제 목	01장 CI/CD 기본 개념과 Github Actions	
상세내용	CI/CD 기본 개념 / Github Actions 기본 개념	

1. CI/CD를 왜 배우는 걸까?

✓ CI/CD란?

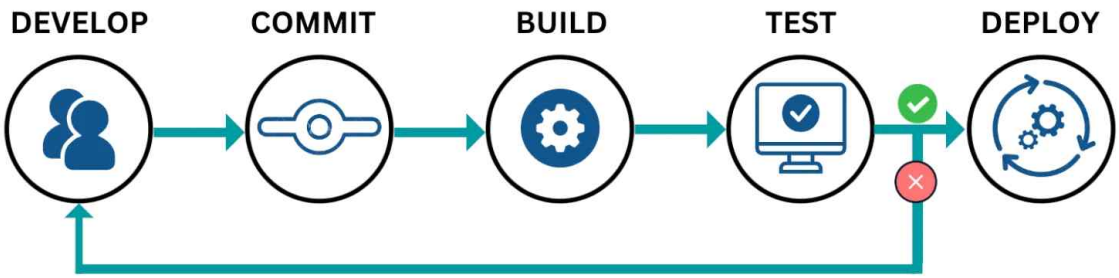
CI/CD란 Continuous Integration, Continuous Deployment라는 의미를 가지고 있다. 말이 너무 어렵다. 쉽게 표현하자면 **CI/CD는 테스트(Test), 통합(Merge), 배포(Deploy)의 과정을 자동화하는 걸 의미**한다.

▶ CI/CD를 왜 배우는 걸까?

서비스를 운영하다보면 새로운 기능을 추가하는 일이 많아진다. 새로운 기능에 대한 코드를 작성한 뒤에 Commit을 찍는다. 그런 뒤에 브랜치에 Merge를 하고 배포를 한다. 배포를 할 때 직접 컴퓨터 서버(ex. AWS EC2)에 접속해서 새로운 코드를 다운받아 실행시켜주어야 한다.

이 과정을 코드의 수정이 일어날 때마다 반복하기란 너무 귀찮은 일이다. 그래서 이런 **반복적인 과정을 자동화시키기 위해 CI/CD를 배우는 것**이다.

CI/CD 과정은 일반적으로 다음과 같은 과정으로 일어난다.



특정 기능을 개발 완료해서 Commit을 찍으면 빌드가 되게 셋팅한다. 빌드가 완료되면 작성한 테스트 코드를 실행시킨다.(테스트 코드를 작성하지 않은 서비스에서는 이 과정을 생략하기도 한다.) 그런 뒤 테스트가 통과하면 실제 서버 컴퓨터에 최신 코드가 배포된다.

2. CI/CD 구축할 때 사용할 Github Actions

✓ CI/CD를 구축할 때 사용할 툴

CI/CD를 구축할 수 있는 툴에는 여러가지가 있다.

- Github Actions
- Jenkins
- Circle CI
- Travis CI
- 등등

이 중에서 현업에서도 많이 사용하면서, 무료로 사용할 수 있고, 빌드용 서버가 따로 필요없는 Github Actions를 활용해서 CI/CD를 구축할 것이다.

✓ CI/CD에서 Jenkins를 활용하지 않아도 되나요?

현업에서 Github Actions 뿐만 아니라 Jenkins도 많이 활용한다. Github Actions와 Jenkins 둘 중에 하나만 쓰더라도 필요한 CI/CD 구성을 전부 할 수 있다. **<u>**Github Actions를 사용할 지, Jenkins를 사용할 지는 장단점을 비교해서 상황에 맞게 선택하면 된다**</u>.** 개인적으로 추천하는 툴은 **<u>**Github Actions**</u>이다.

Jenkins의 단점으로는 별도의 서버에 구축을 해야 한다는 단점이 있다. 이 때문에 서버를 빌리는 비용이 발생하게 된다. 하지만 Github Actions는 별도의 서버 구축 없이 Github에 내장되어 있는 Github Actions 기능을 사용할 수 있다. 비용적인 측면도 유리하고 셋팅하는 데 시간을 쓸 필요도 없다.

실제 현업에서도 정말 Jenkins를 안 쓰고 Github Actions를 쓰는지 확인하고 싶다면 아래 링크를 살펴보자.

