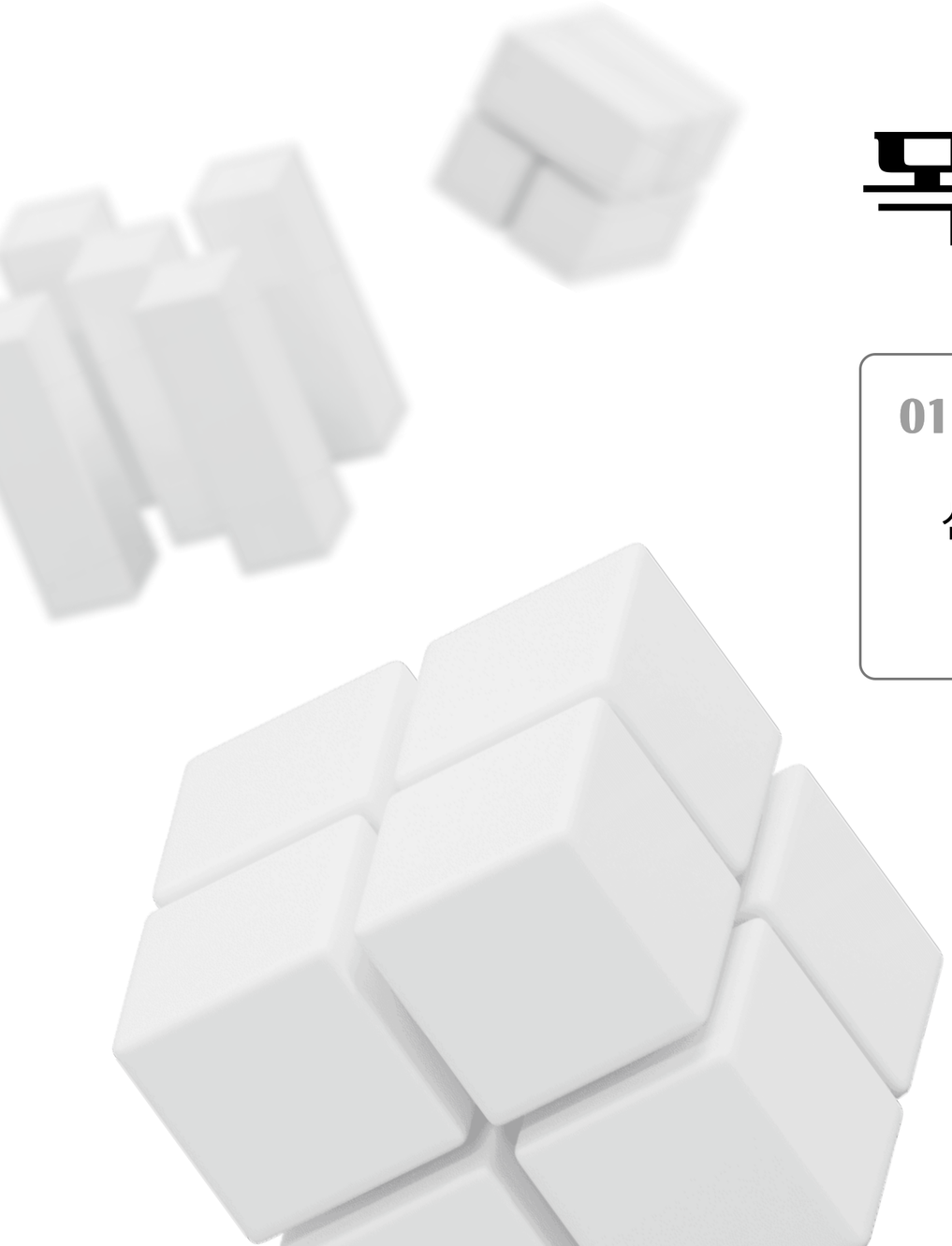
10304 김지환

CI/CD를 사용해 갤러리 배포



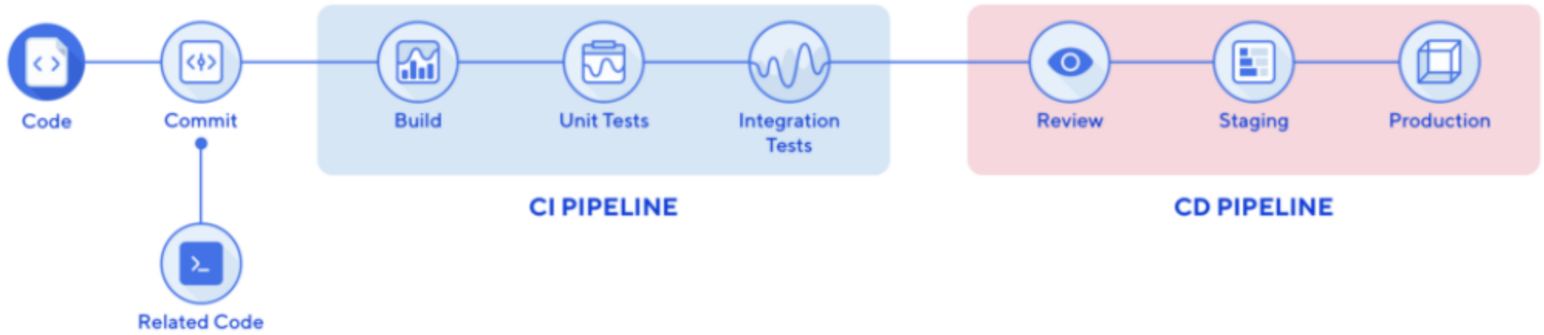
목차

선정이유

AWS 공부중 git clone 말고 그냥 배포까지 한번에 되는것이 없나
고민하던중 CI/CD에 대해 알게 되었음
재밌어보여서 발표주제로 선정함.

CI/CD의 배경

협업을 하기 위한 도구



- 소프트웨어 개발 라이프사이클을 간소화하고 가속화하는 것을 목표
- 소프트웨어 개발 라이프사이클이란?
  소프트웨어를 개발하기 위한 모든 과정

CI(지속적 통합)
github Actions

지속적 통합을 자동화 해주는 툴

주요기능

- Work flow

  github Actions에서 실행되는 자동화 작업의 집합

  레포지토리의 github/workflow Yaml 파일로 정의
- Event

  work flow를 트리거 하는 github이벤트
- Job

  워크플로우에서 실행가능한 가장 작은 단위
- Step

  Job내에서 순차적으로 실행되는 명령어
- Actions

  Github Actions에서 실행 가능한 재사용 가능한 작업단위

CI(지속적 통합)?

개발자가 작성한 코드를 중앙저장소에 자주 통합하며
자동으로 빌드 및 테스트를 하는 것

- Build

  컴파일과 빌드를 하는것

- Unit Tests & Integration tests

  Unit Tests(유닛 테스트)

  전체 코드중 일부분만 테스트 하는것

  Integration tests(통합 테스트)

  각각의 시스템과의 상화작용을 테스트

CD(지속적 배포)

- Review

  본격적인 배포 전 마지막으로 코드를 검토하고 초기 검증을 수행하는 단계

- Staging

  실제 운영 환경(Production)과 최대한 동일한 환경에서 소프트웨어를 실행해 보는 단계

- Production

  모든 검증이 완료된 코드를 실제 서비스 환경에 적용하여 사용자에게 제공하는 최종 단계

CD(지속적 배포)
Argo CD

Kubernetes 전용

주요기능

- Retrieval(조회/가져오기)

  Git 저장소에 저장된 최신 설정 파일(Manifest)을 가져옵니다
- Comparison(비교)

  현재 실제 서버(Cluster)에 배포된 상태와 Git에 있는 코드를 서로 비교합니다
- Sync(동기화)

  Git의 내용이 서버와 다를 경우, 서버의 상태를 Git의 코드와 동일하게 업데이트
- Health Assessment(상태평가)

