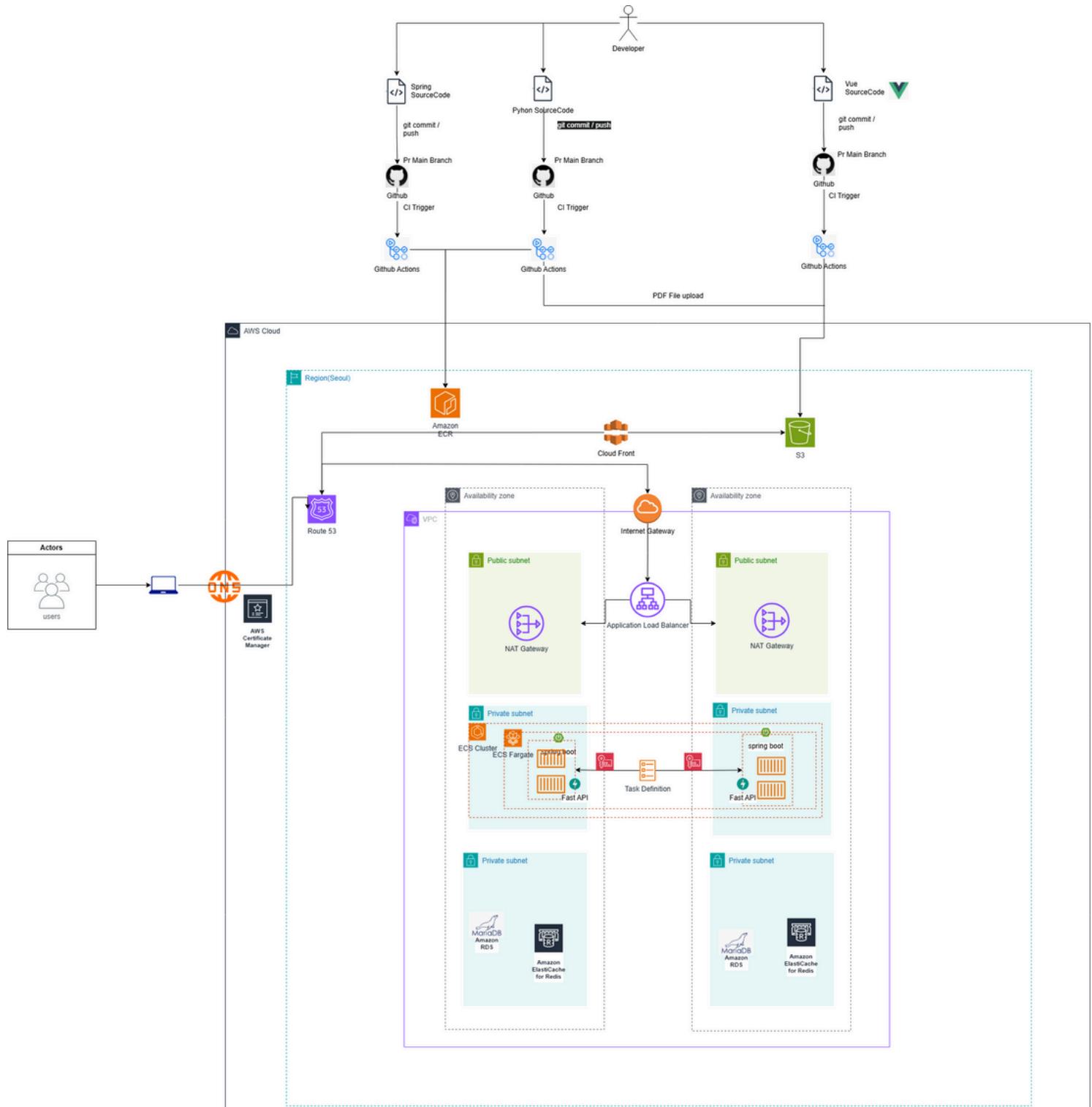


객담 CI/CD 계획서

team4
스무고객

시스템 아키텍처



1. Infra 구성

- **Front End**

- Amazon Route 53 (도메인 호스팅)
- Amazon S3 (정적 웹 호스팅 버킷)
- Amazon CloudFront (CDN, 캐시)

- **Back End**

- Amazon ECR (도커 이미지 저장소)
- Amazon ECS + AWS Fargate (컨테이너 런타임)
- Elastic Load Balancing (ALB)

- **Machine Learning (AI)**

- Amazon ECR
- Amazon ECS + AWS Fargate
- Amazon S3 (PDF 원본 저장, 파싱 결과 저장 가능)

- **Database**

- Amazon RDS (MariaDB)
- Amazon ElastiCache (Redis)

