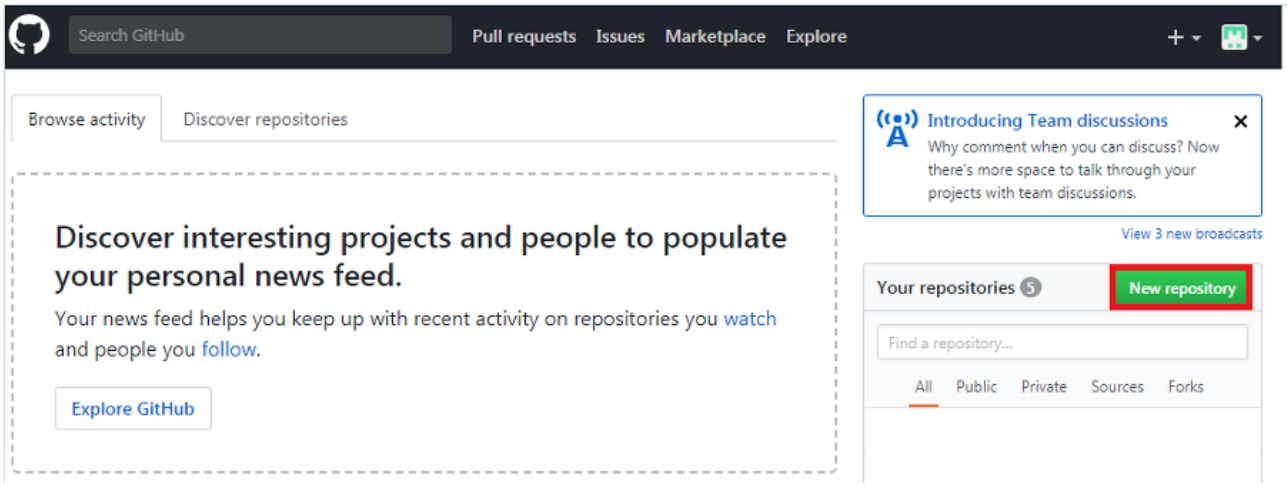
Git 명령어(2) – remote, push, clone, pull

1. 준비 작업

Remote Repository는 Github(또는 Bitbucket, Gitlab)에 존재하는 repository를 의미합니다.

협업을 하기 위해서는 local repository가 아닌 웹 상에 존재하는 repository가 필요합니다.
웹에 접근해서 자신의 작업 내용을 올리거나(push) 동료의 최신 코드를 받아와서(pull) 작업한 내용을 합칠 수(merge)가 있으니까요.

만약 Github 아이디가 존재하지 않으시면 회원가입을 먼저 해야 합니다.
push 명령어를 실행하기 위해서는 Github 계정이 필요하기 때문이죠.
회원 가입이 되셨다면 실습을 진행하기 위해 새로운 repository를 생성해주세요.



2. 명령어

- **git remote**
  - add : 원격 저장소 등록
  - show : 원격 저장소 정보
- **git push**
  - --delete : 원격 저장소 브랜치 삭제
- **git clone**
  - -b : 특정 브랜치를 clone
- **git pull**
  - --rebase : 기본 방식인 merge가 아니라 rebase 방식으로 pull 수행
- **git fetch** : local repository에서 remote repository의 내용들을 업데이트

3. 실습

remote repository에 파일을 업로드 하기 위해서는 local repository에 commit까지 한 상태여야 합니다.
따라서 commit을 한 상태라 가정을 하고 진행하도록 하겠습니다.

1) git remote

먼저 local repository에 연결된 remote repository를 확인하는 명령어를 실행해보도록 하겠습니다.

