[Git] 명령어(3) – branch

1. branch

영어 단어 branch는 나뭇가지입니다.
이를 Git에 적용해보면 큰 줄기(master)에서 뻗어나간 가지(branch)라고 생각할 수 있습니다.

master 브랜치

- git init 할 때 자동으로 생성해주는 기본 브랜치
- 동료들의 작업 내용을 하나로 합칠 때 사용하는 뼈대 브랜치로 사용할 수 있습니다. (브랜치 전략은 회사마다 팀마다 다르므로..)
  - 배포용 브랜치로 사용할 수 있습니다.

브랜치는 **독립적인 작업 공간을 생성**하는 중요한 개념입니다.
작업을 진행하면서 테스트가 필요하다거나, 여러 가지 버전으로 구현해보고 싶은 경우 어떻게 해야 할까요?
프로젝트 폴더를 복사해서 테스트를 해야 할까요?
branch를 사용하면 그럴 필요 없이 현재 버전을 그대로 가져와 새로운 작업공간으로 활용할 수 있습니다.

또한 원격 저장소(Github)에 여러 브랜치를 만들면 팀원들이 각각 독립적인 원격 저장소를 가질 수 있게 됩니다.
통합이 필요하면 특정 브랜치에 merge를 수행하면 되구요.

이처럼 branch는 독립적인 공간을 제공하므로 테스트, 통합 등으로 활용할 수 있습니다.

2. 명령어

- **git branch** : local repository 브랜치 목록 확인
  - 브랜치명 : 브랜치 새성
    - {원격저장소}/{브랜치} : 원격 저장소의 특정 브랜치로부터 새로운 브랜치 생성
  - -r : remote repository 브랜치 목록 확인
  - -a : local / remote repository 모두의 브랜치 목록 확인
  - -D : local repository 브랜치 삭제
- **git checkout** : 브랜치 변경
  - -b : 브랜치 생성 후, 그 브랜치로 이동

3. 실습

1) git branch

브랜치를 생성하는 명령어는 다음과 같습니다.

```
# git branch { 브랜치명 }
```

새로운 브랜치를 생성하면, 기반이 되는 브랜치(부모 브랜치)의 버전을 그대로 복사합니다.
커밋 내역(log)도 부모 브랜치와 같게 됩니다.

브랜치를 이동하는 명령어는 다음과 같습니다.

```
# git checkout { 브랜치명 }
```

두 명령어를 합친, 즉 브랜치를 생성함과 동시에 이동하는 명령어는 다음과 같습니다.

```
# git checkout -b { 브랜치명 }
```

이제 새로운 브랜치 foo를 생성해서 foo 브랜치로 이동해보겠습니다.
그리고 git branch 명령어로 local repository에 있는 브랜치의 목록과 현재 사용하고 있는 브랜치를 확인할 수 있습니다.

```
# git branch foo
# git checkout foo
# git branch
```

```
PS C:₩Users₩samsung₩Desktop₩gitTest> git branch foo
PS C:₩Users₩samsung₩Desktop₩gitTest> git branch
  foo
* master
PS C:₩Users₩samsung₩Desktop₩gitTest> git checkout foo
Switched to branch 'foo'
PS C:₩Users₩samsung₩Desktop₩gitTest> git branch
* foo
  master
```

* 표시가 된 branch가 현재 사용하고 있는 branch입니다.

원격 브랜치 목록만 보고 싶으면 -r 옵션을 추가하면 되고, 로컬/원격 모든 브랜치를 보고 싶으면 -a 옵션을 추가하면 됩니다.

```
# git branch -r
# git branch -a
```

