Feature Branch Workflow

* Feature Branch Workflow의 핵심 컨셉은 기능별 브랜치를 만들어서 작업하는 것이다.
* 다수의 팀 구성원이 메인 코드 베이스(master)를 중심으로 해서 안전하게 새로운 기능을 개발할 수 있다.
* Feature Branch Workflow와 풀 리퀘스트를 결합하면 팀 구성원간에 변경 내용에 대한 소통을 촉진해서 코드 품질을 높일 수 있다.
* 유연성이 큰 협업 방법
* **소규모 인원**의 프로젝트에서 사용하는 협업 방법

**1. 중앙 원격 저장소, 자신의 원격 저장소, 로컬 저장소의 개념**

* 중앙 원격(remote) 저장소
* 여러 명이 같은 프로젝트를 관리하는 데 사용하는 그룹 계정의 중립된 원격 저장소
* Organization을 만드는 방법은 GitHub 페이지 오른쪽 위에 있는 “+” 아이콘을 클릭하고 메뉴에서 “New organization”을 선택하면 된다.
* Organizatoin의 사용자와 저장소는 팀으로 관리되고 저장소의 권한 설정도 팀으로 관리한다.
* 자신의 원격(remote) 저장소
* remote repository 라고 불린다.
* 파일이 GitHub 전용 서버에서 관리되는 원격 저장소
* 로컬(local) 저장소
* local repository 라고 불린다.
* 내 PC에 파일이 저장되는 개인 전용 저장소, 지역 저장소

**2. 프로젝트 참여자는 git clone 명령으로 로컬 저장소를 만든다.**

git clone 명령으로 중앙 원격 저장소(remote repository)를 복제하여 자신의 로컬 저장소(local repository)를 만들 수 있다. 프로젝트 참여자는 이 로컬 저장소에서 작업을 수행한다.

// 터미널에서 자신의 원하는 디렉터리 이동한 후 아래의 명령어를 입력

/\*\*

해당 디렉터리

--하위--> remote repository와 동일한 이름의 디렉터리 생성

\*\*/

$ git clone [중앙 remote repository URL]

* git clone 명령은 아래의 명령들을 포함한 작업이다.

//해당 디렉터리를 빈 Git 저장소로 만드는 작업

$ git init

// 현재 작업 중인 Git 저장소에 팀의 중앙 원격 저장소를 추가한다. 이름을 origin으로 짓고 긴 서버 주소(URL) 대신 사용한다.

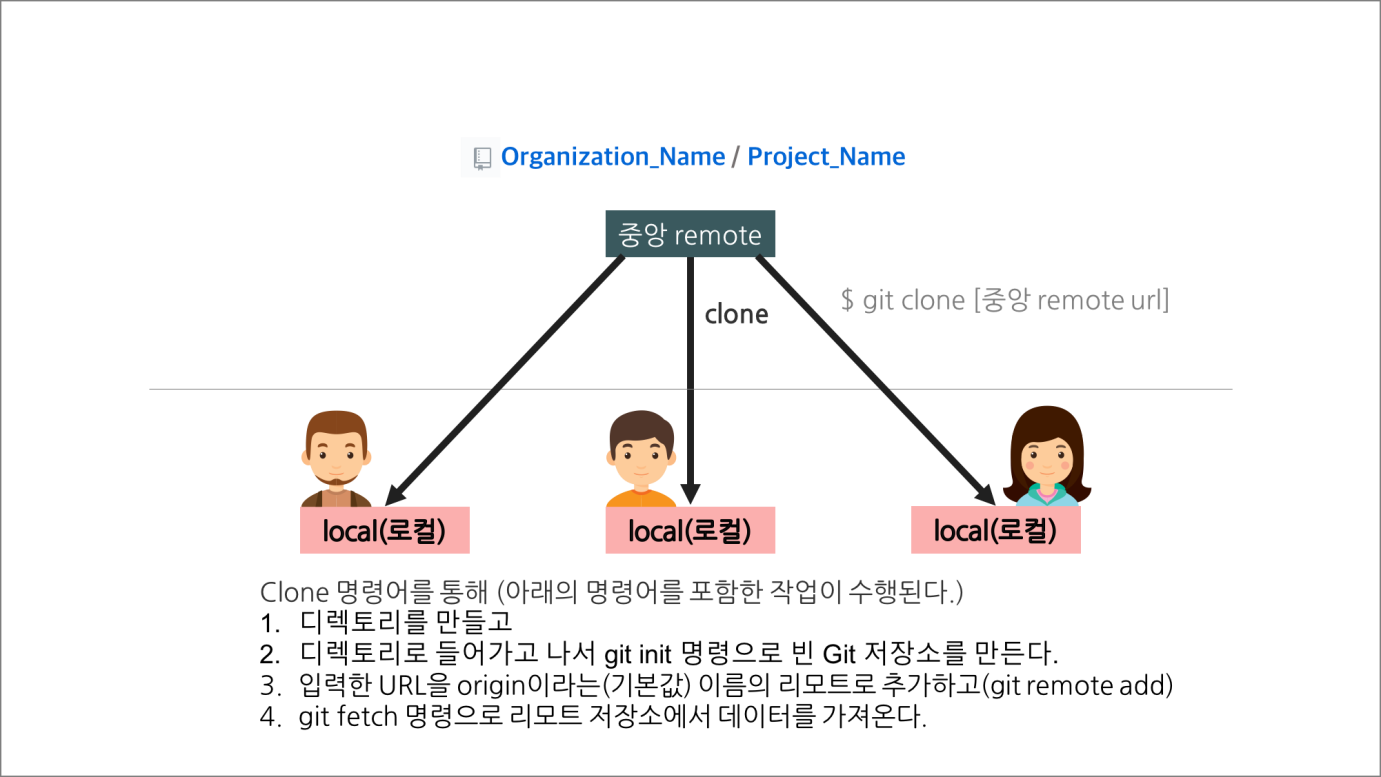
$ git remote add origin [중앙 remote repository URL]

