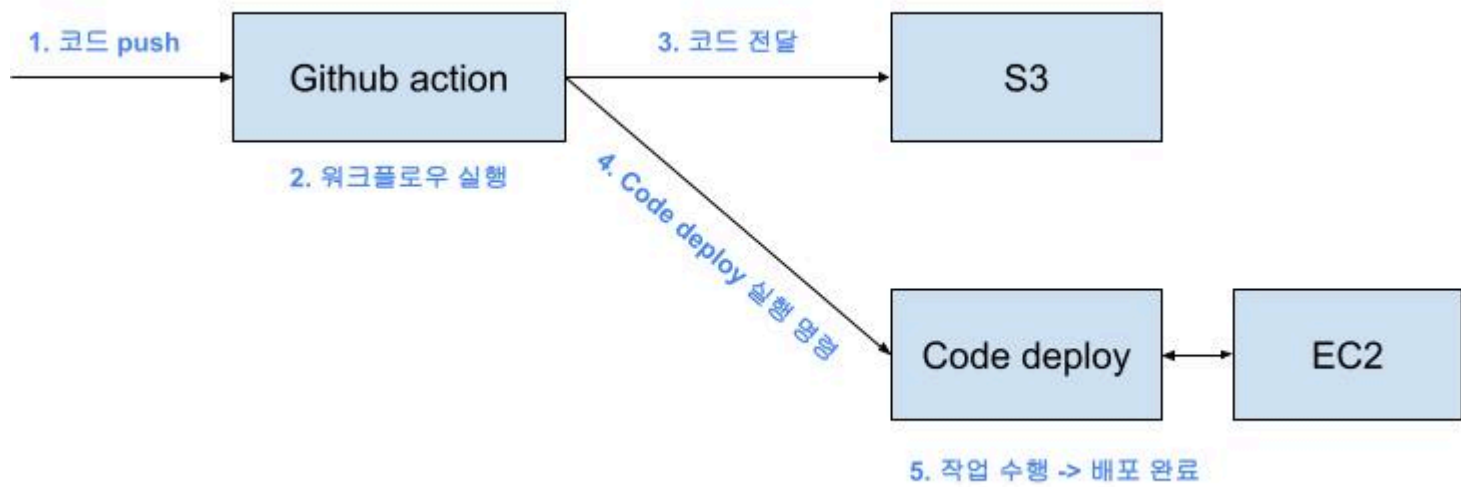


🔥 목표



오늘은 Github actions와 AWS Codedeploy를 이용해서 CI/CD자동화를 해보도록 하겠다!
사용자가 특정 브랜치에 코드를 PUSH하기만 하면 자동으로 빌드하고 서버에 배포까지 하도록 하는 것이다.

🌟 개발환경

프로젝트: Spring boot
빌드툴: Gradle
JDK: openjdk 17

깃헙 레포지토리

💻 개요

Part.1 환경 세팅

- IAM 세팅: 계정 생성, Github과 연동, 역할 생성
- 배포할 서버 세팅 (AWS EC2)
- S3 세팅: 빌드 결과물 저장
- Code deploy 세팅

Part.2 코드 작성

- Github actions 워크플로우 생성
- appspec.yml 작성
- 실행, 정지 script 작성

Part.3 배포 🔥

Part 1. 환경 세팅

★ IAM 세팅

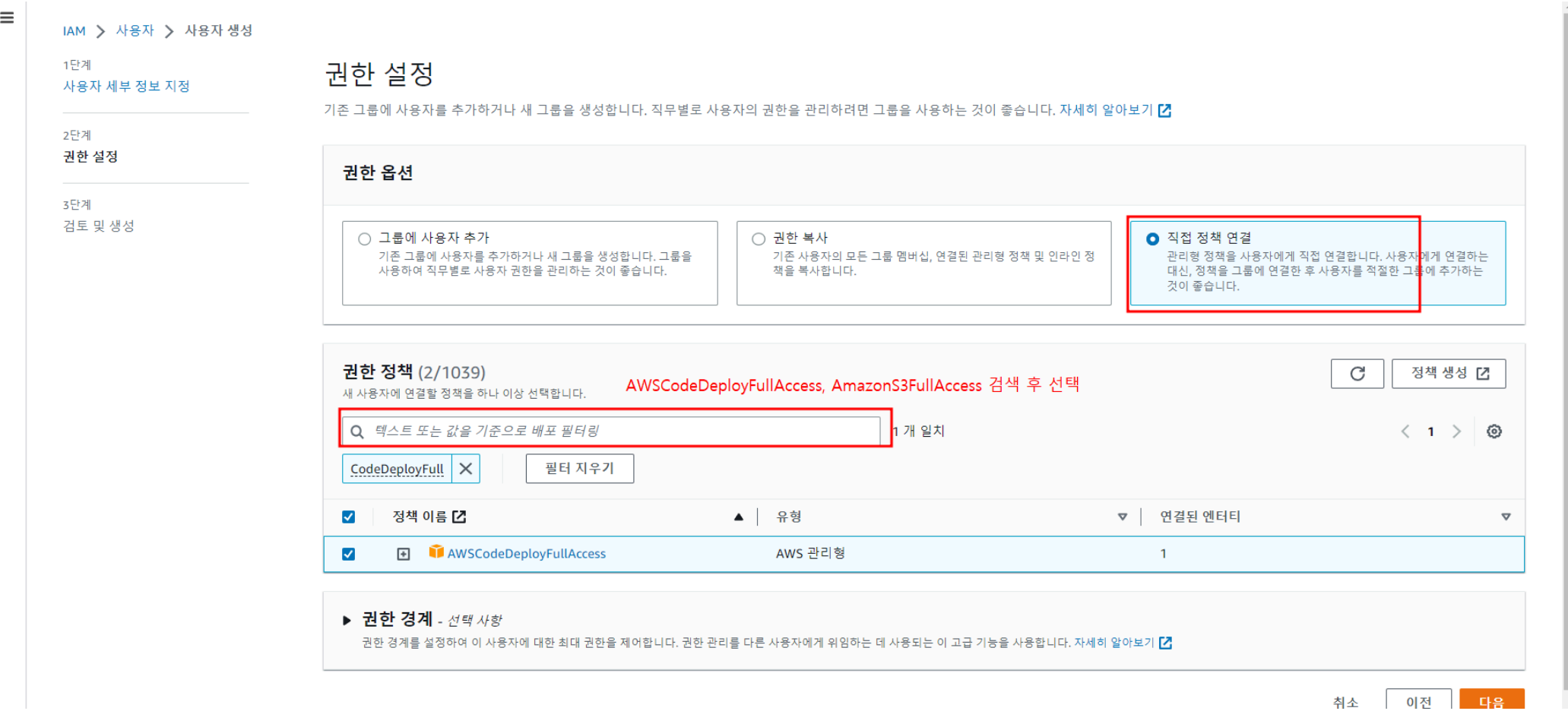
먼저 AWS에서 CICD를 전담할(?) IAM 계정과 필요한 역할을 생성해보겠다.

1) IAM 계정 생성

지금 생성할 IAM 계정은 CICD관련 작업에 사용할 계정이다.
먼저 IAM 콘솔에 접속한 후, 사용자를 생성해보자.



원하는대로 이름을 지어주고, 다음으로 넘어가자.




그러면 생성할 계정에 부여할 권한을 선택해야하는데, 아래의 2가지를 검색한 후 선택해주자

- AmazonS3FullAccess
- AWSCodeDeployFullAccess







그리고 다시 콘솔로 돌아오면, 아래와 같이 사용자가 정상적으로 생성된 것을 확인할 수 있다.

CICD-IAM

삭제

요약		
ARN  <code>arn:aws:iam::123456789012:user/CICD-IAM</code>	콘솔 액세스 비활성화됨	액세스 키 1 활성화되지 않음
생성됨 February 02, 2023, 15:14 (UTC+09:00)	마지막 콘솔 로그인 -	액세스 키 2 활성화되지 않음

- 권한
- 그룹
- 태그
- 보안 자격 증명
- 액세스 관리자


권한 정책 (2)				제거	권한 추가 ▼
사용자에게 직접 연결된 정책을 통해 또는 그룹을 통해 권한을 정의합니다.					
<input type="text" value="정책 찾기"/>			< 1 > 		
<input type="checkbox"/>	정책 이름 	▲ 유형 ▼	연결 방식: 		
<input type="checkbox"/>	 AmazonS3FullAccess	AWS 관리형	직접		
<input type="checkbox"/>	 AWSCodeDeployFullAccess	AWS 관리형	직접		

2) 키페어 발급


이제 Github에 방금 생성한 계정의 Keypair를 입력해줄 것이다.
그러니 Keypair부터 발급받아 보자.

CICD-IAM

삭제

요약		
ARN  <code>arn:aws:iam::123456789012:user/CICD-IAM</code>	콘솔 액세스 비활성화됨	액세스 키 1 활성화되지 않음
생성됨 February 02, 2023, 15:14 (UTC+09:00)	마지막 콘솔 로그인 -	액세스 키 2 활성화되지 않음

- 권한
- 그룹
- 태그
- 보안 자격 증명
- 액세스 관리자

콘솔 로그인		콘솔 액세스 활성화
콘솔 로그인 링크  <code>https://251158812353.signin.aws.amazon.com/console</code>	콘솔 암호 활성화되지 않음	

멀티 팩터 인증(MFA) (0)

액세스 키 (0)

액세스 키를 사용하여 AWS CLI, AWS Tools for PowerShell, AWS SDK 또는 직접 AWS API 호출을 통해 AWS에 프로그래밍 방식 호출을 전송합니다. 한 번에 최대 두 개의 액세스 키(활성 또는 비활성)를 가질 수 있습니다. [Learn more](#)

[액세스 키 만들기](#)

액세스 키 없음

액세스 키와 같은 장기 자격 증명을 사용하지 않는 것이 모범 사례입니다. 대신 단기 자격 증명을 제공하는 도구를 사용하세요. [Learn more](#)

[액세스 키 만들기](#)

생성한 계정의 콘솔 페이지에서 액세스 키를 생성할 수 있는 탭으로 이동하고, "액세스 키 만들기"버튼을 클릭한다.

