

2025 깃헙 스터디

서강대학교 Release 학회





수고하셨습니다~!

SGCS-Release-Git-Project / Git-Study

Code Issues Pull requests 5 Actions Projects Wiki Security Insights Settings

Filters is:pr is:open Labels 9 Milestones 0 New pull request

<input type="checkbox"/>	5 Open ✓ 20 Closed	Author	Label	Projects	Milestones	Reviews	Assignee	Sort
<input type="checkbox"/>	[math]8393-2025.4.30 #25 opened 5 days ago by hyeonwlee							2
<input type="checkbox"/>	[math]1000-2025.04.29 #24 opened last week by kyuhyunglee							
<input type="checkbox"/>	[math]8393-2025.4.29 #23 opened last week by yjs2673							1
<input type="checkbox"/>	[math]10998-2025.04.28 #21 opened last week by wjm9765							
<input type="checkbox"/>	[math]8393-2025.4.28 #19 opened last week by hyeonsang010716							

ProTip! Add [no:assignee](#) to see everything that's not assigned.

지난주도 고생하셨습니다~!



지금까지 저희는 Git Study에서

- 체계적인 소스 코드 저장(add, commit, push)
- 구조적인 협업 관리(branch, pull request)
- 협업 시, 충돌 관리(pull – rebase 0, rebase X)
- 버전 관리(tag)
- 로컬 깃 관리(restore, checkout, status)

위 내용들을 배웠습니다!



오늘 배울 내용은

- Github Issue 사용 방법
- Github에서 사용하는 파일(README.md, gitignore)
- Git action을 사용한 CI / CD 1

입니다.

다음 주가 Git Study 마지막입니다!

- Git action을 사용한 CI / CD 2

고지까지 얼마 안 남았습니다~! 모두 마지막까지 화이팅!

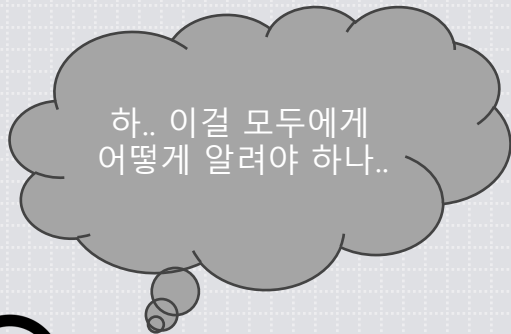


먼저! 지난 시간(스프린트4)까지 내용을 버전 관리 해볼까요?

Git tag를 사용해서 v1.0.0을 만들어 봅시다!

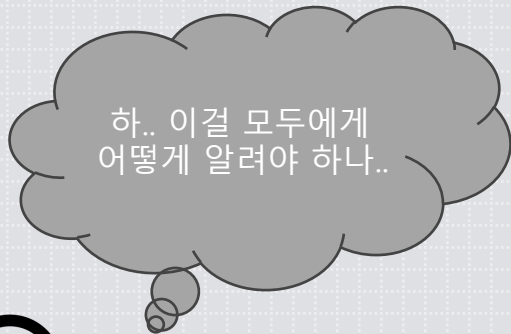


깃헙에서 이슈는 투명하게 공유 되어야 합니다!



