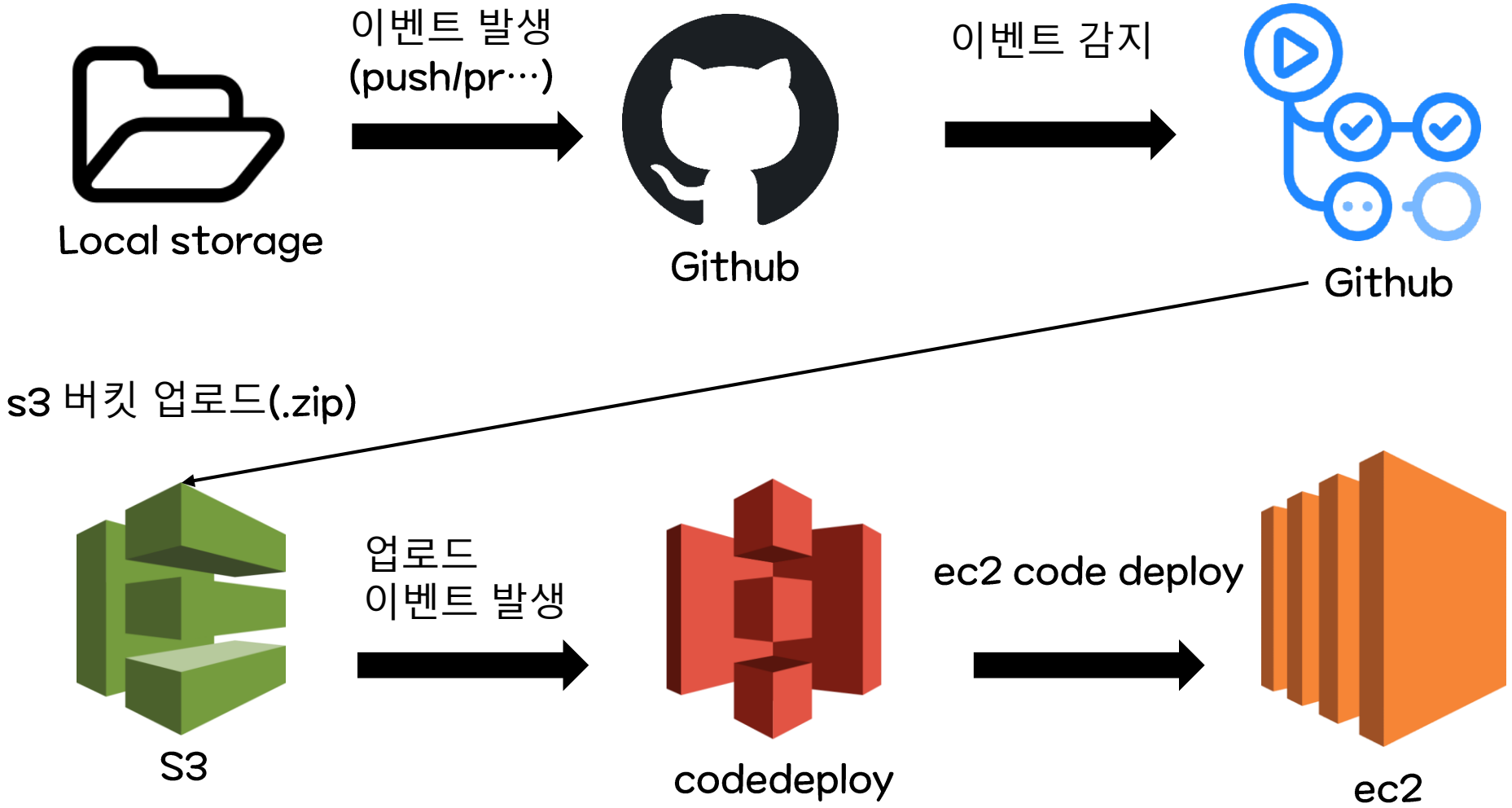
CI-CD

ssosso.table

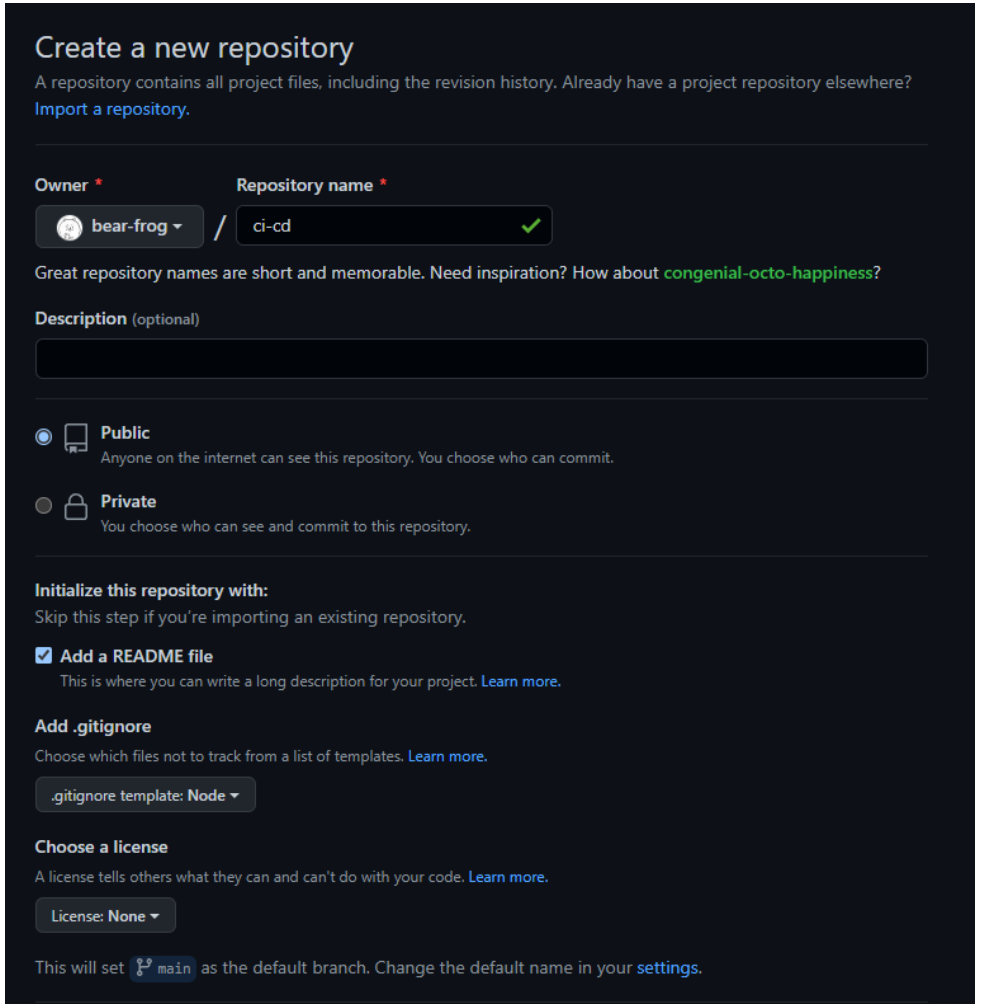
1. introduction

- github action을 이용한 배포 자동화 구현
- 사용 도구
  - vue.js quasar
    - client app
    - SSR 예제 프로젝트 활용
  - aws
    - ec2: client app 배포 서버
    - s3: client app 저장 버킷
    - codedeploy: s3 -> ec2 배포 자동화
  - github action
    - local -> s3 배포 자동화
- 참조
  - https://ms3864.tistory.com/383?category=1003779

1. introduction: flowchart



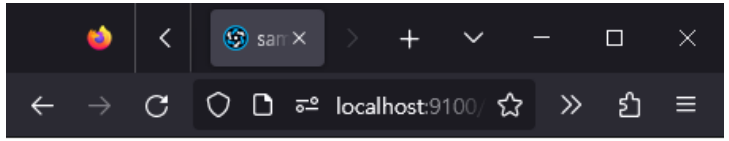
2. create repository



2. create repository: upload vue project

1. team-study의 SSR 프로젝트 파일을 가져옵니다
2. 프로젝트 빌드(npm run build)
3. 로컬 환경 프로젝트 정상 동작 확인

(npm run dev or npm start)



안녕하세요
김고구마 님!

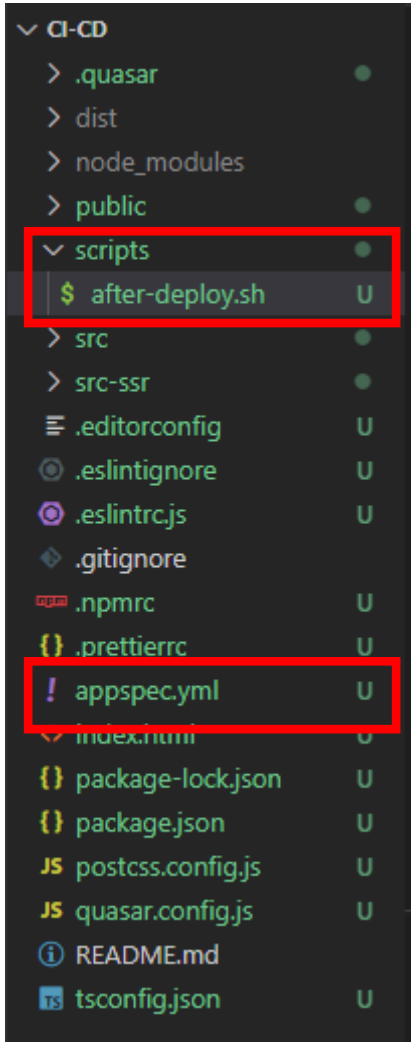
지금 시간은
"2023-03-15T08:16:25.916Z" 입니다

3. create repository: add deploy script

- appspec.yml
- scripts/after-deploy.sh
- 두 파일을 추가합니다

- codedeploy에 의해

서버에 배포된 이후를 정의합니다



3. create repository: appspec.yml

```yaml
version: 0.0
os: linux
files:
    - source: /
    destination: /home/ubuntu/app
    overwrite: yes
permissions:
    - object: /home/ubuntu
    pattern: '**'
    owner: ubuntu
    group: ubuntu
hooks:
    AfterInstall:
        - location: scripts/after-deploy.sh
        timeout: 180
        runas: ubuntu
```