[IAM](#) > [사용자](#) > [CICD-IAM](#) > [액세스 키 만들기](#)

1단계

액세스 키 모범 사례 및 대안

2단계 - 선택 사항

설명 태그 설정

3단계

액세스 키 검색

액세스 키 모범 사례 및 대안

보안 개선을 위해 액세스 키와 같은 장기 자격 증명을 사용하지 마세요. 다음과 같은 사용 사례와 대안을 고려하세요.

☐ Command Line Interface(CLI)

AWS CLI를 사용하여 AWS 계정에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.

☐ 로컬 코드

로컬 개발 환경의 애플리케이션 코드를 사용하여 AWS 계정에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.

☐ AWS 컴퓨팅 서비스에서 실행되는 애플리케이션

Amazon EC2, Amazon ECS 또는 AWS Lambda와 같은 AWS 컴퓨팅 서비스에서 실행되는 애플리케이션 코드를 사용하여 AWS 계정에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.

☐ 서드 파티 서비스

AWS 리소스를 모니터링 또는 관리하는 서드 파티 애플리케이션 또는 서비스에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.

☐ AWS 외부에서 실행되는 애플리케이션

애플리케이션을 온프레미스 호스트에서 실행하거나 로컬 AWS 클라이언트 또는 서드 파티 AWS 플러그 인을 사용할 수 있도록 이 액세스 키를 사용할 것입니다.

☒ 기타

귀하의 사용 사례가 여기에 나열되어 있지 않습니다.



이 사용 사례에서는 액세스 키를 사용해도 되지만 모범 사례를 따릅니다.

- 액세스 키를 일반 텍스트 코드 리포지토리 또는 코드 저장해서는 안 됩니다

그러면 뭔가를 고르는 화면이 나오는데 나는 '기타'를 선택해줬다 ㅎㅎ

1단계
액세스 키 모범 사례 및 대안

2단계 - 선택 사항
설명 태그 설정

3단계
액세스 키 검색

설명 태그 설정 - 선택 사항

이 액세스 키에 대한 설명은 이 사용자에게 태그로 연결되고, 액세스 키와 함께 표시됩니다.

설명 태그 값

이 액세스 키의 용도와 사용 위치를 설명합니다. 좋은 설명은 나중에 이 액세스 키를 자신있게 교체하는 데 유용합니다.

GithubActionKeypair

최대 256자까지 가능합니다. 허용되는 문자는 문자, 숫자, UTF-8로 표현할 수 있는 공백 및 _./=+-@입니다.

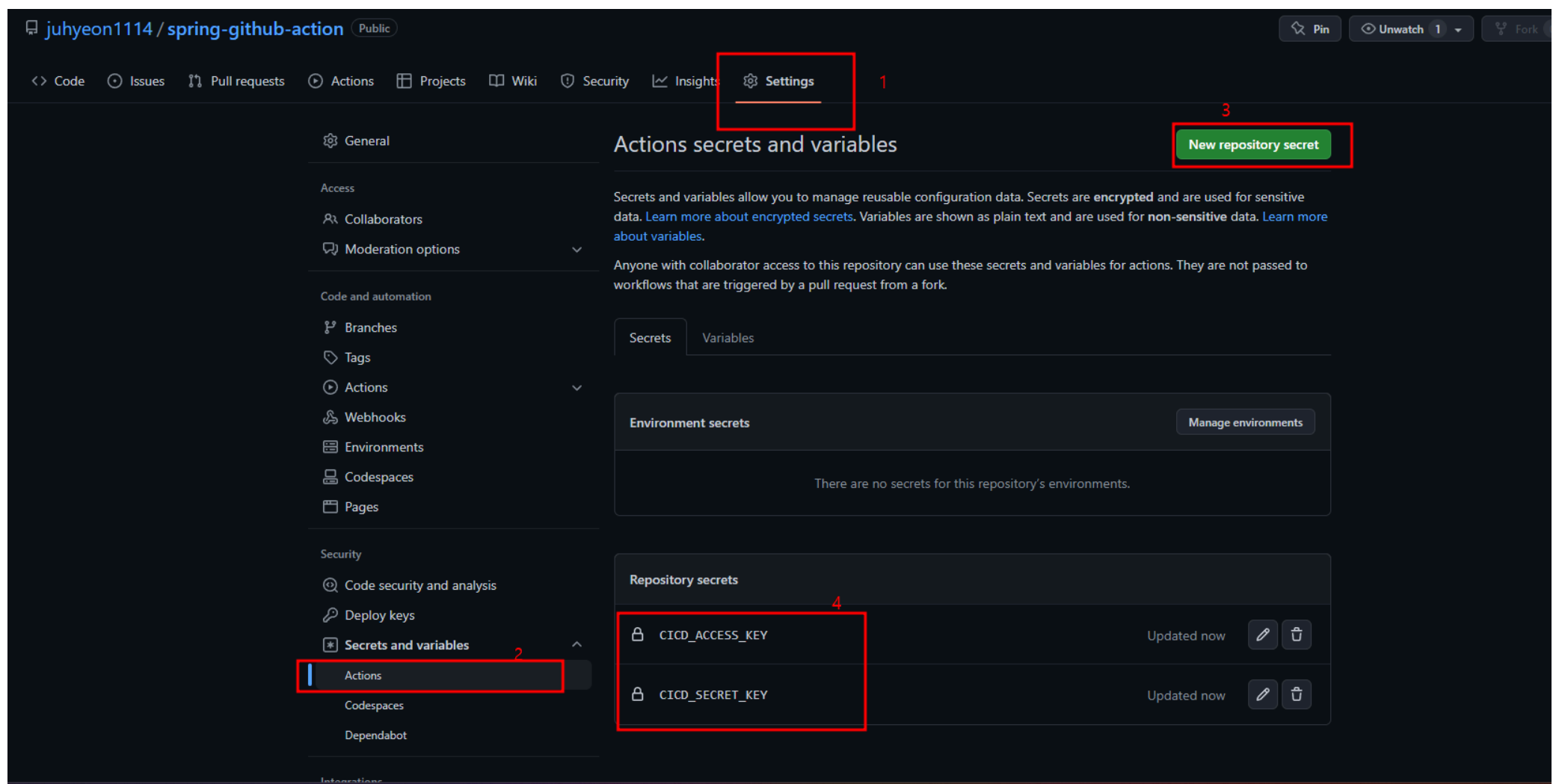
취소

이전

액세스 키 만들기

그리고 태그값을 본인이 구분하기 쉽게 입력하고, 키페어를 생성하게 되면, Access key와 Secret key를 발급받게 된다. Secret key같은 경우는 한번만 볼 수 있고, 다시 볼 수 없게 되니까 꼭! 발급받은 키페어를 안전한 곳 어딘가에 메모해두자.

3) Github에 키페어 입력하기



배포할 코드가 있는 깃헙 레포지토리로 이동해서 발급받은 Keypair를 입력해주자. 입력할 때, 각 값의 이름은 아무렇게나 지어도 된다.

이렇게 하면 이제 Github에서 나의 S3와 Code deploy에 접근할 수 있게 된다. 🍌

4) IAM 역할 생성

이따가 필요하게 될 IAM 역할 2가지를 미리 생성해놓도록 하자.

IAM > 역할 콘솔에 들어가면, 역할을 생성할 수 있다.

4-1) EC2에서 S3 접근을 위한 IAM 역할

신뢰할 수 있는 엔터티 선택 정보

신뢰할 수 있는 엔터티 유형

- ☒ **AWS 서비스**
EC2, Lambda 등의 AWS 서비스가 이 계정에서 작업을 수행하도록 허용합니다.
- ☐ **AWS 계정**
사용자 또는 서드 파티에 속한 다른 AWS 계정의 엔터티가 이 계정에서 작업을 수행하도록 허용합니다.
- ☐ **웹 자격 증명**
지정된 외부 웹 자격 증명 공급자와 연동된 사용자가 이 역할을 맡아 이 계정에서 작업을 수행하도록 허용합니다.
- ☐ **SAML 2.0 연동**
기업 디렉터리에서 SAML 2.0과 연동된 사용자가 이 계정에서 작업을 수행할 수 있도록 허용합니다.
- ☐ **사용자 지정 신뢰 정책**
다른 사용자가 이 계정에서 작업을 수행할 수 있도록 사용자 지정 신뢰 정책을 생성합니다.

사용 사례

EC2, Lambda 등의 AWS 서비스가 이 계정에서 작업을 수행하도록 허용합니다.

일반 사용 사례

- ☒ **EC2**
Allows EC2 instances to call AWS services on your behalf.
- ☐ **Lambda**
Allows Lambda functions to call AWS services on your behalf.