  배포된 애플리케이션이 정상적으로 작동하고 있는지 실시간으로 확인

Github Actions VS Github & Git

Git

- 소스코드 버전 관리 도구
- Local에서 변경사항관리를 위해 사용하는 소프트웨어

VS

Github

- 클라우드 저장소 및 협업 플랫폼
- Git으로 관리하는 Project를 올릴수 있는 웹사이트

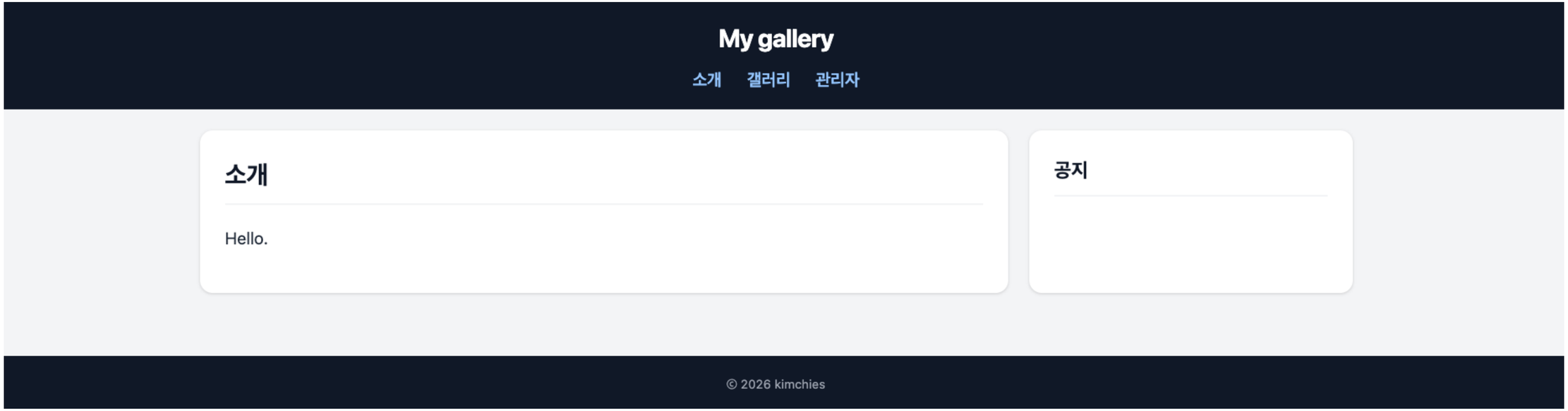
VS

Github Actions

- CI/CD 자동화 플렛폼
- Github내의 이벤트를 처리해주는 CI/CD도구