참조
https://github.com/bear-frog/ci-cd/blob/main/appspec.yml

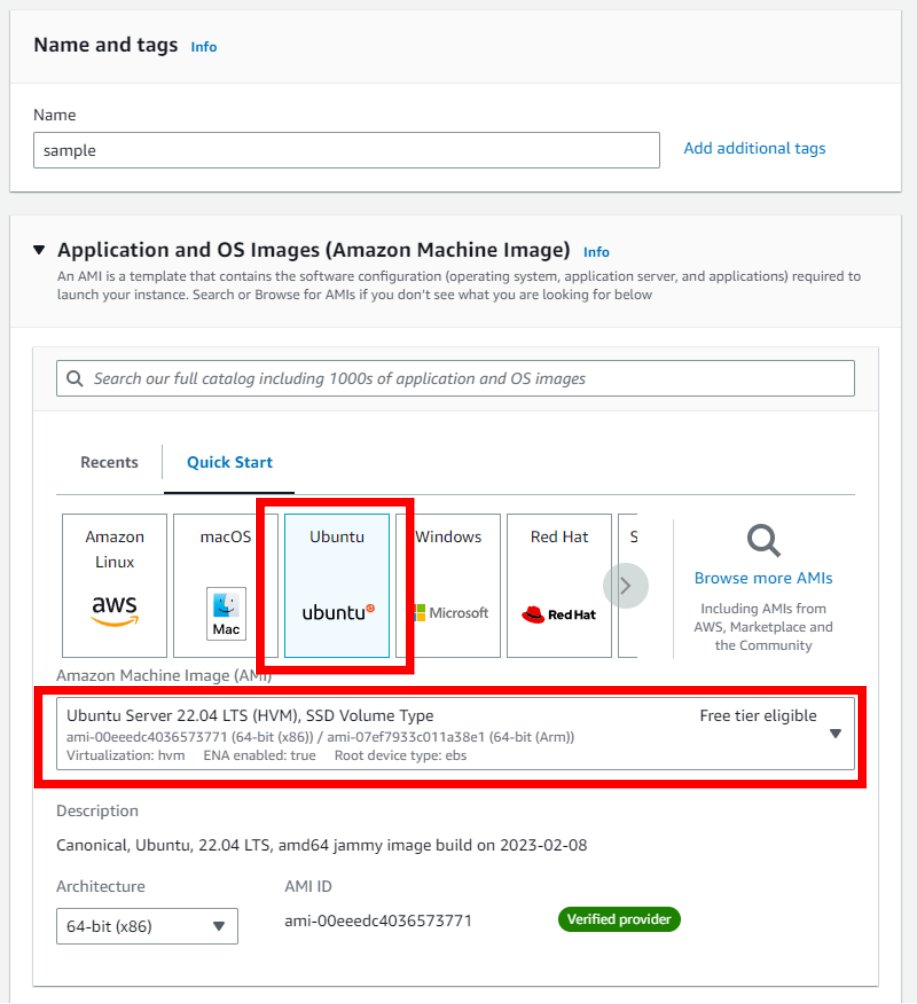
3. create repository: scripts/after-deploy.sh

```
#!/bin/bash

# 어플리케이션 디렉터리
REPOSITORY=/home/ubuntu/app/dist/ssr


# 배포 디렉터리로 이동
cd $REPOSITORY

# 종속성 설치
sudo npm i

# MARK: pm2를 사용해 80번 포트에 앱 실행
# 참조
# https://unchae.tistory.com/entry/PM2-80-443%ED%8F%AC%ED%8A%B8-%EC%82%AC%EC%9A%A9
authbind --deep pm2 reload index.js --watch
```

참조

https://github.com/bear-frog/ci-cd/blob/main/scripts/after-deploy.sh

4. create ec2 instance



동일화를 위해 aws Ubuntu 22.04를 사용하세요

4. setup ec2 instance: install node

1. install latest node.js in Ubuntu

참조: https://github.com/nodesource/distributions

Installation instructions

Node.js v19.x:

Using Ubuntu

```
curl -fsSL https://deb.nodesource.com/setup_19.x | sudo -E bash - &&\
sudo apt-get install -y nodejs
```

```
ubuntu@ip-172-31-10-195:~$ node -v
v19.7.0
```

4. setup ec2 instance: install codedeploy

1. 우분투에 codedeploy를 설치합니다

$ sudo apt update
$ sudo apt install ruby-full
$ sudo apt install wget
$ cd /home/Ubuntu

 # 복붙이 아니고!! 사용중인 ec2 region을 확인하고 bucket-name과 region-identifier를 작성
 # e.g us-east-2 〉 wget https://aws-codedeploy-us-east-2.s3.us-east-2.amazonaws.com/latest/install
$ wget https://bucket-name.s3.region-identifier.amazonaws.com/latest/install
$ chmod +x ./install
$ sudo ./install auto 〉 /tmp/logfile

# 실행중인지 확인
$ sudo service codedeploy-agent status

```
● codedeploy-agent.service - LSB: AWS CodeDeploy Host Agent
   Loaded: loaded (/etc/init.d/codedeploy-agent; generated)
   Active: active (running) since Wed 2023-03-15 09:15:43 UTC; 8s ago
```

CodeDeploy 에이전트가 설치되어 실행 중이면 다음과 같은 메시지가 표시되어야 합니다.

The AWS CodeDeploy agent is running.

# error: No AWS CodeDeploy agent running과 같은 메시지가 표시되면 서비스를 시작하고 다음 두 명령을 한 번에 하나씩 실행
$ sudo service codedeploy-agent start
$ sudo service codedeploy-agent status

참조:

https://docs.aws.amazon.com/ko_kr/codedeploy/latest/userguide/codedeploy-agent-operations-install-ubuntu.html

4. setup ec2 instance: setup app

1. ec2 Ubuntu 의 홈 디렉터리에 app 폴더를 생성하세요
   - 해당 디렉터리에 코드가 배포돼요

```
ubuntu@ip-172-31-0-255:~$ mkdir /home/ubuntu/app
ubuntu@ip-172-31-0-255:~$ cd /home/ubuntu/
ubuntu@ip-172-31-0-255:~$ ls
app
```

2. sudo npm install pm2 –g 명령어를 통해 pm2를 설치하세요

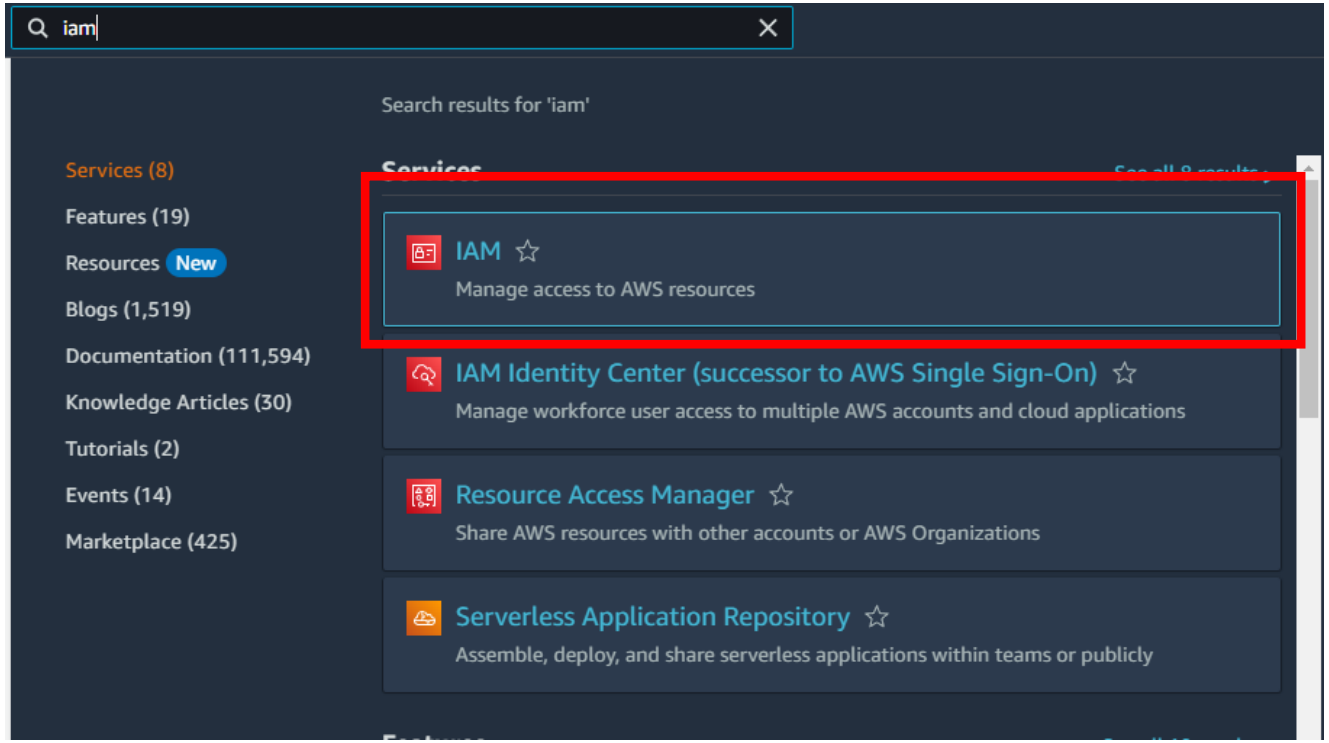
3. 80번 포트에서 node 앱이 실행되기 위한 설정을 하세요

# 참조: https://unchae.tistory.com/entry/PM2-80-443%ED%8F%AC%ED%8A%B8-%EC%82%AC%EC%9A%A9
$ sudo apt-get install authbind
$ sudo touch /etc/authbind/byport/80
$ sudo chown ubuntu /etc/authbind/byport/80
$ sudo chmod 755 /etc/authbind/byport/80
$ sudo touch /etc/authbind/byport/443
$ sudo chown ubuntu /etc/authbind/byport/443
$ sudo chmod 755 /etc/authbind/byport/443

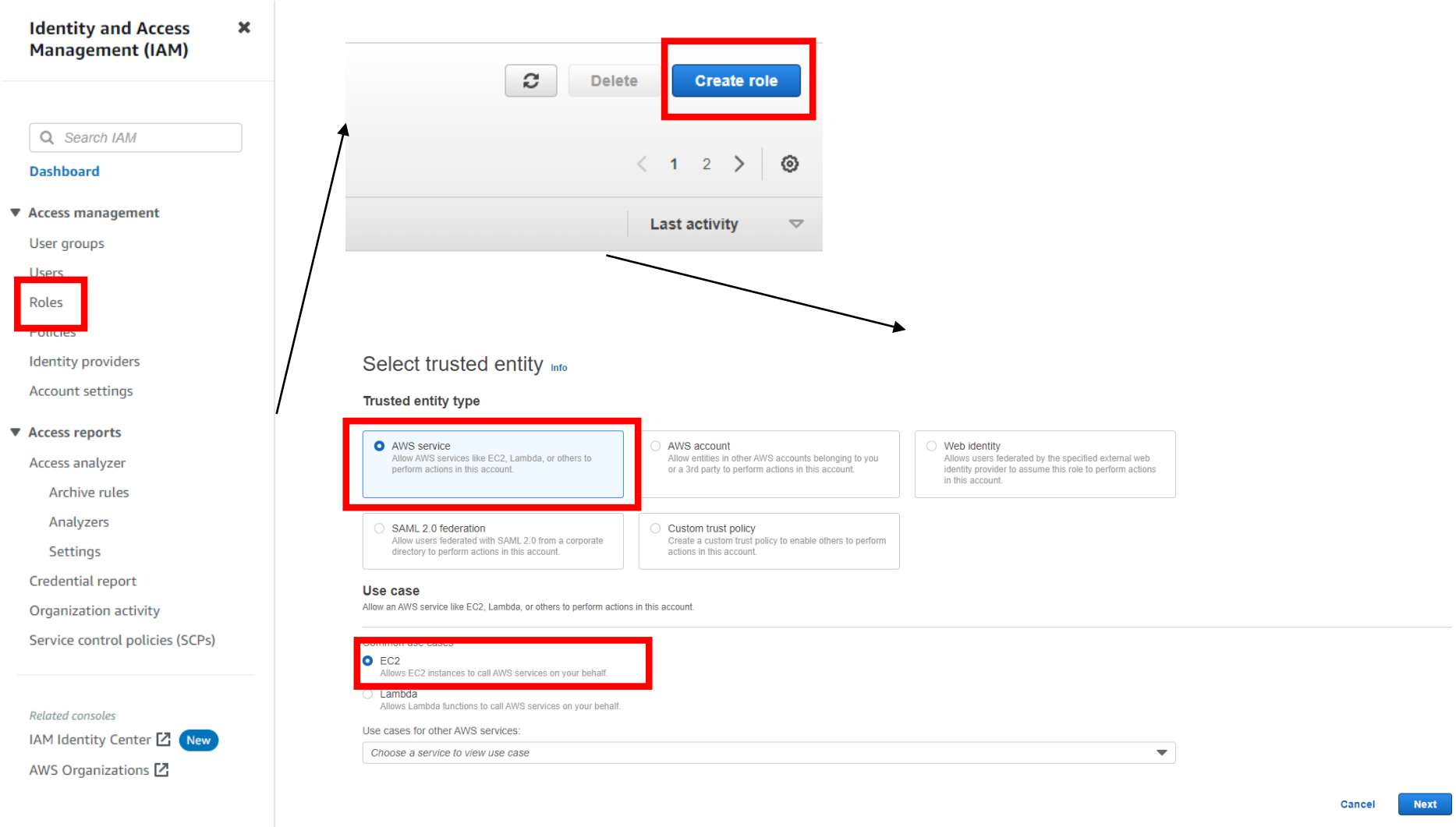
실행:
$ authbind --deep pm2 start index.js