다른 AWS 서비스의 사용 사례:

사용 사례를 조회할 서비스 선택

취소

다음

권한 추가 정보

권한 정책 (선택됨 1/811) 정보

새 역할에 연결할 정책을 하나 이상 선택합니다.

Q 속성 또는 정책 이름을 기준으로 정책을 필터링하고 Enter를 누릅니다.

5 개 일치

"AmazonS3" X

필터 지우기

	정책 이름	유형	설명
<input checked="" type="checkbox"/>	<div><div></div>AmazonS3FullAccess</div>	AWS 관...	Provides full access to all buckets via the AWS Management Console.
<input type="checkbox"/>	<div><div></div>AmazonS3ReadOnlyAccess</div>	AWS 관...	Provides read only access to all buckets via the AWS Management Console.
<input type="checkbox"/>	<div><div></div>AmazonS3OutpostsFullAccess</div>	AWS 관...	Provides full access to Amazon S3 on Outposts via the AWS Management Console.
<input type="checkbox"/>	<div><div></div>AmazonS3ObjectLambdaExecutionRolePolicy</div>	AWS 관...	Provides AWS Lambda functions permissions to interact with Amazon S3 Object Lambda.
<input type="checkbox"/>	<div><div></div>AmazonS3OutpostsReadOnlyAccess</div>	AWS 관...	Provides read only access to Amazon S3 on Outposts via the AWS Management Console.

▶ 권한 경계 설정 - 선택 사항 정보

권한 경계를 설정하여 이 역할이 가질 수 있는 최대 권한을 제어합니다. 이는 일반적인 설정은 아니지만 권한 관리를 다른 사용자에게 위임하는 데 사용할 수 있습니다.

이름 지정, 검토 및 생성

역할 세부 정보

역할 이름

이 역할을 식별하는 의미 있는 이름을 입력합니다.

MyS3AccessRole

최대 64자입니다. 영숫자 및 '+', '@', '_' 문자를 사용하세요.

설명

이 역할에 대하여 간단한 설명을 추가합니다.

Role for access to S3 from EC2

최대 1,000자입니다. 영숫자 및 '+', '@', '_' 문자를 사용하세요.

1단계: 신뢰할 수 있는 엔터티 선택

```
1 {  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Effect": "Allow",  
6       "Action": "s3:*",  
7       "Resource": "*" }  
8     ]  
9 }
```

이 순서대로 생성하면 된다.
이 역할은 이따가 생성할 EC2에게 부여해줄 것이다.

역할의 이름은 자유롭게 만들어주면 된다.
작명 팁은 AWS에서 기본적으로 제공되는 수 많은 역할들과는 조금 다른 느낌(?)으로 이름을 짓는 것을 권장한다. 🤔
왜냐하면 기본 역할과 내가 만든 역할을 쉽게 구분하기 위해서 그렇다. 👍

4-2) Code deploy를 위한 IAM 역할

IAM > 역할 > 역할 생성

1단계
신뢰할 수 있는 엔터티 선택

2단계
권한 추가

3단계
이름 지정, 검토 및 생성

신뢰할 수 있는 엔터티 선택 정보

신뢰할 수 있는 엔터티 유형

☒ AWS 서비스
EC2, Lambda 등의 AWS 서비스가 이 계정에서 작업을 수행하도록 허용합니다.

☐ AWS 계정
사용자 또는 서드 파티에 속한 다른 AWS 계정의 엔터티가 이 계정에서 작업을 수행하도록 허용합니다.

☐ 웹 자격 증명
지정된 외부 웹 자격 증명 공급자와 연동된 사용자가 이 역할을 맡아 이 계정에서 작업을 수행하도록 허용합니다.

☐ SAML 2.0 연동
기업 디렉터리에서 SAML 2.0과 연동된 사용자가 이 계정에서 작업을 수행할 수 있도록 허용합니다.

☐ 사용자 지정 신뢰 정책
다른 사용자가 이 계정에서 작업을 수행할 수 있도록 사용자 지정 신뢰 정책을 생성합니다.

사용 사례

EC2, Lambda 등의 AWS 서비스가 이 계정에서 작업을 수행하도록 허용합니다.

일반 사용 사례

☐ EC2
Allows EC2 instances to call AWS services on your behalf.

☐ Lambda
Allows Lambda functions to call AWS services on your behalf.

다른 AWS 서비스의 사용 사례:

CodeDeploy

☒ CodeDeploy
Allows CodeDeploy to call AWS services such as Auto Scaling on your behalf.

☐ CodeDeploy for Lambda
Allows CodeDeploy to route traffic to a new version of an AWS Lambda function version on your behalf.

IAM > 역할 > 역할 생성

1단계
신뢰할 수 있는 엔터티 선택

2단계
권한 추가

3단계
이름 지정, 검토 및 생성

권한 추가 정보

권한 정책 (1) 정보

선택한 역할 유형에는 다음 정책이 필요합니다.

정책 이름	유형	연결된 엔터티
AWSCodeDeployRole	AWS 관...	0

▶ 권한 경계 설정 - 선택 사항 정보

권한 경계를 설정하여 이 역할이 가질 수 있는 최대 권한을 제한합니다. 이는 일반적인 설정은 아니지만 권한 관리를 다른 사용자에게 위임하는 데 사용할 수 있습니다.

취소

이전

다음

1단계
신뢰할 수 있는 엔터티 선택

2단계
권한 추가

3단계
이름 지정, 검토 및 생성

이름 지정, 검토 및 생성

역할 세부 정보

역할 이름
이 역할을 식별하는 의미 있는 이름을 입력합니다.

MyCodeDeployRole

최대 64자입니다. 영숫자 및 '+', '@', '_' 문자를 사용하세요.

설명
이 역할에 대하여 간단한 설명을 추가합니다.

Role for my code deploy

최대 1,000자입니다. 영숫자 및 '+', '@', '_' 문자를 사용하세요.

1단계: 신뢰할 수 있는 엔터티 선택

1. {
2. "Version": "2012-10-17"

Code deploy를 위한 역할은 처음에 역할을 생성할 때, Code deploy를 검색해서 선택해주고 이전에 했던 것과 동일한 절차를 거쳐주면 된다.

★ EC2 세팅

1) 인스턴스 생성

EC2 > 인스턴스 콘솔로 이동한 후, 새로운 인스턴스를 하나 만들어주자.

EC2 > 인스턴스 > 인스턴스 시작

인스턴스 시작

정보

Amazon EC2를 사용하면 AWS 클라우드에서 실행되는 가상 머신 또는 인스턴스를 생성할 수 있습니다. 아래의 간단한 단계에 따라 빠르게 시작할 수 있습니다.

이름 및 태그

정보

이름

cicdtest

추가 태그 추가

▼ 애플리케이션 및 OS 이미지(Amazon Machine Image)

정보

AMI는 인스턴스를 시작하는 데 필요한 소프트웨어 구성(운영 체제, 애플리케이션 서버 및 애플리케이션)이 포함된 템플릿입니다. 아래에서 찾고 있는 항목이 보이지 않으면 AMI를 검색하거나 찾아보십시오.

수천 개의 애플리케이션 및 OS 이미지를 포함하는 전체 카탈로그 검색

최근 사용

Quick Start

Amazon Linux

macOS

Ubuntu

Windows

Red Hat

S

더 많은 AMI 찾아보기

AWS, Marketplace 및 커뮤니티의 AMI 포함

Amazon Machine Image(AMI)

Ubuntu Server 20.04 LTS (HVM), SSD Volume Type

ami-051d287d02a19035f (64비트(x86)) / ami-09a9b3cefb0428387 (64비트(Arm))

가상화: hvm ENA 활성화됨: true 루트 디바이스 유형: ebs

프리 티어 사용 가능

▼ 요약

인스턴스 개수

정보

1

소프트웨어 이미지(AMI)

Canonical, Ubuntu, 20.04 LTS, ...더 보기

ami-051d287d02a19035f

가상 서버 유형(인스턴스 유형)

t2.micro

방화벽(보안 그룹)

새 보안 그룹

스토리지(볼륨)

1개의 볼륨 – 8GiB

취소

인스턴스 시작

내가 쓰기 편한 OS를 선택하면 되는데, 나는 ubuntu로 설정을 하도록 하겠다. 그리고 ubuntu의 버전은 20.04 버전을 하겠다.

22.04 버전의 경우 Code deploy agent를 설치하는데 있어서 뭔가 매끄럽지가 못해서, 설치 매뉴얼이 존재하는 20.04버전을 선택했다.

네트워크 설정

네트워크 설정은 EC2를 위치시킬 VPC와 서브넷을 선택해주면 된다.

키페어

키페어는 필요하다면, 새롭게 발급 받아서 연결해주면 된다.
나는 이 서버만을 위한 키페어를 새롭게 발급받았다.

보안그룹

EC2 > 보안 그룹 > sg-0b7717137d9106b2d - cidc-test-ec2-sg > 인바운드 규칙 편집

인바운드 규칙 편집 정보

인바운드 규칙 정보

보안 그룹 규칙 ID

유형 정보

프로토콜 정보

포트 범위 정보

소스 정보

설명 - 선택 사항 정보

sg-035ef88d57b40595d

HTTP

TCP

80

사용자 지정

Q

0.0.0.0/0 X

삭제

sg-0b732fb7cefa55273

SSH

TCP

22

사용자 지정

Q

0.0.0.0/0 X

삭제

sg-0acfd4876b63a543f

HTTPS

TCP

443

사용자 지정

Q

0.0.0.0/0 X

삭제

sg-0eda3ebdd78fcdab00

사용자 지정 TCP

TCP

8080

사용자 지정

Q

0.0.0.0/0 X

삭제

규칙 추가

취소

변경 사항 미리 보기

규칙 저장

보안그룹은 이렇게 미리 만들어두고, 연결해주자.
나는 8080포트로 스프링을 돌릴거여서 8080포트도 열어주었다.

