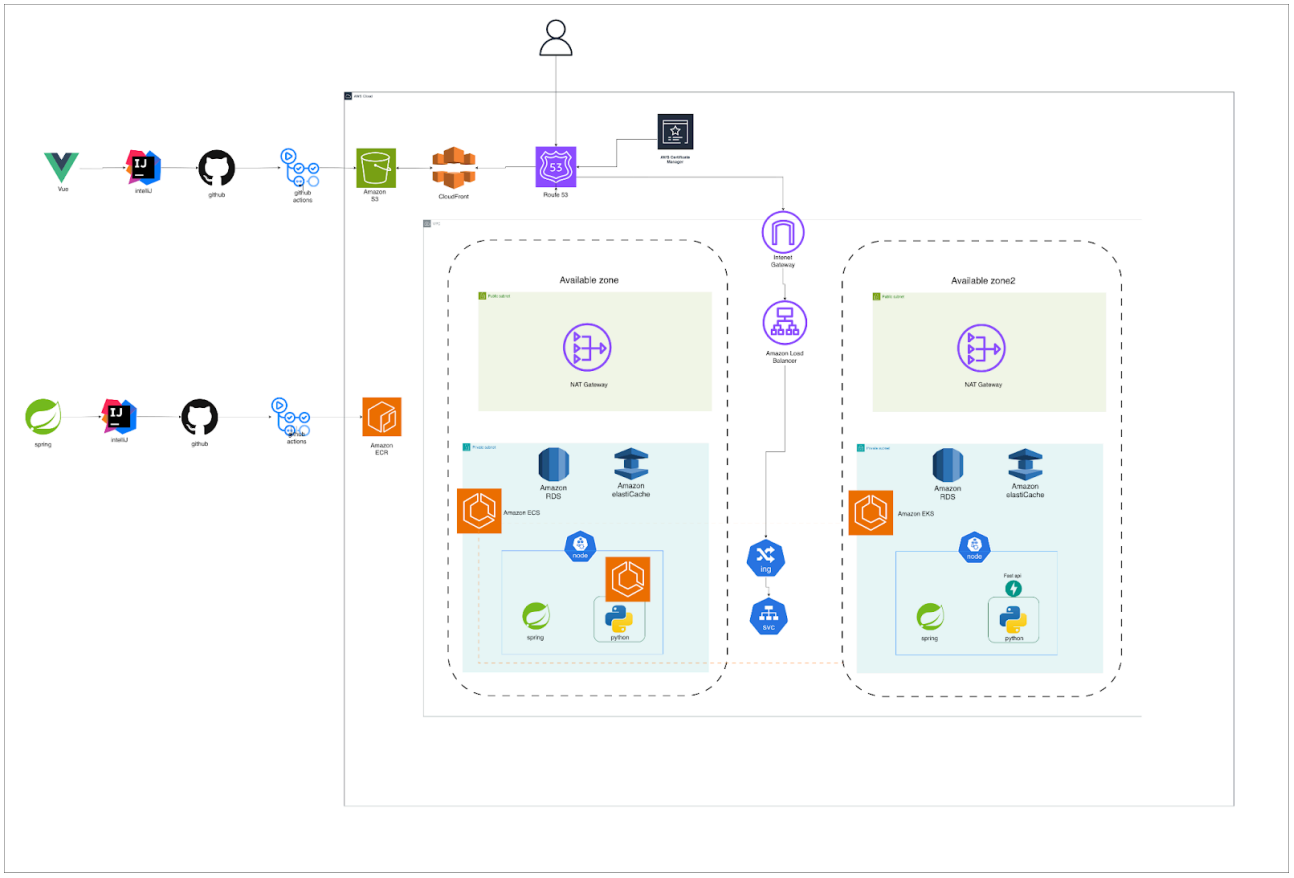
CI/CD 계획서



Infra

- **Front End : Route53, S3, CloudFront**
- **Back End : S3, ECS, AZ(1-2)**
- **Common : Route53, IAM, ALB, ACM**

