

# 정리본(인프라)

|      |                 |
|------|-----------------|
| ▣ 과목 | miniProject(정리) |
| ▣ 날짜 | @2026년 1월 30일   |

## 요약

### 1단계: AWS 기본 세팅 및 테라폼 인프라 구축

가장 먼저 클라우드 자원을 코드로 관리하는 **IaC(Infrastructure as Code)** 단계를 진행하셨습니다.

- **AWS 계정 및 IAM 설정:** AWS에서 API 호출을 위한 Access Key와 Secret Key를 발급받아 내 컴퓨터에 세팅하셨습니다.
- **테라폼(Terraform) 코드 작성:** `main.tf` 파일을 만들어 VPC, 서브넷, 그리고 가장 중요한 **EC2 인스턴스**와 \*\*보안 그룹(Security Group)\*\*을 정의하셨습니다.
- **보안 그룹 설정:** 외부에서 접속할 수 있도록 \*\*22번(SSH)\*\*과 **80번(HTTP)** 포트를开放하는 설정을 포함하셨습니다.
- **인프라 생성:** `terraform init` 후 `terraform apply` 를 통해 실제 AWS상에 `13.124.78.95` 주소를 가진 서버를 탄생시키셨습니다.

### 2단계: GitHub Repository Secrets 등록 (보안)

민감한 정보들을 코드에 노출하지 않기 위해 깃허브 저장소 설정(`Settings > Secrets and variables > Actions`)에 보안 키들을 등록하셨습니다.

- **DOCKERHUB\_USERNAME / TOKEN:** 도커 허브에 이미지를 올리고 내리기 위한 계정 정보입니다.
- **SSH\_PRIVATE\_KEY:** 테라폼으로 EC2를 만들 때 사용한 `.pem` 키 내용입니다.
- **기타 환경 변수:** 필요한 경우 `.env` 파일에 들어갈 비밀 정보들을 등록하셨습니다.

### 3단계: 로컬 프로젝트 및 CI/CD 스크립트 작성

VS Code에서 실제 배포에 필요한 지시서들을 만드셨습니다.

- **Workflow (`main.yml`):** 깃허브 액션의 엔진입니다. 여기서 코드를 체크아웃하고, 도커 이미지를 빌드하여 도커 허브에 `push` 하는 과정을 설계하셨습니다.

- **Inventory** (`hosts.ini`): 앤서블이 찾아갈 EC2 IP 주소(`13.124.78.95`)를 적어둔 주소록입니다.
- **Playbook** (`setup-docker.yml`): EC2 내부에서 실행될 작업들(도커 설치, 이미지 실행 등)을 정의한 지시서입니다.

## 4단계: GitHub Actions를 통한 CI/CD 가동

작성한 코드를 `git push` 하면 깃허브 액션이 아래 과정을 자동으로 수행합니다.

1. **Build:** 리눅스 환경에서 `Dockerfile` 을 읽어 `npm install` 이 포함된 도커 이미지를 생성합니다.
2. **Push:** 생성된 이미지를 도커 허브로 보냅니다.
3. **Ansible 실행:** 깃허브 액션이 서버에 SSH로 접속하여 앤서블을 돌립니다.
  - **도커 설치:** 서버에 도커가 없으면 자동으로 설치합니다.
  - **이미지 교체:** 기존 컨테이너를 삭제(`rm -f`)하고 새 이미지를 받아와 80번 포트로 띄웁니다.