5. create IAM role

ec2가 s3와 codedeploy를 이용할 수 있도록 권한 설정



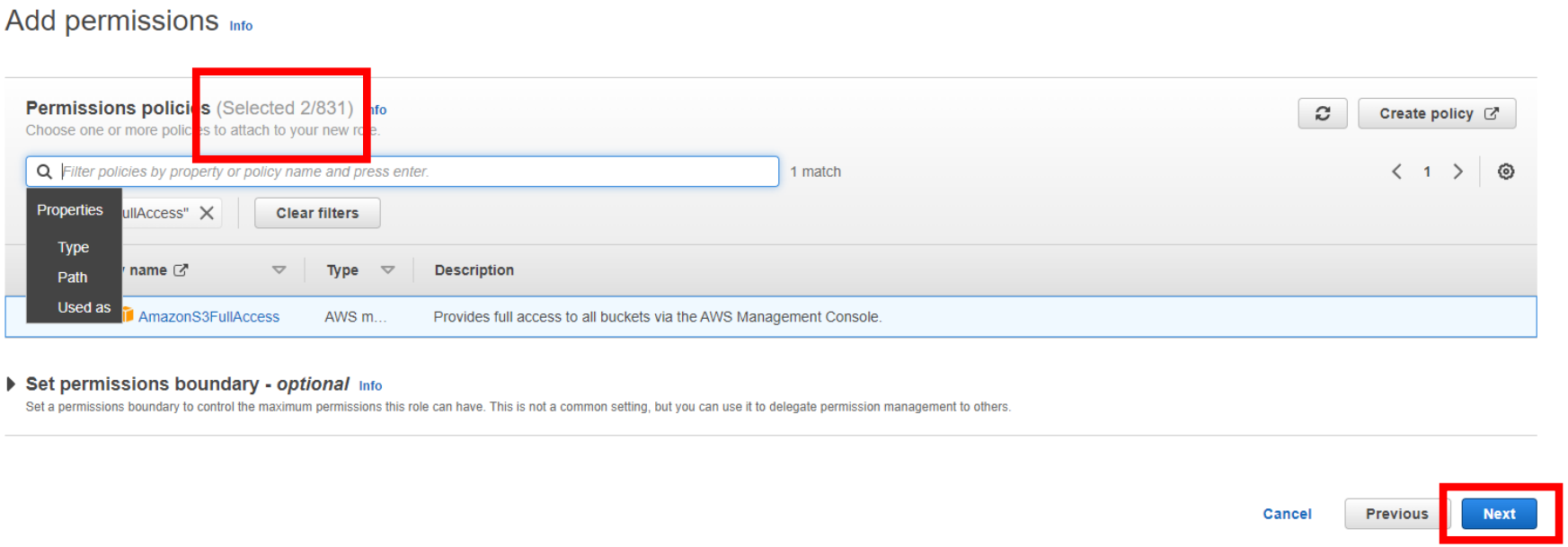
5. create IAM role: create ec2 role



5. create IAM role: create ec2 role

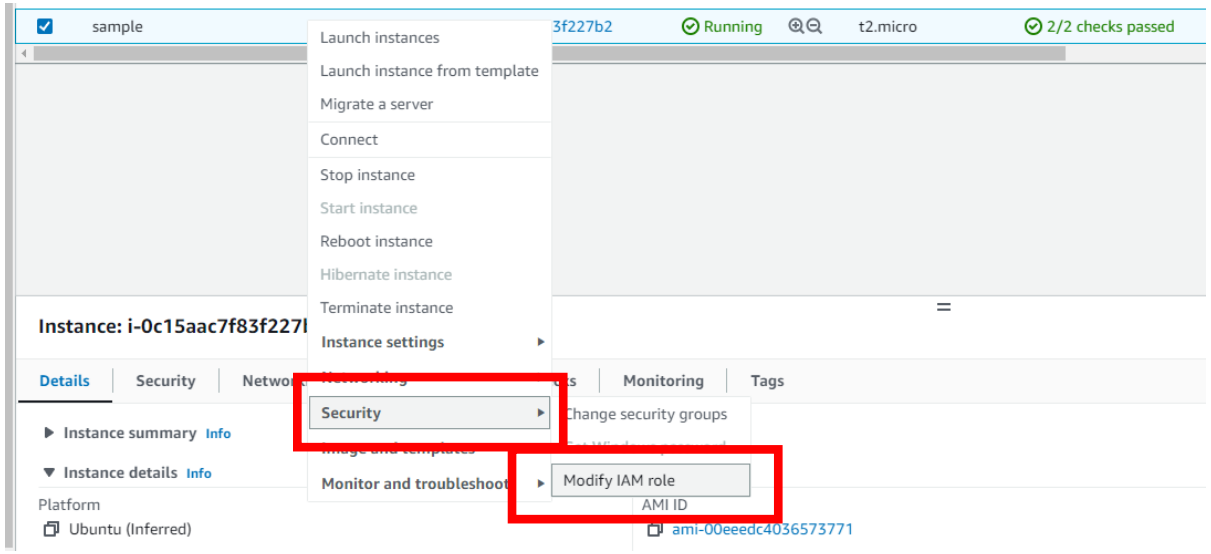
1. AWSCodeDeployFullAccess
2. AmazonS3FullAccess

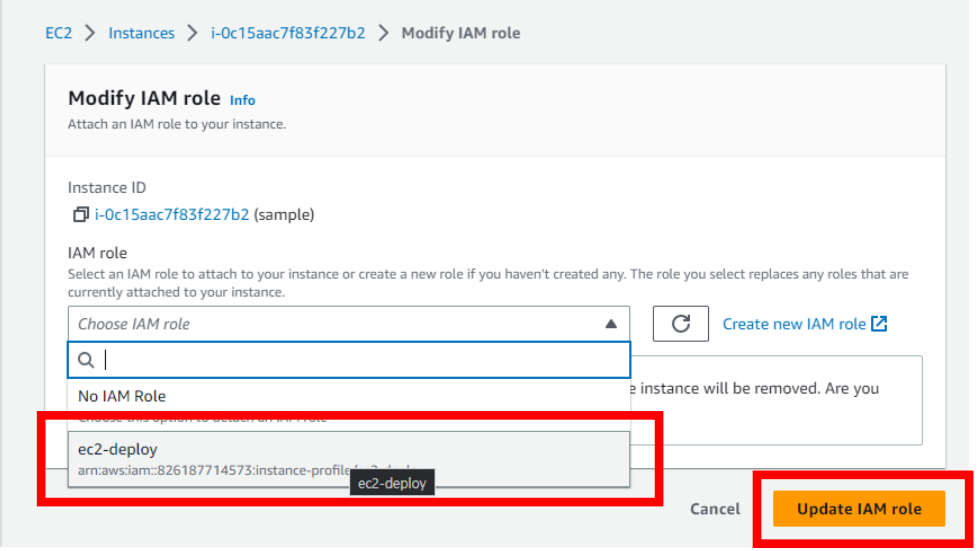
두 개의 권한 추가 / 생성



5. create IAM role: add ec2 role

생성한 ec2 instance에 IAM role 추가





5. create IAM role: create codedeploy role

codedeploy를 위한 권한을 추가합니다

Select trusted entity Info

Trusted entity type

- AWS service
  Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account
  Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity
  Allows users feder identity provider to in this account.
- SAML 2.0 federation
  Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy
  Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Common use cases

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

Use cases for other AWS services:

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.
- CodeDeploy - ECS
  Allows CodeDeploy to read S3 objects, invoke Lambda functions, publish to SNS topics, and update ECS services on your behalf.

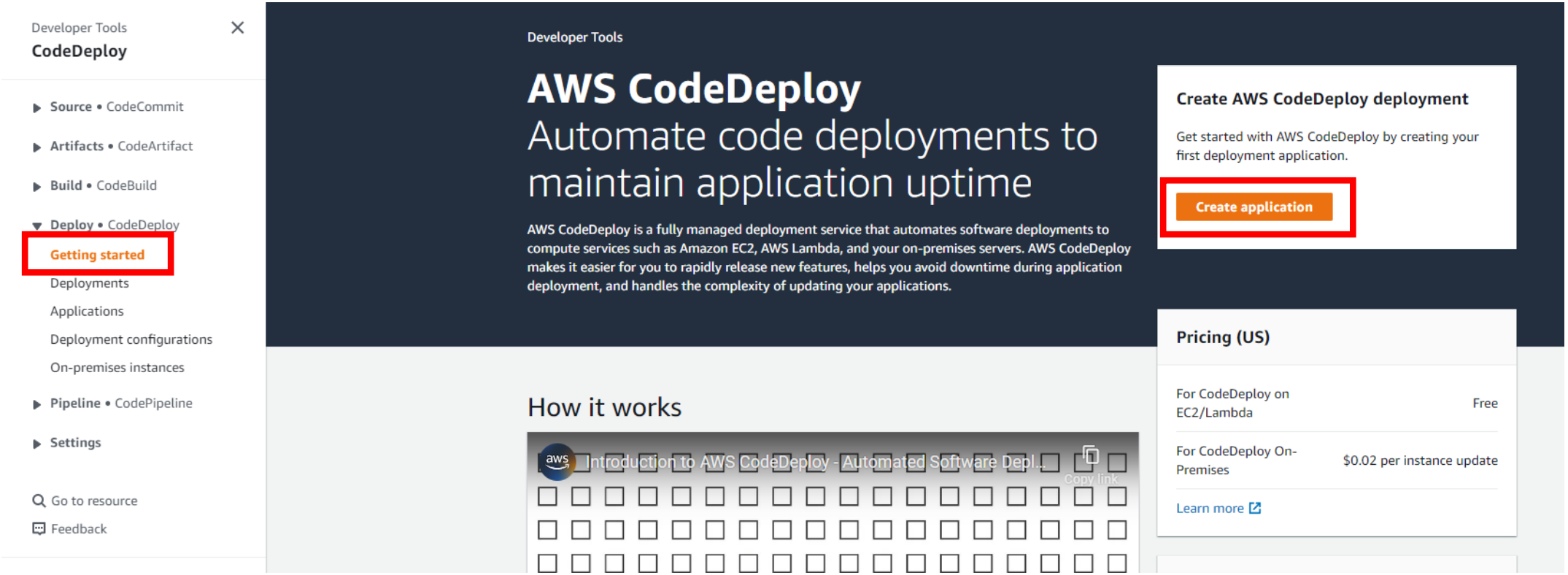
Permissions policies (1) Info

The type of role that you selected requires the following policy.

