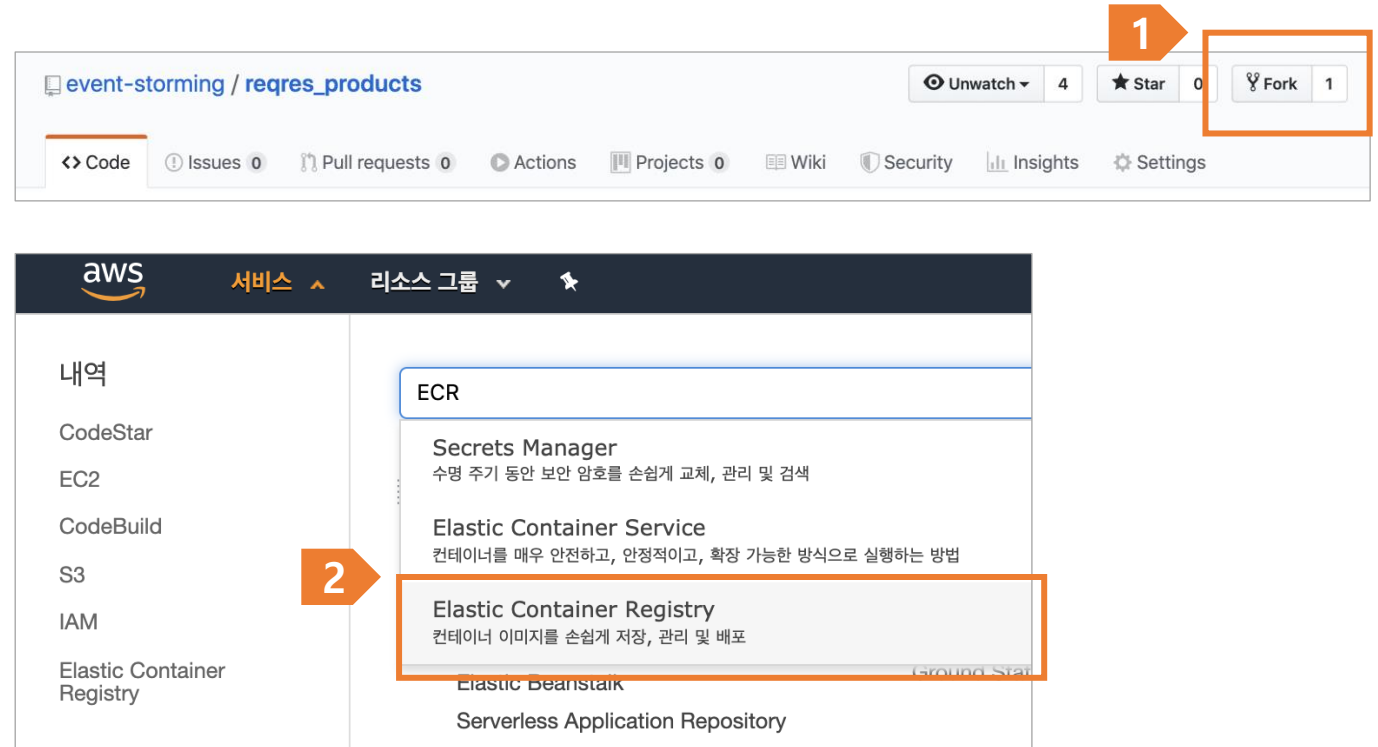


CodeBuild

- Github 의 소스를 빌드 / 패키징 하여 Docker Image 로 변경
- Docker Image 를 ECR 로 업로드
- 업로드된 Image 를 EKS(Amazon Elastic Kubernetes Service) 에 배포

1. Github 리파지토리 Fork
https://github.com/event-storming/reqres_products
2. Docker image 를 저장하기 위한 공간인 ECR 에 “user00-products” Repository 생성
3. Fork 받은 리파지토리의 Root에서 buildspec.yml을 편집하여 5행의 이미지명 수정



CodeBuild : 생성 설정 (1/3)

1. CodeBuild 프로젝트 생성
2. Github 계정 연결 후 Fork 한 리포지토리 연결

1 개발자 도구 > CodeBuild > 빌드 프로젝트 > 빌드 프로젝트 생성

빌드 프로젝트 생성

프로젝트 구성

프로젝트 이름

sample-products

프로젝트 이름은 2~255자여야 합니다. 글자(A-Z 및 a-z), 숫자(0-9) 및 특수 문자(- 및 _)를 포함할 수 있습니다.