CI/CD를 이용한 Gallary 배포

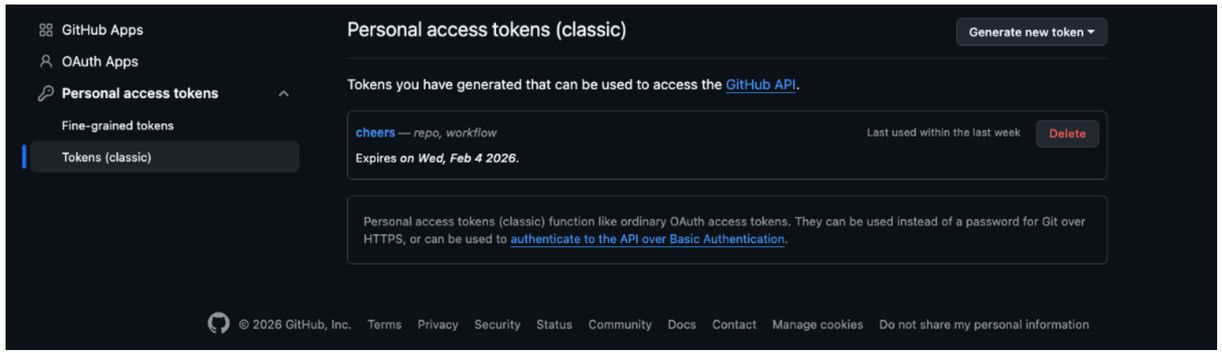
부제 : 도제때 사용했던 html 우려먹기

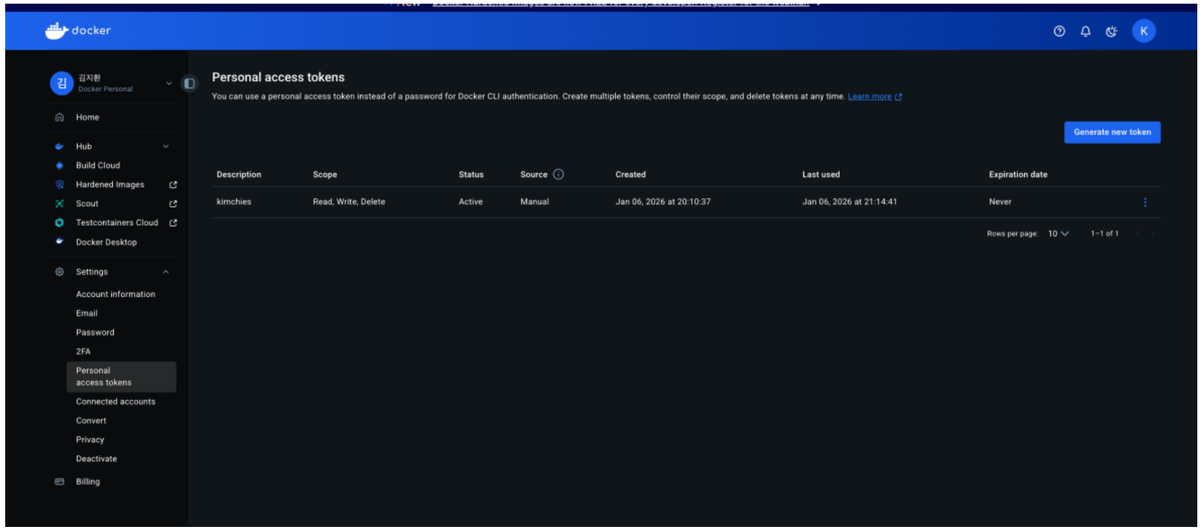


완성된 Gallary의 형태이다.

CI/CD를 이용한 Gallary 배포

- 사전 설정



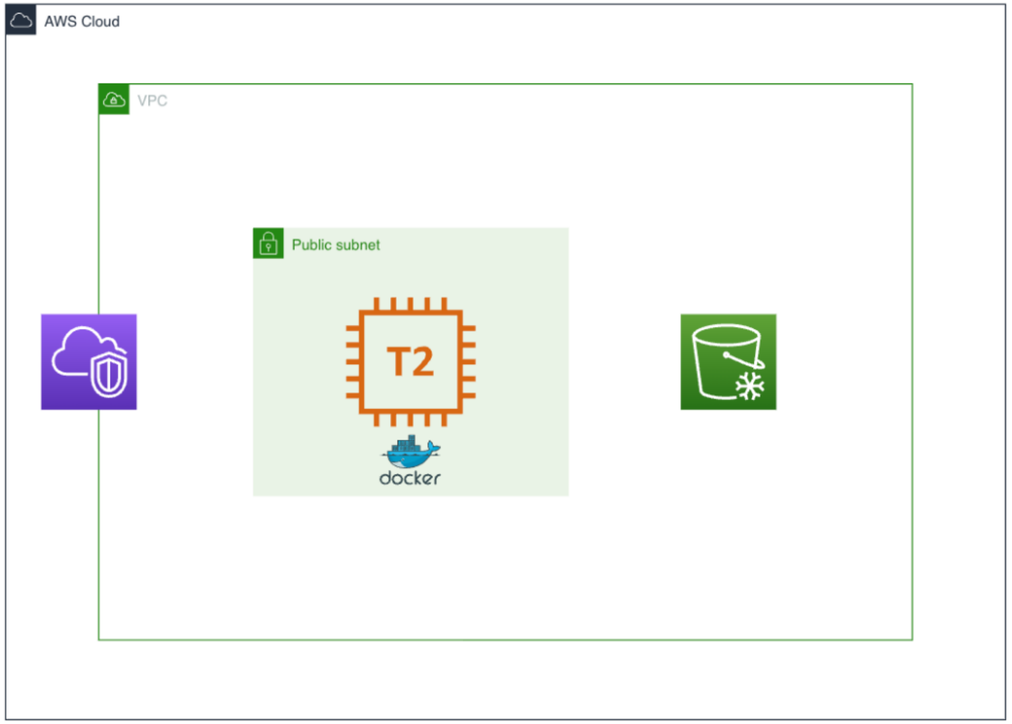


배포할때 사용할 github token 그리고 docker token
aws accress key를 발급받아 저장한다.