Policy name	Type	Attached entities
AWSCodeDeployRole	AWS m...	0

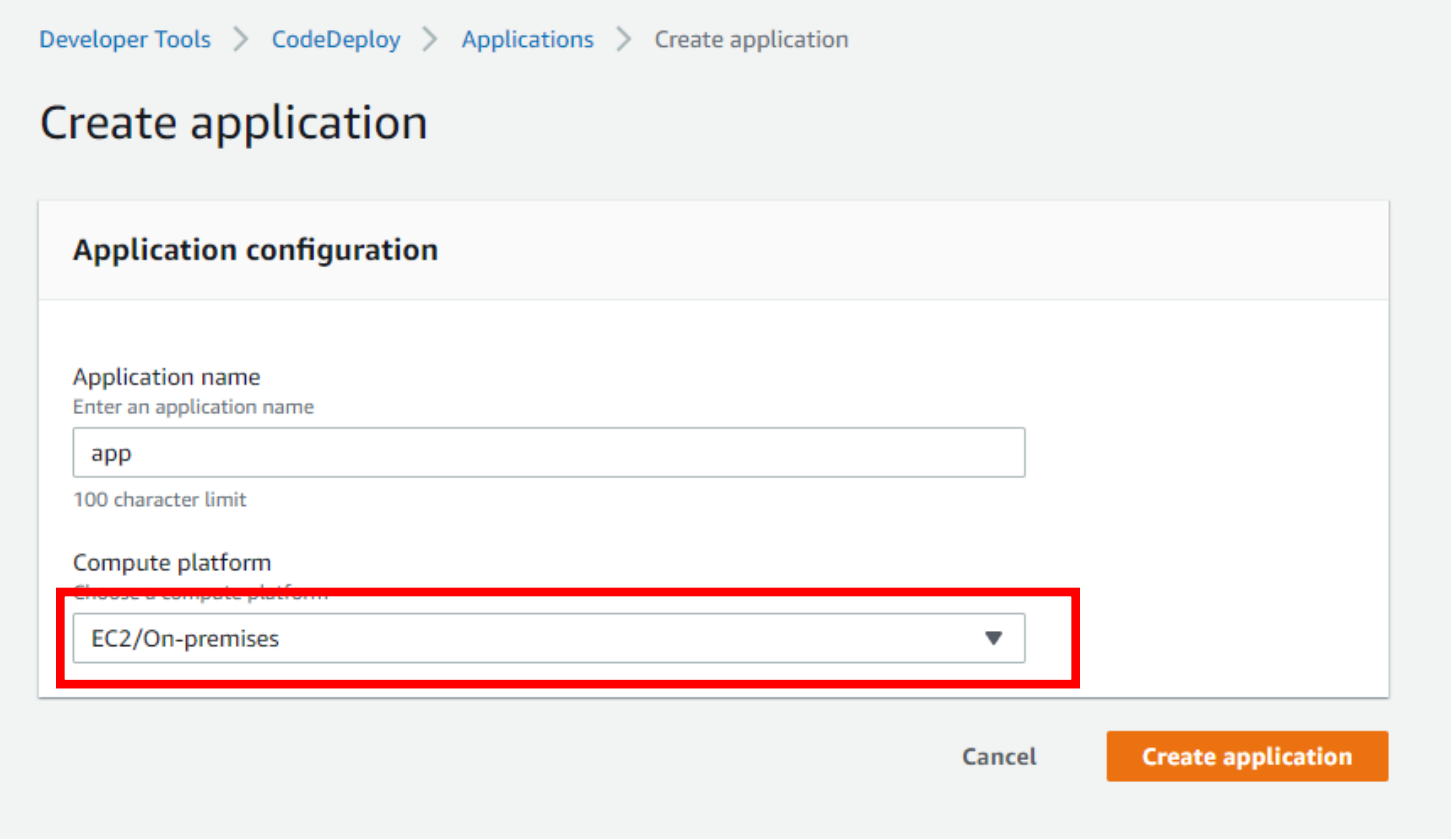
권한 추가 / 생성

6. create Codedeploy



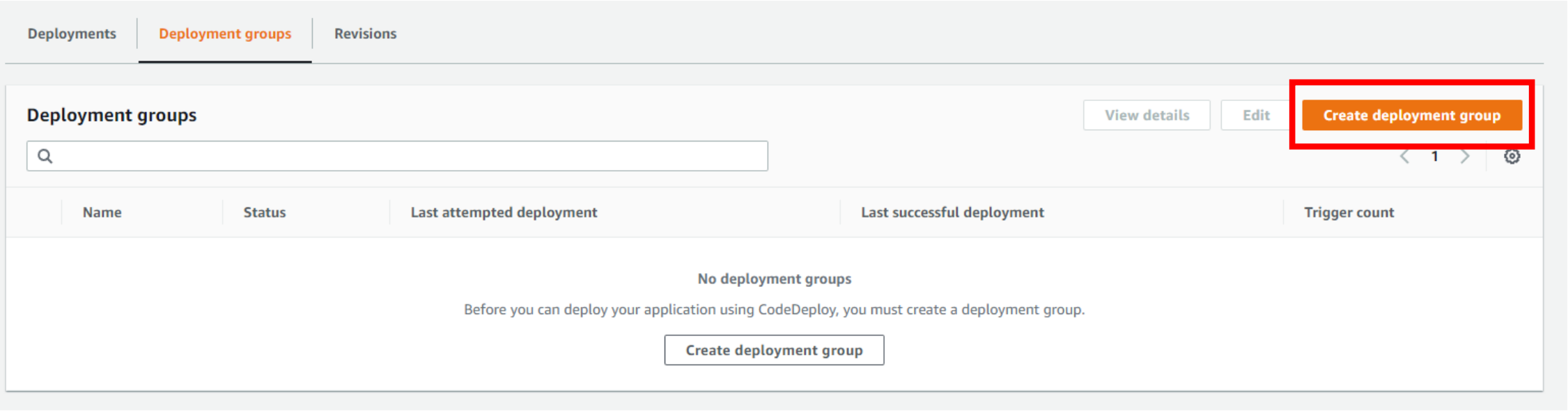
codedeploy를 생성합니다

6. create Codedeploy



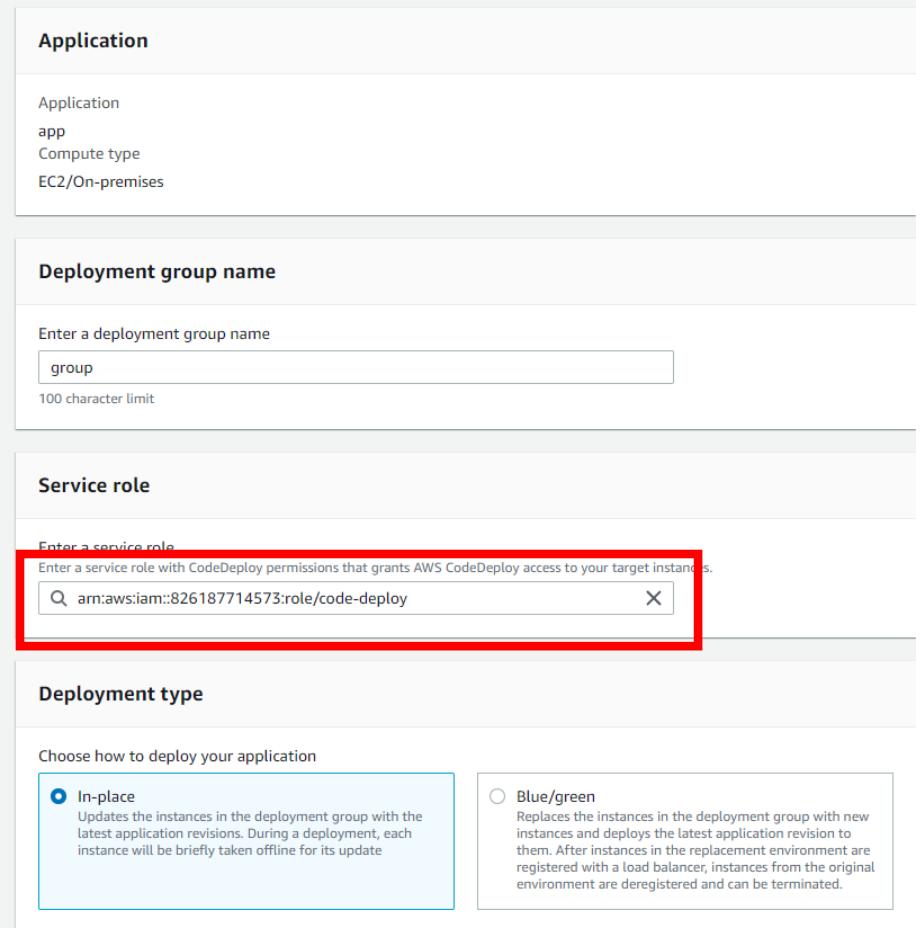
ec2상에 배포하기 때문에 ec2로 플랫폼을 선택하세요

6. create Codedeploy

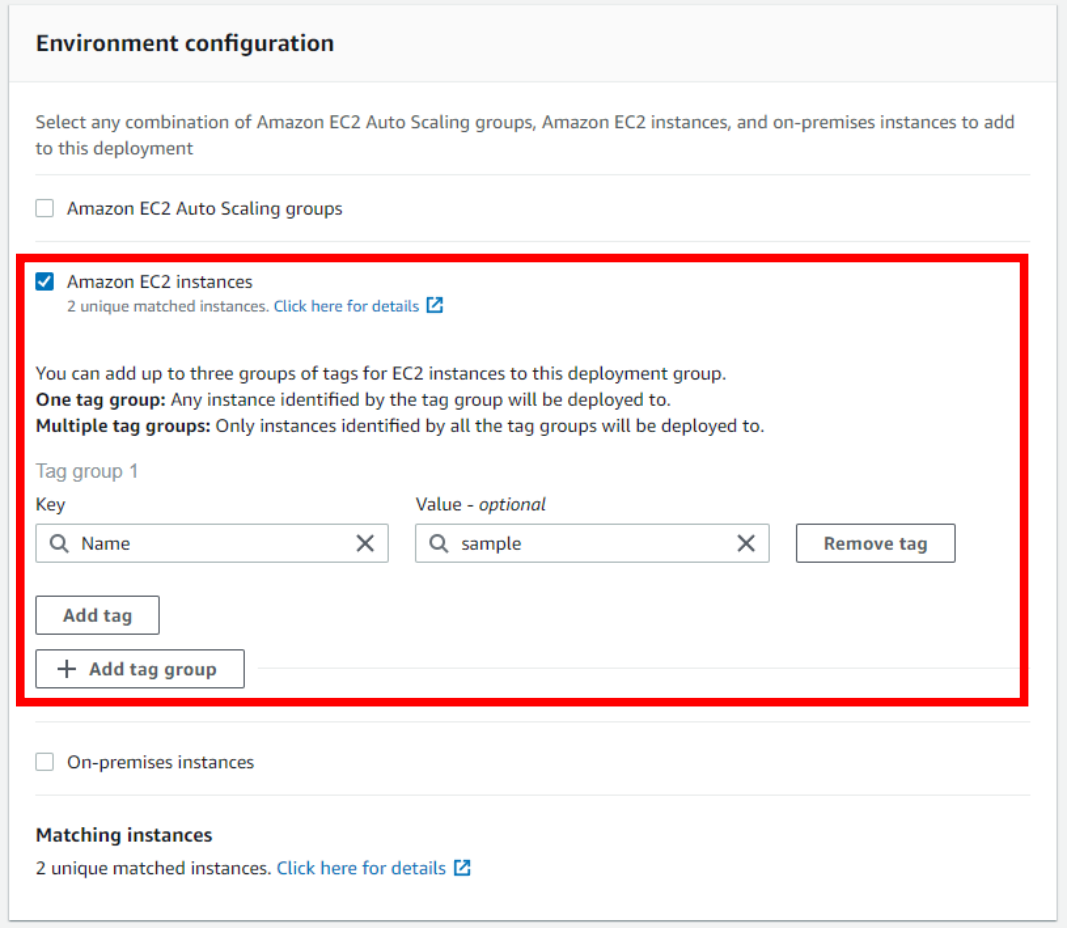


codedeploy의 배포 그룹을 추가합니다

6. create Codedeploy

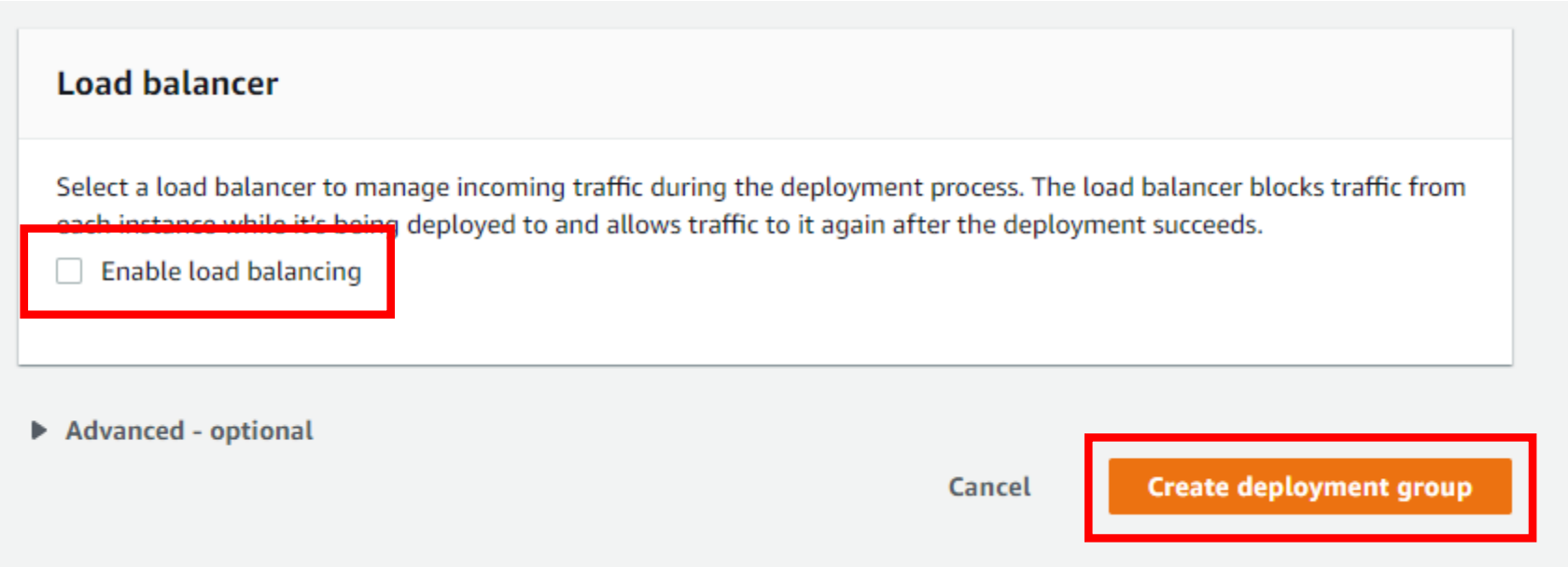


1. 생성한 codedeploy IAM role을 추가합니다