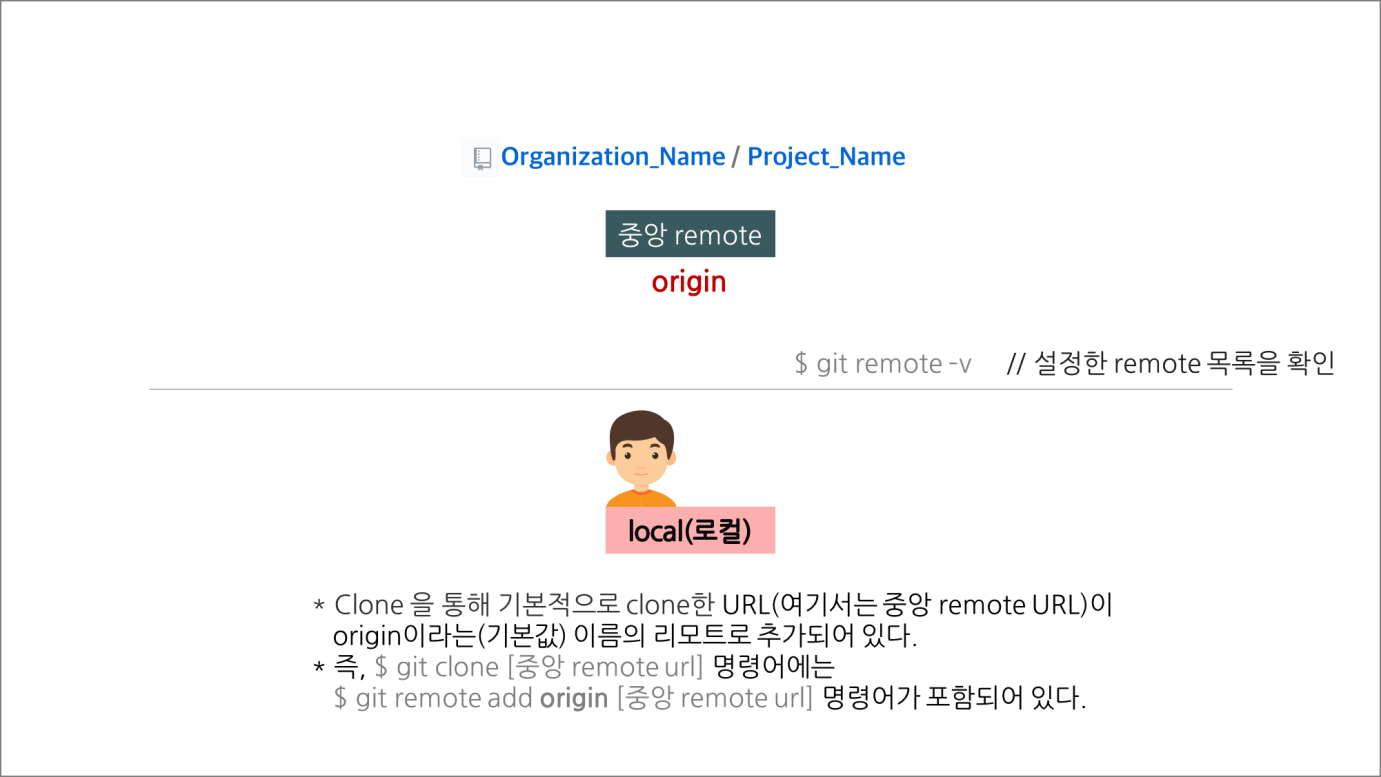
// 중앙 원격 저장소(origin)의 master 브랜치 데이터를 로컬에 가져오기만 하는 작업

