

원격 저장소 활용하기
(pull, push)

원격 저장소 활용하기 - 1

- pull

원격의 커밋을 가져오기 위한 명령

- push

원격의 커밋을 올리기 위한 명령

커밋 이후 `git push origin master`를 입력하면 내가 수정한 코드가 깃허브에 올라감

원격 저장소 활용하기 - 2

- pull 사용해보기

깃허브 저장소 → index.html 파일 클릭 → 우측 상단 연필 아이콘 클릭



The screenshot shows the GitHub interface for a file named `index.html`. The top bar indicates the file is 12 lines (12 loc) and 284 Bytes. The right sidebar contains icons for 'Raw', 'Copy', 'Download', 'Edit' (highlighted with a red box), and 'Compare'. The main area displays the HTML code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>git</title>
8 </head>
9 <body>
10   깃을 사용해봅시다
11 </body>
12 </html>
```

원격 저장소 활용하기 - 3

- pull을 해야할 상황

동료 개발자가 코드를 수정하거나 새로운 소스 파일을 깃허브에 올렸을 때 해당 코드 또는 파일을 내 컴퓨터에 가져와서 사용해야 할 경우

원격 저장소 활용하기 - 4

- 상황 만들어보기 (동료 개발자가 코드 수정하여 올림)

파일 내용 수정 → Commit changes... → Commit changes

First-GitHub / index.html in master

Cancel changes Commit changes...

Edit Preview Spaces 4 No wrap

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>git</title>
8 </head>
9 <body>
10   깃을 사용해봅시다
11   동료 개발자1이 내용을 추가 했습니다.
12 </body>
13 </html>
--
```

원격 저장소 활용하기 - 5

- pull 사용해보기

VSCode로 돌아가서 git pull origin master를 사용할 경우 동료 개발자가 깃허브에 새로 올린 코드가 나의 작업 공간에 생김

```
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>git</title>
</head>
<body>
  깃을 사용해봅시다
  동료 개발자1이 내용을 추가 했습니다.
</body>
</html>
```

원격 저장소 활용하기 (충돌 상황)

원격 저장소 활용하기 - 1

• 상황

1. 동료 개발자가 다른 소스 파일을 수정하여 업로드함
→ 나는 이 소스 파일을 pull 받지 않음
2. 내가 새로운 소스 파일을 작업 후 push 하려함
3. 나와 동료 개발자가 올린 파일과 내용 차이가 생김

원격 저장소 활용하기 - 2

• 충돌 상황 만들어보기

슬라이드 5p를 참고하여 index.html에 내용 추가

Code Blame 17 lines (14 loc) · 407 Bytes Raw Copy Download Edit Dropdown Diff

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>git</title>
8  </head>
9  <body>
10     깃을 사용해봅시다
11     동료 개발자1이 내용을 추가 했습니다.
12
13     동료 개발자1이 내용을 또 추가 했습니다.
14 </body>
15 </html>
16
17
```

원격 저장소 활용하기 - 3

• 충돌 상황 만들어보기

push하면 깃허브 저장소에 있는 코드와 내 컴퓨터(로컬)에 있는 코드와의 차이가 있어서 에러 발생

* 강제 push 옵션(`git push -f`) 절대 사용 X

```
user@DESKTOP-I9RHACC MINGW64 /d/gitWorkspace (master)
$ git push origin master
To https://github.com/DevNirsa/First-GitHub.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/DevNirsa/First-GitHub.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

user@DESKTOP-I9RHACC MINGW64 /d/gitWorkspace (master)
```

원격 저장소 활용하기 - 4

• 충돌 상황 만들어보기

VSCode에서 pull 할 경우 충돌(conflict) 발생
빨간 박스 안을 하나씩 눌러서 어떻게 다른지 확인

```
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>git</title>
</head>
<body>
  깃을 사용해봅시다
  동료 개발자1이 내용을 추가 했습니다.
  Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<<< HEAD (Current Change)