- **Common / Security**

- AWS Identity and Access Management
- AWS Certificate Manager (HTTPS 인증서)
- AWS Key Management Service (시크릿key)
- AWS Secrets Manager (DB/Redis/JWT 등 비밀정보)
- 모니터링/로그: CloudWatch

2. CI/CD 공통 원칙

2.1 브랜치 전략 (권장)

- develop : 개발/통합
- main : 운영 배포 트리거
- 배포 트리거는 PR merge로만(직접 push 금지 권장)

2.2 버전/태그

- Docker 이미지 태그: git sha
- 추가로 latest 태그는 편의용으로만 사용(배포는 sha 기반)

2.3 보안

최소 권한:

- Front: S3 sync + CloudFront
- Backend : ECR push + ECS service update
- AI: ECR push + S3 + ECS service update

민감정보는:

- Secrets Manager
- 또는 ECS Task Definition의 Secret 매핑으로 주입

2.4 롤백

- Front: S3에 빌드 보관 또는 CloudFront + S3 버전관리로 롤백
- Backend/AI: ECS에서 이전 태스크 정의로 재배포

3 . 배포 전략

Front End CI/CD

배포 트리거

: main 브랜치로 PR merge 시 GitHub Actions 실행

빌드

: Vite 기반 정적 파일 빌드

환경변수

: GitHub Secrets를 통해 빌드 시점에 주입

배포

: S3 정적 호스팅 버킷에 dist 파일 업로드

캐시 전략:

- index.html: 캐시 비활성화
- JS/CSS: 해시 기반 캐시
- CDN: CloudFront 캐시 무효화

무중단 배포

: S3 + CloudFront 구조로 무중단 제공

롤백

: 이전 빌드 재업로드 또는 S3 버전 관리

Back End CI/CD

배포 트리거

: develop 브랜치 PR merge

빌드

: Spring Boot JAR 생성 후 Docker 이미지 빌드

이미지 저장소

: Amazon ECR

배포 방식

: ECS Fargate Service Rolling Update

Task Definition

: 기존 서비스가 사용하는 Task Definition 기반으로 image만 교체

무중단 배포

: ALB Health Check 통과 후 기존 Task 종료

버전 관리

: Docker image SHA digest 기반

롤백

: 이전 Task Definition revision으로 즉시 복구 가능

AI (FastAPI) CI/CD

배포 트리거

: develop 브랜치 Pull Request merge 시 GitHub Actions 실행

빌드

: FastAPI 애플리케이션을 Docker 이미지로 빌드

이미지 저장소

: Amazon ECR (Elastic Container Registry)

배포 방식

: ECS Fargate Service 기반 Rolling Update 배포

Task Definition 관리

: 기존 AI 서비스가 사용하는 ECS Task Definition을 기준으로

컨테이너 이미지(Image)만 신규 FastAPI Docker 이미지로 교체하여 새로운 Revision 생성

무중단 배포

: ALB Health Check(FastAPI api/v1/health 엔드포인트)를 통과한 신규 Task가 정상 상태가 된 이후 기존 Task를 종료하여 서비스 중단 없이 배포 수행

파일 업로드

: 응대 메뉴얼 pdf 파일 S3버킷에 저장 및 리드 gaekdam-ai-bot-bucket

버전 관리

: Docker Image SHA Digest 기반 버전 관리

롤백 전략

: 장애 발생 시 이전 Task Definition Revision으로 즉시 롤백 가능

4 . Database 운영 전략 (RDS + Redis)

분리 전략

- 현재 단계에서는 RDS Multi-AZ 기반 단일 Writer 구조로 운영 + Read Replica
- CQRS(Query = MyBatis) 구조 특성상 읽기 트래픽 증가 시 Read Replica를 Query 전용으로 확장 가능한 구조로 설계되어 있다
- Redis(ElastiCache)는 인증 세션 및 캐시 전용으로 사용하여
- RDS 부하와 장애 전파를 분리한다.