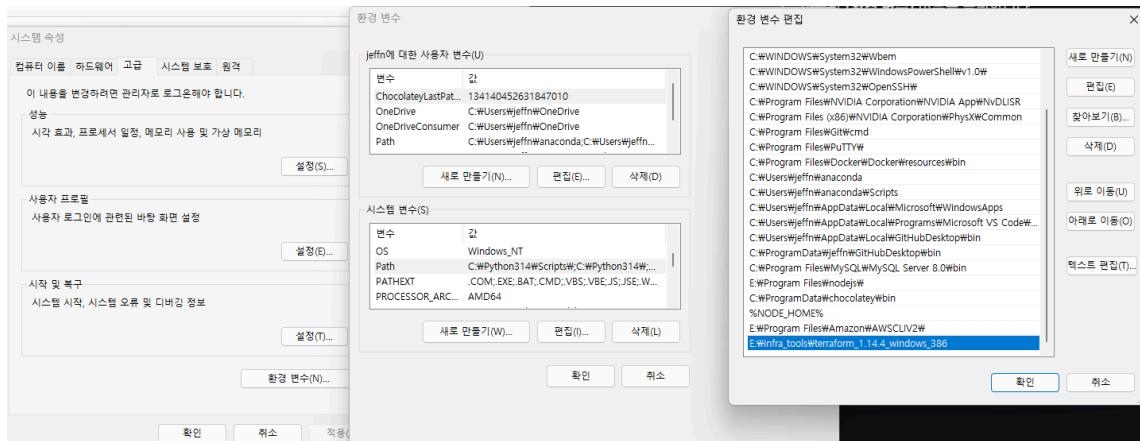
## 세부 과정

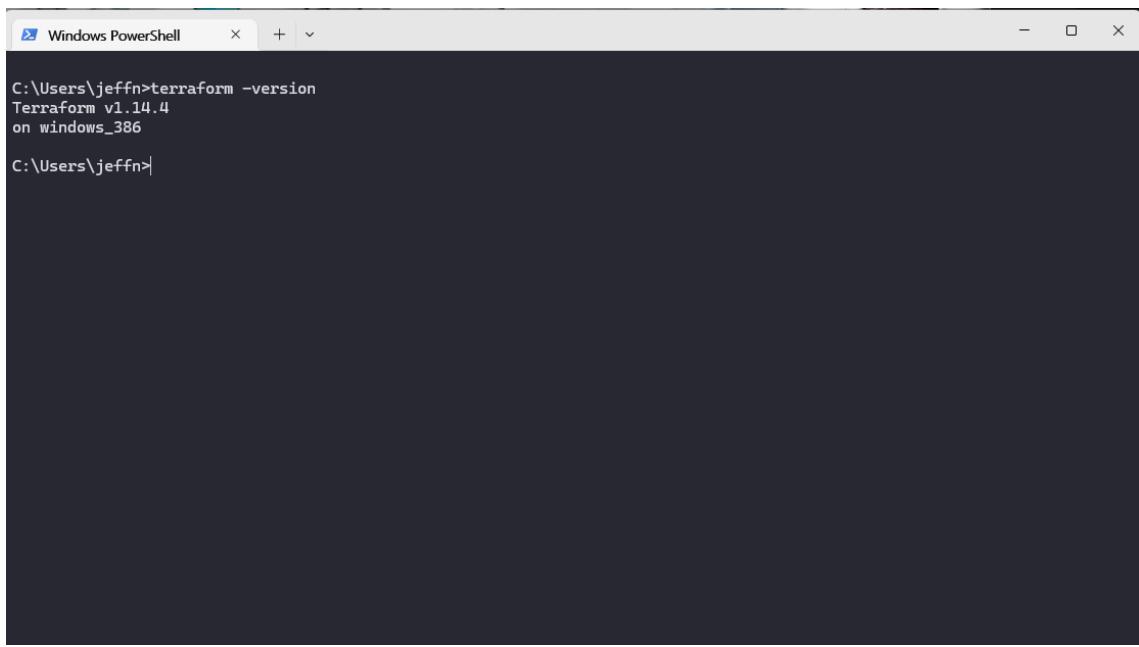
- 설치

Terraform

The screenshot shows the Terraform installation page. On the left, there's a sidebar with links for Operating Systems (macOS, Windows, Linux, FreeBSD, OpenBSD, Solaris), Release information, Resources (Tutorial Library, Certifications, Sandbox, Community Forum, Support, GitHub, Terraform Registry), and a Next steps link. The main content area is titled "Install Terraform" and shows the "1.14.4 (latest)" version. It has sections for "macOS", "Windows", and "Linux". Each section contains a "Binary download" section with "AMD64" and "ARM64" or "386" options, each with a "Download" button. Below these are "Package manager" sections for each operating system.

## 경로 설정





A screenshot of a Windows PowerShell window titled "Windows PowerShell". The window shows the command "terraform -version" being run in the directory "C:\Users\jeffn". The output indicates that Terraform v1.14.4 is installed on a windows\_386 system. The window has standard minimize, maximize, and close buttons at the top right.

```
C:\Users\jeffn>terraform -version
Terraform v1.14.4
on windows_386
C:\Users\jeffn>
```

## Kubernetes

← → C kubernetes.io/docs/tasks/tools/install-kubectl-windows/#install-kubectl-binary-on-windows... 🌐 🔍 ⚡ 🌐

 **kubernetes**

Documentation Kubernetes Blog Training Careers Partners Community Versions English

Kubernetes Documentation / Tasks / Install Tools / Install and Set Up kubectl on Windows

Search this site

▶ Documentation  
▶ Getting started  
▶ Concepts  
▼ Tasks  
  ▶ Install Tools  
    Install and Set Up kubectl on Linux  
    Install and Set Up kubectl on macOS  
    Install and Set Up kubectl on Windows  
  ▶ Administer a Cluster  
  ▶ Configure Pods and Containers  
  ▶ Monitoring, Logging, and Debugging  
  ▶ Manage Kubernetes Objects  
  ▶ Managing Secrets  
  ▶ Inject Data Into Applications  
  ▶ Run Applications  
  ▶ Run Jobs  
  ▶ Access Applications in a Cluster  
  ▶ Extend Kubernetes

# Install and Set Up kubectl on Windows

## Before you begin

You must use a kubectl version that is within one minor version difference of your cluster. For example, a v1.35 client can communicate with v1.34, v1.35, and v1.36 control planes. Using the latest compatible version of kubectl helps avoid unforeseen issues.

## Install kubectl on Windows

The following methods exist for installing kubectl on Windows:

- [Install kubectl binary on Windows \(via direct download or curl\)](#)
- [Install on Windows using Chocolatey, Scoop, or winget](#)

