

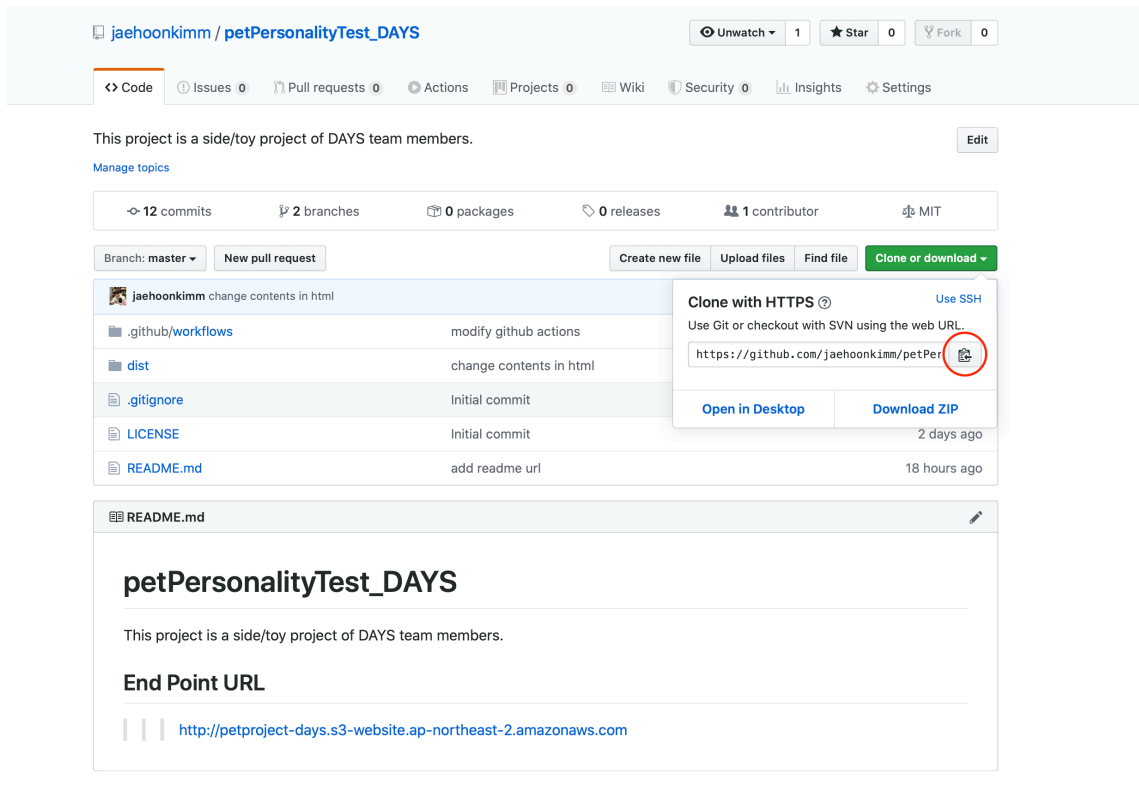
DAYS 프로젝트를 위한 Git&Github 사용해보기

준비물 : VSCode(Visual Studio Code), Git, Github 계정(은 모두 있으실거예요)

