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 개방하는 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

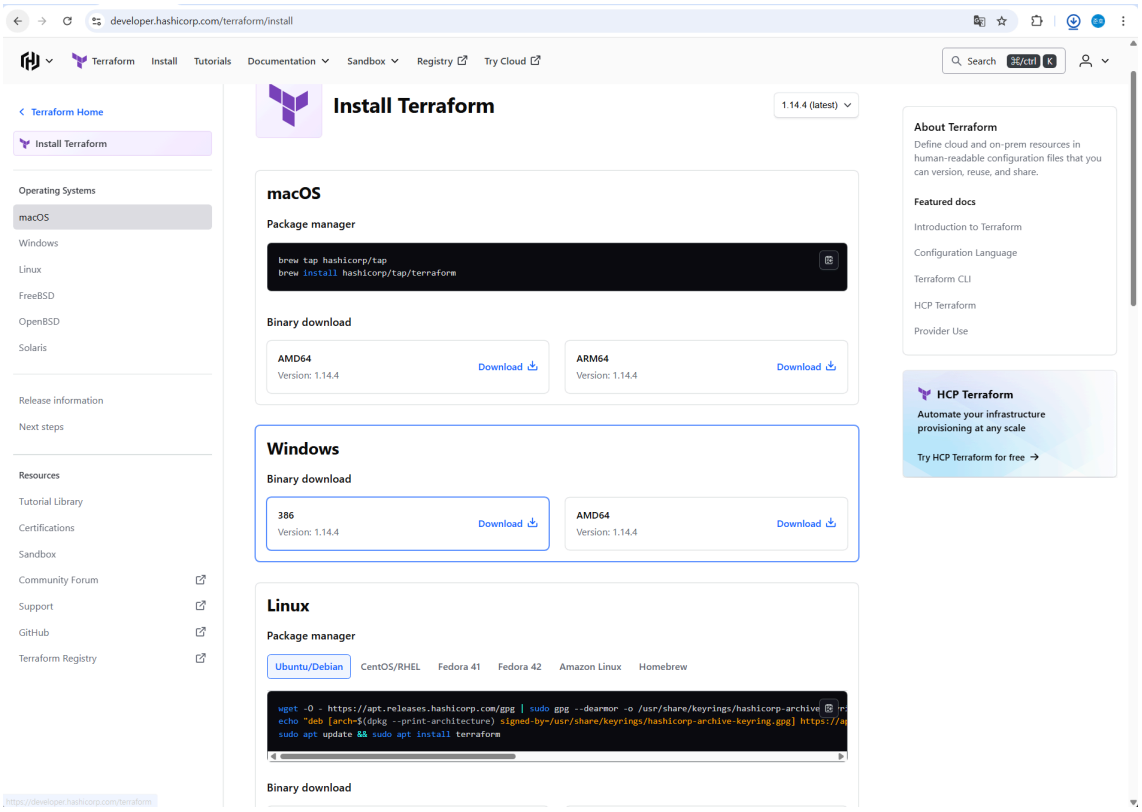
작성한 코드를 `git push` 하면 깃허브 액션이 아래 과정을 자동으로 수행합니다.

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.