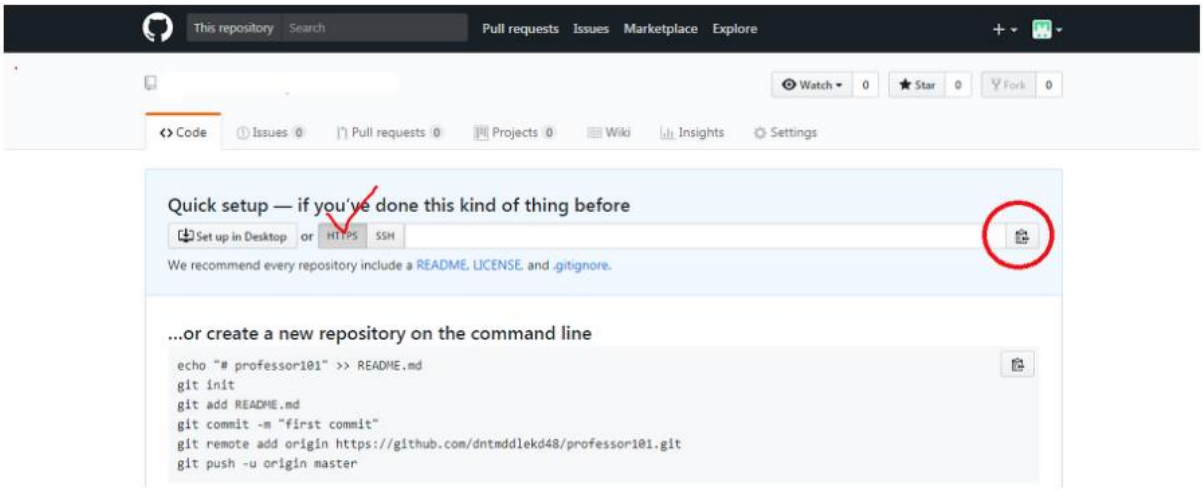
```
# git remote
```

연결된 remote repository가 없다면, 아무것도 출력이 되지 않을 것입니다.

local repository에 remote repository를 등록하기 위해서는 아래의 명령어를 실행합니다.

```
# git remote add {별칭} {remote repository URL 주소}
```

{별칭}은 일반적으로 origin을 사용하며, {remote repository URL 주소}는 아래 사진을 참고해서 복사하시면 됩니다.



remote repository가 등록 됐으면, git remote 명령어를 실행해서 연결된 remote repository를 확인해보겠습니다.

```
# git remote
```

참고로 원격 저장소의 세부 정보를 보고 싶다면, 아래의 명령어를 실행하면 됩니다.

```
# git remote show <원격 저장소명>
```

2) git push

이제 local repository의 파일들을 remote repository에 업로드 해보도록 하겠습니다.

업로드하는 명령어 git push의 구조는 다음과 같습니다.

```
# git push { 원격 저장소명 } { 원격 브랜치명 }
```

예제에서 remote repository의 별칭으로 **origin**을 사용했고, 브랜치(branch)는 생성한 적이 없습니다.

git init 명령어를 실행하면 default 브랜치로 **master** 브랜치를 생성해줍니다.

(브랜치는 **다음 글**에서 자세히 알아보겠습니다.)

따라서 아래와 같이 push 명령을 실행할 수 있습니다.

```
# git push origin master
```

```
PS C:₩Users₩samsung₩Desktop₩gitTest> git push origin master
Username for 'https://github.com':
Password for 'https://github.com':
Counting objects: 3, done.
Writing objects: 100% (3/3), 249 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/dntmddlekd48/test.git
 * [new branch]      master -> master
```

- 처음 push를 했다면, 계정 정보 확인을 위해 Github 이메일과 비밀번호를 입력하라고 합니다.
  - 비밀번호를 타이핑할 때는 출력이 되지 않으므로 그냥 입력하시면 됩니다.

위와 같이 100%가 채워지면, 성공적으로 push가 된 것입니다.

실제로 Github에 올라갔는지 새로고침을 눌러서 확인해보세요.

다음으로 push 명령어의 몇 가지 옵션들을 알아보겠습니다.

```
# git push -u { 원격 저장소 별명 } { 로컬 브랜치명 }
# git push --set-upstream { 원격 저장소 별명 } { 로컬 브랜치명 }
```

위 두 명령어는 같은 일을 합니다.

브랜치를 추적하도록 해서, 이후에 git push 명령어만 입력해도 원격 저장소로 push 할 수 있게 됩니다.

즉, git push { 원격 저장소명 } { 브랜치명 } 명령어가 git push으로 줄어들게 됩니다.

```
# git push --delete { 원격 브랜치명 }
```

remote repository에 있는 브랜치를 삭제합니다.

