



Docker+CI/CD - 2주차

[목차]

00. 2주차 오늘 배울 것

✓ 목표

01. Github Actions 을 활용한 CI/CD 파이프라인 (1/3)

02. Github Actions 을 활용한 CI/CD 파이프라인 (2/3)

03. Github Actions 을 활용한 CI/CD 파이프라인 (3/3)

2주차 끝



모든 토글을 열고 닫는 단축키

Windows :

Ctrl + alt + t

Mac :

⌘ + ⌥ + t

00. 2주차 오늘 배울 것



Github Actions에 대해 이해하고 실제로 적용해 봅니다.

✓ 목표

- Github Actions를 이해합니다.
- 간단한 CI/CD 파이프라인을 구성합니다.

01. Github Actions 을 활용한 CI/CD 파이프라인 (1/3)



Github Actions가 무엇인지 알아보아요

▼ ? GitHub에서 Github Actions란?

- Github Actions 는 Github에 내장된 CI/CD 도구

Get started with GitHub Actions

Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you want. Select a workflow to get started.

Skip this and [set up a workflow yourself](#) →

Q Search workflows

Suggested for this repository

Simple workflow

By GitHub

Start with a file with the minimum necessary structure.

[Configure](#)

Deployment

[View all](#)

Deploy Node.js to Azure Web App

By Microsoft Azure

Build a Node.js project and deploy it to an Azure Web App.

[Configure](#)

Deployment

Deploy to Amazon ECS

By Amazon Web Services

Deploy a container to an Amazon ECS service powered by AWS Fargate or Amazon EC2.

[Configure](#)

Deployment

Build and Deploy to GKE

By Google Cloud

Build a docker container, publish it to Google Container Registry, and deploy to GKE.

[Configure](#)

Deployment

Terraform

By HashiCorp

Set up Terraform CLI in your GitHub Actions workflow.

[Configure](#)

Deployment

Deploy to Alibaba Cloud ACK

By Alibaba Cloud

Deploy a container to Alibaba Cloud Container Service for Kubernetes (ACK).

[Configure](#)

Deployment

Deploy to IBM Cloud Kubernetes Service

By IBM

Build a docker container, publish it to IBM Cloud Container Registry, and deploy to

[Configure](#)

Deployment

Tencent Kubernetes Engine

By Tencent Cloud

This workflow will build a docker container, publish and deploy it to

[Configure](#)

Deployment

OpenShift

By Red Hat

Build a Docker-based project and deploy it to OpenShift.

[Configure](#)

Deployment

- Github에 내장되어 있는 CI/CD라 github와 통합이 쉽고, CI/CD 서버가 내장 되어 CI/CD서버를 따로 구축할 필요 없으며, 일정 수준까지 가격이 무료
 - 무료 버전 : 스토리지 500MB, 월 2000분
- Github Actions 동작 방법
 - repository의 `.github/workflows` 디렉토리에 필요한 Actions 파일들을 yaml 형식으로 작성
 - 반드시 `.github/workflows` 폴더 아래에 YAML 파일이 위치
 - 작성된 actions 파일들을 github에서 자동으로 실행

▼ ? Github Actions 에서 CI란?

- Github Actions 의 CI

- test를 통과한 코드만 `develop` 브랜치와 `main` 브랜치에 merge되도록 하여 오류를 방지하고 안정적인 코드가 배포되고 버그를 빠르게 발견
- ▶ 활용 예
 - `develop` 브랜치에 merge 된 경우, gradle test 를 진행
 - `feature/**` (feature하위) 브랜치가 push 된 경우, gradle test 를 진행
 - 만약 gradle test가 실패한 경우, slack 등 알림을 보내 코드를 수정하도록 개발자에게 안내
 - 이메일은 자동으로 발송되도록 기본 설정

▼ 샘플 (실제 동작하지 않음)

```
# Actions 이름 github 페이지에서 볼 수 있다.
name: 'CI'

# Event Trigger 특정 액션 (Push, Pull_Request)등이 발생
on:
  push:
    # 배열로 여러 브랜치를 넣을 수 있다.
    branches: [ develop, feature/* ]
  # github pull request 생성시
  pull_request:
    branches:
      - develop # -를 쓴 여러 줄로 여러 브랜치를

# 실제 어떤 작업을 실행할지에 대한 명시
jobs:
  ci:
    # 스크립트 실행 환경 (OS)
    # 배열로 선언시 개수 만큼 반복해서 실행한다. ( 예제 : 1번 )
    runs-on: [ ubuntu-latest ]

    # 실제 실행 스크립트
    steps:
      # uses는 github actions에서 제공하는 플러그인을 사용
      - name: checkout
        uses: actions/checkout@v4
```

```
# with은 plugin 파라미터 입니다. (java 11버전 셋
- name: java setup
  uses: actions/setup-java@v2
  with:
    distribution: 'adopt' # See 'Supported
    java-version: '17'

# run은 사용자 지정 스크립트 실행
- name: run unittest
  run: |
    ./gradlew clean test
```

▼ ? Github Actions에서 CD란?