  내가 내용을 추가 했습니다.
=====
  동료 개발자1이 내용을 또 추가 했습니다.
>>>>>>> 49a30f6fa238859c7c38f569b11075bd20f666ff (Incoming Change)
</body>
</html>
```

원격 저장소 활용하기 - 5

• 충돌 상황 해결하기

위 슬라이드에서 원하는 걸 선택 했다면 add → commit → push
이후 깃허브 저장소에 정상적으로 올라갔는지 확인

```
user@DESKTOP-I9RHACC MINGW64 /d/gitWorkspace (master|MERGING)
$ git add .
```

```
user@DESKTOP-I9RHACC MINGW64 /d/gitWorkspace (master|MERGING)
$ git commit -m "충돌 에러 해결"
[master bc2cd35] 충돌 에러 해결
```

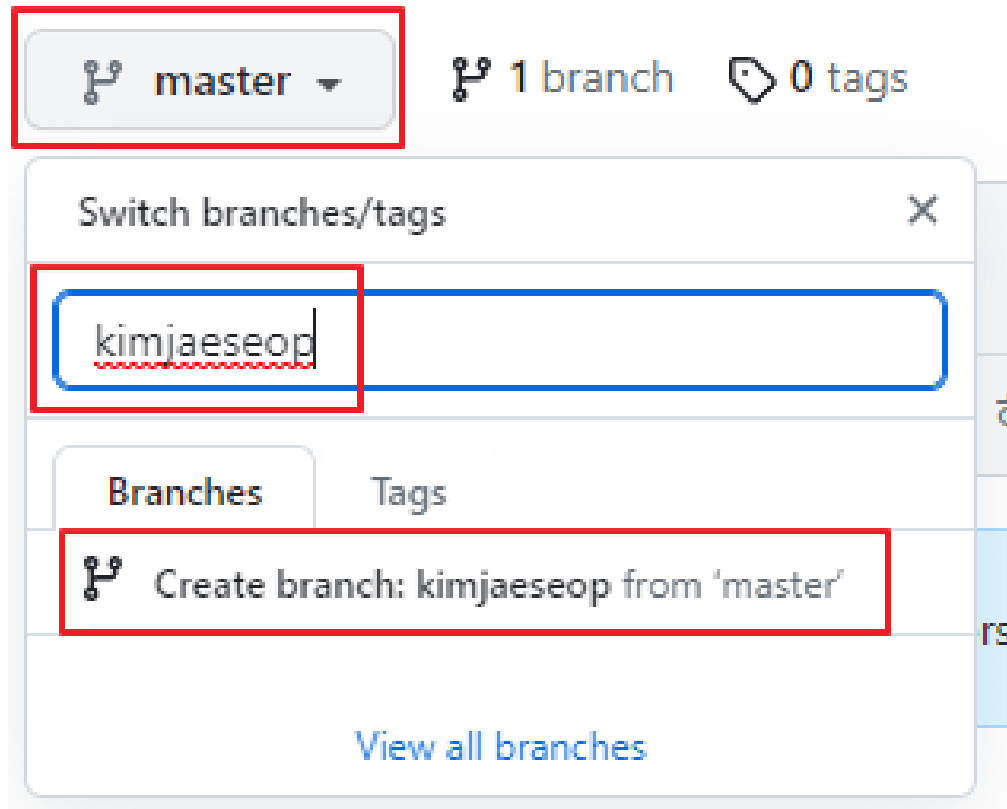
```
user@DESKTOP-I9RHACC MINGW64 /d/gitWorkspace (master)
$ git push origin master
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 723 bytes | 723.00 KiB/s, done.
Total 6 (delta 4), reused 0 (delta 0), pack-reused 0
```

원격 저장소 활용하기 (원격 저장소 브랜치)

원격 저장소 활용하기 - 1

- 설명

깃허브 홈페이지에서 새로운 브랜치 생성



원격 저장소 활용하기 - 2

• 설명

git branch -a 옵션을 사용하여 확인하기

1. firstBranch, master, secondBranch : 내 로컬 브랜치
2. remotes 아래 목록 : 깃허브 저장소에 있는 브랜치