3) 원격 저장소에서 파일 가져오기 - 세 가지 방식

지금까지 push 명령어로 remote repository에 파일을 올리는 작업을 해봤습니다.

그런데 반대로 동료가 Github에 올려 놓은 소스를 local로 가져오려면 어떻게 해야할까요?
이에 대한 명령어로 clone과 pull 두 가지가 있고, 그냥 zip 파일로 받는 방법이 있습니다.

- **zip 파일**
  - .git 폴더가 없는 채로 소스만 받을 수 있으므로, local에서 Git을 새롭게 꾸려나가야 합니다.
- **clone**
  - .git 폴더까지 포함해서 소스를 받을 수 있습니다.
- **pull**
  - remote repository에 저장된 파일을 가져와 local repository의 내용을 갱신합니다.
    - 병합(merge) 과정이 발생합니다.
  - 해당 remote repository에 권한이 있어야 pull 명령어를 실행할 수 있습니다.

4) git clone

먼저 clone 명령어에 대해 알아보도록 하겠습니다.

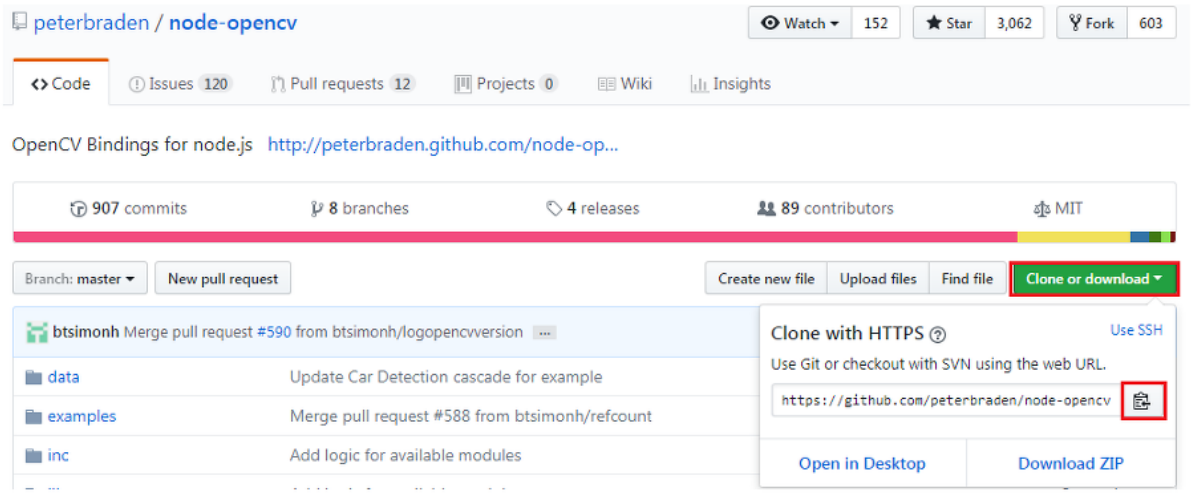
기본 구조는 다음과 같습니다.

```
# git clone { 원격 저장소 URL }
```

특정 브랜치를 clone 하고 싶다면,

```
# git clone -b { 브랜치명 } { 원격 저장소 URL }
```

현재 디렉토리에다가 오픈소스를 clone 해볼까요?
원하시는 오픈 소스의 github 페이지에서 URL을 복사해주세요.
저는 node-opencv 오픈소스를 clone해보도록 하겠습니다.



```
# git clone https://github.com/peterbraden/node-opencv.git
```

```
PS C:₩Users₩samsung₩Desktop₩gitTest> git clone https://github.com/peterbraden/node-opencv.git
Cloning into 'node-opencv'...
remote: Counting objects: 4059, done.
remote: Compressing objects: 100% (54/54), done.
rReceiving objects: 100% (4059/4059), 9.91 MiB | 1.51 MiB/s, done.
emote: Total 4059 (delta 70), reused 106 (delta 65), pack-reused 3940
Resolving deltas: 100% (2600/2600), done.
Checking connectivity... done.
PS C:₩Users₩samsung₩Desktop₩gitTest> ls


    디렉터리: C:₩Users₩samsung₩Desktop₩gitTest


Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
d----          2017-11-27  오후 12:36            node-opencv
-a---          2017-11-27  오전 11:35          8 victolee.txt
```