$ git fetch origin master

fetch와 pull의 차이

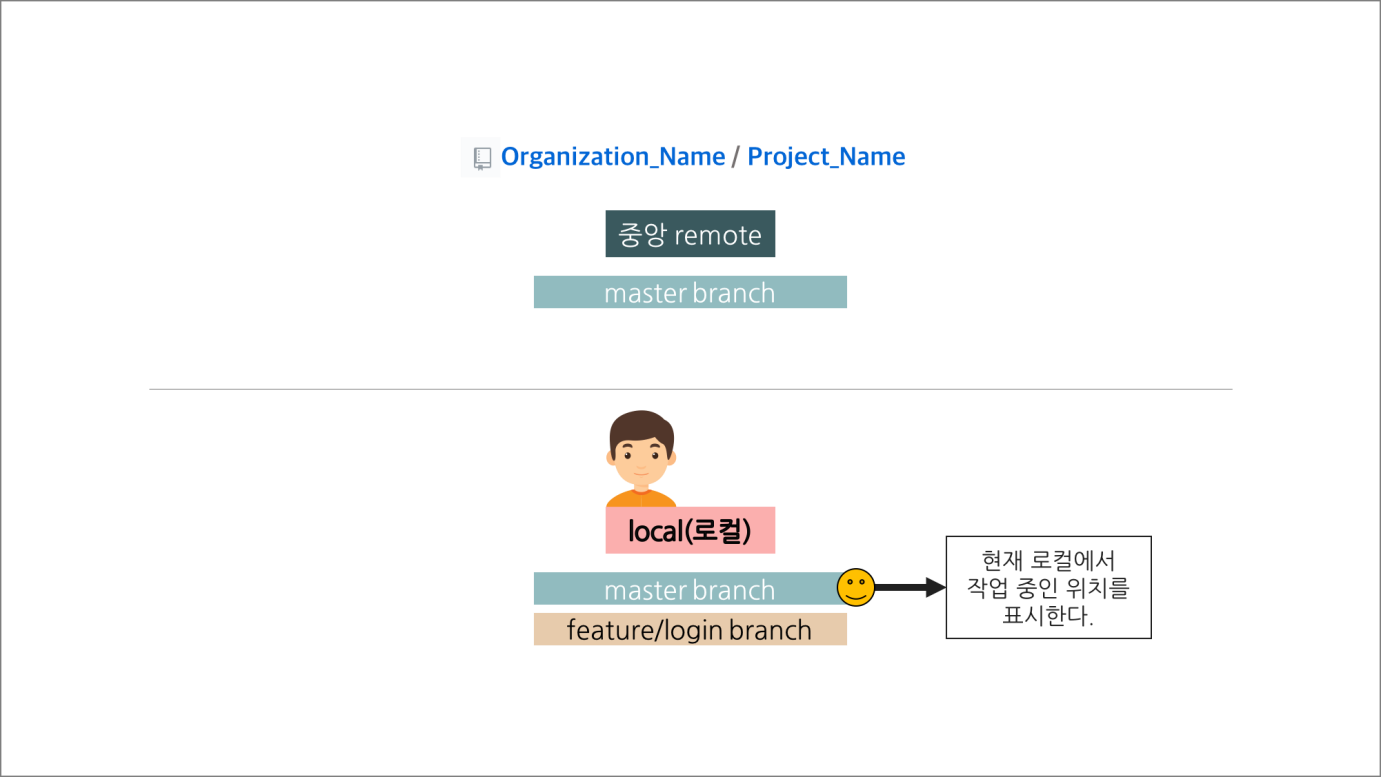
* fetch: 원격 저장소의 데이터를 로컬에 가져오기만 하기
* pull: 원격 저장소의 내용을 가져와 자동으로 병합 작업을 실행
* 즉, 단순히 원격 저장소의 내용을 확인만 하고 로컬 데이터와 병합은 하고 싶지 않은 경우에는 fetch 명령어를 사용한다.
* **pull = fetch + merge**





**3. 설명을 위해 현재 로컬에서 작업 중인 branch 위치를 표시한다.**

중앙 원격 저장소에는 master branch가 있고, 자신의 로컬 저장소에는 master branch와 로그인 기능을 구현할 feature/login branch(아래에서 설명)가 있다고 가정한다.