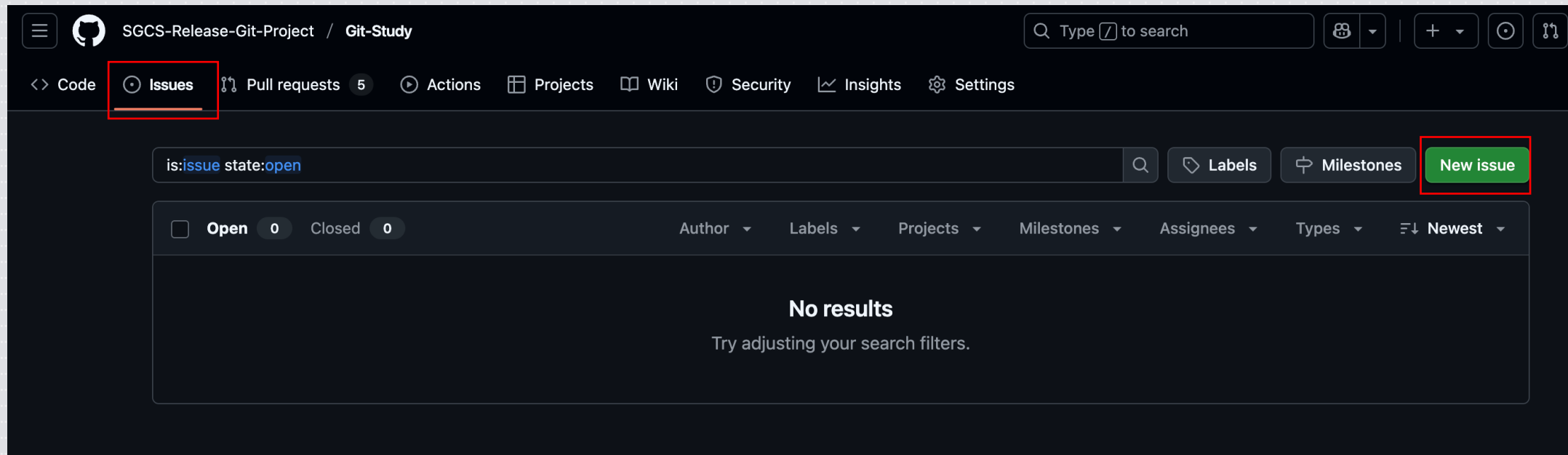


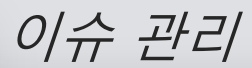
팀 규모가 커질수록 **모두**에게 **같은 내용**을 알리기는 쉽지 않죠





그래서 깃헙은 이슈를 공유하는 표준 규칙을 제공합니다!





SGCS-Release-Git-Project / Git-Study

Q Type / to search

<> Code

Issues

Pull requests 5

Actions

Projects

Wiki

Security

Insights

Settings

Create new issue

이슈 제목

Add a title *

Title

Add a description

이슈 내용

Write

Preview

H B I | = <> | : : : | @ ↗ ↶

Type your description here...

Paste, drop, or click to add files

Create more

Cancel

Create ↗ ↶

Assignees

No one - Assign yourself

Labels

No labels

Type

No type


Projects

No projects

Milestone

No milestone





Create new issue

Add a title *

Title


Add a description

Write

Preview

H B I | [List Icons] | [Link Icon] | [Quote Icon] | [Code Icon] | [Image Icon] | [Globe Icon] | [Share Icon] | [Close Icon]

Type your description here...

 Paste, drop, or click to add files

☐ Create more


Cancel


Create ↗ ↶


이슈와 관련된 사람 체크


Assignees


Assign up to 10 people to this issue


☐  hyeonsang010716


☐  bigwave100 김찬영


☐  HwangInseok Mango


☐  hyeonwlee hyeonw


☐  jungsb0415

☐  kyuhyunlee

☐  pinkgorae

☐  wjm9765

☐  xxxiv-34

☐  yjs2673 Yun Junseo



Create new issue

Add a title *

Title

Add a description

Write

Preview

H B I | [list icons] | [link icon] | [mention icon] [share icon] [undo icon] [redo icon]

Type your description here...

Paste, drop, or click to add files

☐ Create more

Cancel

Create ↩

Assignees

No one [Assign yourself](#)

이슈 종류 선택

Labels

Apply labels to this issue

☐ **bug**
Something isn't working

☐ **documentation**
Improvements or additions to documentation

☐ **duplicate**
This issue or pull request already exists

☐ **enhancement**
New feature or request

☐ **good first issue**
Good for newcomers

☐ **help wanted**
Extra attention is needed

☐ **invalid**
This doesn't seem right

☐ **question**
Further information is requested

☐ **wontfix**



README.md 는 해당 레포 디렉토리 가이드라인, 안내문을 작성할 때 사용합니다.

- README.md 는 마크다운 문법을 사용할 수 있음.
- 각 디렉토리마다 설정할 수 있음.



data-AI-competition-2024 / README.md

hyeonsang010716 Update README.md

Preview Code Blame 60 lines (41 loc) · 3.02 KB

```
1  ✓ # 국회 회의록 기반 RAG 시스템 채팅 서비스
2
3  이 프로젝트는 RAG (Retrieval-Augmented Generation) 시스템을 이용
4
5  ✓ ## 주요 기능
6
7  - **국회 회의록 데이터 질의응답** : 국회 회의록 데이터를 기반으로 자연어
8  - **RAG 기반 검색** : 문서 검색 및 답변 생성에 RAG 시스템을 사용하여 높은
9  - **Streamlit UI** : 사용자 친화적인 Streamlit 인터페이스로 쉽게 접
10
11  ✓ ## 설치 방법
12
13  로컬 환경에서 프로젝트를 실행하려면 다음 단계를 따르세요:
14
15  1. **레포지토리 클론** :
16  ```bash
```