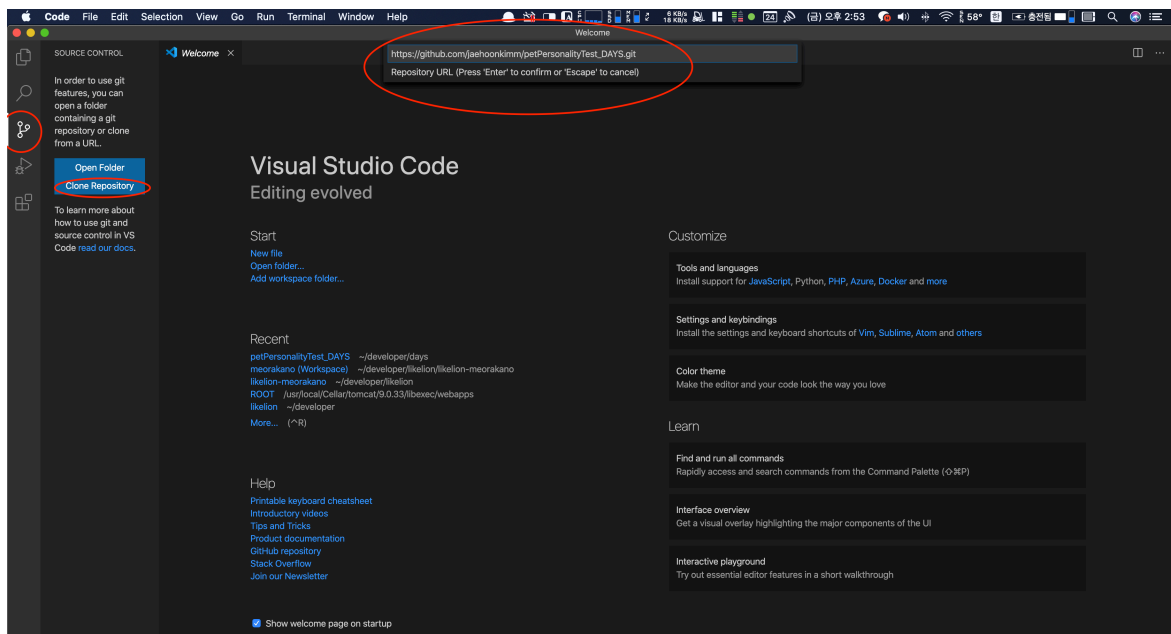
1. VSCode, Git 설치하기

1. <https://archmond.net/?p=9587> 링크 참고하여 설치
2. * 참고자료 <https://pmadviser.tistory.com/29> (깃 설명이 잘 나와있어요)