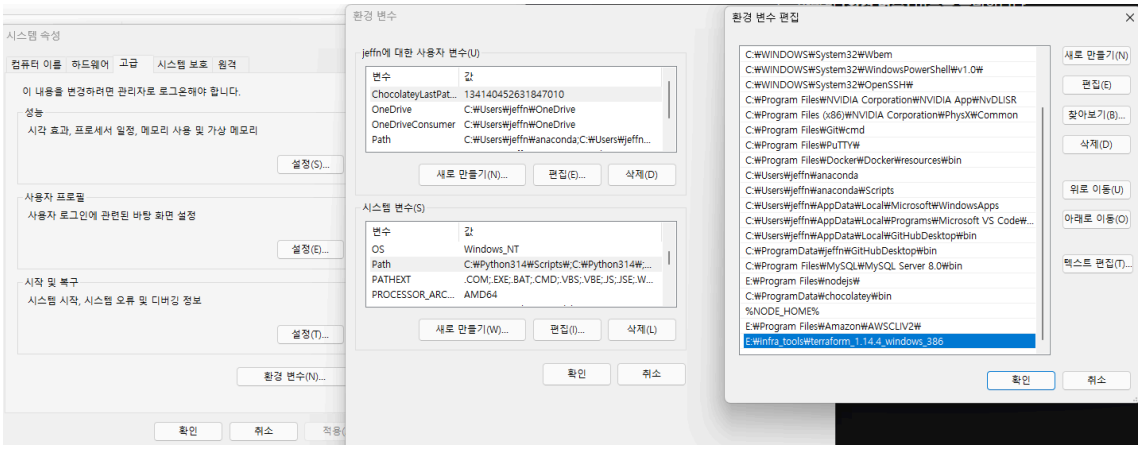
세부 과정

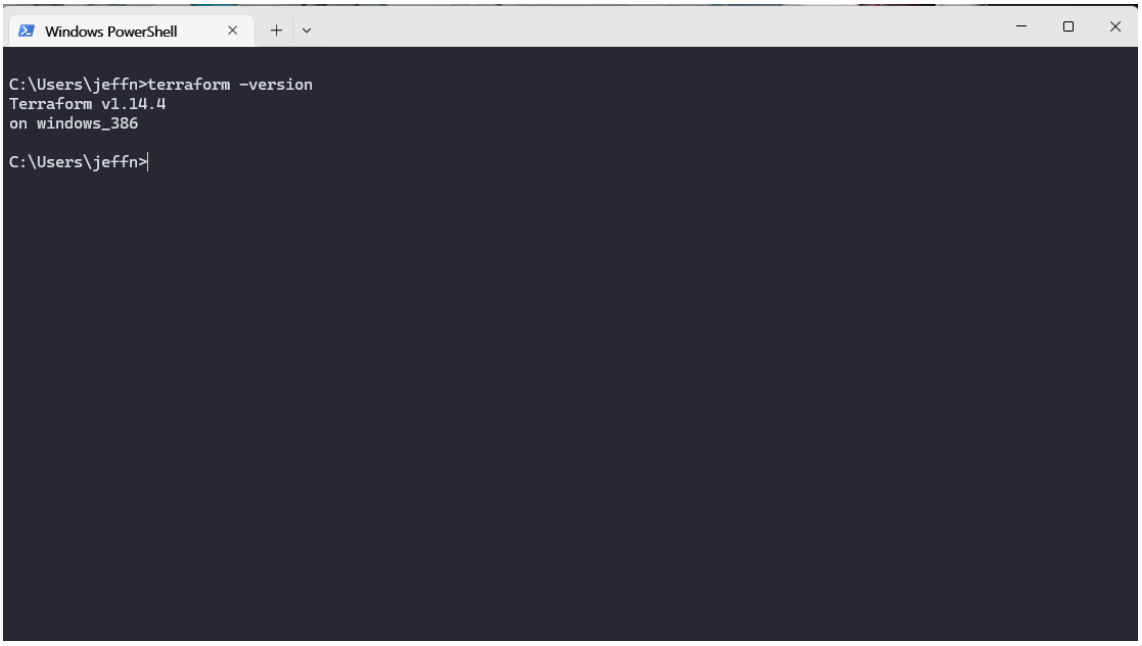
- 설치

Terraform

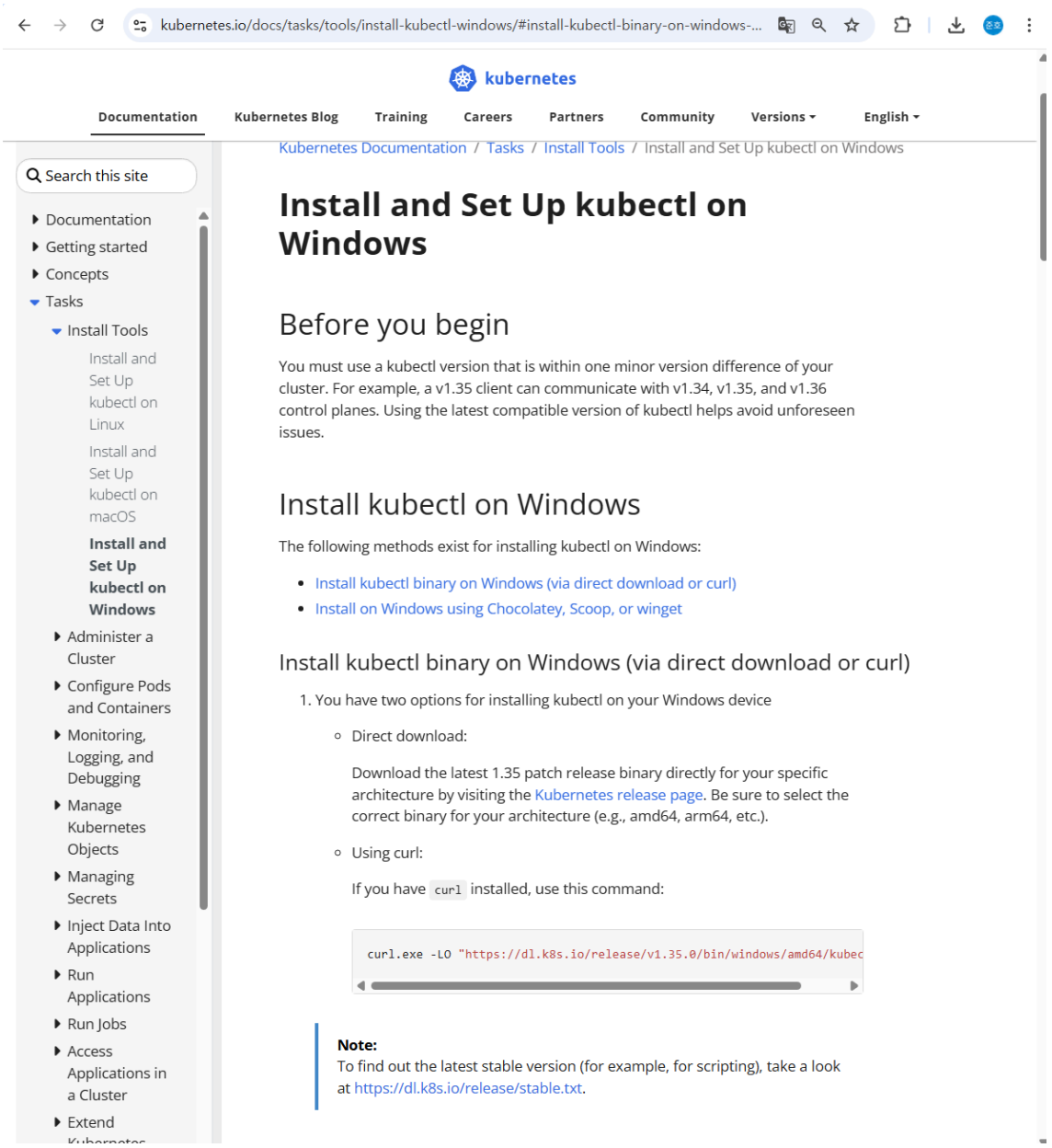

developer.hashicorp.com/terraform/install
Terraform
Install
Tutorials
Documentation
Sandbox
Registry
Try Cloud
Search
Terraform Home
Install Terraform
Operating Systems
macOS
Windows
Linux
FreeBSD
OpenBSD
Solaris
Release information
Next steps
Resources
Tutorial Library
Certifications
Sandbox
Community Forum
Support
GitHub
Terraform Registry
Install Terraform
1.14.4 (latest)
macOS
Package manager
brew tap hashicorp/tap
brew install hashicorp/tap/terraform
Binary download
AMD64
Version: 1.14.4
Download
ARM64
Version: 1.14.4
Download
Windows
Binary download
386
Version: 1.14.4
Download
AMD64
Version: 1.14.4
Download
Linux
Package manager
Ubuntu/Debian
CentOS/RHEL
Fedora 41
Fedora 42
Amazon Linux
Homebrew
sudo apt update && sudo apt install terraform
Binary download
About Terraform
Define cloud and on-prem resources in human-readable configuration files that you can version, reuse, and share.
Featured docs
Introduction to Terraform
Configuration Language
Terraform CLI
HCP Terraform
Provider Use
HCP Terraform
Automate your infrastructure provisioning at any scale
Try HCP Terraform for free

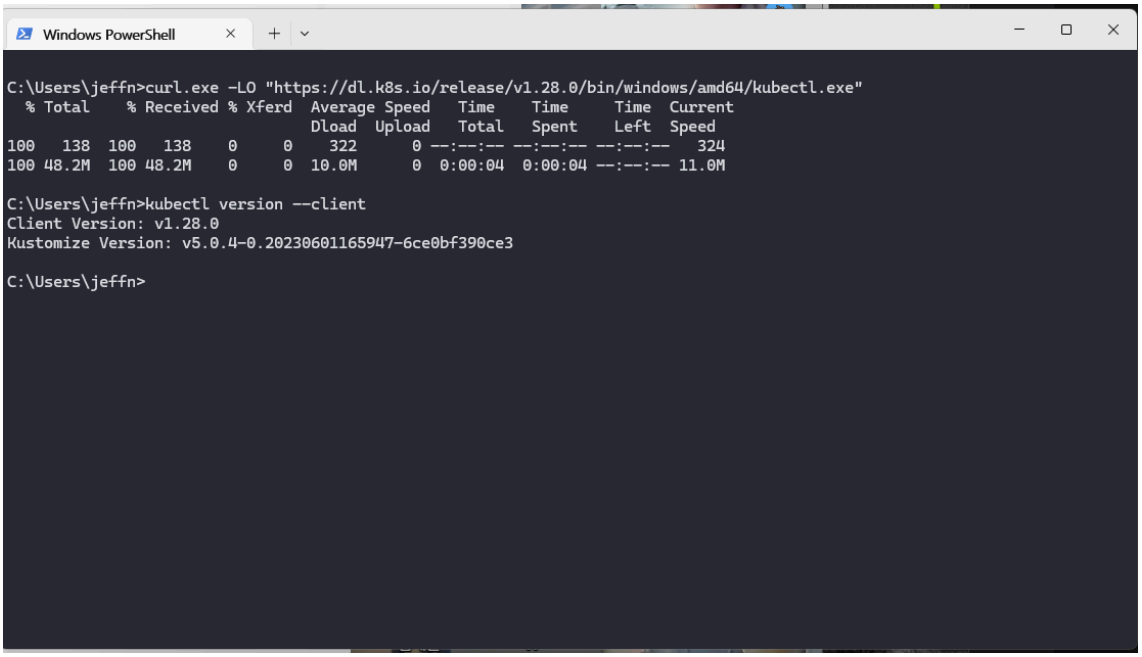
경로 설정


시스템 속성
환경 변수
환경 변수 편집
jeffn에 대한 사용자 변수(U)
시스템 변수(S)
확인
취소

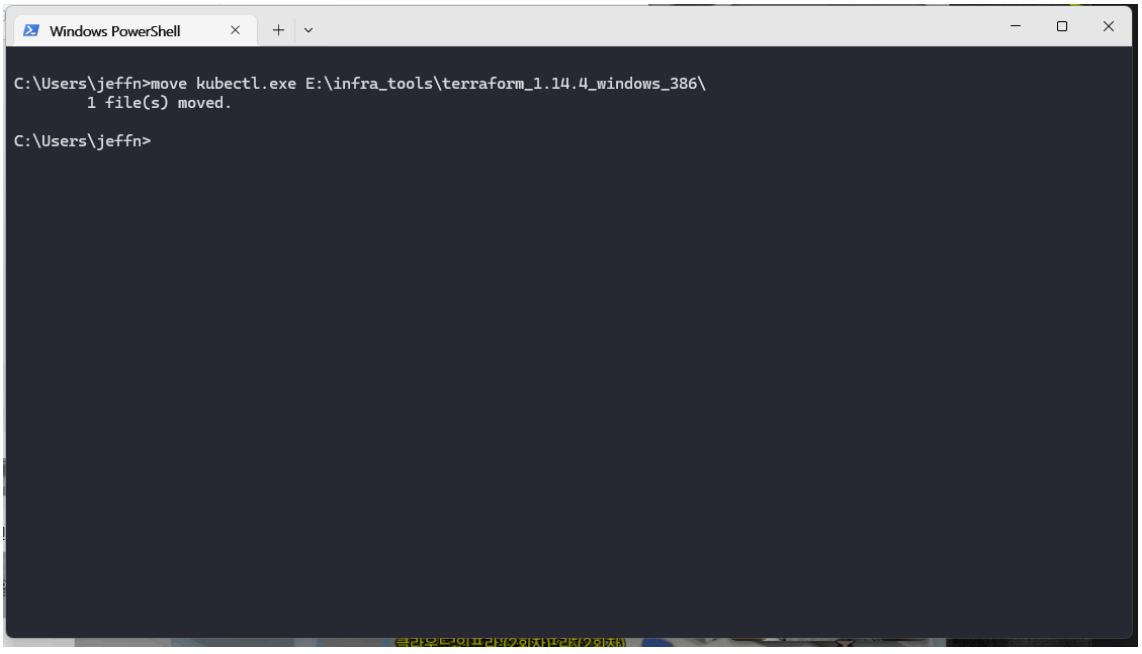

Windows PowerShell
C:\Users\jeffn>terraform -version
Terraform v1.14.4
on windows_386
C:\Users\jeffn>

Kubernetes

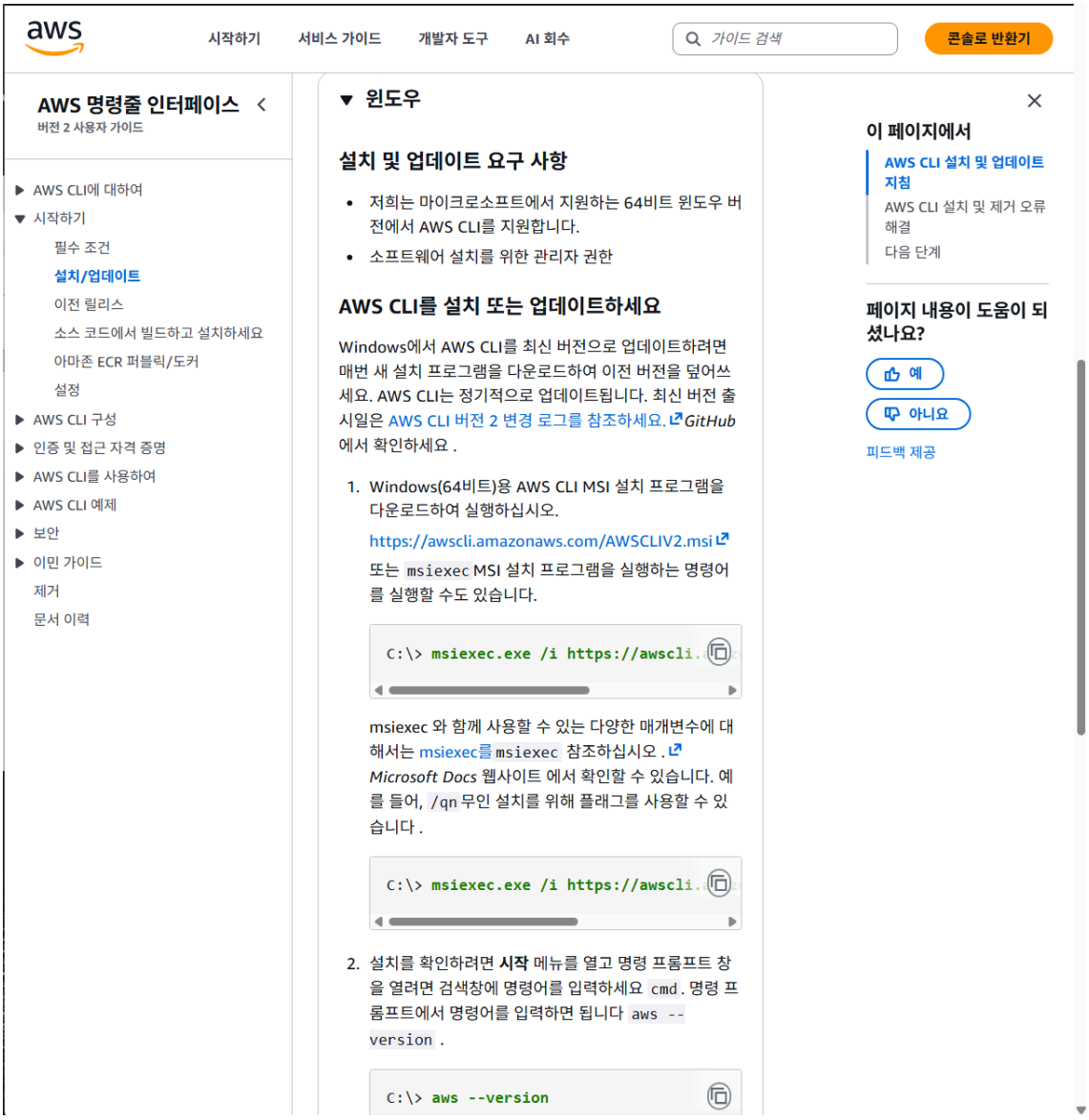

kubernetes.io/docs/tasks/tools/install-kubectl-windows/#install-kubectl-binary-on-windows-...
kubernetes
Documentation
Kubernetes Blog
Training
Careers
Partners
Community
Versions
English
Search this site
Documentation
Getting started
Concepts
Tasks
Install Tools
Install and Set Up kubectl on Linux
Install 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 Documentation / Tasks / Install Tools / Install and Set Up kubectl on Windows
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.