README

국회 회의록 기반 RAG 시스템 채팅 서비스

이 프로젝트는 RAG (Retrieval-Augmented Generation) 시스템을 이용하여 국회 회의록 데이터를 기반으로 질의응답 서비스를 제공합니다. 사용자는 Streamlit을 통해 웹에서 직접 국회 회의록 데이터에 대한 질문을 하고, AI가 해당 질문에 답변을 제공합니다.

주요 기능

- 국회 회의록 데이터 질의응답: 국회 회의록 데이터를 기반으로 자연어 질문에 대한 답변을 생성합니다.
- RAG 기반 검색: 문서 검색 및 답변 생성에 RAG 시스템을 사용하여 높은 정확도를 제공합니다.
- Streamlit UI: 사용자 친화적인 Streamlit 인터페이스로 쉽게 접근할 수 있는 채팅 서비스입니다.

설치 방법

로컬 환경에서 프로젝트를 실행하려면 다음 단계를 따르세요:

1. 레포지토리 클론:

```
git clone https://github.com/hyeonsang010716/data-AI-competition-2024.git
```

.gitignore 는 깃헙에 올라가면 안 되는 소스 코드/파일들을 지정할 때 사용합니다.

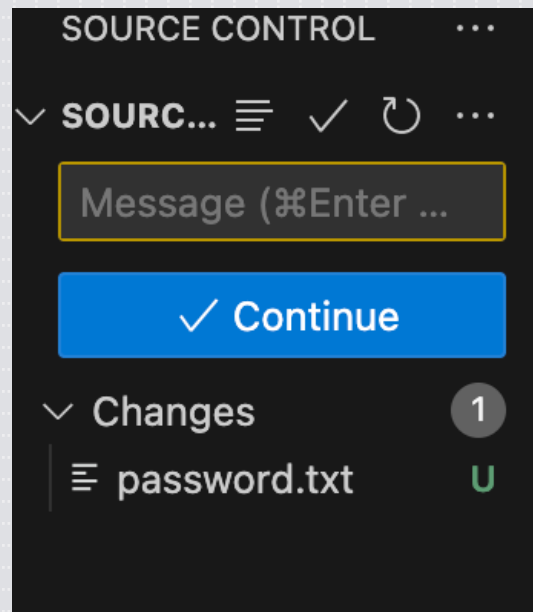
예) 사용자 개인정보 파일, 데이터베이스

한 개라도 올라가게 된다면.. 매우 큰일..!

.gitignore 을 사용하지 않는다면?

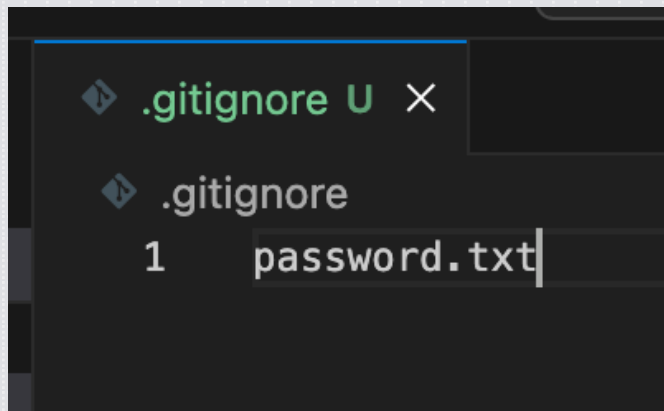
매번 add, commit, push 과정에서 하나 하나 체크 해줘야 함..