2) branch push해서 원격 저장소에서 독립된 공간임을 확인

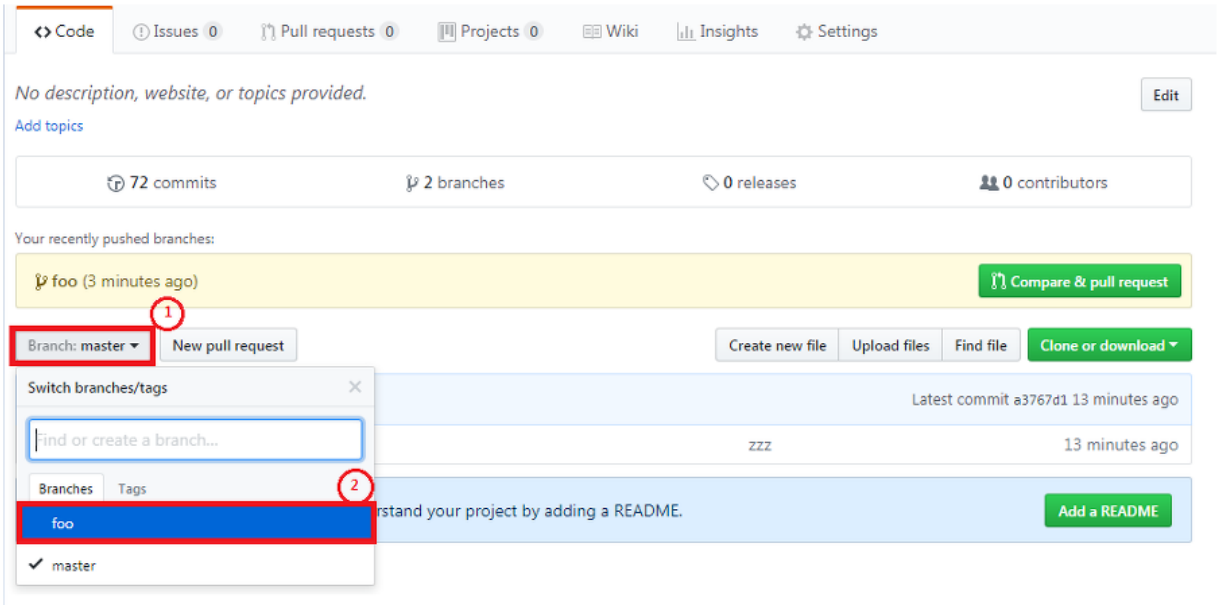
이제 브랜치가 독립된 공간인지 확인하기 위해, 새로운 텍스트 파일을 생성하여 아무 글이나 작성한 후 remote repository에 push 해보도록 하겠습니다.

```
# git add .
# git commimt -m " 새로운 브랜치에서 업로드 "
# git push origin foo
```

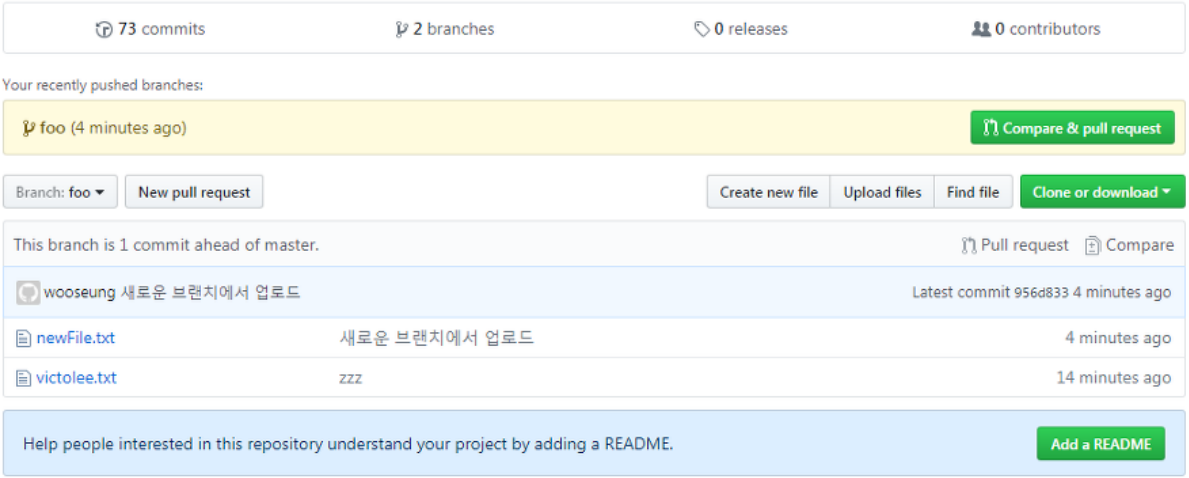
master가 아닌 foo 브랜치에 push 할 것이므로 git push origin foo 명령어임에 주의해주세요.