설명 - 선택 사항

빌드 배치 - 선택 사항

☒ 빌드 배치 활성화

▶ 추가 구성

태그

2

소스

소스 추가

소스 1 - 기본

소스 공급자

GitHub

리포지토리

☐ 퍼블릭 리포지토리

☒ 내 GitHub 계정의 리포지토리

GitHub 리포지토리

https://github.com/kimscott/reqres_products.git

https://github.com/<user-name>/<repository-name>

연결 상태

OAuth를 사용하여 GitHub에 연결되었습니다.

GitHub에서 연결 해제

소스 버전 - 선택 사항 정보

풀 요청, 브랜치, 커밋 ID, 태그 또는 참조와 커밋 ID를 입력합니다.

▶ 추가 구성

Git clone 깊이, Git 하위 모듈

CodeBuild : 생성 설정 (2/3)

1. Webhook 을 설정하여 Github에 코드가 푸시될 때마다 트리거 동작
2. 빌드가 돌아갈 환경 설정
 - 운영체제 : Ubuntu
 - 런타임 : Standard
 - 이미지 : Standard4.0 (이미지별로 환경이 다르니 주의)
 - 환경유형 : Linux
 - 도커 권한 체크 필수

2

1

기본 소스 Webhook 이벤트 정보

필터 그룹 추가

Webhook - 선택 사항

- ☒ 코드 변경이 이 리포지토리에 푸시될 때마다 다시 빌드

한 개 이상의 Webhook 이벤트 필터 그룹을 추가하여 새 빌드를 트리거하는 이벤트를 지정합니다. Webhook 이벤트 필터 그룹을 추가하지 않은 경우에는 코드 변경이 리포지토리에 푸시될 때마다 새 빌드가 트리거됩니다.

Webhook 이벤트 필터 그룹 1

이벤트 유형

- ▶ 이러한 조건에서 빌드 시작
- ▶ 이러한 조건에서 빌드 시작 안 함

환경

환경 이미지

- ☒ 관리형 이미지
AWS CodeBuild에서 관리하는 이미지 사용

- ☐ 사용자 지정 이미지
도커 이미지 지정

운영 체제

Ubuntu

i 이제 프로그래밍 언어 런타임은 Ubuntu 18.04의 표준 이미지에 포함됩니다. 이 이미지는 콘솔에서 생성된 새 CodeBuild 프로젝트에 대해 권장됩니다. 자세한 내용은 CodeBuild에서 제공하는 Docker 이미지 [URL](#)을 참조하십시오.

런타임

Standard

이미지

aws/codebuild/standard: 4.0

이미지 버전

이 런타임 버전에 항상 최신 이미지 사용

환경 유형

Linux

권한이 있음

- ☒ 도커 이미지를 빌드하거나 빌드의 권한을 승격하려면 이 플래그를 활성화합니다.

서비스 역할

- ☒ 새 서비스 역할
계정에 서비스 역할 생성

- ☐ 기존 서비스 역할
계정에서 기존 서비스 역할 선택

역할 이름

codebuild-sample-products-service-role

서비스 역할 이름 입력

▶ 추가 구성

제한 시간, 인증서, VPC, 컴퓨팅 유형, 환경 변수, 파일 시스템

✓ 런타임 이미지 참조문서 : https://docs.aws.amazon.com/ko_kr/codebuild/latest/userguide/build-env-ref-available.html

CodeBuild : 생성 설정 (3/3)

1. 추가 구성에 환경 변수값을 입력 : AWS 계정 ID
 - AWS_ACCOUNT_ID
 - ECR 에서 이미지 주소의 가장 앞에 값임
 - echo \$(aws sts get-caller-identity --query Account --output text) 명령으로 조회 가능

2. buildspec.yml 파일을 사용 하겠다고 체크 후, 저장

1

▼ 추가 구성

제한 시간, 인증서, VPC, 컴퓨팅 유형, 환경 변수, 파일 시스템

환경 변수

이름	값	유형	
AWS_ACCOUNT_ID	97905023****	일반 텍스트 ▼	제거

환경 변수 추가

2

Buildspec

빌드 사양

☒ buildspec 파일 사용
YAML 형식의 buildspec 파일에 빌드 명령 저장

☐ 빌드 명령 삽입
빌드 명령을 빌드 프로젝트 구성으로 저장

Buildspec 이름 - 선택 사항

기본적으로 CodeBuild는 소스 코드 루트 디렉터리에서 buildspec.yml 파일을 찾습니다. buildspec 파일이 다른 이름 또는 위치를 사용하는 경우 여기에 소스 루트의 경로를 입력하십시오(예: buildspec-two.yml 또는 configuration/buildspec.yml).

