key: ${SECRET-KEY}
    s3:
      bucket: moim-bucket
      region:
        static: ap-northeast-2
      stack:
        auto: false
  custom:
    api:
      secretKey: ${SECRET}

```

주요 정보 **secret** 적용

11. git actions

a. secret 설정

Repository secrets			New repository secret	
Name 			Last updated	
 AWS_ACCESS_KEY			yesterday	 
 AWS_SECRET_KEY			yesterday	 
 DOCKER_PASSWORD			yesterday	 
 DOCKER_USERNAME			yesterday	 

주요정보 secret 설정

b. 자동화 스크립트 작성

```
name: deploy k8s-moim

on:
  push:
    branches:
      - goth/depl

jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
    steps:
      - name: checkout github
        uses: actions/checkout@v2

      - name: install kubectl
        uses: azure/setup-kubectl@v3
        with:
          version: "v1.25.9"
        id: install

      # aws 권한 세팅
      - name: configure aws
        uses: aws-actions/configure-aws-credentials@v1
        with:
          aws-access-key-id: ${secrets.AWS_ACCESS_KEY}
          aws-secret-access-key: ${secrets.AWS_SECRET_KEY}
          aws-region: ap-northeast-2
```

```

- name: update cluster infomation
  run: aws eks update-kubeconfig --name moim --region
ap-northeast-2

- name: Build Docker Image
  run: docker build -t bny1324/moim:v1 .
- name: DockerHub Login
  uses: docker/login-action@v1
  with:
    username: ${ secrets.DOCKER_USERNAME }
    password: ${ secrets.DOCKER_PASSWORD }
- name: Push Docker Image to DockerHub
  run: docker push bny1324/moim:v1

- name: eks kubectl apply
  run: |
    kubectl apply -f ./k8s/depl-serv.yml
    kubectl rollout restart deployment moim-depl

```

eks 워커노드 접근 및 dockerhub 이미지 업로드 후 deployment, service 재시작 자동화

c. 소스코드 변경 및 push - 백엔드 배포 자동화

All workflows

Showing runs from all workflows

16 workflow runs

스케줄러 현재시간 kst로 세팅

12. deploy k8s-moim #16: Commit [5eca7cd](#) pushed by goth-0

13.

goth/depl

localdatetime - kst 변환

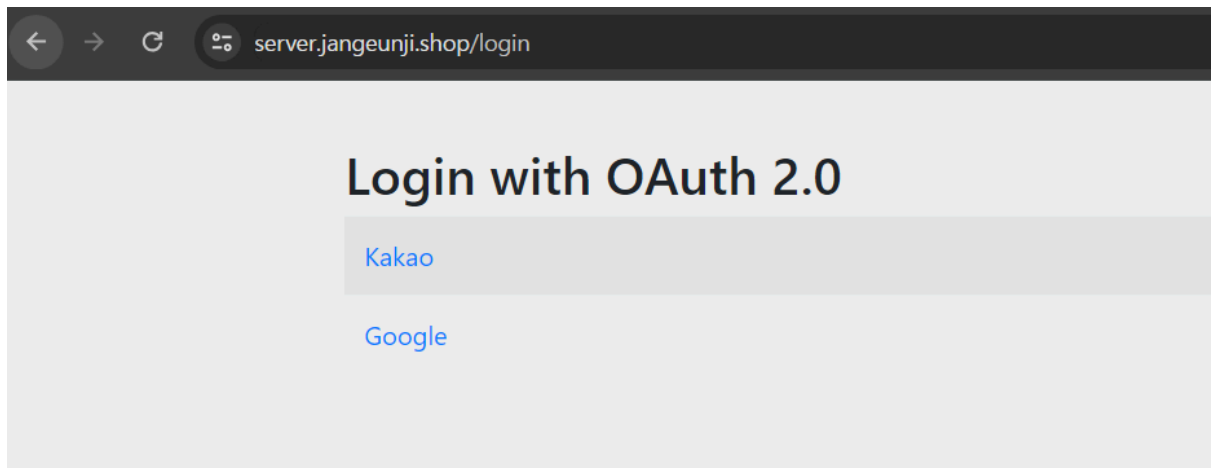
14. deploy k8s-moim #15: Commit [31ff3d9](#) pushed by goth-0

15.

goth/depl

16.

12. 배포 확인



<https://server.jangeunji.shop>