- 배포를 자동화하는 작업을 기술해서 빠르고 간편하게 배포
- Github Actions 의 CD
 - `main` 브랜치에 코드가 통합된 경우 운영 환경에 빠르게 배포할 수 있게 함
 - ▶ 예제
 - `main` 브랜치에 merge된 경우, `gradle test` 를 실행
 - `main` 브랜치의 코드 기준으로 `jar` 파일을 생성
 - 생성된 `jar` 파일을 특정 환경(AWS, GCP 등)에 배포
 - 샘플(실제 동작하지 않음)

```
name: 'CD'

on:
  push:
    branches: [ main ]
jobs:
  cd:
    runs-on: [ ubuntu-latest ]

    steps:
      - name: checkout
        uses: actions/checkout@v4
```

```

- name: java setup
  uses: actions/setup-java@v3
  with:
    distribution: 'adopt' # See 'Supported
    java-version: '17'

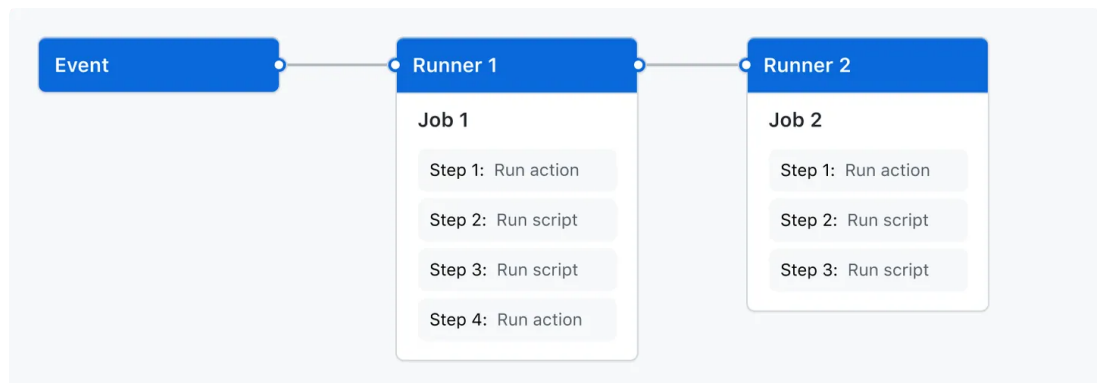
- name: run unittest
  run: |
    ./gradlew clean test

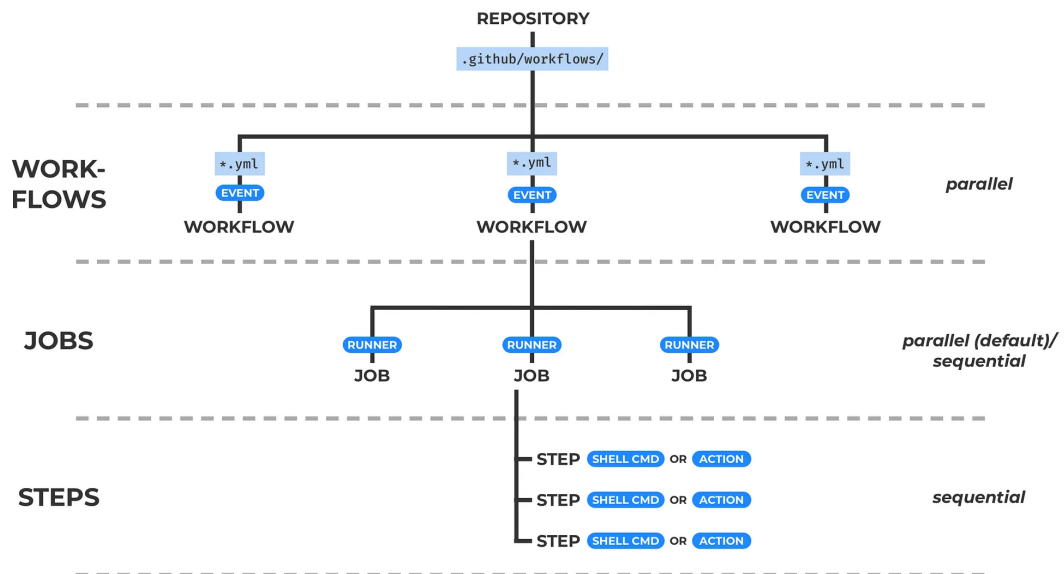
- name: deploy to heroku
  uses: akhileshns/heroku-deploy@v3.12.12
  with:
    heroku_api_key: ${secrets.HEROKU_API_K
    heroku_app_name: "sampleapp-github-acti
    heroku_email: "nbcdocker@proton.me"

```

▼ 🔍 Github Actions 뜯어보기

- event, runner, job, step



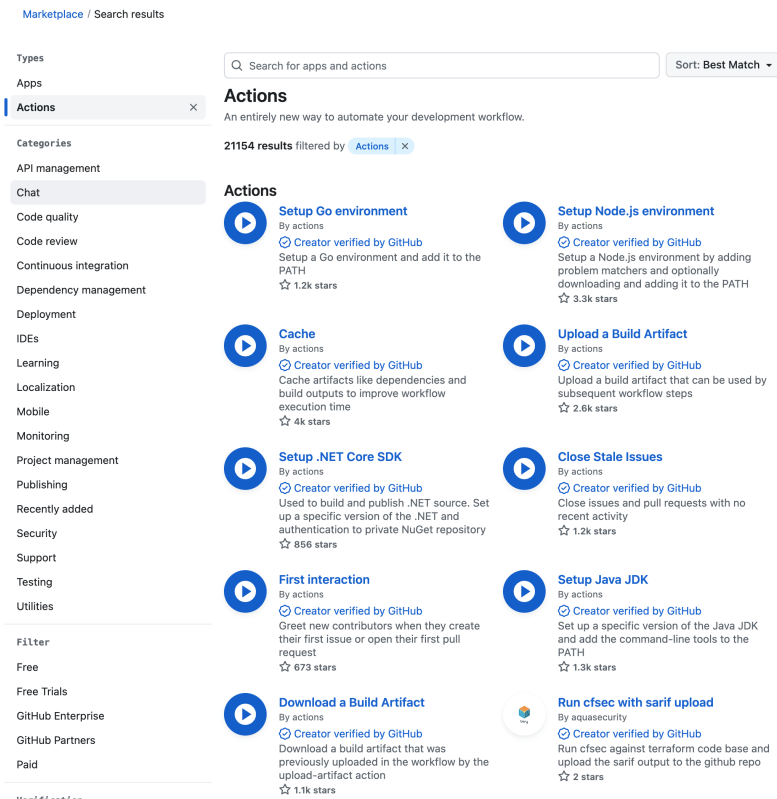


- Workflow
 - 최상위 개념
 - 여러 Job으로 구성되고, Event에 의해 트리거될 수 있는 자동화된 프로세스
 - Workflow 파일은 YAML로 작성되고, Github Repository의 `.github/workflows` 폴더 아래에 저장됨
- event
 - Github Repository에서 발생하는 push, pull request open, issue open, 특정 시간대 반복(cron) 등의 특정한 규칙
 - workflow 를 실행(trigger)함
- runner
 - Github Action Runner app이 설치된 VM
 - Workflow가 실행될 instance로, 각각의 Job 들은 개별적인 runner에서 실행
- job
 - 하나의 runner에서 실행될 여러 step의 모음을 의미
- step
 - 실행 가능한 하나의 shell script 또는 action