하지만.. 100% 실수를 하지 않는다는 보장이 없음.. -> 스트레스

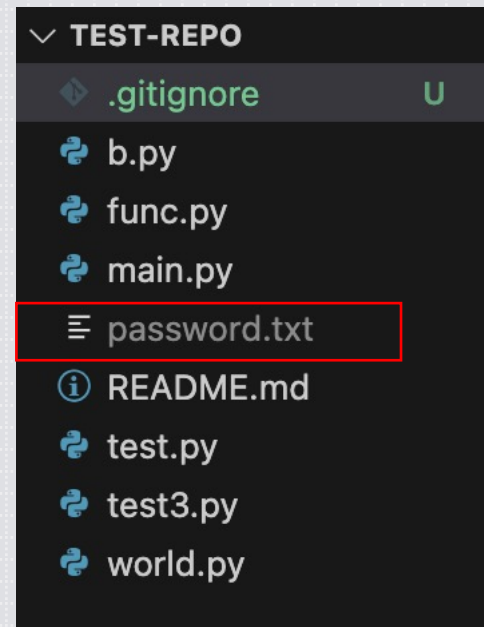
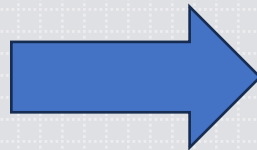


.gitignore 을 사용한다면?

절대 지정된 파일은 올라갈 일이 없음!



```
.gitignore U x
.gitignore
1 password.txt
```





.gitignore 사용 방법

#~ : 주석

*.txt : .txt 확장자 모두 무시

temp/ : 모든 위치의 temp 디렉토리 무시 (temp 안에 있는 소스 파일들 함께)

/특정경로/temp/ : 특정경로에 있는 temp 디렉토리 무시

!share.txt : 앞에서 *.txt로 모든 txt 파일이 못 올라가게 되었을 때, !을 사용하면 이 파일은 올라갈 수 있게 됨.



CI/CD 란 무엇인가?

CI : Continuous Integration – 지속적인 통합

CD : Continuous Deployment – 지속적인 배포



CI

안정적인 개발을 하기 위해선, 한 기능이 완성 될 때마다 테스트를 진행해야 합니다.

예를 들어

A기능 개발 완료 -> PR -> 테스트 -> 테스트 완료 -> 머지

[테스트 목록]

1.

2.

3.

4.

5.

....



CI

하지만.. 여러 명에서 개발을 하면.. 기능이 완성 될 때마다 테스트 해야 하는 문제!

[테스트 목록]

1.
2.
3.
4.
5.
....

[테스트 목록]

1.
2.
3.
4.
5.
....

[테스트 목록]

1.
2.
3.
4.
5.
....

[테스트 목록]

1.
2.
3.
4.
5.
....

[테스트 목록]

1.
2.
3.
4.
5.
....

[테스트 목록]

1.
2.
3.
4.
5.
....

코드 개발 보다 테스트 하는 과정이 더 시간이 **오래 걸림**.. 그리고 **매우 지루함**
자동 CI 프로세스가 없다면.. 노가다로 해결 해야 함..!



CD

CD도 CI랑 마찬가지로 AWS, Azure 등 클라우드로 배포할 때

기능 구현이 될 때마다 배포해야 되는데..

그러면

깃헙 코드 -> 이미지 생성 -> 클라우드 전송 -> 기존 이미지 교체

위 과정을 계속해서 반복해야 함..



그래서 등장한! 자동 CI / CD 도구!! (역시 개발자 세계에는 존재하지 않는 건 없다..)

대표 적으로 **젠킨스**, **깃액션** 등이 있다.

그 중에서도 우리는 깃액션을 사용해볼 것!