정상적으로 push가 됐으면, Github에서 master 브랜치와 새로운 브랜치의 commit 내용을 보도록 하겠습니다.

master 브랜치



foo 브랜치



master 브랜치에서는 commit 수가 72이었는데, foo 브랜치에서는 commit 수가 73으로 되었네요.
이 말은 branch가 서로 독립적인 공간이라는 것을 의미합니다.
즉, **새로운 foo 브랜치에서 작업하는 내용은 master에 반영되지 않습니다.**

3) master 브랜치에서 foo 브랜치의 내용을 가져온 후, 원격 저장소에 push

foo 브랜치는 새로운 기능을 구현하기 위한 브랜치이며, master 브랜치는 여러 기능들을 통합해서 관리하는 브랜치라 가정하겠습니다.

foo 브랜치에서 새로 생성한 파일을 아무렇게나 수정을 하는 행위로 기능을 구현했다고 가정하겠습니다.

```
# git add .
# git commit -m "기능구현 완료"
# git push origin foo
```

그러면 원격 저장소에서 foo branch에는 커밋이 됐지만, master branch에는 아직 적용되지 않은 상태입니다.
따라서 local repository의 master 브랜치에서는 foo 브랜치를 pull 명령어로 땡겨와서 foo 브랜치의 작업 내용을 반영하도록 하겠습니다.

```
# git checkout master
# git pull origin foo
```

pull 명령어는 merge까지 이뤄지므로, foo 브랜치의 커밋 내용이 mater 브랜치에도 포함되어 있습니다.
따라서 바로 master branch에 push 할 수 있습니다.

```
# git push origin master
```

4) 브랜치 삭제

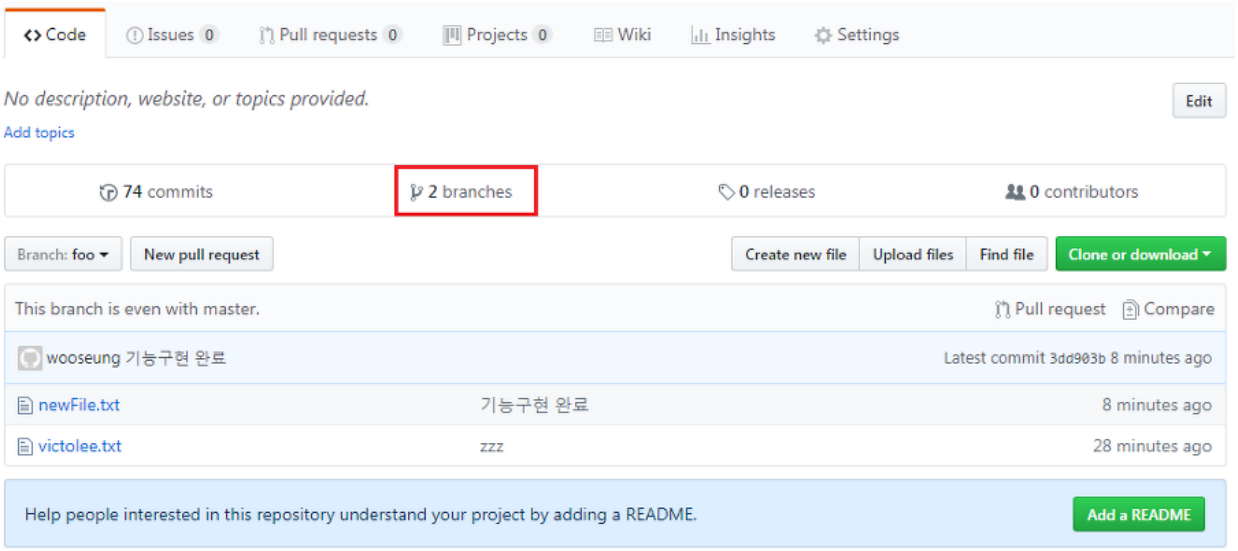
foo 브랜치는 새로운 기능을 구현함으로써 branch의 역할을 다했습니다.
기능 구현이 끝난 브랜치는 local repository에서 삭제하는 것이 브랜치 관리면에서 편하므로 브랜치를 삭제하도록 하겠습니다.
필요하면 다시 remote repository에서 가져오면 되니까요.