2) EIP 생성, 연결

탄력적 IP 주소 (1/4)

작업

세부 정보 보기

탄력적 IP 주소 릴리스

탄력적 IP 주소 연결

탄력적 IP 주소 연결 해제

역방향 DNS 업데이트

전송 활성화

탄력적 IP 주소 할당

<input type="checkbox"/>	Name	할당된 IPv4 주소	유형	할당 ID	역방향 DNS 레코드	연결된 인스턴스
<input type="checkbox"/>	elastic-ip-1	3.37.2.45	퍼블릭 IP	elastic-ip-1	-	elastic-instance-1
<input checked="" type="checkbox"/>	cicdtest-eip	3.37.2.45	퍼블릭 IP	elastic-ip-2	-	-
<input type="checkbox"/>	elastic-ip-2	3.37.2.45	퍼블릭 IP	elastic-ip-3	-	elastic-instance-2

탄력적 IP를 하나 생성해서, EC2에 연결해주자.
EIP생성, 연결은 간단해서 길게 이야기하지는 않겠다.

3) CodeDeploy agent 설치

이제 EC2 서버에 CodeDeploy Agent를 설치해줄 차례이다.
Ubuntu를 위한 agent 설치 매뉴얼을 AWS에서 제공하고 있는데, 이 매뉴얼을 참고해서 서버에 Agent프로그램을 설치해주자.

Ubuntu 20.04버전으로 EC2를 만들었다면, 서버에 접속한 후 아래의 명령어를 순차적으로 실행해주면 된다.

3-1) 설치

```
sudo apt update
sudo apt install ruby-full
sudo apt install wget
cd /home/ubuntu
wget https://aws-codedeploy-ap-northeast-2.s3.ap-northeast-2.amazonaws.com/latest/install
chmod +x ./install
sudo ./install auto > /tmp/logfile
```

3-2) 실행 확인

```
sudo service codedeploy-agent status
```

정상적으로 설치가 됐다면 이렇게, 서비스가 잘 돌아가고 있다고 메시지가 출력될 것이다.

```
● codedeploy-agent.service - LSB: AWS CodeDeploy Host Agent
   Loaded: loaded (/etc/init.d/codedeploy-agent; generated)
   Active: active (running) since Thu 2023-02-02 07:00:27 UTC; 2s ago
     Docs: man:systemd-sysv-generator(8)
    Tasks: 3 (limit: 1143)
   Memory: 66.3M
    CGroup: /system.slice/codedeploy-agent.service
            └─3393 codedeploy-agent: master 3393
               └─3395 codedeploy-agent: InstanceAgent::Plugins::CodeDeployPlugin::CommandPoller of master 3393

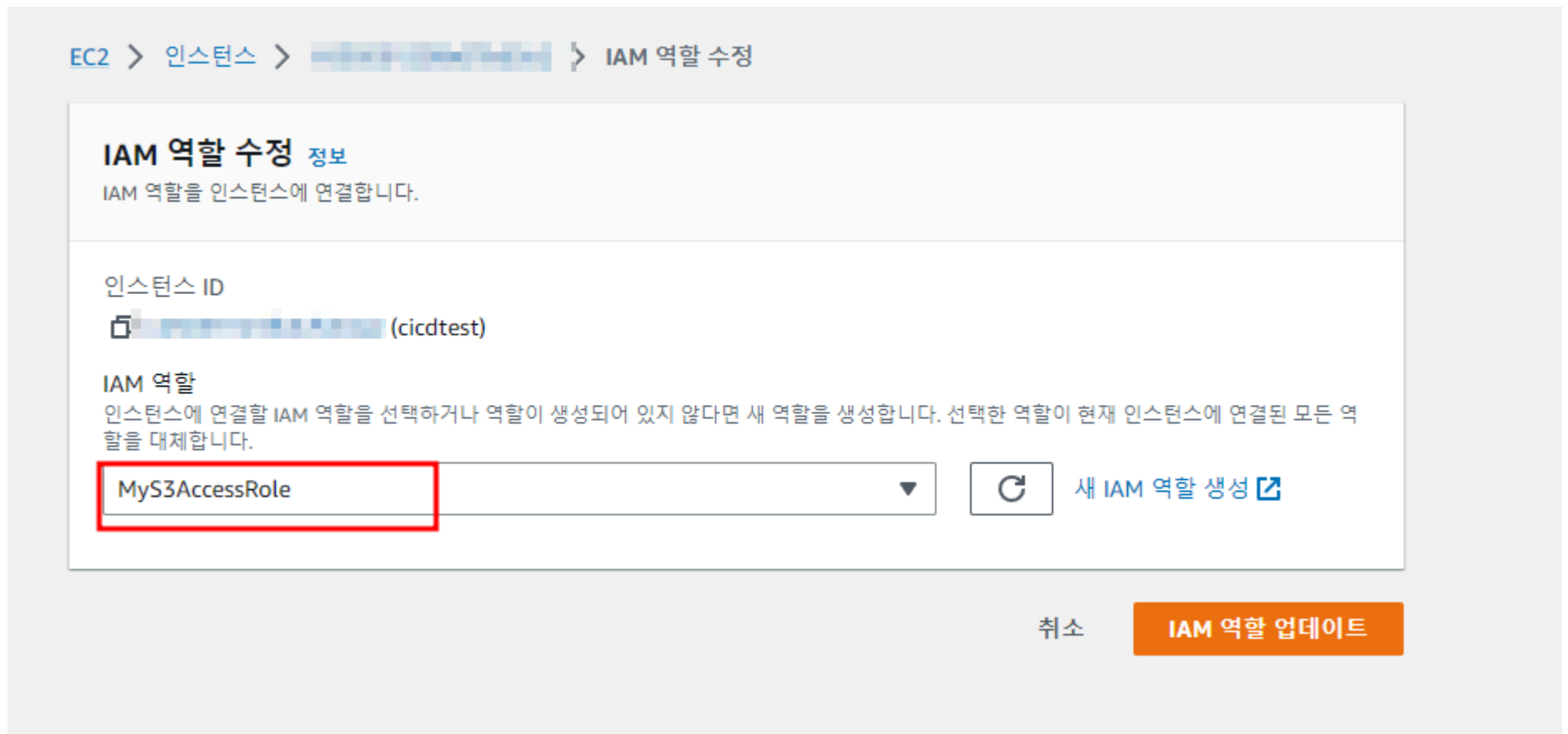
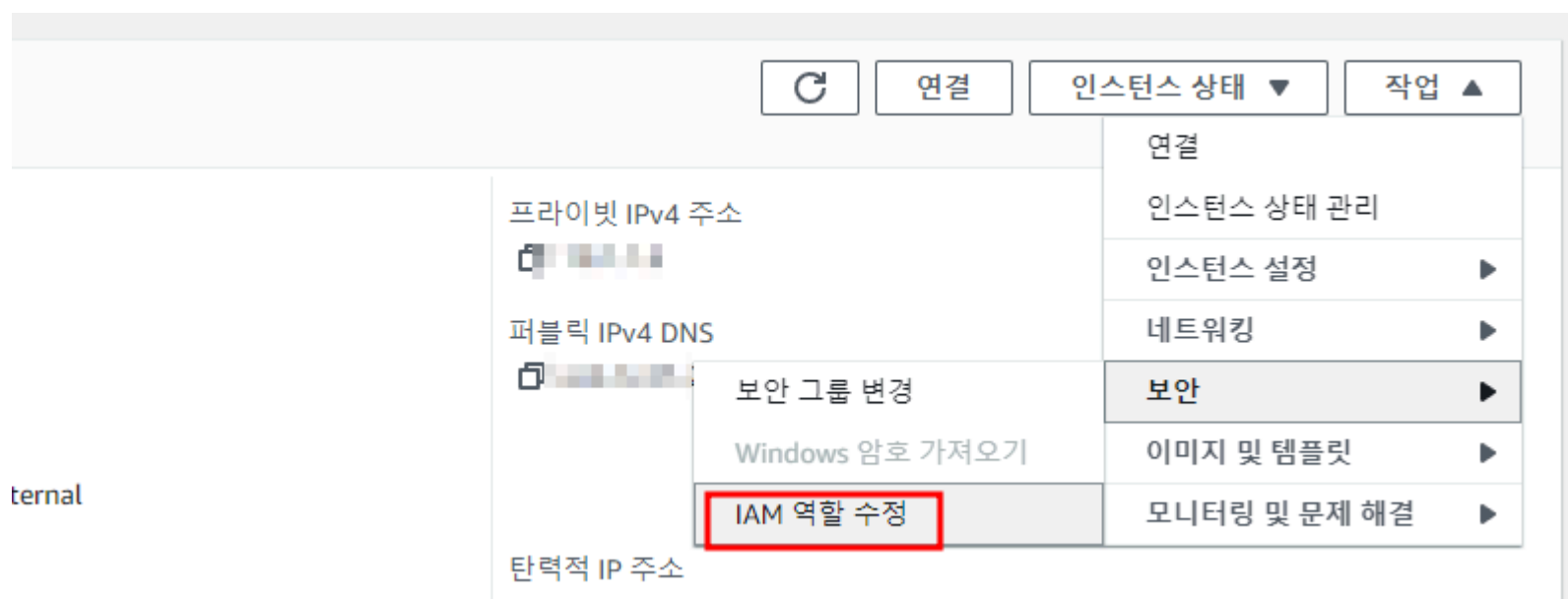
Feb 02 07:00:27 ip-10-1-1-5 systemd[1]: Starting LSB: AWS CodeDeploy Host Agent...
Feb 02 07:00:27 ip-10-1-1-5 codedeploy-agent[3387]: Starting codedeploy-agent:
Feb 02 07:00:27 ip-10-1-1-5 systemd[1]: Started LSB: AWS CodeDeploy Host Agent.
```

4) Java 설치

아래 명령어들로 Java 17버전을 설치하자.

```
sudo apt update
sudo apt install openjdk-17-jdk
```

5) IAM 역할 연결



아까 생성한 S3에 접근하기 위한 IAM 역할을 연결해주자.