상태를 확인해보면, working directory에 node-opencv 오픈소스의 코드가 clone 된 것을 확인할 수 있습니다.

```
# git status
```

```
PS C:₩Users₩samsung₩Desktop₩gitTest> git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        node-opencv/

nothing added to commit but untracked files present (use "git add" to track)
```

이처럼 clone 명령어는 처음 프로젝트에 투입되거나, 책에서 제공하는 예제 소스를 가져올 때, 중간에 git이 꼬였을 때 사용할 수 있습니다.
저장소를 통째로 가져오는 것이죠.

5) git clone / git pull 비교

다음으로 pull 명령어 역시 remote repository에 있는 파일들을 local로 가져오는 명령어이며, 차이점은 다음과 같습니다.

- git clone은 Github의 모든 파일들을 가져오기만 함
- git pull은 local repository와 비교하여 병합을 하고, local repository에 저장(add)까지 수행
    - git pull = git fetch + git merge
        - git fetch는 local에 연결된 remote repository의 브랜치 목록과 그 파일 내용을 최신으로 업데이트 하는 명령어입니다.
            - local과 remote의 싱크를 맞추는 새로고침 역할입니다.
        - git merge는 두 개의 branch를 병합하는 명령어입니다.

git pull은 협업 과정에서 프로젝트의 최신 코드를 local로 가져오는 역할로 많이 사용합니다.

예를 들어, 팀 프로젝트를 진행하다가 동료가 기능 구현을 완료해서 push를 했다고 가정해보겠습니다.
저는 새로운 작업 내용을 반영하기 위해, Github에 있는 코드를 가져와야 하는데 어떤 방식으로 가져와야 할까요?

git clone을 하면 제가 그동안 작업했던 내용들과 동료가 작업한 최신 버전은 독립된 존재가 되어버립니다.
즉, clone을 해서 받은 폴더에 제가 작업했던 내용들을 일일이 수작업으로 적용시켜야 하죠.
이는 좋은 방법이 아닙니다.

git pull을 하면 어떨까요?
현재 제가 작업 중인 local repository와 최신 코드가 비교및 병합되어, 최신 버전 파일들이 저의 local repository에 적용됩니다.
따라서 제가 작업하고 있던 코드들과 최신 버전 파일의 코드는 함께 존재하게 되죠.
즉, 제가 작업한 코드와 동료의 작업 코드가 자동으로 합쳐지게 되니까 매우 바람직합니다.

그런데 만약 저와 동료가 동일한 파일 내역을 수정했다면 어떻게 될까요?
이 경우 **충돌**(conflicts)이 발생하며 일일이 수작업으로 충돌된 부분을 수정해야 합니다.
(이와 관련된 내용은 **여기**를 참고해주세요.)

6) git pull

git pull 명령어의 기본구조는 다음과 같습니다.

```
# git pull { 원격 저장소 별명 } { 브랜치명 }
```

git pull = git fetch + git merge와 같다고 했습니다.

병합하는 명령어 중에 rebase라는 것이 있는데, Git에서는 최신 코드로 업데이트 할 때 rebase를 사용할 것을 권장하고 있습니다.
(rebase에 대해서는 **여기**를 참고해주세요.)

rebase 명령어는 merge 명령어보다 히스토리(log)가 깔끔해진다는 장점이 있는데요,
어쨋든 pull 명령어를 실행할 때 merge가 아닌 rebase 방식으로 병합을 하려면 다음과 같이 --rebase 옵션을 주면 됩니다.

```
# git pull --rebase { 원격 저장소 별명 } { 브랜치명 }
```

git pull --rebase = git fetch + git rebase와 같습니다.

이상으로 remote repository에 연결하는 git remote 명령어와 git push, git clone, git pull 명령어에 대해 알아보았습니다.
협업 시 push를 할 때 주의할 점은 push 하기 전에 local repository는 항상 remote repository의 최신 버전을 유지해야 한다는 것입니다.
remote repository에 존재하는 커밋이 local 저장소에 없으면 에러가 발생하기 때문입니다.