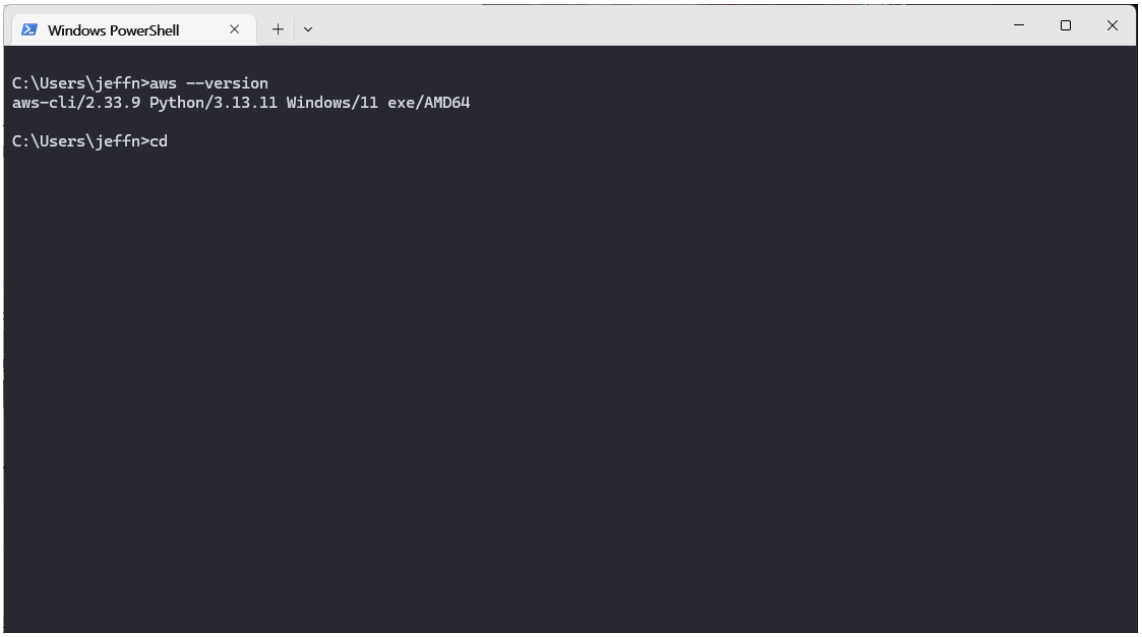


위에 테라폼 경로에 같이 넣어두기

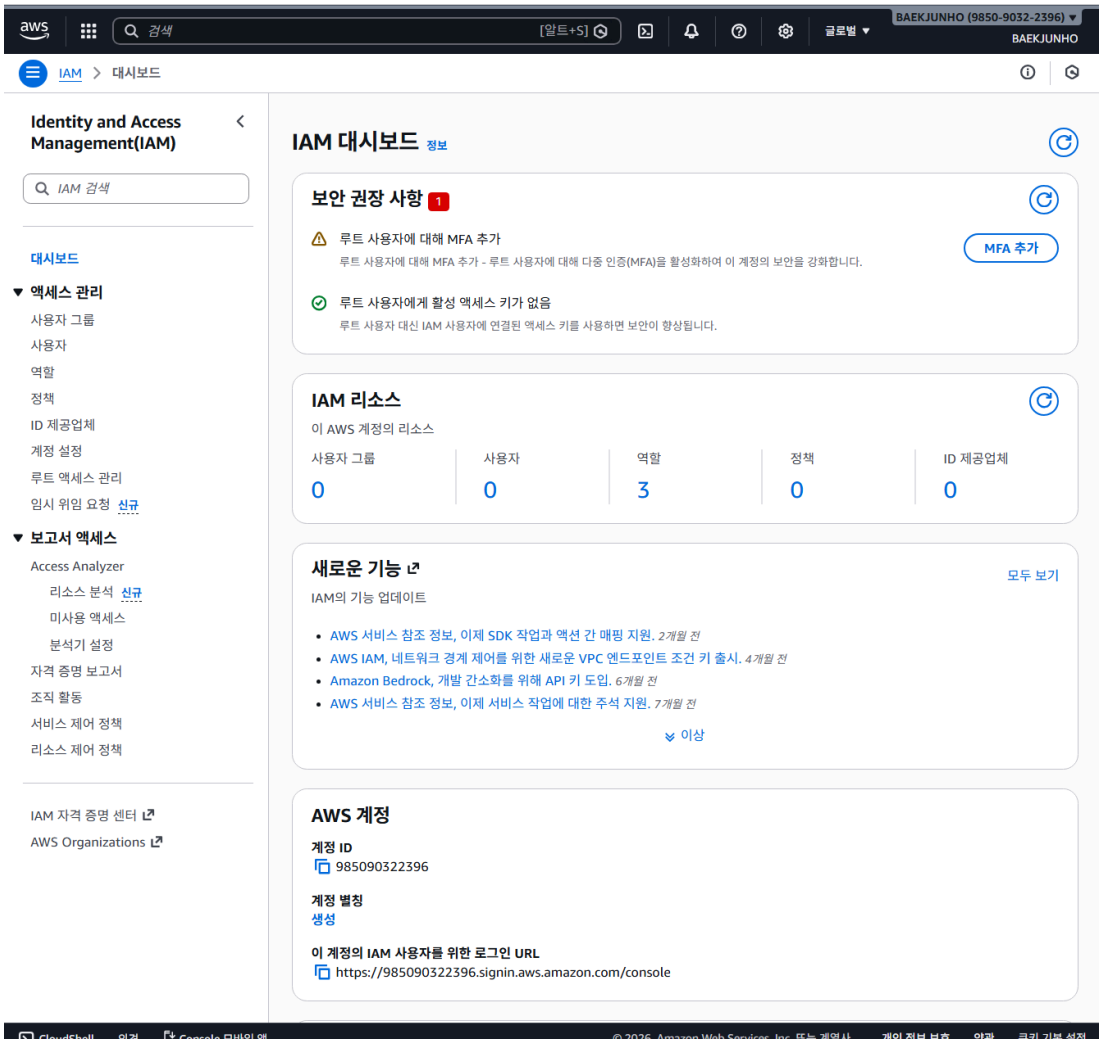


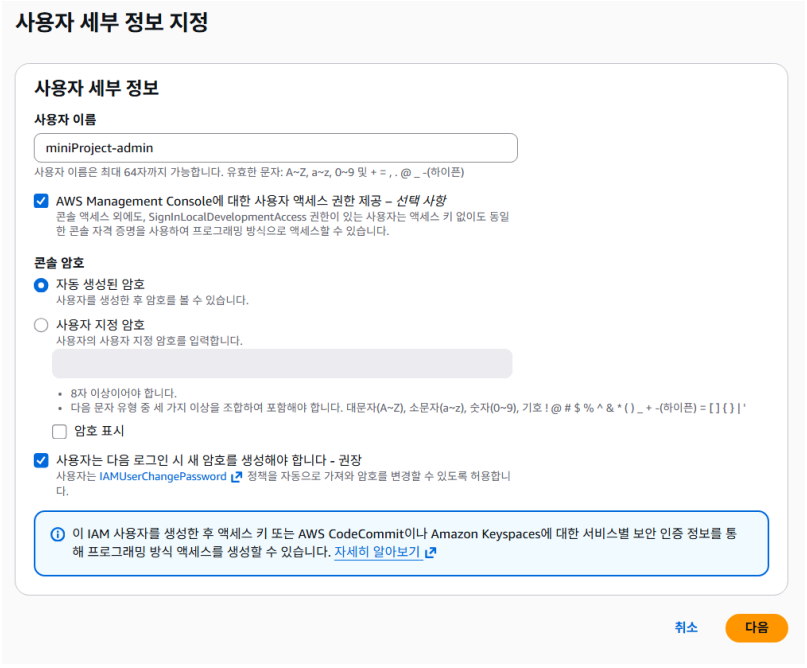
AWS CLI


aws
시작하기
서비스 가이드
개발자 도구
AI 회수
가이드 검색
콘솔로 반환기
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. GitHub 에서 확인하세요 .
1. Windows(64비트)용 AWS CLI MSI 설치 프로그램을 다운로드하여 실행하십시오.
https://awscli.amazonaws.com/AWSCLIV2.msi
또는 msiexec MSI 설치 프로그램을 실행하는 명령어를 실행할 수도 있습니다.
C:\> msiexec.exe /i https://awscli.
msiexec 와 함께 사용할 수 있는 다양한 매개변수에 대해서는 msiexec를 msiexec 참조하십시오 . Microsoft Docs 웹사이트 에서 확인할 수 있습니다. 예를 들어, /qn 무인 설치를 위해 플래그를 사용할 수 있습니다 .
C:\> msiexec.exe /i https://awscli.
2. 설치를 확인하려면 시작 메뉴를 열고 명령 프롬프트 창을 열려면 검색창에 명령어를 입력하세요 cmd . 명령 프롬프트에서 명령어를 입력하면 됩니다 aws --version .
C:\> aws --version
이 페이지에서
AWS CLI 설치 및 업데이트 지침
AWS CLI 설치 및 제거 오류 해결
다음 단계
페이지 내용이 도움이 되셨나요?
예
아니요
피드백 제공