Front End CI/CD

배포 프로세스

1. 개발자 변경 사항 Git Commit & Push (main 브랜치) → GitHub Action CI 실행
2. .dist 파일 → zip파일로 압축, S3 Bucket에 업로드
3. S3에 업로드 된 zip 파일 압축 해제, dist 내부 파일 정적 웹 호스팅 버킷 루트에 업로드
4. 캐시된 컨텐츠 갱신을 위해 CloudFront 캐시 무효화
5. 최신 버전 업로드

배포 전략

1. 정적 웹 자산의 무중단 배포
   1.1. S3 파일 교체 실시간 반영 및 캐시 무효화로 사용자에게 최신 버전 제공
2. RollBack
   2.1. 이전 파일 백업 및 재업로드로 롤백 가능
   2.2. 버전 관리 - S3에 build-2025-06-09-1230.zip 형식으로 보관하여 롤백 가능하게 구성
3. 모든 배포 로그 저장
   3.1. Github Action 로그 + S3 업로드 로그 기록

.github/workflows 코드

```yaml
name: deploy
on:
 push:
   branches: ["main"]
 workflow_dispatch:

permissions:
 contents: read

env:
 AWS_REGION: ap-northeast-2
 FRONT_PREFIX: front

jobs:
 deploy:
   runs-on: ubuntu-latest
#    env:
#      CLOUDFRONT_DISTRIBUTION_ID: ${{ secrets.CLOUDFRONT_DISTRIBUTION_ID }}

   steps:
     - name: Checkout
       uses: actions/checkout@v4

     - name: Setup Node.js
       uses: actions/setup-node@v4
       with:
         node-version: "24"
         cache: "npm"

     - name: Install dependencies
       run: npm ci

     - name: Build (Vue)
       run: npm run build
```

```yaml
    # Access Key 방식 AWS 인증
    - name: Configure AWS credentials (Access Key)
      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 }}

    # front/ 만 배포 (uploads/는 절대 건드리지 않음)
    - name: Deploy to S3 (front/ only)
      env:
        S3_BUCKET: ${{ secrets.S3_BUCKET }}
        FRONT_PREFIX: ${{ env.FRONT_PREFIX }}
      run: |
        set -euo pipefail

        DEST="s3://$S3_BUCKET/$FRONT_PREFIX"
        echo "Deploying dist/ -> $DEST"
        echo "NOTE: uploads/ will NOT be touched."

        # 1) index.html: 캐시 짧게 (배포 반영 빠르게)
        aws s3 cp dist/index.html "$DEST/index.html" \
          --cache-control "no-cache, no-store, must-revalidate" \
          --content-type "text/html"

        # 2) 나머지: 변경분만 업로드 + front/ 내에서만 찌꺼기 삭제
        #    Vite/Vue는 보통 해시 파일명이라 캐시 길게가 안전함
        aws s3 sync dist "$DEST" \
          --delete \
          --exclude "index.html" \
          --cache-control "public, max-age=31536000, immutable"

    # CloudFront 캐시 무효화 (최소화: index.html만)
    # secrets.CLOUDFRONT_DISTRIBUTION_ID 설정 시에만 수행
#     - name: CloudFront invalidate index.html (optional)
#       if: ${{ secrets.CLOUDFRONT_DISTRIBUTION_ID != '' }}
#       env:
#         DIST_ID: ${{ secrets.CLOUDFRONT_DISTRIBUTION_ID }}
#       run: |
#         set -euo pipefail
#         aws cloudfront create-invalidation \
#           --distribution-id "$DIST_ID" \
#           --paths "/index.html"
```

Back End CI/CD

배포 흐름

1. 개발자 변경 사항 Git Commit & Push (Main 브랜치) → GitHub Actions `deploy` 실행
2. JDK 21 세팅 → Gradle로 **빌드(테스트 제외)** 해서 애플리케이션 산출물 준비
3. AWS 자격증명(Access Key) 설정 → Amazon ECR 로그인
4. Docker 이미지 **build** → ECR에 **SHA 태그로 push** (선택으로 `latest`도 push)
5. `ecs/task-definition.json`에서 지정 컨테이너(`test-container`)의 **image를 새 SHA 이미지로 치환**(render)
6. 치환된 task definition으로 **새 Task Definition revision 등록**
7. ECS Service(`harvest-task-service-14ejfaz3`)가 새 리비전을 사용하도록 **업데이트 → 서비스 안정화(wait-for-service-stability)** 될 때까지 대기 → 배포 완료