### Install kubectl binary on Windows (via direct download or curl)

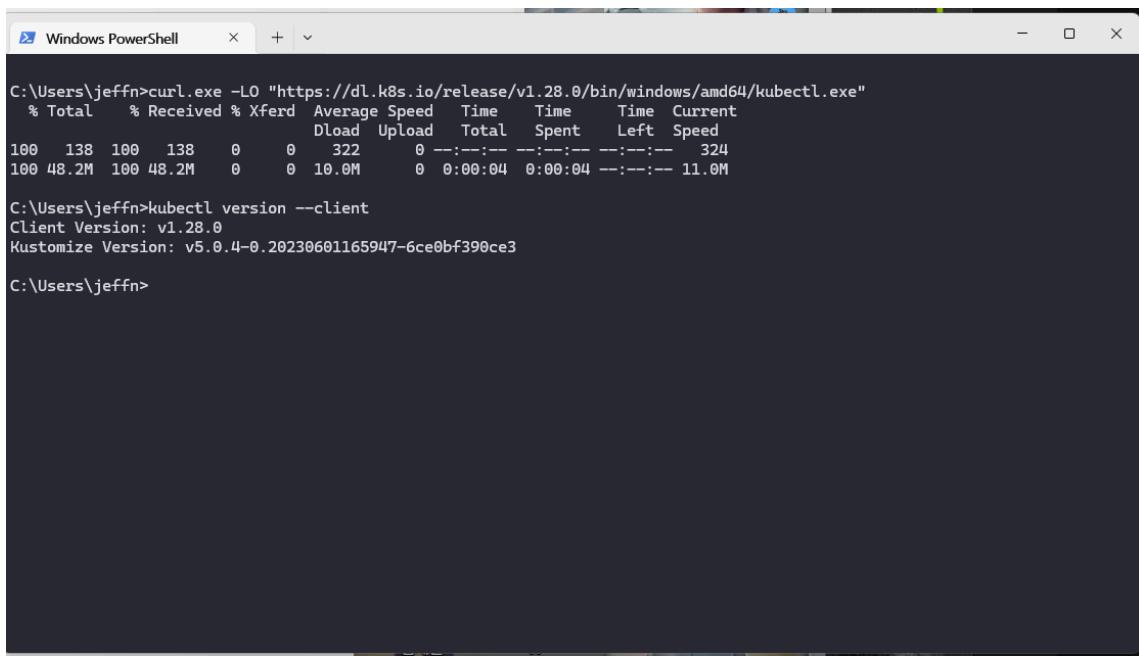
1. You have two options for installing kubectl on your Windows device
  - Direct download:

Download the latest 1.35 patch release binary directly for your specific architecture by visiting the [Kubernetes release page](#). Be sure to select the correct binary for your architecture (e.g., amd64, arm64, etc.).
  - Using curl:

If you have `curl` installed, use this command:

```
curl.exe -LO "https://dl.k8s.io/release/v1.35.0/bin/windows/amd64/kubec
```

**Note:**  
To find out the latest stable version (for example, for scripting), take a look at <https://dl.k8s.io/release/stable.txt>.

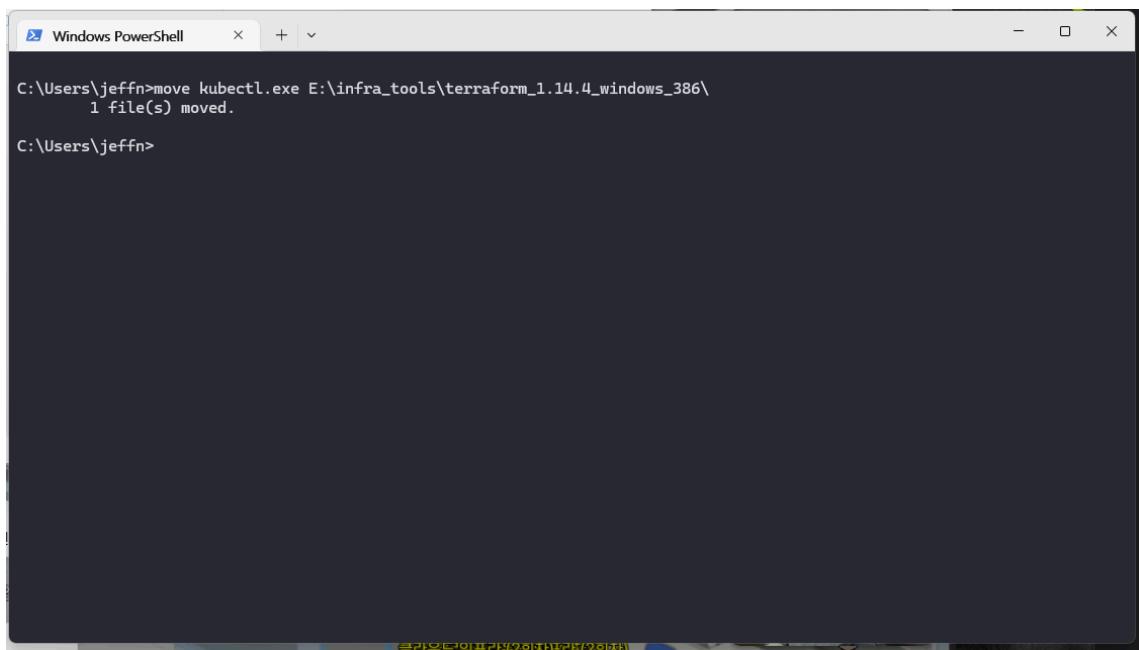


```
C:\Users\jeffn>curl.exe -LO "https://dl.k8s.io/release/v1.28.0/bin/windows/amd64/kubectl.exe"
% Total    % Received % Xferd  Average Speed   Time   Time  Current
          Dload  Upload Total Spent   Left Speed
100  138  100  138    0     0  322      0 --:--:-- --:--:-- 324
100 48.2M  100 48.2M    0     0  10.0M      0  0:00:04  0:00:04 --:--:-- 11.0M

C:\Users\jeffn>kubectl version --client
Client Version: v1.28.0
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3

C:\Users\jeffn>
```

위에 테라폼 경로에 같이 넣어두기



```
C:\Users\jeffn>move kubectl.exe E:\infra_tools\terraform_1.14.4_windows_386\
1 file(s) moved.

C:\Users\jeffn>
```