★ S3 세팅

Github action의 워크플로우가 수행되면, 코드를 압축해서 S3에 업로드할 것이다.

버킷 만들기 Info

버킷은 S3에 저장되는 데이터의 컨테이너입니다. [자세히 알아보기](#)

일반 구성

버킷 이름

myawsbucket

버킷 이름은 전역에서 고유해야 하며 공백 또는 대문자를 포함할 수 없습니다. [버킷 이름 지정 규칙 참조](#)

AWS 리전

아시아 태평양(서울) ap-northeast-2

기존 버킷에서 설정 복사 - 선택 사항
다음 구성의 버킷 설정만 복사됩니다.

버킷 선택

객체 소유권 Info

다른 AWS 계정에서 이 버킷에 작성한 객체의 소유권 및 액세스 제어 목록(ACL)의 사용을 제어합니다. 객체 소유권은 객체에 대한 액세스를 지정할 수 있는 사용자를 결정합니다.

버킷은 이름만 적당히 지어줄 것이다.
그리고 퍼블릭 액세스를 허용할 것은 없으므로 나머지 설정은 수정할 것 없이 그대로 두면 된다.

★ Code deploy 세팅

1) 애플리케이션 생성

≡

애플리케이션 생성

애플리케이션 구성

애플리케이션 이름
애플리케이션 이름을 입력합니다

cicid-test-CD

100자 이내

컴퓨팅 플랫폼
컴퓨팅 플랫폼 선택

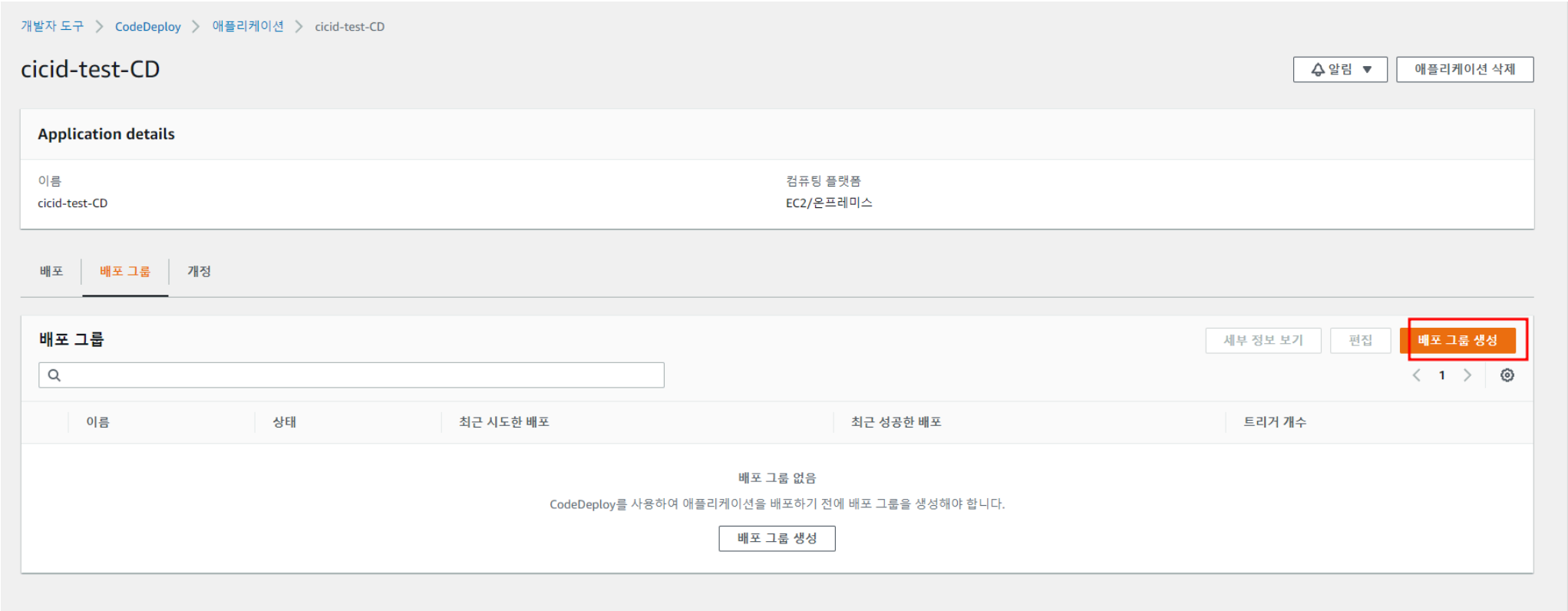
EC2/온프레미스

취소

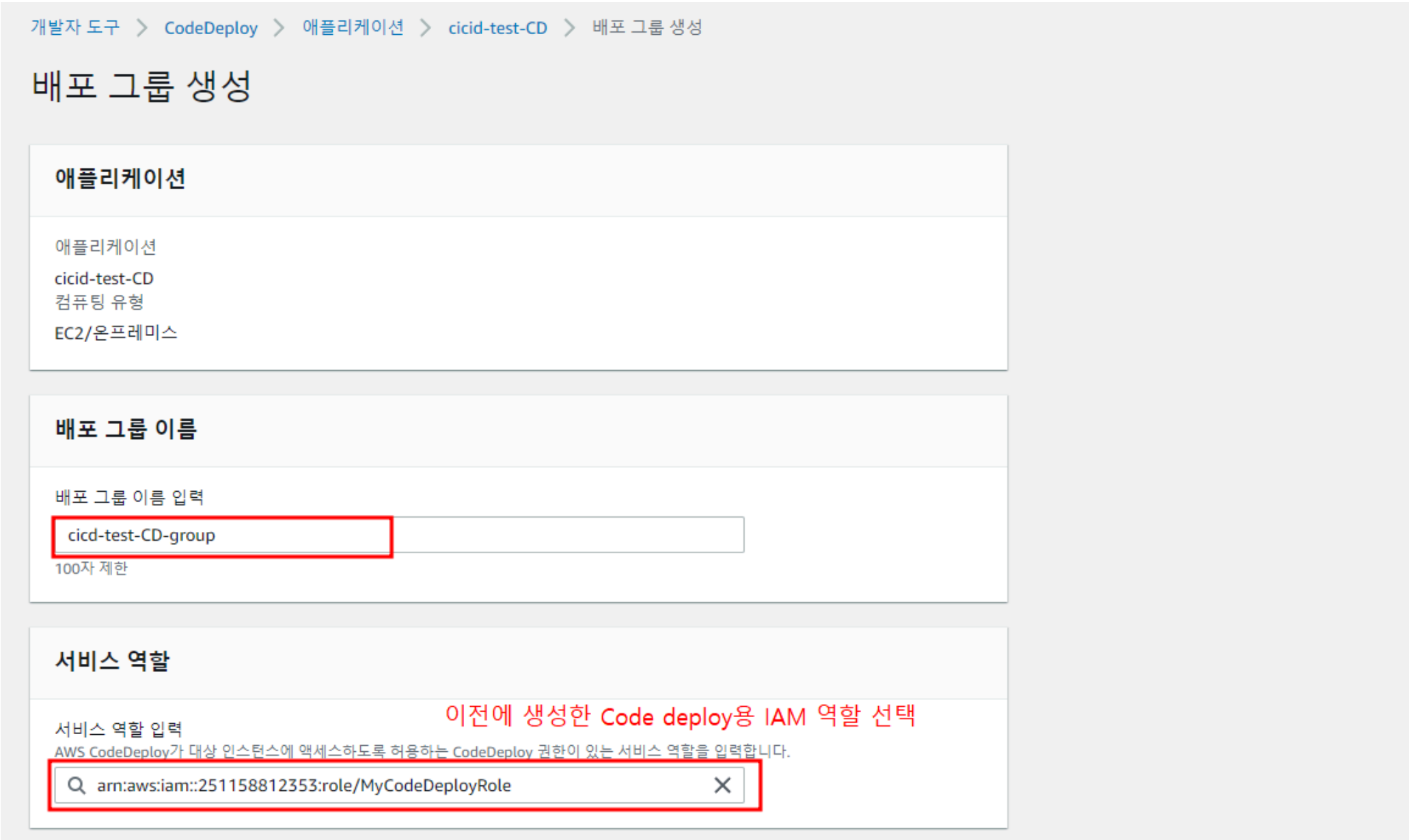
애플리케이션 생성

Code deploy 콘솔로 이동한 후 애플리케이션을 똑딱 만들어주자.

2) 배포 그룹 생성



배포 그룹을 반드시 만들어줘야해서, 생성 버튼을 누르고 쪽 진행하자.