또한 현재 master branch에서 작업 중이라고 가정하고 아래와 같이 작업 중인 위치를 표시한다. 

**4. 새로운 기능 개발을 위해 격리된 branch를 만든다.**

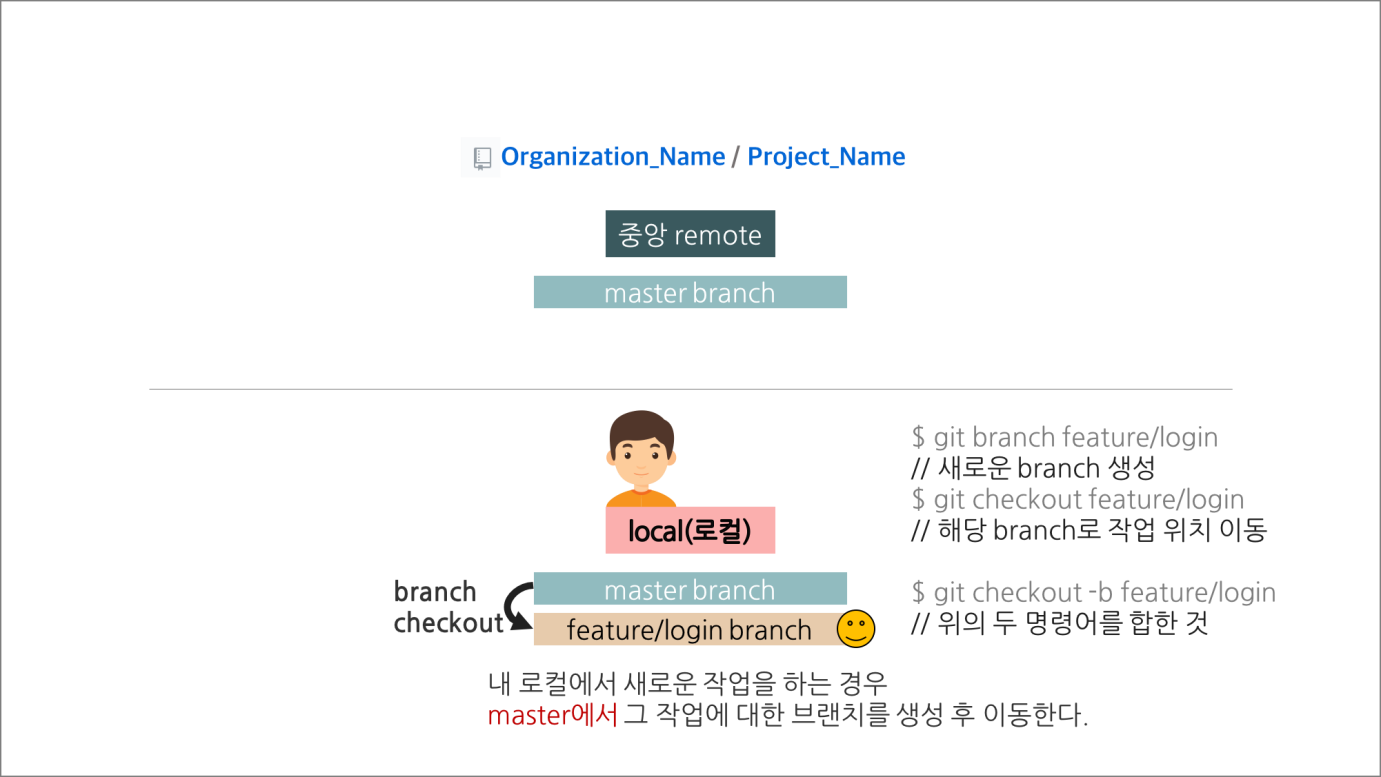
로컬 저장소에서 branch를 따고, 코드를 수정하고, 변경 내용을 커밋한다.

$ git checkout -b [branch name]

# 위의 명령어는 아래의 두 명령어를 합한 것

$ git branch [branch name]

$ git checkout [branch name]



**5. 로컬 저장소의 새로운 기능 브랜치를 중앙 원격 저장소(remote repository)에 푸시한다**.

* 새로 만든 브랜치(feature/login branch)에 새로운 기능에 대한 내용을 커밋한다.