CI/CD를 이용한 Gallary 배포

- AWS 인프라 구축하기

AWS의 다이어그램이다.



갤러리에서는 S3 Backet을 연동하였고
관리자 로그인을 구현하여 S3Backet에 올리기로 하였음
Github Actions에서 CI/CD배포 진행함.

CI/CD를 이용한 Gallary 배포

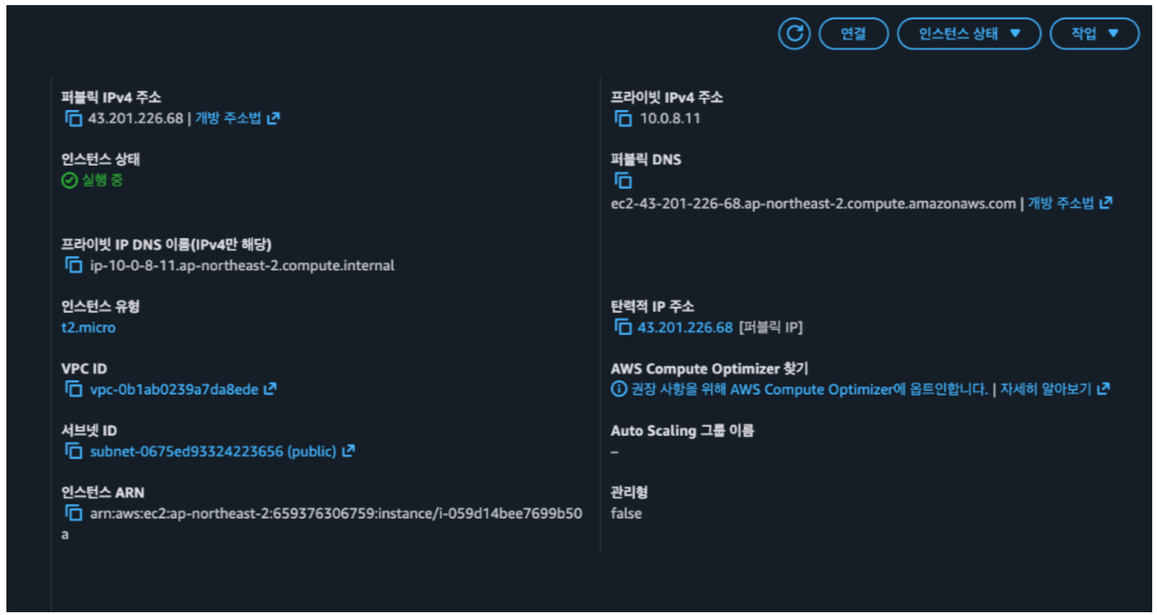
- AWS 인프라 구축하기

보안 그룹 규칙 ID	IP 버전	유형	프로토콜	포트 범위	소스
sgr-01178ce3624cbac43	IPv4	SSH	TCP	22	0.0.0.0/0
sgr-05fdeeab371be41a9	IPv4	HTTP	TCP	80	0.0.0.0/0
sgr-0c464659deb7130d9	IPv4	사용자 지정 TCP	TCP	3000	0.0.0.0/0

VPC의 맵을 보자
SSH와 HTTP연결을 위한 포트와
docker포트인 22,80,3000번
포트를 개방한다.