AWS CLI

AWS 시작하기 서비스 가이드 개발자 도구 AI 회수  가이드 검색 콘솔로 반환하기

## AWS 명령줄 인터페이스 < 버전 2 사용자 가이드

- ▶ AWS CLI에 대하여
- ▼ 시작하기
  - 필수 조건
  - 설치/업데이트**
    - 이전 릴리스
    - 소스 코드에서 빌드하고 설치하세요
    - 아마존 ECR 퍼블릭/도커 설정
  - ▶ AWS CLI 구성
  - ▶ 인증 및 접근 자격 증명
  - ▶ AWS CLI를 사용하여
  - ▶ AWS CLI 예제
  - ▶ 보안
  - ▶ 이민 가이드
  - 제거
  - 문서 이력

### ▼ 윈도우

#### 설치 및 업데이트 요구 사항

- 저희는 마이크로소프트에서 지원하는 64비트 윈도우 버전에서 AWS CLI를 지원합니다.
- 소프트웨어 설치를 위한 관리자 권한

#### AWS CLI를 설치 또는 업데이트하세요

Windows에서 AWS CLI를 최신 버전으로 업데이트하려면 매번 새 설치 프로그램을 다운로드하여 이전 버전을 덮어쓰세요. AWS CLI는 정기적으로 업데이트됩니다. 최신 버전 출시일은 [AWS CLI 버전 2 변경 로그를 참조하세요.](#) GitHub에서 확인하세요.

- Windows(64비트)용 AWS CLI MSI 설치 프로그램을 다운로드하여 실행하십시오.  
<https://awscli.amazonaws.com/AWSCLIV2.msi>  
 또는 msieexec MSI 설치 프로그램을 실행하는 명령어를 실행할 수도 있습니다.



msieexec 와 함께 사용할 수 있는 다양한 매개변수에 대해서는 [msieexec를 msieexec 참조하십시오.](#) Microsoft Docs 웹사이트에서 확인할 수 있습니다. 예를 들어, /qn 무인 설치를 위해 플래그를 사용할 수 있습니다.



- 설치를 확인하려면 시작 메뉴를 열고 명령 프롬프트 창을 열려면 검색창에 명령어를 입력하세요 cmd. 명령 프롬프트에서 명령어를 입력하면 됩니다 aws --version .



```
C:\Users\jeffn>aws --version
aws-cli/2.33.9 Python/3.13.11 Windows/11 exe/AMD64

C:\Users\jeffn>cd
```

#### • IAM 계정 생성 방법

aws 검색 [알트+S] 글로벌 ▾ BAEKJUNHO (9850-9032-2396) BAEKJUNHO

IAM 대시보드

## IAM 대시보드 정보

### 보안 권장 사항 1

⚠️ 루트 사용자에 대해 MFA 추가  
루트 사용자에 대해 MFA 추가 - 루트 사용자에 대해 다중 인증(MFA)을 활성화하여 이 계정의 보안을 강화합니다.

⚠️ 루트 사용자에게 활성 액세스 키가 없음  
루트 사용자 대신 IAM 사용자에 연결된 액세스 키를 사용하면 보안이 향상됩니다.

MFA 추가

### IAM 리소스

이 AWS 계정의 리소스

| 사용자 그룹 | 사용자 | 역할 | 정책 | ID 제공업체 |
|--------|-----|----|----|---------|
| 0      | 0   | 3  | 0  | 0       |

### 새로운 기능

IAM의 기능 업데이트

모두 보기

- AWS 서비스 참조 정보, 이제 SDK 작업과 액션 간 매핑 지원. 2개월 전
- AWS IAM, 네트워크 경계 제어를 위한 새로운 VPC 엔드포인트 조건 키 출시. 4개월 전
- Amazon Bedrock, 개발 간소화를 위해 API 키 도입. 6개월 전
- AWS 서비스 참조 정보, 이제 서비스 작업에 대한 주석 지원. 7개월 전

▼ 이상

### AWS 계정

계정 ID  
985090322396

계정 별칭  
생성

이 계정의 IAM 사용자를 위한 로그인 URL  
<https://985090322396.signin.aws.amazon.com/console>

CloudShell 워크스페이스 AWS Organizations IAM 사용자 IAM 대시보드 AWS Organizations IAM 사용자 IAM 대시보드 © 2024 Amazon Web Services, Inc. 또는 계열사 개인 정보 보호 약관 고객 기본 설정

## 사용자 세부 정보 지정

### 사용자 세부 정보

사용자 이름  
miniProject-admin

사용자 이름은 최대 64자까지 가능합니다. 유효한 문자: A~Z, a~z, 0~9 및 +, ., @, \_-(하이픈)

AWS Management Console에 대한 사용자 액세스 권한 제공 – 선택 사항  
콘솔 액세스 외에도, SignInLocalDevelopmentAccess 권한이 있는 사용자는 액세스 키 없이도 동일한 콘솔 자격 증명을 사용하여 프로그래밍 방식으로 액세스할 수 있습니다.

콘솔 암호

자동 생성된 암호  
사용자를 생성한 후 암호를 볼 수 있습니다.

사용자 지정 암호  
사용자의 사용자 지정 암호를 입력합니다.