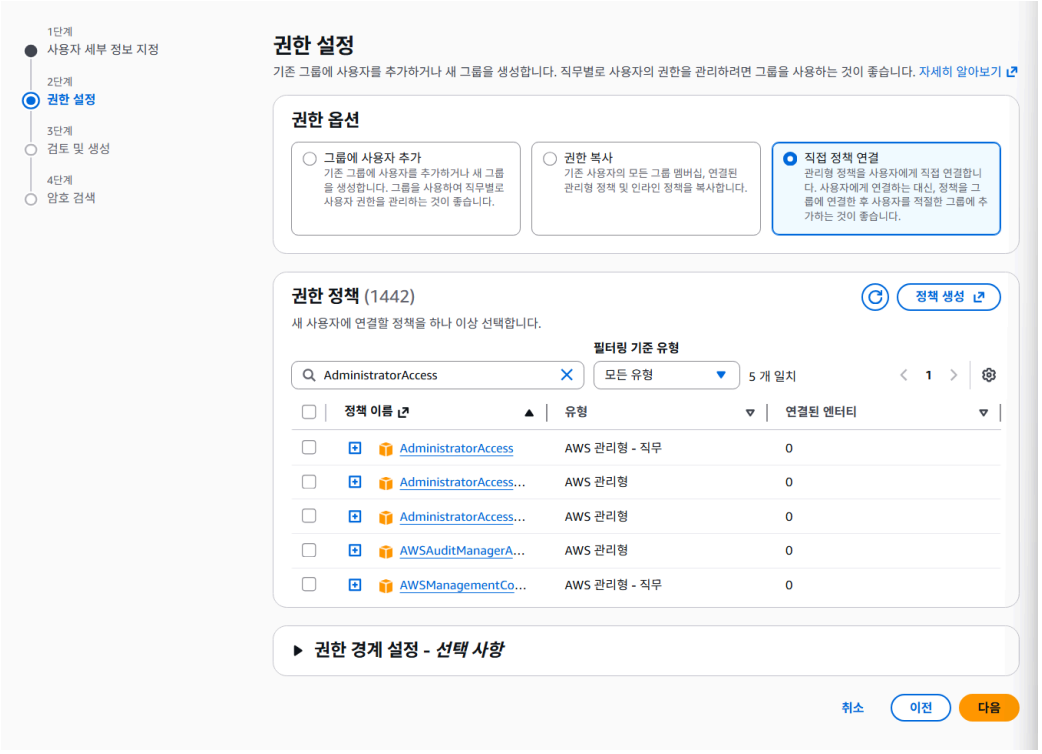

Windows PowerShell
C:\Users\jeffn>aws --version
aws-cli/2.33.9 Python/3.13.11 Windows/11 exe/AMD64
C:\Users\jeffn>cd

- IAM 계정 생성 방법

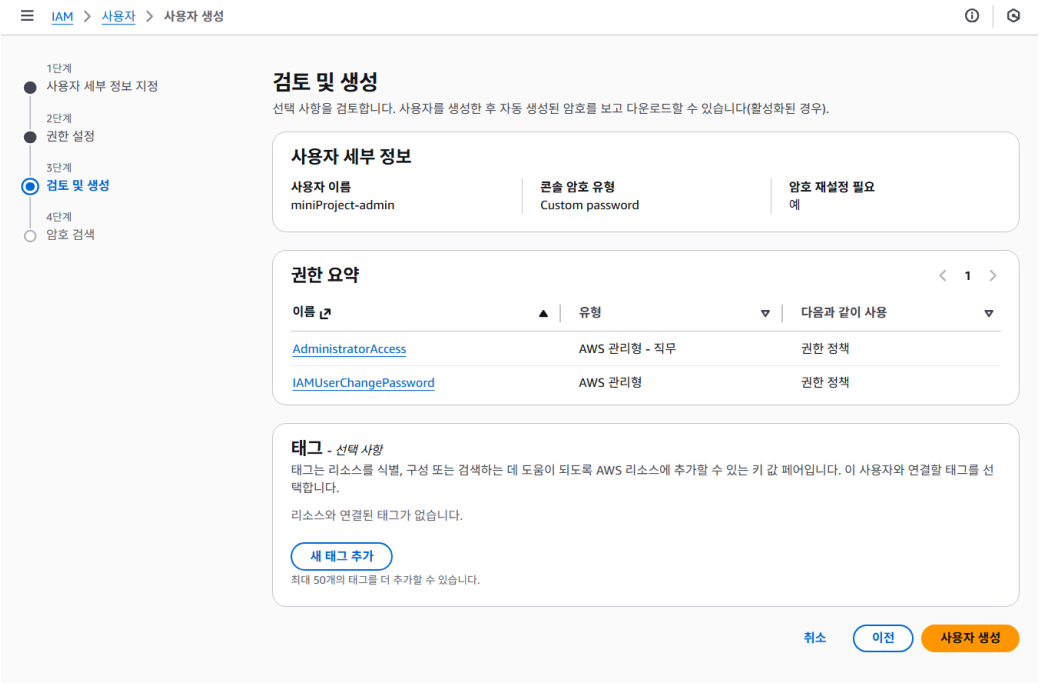

aws
검색
[알트+S]
글로벌
BAEKJUNHO (9850-9032-2396)
BAEKJUNHO
IAM > 대시보드
Identity and Access Management(IAM)
IAM 검색
대시보드
액세스 관리
사용자 그룹
사용자
역할
정책
ID 제공업체
계정 설정
루트 액세스 관리
임시 위임 요청 신규
보고서 액세스
Access Analyzer
리소스 분석 신규
미사용 액세스
분석기 설정
자격 증명 보고서
조직 활동
서비스 제어 정책
리소스 제어 정책
IAM 자격 증명 센터
AWS Organizations
IAM 대시보드 정보
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.
MFA 추가
루트 사용자에게 활성 액세스 키가 없음
루트 사용자 대신 IAM 사용자에 연결된 액세스 키를 사용하면 보안이 향상됩니다.
IAM 리소스
이 AWS 계정의 리소스
사용자 그룹 0
사용자 0
역할 3
정책 0
ID 제공업체 0
새로운 기능
모두 보기
IAM의 기능 업데이트
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


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.
8자 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
취소
다음

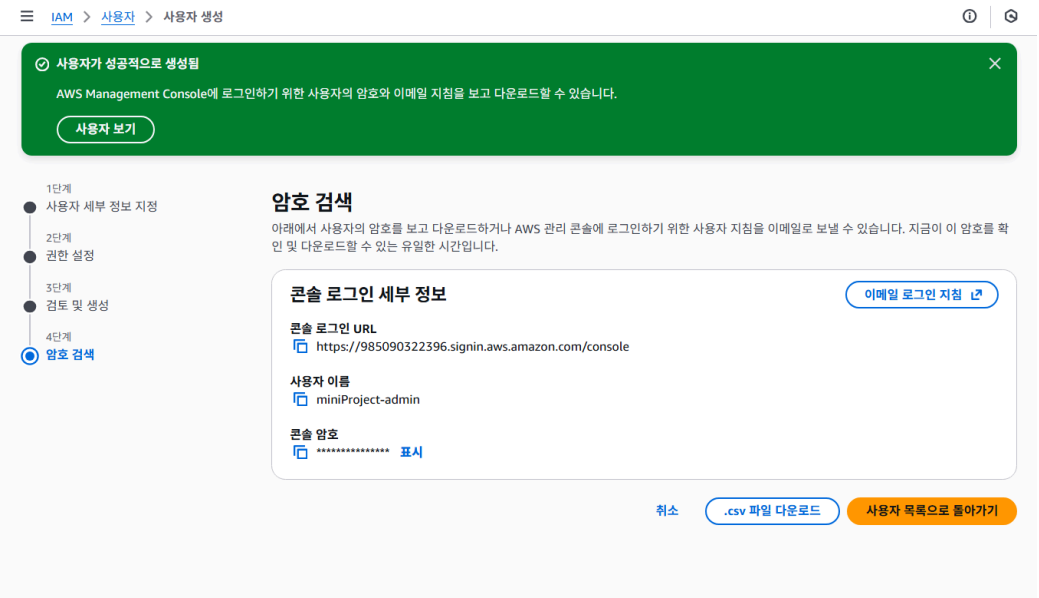
- 권한 정책에 AdministratorAccess 검색 후 가장 위 정책 선택