아티팩트

아티팩트 추가

아티팩트 1 - 기본

유형

아티팩트 없음 ▼

테스트를 실행하거나 도커 이미지를 Amazon ECR에 넣는 경우 [아티팩트 없음]을 선택할 수 있습니다.

▶ 추가 구성

캐시, 암호화 키

CodeBuild : buildspec.yml

```
1  version: 0.2
2
3  env:
4    variables:
5      IMAGE_REPO_NAME: "products"
6
7  phases:
8    install:
9      runtime-versions:
10       java: corretto8
11   #    docker: 18
12   pre_build:
13     commands:
14       - echo Logging in to Amazon ECR...
15       - echo $IMAGE_REPO_NAME
16       - echo $AWS_ACCOUNT_ID
17       - echo $AWS_DEFAULT_REGION
18       - echo $CODEBUILD_RESOLVED_SOURCE_VERSION
19       - echo start command
20   #    - $(aws ecr get-login --no-include-email --region $AWS_DEFAULT_REGION)
21   build:
22     commands:
23       - echo Build started on `date`
24       - echo Building the Docker image...
25       - mvn package -Dmaven.test.skip=true
26   #    - docker build -t $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$CODEBUILD_RESOLVED_SOURCE_VERSION .
27   post_build:
28     commands:
29       - echo Build completed on `date`
30       - echo Pushing the Docker image...
31   #    - docker push $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$CODEBUILD_RESOLVED_SOURCE_VERSION
32
33   # cache:
34   #   paths:
35   #     - '/root/.m2/**/*'
```

1. buildspec 에 대한 상세 가이드 -
<https://docs.aws.amazon.com/ko-kr/codebuild/latest/userguide/build-spec-ref.html>
2. phases: 필수 시퀀스
3. 빌드의 각 단계 동안 CodeBuild가 실행하는 명령을 표시
4. Install, pre_build, build, post_build 4개의 시퀀스를 적절히 사용할 수 있음

CodeBuild : ECR Connection Policy 설정

The screenshot shows the AWS CodeBuild console. On the left, the '프로젝트 빌드' (Project Build) menu is expanded, and the '설정' (Settings) option is selected. The main panel displays the '환경' (Environment) configuration for the selected project. The '이미지' (Image) is set to 'aws/codebuild/standard:2.0' and the '환경 유형' (Environment Type) is 'Linux'. The '서비스 역할' (Service Role) is highlighted with an orange box and labeled with a '1' in an orange arrow. The role name is 'arn:aws:iam::979050235289:role/service-role/codebuild-sample-products-service-role'. Below this, the '제한 시간' (Timeout) is set to '1시간 0분'. The '레지스트리 자격 증명' (Registry Credentials) field is empty. Below the environment configuration, the '시각적 편집기' (Visual Editor) and 'JSON' tabs are visible. The 'JSON' tab is active, showing the IAM policy JSON. An orange box labeled with a '2' in an orange arrow highlights the 'Statement' array in the JSON, which contains a single statement with 'Action' set to a list of ECR actions and 'Effect' set to 'Allow'.

▼ 빌드 • CodeBuild

- 시작하기
- 프로젝트 빌드
- 프로젝트 빌드
- 설정
- 빌드 내역
- 보고서 그룹 베타
- 보고서 기록
- 계정 지표

환경

이미지	환경 유형
aws/codebuild/standard:2.0	Linux
서비스 역할	제한 시간
arn:aws:iam::979050235289:role/service-role/codebuild-sample-products-service-role	1시간 0분
레지스트리 자격 증명	-