다음 이상이어야 합니다.  
 다음 문자 유형 중 세 가지 이상을 조합하여 포함해야 합니다. 대문자(A~Z), 소문자(a~z), 숫자(0~9), 기호 ! @ # \$ % ^ & \* ( ) \_ -(하이픈) = [ ] { } ` '

암호 표시

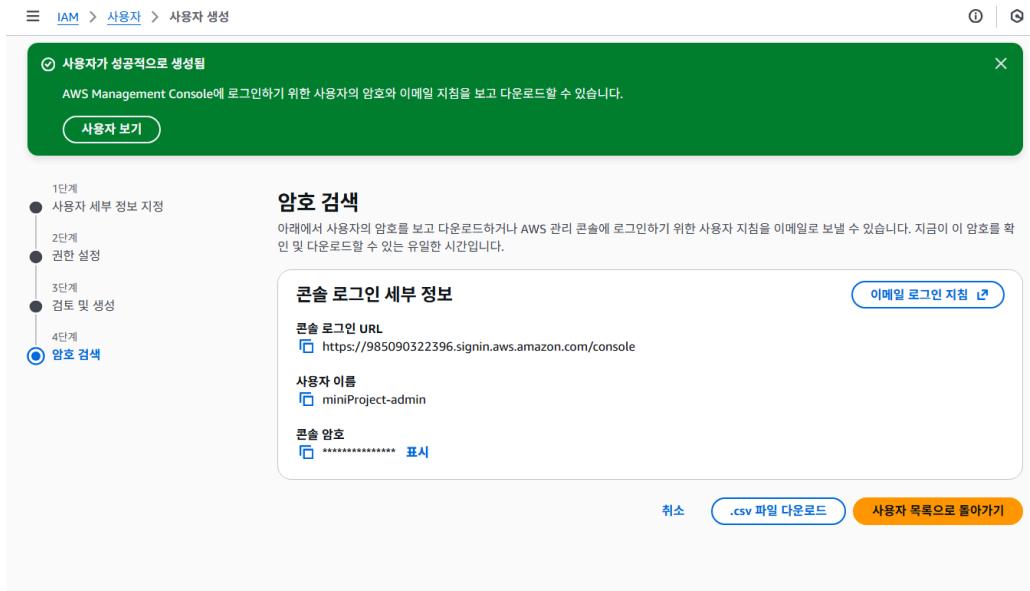
사용자는 다음 로그인 시 새 암호를 생성해야 합니다 - 권장  
사용자는 IAMUserChangePassword 정책을 자동으로 가져와 암호를 변경할 수 있도록 허용합니다.

이 IAM 사용자를 생성한 후 액세스 키 또는 AWS CodeCommit이나 Amazon Keyspaces에 대한 서비스별 보안 인증 정보를 통해 프로그래밍 방식 액세스를 생성할 수 있습니다. 자세히 알아보기

## • 권한 정책에 AdministratorAccess 검색 후 가장 위 정책 선택

## • 사용자 생성

## • 세부 정보는 꼭 따로 저장



- 액세스 키 만들기(반드시 리전 확인하기)

- 리전 - 만드는 위치



아시아 태평양(서울)

IAM > 사용자 > miniProject-admin

## Identity and Access Management(IAM)

Q IAM 검색

대시보드

▼ 액세스 관리

사용자 그룹

사용자

역할

정책

ID 제공업체

계정 설정

루트 액세스 관리

임시 위임 요청 신규

▼ 보고서 액세스

Access Analyzer

리소스 분석 신규

미사용 액세스

분석기 설정

자격 증명 보고서

조직 활동

서비스 제어 정책

리소스 제어 정책

## miniProject-admin 정보

요약

ARN  
arn:aws:iam::985090322396:user/miniProject-admin

콘솔 액세스  
⚠️ MFA 없이 활성화됨

액세스 키 1  
액세스 키 만들기

생성됨  
January 29, 2026, 11:15 (UTC+09:00)

마지막 콘솔 로그인  
① 안 함

권한 | 그룹 | 태그 | **보안 자격 증명** | 마지막 액세스

### 콘솔 로그인

콘솔 로그인 링크  
https://985090322396.signin.aws.amazon.com/console

콘솔 암호  
활성화되지 않음

마지막 콘솔 로그인  
① 안 함

콘솔 액세스 관리

### 멀티 팩터 인증(MFA) (0)

MFA를 사용하여 AWS 환경의 보안을 강화합니다. MFA로 로그인하려면 MFA 디바이스의 인증 코드가 필요합니다. 각 사용자는 MFA 디바이스를 최대 8개까지 할당할 수 있습니다. [자세히 알아보기](#)

삭제 | 재등기화 | MFA 디바이스 할당

| 유형  | 식별자 | 인증 | 생성 날짜 |
|---|-----|----|-------|
| MFA 디바이스가 없습니다. MFA 디바이스를 할당하여 AWS 환경 보안 개선하기 |     |    |       |

MFA 디바이스 할당

### 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. [자세히 알아보기](#)

액세스 키가 없습니다. 액세스 키와 같은 장기 보안 인증을 사용하지 않는 것이 모범 사례입니다. 대신 단기 보안 인증을 제공하는 도구를 사용하세요. [자세히 알아보기](#)

액세스 키 만들기

- **테라폼(Terraform) 연결:** 사용자의 VS Code에서 테라폼 명령을 내릴 때, AWS가 "너 누구야?"라고 물으면 이 액세스 키를 보여주어 본인임을 인증하고 인프라(EC2 등)를 생성하게 합니다.
  - **GitHub Actions 연결:** 깃허브 액션 서버가 사용자 대신 AWS EC2에 접속하거나 이미지를 다룰 때 사용할 '디지털 열쇠' 역할을 합니다.

☰ IAM > 사용자 > miniProject-admin > 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 정보

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

**사용 사례**

- 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**  
이 액세스 키를 사용하여 AWS 리소스에 액세스해야 하는 AWS 외부의 데이터 센터 또는 기타 인프라에서 실행 중인 워크로드를 인증할 것입니다.
- 기타**  
귀하의 사용 사례가 여기에 나열되어 있지 않습니다.