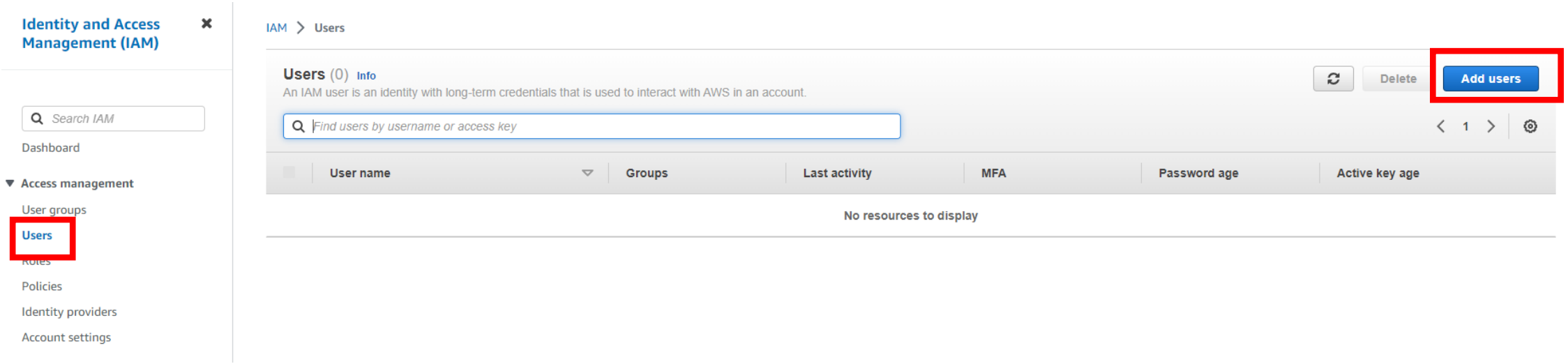
2. 생성한 ec2 instace와 연동합니다

6. create Codedeploy



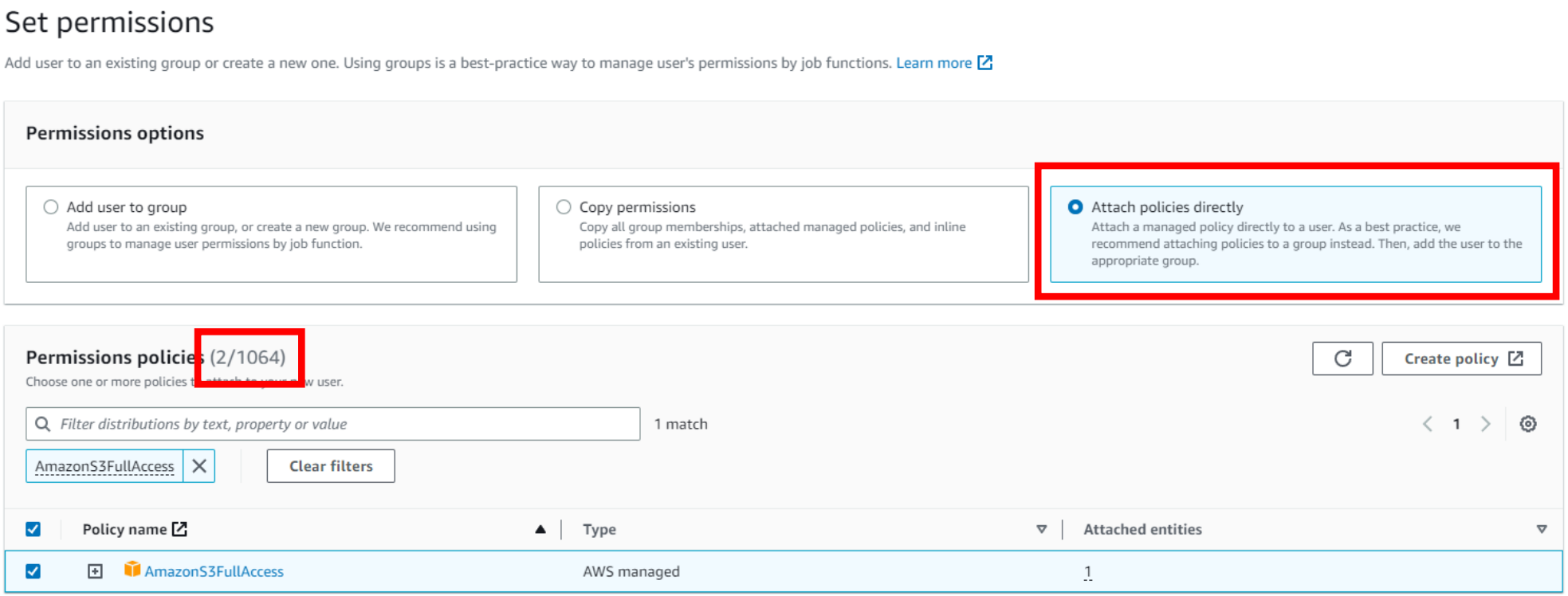
로드밸런싱을 해제합니다

7. create IAM user



ec2 서버 상에서 사용할 IAM user를 생성합니다

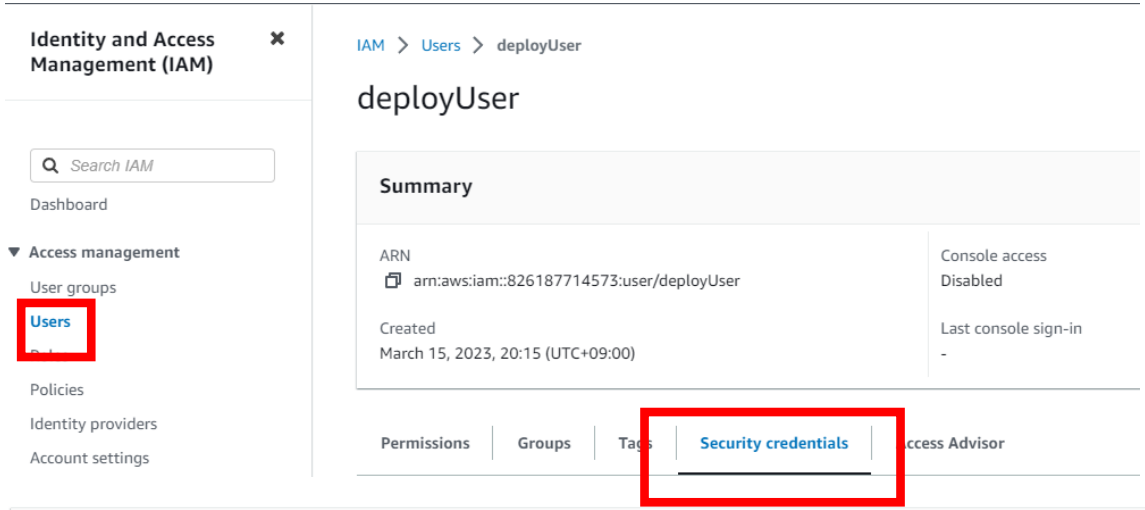
7. create IAM user

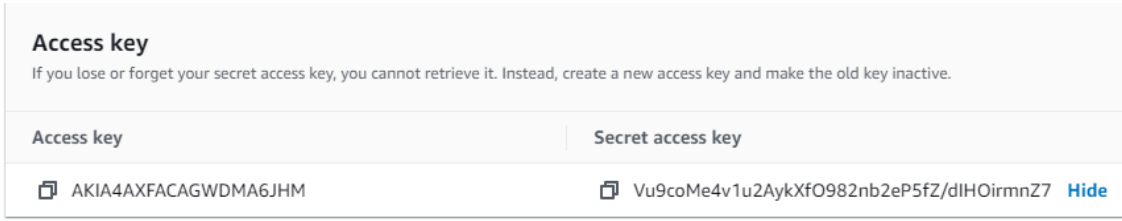


1. AWSCodeDeployFullAccess
2. AmazonS3FullAccess

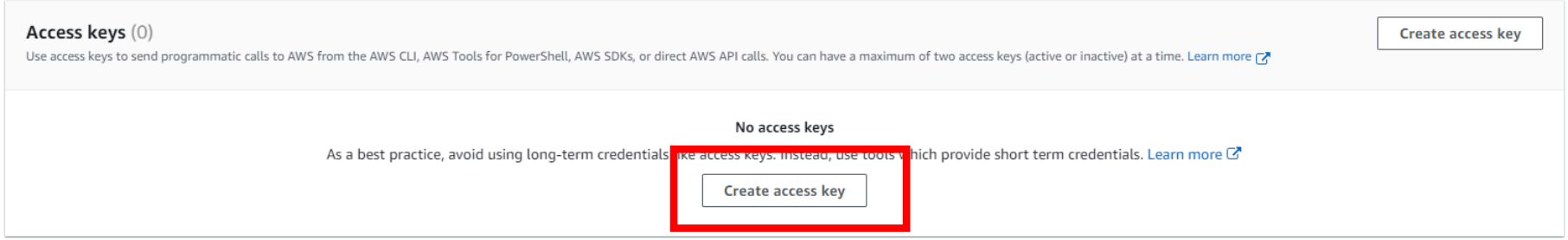
두 개의 권한 체크

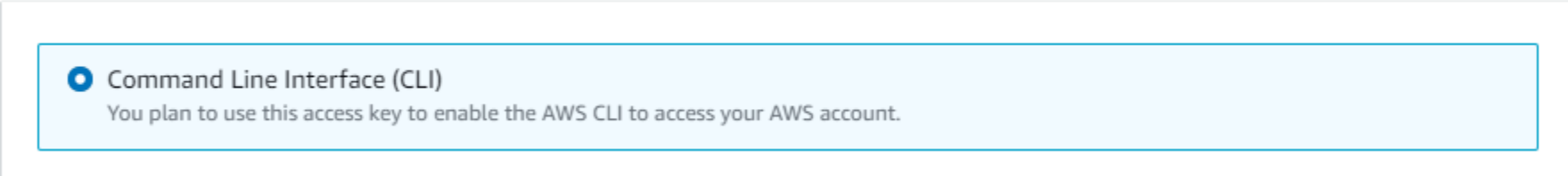
7. create IAM user: create access key



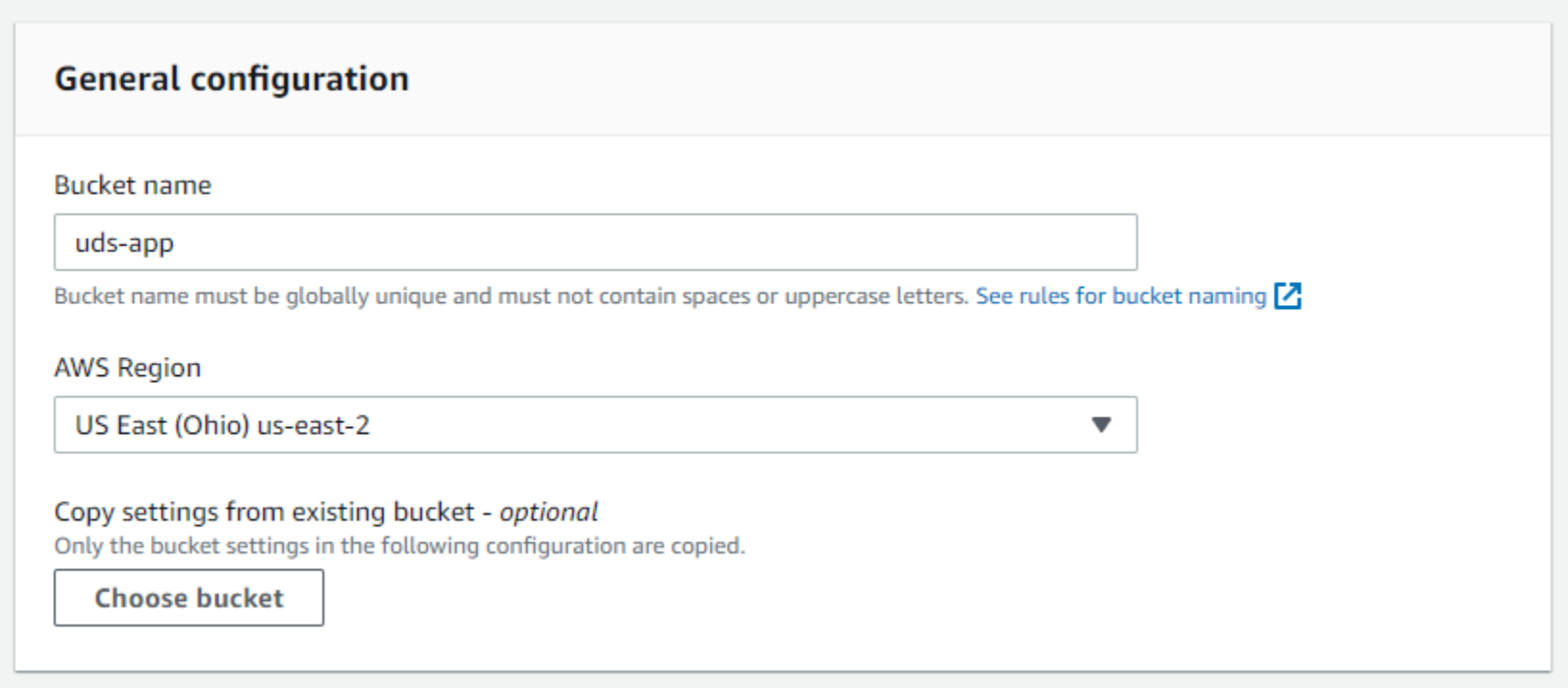