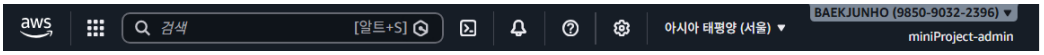
- 사용자 생성



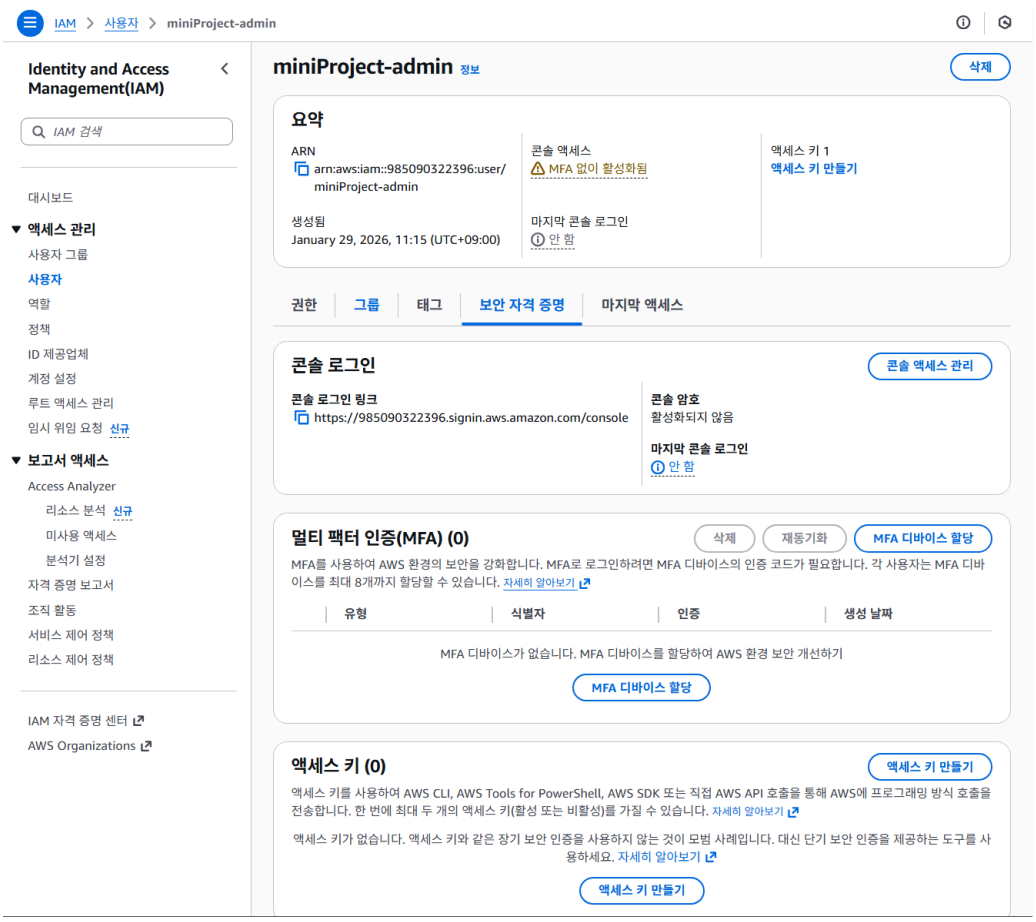
- 세부 정보는 꼭 따로 저장



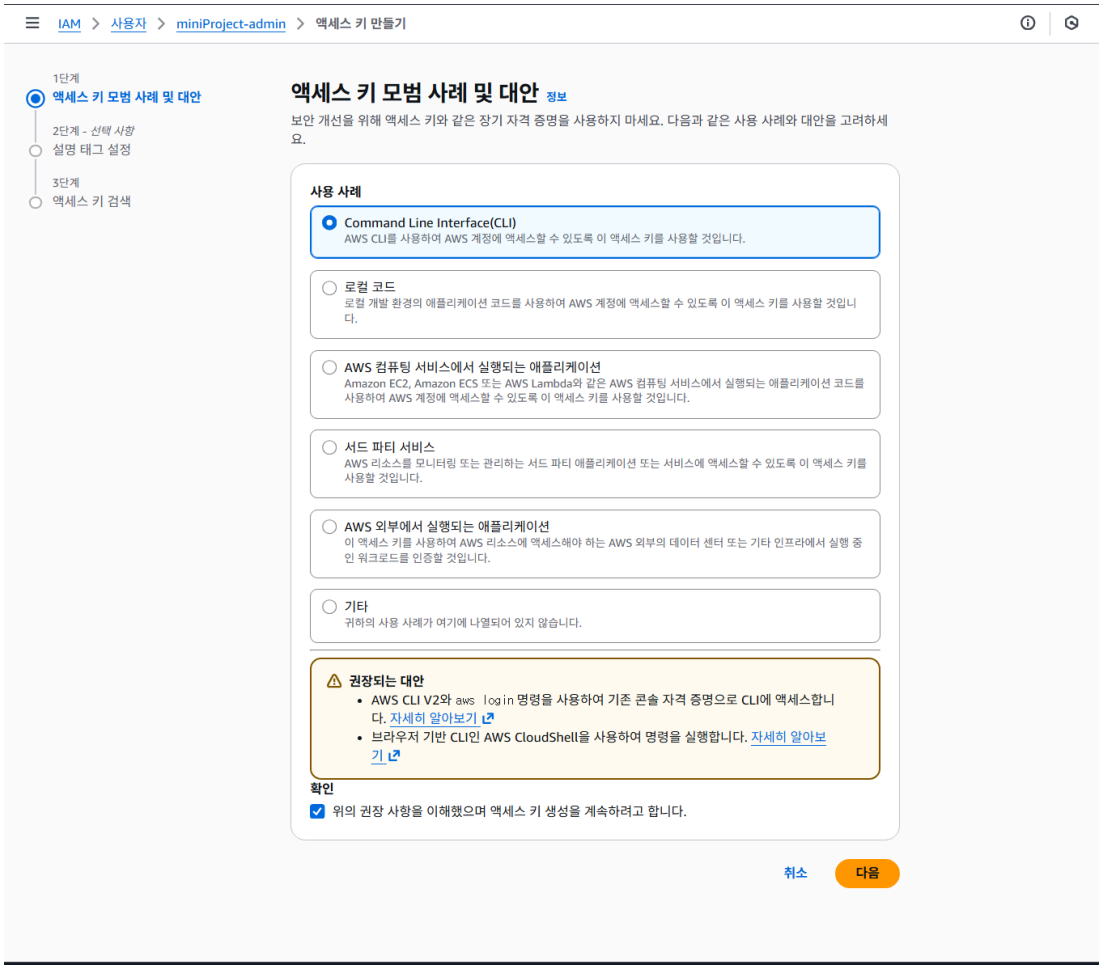
- 액세스 키 만들기(반드시 리전 확인하기)
  - 리전 - 만드는 위치



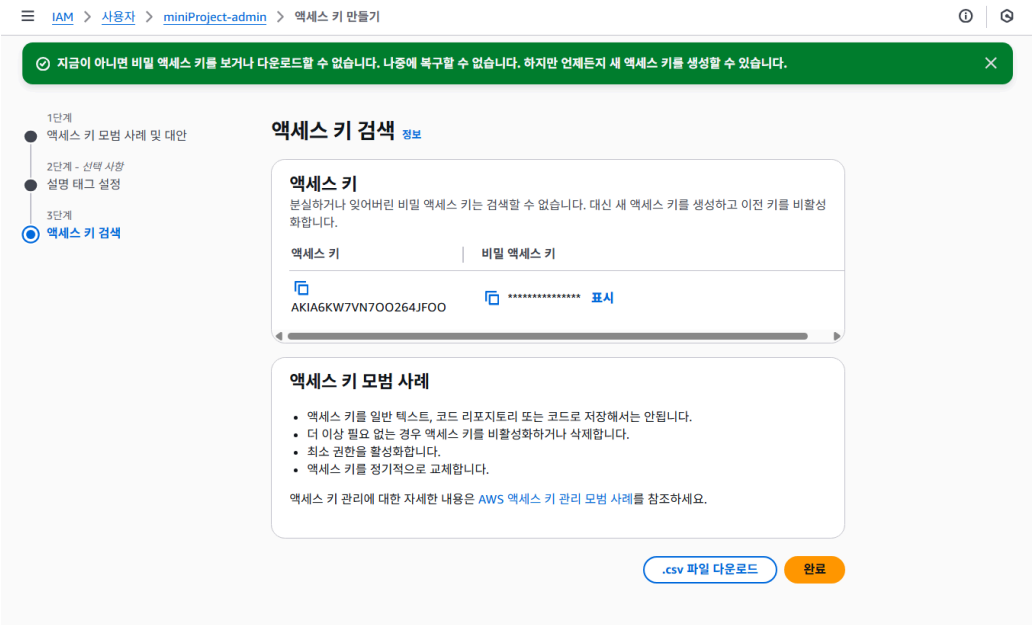
아시아 태평양(서울)



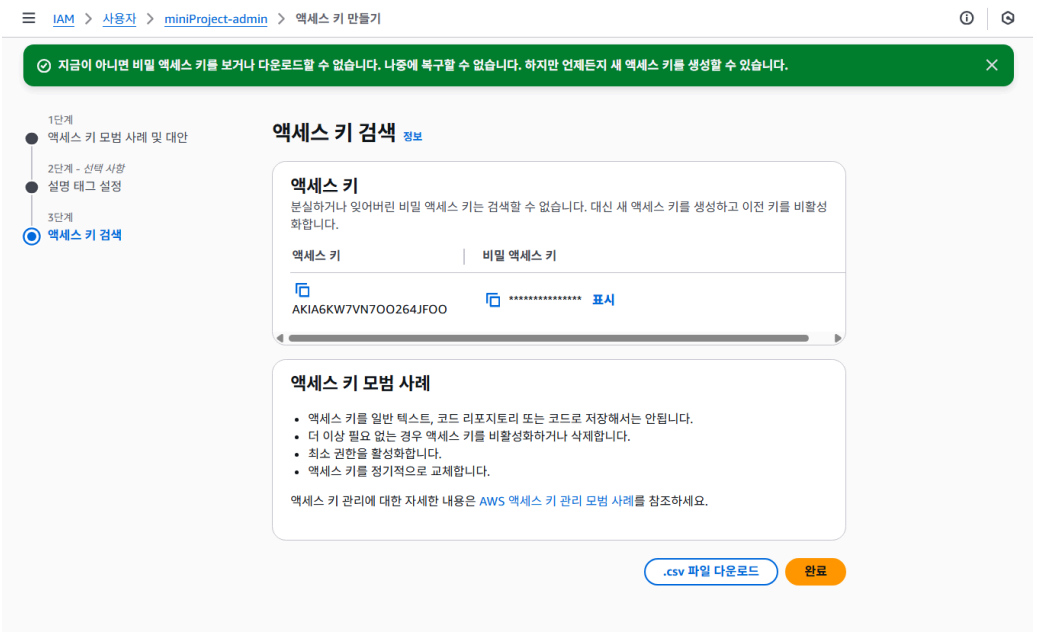
- **테라폼(Terraform) 연결**: 사용자님의 VS Code에서 테라폼 명령을 내릴 때, AWS가 "너 누구야?"라고 물으면 이 액세스 키를 보여주어 본인임을 인증하고 인프라(EC2 등)를 생성하게 합니다.
- **GitHub Actions 연결**: 깃허브 액션 서버가 사용자님 대신 AWS EC2에 접속하거나 이미지를 다룰 때 사용할 '디지털 열쇠' 역할을 합니다.