2. Github 프로젝트를 Local Repository(내 컴퓨터의 저장소)로 Clone(복사) 해오기

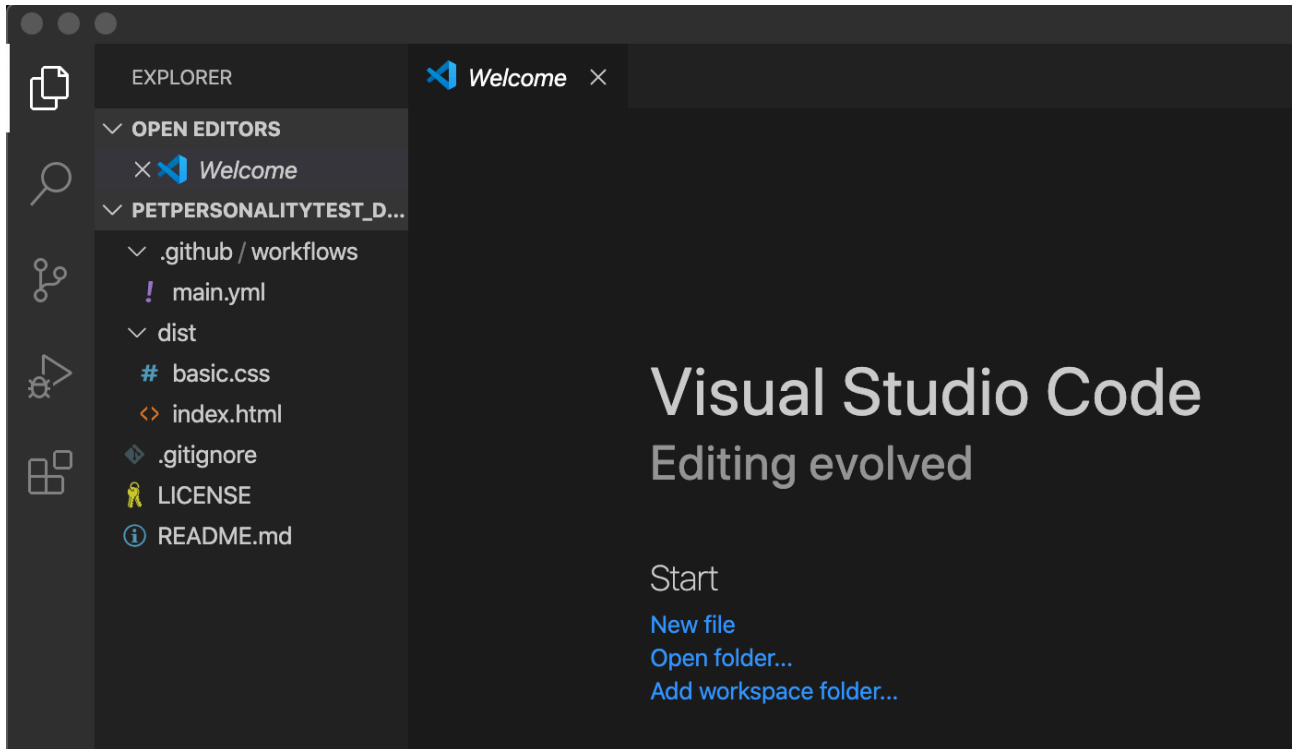


Github 프로젝트 페이지에서 초록색 버튼(Clone or download)를 누른 후에 해당 프로젝트의 주소를 복사합니다(빨간색 동그라미 쳐진 버튼)



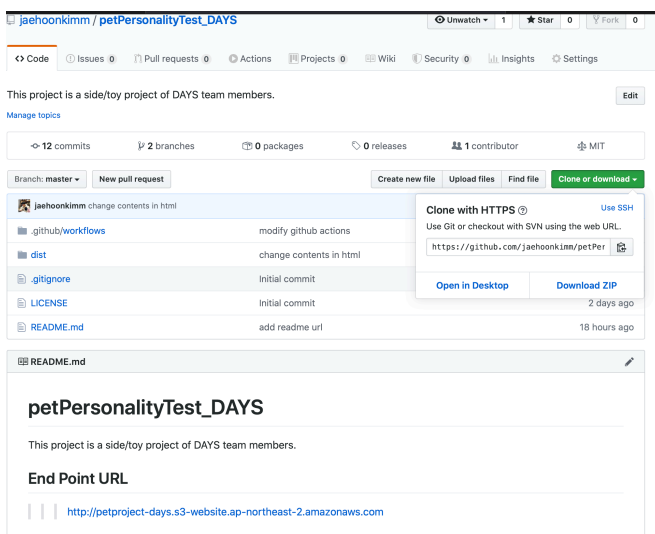
VScode를 켜서 맨 좌측 메뉴 5개 중 가운데(3번째) 메뉴를 누르고, clone repository 버튼을 누른 후 상단에 주소창이 뜨면 복사한 github 주소를 붙여넣고 엔터를 누릅니다.

그럼 파일 탐색기 창이 뜰텐데, 적당한 폴더를 지정한 후 확인을 눌러서 github 프로젝트 파일을 해당 폴더로 복사합니다.



제대로 Clone이 되었다면, 아마 이미지에 보이는 탐색창과 같이 petpersonalitytest_days 폴더가 복사되었고, LICENSE 파일과 README.md 파일, 그리고 이 안에 .github/workflows 폴더와 그 안에 main.yml, 그리고 dist 폴더 안에 basic.css와 index.html 파일이 들어있을겁니다.

README.md 파일은 마크다운 문법(md)로 작성된 텍스트 파일로, GITHUB 프로젝트 화면 아래 설명을 나타내는 문서입니다.



왼쪽 이미지 아래에 보면 README.md라고 적혀있고, 그 아래에 readme.md 문서에 적힌 프로젝트 이름, 프로젝트 웹페이지 url 등이 나와있네요.