▶ 카카오엔터프라이즈가 GitHub Actions를 사용하는 이유

3. Github Actions를 활용한 전체적인 CI/CD 흐름

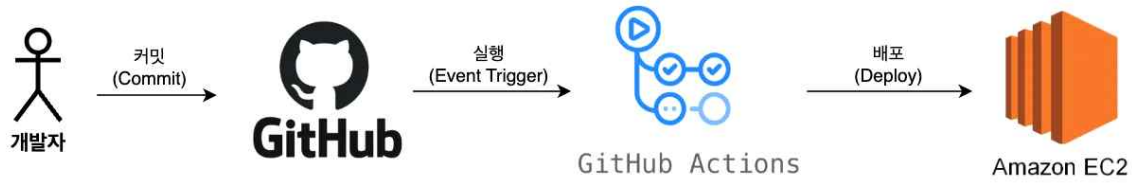
✓ CI/CD 흐름을 이해하기 위한 Github Actions 개념 정리

Github Actions를 로직을 실행시킬 수 있는 일종의 컴퓨터라고 생각하면 된다. CI/CD 과정에서 Github Actions는 "빌드, 테스트, 배포"에 대한 로직을 실행시키는 역할을 하게 된다.

▶ **Github Actions에 대한 이미지를 이 정도로 잡아두고, CI/CD 전체 흐름 살표보기**

✓ CI/CD 전체 흐름

CI/CD의 구성 방식은 다양하지만 일반적으로 아래의 흐름을 가진다.



1. 코드 작성 후 Commit
2. Github에 Push
3. Push를 감지해서 Github Actions에 작성한 로직이 실행
   a. 빌드(Build)
   b. 테스트(Test)
   c. 서버로 배포(Deploy)
4. 서버에서 배포된 최신 코드로 서버를 재실행

[실습] Github Actions 기본 문법 정리

✓ 처음으로 Github Actions 작동시켜보기

1. 새로운 프로젝트 폴더 만들기

2. .github/workflows/deploy.yml 만들기

```
# Workflow의 이름
# Workflow : 하나의 yml 파일을 하나의 Workflow라고 부른다.
name: Github Actions 실행시켜보기

# Event : 실행되는 시점을 설정
# main이라는 브랜치에 push 될 때 아래 Workflow를 실행
on:
  push:
    branches:
      - main

# 하나의 Workflow는 1개 이상의 Job으로 구성된다.
# 여러 Job은 기본적으로 병렬적으로 수행된다.
jobs:
  # Job을 식별하기 위한 id
  My-Deploy-Job:
    # Github Actions를 실행시킬 서버 종류 선택
    runs-on: ubuntu-latest

    # Step : 특정 작업을 수행하는 가장 작은 단위
    # Job은 여러 Step들로 구성되어 있다.
    steps:
      - name: Hello World 찍기 # Step에 이름 붙이는 기능
        run: echo "Hello World" # 실행시킬 명령어 작성

      - name: 여러 명령어 문장 작성하기
        run: |
          echo "Good"
          echo "Morning"
```