배포 유형

애플리케이션 배포 방법 선택

☒ 현재 위치

배포 그룹의 인스턴스를 최신 애플리케이션 개정으로 업데이트합니다. 배포 중에 각 인스턴스가 업데이트를 위해 잠시 오프라인 상태로 전환됩니다.

☐ 블루/그린

배포 그룹의 인스턴스를 새 인스턴스로 교체하고 최신 애플리케이션 개정을 해당 인스턴스에 배포합니다. 대체 환경의 인스턴스가 로드 밸런서에 등록된 후 원본 환경의 인스턴스는 등록 취소되고 종료할 수 있습니다.

환경 구성

이 배포에 추가할 Amazon EC2 Auto Scaling 그룹, Amazon EC2 인스턴스 및 온프레미스 인스턴스의 조합 선택

☐ Amazon EC2 Auto Scaling 그룹

☒ Amazon EC2 인스턴스

1개의 일치하는 고유한 인스턴스. 자세한 내용을 보려면 여기를 클릭하십시오. [🔗](#)

EC2 인스턴스의 태그 그룹을 세 개까지 이 배포 그룹에 추가할 수 있습니다.

단일 태그 그룹: 해당 태그 그룹이 식별한 인스턴스가 배포됩니다.

다중 태그 그룹: 모든 태그 그룹이 식별한 인스턴스만 배포됩니다.

태그 그룹 1

키

🔍 Name

×

값 - 선택 사항

🔍 cidctest

×

태그 제거

태그 추가

+ 태그 그룹 추가

이전에 생성한 EC2를 선택한다

☐ 온프레미스 인스턴스

AWS Systems Manager를 사용한 에이전트 구성 정보

AWS Systems Manager는 모든 인스턴스에 CodeDeploy 에이전트를 설치하고 구성된 빈도에 따라 업데이트합니다.



AWS Systems Manager가 CodeDeploy 에이전트를 설치하기 전에 필요한 사전 요구 사항을 완료합니다.

AWS Systems Manager Agent가 모든 인스턴스에 설치되어 있는지 확인하고 필요한 IAM 정책을 인스턴스에 연결합니다. [자세히 알아보기](#) [🔗](#)

AWS CodeDeploy 에이전트 설치

☐ 안 함

☒ 한 번만

☐ 지금 업데이트 및 업데이트 일정 예약

배포 설정

배포 구성

기본 및 사용자 지정 배포 구성 목록에서 선택합니다. 배포 구성은 애플리케이션이 배포되는 속도와 배포 성공 또는 실패 조건을 결정하는 규칙 세트입니다.

CodeDeployDefault.AllAtOnce

▼

또는

배포 구성 만들기

로드 밸런서

배포 프로세스 중에 수신 트래픽을 관리할 로드 밸런서를 선택합니다. 로드 밸런서는 배포 중인 각 인스턴스에서 트래픽을 차단하고 배포 성공 후 인스턴스에 대한 트래픽을 다시 허용합니다.

☐ 로드 밸런싱 활성화

▶ 고급 - 선택 사항

취소

배포 그룹 생성

설정은 이렇게 해주면 된다.
중요한 부분은 아까 생성했던 Code deploy를 위한 IAM 역할을 맵핑해주는 것과 태그로 조회해서 EC2를 연결해주는 부분이다.
(참고로 이 설정들은 나중에 수정이 가능하다.)

자! 이렇게 모든 환경 세팅이 완료됐다! 🥳
이제 워크플로우와 필요한 스크립트 파일을 작성해보도록 하겠다.

Part 2. 코드 작성

⚡ GitHub Actions 워크플로우 작성

혹시나 Github Actions이나 워크플로우에 대해서 더 알고 싶다면 이 글이 속한 시리즈의 이전 글들을 참고하자.

```
name: CICD Test
run-name: Running
on:
  push:
    branches:
      - production
      - 'releases/**'

env:
  AWS_REGION: ap-northeast-2
  AWS_S3_BUCKET: app-release-files
  AWS_CODE_DEPLOY_APPLICATION: cicid-test-CD
  AWS_CODE_DEPLOY_GROUP: cicd-test-CD-group

jobs:
  build-with-gradle:
    runs-on: ubuntu-20.04
    steps:
      - name: production 브랜치로 이동
        uses: actions/checkout@v3
        with:
          ref: production
      - name: JDK 17 설치
        uses: actions/setup-java@v3
        with:
          java-version: '17'
          distribution: 'corretto'
      - name: gradlew에 실행 권한 부여
        run: chmod +x ./gradlew
      - name: 프로젝트 빌드
        run: ./gradlew clean build -x test
      - name: AWS credential 설정
        uses: aws-actions/configure-aws-credentials@v1
        with:
          aws-region: ${ env.AWS_REGION }
          aws-access-key-id: ${ secrets.CICD_ACCESS_KEY }
          aws-secret-access-key: ${ secrets.CICD_SECRET_KEY }
      - name: S3에 업로드
        run: aws deploy push --application-name ${ env.AWS_CODE_DEPLOY_APPLICATION } --ignore-hidden-files --s3-location s3://$AWS_S3_BUCKET
      - name: EC2에 배포
        run: aws deploy create-deployment --application-name ${ env.AWS_CODE_DEPLOY_APPLICATION } --deployment-config-name CodeDeployDefaultLinux --s3-location s3://$AWS_S3_BUCKET --deployment-group-name $AWS_CODE_DEPLOY_GROUP
```

나는 요로코롬 작성해주었고, 각 스텝들의 역할은 name 으로 작성해두었으니 참고하자.

⚡ appspec.yml 작성

그리고 프로젝트의 루트 경로에 `appspec.yml` 도 작성해주어야한다.
이 파일은 `Code deploy` 가 수행할 일들을 작성하는 곳이다.

`appspec.yml` 은 공식 매뉴얼을 보고 작성하면 좋을 것 같다.

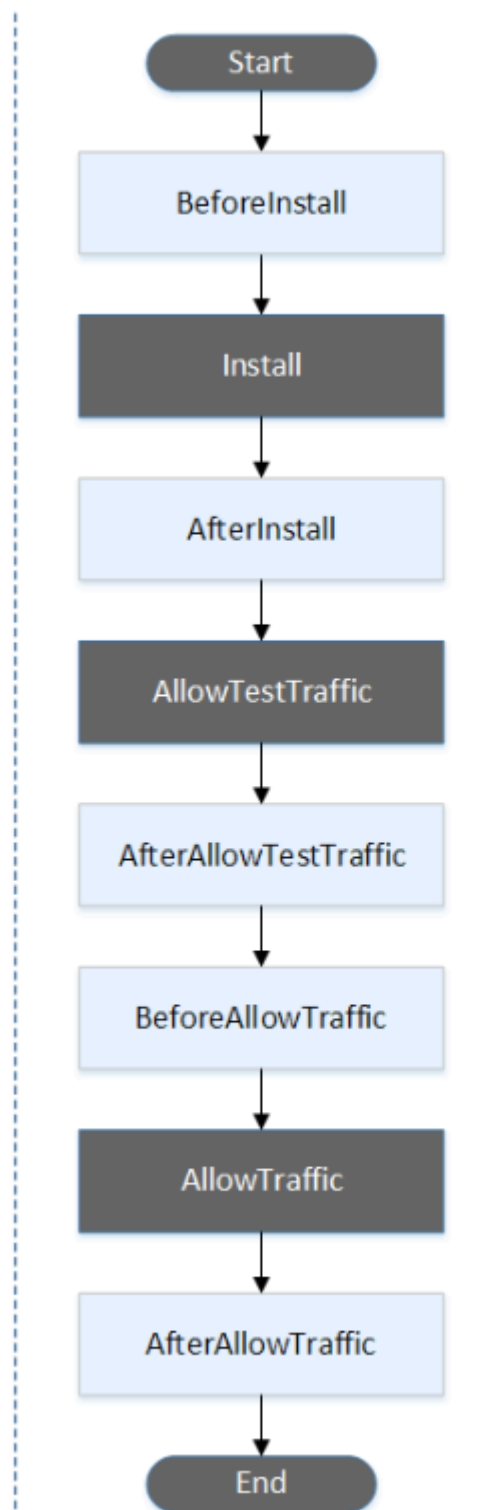
```
version: 0.0
os: linux

files:
  - source: /
    destination: /home/ubuntu/spring-github-action
    overwrite: yes

permissions:
  - object: /
    owner: ubuntu
    group: ubuntu

hooks:
  AfterInstall:
    - location: scripts/stop.sh
      timeout: 60
  ApplicationStart:
    - location: scripts/start.sh
      timeout: 60
```

여기서 중요한 부분은 `hooks` 이다.



Code deploy의 작업은 이런 Lifecycle을 갖고 있는데, 필요한 부분에 `hooks` 를 작성해주면 된다.

'hooks' 섹션의 구조

'hooks' 섹션의 구조는 다음과 같습니다.

```

hooks:
  deployment-lifecycle-event-name:
    - location: script-location
      timeout: timeout-in-seconds
      runas: user-name
  
```

배포 수명 주기 이벤트 이름 뒤의 **hook** 항목에 다음 요소를 포함할 수 있습니다.

location

필수. 수정의 스크립트 파일 번들 위치 `hooks` 섹션에서 지정한 스크립트 위치는 애플리케이션 수정 번들의 루트를 기준으로 합니다. 자세한 정보는 [CodeDeploy의 개정 계획](#)을 참조하십시오.

timeout

선택 사항. 스크립트가 실패로 간주되기 전에 실행할 수 있는 기간(초). 기본값은 3600초(1시간)입니다.