백업/복구

- RDS 자동 백업 + 스냅샷
- 장애 시: 최신 스냅샷 복구 → 엔드포인트 교체(또는 프록시 도입)

5 . 운영 체크리스트 (배포 직후)

Front

CloudFront 캐시 무효화 확인
주요 페이지 로딩/라우팅 확인

Backend

/health 200
DB 연결
Redis 연결 (VPC DNS / SG 확인)
로그인/권한 API smoke test

AI

/health 200
S3 read 권한(Task Role) 확인
메뉴얼 응답 확인
PDF 1건 처리 smoke test

Code

.github/workflows/

gaekdam-be

```
name: deploy-s3-develop

on:
  pull_request:
    types: [closed]
    branches: [ "main" ]

permissions:
  contents: read

env:
  AWS_REGION: ${{ secrets.AWS_REGION }}
  S3_BUCKET_NAME: ${{ secrets.AWS_S3_BUCKET_NAME }}
  CLOUDFRONT_DISTRIBUTION_ID: ${{ secrets.CLOUDFRONT_DISTRIBUTION_ID }}

jobs:
  deploy:
    if: github.event.pull_request.merged == true
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v4

      - name: Setup Node.js
        uses: actions/setup-node@v4
        with:
          node-version: 20
          cache: npm

      - name: Install dependencies
        run: npm ci

      - name: Build
        run: npm run build
        env:
          VITE_API_BASE: ${{ secrets.VITE_API_BASE }}
          VITE_API_AI: ${{ secrets.VITE_API_AI }}
          VITE_API_KEY: ${{ secrets.VITE_API_KEY }}

      - name: Configure AWS credentials
        uses: aws-actions/configure-aws-credentials@v4
        with:
          aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          aws-region: ${{ env.AWS_REGION }}

      - name: Upload static files
        run: aws s3 sync dist s3://${{ env.S3_BUCKET_NAME }} --delete

      - name: Disable cache for index.html
        run: |
          aws s3 cp dist/index.html s3://${{ env.S3_BUCKET_NAME }}/index.html \
            --cache-control "no-store, no-cache, must-revalidate" \
            --content-type "text/html"

      - name: Invalidate CloudFront
        run: |
          aws cloudfront create-invalidation \
            --distribution-id ${{ env.CLOUDFRONT_DISTRIBUTION_ID }} \
            --paths "/"
```

gaekdam-fe

```
name: deploy-s3-develop

on:
  pull_request:
    types: [closed]
    branches: [ "main" ]

permissions:
  contents: read

env:
  AWS_REGION: ${{ secrets.AWS_REGION }}
  S3_BUCKET_NAME: ${{ secrets.AWS_S3_BUCKET_NAME }}
  CLOUDFRONT_DISTRIBUTION_ID: ${{ secrets.CLOUDFRONT_DISTRIBUTION_ID }}

jobs:
  deploy:
    if: github.event.pull_request.merged == true
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v4

      - name: Setup Node.js
        uses: actions/setup-node@v4
        with:
          node-version: 20
          cache: npm

      - name: Install dependencies
        run: npm ci

      - name: Build
        run: npm run build
        env:
          VITE_API_BASE: ${{ secrets.VITE_API_BASE }}
          VITE_API_AI: ${{ secrets.VITE_API_AI }}
          VITE_API_KEY: ${{ secrets.VITE_API_KEY }}

      - name: Configure AWS credentials
        uses: aws-actions/configure-aws-credentials@v4
        with:
          aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
          aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
          aws-region: ${{ env.AWS_REGION }}

      - name: Upload static files
        run: aws s3 sync dist s3://${{ env.S3_BUCKET_NAME }} --delete

      - name: Disable cache for index.html
        run: |
          aws s3 cp dist/index.html s3://${{ env.S3_BUCKET_NAME }}/index.html \
            --cache-control "no-store, no-cache, must-revalidate" \
            --content-type "text/html"

      - name: Invalidate CloudFront
        run: |
          aws cloudfront create-invalidation \
            --distribution-id ${{ env.CLOUDFRONT_DISTRIBUTION_ID }} \
            --paths "/"
```