$ git commit -a -m "Write commit message"

# 위의 명령어는 아래의 두 명령어를 합한 것

$ git add . # 변경된 모든 파일을 스테이징 영역에 추가

$ git add [some-file] # 스테이징 영역에 some-file 추가

$ git commit -m "Write commit message" # local 작업폴더에 history 하나를 쌓는 것

* 커밋을 완료했다면, 내가 작업한 내용을 포함한 브랜치(feature/login branch)를 중앙 원격 저장소에 올린다.
* 이는 로컬 저장소의 백업 역할을 할 뿐만 아니라, 다른 팀 구성원들이 나의 작업 내용과 진도를 확인할 수도 있어 좋은 습관이라 할 수 있다.

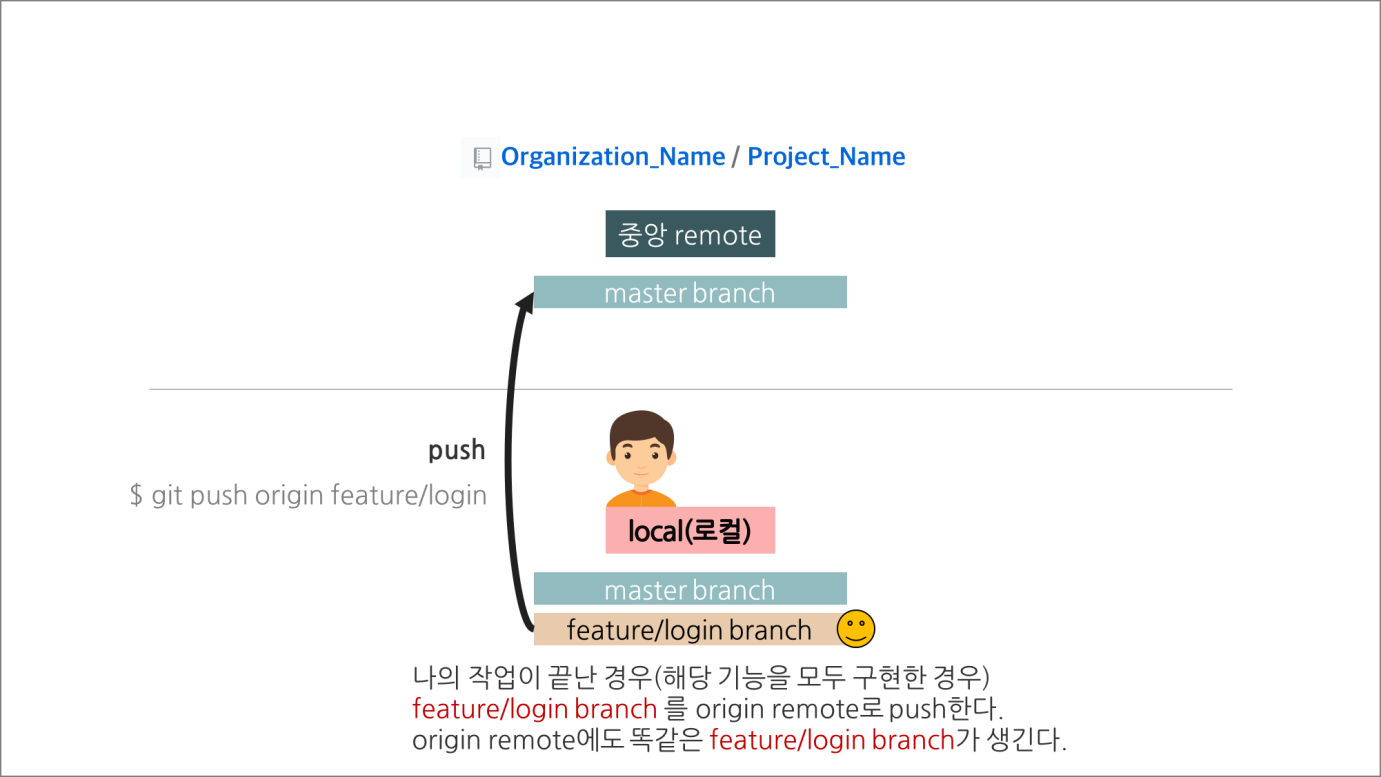
# -u 옵션: 새로운 기능 브랜치와 동일한 이름으로 중앙 원격 저장소의 브랜치로 추가한다.

// 로컬의 기능 브랜치를 중앙 원격 저장소 (origin)에 올린다.

$ git push -u origin feature/login branch

// -u 옵션으로 한 번 연결한 후에는 옵션 없이 아래의 명령만으로 기능 브랜치를 올릴 수 있다.