참고

3600초(1시간)은 각 배포 수명 주기 이벤트에 대한 스크립트 실행에 허용되는 최대 시간입니다. 스크립트가 이 한도를 초과하면 배포가 중지되고 인스턴스에 대한 배포가 실패합니다. 각 배포 수명 주기 이벤트의 모든 스크립트에 대해 **timeout**에 지정된 총 시간(초)이 이 한도를 초과하지 않아야 합니다.

runas

선택 사항. 스크립트 실행 시 가장하는 사용자. 기본적으로 인스턴스에서 실행 중인 CodeDeploy 에이전트입니다. CodeDeploy는 암호를 저장하지 않기 때문에 **runas** 사용자에게 암호가 필요합니다. 이 요소는 Amazon Linux, Ubuntu Server 및 RHEL 인스턴스에만 적용됩니다.

hooks에 작성할 요소들은 이런 것들이 있다.

⚡ stop.sh 작성

```
#!/bin/bash

ROOT_PATH="/home/ubuntu/spring-github-action"
JAR="$ROOT_PATH/application.jar"
STOP_LOG="$ROOT_PATH/stop.log"
SERVICE_PID=$(pgrep -f $JAR) # 실행중인 Spring 서버의 PID

if [ -z "$SERVICE_PID" ]; then
    echo "서비스 NouFound" >> $STOP_LOG
else
    echo "서비스 종료 " >> $STOP_LOG
    kill "$SERVICE_PID"
    # kill -9 $SERVICE_PID # 강제 종료를 하고 싶다면 이 명령어 사용
fi
```

이 스크립트는 실행되고 있는 Spring 서버를 종료하는 역할을 한다.

⚡ start.sh 작성

```
#!/bin/bash

ROOT_PATH="/home/ubuntu/spring-github-action"
JAR="$ROOT_PATH/application.jar"

APP_LOG="$ROOT_PATH/application.log"
ERROR_LOG="$ROOT_PATH/error.log"
START_LOG="$ROOT_PATH/start.log"

NOW=$(date +%c)

echo "[$NOW] $JAR 복사" >> $START_LOG
cp $ROOT_PATH/build/libs/spring-github-action-1.0.0.jar $JAR

echo "[$NOW] > $JAR 실행" >> $START_LOG
nohup java -jar $JAR > $APP_LOG 2> $ERROR_LOG &

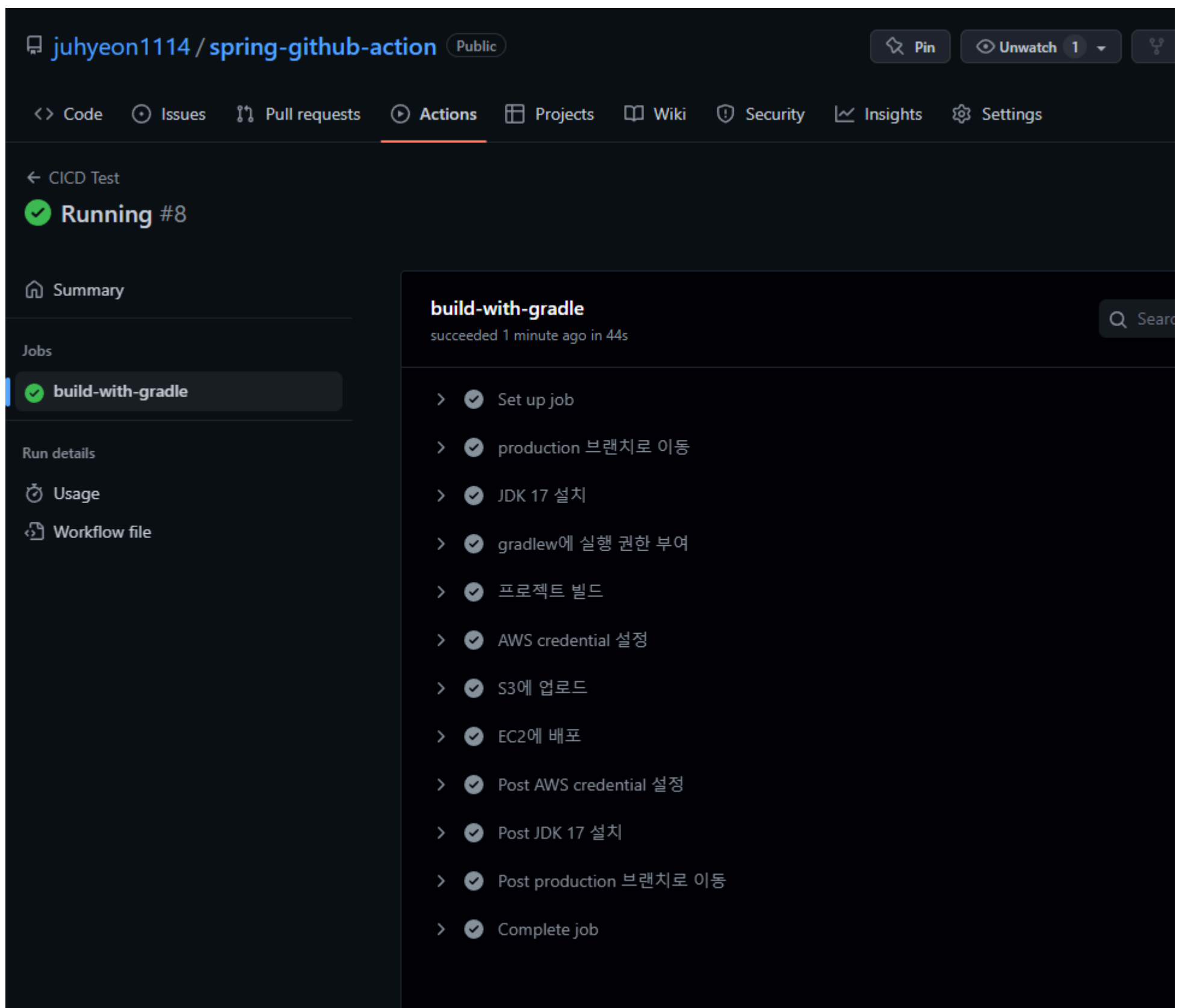
SERVICE_PID=$(pgrep -f $JAR)
echo "[$NOW] > 서비스 PID: $SERVICE_PID" >> $START_LOG
```

Part 3. 배포

🔥 실행!

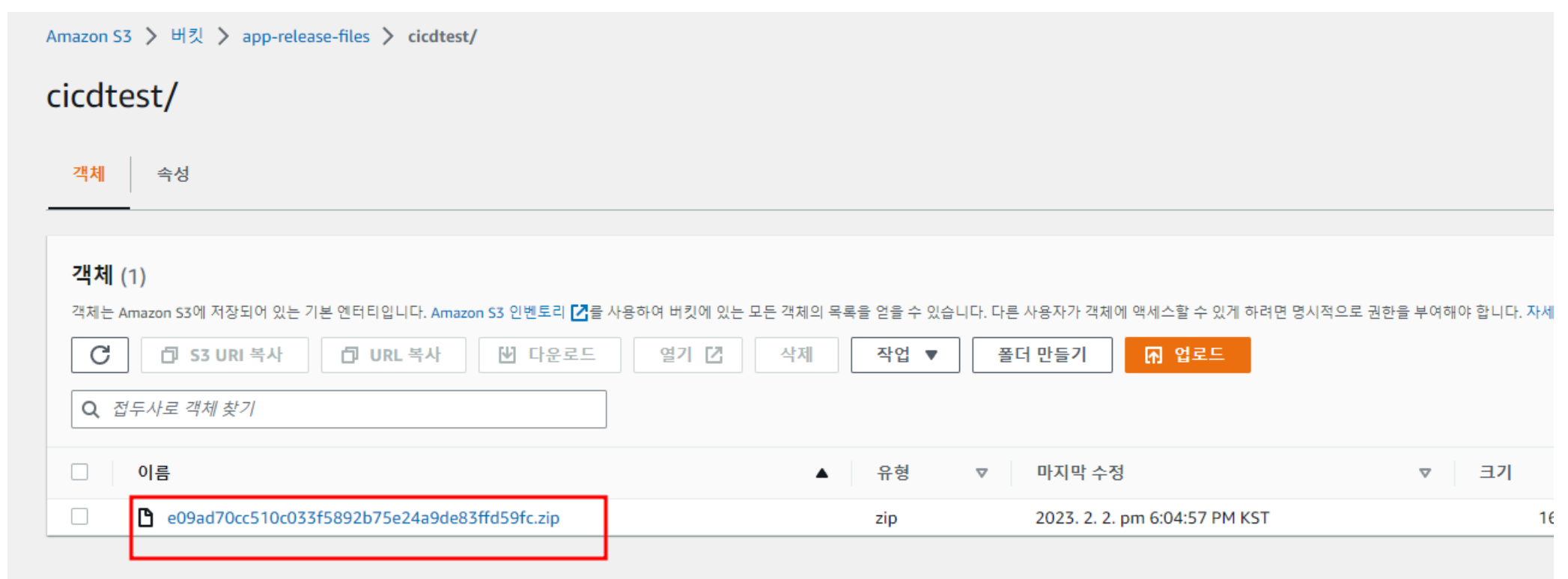
자 드디어 실행할 시간이다 ㅎㅎ
production 브랜치에 코드 push를 발생시키고, 잘 진행되는 지 확인해보자.