**권장되는 대안**

- AWS CLI V2와 aws login 명령을 사용하여 기존 콘솔 자격 증명으로 CLI에 액세스합니다. [자세히 알아보기 ↴](#)
- 브라우저 기반 AWS CloudShell을 사용하여 명령을 실행합니다. [자세히 알아보기 ↴](#)

**확인**

위의 권장 사항을 이해했으며 액세스 키 생성을 계속하려고 합니다.

취소 다음

## • 액세스 키도 반드시 따로 저장 + .csv 파일 다운로드

☰ IAM > 사용자 > miniProject-admin > 액세스 키 만들기

◎ 지금이 아니면 비밀 액세스 키를 보거나 다운로드할 수 없습니다. 나중에 복구할 수 없습니다. 하지만 언제든지 새 액세스 키를 생성할 수 있습니다. ×

1단계  
액세스 키 모범 사례 및 대안

2단계 - 선택 사항  
설명 태그 설정

3단계  
액세스 키 검색

### 액세스 키 검색 정보

분실하거나 잊어버린 비밀 액세스 키는 검색할 수 없습니다. 대신 새 액세스 키를 생성하고 이전 키를 비활성화합니다.

| 액세스 키 | 비밀 액세스 키                      |
|-------|-------------------------------|
|       | AKIA6KW7VN7OO264JFOO  **** 표시 |

**액세스 키 모범 사례**

- 액세스 키를 일반 텍스트, 코드 리포지토리 또는 코드로 저장해서는 안됩니다.
- 더 이상 필요 없는 경우 액세스 키를 비활성화하거나 삭제합니다.
- 최소 권한을 활성화합니다.
- 액세스 키를 정기적으로 교체합니다.

액세스 키 관리에 대한 자세한 내용은 [AWS 액세스 키 관리 모범 사례](#)를 참조하세요.

.csv 파일 다운로드 완료

## • AWS 자격 증명 등록

- 리전 확인

- 없으면 리전 확인하고 다시 만들어야 함

```
C:\Users\jeffn>aws configure
AWS Access Key ID [None]: AKIA6KW7VN700264JFOO
AWS Secret Access Key [None]: jdxYaxL288A9c6iRsjsKFIIVY7yr/MY7HNeDKtdsb
Default region name [None]: ap-northeast-2
Default output format [None]: json
```

- 키 페어 만들기 (서버로 들어가는 유일한 현관문 열쇠)

- 보안을 위한 전용 열쇠 (SSH 인증)]
- 없으면?

- **접속 불가:** 서버가 만들어져도 그 안으로 들어가서 도커를 설치하거나 앱을 실행할 방법이 아예 없습니다.
- **보안 취약:** 만약 비밀번호 방식을 쓴다면, 서버가 생성되자마자 전 세계 해커들의 공격 대상이 되어 금방 뚫릴 수 있습니다.

| 구분   | 액세스 키 (Access Key)                   | 키 페어 (Key Pair)                      |
|------|--------------------------------------|--------------------------------------|
| 하는 일 | 테라폼이 AWS에 접속해서 "여기 서버 한 대 지어줘"라고 요청함 | 앤서블이 지어진 서버 문을 열고 들어가 "도커 설치해"라고 명령함 |
| 비유   | 아파트 관리실에 가서 입주 허가를 받는 도장             | 우리 집 현관문을 여는 실물 열쇠                   |

|            |                          |                           |
|------------|--------------------------|---------------------------|
| 파일 형태      | 문자열(ID/Secret) 형태        | .pem 파일 형태                |
| Secrets 등록 | DOCKERHUB_TOKEN 등과 함께 등록 | SSH_PRIVATE_KEY라는 이름으로 등록 |

The screenshot shows the AWS Management Console with the EC2 service selected. In the left sidebar, under the EC2 section, the 'Key Pairs' option is highlighted. The main content area is titled 'Key Pairs' and contains a search bar and a table header with columns for 'Name', 'Type', 'Status', and 'Last modified'. Below the table, a message says 'No key pairs found'.

- 생성한 .pem 키는 terraform 폴더 안에 넣기

## 키 페어 생성 정보

### 키 페어

프라이빗 키와 퍼블릭 키로 구성되는 키 페어는 인스턴스에 연결할 때 자격 증명을 증명하는 데 사용하는 보안 자격 증명 세트입니다.

#### 이름

이름에는 최대 255개의 ASCII 문자가 포함됩니다. 앞 또는 뒤에 공백을 포함할 수 있습니다.

#### 키 페어 유형 정보

 RSA ED25519

#### 프라이빗 키 파일 형식

- .pem  
OpenSSH와 함께 사용
- .ppk  
PuTTY와 함께 사용

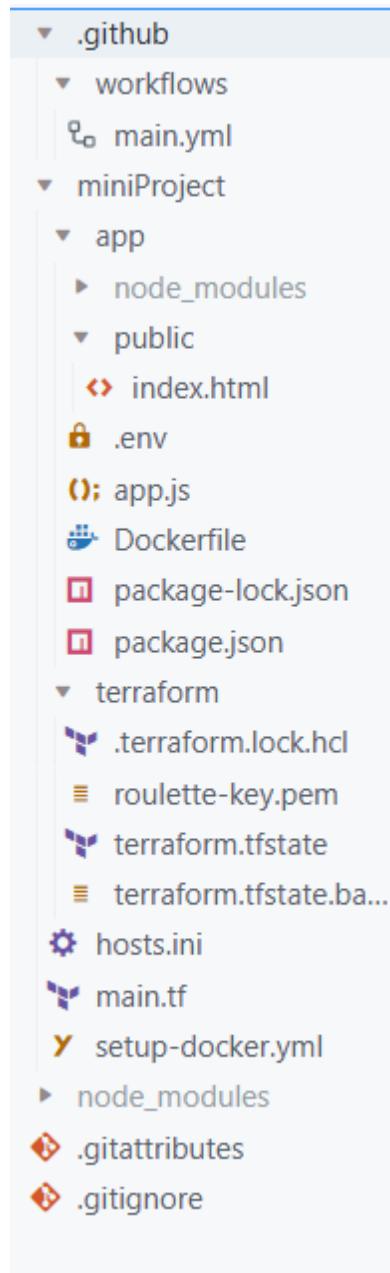
#### 태그 - 선택 사항

리소스에 연결된 태그가 없습니다.