$ git push -origin feature/login branch

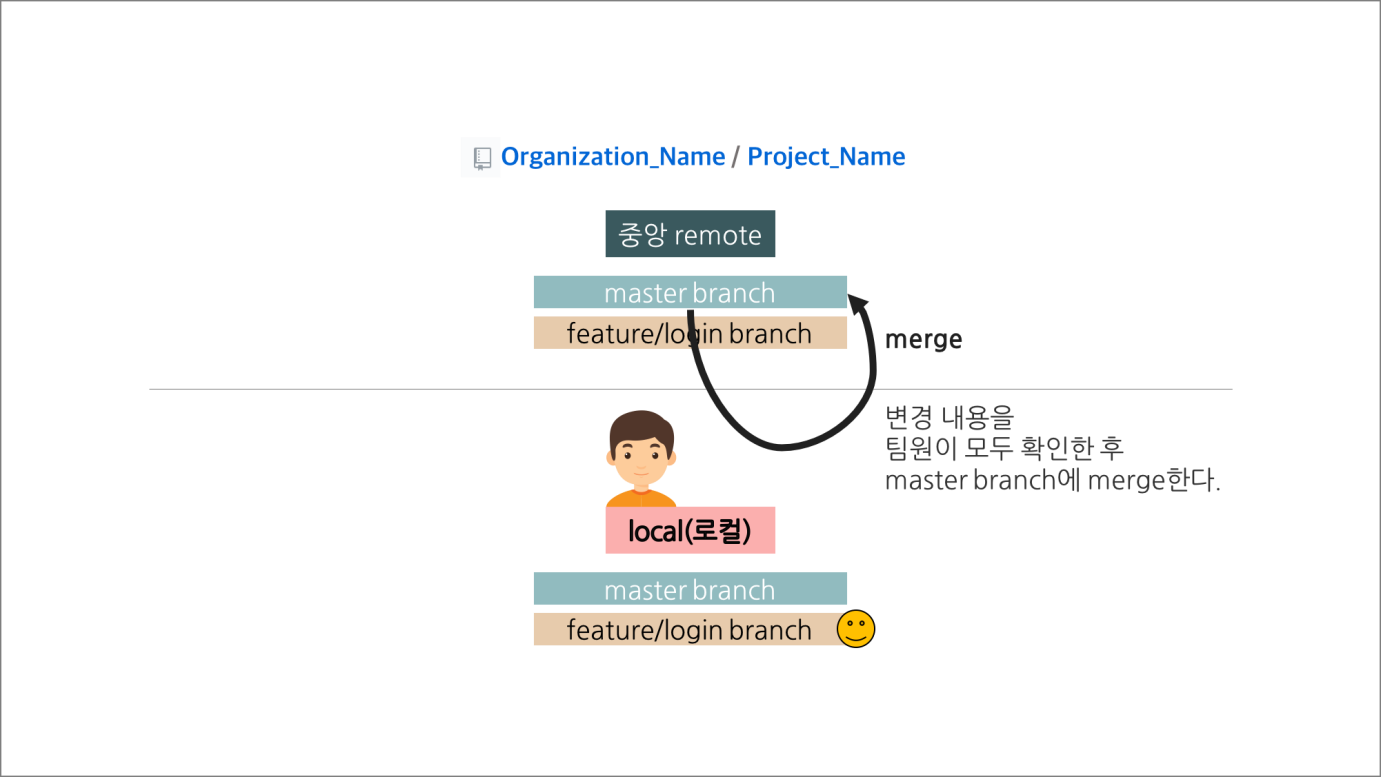


**6. 프로젝트 관리자(소규모 팀에서는 모두가 관리자가 될 수 있음)에게 자신의 기여분을 반영해 달라는 풀 리퀘스트를 던진다.**

이제, 프로젝트 관리자(소규모 팀에서는 모두가 관리자가 될 수 있음)에게 자신의 기여분을 중앙 원격 코드 베이스에 반영해 달라고 요청해야 한다. 새로 만든 기능 개발용 브랜치도 중앙 저장소에 올려서 팀 구성원들과 개발 내용에 대한 의견(코드 리뷰 등)을 나눌 수 있다.

* master 브랜치를 손대지 않기 때문에 다른 기능 개발 브랜치를 얼마든지 올려도 된다.
* 이는 일종의 로컬 저장소 백업 역할을 하기도 한다.
* 풀 리퀘스트(Pull requests)란?