- Actions
 - Workflow의 가장 작은 단위로 재사용이 가능
 - Job을 만들기 위해 Step들을 연결

▼ workflow 뜯어보기

- name
 - github actions의 이름을 정하는 부분
- on
 - 이 action이 언제 실행되는지에 대한 부분
- jobs
 - 실제 실행할 내용에 대한 부분
 - runs-on: 어떤 환경에서 실행하는지 기술
 - <https://docs.github.com/ko/actions/using-github-hosted-runners/about-github-hosted-runners/about-github-hosted-runners>
 - steps: 실제 실행할 단계들을 기술
 - name: 실행에 표시될 이름
 - uses: 여러 가지 plugin 사용
 - 다양한 action들을 사용 - <https://github.com/marketplace?type=actions>



- with: plugin 에서 사용할 파라미터들
- run: 실제로 실행할 스크립트



Github Actions 예제를 내 repository에서 실행시켜요

▼ ▶ github actions 예제

- `.github/workflows/github-actions-demo.yaml`

```
name: GitHub Actions Demo
run-name: ${{ github.actor }} is testing out GitHub Actions
on: [push]
jobs:
  Explore-GitHub-Actions:
    runs-on: ubuntu-latest
    steps:
      - run: echo "🎉 The job was automatically triggered by a ${{ github.actor }} push"
      - run: echo "🐧 This job is now running on a ${{ runner.os }} machine hosted by GitHub Actions"
```



```

- run: echo "🔍 The name of your branch is ${GITHUB_REF_NAME}
- name: Check out repository code
  uses: actions/checkout@v4
- run: echo "💡 The repository is ${{ github.repository }}
- run: echo "💻 The workflow is now ready to run
- name: List files in the repository
  run: |
    ls ${{ github.workspace }}
- run: echo "🍏 This job's status is ${{ job.status }}"

```

- 참고: <https://github.com/nbcdocker/github-actions/tree/main>
- 비어 있는 github repository를 만들고 위 데모를 push 해서 실행시켜봅시다!

02. Github Actions 을 활용한 CI/CD 파이프라인 (2/3)

✓ 간단한 Spring Boot 앱을 만들고 Github Actions 를 사용해서 CI를 경험해요

▼ 내 repository에 Github Actions 실행하기

- 파일 다운로드 후 새로운 프로젝트 시작

[sampleapp.zip](#)

- 다운로드 한 폴더를 IntelliJ 에서 열기
- Menu → VCS → **Enable Version Control Integration** 선택 → **Git** 선택
- github 에 신규 repository 생성
- Menu → git → **Mange Remotes** 선택
- **+** 선택 후, **<https://github.com/nbcdocker/spring-boot.git>** 와 같이 나의 repository의 URL 입력 → **OK**
- 참고: <https://github.com/nbcdocker/sampleapp>
- workflow 설명

- `develop` 이나 `feature` 로 시작하는 브랜치에 코드가 push 되거나 `develop` 을 destination으로 하는 pull request가 생성되면,
- `./gradlew clean test` 를 실행한다.
- ▶ `.github/workflows/run-test.yaml`

```
# Actions 이름 github 페이지에서 볼 수 있다.
name: Run Test

# Event Trigger 특정 액션 (Push, Pull_Request)등이 명시한 B
on:
  push:
    # 배열로 여러 브랜치를 넣을 수 있다.
    branches: [ develop, feature/* ]
  # github pull request 생성시
  pull_request:
    branches:
      - develop # -로 여러 브랜치를 명시하는 것도 가능

# 실제 어떤 작업을 실행할지에 대한 명시
jobs:
  build:
    # 스크립트 실행 환경 (OS)
    # 배열로 선언시 개수 만큼 반복해서 실행한다. ( 예제 : 1번 실행 )
    runs-on: [ ubuntu-latest ]

    # 실제 실행 스크립트
    steps:
      # uses는 github actions에서 제공하는 플러그인을 실행.(g
      - name: checkout
        uses: actions/checkout@v4

      # with은 plugin 파라미터 입니다. (java 17버전 셋업)
      - name: java setup
        uses: actions/setup-java@v2
        with:
          distribution: 'adopt' # See 'Supported distri
          java-version: '17'
```

```

- name: make executable gradlew
  run: chmod +x ./gradlew

# run은 사용자 지정 스크립트 실행
- name: run unittest
  run: |
    ./gradlew clean test

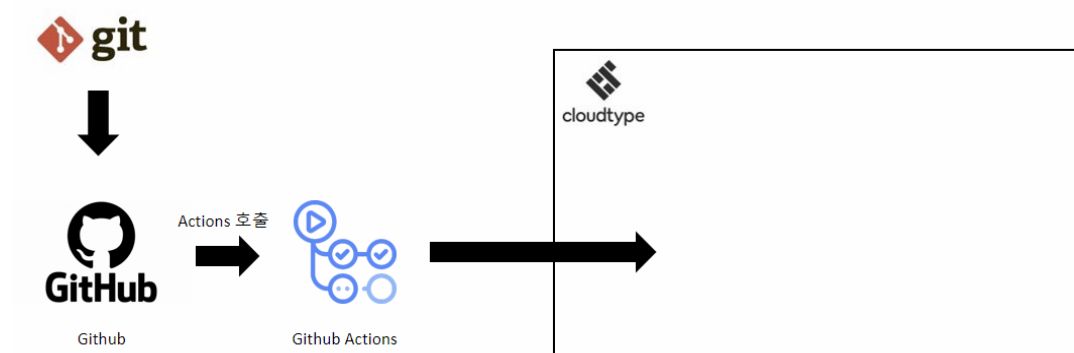
```

03. Github Actions 을 활용한 CI/CD 파이프라인 (3/3)

✓ 내가 만든 앱을 인터넷에 배포해요 (cloudtype 사용)

▼ 내 repository의 spring boot app 을 Github Actions로 배포하기

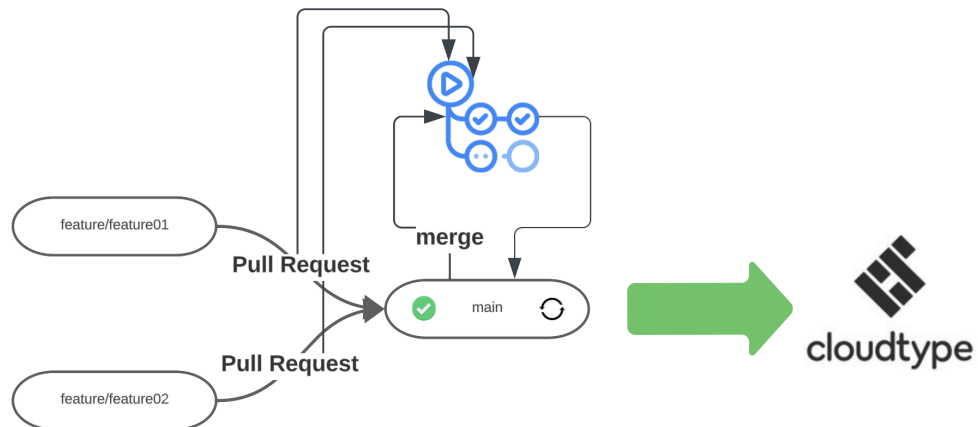
- 개념도



▼ 전체 흐름

- 개발자는 `feature/` 로 시작하는 브랜치를 만들어서 test코드를 포함한 수정 작업을 완료한 뒤 Pull Request 생성
- (자동화) Pull Request를 만들면 해당 브랜치에 대해 `gradle test` 를 수행