배포 전략

- 자동화 수준
  - GitHub Actions를 통한 자동 배포
- 버전 관리
  - zip 파일을 GITHUB_SHA로 저장하여 롤백 대비
- 보안 관리
  - GitHub Secrets 활용하여 민감 정보 분리(application.properties, AccessKey, SecretKey 등)

.github/workflows 코드

```
name: deploy

on:

 push:

   branches: ["main"]



permissions:

 contents: read
```

```yaml
env:
  AWS_REGION: ap-northeast-2
  ECR_REPOSITORY: gold/harvest
  IMAGE_TAG: ${{ github.sha }}

  ECS_CLUSTER: harvest-cl
  ECS_SERVICE: harvest-task-service-14ejfaz3
  ECS_TASK_DEFINITION: ecs/task-definition.json
  ECS_CONTAINER_NAME: test-container

jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout
        uses: actions/checkout@v4
```

```yaml
    - name: Set up JDK 21
      uses: actions/setup-java@v4
      with:
        java-version: "21"
        distribution: "corretto"
        cache: "gradle"

    - name: Grant execute permission for gradlew
      run: chmod +x ./gradlew

    # 테스트는 스킵
    - name: Build (Gradle)
      run: ./gradlew clean build -x test

    # Access Key로 AWS 인증
    - name: Configure AWS credentials (Access Key)
      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 }}



    # ECR 로그인

    - name: Login to Amazon ECR

      id: login-ecr

      uses: aws-actions/amazon-ecr-login@v2



    # Docker build & push (SHA 태그)

    - name: Build, tag, and push image to ECR (sha)

      env:

        ECR_REGISTRY: ${{ steps.login-ecr.outputs.registry }}

      run: |

        echo "ECR_REGISTRY=$ECR_REGISTRY"

        docker build -t $ECR_REGISTRY/${{ env.ECR_REPOSITORY }}:${{ env.IMAGE_TAG
}} .

        docker push $ECR_REGISTRY/${{ env.ECR_REPOSITORY }}:${{ env.IMAGE_TAG }}
```

```
    # (선택) latest도 같이 유지하고 싶으면 남겨도 됨. 방법2에 필수는 아님.
    - name: Tag and push latest (optional)
      env:
        ECR_REGISTRY: ${{ steps.login-ecr.outputs.registry }}
      run: |
        docker tag  $ECR_REGISTRY/${{ env.ECR_REPOSITORY }}:${{ env.IMAGE_TAG }}
\
                    $ECR_REGISTRY/${{ env.ECR_REPOSITORY }}:latest
        docker push $ECR_REGISTRY/${{ env.ECR_REPOSITORY }}:latest

    # task-definition.json에서 컨테이너 'test-container'의 image만 SHA로 치환
    - name: Render ECS task definition
      id: render-task-def
      uses: aws-actions/amazon-ecs-render-task-definition@v1
      with:
        task-definition: ${{ env.ECS_TASK_DEFINITION }}
        container-name: ${{ env.ECS_CONTAINER_NAME }}
```

```
        image: ${{ steps.login-ecr.outputs.registry }}/${{ env.ECR_REPOSITORY }}:${{ env.IMAGE_TAG }}



    # 새 리비전 등록 + 서비스 롤링 배포

    - name: Deploy ECS task definition

      uses: aws-actions/amazon-ecs-deploy-task-definition@v2

      with:

        task-definition: ${{ steps.render-task-def.outputs.task-definition }}

        service: ${{ env.ECS_SERVICE }}

        cluster: ${{ env.ECS_CLUSTER }}

        wait-for-service-stability: true
```