기능 개발을 끝내고 master에 바로 병합(merge)하는 것이 아니라, 브랜치를 중앙 원격 저장소에 올리고 master에 병합(merge)해달라고 요청하는 것



* GUI 도구를 이용한 풀 리퀘스트

1. GitHub 페이지에서 “Pull requests” 버튼을 이용하면, 어떤 branch를 제출할지 정할 수 있다.
2. 기능을 구현한 branch(여기서는 feature/login branch)를 프로젝트 중앙 원격 저장소의 master branch에 병합해 달라고 요청한다.

**7. 프로젝트 관리자는 변경 내용을 확인한 후 중앙 원격 코드 베이스에 병합(merge)한다.**

이후에는 모든 팀원이 변경한 코드 내용을 확인하고 마지막으로 확인한 팀원이 변경 내용을 중앙 원격 코드 베이스에 병합(merge)하는 작업을 한다. 병합하는 과정은 아래와 같다.

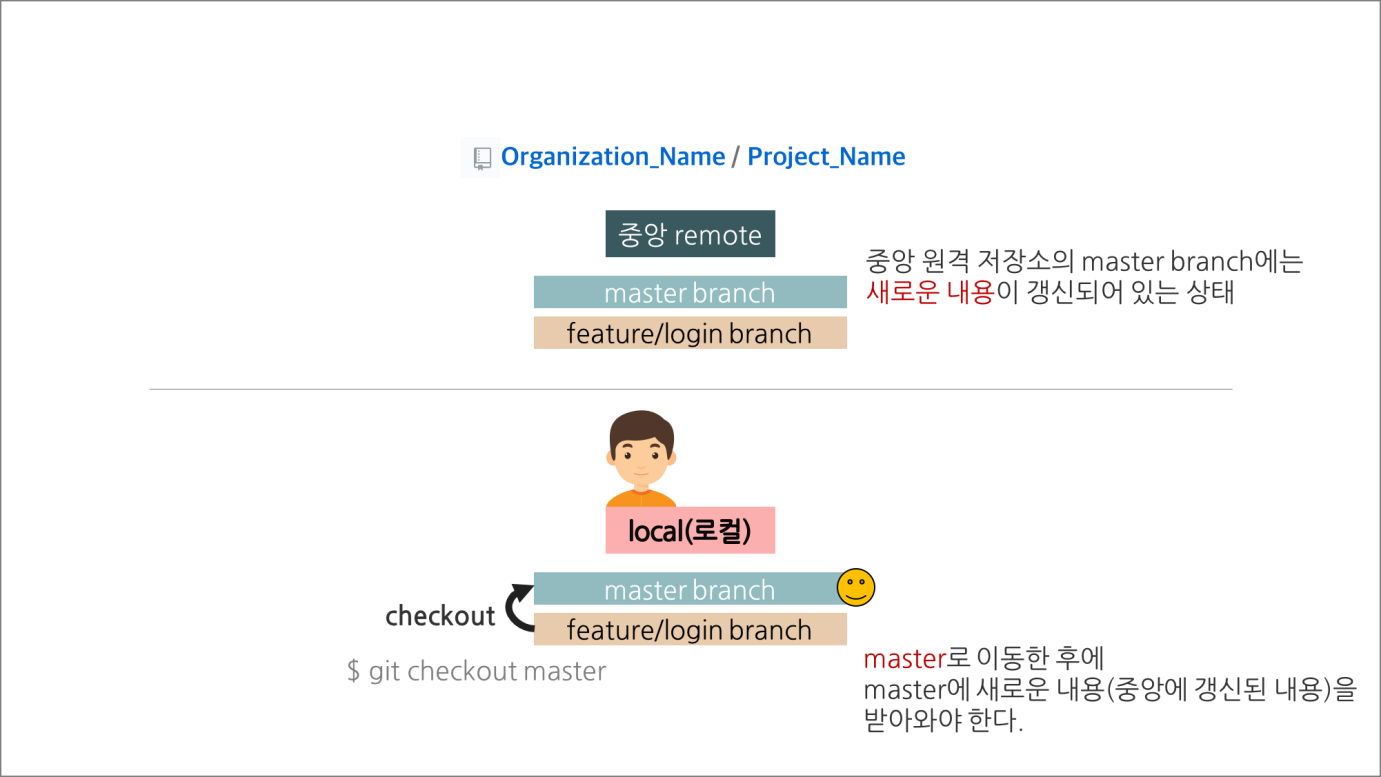
* GUI 도구를 이용한 병합

1. GitHub 페이지에서 “Pull requests” 버튼을 누른 후, File changed 탭에서 변경 내용을 확인한다.
2. Conversation 탭으로 이동하여 “Confirm merge”를 하면 중앙 원격 코드 베이스에 병합(merge)된다.
3. 충돌이 일어난 경우는 팀원들과 합의 하에 충돌 내용을 수정한 후 병합을 진행한다.