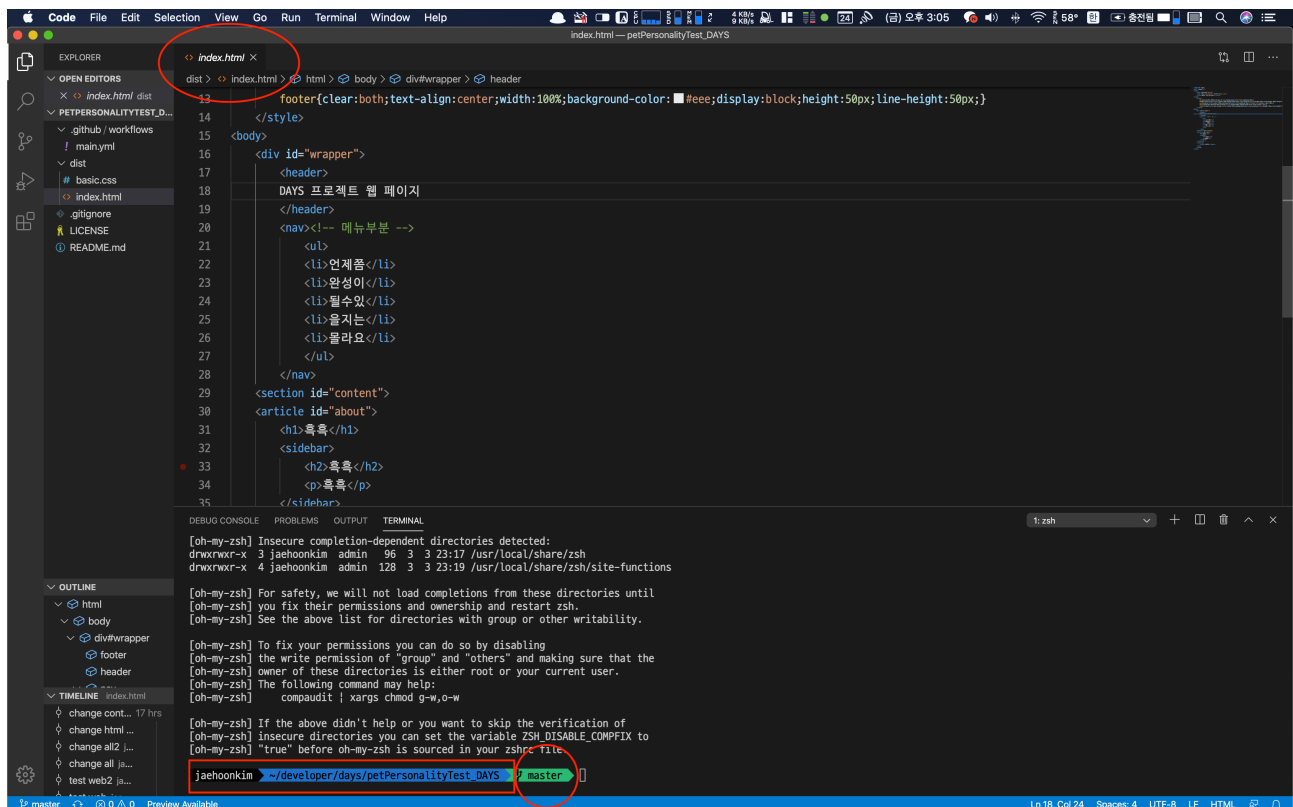
.github / workflows 경로 안에 든 main.yml 파일은 github에 push(업로드)된 것을 자동으로 감지하여 AWS(Amazon web service)에 연동시키고, 이를 웹페이지(readme.md에 적혀있는 End Point URL)에 자동으로 업로드 시켜주는 Github Actions 기능과 관련된 코드 파일로, 수정하여 github에 push 하시게 되면 해당 기능이 멈추게 되니 절대 수정하시면 안 됩니다!

index.html 파일은 EndPoint URL (<http://petproject-days.s3-website.ap-northeast-2.amazonaws.com>) 의 첫 화면을 구성하는 html 파일이고, basic.css 파일은 이 html 파일과 연결되어 화면의 style(폰트, 폰트 사이즈와 컬러, 화면 레이아웃의 여러 스타일들(디자인)을 변경해주는 css 파일입니다.

여러분이 html 파일의 각종 내용을 수정하고, 이것을 git을 통해 github에 업로드 하신다면 10초~1분 뒤에 실제로 반영된 모습을 EndPoint URL에서 보실 수 있습니다.

3. 실제로 변경, 적용해보기

VSCode로 돌아와서, 상단 메뉴에서 View(보기) - Terminal을 눌러주시면 VSCode 하단에 터미널이 나타나게 됩니다. 여기서 Git과 관련된 각종 명령을 내릴 수 있고, Mac 사용자는 맥에 내장된 Terminal(Ctrl+C 또는 Command+C로 Spotlight 검색을 킨 후 terminal 입력)로도 똑같은 활용이 가능합니다.



VSCode에서 Local(내 컴퓨터)에 저장된 index.html 파일을 실행시켜보았습니다.

아래 Terminal 창을 보시면 현재 html 파일이 있는 경로까지 접근되어 있고(파란색), master 브랜치(초록색)가 할당되어 있는 것을 알 수 있습니다. master 브랜치란, git과 관련된 용어로, 간단히 개념을 설명 드리면 다음과 같습니다.