- Pull Request 코드의 test가 실패한 경우, Pull Request 를 생성한 개발자는 test 코드를 수정하여 Pull Request를 변경
- Pull Request 코드의 test가 성공한 경우, 다른 개발자들의 승인을 기다림
- 다른 개발자들은 Pull Request의 코드를 승인하거나 댓글로 소통
- **(자동화) main 브랜치에 merge 되면 해당 브랜치를 cloudtype 서버에 배포**



▼ Pull Request가 만들어지면 test를 수행하는 Github Action

```

name: test every pr
on:
  workflow_dispatch:
  pull_request:
permissions:
  contents: read
  pull-requests: read
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v3
      - name: setup jdk
        uses: actions/setup-java@v3
        with:
          java-version: '17'
          distribution: 'temurin'

```

```
    cache: gradle
  - name: Grant execute permission for gradlew
    run: chmod +x ./gradlew
  - name: gradlew test
    run: ./gradlew test
```

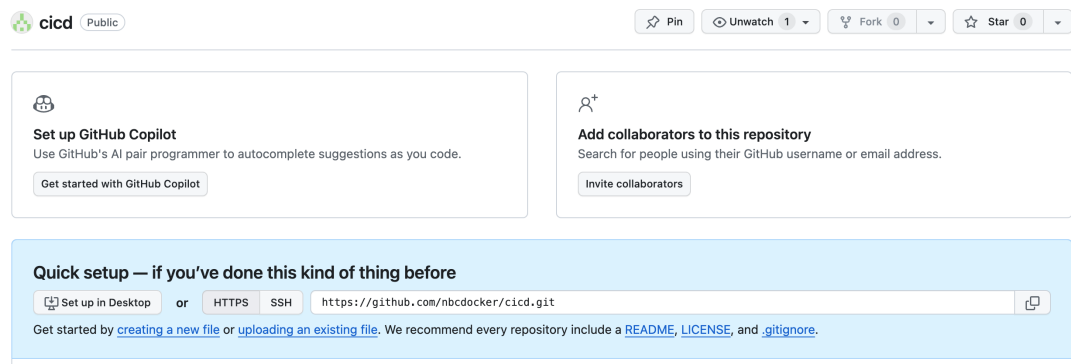
▼ cloudtype에 main 브랜치를 배포하는 Github Action

```
name: Deploy to cloudtype
on:
  workflow_dispatch:
  push:
    branches:
      - main
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v3
      - name: Connect deploy key
        uses: cloudtype-github-actions/connect@v1
        with:
          token: ${ secrets.CLOUDTYPE_TOKEN }
          ghtoken: ${ secrets.GHP_TOKEN }
      - name: Deploy
        uses: cloudtype-github-actions/deploy@v1
        with:
          token: ${ secrets.CLOUDTYPE_TOKEN }
          project: nbc.docker/cicd
          stage: main
          yaml: |
            name: cicd
            app: java@17
            options:
              ports: 8080
            context:
              git:
```

```
url: git@github.com:${{ github.reposito
ref: ${{ github.ref }}
preset: java-springboot
```