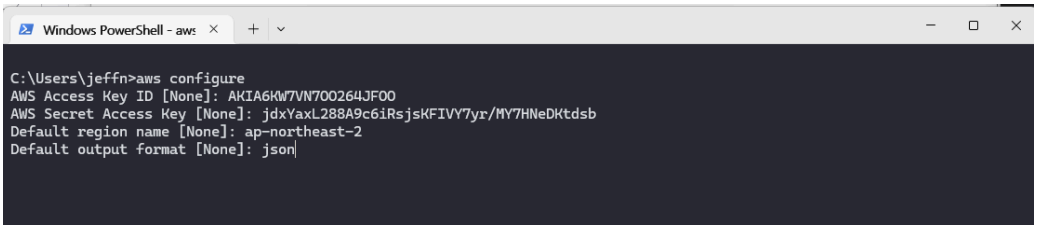
- 액세스 키도 반드시 따로 저장 + .csv 파일 다운로드



- AWS 자격 증명 등록



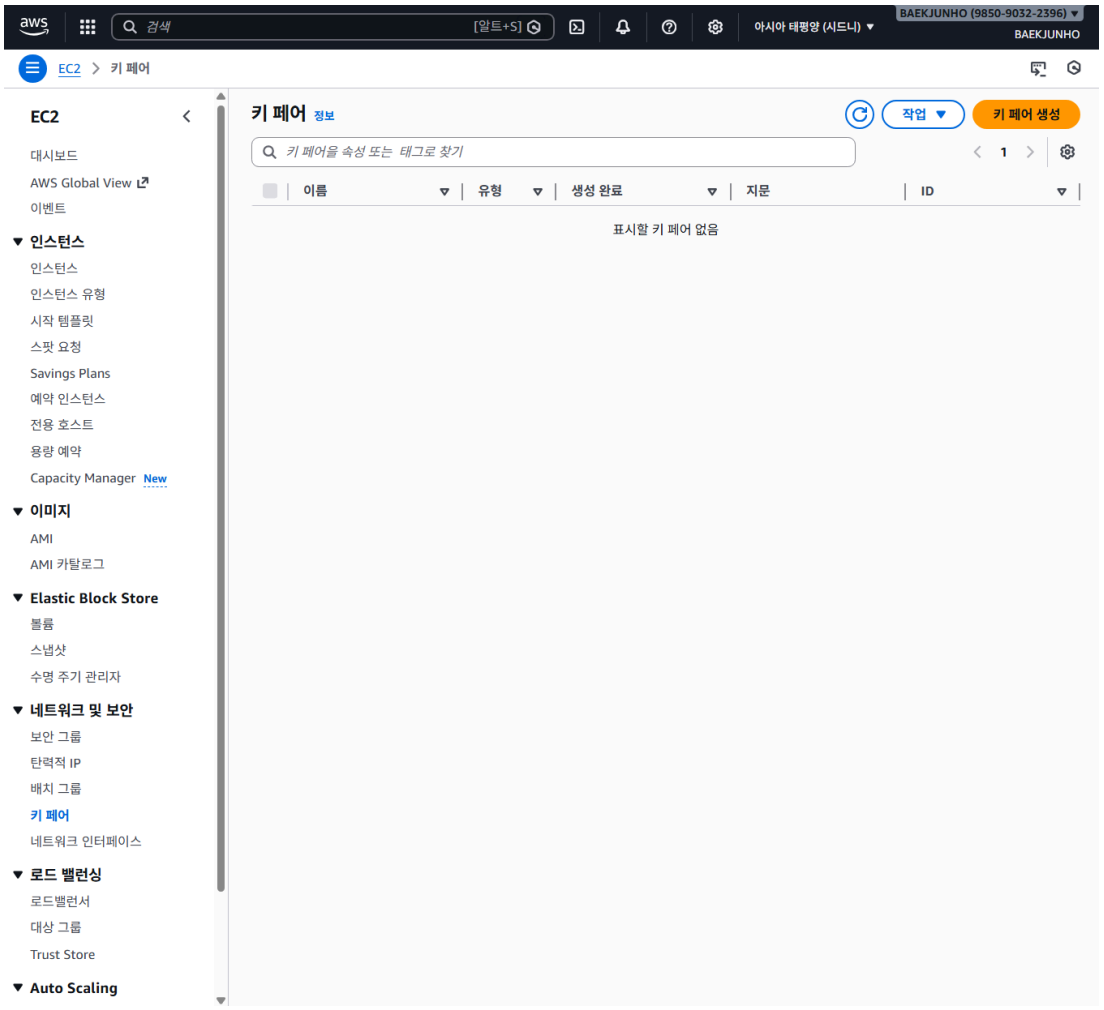
- 리전 확인
  - 없으면 리전 확인 하고 다시 만들어야 함



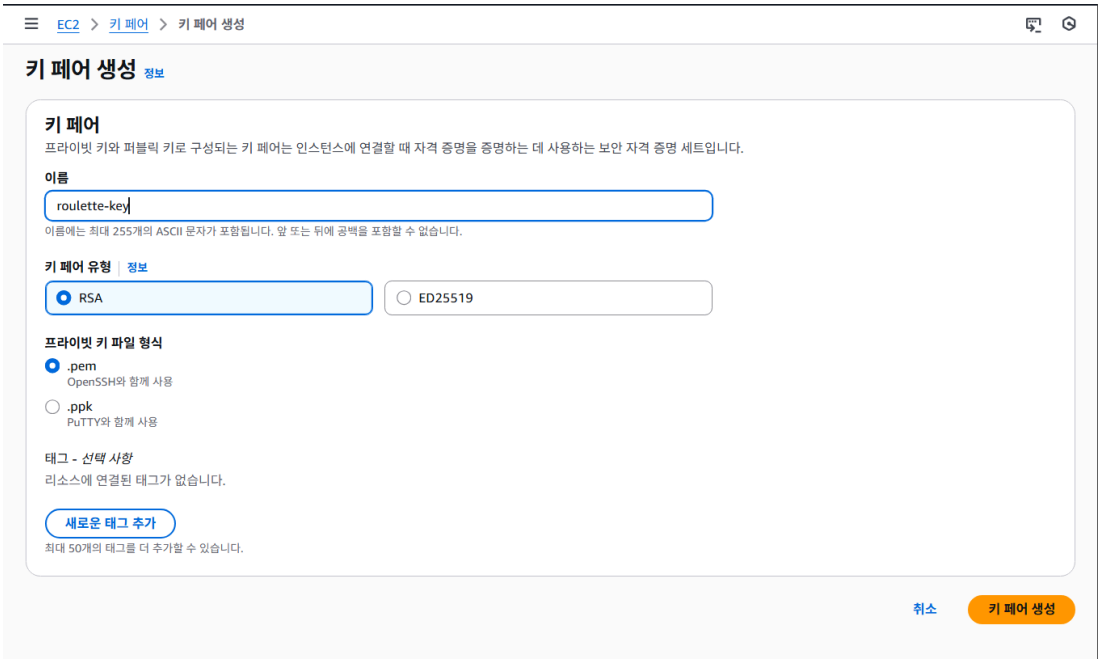
- 키 페어 만들기 (서버로 들어가는 유일한 현관문 열쇠)
  - 보안을 위한 전용 열쇠 (SSH 인증)]
  - 없으면?
    - **접속 불가**: 서버가 만들어져도 그 안으로 들어가서 도커를 설치하거나 앱을 실행할 방법이 아예 없습니다.
    - **보안 취약**: 만약 비밀번호 방식을 쓴다면, 서버가 생성되자마자 전 세계 해커들의 공격 대상이 되어 금방 뚫릴 수 있습니다.

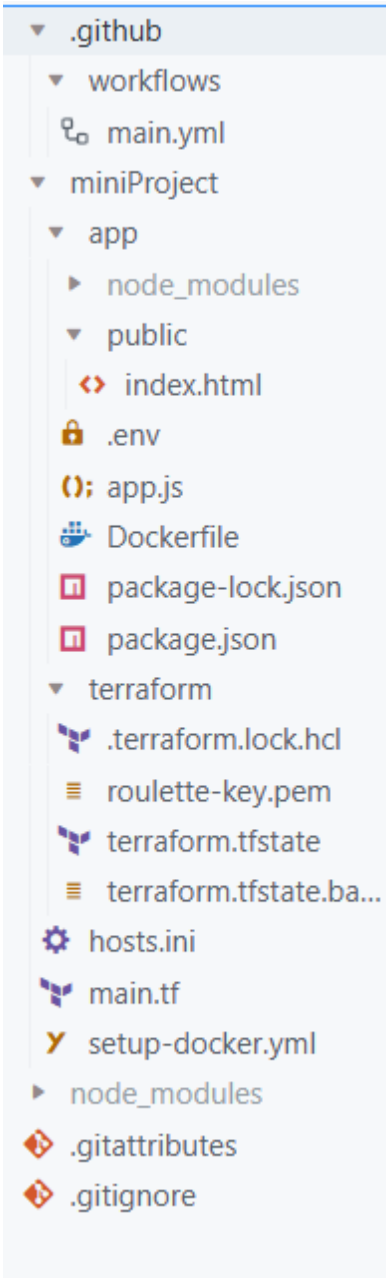
구분	액세스 키 (Access Key)	키 페어 (Key Pair)
하는 일	**테라폼**이 AWS에 접속해서 "여기 서버 한 대 지어줘"라고 요청함	**앤서블**이 지어진 서버 문을 열고 들어가 "도커 설치해"라고 명령함
비유	아파트 관리실에 가서 **입주 허가**를 받는 도장	우리 집 현관문을 여는 **실물 열쇠**

파일 형태	문자열(ID/Secret) 형태	`.pem` 파일 형태
Secrets 등록	`DOCKERHUB_TOKEN` 등과 함께 등록	`SSH_PRIVATE_KEY` 라는 이름으로 등록