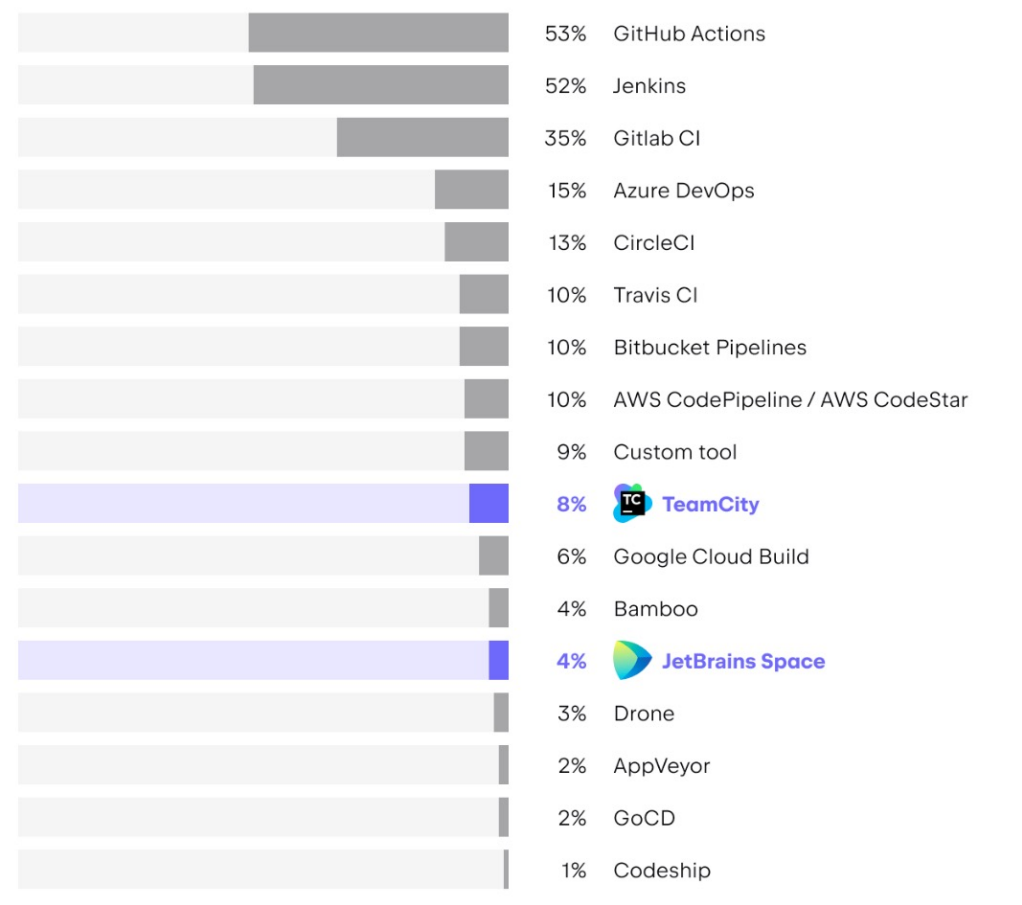


Jenkins



GitHub Actions

Git actions는 다른 CI/CD 도구 보다 문법이 간단하기 때문에 입문하기 좋음!



★ Jenkins 과 GitHub Actions 차이점

Jenkins	GitHub Actions
서버 설치 필요	클라우드가 있으므로, 별도 설치 필요없음
작업 또는 작업이 동기화되어 제품을 시장에 배포하는 데 더 많은 시간이 소요	비동기 CI / CD 달성
계정 및 트리거를 기반으로하며 github 이벤트를 준수하지 않는 빌드를 중심으로합니다.	모든 github 이벤트에 대한 작업을 제공하고 다양한 언어와 프레임 워크를 지원합니다.
환경 호환성을 위해 도커 이미지에서 실행해야 함	모든 환경과 호환
캐싱 메커니즘을 지원하기 위해 플러그인을 사용할 수 있습니다.	캐싱이 필요한 경우 자체 캐싱 메커니즘을 작성해야 합니다.
공유 할 수있는 능력이 없습니다.	github 마켓 플레이스를 통해 공유 가능
전세계 많은 사람들이 이용하여 문서가 다양	젠킨스에 비해 문서가 없음
페이스북, 넥플릭스, 쿠팡, 카페24, 11번가 등...	업스테이지 AI, Be pro 회사 등...



Docker

Server



Push

CI/CD

성공/실패

Docker, Jenkins, Sever.. 다양한 선례 지식 필요



GitHub Actions

Github



Push

CI/CD

성공/실패

Github에서 제공한 문법만 알면,
나머지는 Github가 처리함



Git action을 사용하기 위해선, `.github/workflows` 디렉토리를 생성해줘야 합니다.

그리고 `~.yaml` 파일을 생성합니다.

```
✓ .github/workflows
```

```
! test.yaml
```





먼저 간단히 Git action 문법 살펴보기!

```
1  name: pull-request-workflow
2  on:
3    pull_request:
4      types: [opened]
5
6  jobs:
7    pull-request-job:
8      runs-on: ubuntu-latest
9      steps:
10     - name: step1
11       run: echo hello world
12     - name: step2
13       run: |
14         echo line1
15         echo line2
```

name : 현재 만들려는 Git action 이름을 정해줍니다.
(변수명과 같음!)

```
1  name: pull-request-workflow
2  on:
3    pull_request:
4      types: [opened]
5
6  jobs:
7    pull-request-job:
8      runs-on: ubuntu-latest
9      steps:
10     - name: step1
11       run: echo hello world
12     - name: step2
13       run: |
14         echo line1
15         echo line2
```

on : Git action 명령어가 어느 이벤트에 트리거 될 건지 정해줍니다.