CI/CD를 이용한 Gallary 배포

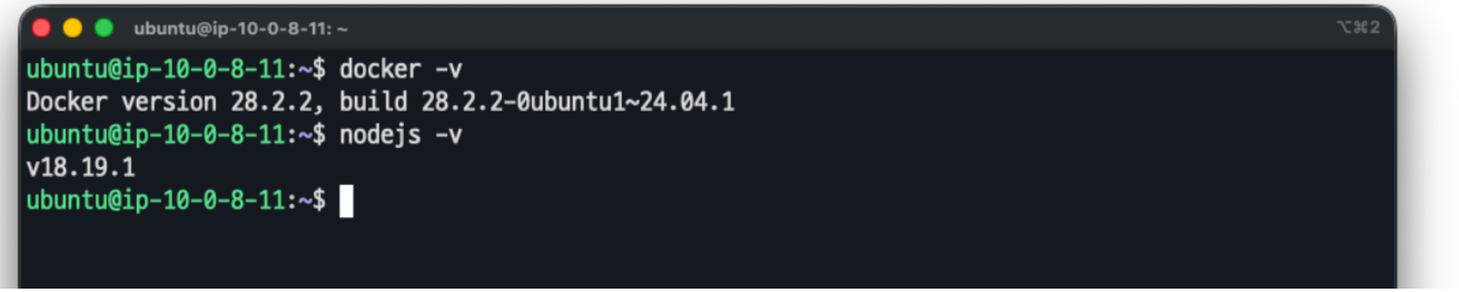
- AWS 인프라 구축하기



EC2가 꺼져도 IPv4주소가 바뀌지 않도록
탄력적 IP를 부여함.

CI/CD를 이용한 Gallary 배포

- EC2 설정하기



SSH로 EC2에 접근해 docker과 nodejs를 깔아준다.
그리고 cheers-09라는 버킷의 권한을 퍼블릭으로 바꿔줘야한다.
또한 내 도메인 한국에서 도메인을 등록해준다.