창 나가면 다시는 못 보니까
csv파일 다운 or 적어 두세요



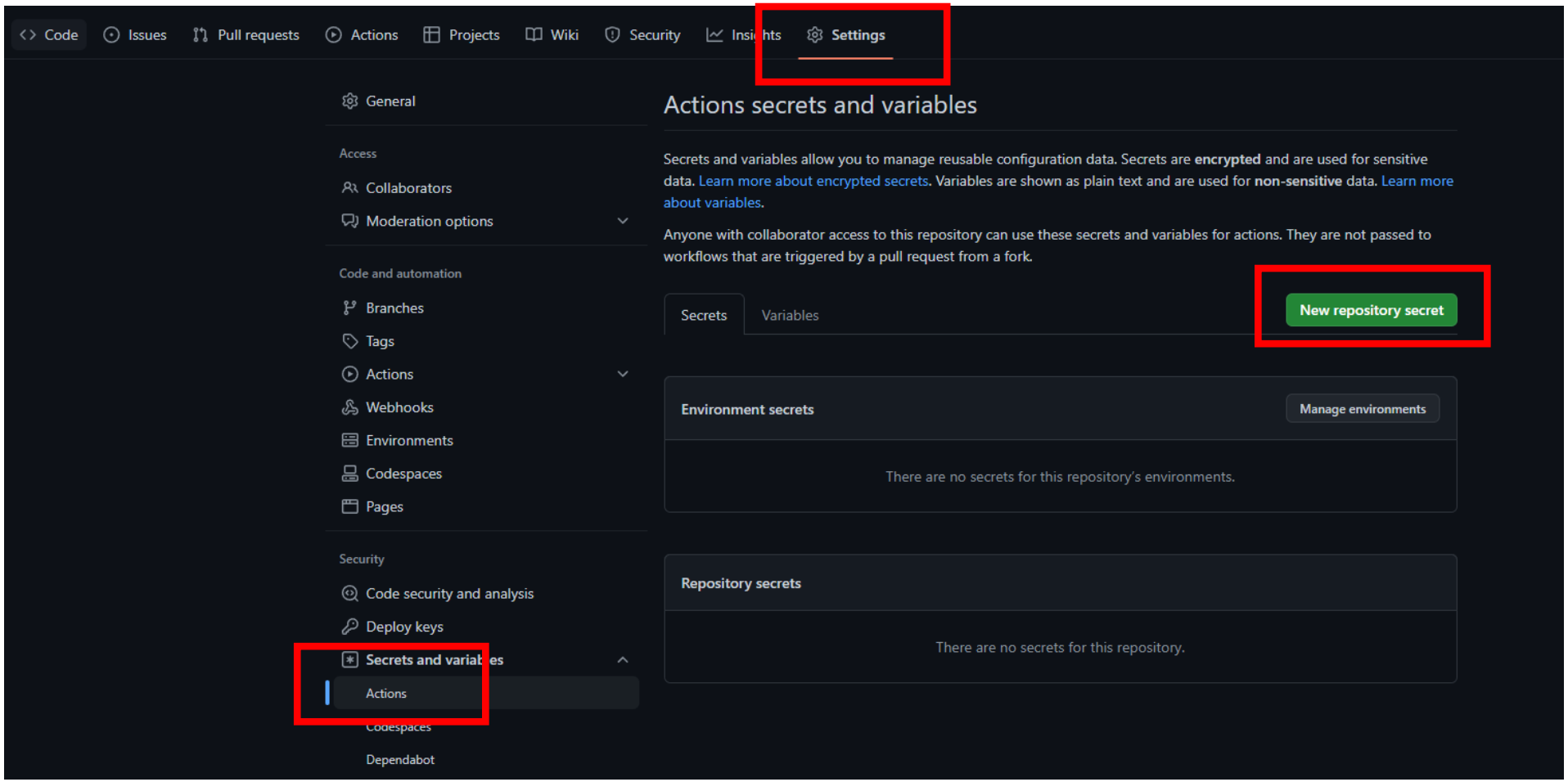


7. create S3 bucket



따로 설정할 것 없이 만드세요

4. create action workflow: add access key



깃허브 setting 〉 action에서 사용할 비밀 키를 추가합니다

4. create action workflow: add access key

Name *

AWS_ACCESS_KEY_ID

Secret *

AKIA4AXFACAGZUVLO5YU

Name *

AWS_SECRET_ACCESS_KEY

Secret *

MSxDhYhcVpTZSy9dTOP2IIvZfqj9x0LUoR5uiqHa

IAM user에서 생성한
AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY
추가합니다
github action script에서 사용됩니다