[새로운 태그 추가](#)

최대 50개의 태그를 더 추가할 수 있습니다.

[취소](#)[키 페어 생성](#)



- 인스턴스 확인

AWS | 검색 | [일정+S] | 아시아 태평양 (서울) | BAEKJUNHO (9850-9032-2396) | BAEKJUNHO

EC2 > 인스턴스 > i-0215122488ed3bd5c

### i-0215122488ed3bd5c (Roulette-Server)에 대한 인스턴스 요약 정보

less than a minute 전에 업데이트됨

|  |  |   |
|--|--|---|
| 인스턴스 ID  | 퍼블릭 IPv4 주소  | 프라이빗 IPv4 주소  |
| i-0215122488ed3bd5c                                    | 13.125.132.56   개방 주소법   | 172.31.39.52  |
| IPv6 주소  | 인스턴스 상태  | 퍼블릭 DNS   |
| -  | 실행 중   | ec2-13-125-132-56.ap-northeast-2.compute.amazonaws.com   개방 주소법 |
| 호스트 이름 유형  | 프라이빗 IP DNS 이름(IPv4만 해당)   | 탄력적 IP 주소   |
| IP 이름: ip-172-31-39-52.ap-northeast-2.compute.internal | ip-172-31-39-52.ap-northeast-2.compute.internal                      | -   |
| 프라이빗 리소스 DNS 이름 응답                                     | 인스턴스 유형  | AWS Compute Optimizer 찾기  |
| -  | t3.micro   | ① 권장 사항을 위해 AWS Compute Optimizer에 옵트인합니다.                      |
| 자동 할당된 IP 주소   | VPC ID   | 자세히 알아보기  |
| 13.125.132.56 [퍼블릭 IP]                                 | vpc-0575fc7da26b3e404  | -   |
| IAM 역할   | 서브넷 ID   | Auto Scaling 그룹 이름  |
| -  | subnet-0ac81a58c52ade18d   | -   |
| IMDSv2   | 인스턴스 ARN   | 관리형   |
| Required   | arn:aws:ec2:ap-northeast-2:985090322396:instance/i-0215122488ed3bd5c | false   |
| 연산자  |  |   |

세부 정보 | 상태 및 경보 | 모니터링 | 보안 | 네트워킹 | 스토리지 | 태그

▼ 인스턴스 세부 정보 정보

|  |         |            |
|--|---------|------------|
| AMI ID   | 모니터링    | 플랫폼 세부 정보  |
| ami-040c33c6a51fd5d96  | 비활성     | Linux/UNIX |
| AMI 이름   | 허용된 이미지 | 종료 방지      |
| ubuntu/images/hvm-ssd-gp3/ubuntu-noble-24.04-amd64-server-20240927 | -       | 비활성        |

- 접속하기

```
C:\Users\jeffn\OneDrive\OneDrive\GitHub\GitHub\miniProject\terraform>ssh -i "roulette-key.pem" ubuntu@13.124.78.95
The authenticity of host '13.124.78.95 (13.124.78.95)' can't be established.
ED25519 key fingerprint is SHA256:Kw2W7H/OUoXEFoJ4STgd8t+WeYE80TpPe2AYpJw0s1M.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '13.124.78.95' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1016-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Thu Jan 29 03:11:01 UTC 2026

System load: 0.0 Temperature: -273.1 C
Usage of /: 22.8% of 6.71GB Processes: 111
Memory usage: 22% Users logged in: 0
Swap usage: 0% IPv4 address for ens5: 172.31.44.97

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
```

- 테라폼 초기 설정( 작성된 코드가 있다면)

**1-3. 인프라 배포 (실제로 서버 띄우기)**

파일 저장이 끝났다면 터미널에서 `terraform` 폴더 위치 그대로 아래 명령어들을 순서대로 입력해 보세요.

1. 초기화 (도구 준비):

```
DOS
terraform init
```

2. 계획 확인 (미리보기):

```
DOS
terraform plan
```

- 여기서 "1 to add"라는 메시지가 나오면 정상입니다!

3. 실행 (진짜 생성):

```
DOS
terraform apply
```

- 중간에 `yes` 를 입력하라고 나오면 직접 타이핑하고 엔터를 치세요.

- 배포 성공한 모습(추후에 인프라를 더 확장한다면 위 과정을 반복)

```
> > ▾ 터미널
%  +
+ root_block_device (known after apply)
}

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_instance.roulette_server: Creating...
aws_instance.roulette_server: Still creating... [00m10s elapsed]
aws_instance.roulette_server: Creation complete after 13s [id=i-0215122488ed3bd5c]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\jeffn\OneDrive\OneDrive\GitHub\GitHub\miniProject\terraform>
```

- GitHub Secrets 설정
  - DOCKERHUB\_USERNAME
    - 도커 유저 이름

- DOCKERHUB\_TOKEN
  - 도커 토큰
- SERVER\_IP
  - 테라폼으로 생성한 AWS EC2 서버의 주소
- SSH\_PRIVATE\_KEY
  - 실물 열쇠(.pem 파일)의 내용

The screenshot shows the GitHub Actions settings page for a repository named JUNHO0712. The left sidebar has a tree view of repository settings, with 'Secrets and variables' selected under 'Actions'. The main content area is titled 'Actions secrets and variables'.