시각적 편집기 JSON

```
1 {  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Action": [  
6         "ecr:BatchCheckLayerAvailability",  
7         "ecr:CompleteLayerUpload",  
8         "ecr:GetAuthorizationToken",  
9         "ecr:InitiateLayerUpload",  
10        "ecr:PutImage",  
11        "ecr:UploadLayerPart"  
12      ],  
13      "Resource": "*",  
14      "Effect": "Allow"  
15    }  
16  ]  
17 }
```

1. 프로젝트 빌드 - 빌드 선택 - 빌드 세부정보 - 환경 - 서비스 역할 클릭하여 IAM 역할 페이지로 이동
2. 인라인 정책 추가 > json 하여 ECR 관련 정책을 추가
3. 관련 스크립트는 WorkFlowy에 "CodeBuild 와 ECR 연결 정책" 검색
4. 정책명 입력 후, user00-codebuild-policy 검토 및 생성


CodeBuild : Run Pipeline (Docker Build/Push)

reqres_products / buildspec.yml

Cancel

↔ Edit file Preview changes Spaces 2 No wrap

```
7 phases:
8   install:
9     runtime-versions:
10      java: openjdk8
11      docker: 18
12   pre_build:
13     commands:
14       - echo Logging in to Amazon ECR...
15       - echo $IMAGE_REPO_NAME
16       - echo $AWS_ACCOUNT_ID
17       - echo $AWS_DEFAULT_REGION
18       - echo $CODEBUILD_RESOLVED_SOURCE_VERSION
19       - echo start command
20       - $(aws ecr get-login --no-include-email --region $AWS_DEFAULT_REGION)
21   build:
22     commands:
23       - echo Build started on `date`
24       - echo Building the Docker image...
25       - mvn package -Dmaven.test.skip=true
26       - docker build -t $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$CODEBUILD_RESOLVED_SOURCE_VERSION
27   post_build:
28     commands:
29       - echo Build completed on `date` |
30       - echo Pushing the Docker image...
31       - docker push $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$CODEBUILD_RESOLVED_SOURCE_VERSION
32
```

 Commit changes

Update buildspec.yml

Add an optional extended description...

☒ Commit directly to the `master` branch.

☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes Cancel

1. buildspec.yml 파일의 Docker 관련 주석을 모두 해제 후, 코드 커밋/푸쉬
2. 빌드 성공 후 ECR 에 Docker Image 가 정상적으로 생성 되었는지 확인