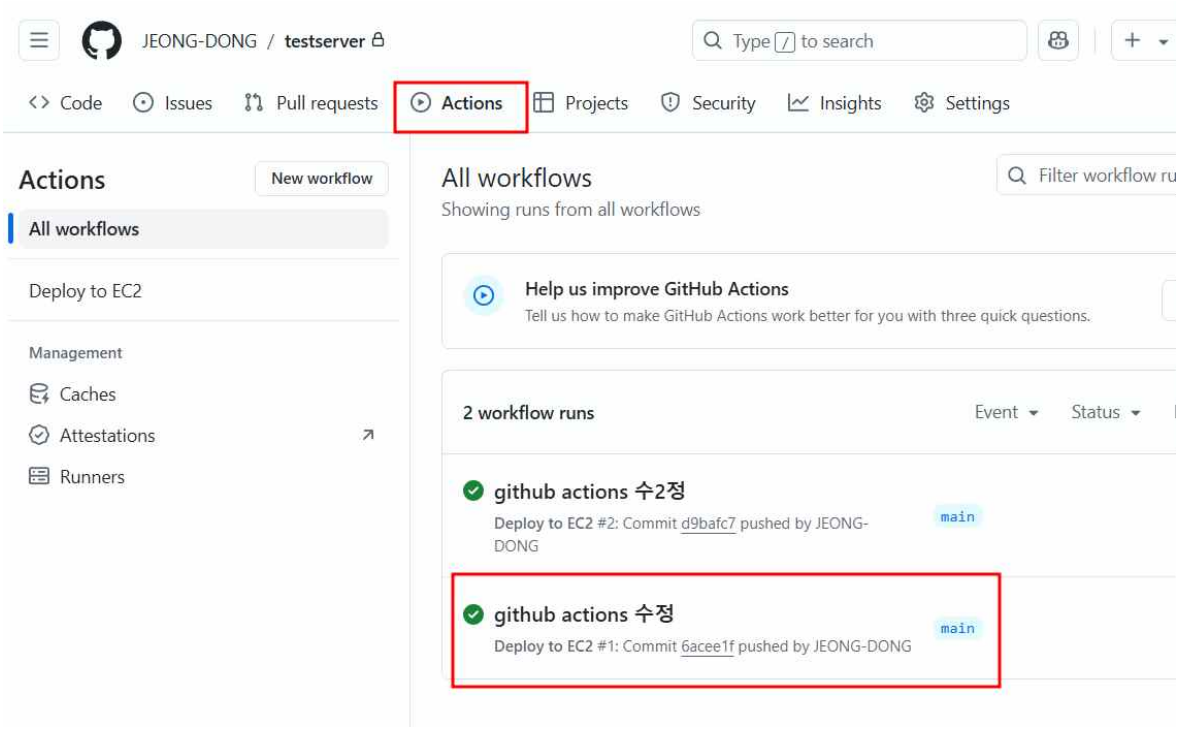
▶ Github Actions의 문법은 아래 문서에서 확인할 수 있다.

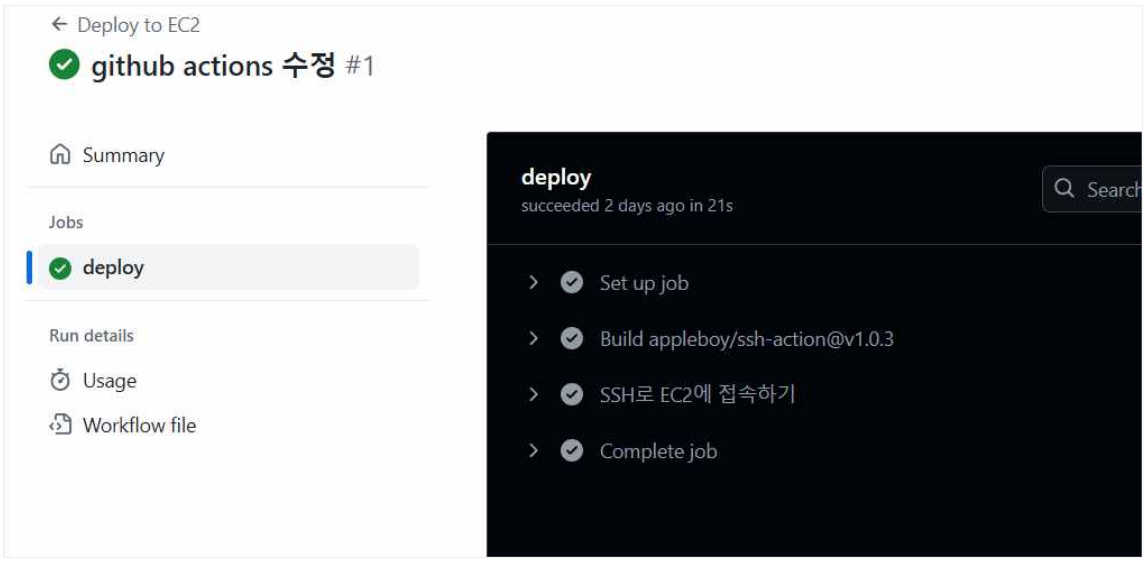
- Github Actions 설명서 : https://docs.github.com/ko/actions

3. Github Repository 만들어서 업로드하기

```
$ git init
$ git add .
$ git commit -m "first commit"
$ git branch -M main
$ git remote add origin {Repository 주소}
$ git push -u origin main
```