Git은 버전관리 도구로, 우리가 코드를 수정한 내역을 모두 저장하고, 때로는 다시 원래대로 돌리고, 다른 사람과 함께 작성한 코드가 충돌되지 않도록 하는 여러가지 기능을 가지고 있습니다. 자세한건 예전에 드렸던 Git & Github 관련 링크로 익혀주세요.

어쨌든, 현재 master 브랜치, 즉 우리가 수정하는 내용이 전부 바로바로 반영되도록 설정되어 있으므로 개인 Branch를 하나씩 만들어줘야합니다. 우리가 수정한 내용을 바로 master로 반영해버리면 (master로 push), 큰 문제가 생깁니다! 우리가 수정한 내용을 미처 검토도 안 하고 바로 반영하는 것이기 때문

에, 만약 잘못 수정하거나, 팀원과 똑같은 내용을 수정해서 코드를 충돌시켜버리면 사용자들이 서비스를 사용 못 하거나, 서비스에 치명적인 문제가 생길 수 있습니다.

따라서 Branch(나뭇가지)를 생성해서 작업한 후에, 나중에 Master Branch와 병합 (Merge)하는게 좋은 방법입니다. 키워드의 의미와 똑같이, 나무의 큰 메인 줄기는 Master, 그리고 우리가 생성하는 개별적인 Branch들은 메인 줄기인 Master에서 뻗어나온 가지들이라고 연상하시면 됩니다.

그럼 이제 Branch를 만들어봅시다.

```
jaehoonkim ~/developer/days/petPersonalityTest_DAYS master git remote -v
origin https://github.com/jaehoonkimm/petPersonalityTest_DAYS.git (fetch)
origin https://github.com/jaehoonkimm/petPersonalityTest_DAYS.git (push)
jaehoonkim ~/developer/days/petPersonalityTest_DAYS master git pull
From https://github.com/jaehoonkimm/petPersonalityTest_DAYS
* [new branch]      chajuhui123 -> origin/chajuhui123
Already up to date.
jaehoonkim ~/developer/days/petPersonalityTest_DAYS master
```

git remote -v 를 terminal에 입력하면, 현재 연결된 주소를 확인할 수 있습니다. Github 프로젝트 주소가 뜨는걸 보면서, 잘 연결된걸 볼 수 있습니다.

git pull은 현재까지 github 저장소에 업로드 된 내용을 모두 local 저장소로 가져와서, 최신화하는 명령어입니다. 꼭 작업 시작 전에 git pull을 해줘야 합니다. (pull을 땡기고 시작해라, 라고도 합니다.) 안 그럼 팀원이 이미 작업한 내용을 또 수정하는 삽질을 하게 될 수 있습니다!! git pull 하기 전엔 서버에 최신화된 내용이 내 컴퓨터로 가져와져있지 않으니깐요.

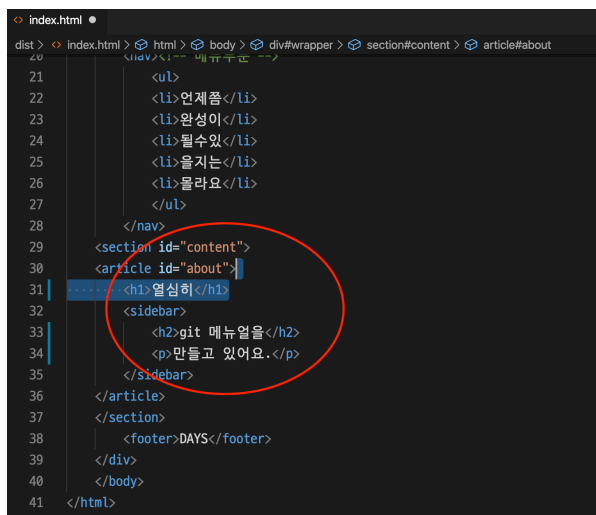
git pull 한 내용에 보시면 새롭게 서버에 반영된 정보가 있네요. 바로 Chajuhui님이 새 브랜치를 생성했던 내용입니다. 서버에만 저장되어 있던 "차주희 님의 새 브랜치 정보"가 git pull 명령어를 통해 이제 우리 Local 저장소에도 들어오게 된 것입니다.