▼ 설정하기

▼ 저장소 생성 및 대상 코드 push

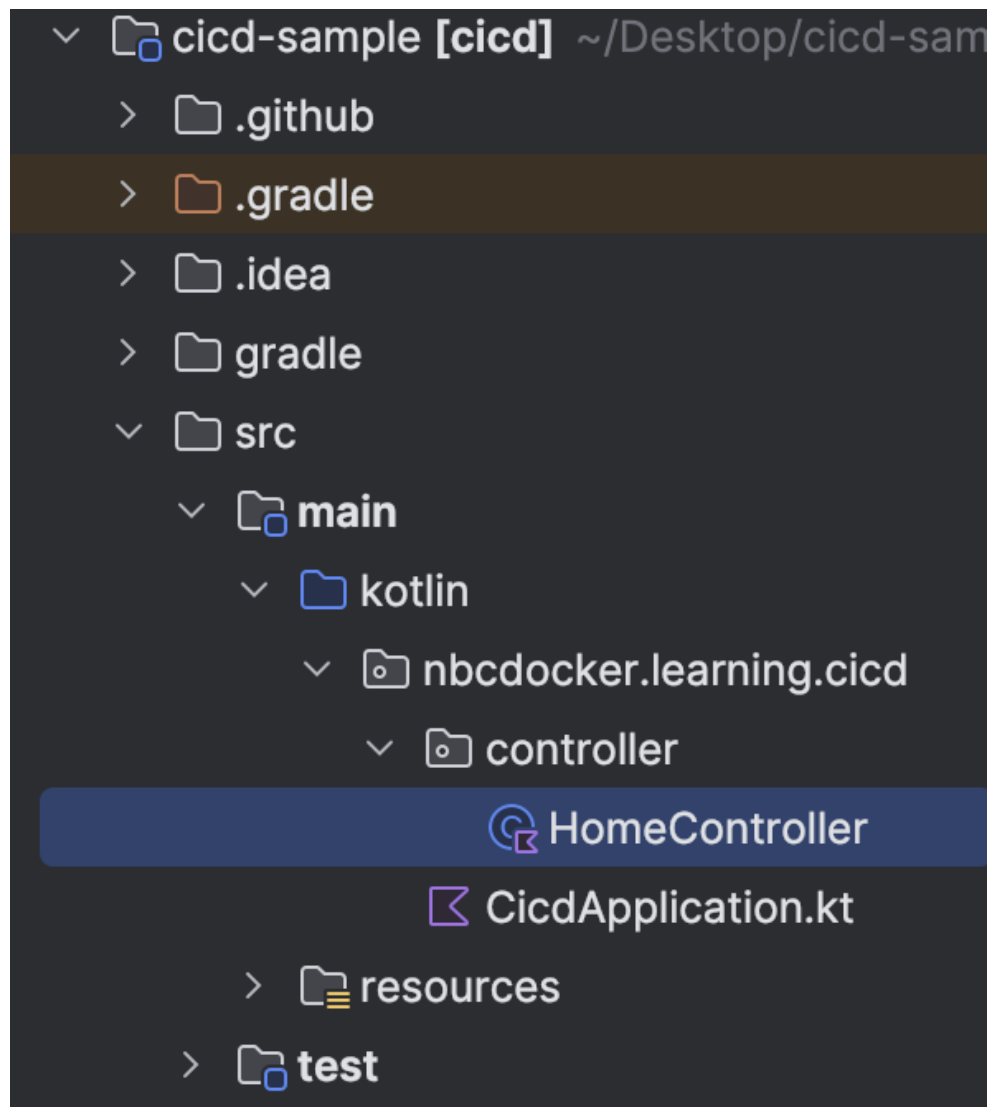


• 파일 다운로드 후 스프링 프로젝트 시작

- 프로젝트 이름은 `cicd`

cicd-sample.zip

- 참고 - <https://github.com/nbcdocker/cicd-sample>



- HomeController 만 있는 스프링 부트 앱을 인터넷 서버에 배포

```
@RestController
class HomeController {

    @GetMapping("/")
    fun home(): String {
        return "home"
    }

    @GetMapping("/healthz")
    fun healthz(): String {
        return "healthz"
    }
}
```

```
}  
}
```

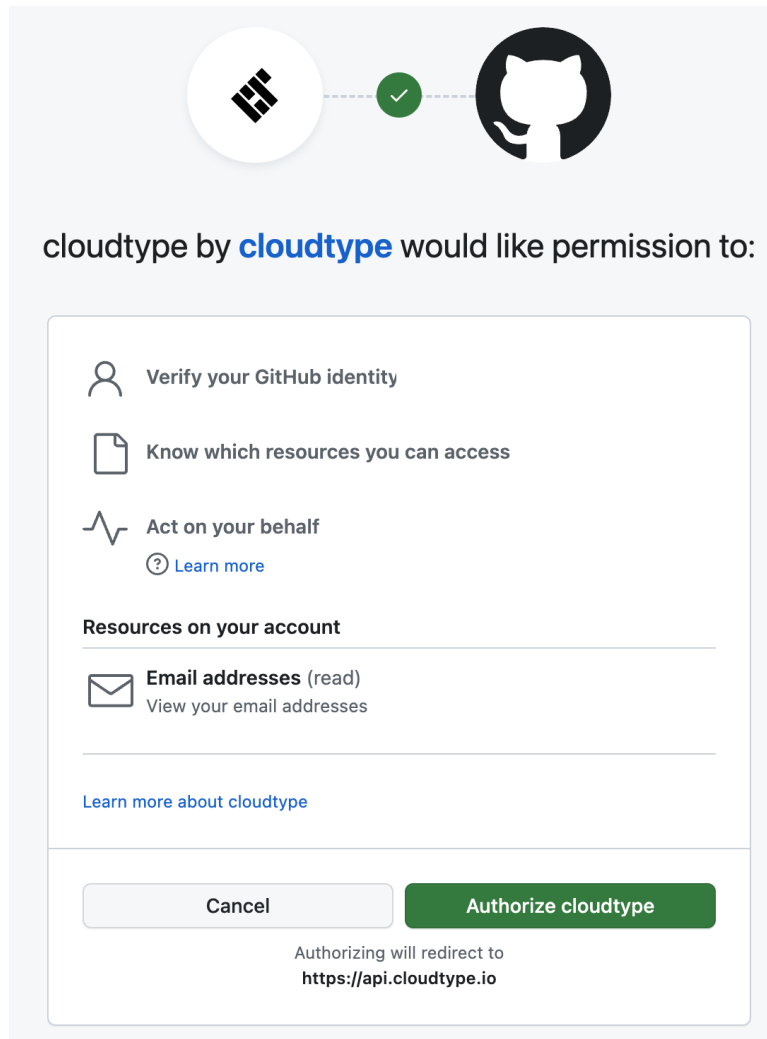
▼ cloudtype 가입

- <https://app.cloudtype.io/auth/signup> 클릭 후, [github 계정으로 가입하기](#) 클릭



- github 로그인 후 github의 이메일 정보를 cloudtype 이 가져갈 수 있도록

Authorize cloudtype




- 가입 완료

가입 완료하기

가입 정보를 확인해주세요.

이메일 주소

 nbc.docker@proton.me

인증됨 ✓

스페이스 이름

@ nbc.docker

✓

이름

nbcdocker

☒ 서비스이용약관 & 개인정보처리방침 에 동의합니다.

완료하기

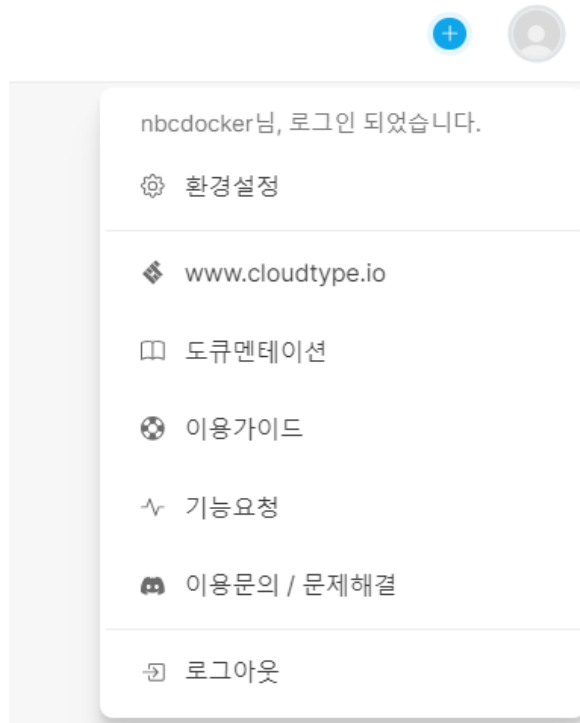
잘못된 이메일 주소로 가입하셨거나
회원가입 신청을 취소하고 싶으신가요?