**8. 중앙 원격 저장소와 자신의 로컬 저장소를 동기화하기 위해 로컬 저장소의 branch를 master branch로 이동한다.**

// 로컬 저장소의 branch를 master branch로 이동

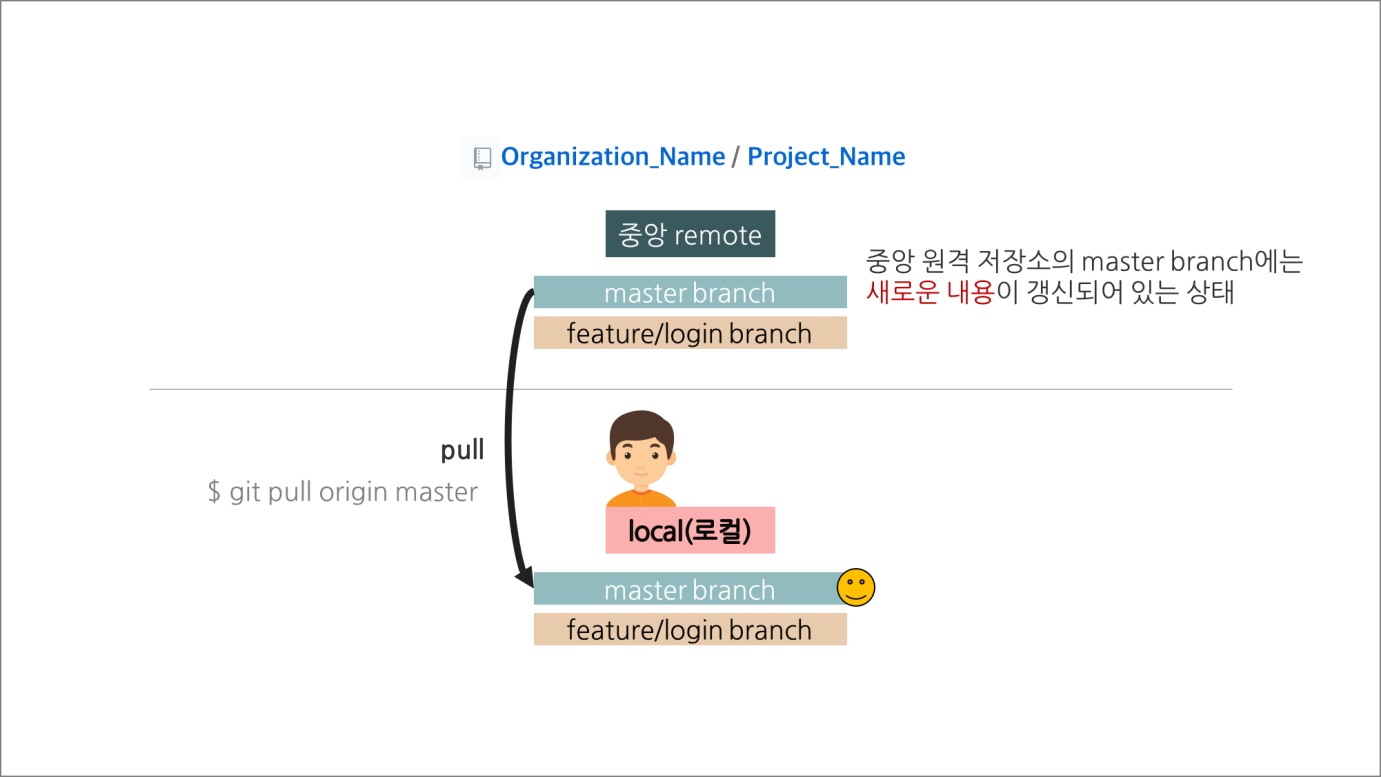
$ git checkout master



**9. 중앙 원격 저장소의 코드 베이스에 새로운 커밋이 있다면 다음과 같이 가져온다.**

중앙 원격 저장소(origin)의 메인 코드 베이스가 변경되었으므로, 프로젝트 참여하는 모든 개발자가 자신의 로컬 저장소를 동기화해서 최신 상태로 만들어야 한다.

$ git pull origin master



**10. 새로운 기능을 추가하기 위해서 그 작업에 대한 branch를 생성하여 작업한다.**

중앙 원격 저장소와 동기화된 로컬 저장소의 master branch에서 새로운 작업에 대한 branch를 생성하여 작업을 시작한다.