2

ECR에서 파이프라인으로 배포된 이미지 확인

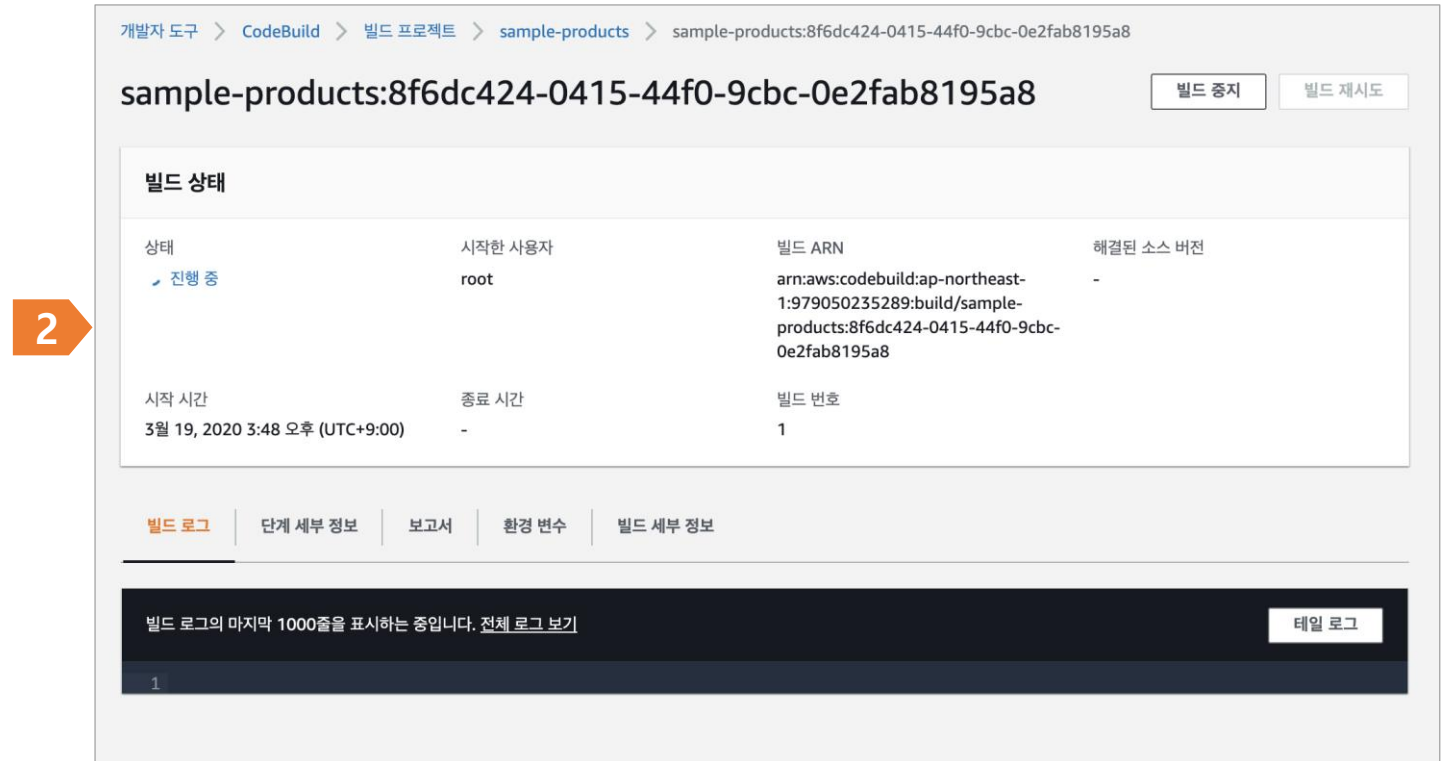
products

이미지 (1)

🔍 이미지 찾기

<input type="checkbox"/>	이미지 태그	이미지 URI
<input type="checkbox"/>	6dd5866a8971279f62376425d6f374e6fd682c9b	 979050235289.dkr.ecr.ap-northeast-1.amazonaws.com/products:6dd5866a8971279f62376425d6f374e

CodeBuild : 빌드 내역 확인



CodeBuild : Cache 적용

1 S3 서비스에서 Bucket 생성

aws 서비스 리소스 그룹

캐시 유형

Amazon S3

캐시 버킷

mvn-cache-codebuild

캐시 경로 접두사 - 선택 사항

캐시 수명 주기 - 선택 사항

경로 접두사에 따라 캐시 버킷 내 전체 또는 일부 객체에 수명 주기 만료 작업을 적용할 수 있습니다.


+ 만료 추가

취소 아티팩트 업데이트

1. Maven 으로 빌드시, 라이브러리를 캐쉬화 시킴
(캐쉬가 없으면 빌드때마다 1~2분 정도 더 소요됨)
2. 캐쉬를 저장할 S3 스토리지 생성
3. 버킷 생성 - 리전을 CodeBuild 생성 리전과 일치해야함
4. 빌드 선택 - 아티팩트 편집
5. 추가구성 - 캐시 유형을 S3 로 선택 후 버킷 선택
6. buildspec.yml 에서 cache 부분 주석 해제 후 커밋

CodeBuild : VM에서 EKS Connection 설정 (1/4)

1. 쿠버네티스에 접속 하기 위해서는 쿠버네티스 API 주소와 클러스터 어드민 권한이 있는 토큰이 있으면 접속 가능 (실제로 kubectl 명령어가 두개의 조합으로 이루어 져 있음)
2. 쿠버네티스 API 주소 위치

Amazon EKS 클러스터	Kubernetes 버전 1.15	플랫폼 버전 eks.1
Amazon ECR 리포지토리	API 서버 엔드포인트 	https://A4CB98CD955A2E274B343BCEA284621F.yl4.ap-northeast-1.eks.amazonaws.com

CodeBuild : VM에서 EKS Connection 설정 (2/4)

1. 쿠버네티스 토큰 생성은 ServiceAccount 생성 후 해당 SA 에 cluster-admin 롤을 주면 된다.
2. WorkFlowy 에서 “CodeBuild 와 EKS 연결” 검색 후 토큰 생성

```
→ ~ kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep eks-admin | a
wk '{print $1}')
Name:          eks-admin-token-hffgq
Namespace:     kube-system
Labels:        <none>
Annotations:   kubernetes.io/service-account.name: eks-admin
                kubernetes.io/service-account.uid: 18c3c8c6-a0dc-4f0b-9ba9-a85b6322aa7f

Type:          kubernetes.io/service-account-token

Data
====
namespace:    11 bytes
token:        eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJ1cm5ldGVzL3N1cnZpY2VhY2NvdW50Iiwia3ViZXJ
uZXRLcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJ1LXN5c3RlbSI6Imt1YmVybmV0ZXMuaW8vc2VydmljZWFjY29
```