정보 삭제 & 가입취소

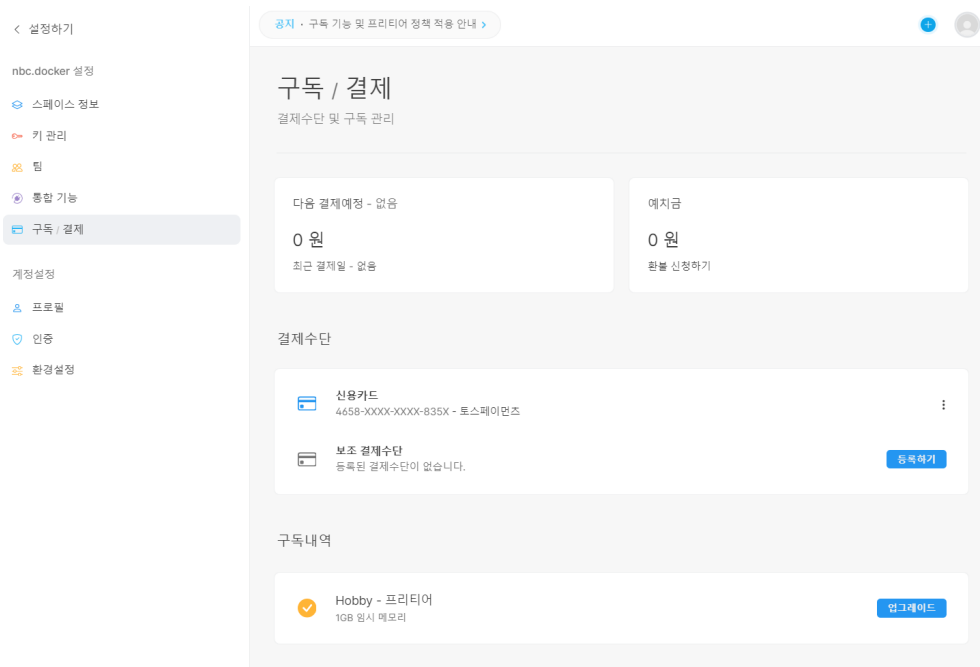
▼ cloudtype 설정 - 카드 등록 후 무료 요금제 활성화

▼ 카드 등록

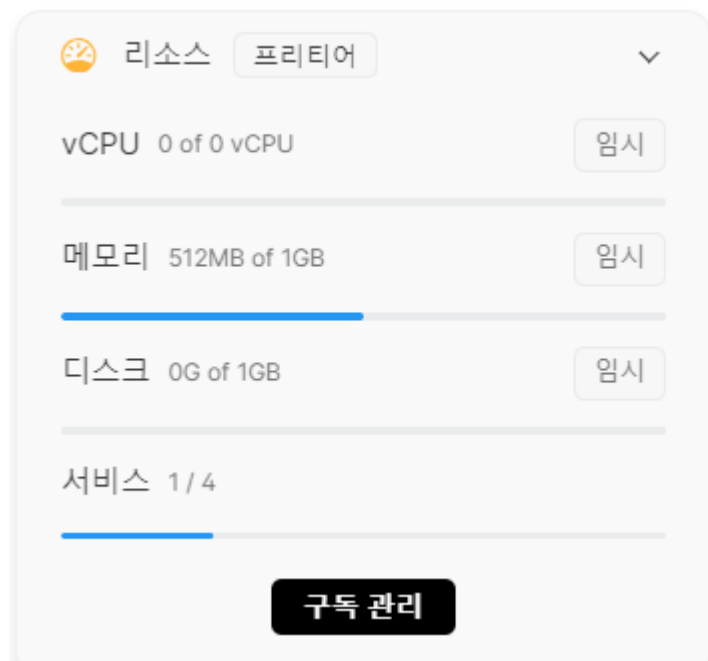
- 카드 등록을 해야 무료 요금제를 이용하실 수 있어요
- 오른쪽 위 프로필 이미지 클릭 후 '환경 설정' 클릭



- 왼쪽 메뉴 '구독/결제' 클릭 후 오른쪽 화면中间的 '결제수단' '등록하기'

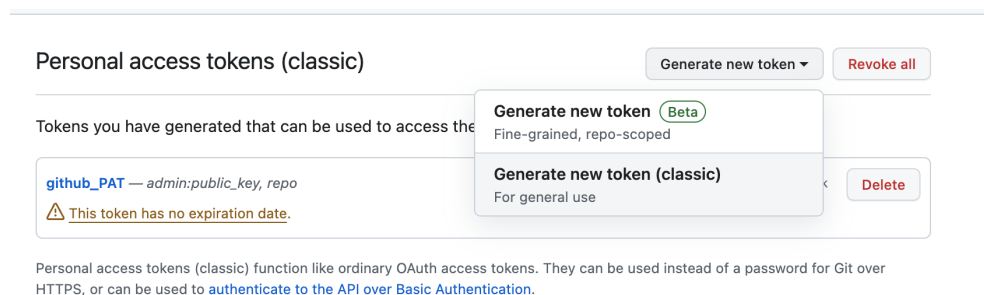


- 왼쪽 아래에 '리소스' 부분에 '메모리'가 1GB로 되어 있는지 확인



▼ github 설정 - cloudtype 과 연동하기 위한 설정

- <https://github.com/settings/tokens> 로 이동
- GitHub Personal Token 발급하기
 - GitHub Personal Token 을 생성하고 저장소의 설정에서 시크릿으로 추가하는 작업이 필요합니다. GitHub 계정의 **Settings** 에 진입 후 **Developer settings - Personal access tokens (classic)** 페이지로 이동합니다.
 - **Generate new token - Generate new token(classic)** 버튼을 클릭한 후 아래와 같이 권한을 부여하고 토큰을 생성합니다.



■ 필요 권한

- repo
- admin:public_key

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

cloudtype-delay

What's this token for?

Expiration *

30 days

The token will expire on Fri, Mar 17 2023

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> workflow	Update GitHub Action workflows
<input type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input type="checkbox"/> delete:packages	Delete packages from GitHub Package Registry
<input type="checkbox"/> admin:org	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input type="checkbox"/> read:org	Read org and team membership, read org projects
<input type="checkbox"/> manage_runners:org	Manage org runners and runner groups
<input checked="" type="checkbox"/> admin:public_key	Full control of user public keys
<input checked="" type="checkbox"/> write:public_key	Write user public keys
<input checked="" type="checkbox"/> read:public_key	Read user public keys
<input type="checkbox"/> admin:repo_hook	Full control of repository hooks

- 토큰 생성이 완료되면 아래 이미지와 같이 토큰 값이 표시됩니다. 토큰은 생성 시 최초 한 번만 값을 확인할 수 있으며, 이동하게 되면 이후에는 다시 그 값을 조회할 수 없기 때문에 메모장에 복사 후 파일로 저장해 주세요.