이제 우리도 차주희 님처럼 새 브랜치를 만들어봅시다.

git checkout -b jh_branch 명령어를 통해 새 브랜치를 만들 수 있습니다.

```
jaehoonkim ~/developer/days/petPersonalityTest_DAYS master git checkout -b jh_branch
Switched to a new branch 'jh_branch'
jaehoonkim ~/developer/days/petPersonalityTest_DAYS jh_branch
```

새 브랜치가 생성되고, 자동으로 jh_branch로 전환이 되었습니다. 이제 VSCode에서 html의 내용을 수정해봅시다.



```
21 <ul>
22 <li>연제품</li>
23 <li>완성이</li>
24 <li>될수있</li>
25 <li>을지는</li>
26 <li>올라요</li>
27 </ul>
28 </nav>
29 <section id="content">
30 <article id="about">
31 <h1>열심히</h1>
32 <div>
33 <h2>git 메뉴얼을</h2>
34 <p>만들고 있어요.</p>
35 </div>
36 </article>
37 </section>
38 <div>
39 <div>
40 <div>
41 </div>
```

원래 '흑흑' '흑흑' '흑흑'으로 들어가있던 html의 h1, h2, p 태그의 내용을 열심히, git 메뉴얼을, 만들고 있어요. 라는 내용으로 바꿨습니다.

수정 후에는 Ctrl+S 로 꼭 문서를 저장해주세요. 안 그러면 git에서는 문서 수정한 내용을 인식하지 못 합니다.

그럼 이제 다시 Terminal에서 git 작업을 할 시간입니다.

```

jaehoonkim ~/developer/days/petPersonalityTest_DAYS jh_branch git add .
jaehoonkim ~/developer/days/petPersonalityTest_DAYS jh_branch +

```

git add . 을 치니, jh_branch 앞에 +가 붙었습니다.

git add는 나의 local 저장소에 수정된 내용을 git에 인식하게끔 하고, '.' 닳은 우리의 git 경로 (petPersonalityTest_DAYS)의 모든 파일의 수정 내용을 반영하라는 의미입니다.

git add index.html 처럼 특정 파일을 지정해서 add 할 수도 있습니다.

```

jaehoonkim ~/developer/days/petPersonalityTest_DAYS jh_branch + git commit -m "modify test for git manual"
[jh_branch 6d94119] modify test for git manual
1 file changed, 3 insertions(+), 3 deletions(-)

```

git commit -m "커밋 메시지" 를 통해서 commit을 해줍니다. 커밋은 버전을 만드는 것으로, 즉 내가 지금까지 수정한 내용의 버전(html의 h1, h2, p 태그 내용을 바꿈)을 git에 기록해서, 나중에 이 수정한 내용의 버전으로 언제든 돌아갈수도 있고, 또 이 버전에 해당하는 수정 내용을 확인 할 수 있도록 기록을 남기는 것입니다.

"" 쌍따옴표 안의 커밋 메시지를 통해 팀원과 본인이 어떤 내용이 수정되었는지 손쉽게 알아볼 수 있도록 기록합니다. 커밋 메시지는 관사 등 문법의 상당수를 생략하고 최대한 간소하게 작성하는게 좋습니다.

```

jaehoonkim ~/developer/days/petPersonalityTest_DAYS jh_branch git push origin jh_branch
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 445 bytes | 445.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'jh_branch' on GitHub by visiting:
remote:   https://github.com/jaehoonkimm/petPersonalityTest_DAYS/pull/new/jh_branch
remote:
To https://github.com/jaehoonkimm/petPersonalityTest_DAYS.git
* [new branch]      jh_branch -> jh_branch

```

마지막으로 git push origin jh_branch 명령어를 통해 서버로 push(업로드) 해줍니다. jh_branch가 새로 생성된 것이므로 처음 한번은 이 명령어를 써주고, 다음부터는 git push만 써도 됩니다.

그럼 이제 github로 가볼까요?

jaehoonkimm / petPersonalityTest_DAYS

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Actions Projects 0 Wiki Security 0 Insights Settings

This project is a side/toy project of DAYS team members. [Manage topics](#)

12 commits 3 branches 0 packages 0 releases 1 contributor MIT

Your recently pushed branches:

- jh_branch (1 minute ago) [Compare & pull request](#)

아래 노란색 박스에 jb_branch가 새로 push 했다고 알려주는 내용이 생겼습니다.

또, 중간에 보이는 3 branches도 기존에는 master 브랜치와, chajuhui 브랜치만 있어서 숫자가 2였는데, 3으로 늘어났습니다.