대표적으로 pull_request, issues, push 가 있음.

pull_request와 issues는 types를 지정해줘야 함.

[opened] : PR, 이슈가 오픈 될 때 트리거 할 것임

[closed] : PR, 이슈가 종료 될 때 트리거 할 것임

[reopened] : .. 재오픈 할 때 트리거 할 것임

이외에도 많이 있음!

```
1  name: push-workflow
2  on: push
```

push 같은 경우에는 types가 필요 없음



```
1  name: pull-request-workflow
2  on:
3    pull_request:
4      types: [opened]
5
6  jobs:
7    pull-request-job:
8      runs-on: ubuntu-latest
9      steps:
10     - name: step1
11       run: echo hello world
12     - name: step2
13       run: |
14         echo line1
15         echo line2
```

jobs : 이벤트가 실행이 됐을 때, 어떤 명령어를 실행할 것인지 정의

```
1  name: pull-request-workflow
2  on:
3    pull_request:
4      types: [opened]
5
6  jobs:
7    pull-request-job:
8      runs-on: ubuntu-latest
9      steps:
10     - name: step1
11       run: echo hello world
12     - name: step2
13       run: |
14         echo line1
15         echo line2
```

pull-request-job : 실행되는 명령어 이름 (변수랑 같은 개념)

runs-on : 명령어 실행 환경 (ubuntu-latest : 우분투 환경으로 실행함)

steps: 명령어 실행 순서를 의미함

- name : 해당 명령어 라인 이름
- run : 실행할 명령어 (만약 여러 줄로 실행할 경우 | 를 사용해야 함)

(echo 는 터미널에서 문자를 찍을 때 사용하는 명령어)

<> Code

Issues 3

Pull requests 2

Actions

Projects

Wiki

Security

Insights

Settings

← pull-request-workflow

✓ [FEAT] pull request git action #1

⋮

Summary

Jobs

✓ pull-request-job

Run details

Usage

Workflow file

Triggered via pull request last month

👤 hyeonsang010716 opened #1

test/pull_request

Status

Success

Total duration

8s

Artifacts

—

pull_request.yaml

on: pull_request

✓ pull-request-job

3s

⌵

—

+

← pull-request-workflow

✓ [FEAT] pull request git action #1

🏠 Summary

Jobs

✓ pull-request-job

Run details

🕒 Usage

📄 Workflow file

pull-request-job

succeeded on Apr 1 in 3s

🔍 Search logs



> ✓ Set up job

0s

▼ ✓ step1

0s

1 ▶ Run echo hello world
4 hello world

▼ ✓ step2

0s

1 ▶ Run echo line1
5 line1
6 line2

> ✓ Complete job

0s



실습(4주차와 동일)

1. 간단한 알고리즘을 가지고 깃 협업 해보기!
2. 주의사항) 커밋 컨벤션, 브랜치 네이밍 컨벤션, 풀리퀘 컨벤션 맞추기 (실습 목적)

- FEAT: 새로운 기능 추가
- FIX: 버그 수정
- DOCS: 문서 수정
- STYLE: 스타일 관련 기능(코드 포매팅, 세미콜론 누락, 코드 자체의 변경이 없는 경우)
- REFACTOR: 코드 리팩토링
- TEST: 테스트 코드 추가
- CHORE: 빌드 업무 수정, 패키지 매니저 수정(ex .gitignore 수정 같은 경우)

커밋 컨벤션: [FEAT] , [MOD] , [FIX] ~

브랜치 컨벤션: 깃헙 계정 아이디/백준 문제 유형-백준 문제 번호
(만약 hyeonsang010716이 math 유형 백준 1000번을 풀었을 경우 → hyeonsang010716/math-1000)

풀리퀘 컨벤션: [백준 문제 유형]문제 번호-최초 풀리퀘를 보낸 날짜
(만약 2025.3.16일날 math 1000번을 풀어서 풀리퀘를 보낼 경우 → [math]1000-2025.3.16)

그리고 풀리퀘 상세 설명에 성공한 화면 사진 붙여넣기

(+ dev 브랜치에 풀리퀘를 보내기!)