Personal access tokens (classic)

Generate new token ▼

Revoke all

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your personal access token now. You won't be able to see it again!

✓  

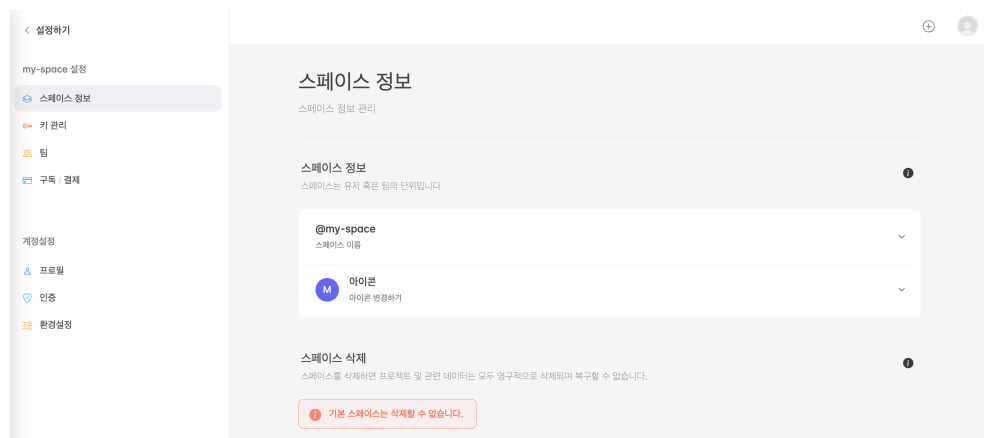
Delete

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

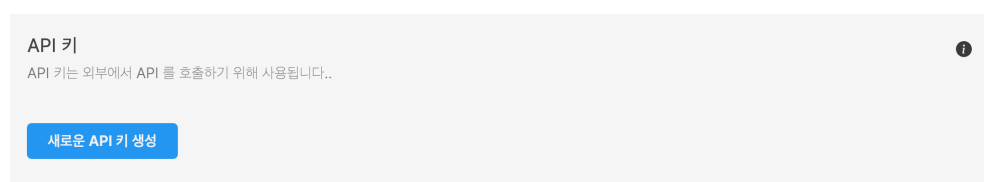
▼ cloudtype 설정

▼ cloudtype 의 API Key 발급

- API Key는 외부에서 클라우드타입의 API를 활용하고자 할 때 사용되는 인증 정보입니다. Github Actions 등 외부의 CI/CD 도구를 활용하여 배포 작업을 진행할 시 API Key의 발급이 필요합니다.
- 오른쪽 위의 톱니바퀴 아이콘을 클릭하여 스페이스 설정 화면으로 진입합니다.

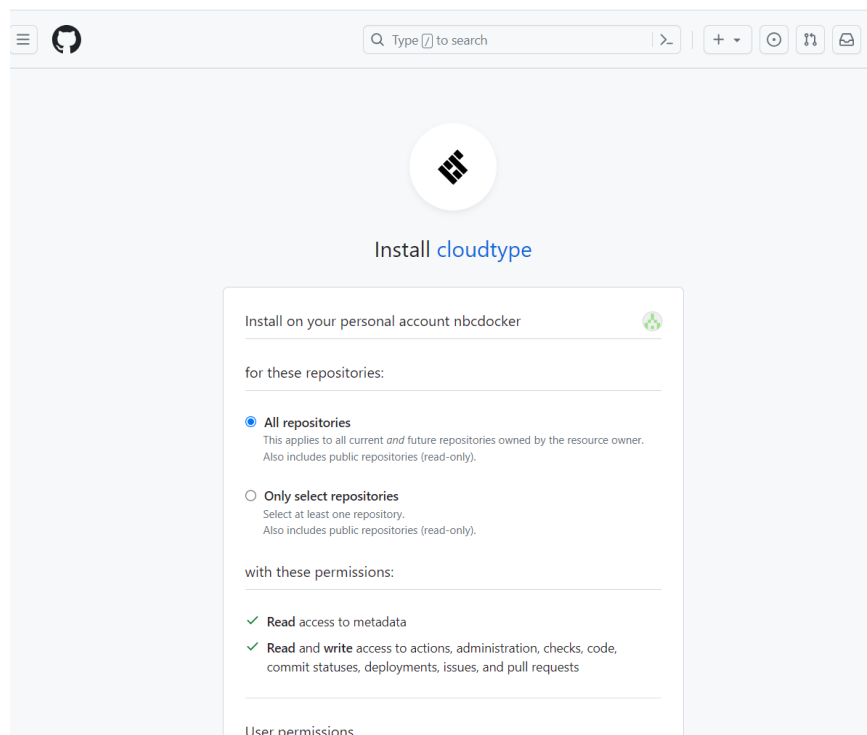
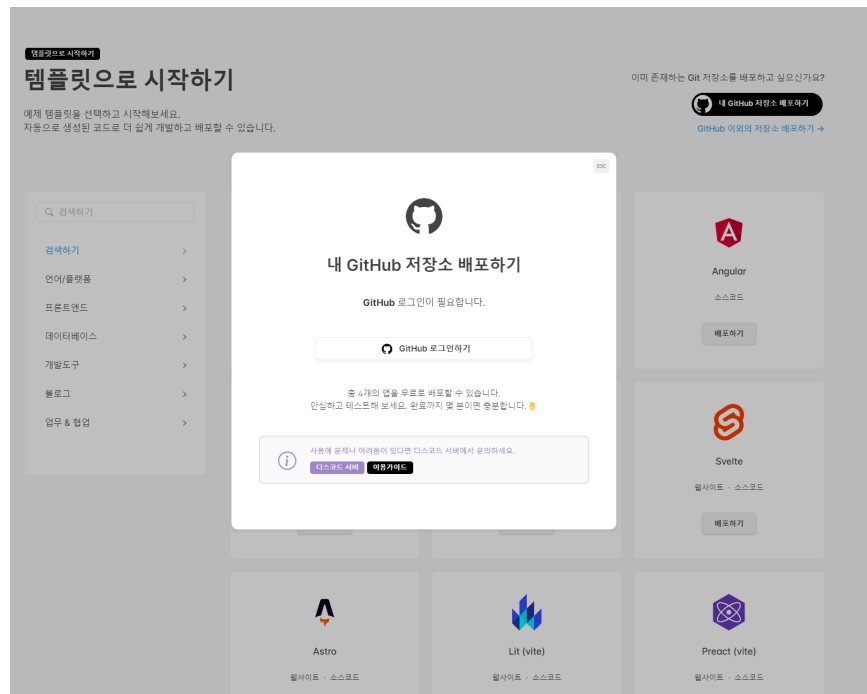


- 왼쪽 중간에 **인증** 메뉴를 선택하고 **API 키** 항목의 **새로운 API 키 생성** 버튼을 클릭합니다.