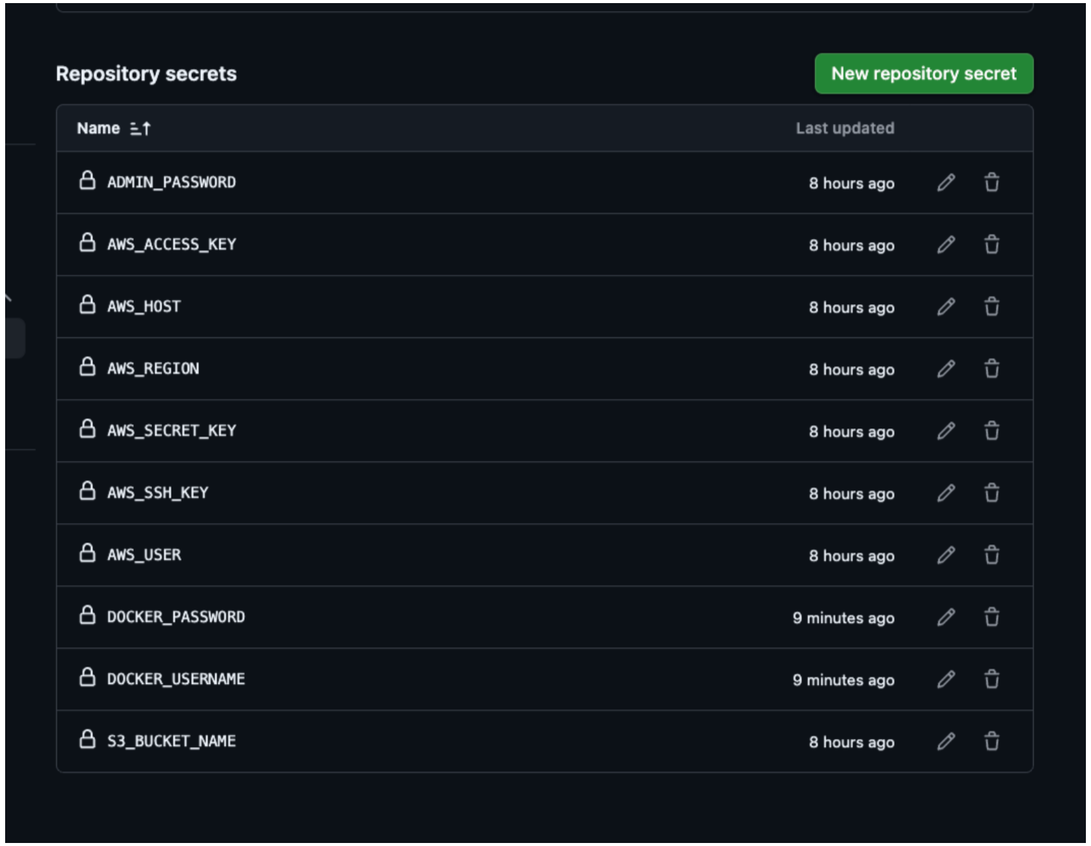
CI/CD를 이용한 Gallary 배포

- CI/CD 배포 설계도

git push → github repo → github actions
→ docker hub login & build → ec2접근
→ docker hub img pull → 배포완료

CI/CD를 이용한 Gallary 배포

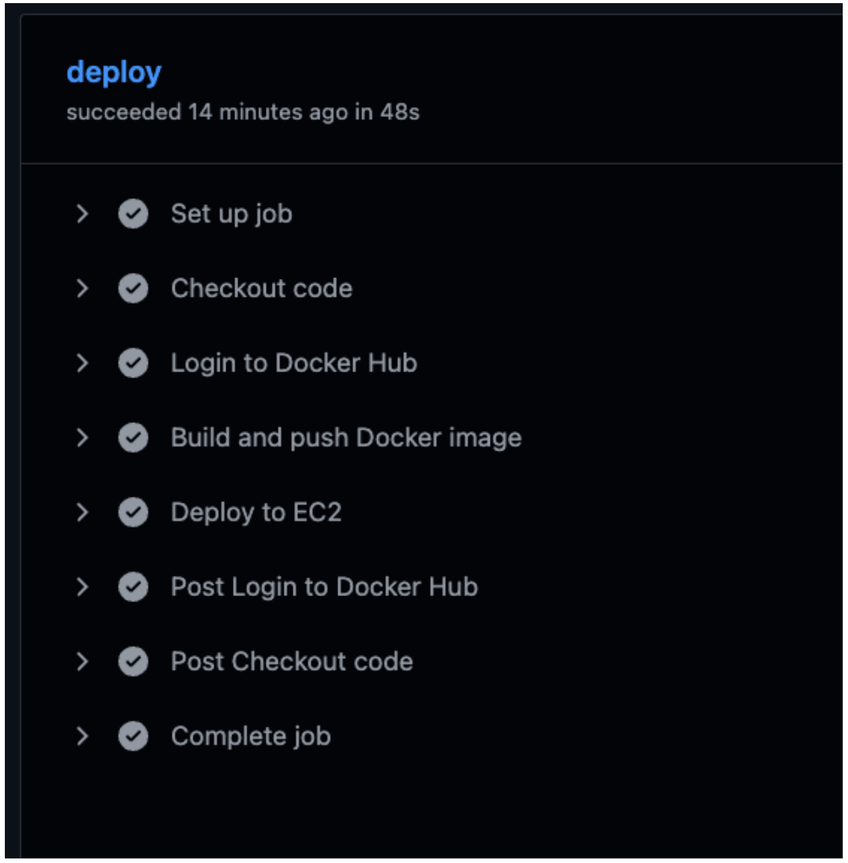
- github Repository 설정하기



local 에서는 .env파일에 저장했던 변수들을 github Screet에 저장함.
여기엔 Docker, aws login을 위한 키들과 ec2 보안키가 있음

CI/CD를 이용한 Gallary 배포

- git push & github actions 확인



이제 마지막으로 git push 하면 끝난다.
push 하면 이렇게 작업이보여지면서
자동배포가 시작된다.

github 페이지
https://github.com/kimcheers/cheers-gallary

마무리

http://cheers-gallary.kro.kr/

AWS의 인프라도 TerraForm이라는 Iac언어를 통해 작성할수 있던데
한번 해보고 싶고 kubernates를 이용해서 하는 프로젝트였으면
ArgoCD도 이용해 볼 수 있었을 텐데 아쉽다.