CodeBuild : VM에서 EKS Connection 설정 (3/4)

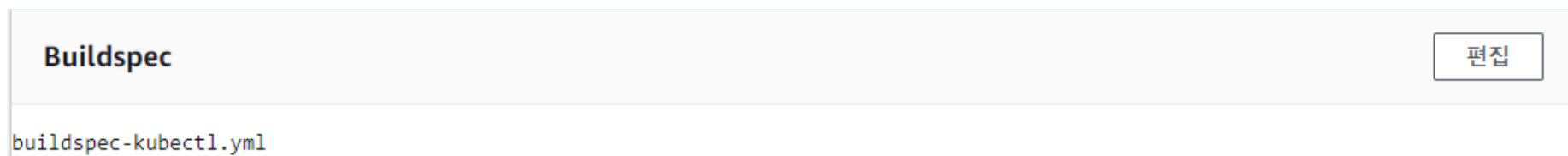
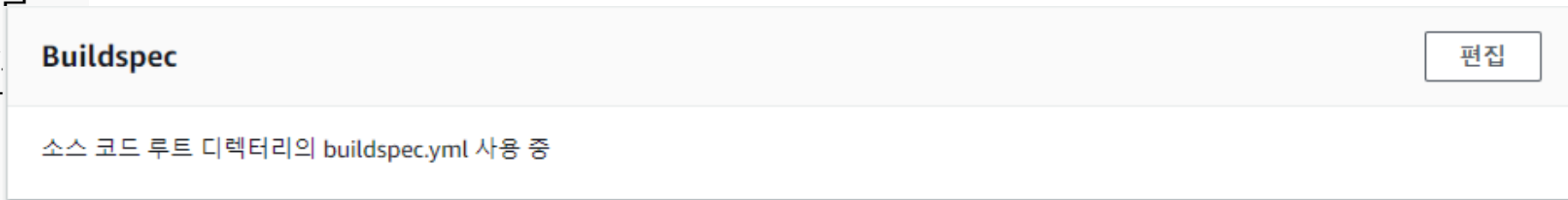
1. 빌드 파일의 “환경” 영역 편집 버튼을 눌러, 아래 값을 넣은 후에 저장한다.
2. 쿠버네티스 API 주소를 “KUBE_URL” 이라는 이름으로 저장한다.
3. 클러스터 어드민 토큰 값을 “KUBE_TOKEN” 이라는 이름으로 저장한다.

환경 변수 이름	값	유형	
AWS_ACCOUNT_ID	979050235****	일반 텍스트 ▼	제거
KUBE_URL	https://A4CB98CD955A	일반 텍스트 ▼	제거
KUBE_TOKEN	eyJhbGciOiJSUzI1NiIsIm	일반 텍스트 ▼	제거
<div>환경 변수 추가</div>			

CodeBuild : VM에서 EKS Connection 설정 (4/4)

1 파이프라인의 Buildspec을 편집하여, 레파지토리에 있는 buildsepc-kubectl.yaml로 수정한다.

2 buildsepc-kubectl.yaml 파일을 열어, _PROJECT_NAME을 내정보에 맞게 수정하여 저장한다.



CodeBuild : 파이프라인으로 생성된 결과 확인

- ✓ CodeBuild가 자동 실행 되어, EKS 클러스터에 배포 후 서비스가 정상적으로 기동 되었는지 확인한다.

- ✓ Kubectl get all

```
root@labs--1147635527:/home/project# kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/user30-products-5f4d77645f-2x86p  1/1     Running   0           45m

NAME                                TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
service/kubernetes                  ClusterIP     10.100.0.1   <none>        443/TCP    5h15m
service/user30-products             ClusterIP     10.100.202.42 <none>        8080/TCP   45m

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/user30-products      1/1     1             1           45m

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/user30-products-5f4d77645f  1         1         1       45m
root@labs--1147635527:/home/project#
```