또한 3 branches 좌측의 12 commits도 지금까지 commit 내용을 모두 보여주는데, 조금 전 제가 커밋한 내용 [git commit -m "modify test for git manual"] 을 담고 있을겁니다.

그럼 이제 수정한 내용을 master branch와 merge(병합)해서 실제 반영을 해줘야겠죠? 노란 박스 오른쪽에 있는 compare & pull request 초록 버튼을 누릅니다.

그리고 적당히 내용을 적어주고 Create Pull Request를 눌러요.

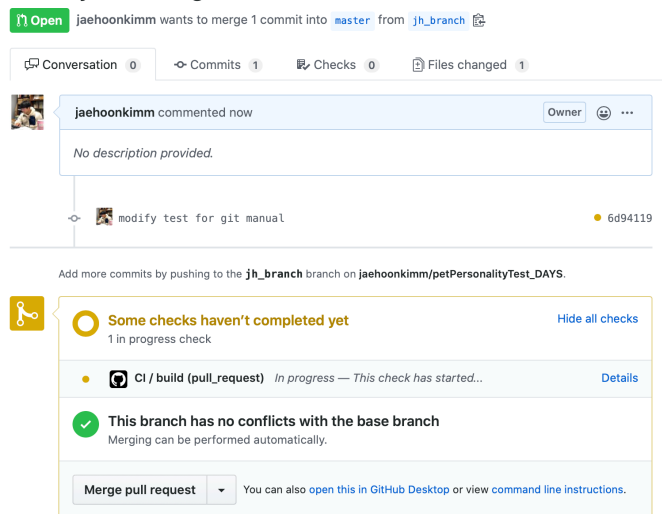
풀 리퀘스트란, 내가 작업한 내용을 Main Branch인 Master에 병합할 것을 검토하고, 병합해달란 요청입니다. 현재 제가 팀원 모두를 공동 작업자로 github에 초대해서 누구나 Master에 Merge 할 수 있는 권한이 있지만, 실제로 많은 깃허브 프로젝트는 관리자가 별도로 있습니다.

수십, 수백, 많으면 수천명이 되는 모든 사람을 공동 작업자로 등록할 수도 없고, 확인되지 않은 사람에게 공동 작업자 권한을 주면 악의적인 의도를 가지고 프로젝트를 망쳐버릴 위험도 있습니다.

따라서, 일반적으로 기여자가 개인 브랜치에서 코드를 작성하고 add, commit, push 후 Pull request 요청을 하면, 프로젝트 매니저 또는 관리자가 확인한 후에, 코드에 이상이 없으면 요청을 수락(Merge)하여 Master Branch로 코드를 병합해줍니다. 그럼 이제 반영되는거죠.

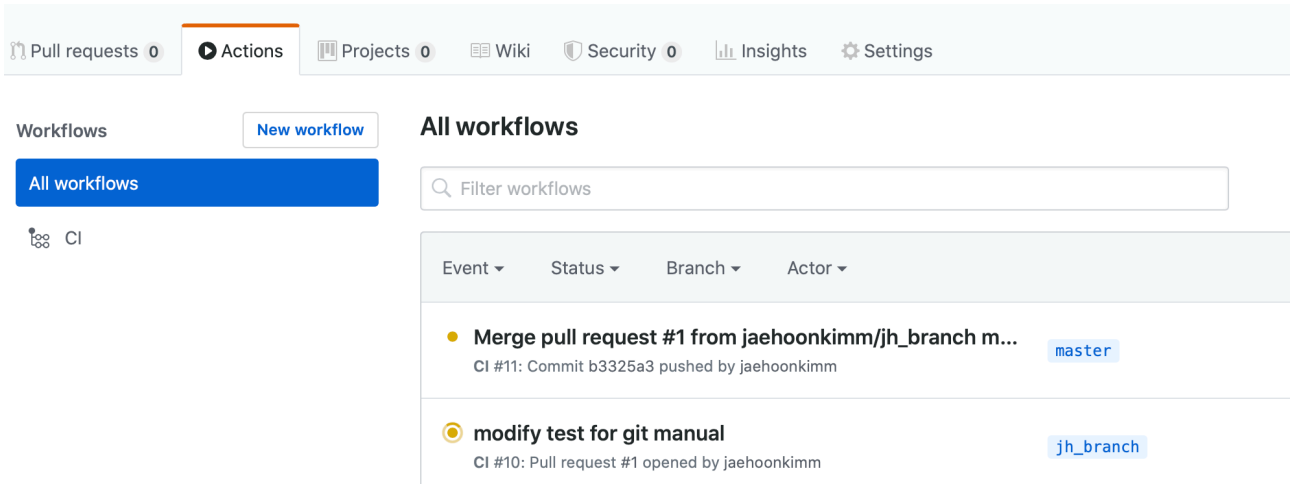
그럼 풀 리퀘스트를 create 했다면, 이제 우리는 모두 공동 작업자니까 바로 Merge할 권한이 있습니다.