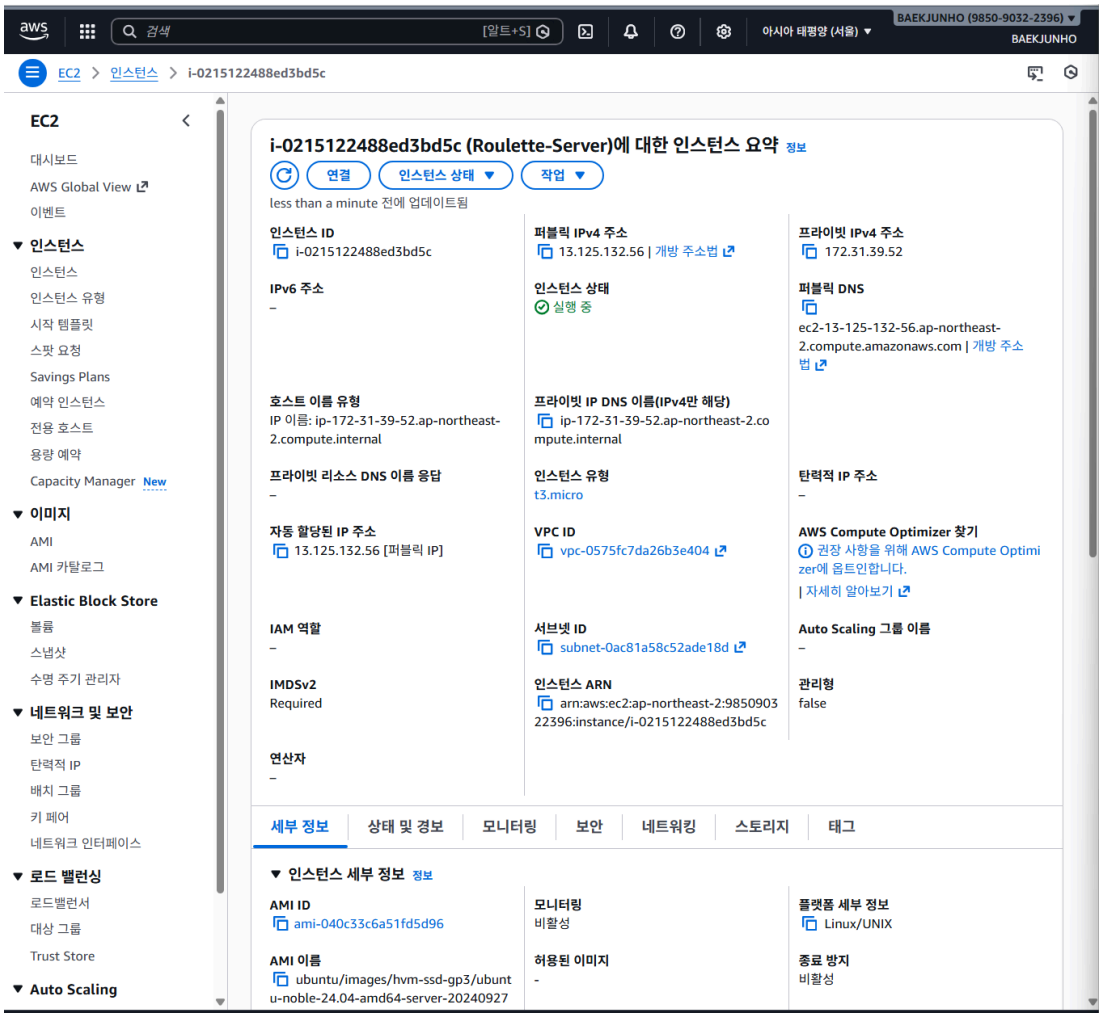


- 생성한 .pem 키는 terraform 폴더 안에 넣기

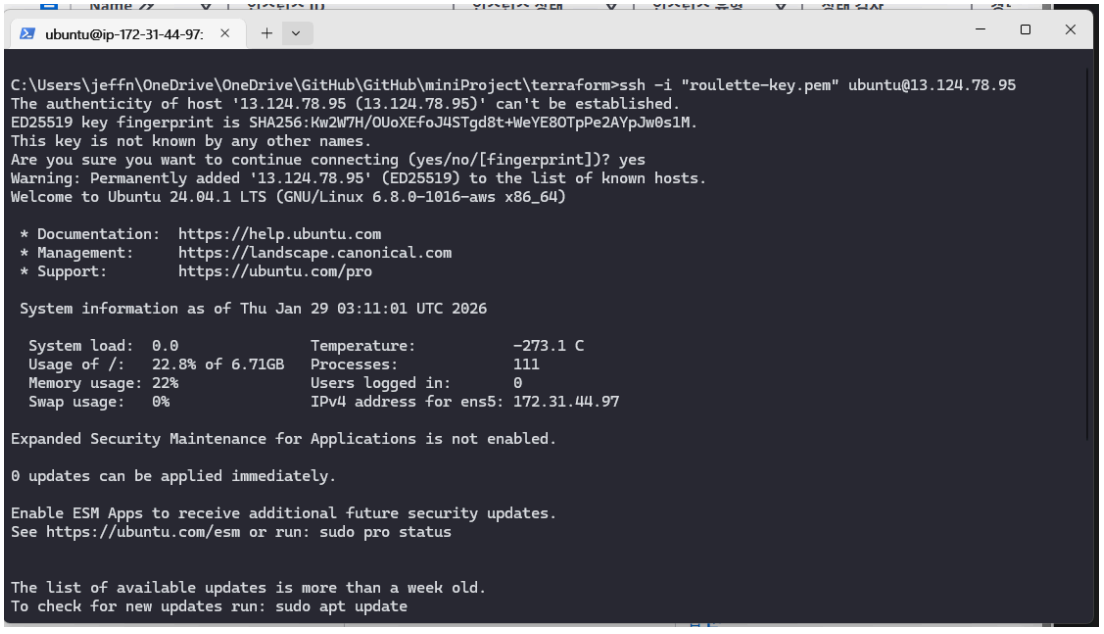

EC2 > 키 페어 > 키 페어 생성
키 페어 생성 정보
키 페어
프라이빗 키와 퍼블릭 키로 구성되는 키 페어는 인스턴스에 연결할 때 자격 증명을 증명하는 데 사용하는 보안 자격 증명 세트입니다.
이름
roulette-key
이름에는 최대 255개의 ASCII 문자가 포함됩니다. 앞 또는 뒤에 공백을 포함할 수 없습니다.
키 페어 유형 | 정보
RSA
ED25519
프라이빗 키 파일 형식
.pem
OpenSSH와 함께 사용
.ppk
PuTTY와 함께 사용
태그 - 선택 사항
리소스에 연결된 태그가 없습니다.
새로운 태그 추가
최대 50개의 태그를 더 추가할 수 있습니다.
취소
키 페어 생성


.github
workflows
main.yml
miniProject
app
node_modules
public
index.html
.env
app.js
Dockerfile
package-lock.json
package.json
terraform
.terraform.lock.hcl
roulette-key.pem
terraform.tfstate
terraform.tfstate.ba...
hosts.ini
main.tf
setup-docker.yml
node_modules
.gitattributes
.gitignore

- 인스턴스 확인


aws
검색
[알트+S]
아시아 태평양 (서울)
BAEKJUNHO (9850-9032-2396)
BAEKJUNHO
EC2 > 인스턴스 > i-0215122488ed3bd5c
EC2
대시보드
AWS Global View
이벤트
인스턴스
인스턴스
인스턴스 유형
시작 템플릿
스팟 요청
Savings Plans
예약 인스턴스
전용 호스트
용량 예약
Capacity Manager New
이미지
AMI
AMI 카탈로그
Elastic Block Store
볼륨
스냅샷
수명 주기 관리자
네트워크 및 보안
보안 그룹
탄력적 IP
배치 그룹
키 페어
네트워크 인터페이스
로드 밸런싱
로드밸런서
대상 그룹
Trust Store
Auto Scaling
i-0215122488ed3bd5c (Roulette-Server)에 대한 인스턴스 요약 정보
연결
인스턴스 상태
작업
less than a minute 전에 업데이트됨
인스턴스 ID
i-0215122488ed3bd5c
퍼블릭 IPv4 주소
13.125.132.56 | 개방 주소법
프라이빗 IPv4 주소
172.31.39.52
IPv6 주소
–
인스턴스 상태
실행 중
퍼블릭 DNS
ec2-13-125-132-56.ap-northeast-2.compute.amazonaws.com | 개방 주소법
호스트 이름 유형
IP 이름: ip-172-31-39-52.ap-northeast-2.compute.internal
프라이빗 IP DNS 이름(IPv4만 해당)
ip-172-31-39-52.ap-northeast-2.compute.internal
프라이빗 리소스 DNS 이름 응답
–
인스턴스 유형
t3.micro
탄력적 IP 주소
–
자동 할당된 IP 주소
13.125.132.56 [퍼블릭 IP]
VPC ID
vpc-0575fc7da26b3e404
AWS Compute Optimizer 찾기
권장 사항을 위해 AWS Compute Optimizer에 옵트인합니다.
| 자세히 알아보기
IAM 역할
–
서브넷 ID
subnet-0ac81a58c52ade18d
Auto Scaling 그룹 이름
–
IMDSv2
Required
인스턴스 ARN
arn:aws:ec2:ap-northeast-2:985090322396:instance/i-0215122488ed3bd5c
관리형
false
연산자
–
세부 정보
상태 및 경보
모니터링
보안
네트워킹
스토리지
태그
인스턴스 세부 정보 정보
AMI ID
ami-040c33c6a51fd5d96
모니터링
비활성
플랫폼 세부 정보
Linux/UNIX
AMI 이름
ubuntu/images/hvm-ssd-gp3/ubuntu-noble-24.04-amd64-server-20240927
허용된 이미지
-
종료 방지
비활성

- 접속하기


ubuntu@ip-172-31-44-97:
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+WeYE8OTpPe2AYpJw0s1M.
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
Usage of /: 22.8% of 6.71GB
Memory usage: 22%
Swap usage: 0%
Temperature: -273.1 C
Processes: 111
Users logged in: 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
```

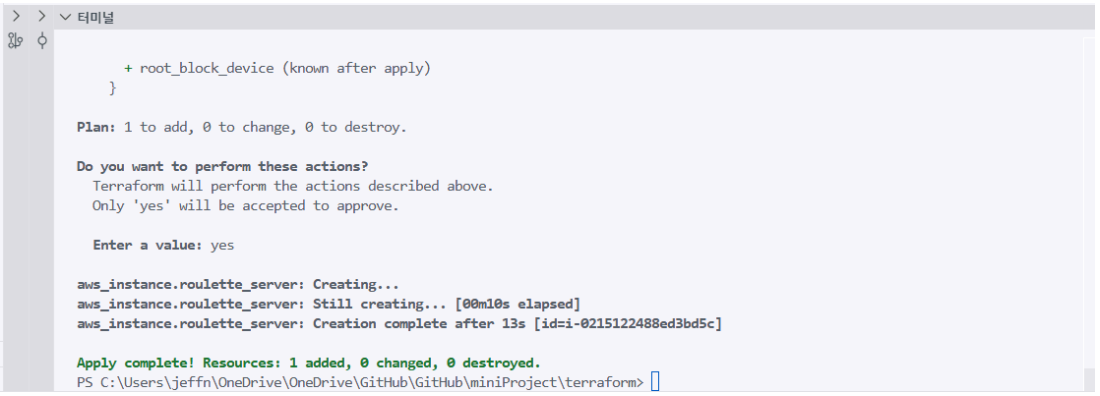
   - *여기서 "1 to add"라는 메시지가 나오면 정상입니다!*

3. **실행 (진짜 생성):**