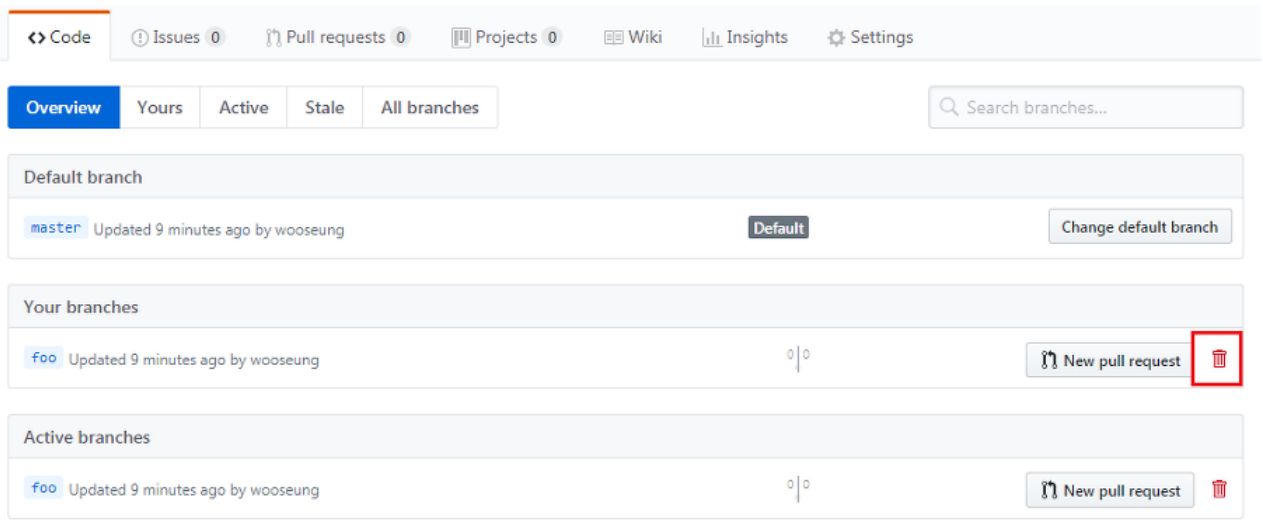
```
#  git branch -D foo
```

참고로 remote repository의 특정 브랜치를 가져와서 새로운 브랜치로 만들고 싶은 경우입니다. (**참고링크**)

```
# git branch {새브랜치명} {원격브랜치}/{원격브랜치명}
```

local repository에서는 foo branch가 삭제되지만, remote repository에서는 삭제되지 않습니다.
깃헙에서 foo branch를 삭제하고 싶으면 아래 사진을 참고해주세요.





이상으로 branch에 대한 개념을 알아보았습니다.
원격 저장소에서 이렇게 브랜치를 다루는 방식은 협업할 때 흔히 다루는 방식입니다.
local 환경에서 테스트 용으로 branch를 생성하는 방법도 똑같습니다만, 코드를 통합할 때 merge 명령어를 사용합니다.
merge 명령어는 **다음 글**에서 자세히 다루도록 하겠습니다.