modify test for git manual #1



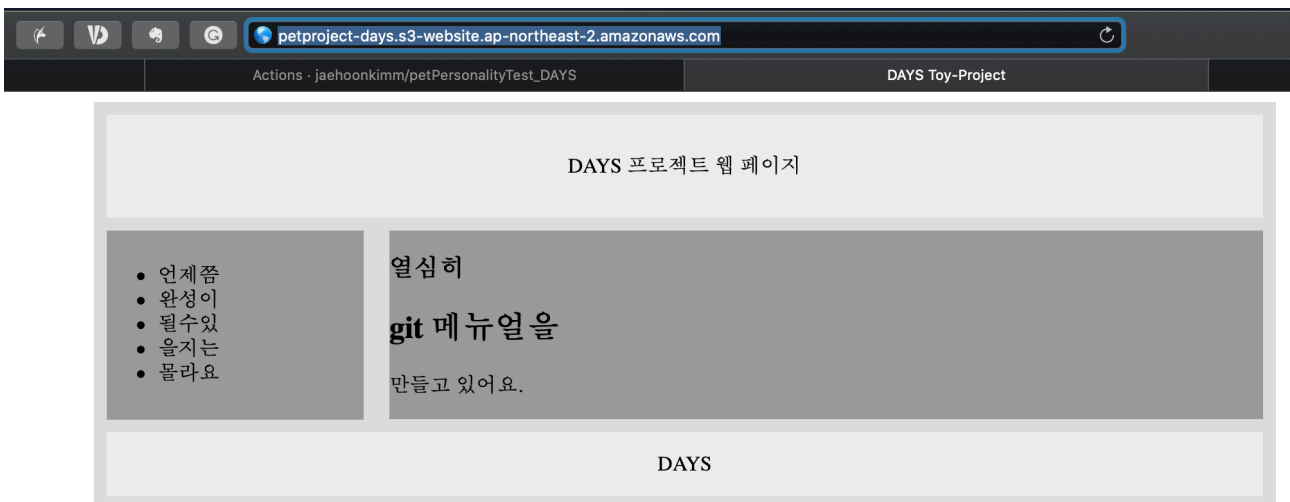
새로 뜬 창의 하단에 있는 Merge pull request 버튼을 눌러서, 우리가 개인 Branch에서 작업한 내용을 병합해줍니다.

Confirm merge를 한번 더 누르면 최종적으로 merge가 승인되고 병합 작업이 시작됩니다.



이제 Github 프로젝트 상단 메뉴에서 Actions을 눌러보면, 제가 작업해놓은 Github Actions 코드가 동작하여 우리 EndPoint URL의 웹페이지를 구성하는 Amazon Web Service로 html, css 파일이 자동 전송되어 곧 URL의 웹페이지에 반영되게 됩니다.

저 merge pull request 요청에 해당하는 master branch의 actions 요청의 노란 동그라미가 모두 완료되어 초록색으로 바뀐다면, 이제 AWS 서버에 모두 적용된 것입니다. 한번 End Point URL로 가볼까요? [<http://petproject-days.s3-website.ap-northeast-2.amazonaws.com>]



우리가 수정한 HTML 태그 내용이 실제 웹페이지로 반영되었습니다!

이제 언제 어디서든 우리는 저 URL로 우리가 만든 페이지에 들어갈 수 있는 것입니다. 다른 사람도 마찬가지구요. 축하합니다. 여기까지 따라 오셨다면, 이제 여러분은 Git과 Github를 사용할 수 있는 Git 사용자입니다!

jaehoonkim.dev@gmail.com
<https://github.com/jaehoonkimm>
 hy.days.official@gmail.com