4. create action workflow: connect aws in ec2

ec2 상에 aws cli를 설치하고 만들어 둔 IAM user와 연동합니다

# 설치
$ sudo apt update
$ sudo apt install awscli

# 설치 확인
$ aws help

# 사용자 설정
$ aws configure

```
ubuntu@ip-172-31-10-195:~$ aws configure
AWS Access Key ID [None]: AKIA4AXFACAGZUVLO5YU
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-2
Default output format [None]:
```

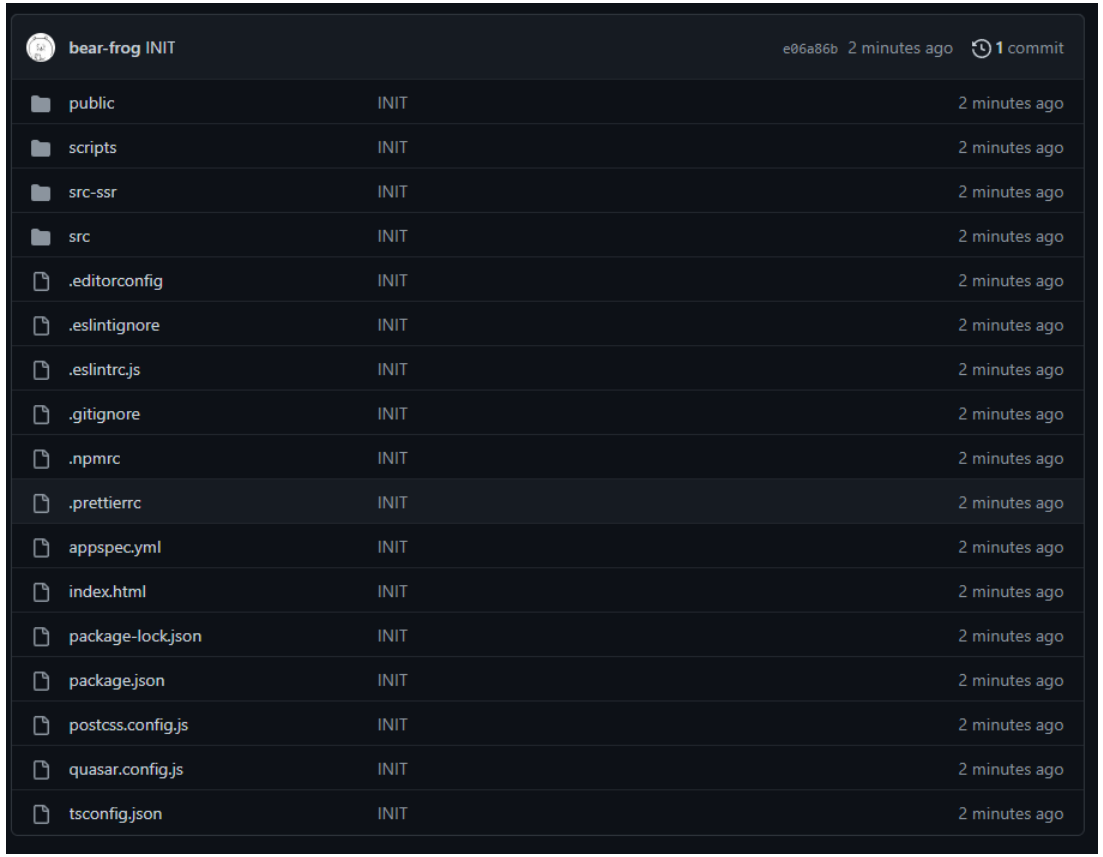
AWS Access Key ID [None]: 액세스 키를 입력
AWS Secret Access Key [None]: 시크릿 액세스 키를 입력
Default region name [None]: us-east-2
혹시 리전이 다르면 해당 리전 기입
Default output format [None]: 그냥 Enter 입력

4. create action workflow: connect aws in ec2

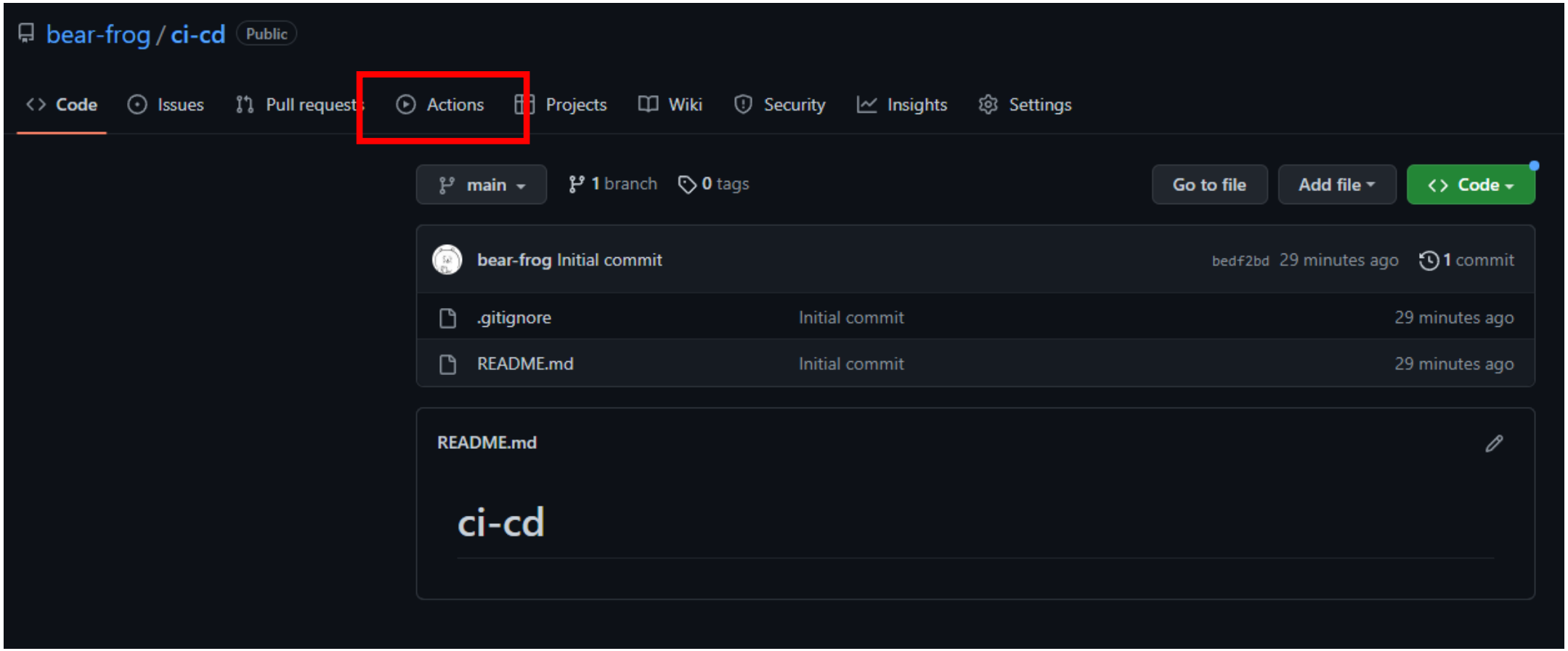
```
# codedeploy 재시작을 통해 IAM role을 갱신합니다
$ sudo service codedeploy-agent restart

# 이후 배포 시 에러 발생한다면
# 아래 명령어로 서버에서 로그를 확인 해 보세요
$ cat /var/log/aws/codedeploy-agent/codedeploy-agent.log
```

4. create action workflow: push file

cicd 스크립트를 작성하기 이전 로컬의 sample ssr app을 레포지토리에 push 해 주세요

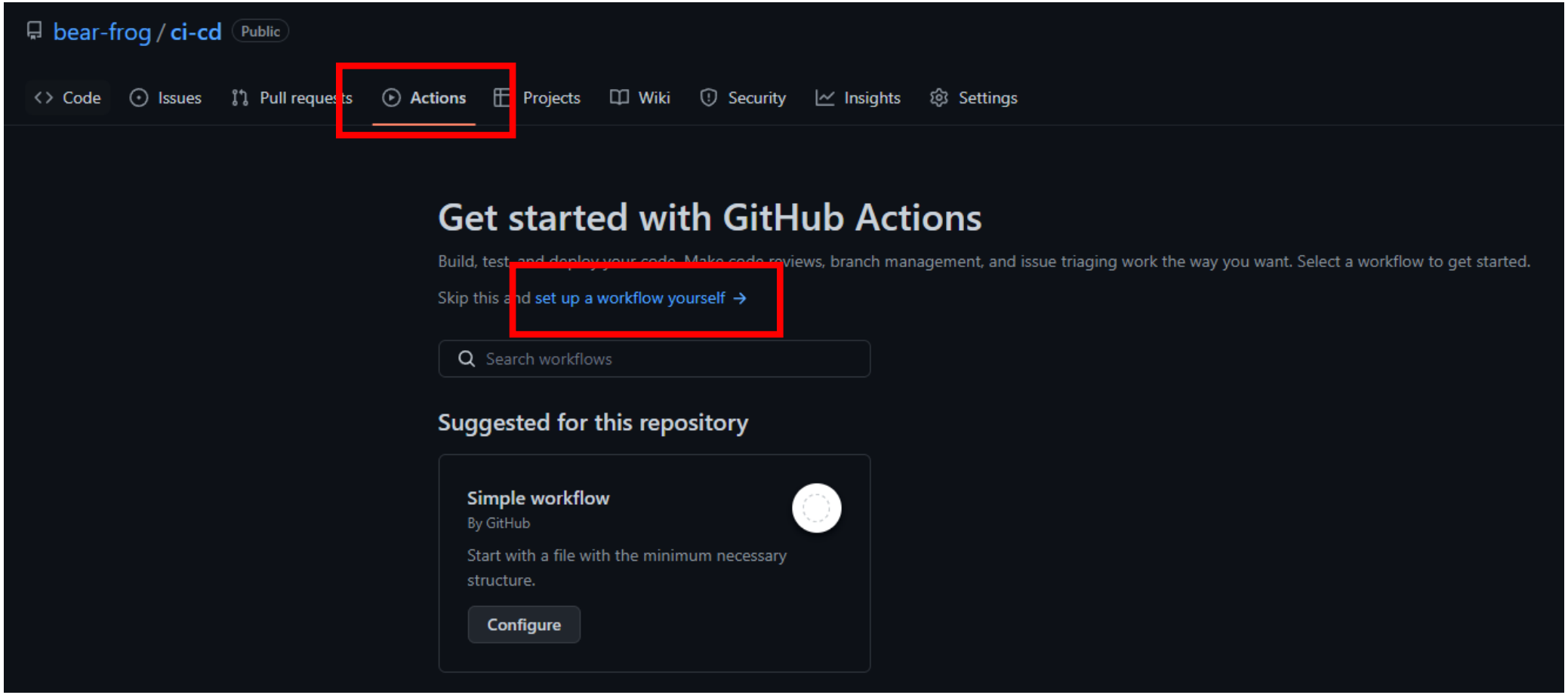


4. create action workflow



생성한 레포지토리 -> Actions

4. create action workflow



set up a workflow yourself

4. create action workflow: add script

- 참조
- https://github.com/bear-frog/ci-cd/blob/main/.github/workflows/main.yml

- 스크립트가 길어서 ppt상에 작성하지 않았어요
- 코드 보고 확인하세요

4. create action workflow: add script

```yaml
# This workflow will do a clean install of node dependencies, cache/restore them, build the source code and run tests across different versions of node
# For more information see: https://help.github.com/actions/language-and-framework-guides/using-nodejs-with-github-actions

name: main
```