**Environment secrets:** A box states 'This environment has no secrets.' with a 'Manage environment secrets' button.

**Repository secrets:** A table lists four secrets:

| Name               | Last updated | Action |
|--------------------|--------------|--------|
| DOCKERHUB_TOKEN    | yesterday    |        |
| DOCKERHUB_USERNAME | yesterday    |        |
| SERVER_IP          | yesterday    |        |
| SSH_PRIVATE_KEY    | yesterday    |        |

A green 'New repository secret' button is located at the top right of the secrets table.

- 인스턴스 인바운드 설정 확인

The screenshot shows the AWS CloudWatch Metrics console for the instance **i-0e8fd691e514155db**. The **보안** (Security) tab is selected. Under **보안 세부 정보**, it shows the following:

- IAM 역할**: -
  -
- 소유자 ID**: 985090322396
- 시작 시간**: Thu Jan 29 2026 12:06:10 GMT+0900 (한국 표준시)

Under **인바운드 규칙**, there are three entries:

| 이름 | 보안 그룹 규칙 ID           | 포트 범위 | 프로토콜 | 원본        |
|----|-----------------------|-------|------|-----------|
| -  | sgr-034c854cd434b083  | 3000  | TCP  | 0.0.0.0/0 |
| -  | sgr-0fc9c7c67e9996ec8 | 80    | TCP  | 0.0.0.0/0 |
| -  | sgr-0cc314acdbd7afeef | 22    | TCP  | 0.0.0.0/0 |

Under **아웃바운드 규칙**, there are no entries.

- .github 폴더 생성 후 workflows로 로그 확인
    - 문제 발생시
      - 터미널에서 인스턴스 ip로 ssh 접속 후 도커 로그 확인
- ```
$ sudo docker logs roulette-app
```

The screenshot shows the GitHub Actions page for the repository **JUNHO0712**. The **Actions** tab is selected. A message at the top says "Workflow run deleted successfully.".

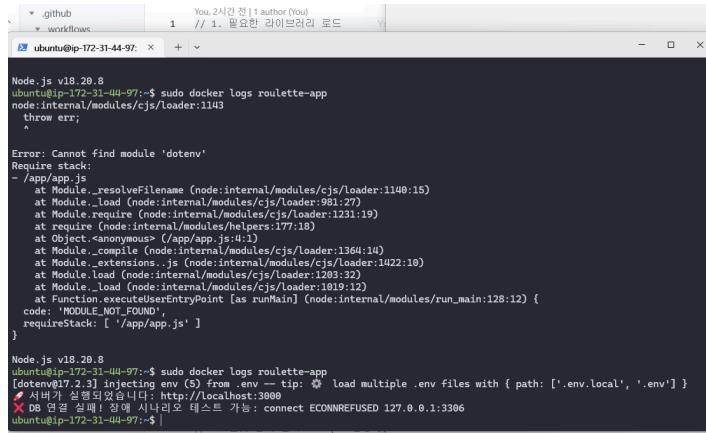
The main area shows the **All workflows** section, which lists one workflow run:

| Event                                                 | Status | Branch        | Actor |
|-------------------------------------------------------|--------|---------------|-------|
| Step 3&4: Add Dockerfile and update Ansible for de... | main   | 3 minutes ago | ...   |

Details for the workflow run show the commit information: "Manual Deploy #2: Commit ebb7c44 pushed by JUNHO0712".

- 접속 확인

- 인스턴스 ip 주소로 웹페이지 접속
- 터미널 도커 로그 확인



Node.js v18.20.8  
ubuntu@ip-172-31-44-97:~\$ sudo docker logs roulette-app  
node:internal/modules/cjs/loader:1140:15  
throw err;  
  
Error: Cannot find module 'dotenv'  
Require stack:  
- /app/app.js  
at Module.\_resolveFilename (node:internal/modules/cjs/loader:1140:15)  
at Module.\_load (node:internal/modules/cjs/loader:981:27)  
at Module.require (node:internal/modules/cjs/loader:1231:19)  
at require (node:internal/modules/helpers:177:18)  
at Object.<anonymous> (/app/app.js:4:11)  
at Module.\_compile (node:internal/modules/cjs/loader:1364:14)  
at Module.\_extensions..js (node:internal/modules/cjs/loader:1422:10)  
at Module.load (node:internal/modules/cjs/loader:1203:32)  
at Module.\_load (node:internal/modules/cjs/loader:1019:12)  
at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run\_main:128:12) {  
 code: 'MODULE\_NOT\_FOUND'  
}  
requireStack: [ '/app/app.js' ]  
  
Node.js v18.20.8  
ubuntu@ip-172-31-44-97:~\$ sudo docker logs roulette-app  
[dotenv] 2.31 injecting env (\$).env tip: ⚡ load multiple .env files with { path: ['.env.local', '.env'] }  
⚡ 서버가 실행되었습니다: http://localhost:3000  
✖ DB 연결 실패! 철에 시나리오 테스트 가능: connect ECONNREFUSED 127.0.0.1:3306  
ubuntu@ip-172-31-44-97:~\$

점심 추천 룰렛



버튼을 눌러주세요!