```
user@DESKTOP-I9RHACC MINGW64 /d/gitWorkspace (master)
```

```
$ git branch -a
```

```
firstBranch
```

```
* master
```

```
secondBranch
```

```
remotes/origin/kimjaeseop
```

```
remotes/origin/master
```

```
remotes/test/kimjaeseop
```

```
remotes/test/master
```

원격 저장소 활용하기 - 3

• 설명

1. kimjaeseop 브랜치 생성
2. 로컬 브랜치를 kimjaeseop 으로 이동
3. 파일 내용 변경
4. 파일 추가
5. 커밋
6. 원격 저장소에 올리기

```
user@DESKTOP-I9RHACC MINGW64 /d/gitWorkspace (master)
$ git branch kimjaeseop
```

```
user@DESKTOP-I9RHACC MINGW64 /d/gitWorkspace (master)
$ git switch kimjaeseop
Switched to branch 'kimjaeseop'
```

```
user@DESKTOP-I9RHACC MINGW64 /d/gitWorkspace (kimjaeseop)
$ git add .
```

```
user@DESKTOP-I9RHACC MINGW64 /d/gitWorkspace (kimjaeseop)
$ git commit -m "김재섭 브랜치"
On branch kimjaeseop
nothing to commit, working tree clean
```

```
user@DESKTOP-I9RHACC MINGW64 /d/gitWorkspace (kimjaeseop)
$ git push origin kimjaeseop
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 390 bytes | 390.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/DevNirsa/First-GitHub.git
    bc2cd35..dce009b  kimjaeseop -> kimjaeseop
```


명령어 정리

기본 명령어

• 기본 명령어

| 명령어 | 설명 | 일반적인 사용 예시 |
|------------|--------------------------|------------------------------------|
| git init | 첫 프로젝트 시 Git 저장소 초기화 | git init |
| git clone | 원격 저장소를 로컬로 가져오기 | git clone [REPO_URL] [DIR] |
| git add | 변경된 파일을 staging area에 추가 | git add [FILE_NAME] |
| git commit | 변경 내용 저장 | git commit -m [MESSAGE] |
| git push | 커밋을 원격 저장소로 올리기 | git push [REPO_NAME] [BRANCH_NAME] |
| git pull | 원격 저장소의 최신 변경 내용 가져오기 | git pull [REPO_NAME] [BRANCH_NAME] |
| git status | 변경된 파일 상태 확인 | git status |
| git log | commit 기록 확인 (나가기 : q) | git log |

기본 명령어

• ORIGIN 의미

원격 저장소의 이름을 뜻하며 프로젝트 연결 시 `git remote add origin [URL]`을 입력할 때의 origin이 있는 필드가 저장소의 이름이 됨

`git remote -v` 명령어를 사용하여 확인해볼 수 있음

```
PS D:\gitWorkspace>
PS D:\gitWorkspace> git remote add test https://github.com/DevNirsa/First-GitHub.git
PS D:\gitWorkspace>
PS D:\gitWorkspace>
PS D:\gitWorkspace> git remote -v
origin https://github.com/DevNirsa/First-GitHub.git (fetch)
origin https://github.com/DevNirsa/First-GitHub.git (push)
test https://github.com/DevNirsa/First-GitHub.git (fetch)
test https://github.com/DevNirsa/First-GitHub.git (push)
PS D:\gitWorkspace> _
```

브랜치 관리 명령어

- git branch

| 명령어(옵션) | 설명 |
|--------------------------------|--------------------|
| git branch | 현재 작업중인 브랜치와 목록 출력 |
| git branch -v | 커밋 내용과 함께 정보 출력 |
| git branch [이름] | 로컬에 새로운 브랜치 생성 |
| git branch -d [이름] | 로컬 브랜치 삭제 |
| git branch -m [변경할 이름] [변경될이름] | 브랜치 이름 변경 |
| git branch -a | 로컬과 원격 브랜치 목록 출력 |

- git switch

| 명령어(옵션) | 설명 |
|------------|-----------|
| git switch | 로컬 브랜치 이동 |