- 팝업 창에 표시된 API 키 값을 복사한 후 안전한 곳에 보관해주세요. 생성된 API 토큰은 다시 조회할 수 없으며, 만약 토큰을 분실하였다면, **삭제** 버튼을 눌러 기존 API Key를 삭제한 후 다시 생성하여 이용해주세요.

▼ 배포하려는 특정 repository 만 선택



▼ 내 repository 연결



내 GitHub 저장소 배포하기

배포하길 원하는 GitHub 저장소를 선택하세요.

GitHub 저장소

nbcdocker ▼

cicd

설정 ▼

서비스 이름

서비스 이름은 호스트 이름으로 사용됩니다.

프로젝트 이름

nbc.doc... ▼

cicd

리전 선택

Seoul - Korea ▼

언어/프레임워크 선택

Spring Boot ▼

애플리케이션 설정

java@17

설정변경

버전

v17 ▼

Environment variables

Name	Value	

Port

- 서비스 이름과 프로젝트 이름은 기본을 설정해 주세요
- 언어/프레임워크 은 **Spring Boot** 을 설정해 주세요
- 애플리케이션 버전은 v17을 선택해 주세요
- 빌드를 시작하고 설정을 진행한 뒤 배포 실패가 나오는 것을 확인해요

▼ github 설정

- 앞서 생성한 GHP(GitHub Personal Token)을 GitHub Actions 을 적용하려는 저장소에 설정합니다. 저장소의 **Settings** > **Secrets** 페이지로 이동하고 **New repository secret** 을 클릭하고 아래와 같이 설정합니다.

The screenshot shows the GitHub Actions secrets page. On the left, a sidebar lists various settings categories, with 'Secrets' highlighted. The main area is titled 'Actions secrets' and includes a 'New repository secret' button. Below this, there are two sections: 'Environment secrets' and 'Repository secrets'. Both sections indicate that there are currently no secrets for the repository and provide a link to manage and add secrets.

- GitHub Personal token: **GHP_TOKEN**

Actions secrets / New secret

Name

GHP_TOKEN

Value

your_generated_token

Add secret

- cloudtype API key: `CLOUDTYPE_TOKEN`

Actions secrets / New secret

Name

CLOUDTYPE_TOKEN

Value








cloudtype_api_key

Add secret

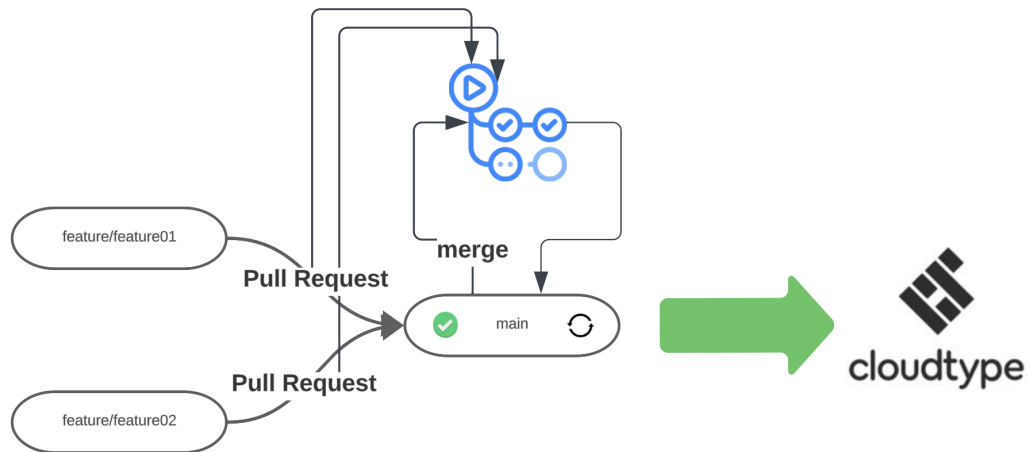
•

Repository secrets

New repository secret

Name 	Last updated	
 CLOUDTYPE_TOKEN	now	 
 GHP_TOKEN	1 minute ago	 

▼ github actions 설정



- `test-pr.yml`

```
name: test every pr
on:
  workflow_dispatch:
  pull_request:
permissions:
  contents: read
  pull-requests: read
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v3
      - name: setup jdk
        uses: actions/setup-java@v3
        with:
          java-version: '17'
          distribution: 'temurin'
          cache: gradle
      - name: Grant execute permission for gradlew
        run: chmod +x ./gradlew
```

```
- name: gradlew test
  run: ./gradlew test
```

- Pull Request가 생성되면, 'gradle test' 를 진행해서 PR 의 테스트코드가 모두 성공인지 확인


- `deploy-main.yml`

```
name: Deploy to cloudtype
on:
  push:
    branches:
      - main
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v3
      - name: Connect deploy key
        uses: cloudtype-github-actions/connect@v1
        with:
          token: ${ secrets.CLOUDTYPE_TOKEN }
          ghtoken: ${ secrets.GHP_TOKEN }
      - name: Deploy
        uses: cloudtype-github-actions/deploy@v1
        with:
          token: ${ secrets.CLOUDTYPE_TOKEN }
          project: # 워크스페이스 이름/프로젝트 이름 ex) nl
          stage: main
          yaml: |
            name: # 프로젝트 이름 ex) `cicd`
            app: java@17
            options:
              ports: 8080
            context:
              git:
                url: git@github.com:${ secrets.github.repos
```

```
ref: ${github.ref}
preset: java-springboot
```


- cloudtype 서비스 배포에 필요한 key들을 가지고 바로 배포

2주차 끝

 [전체 강의자료 보기](#)

 [Docker+CI/CD](#)

[다음 주차](#) 

 [Docker+CI/CD - 3주차](#)

Copyright © TeamSparta All rights reserved.