4. 업로드후 Github Actions에서 확인하기





5. 내용 추가후 COMMIT 실행

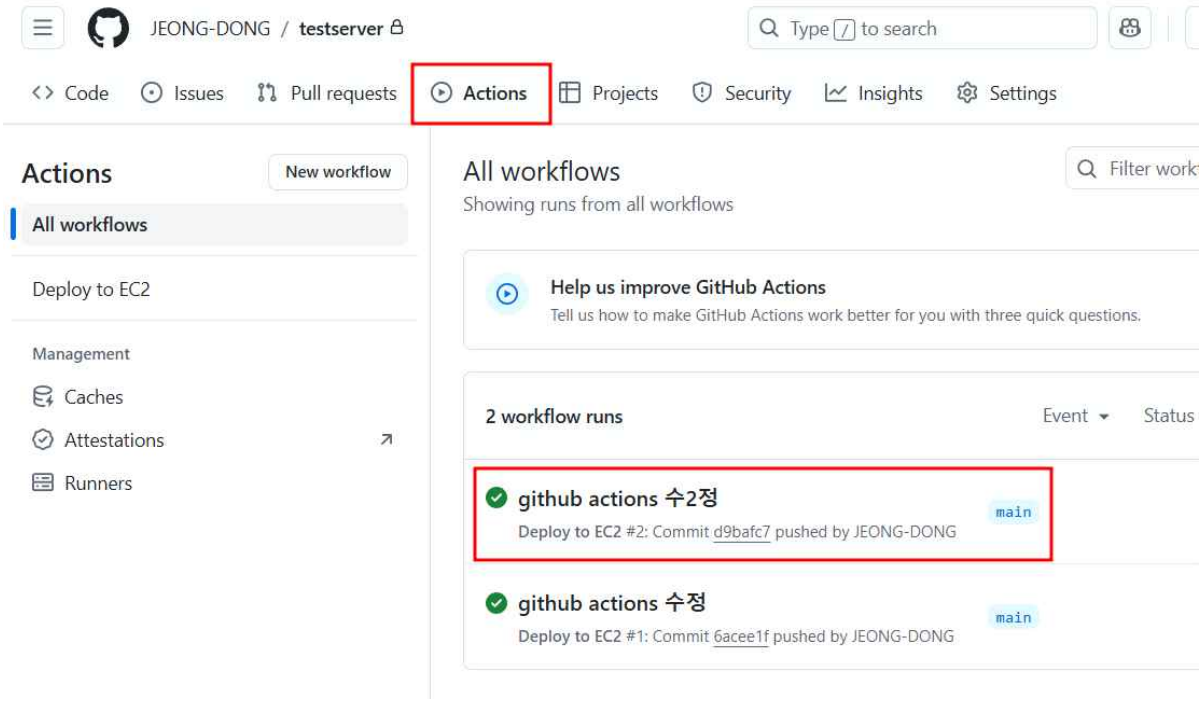
```
# Workflow의 이름
# Workflow : 하나의 yml 파일을 하나의 Workflow라고 부른다.
name: Github Actions 실행시켜보기

# Event : 실행되는 시점을 설정
# main이라는 브랜치에 push 될 때 아래 Workflow를 실행
on:
  push:
    branches:
      - main

# 하나의 Workflow는 1개 이상의 Job으로 구성된다.
# 여러 Job은 기본적으로 병렬적으로 수행된다.
jobs:
  # Job을 식별하기 위한 id
  My-Deploy-Job:
    # Github Actions를 실행시킬 서버 종류 선택
    runs-on: ubuntu-latest
   ~ ...
     ... ~

      # 참고: https://docs.github.com/en/actions/learn-github-actions/variables
      - name: Github Actions 자체에 저장되어 있는 변수 사용해보기
        run: |
          echo $GITHUB_SHA
          echo $GITHUB_REPOSITORY
      - name: Github Actions Secret 변수 사용해보기
        run: |
          echo ${{ secrets.MY_NAME }}
          echo ${{ secrets.MY_HOBBY }}
```

Github Actions를 실행시키기 위해서는 반드시 **.github/workflows**라는 디렉터리에 **.yml 또는 .yaml**의 확장자로 파일을 작성해야 한다. 그리고 **.github/workflows**는 프로젝트 폴더의 최상단에 만들어야 한다.

6. Github Repository 만들어서 업로드하기

```
$ git init
$ git add .
$ git commit -m "first commit"
$ git branch -M main
$ git remote add origin {Repository 주소}
$ git push -u origin main
```

7. git push후 Github Actions에서 확인하기



✓ Github Actions 전체 구조 확인하기