gaekdam-ai

name: CI

```
on:
  push:
    branches: [ main, master, develop ]
  pull_request:
    branches: [ main, master, develop ]
  workflow_dispatch: {}

concurrency:
  group: ci-${{ github.ref }}
  cancel-in-progress: true

jobs:
  test:
    name: Fast tests (unit)
    runs-on: ubuntu-latest
    strategy:
      matrix:
        python-version: ['3.10', '3.11']
    steps:
      - name: Checkout
        uses: actions/checkout@v4

      - name: Set up Python
        uses: actions/setup-python@v4
        with:
          python-version: ${{ matrix.python-version }}

      - name: Cache pip
        uses: actions/cache@v4
        with:
          path: ~/.cache/pip
          key: ${{ runner.os }}-pip-${{ hashFiles('**/requirements.txt') }}-${{ matrix.python-version }}
          restore-keys:
            - ${{ runner.os }}-pip-

      - name: Install dependencies
        # Many projects include platform-specific heavy deps (faiss, torch). Try installing full requirements
        # but fall back to installing only test deps if full install fails to avoid build-time failures in CI.
        run: |
          python -m pip install --upgrade pip setuptools wheel
          set -o pipefail
          pip install -r requirements.txt || pip install pytest pytest-asyncio

      - name: Run pytest (fast, exclude slow)
        run: |
          mkdir -p reports
          pytest -q \
            --junitxml=reports/junit.xml \
            --html=reports/report.html --self-contained-html \
            --cov=app --cov-report=xml:reports/coverage.xml --cov-report=html:reports/coverage_html \
            -m "not slow"

    env:
      CI: true

    - name: Upload test reports
      uses: actions/upload-artifact@v4
      with:
        name: test-reports
        path: reports
```

```
slow-tests:
  name: Slow / bulk tests (manual)
  runs-on: ubuntu-latest
  if: github.event_name == 'workflow_dispatch'
  steps:
    - name: Checkout
      uses: actions/checkout@v4

    - name: Set up Python
      uses: actions/setup-python@v4
      with:
        python-version: '3.11'

    - name: Cache pip
      uses: actions/cache@v4
      with:
        path: ~/.cache/pip
        key: ${{ runner.os }}-pip-${{ hashFiles('**/requirements.txt') }}
        restore-keys: |
          ${{ runner.os }}-pip-

    - name: Install dependencies
      run: |
        python -m pip install --upgrade pip setuptools wheel
        pip install -r requirements.txt || pip install pytest pytest-asyncio

    - name: Run pytest (including slow)
      run: |
        mkdir -p reports
        pytest -q \
          --junitxml=reports/junit.xml \
          --html=reports/report.html --self-contained-html \
          --cov=app --cov-report=xml:reports/coverage.xml --cov-report=html:reports/coverage_html
      env:
        CI: true

    - name: Upload slow test reports
      uses: actions/upload-artifact@v4
      with:
        name: slow-test-reports
        path: reports
```

application.yml

```
server:
  port: 8082

spring:
  datasource:
    driver-class-name: org.mariadb.jdbc.Driver
    url: jdbc:mariadb://${DB_HOST}:${DB_PORT}/${DB_NAME}
    username: ${DB_USERNAME}
    password: ${DB_PASSWORD}

  mail:
    host: smtp.gmail.com
    port: 587
    username: ${MAIL_USERNAME}
    password: ${MAIL_PASSWORD}
    properties:
      mail:
        smtp:
          auth: true
          timeout: 5000
          starttls:
            enable: true
        mail.smtp:
          auth: true
          timeout: 5000
          starttls.enable: true

  data:
    redis:
      host: ${REDIS_HOST}
      port: ${REDIS_PORT}

  flyway:
    enabled: false

  jpa:
    hibernate:
      ddl-auto: update
      open-in-view: false
      properties:
        hibernate:
          # saveAll()을 실제 JDBC batch insert로 묶어주는 옵션
          jdbc:
            time_zone: Asia/Seoul
            batch_size: 500
            # 같은 테이블 insert/update를 묶어 배치 효율 증가
            order_inserts: true
            order_updates: true
          # 더미 생성시 SQL 로그가 (속도 위해 OFF)
          format_sql: false
          show_sql: false
        dialect: org.hibernate.dialect.MariaDBDialect

    sql:
      init:
        mode: never

mybatis:
  configuration:
    map-underscore-to-camel-case: true
    mapper-locations: classpath:/mappers/**/*.{xml}
    type-aliases-package: com.gaekdam.gaekdambe.**

jwt:
  secret-b64: ${JWT_SECRET_B64}
  access-expiration: 900000
  refresh-expiration: 259200000

crypto:
  hmac:
    pepper-b64: ${CRYPTO_HMAC_PEPPER_B64}
  local-kek:
    key-b64: ${CRYPTO_LOCAL_KEK_B64}

logging:
  level:
    org.apache.coyote.http11: INFO
    org.apache.tomcat.util.http.parser: DEBUG
    org.apache.tomcat.util.net: INFO
    # 레포트 로직 테스트용
    com.gaekdam.gaekdambe.analytics_service.report.dataset.query.service: DEBUG

management:
  endpoints:
    health:
      show-details: always
    health:
      redis.enabled: false
      db.enabled: true

app:
  cookie:
    secure: false
    same-site: Lax
```

application-prod.yml

```
server:  
  port: 8082  
  
spring:  
  mvc:  
    servlet:  
      load-on-startup: 1  
  
  jpa:  
    hibernate:  
      ddl-auto:  
  
  data:  
    redis:  
      url: rediss://${REDIS_HOST}:${REDIS_PORT}  
      timeout: 3s  
  
flyway:  
  enabled: true  
  baseline-on-migrate: true  
  locations: classpath:db/migration  
  
app:  
  cookie:  
    secure: true  
    same-site: None
```