- name: yml file 이름

4. create action workflow: add script

```yaml
on:
  push:
    branches: [main]
  pull_request:
    branches: [main]
```

- on: 이벤트에 반응
  - main 브렌치에 대한 push, pull_request 이벤트에 반응하여
  - github action 스크립트 실행

4. create action workflow: add script

```
jobs:
```

- jobs: github action에서 수행할 작업을 작성

```
build:
  runs-on: ubuntu-22.04

  strategy:
    matrix:
      node-version: [19.x]
      # See supported Node.js release schedule at https://nodejs.org/en/about/releases/
```

- build: 빌드 환경 작성
  - runs-on: 운영체제 기입
  - strategy: 개발 환경 기입
    - node-version: 사용 노드 버전 기입

4. create action workflow: add script

```yaml
steps:
  - name: Checkout source code.
    uses: actions/checkout@v2

  - name: Use Node.js ${{ matrix.node-version }}
    uses: actions/setup-node@v2
    with:
      node-version: ${{ matrix.node-version }}
```

- steps: 빌드 과정 서술
  - names: 명령어 이름 해당 이름을 통해 명령어 실행됨

4. create action workflow: add script

```
- name: build files
  working-directory: ./
  run: |
    npm i
    npm run build
```

- app build
  - 현재 디렉터리에서 npm i(종속성 설치)
  - npm run build(앱 빌드)
  - 두 명령어를 통해 빌드된 앱(dist) 폴더 생성

```
- name: zip distributions
  run: zip -r cicd-app.zip ./dist ./appspec.yml ./scripts
```

- S3 버킷 전송을 위한 zip파일 생성
  - 빌드된 폴더(./dist) codedeploy script(./appspec.yml, ./scripts)를 압축(cicd-app.zip)

4. create action workflow: add script

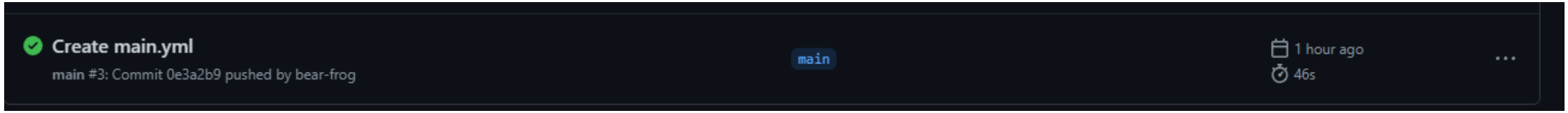
```
- name: upload to S3
  run: aws s3 cp --region us-east-2 ./cicd-app.zip s3://uds-cicd/public/
```

- S3버킷에 zip파일 업로드
  - 압축된 파일(cicd-app.zip)을 S3버킷(uds-cicd)의 public폴더로 이동

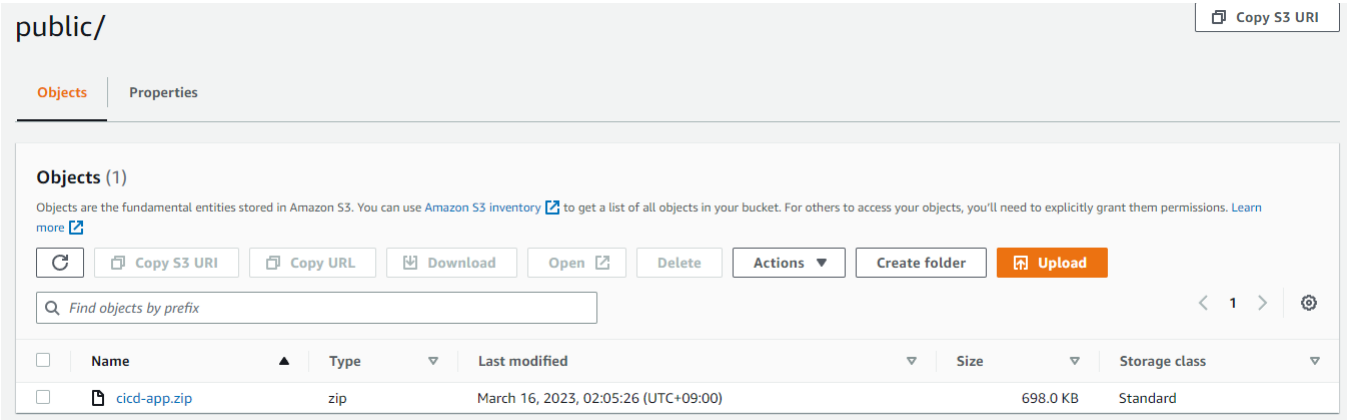
```
- name: deploy with AWS codeDeploy
  run: aws deploy create-deployment
    --application-name app
    --deployment-config-name CodeDeployDefault.OneAtATime
    --deployment-group-name group
    --s3-location bucket=uds-cicd,bundleType=zip,key=public/cicd-app.zip
```

- codedeploy를 통한 서버 배포
  - application name(app)
  - group name(group)을 작성
  - S3 버킷의 배포 파일을 명시(public/cicd-app.zip)

4. CI/CD



- 빌드 과정 완료



- S3버킷에 app이 업로드

4. CI/CD

○	d-PWYZE840M	Succeeded	In-place	EC2/On-premises	app	group	s3://uds-ci...	User action	Mar 16, 2023 2:05 AM (UTC+9:00)	Mar 16 2:05 A (UTC+

- 배포 과정 완료

```
ubuntu@ip-172-31-0-255:~/app$ ls
appspec.yml  dist  scripts
```

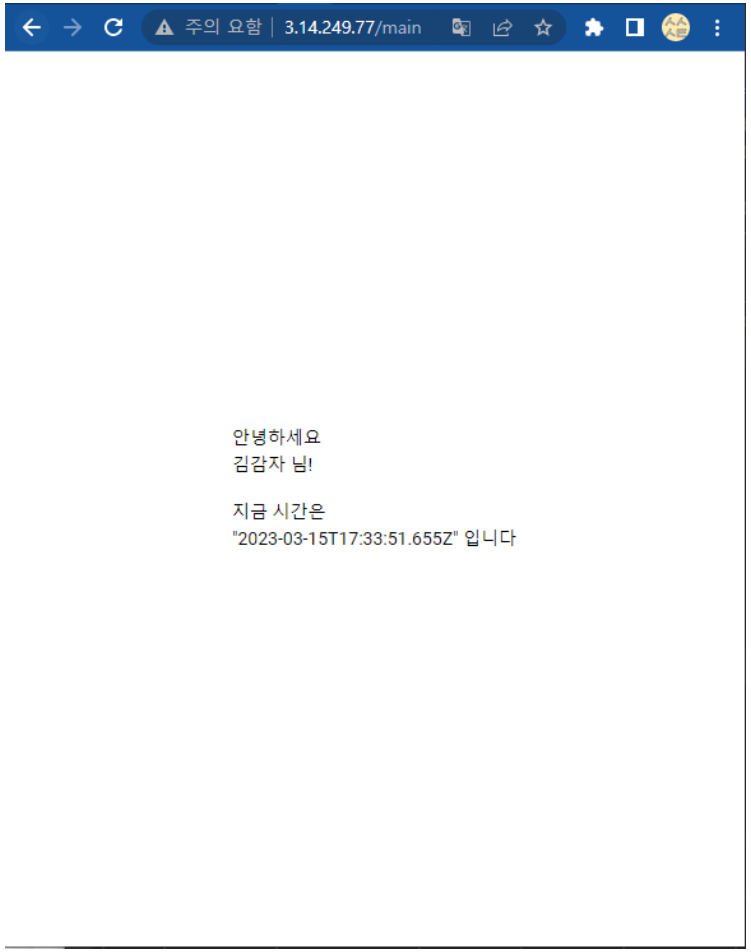
- ec2 서버에 업로드 완료

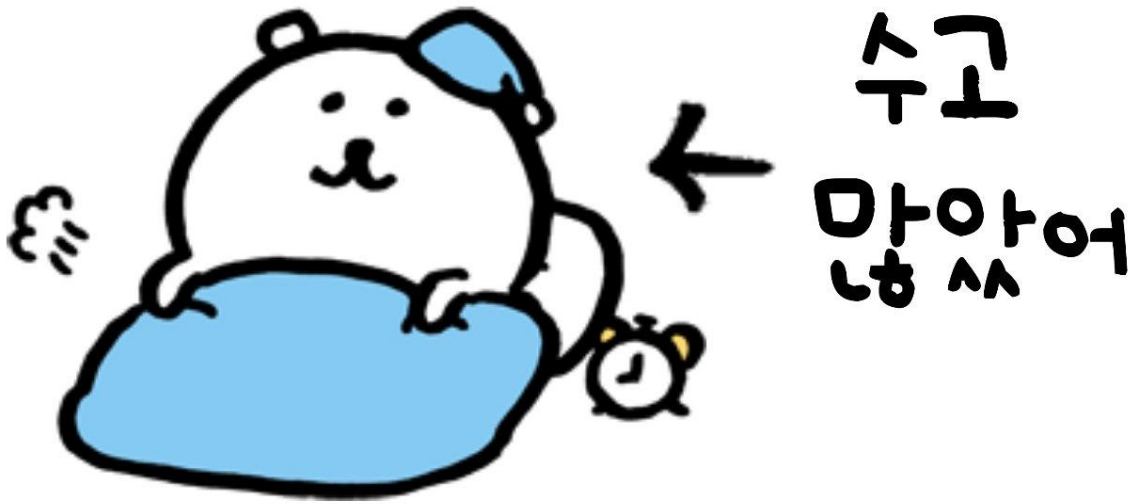
```
ubuntu@ip-172-31-0-255:~/app/dist/ssr$ pm2 status
```

id	name	namespace	version	mode	pid	uptime	↺	status	cpu	mem	user	watching
0	index	default	0.0.1	fork	7767	72s	0	online	0%	63.0mb	ubuntu	enabled

- after-deploy.sh에 따라 배포 후 앱 실행

4. CI/CD


주의 요함 | 3.14.249.77/main
안녕하세요
김감자 님!
지금 시간은
"2023-03-15T17:33:51.655Z" 입니다


수고
많았어
냥쌤