```
DOS
terraform apply
```

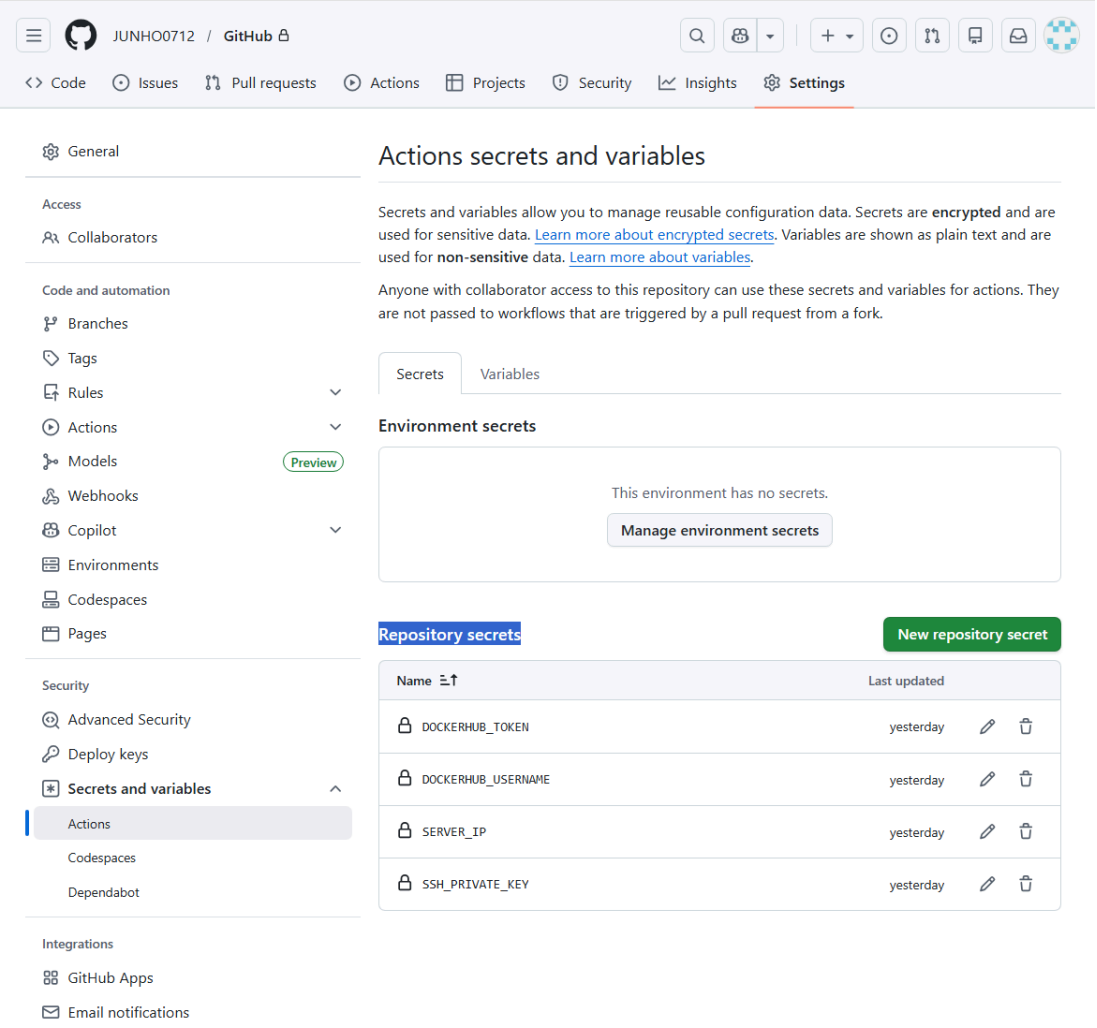
   - *중간에 `yes` 를 입력하라고 나오면 직접 타이핑하고 엔터를 치세요.*

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

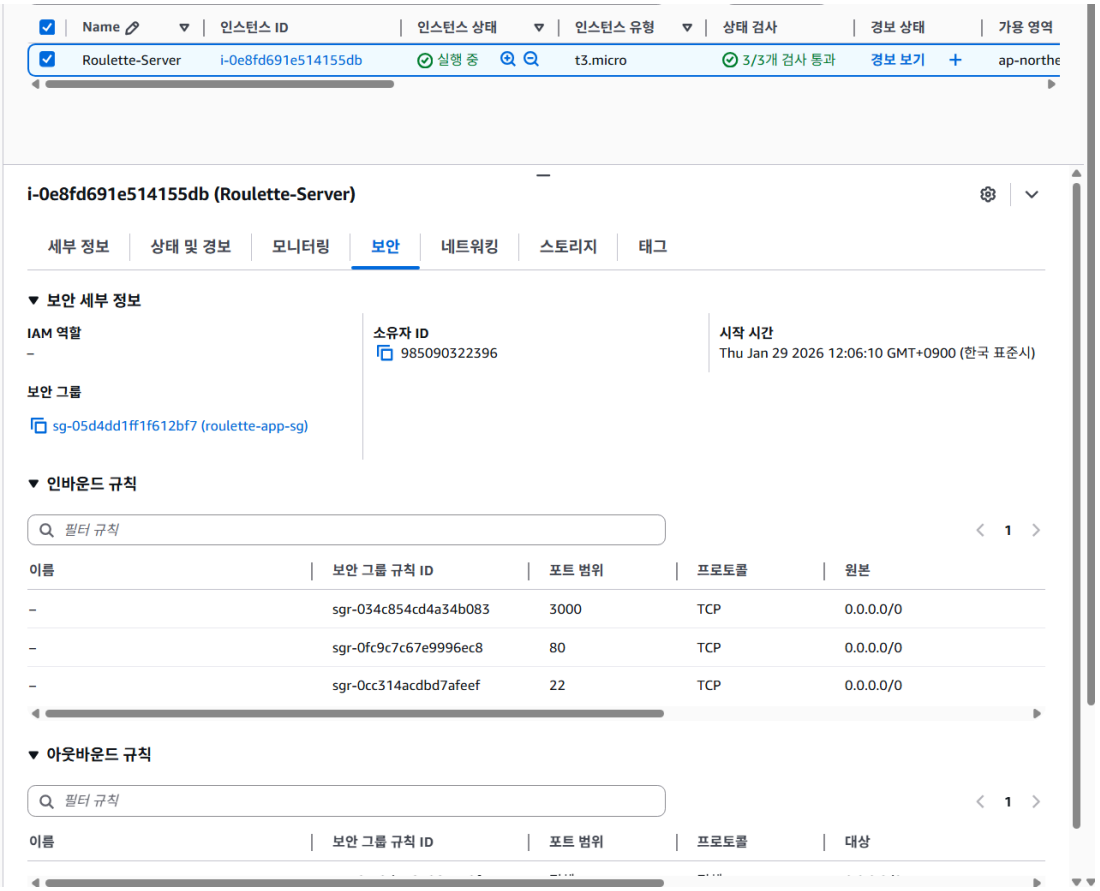


- GitHub Sevcrets 설정
  - DOCKERHUB_USERNAME
    - 도커 유저 이름

- DOCKERHUB_TOKEN
  - 도커 토큰
- SERVER_IP
  - 테라폼으로 생성한 **AWS EC2 서버의 주소**
- SSH_PRIVATE_KEY
  - 실물 열쇠(.pem 파일)의 내용

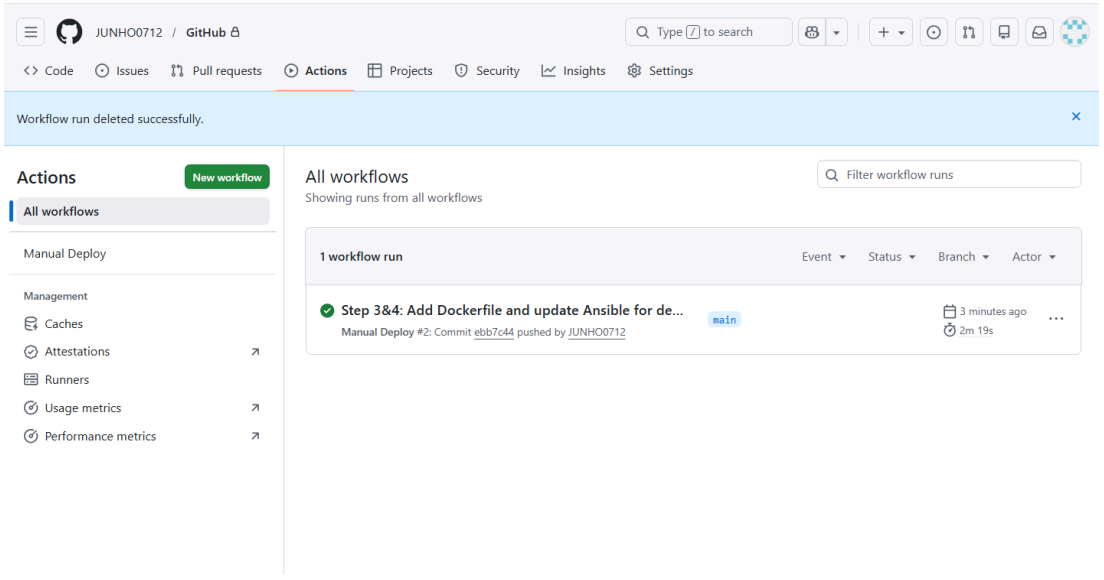


- 인스턴스 인바운드 설정 확인



- .github 폴더 생성 후 workflows로 로그 확인
  - 문제 발생시
    - 터미널에서 인스턴스 ip로 ssh 접속 후 도커 로그 확인

      $ sudo docker logs roulette-app



- 접속 확인
  - 인스턴스 ip 주소로 웹페이지 접속
  - 터미널 도커 로그 확인