1) 워크플로우 확인



음 ~~ 워크플로우가 쌈@뽕하게 완료됐다.

2) S3 확인



S3에도 압축된 코드 파일이 잘 올라와 있다.
이제 EC2에서 빌드만 잘 되면 된다!!

3) Code deploy 확인

개발자 도구 > CodeDeploy > 배포 > d-ENFQR8KC9

d-ENFQR8KC9

↺ 배포 복사

❌ The overall deployment failed because too many individual instances failed deployment, too few healthy instances are available for deployment, or some instances in your deployment group are experiencing p

배포 상태

인스턴스에 애플리케이션 설치 중

1%

0/1개의 인스턴스가 업데이트됨 ❌ 실패

배포 세부 정보

애플리케이션	배포 ID	상태
cicid-test-CD	d-ENFQR8KC9	❌ 실패
배포 구성	배포 그룹	시작
CodeDeployDefault.AllAtOnce	cicid-test-CD-group	사용자 작업
배포 설명		
-		

오! 맙소사...
첫 배포가 실패했다... 원인을 파악해보자 🤔

3-1) 이슈1: 자격증명 에러

먼저 찾아보니 여기에서 Code deploy의 로그를 보는 방법을 안내해주고 있었다.
방법대로 찾아보니...

```
/opt/codedeploy-agent/vendor/gems/process_manager-0.0.13/lib/process_manager/master.rb:36:in `start'
/opt/codedeploy-agent/bin/./lib/codedeploy-agent.rb:43:in `block (2 levels) in <main>'
/opt/codedeploy-agent/vendor/gems/gli-2.11.0/lib/gli/command_support.rb:126:in `execute'
/opt/codedeploy-agent/vendor/gems/gli-2.11.0/lib/gli/app_support.rb:284:in `block in call_command'
/opt/codedeploy-agent/vendor/gems/gli-2.11.0/lib/gli/app_support.rb:297:in `call_command'
/opt/codedeploy-agent/vendor/gems/gli-2.11.0/lib/gli/app_support.rb:79:in `run'
/opt/codedeploy-agent/bin/./lib/codedeploy-agent.rb:90:in `<main>'
2023-02-02 09:14:27 ERROR [codedeploy-agent(3395)]: InstanceAgent::Plugins::CodeDeployPlugin::CommandPoller: Missing credentials - please check if this instance was started with an IAM instance profile
(END)
```

ERROR [codedeploy-agent(3395)]: InstanceAgent::Plugins::CodeDeployPlugin::CommandPoller: Missing credentials - please check if this instance was started with an IAM instance profile

◀ ▶

이런 에러 로그가 찍혀있는 것을 확인할 수 있었다.

공식문서를 확인해보니 자격 증명하는 부분에 문제가 있는 것을 확인했다.

리서치를 해보니, EC2에 IAM을 적용한 게 Code deploy agent입장에서 제대로 반영이 안됐을 수도 있다고 한다.
그렇다면, EC2를 재부팅해주자 🙄

3-2) 이슈2: Permission denied 에러

이벤트 로그

이벤트 세부 정보

오류 코드

❌ ScriptFailed

스크립트 이름

scripts/stop.sh

메시지

Script at specified location: scripts/stop.sh run as user ubuntu failed with exit code 1

로그

LifecycleEvent - AfterInstall

Script - scripts/stop.sh

[stderr]/opt/codedeploy-agent/deployment-root/f5d95695-a2a5-4155-9592-7f1ba83c533b/d-4VTMUJRML/deployment-archive/scripts/stop.sh: line 9: LOG_FILE: Permission denied

그리고 다시 실행해봤는데, 이번에는 script를 실행하는 부분에서 Permission denied 에러가 발생했다.

```
ubuntu@ip-10-1-1-5:/opt$ ls -al
total 12
drwxr-xr-x  3 root root  4096 Feb  2 07:00 .
drwxr-xr-x 19 root root  4096 Feb  2 09:32 ..
drwxr-xr-x  8 2450 users 4096 Feb  2 09:24 codedeploy-agent
ubuntu@ip-10-1-1-5:/opt$
```

찾아보니 code deploy agent의 사용자, 그룹명이 appspec.yml 에 작성한 것과 달라서 발생한 문제였다. 문제를 해결하는 방법은 AWS 공식문서에서 알려주고 있다.

아래의 명령어들을 실행하면 된다.

Agent 소유자 변경

```
sudo service codedeploy-agent stop
sudo sed -i 's/"ubuntu"/g' /etc/init.d/codedeploy-agent
sudo systemctl daemon-reload
sudo chown ubuntu:ubuntu -R /opt/codedeploy-agent/
sudo chown ubuntu:ubuntu -R /var/log/aws/
```

Agent 재실행, 상태 확인

```
sudo service codedeploy-agent start
sudo service codedeploy-agent status
```

사실 이 외에도 수 많은 난관이 있었는데... 다 나의 실수 때문이어서 글에 작성하지는 않았다 😞🙄

4) 이제 진짜 확인 🔥

위 이슈들을 모두 해결하고 다시 확인해보니 드.디.어! 정상적으로 배포된 것을 확인할 수 있었다!!! ㅎㅎㅎ 👍

4-1) Deploy 성공

개발자 도구 > CodeDeploy > 배포

배포 내역

🔄

세부 정보 보기

등작 ▼

배포 복사

배포 재시도

🔍

< 1 > ⚙️

	배포 ID	상태	배포 유형	컴퓨팅 플랫폼	애플리케이션	배포 그룹	개정 위치	이벤트 시작	시작 시간	종료 시간
<input type="radio"/>	d-2C3F9FSML	✅ 성공	현재 위치	EC2/온프레미스	cicid-test-CD	cicd-test-CD-group	s3://app-rele...	사용자 작업	2월 2, 2023 7:40 오후 (UTC+9:00)	2월 2, 2023 7:40 오후 (UTC+9:00)
<input type="radio"/>	d-N0Q0TFSML	❌ 실패	현재 위치	EC2/온프레미스	cicid-test-CD	cicd-test-CD-group	s3://app-rele...	사용자 작업	2월 2, 2023 7:33 오후 (UTC+9:00)	2월 2, 2023 7:33 오후 (UTC+9:00)
<input type="radio"/>	d-FX9F3ORML	❌ 실패	현재 위치	EC2/온프레미스	cicid-test-CD	cicd-test-CD-group	s3://app-rele...	사용자 작업	2월 2, 2023 7:29 오후 (UTC+9:00)	2월 2, 2023 7:29 오후 (UTC+9:00)
<input type="radio"/>	d-5NIWJ8SML	❌ 실패	현재 위치	EC2/온프레미스	cicid-test-CD	cicd-test-CD-group	s3://app-rele...	사용자 작업	2월 2, 2023 7:21 오후 (UTC+9:00)	2월 2, 2023 7:21 오후 (UTC+9:00)
<input type="radio"/>	d-GDFCN7SML	❌ 실패	현재 위치	EC2/온프레미스	cicid-test-CD	cicd-test-CD-group	s3://app-rele...	사용자 작업	2월 2, 2023 7:14 오후 (UTC+9:00)	2월 2, 2023 7:14 오후 (UTC+9:00)
<input type="radio"/>	d-4VTMUJRML	❌ 실패	현재 위치	EC2/온프레미스	cicid-test-CD	cicd-test-CD-group	s3://app-rele...	사용자 작업	2월 2, 2023 7:00 오후 (UTC+9:00)	2월 2, 2023 7:00 오후 (UTC+9:00)
<input type="radio"/>	d-JCGKRYQML	❌ 실패	현재 위치	EC2/온프레미스	cicid-test-CD	cicd-test-CD-group	s3://app-rele...	사용자 작업	2월 2, 2023 6:51 오후 (UTC+9:00)	2월 2, 2023 6:51 오후 (UTC+9:00)
<input type="radio"/>	d-M8P38BRML	❌ 실패	현재 위치	EC2/온프레미스	cicid-test-CD	cicd-test-CD-group	s3://app-rele...	사용자 작업	2월 2, 2023 6:43 오후 (UTC+9:00)	2월 2, 2023 6:48 오후 (UTC+9:00)
<input type="radio"/>	d-ONG32TQML	❌ 실패	현재 위치	EC2/온프레미스	cicid-test-CD	cicd-test-CD-group	s3://app-rele...	사용자 작업	2월 2, 2023 6:40 오후 (UTC+9:00)	2월 2, 2023 6:41 오후 (UTC+9:00)

4-2) 목표한 jar파일 생성

```
ubuntu@ip-10-1-1-5:~/spring-github-action$ ls
application.jar application.log appspec.yml build build.gradle error.log gradle gradlew
ubuntu@ip-10-1-1-5:~/spring-github-action$
```

4-3) 서비스 구동 확인

```
ps -ef | grep java
```

```
ubuntu@ip-10-1-1-5:~$ ps -ef | grep java
ubuntu      8877      1  34 10:47 ?        00:00:05 java -jar /home/ubuntu/spring-github-action/application.jar
ubuntu      9352     9341  0 10:47 pts/0    00:00:00 grep --color=auto java
ubuntu@ip-10-1-1-5:~$ ^C
ubuntu@ip-10-1-1-5:~$
```

4-4) 접속 확인

The image shows a terminal window titled 'EC2 Management Console' with the IP address '3.37.2.45:8080'. The terminal output shows 'Hello world'.