2020년도 2학기 컴퓨터공학설계및실험Ⅰ

5주차 Github 사용 결과 보고서

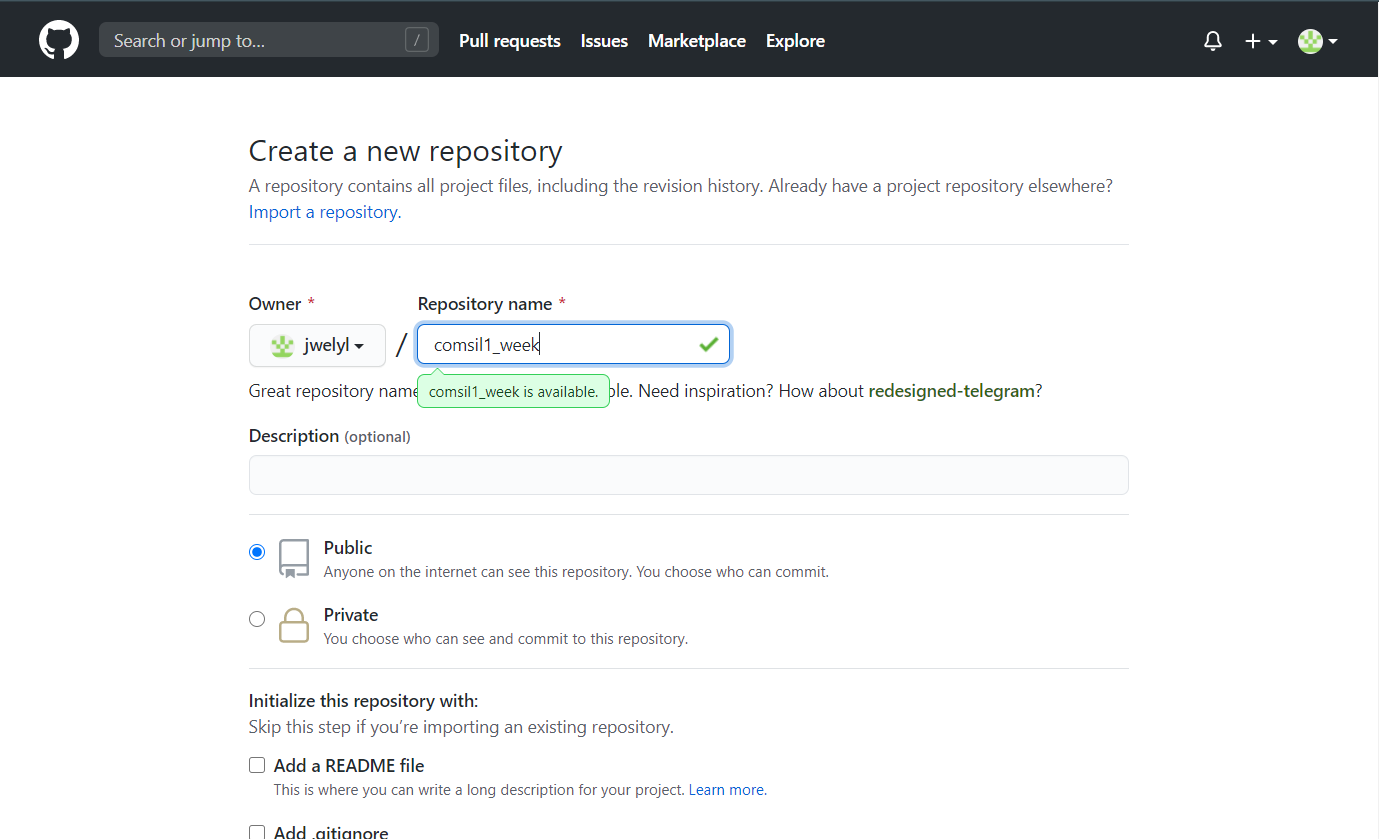
20161663 허재성

1. 실습 목적

버전 관리 시스템이 무엇인지 이해하고, Git과 Github의 사용법을 익힌다..

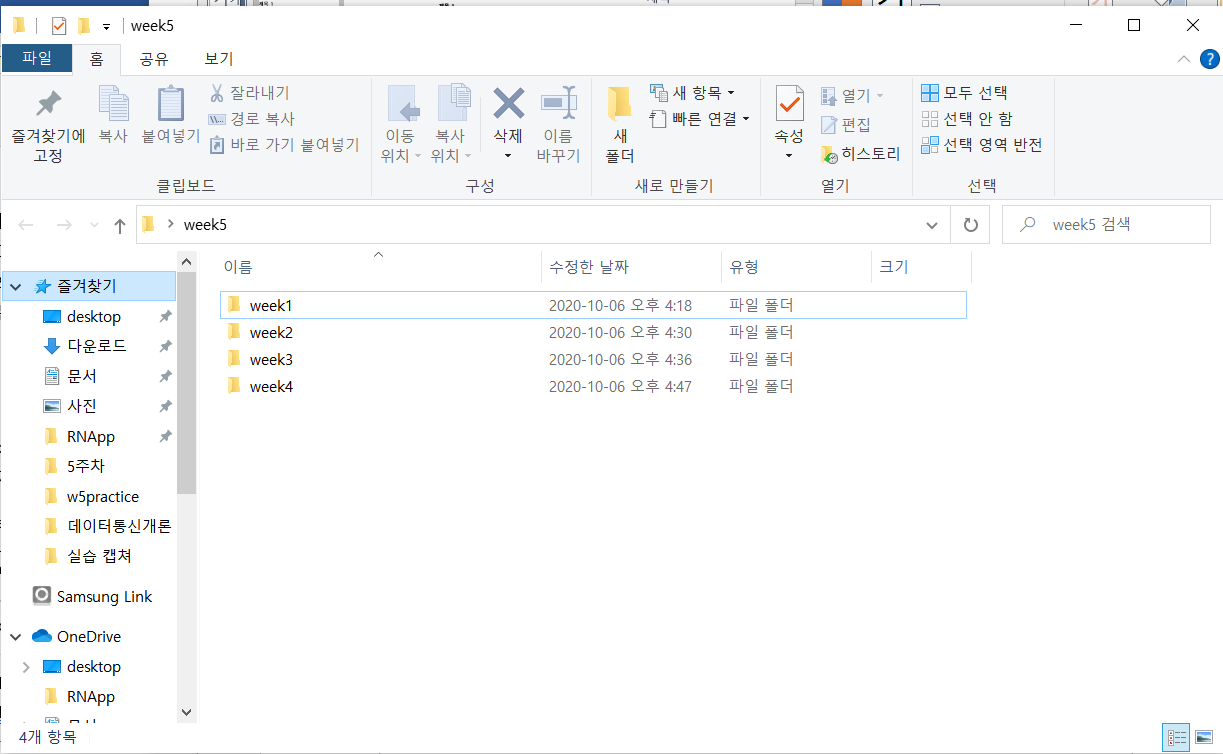
2. 실습 구현 내용

Github에서 4개의 레포지토리(repository)를 만든 후, 1주차부터 4주차까지의 실습 및 과제를 각각 저장소에 push한다.

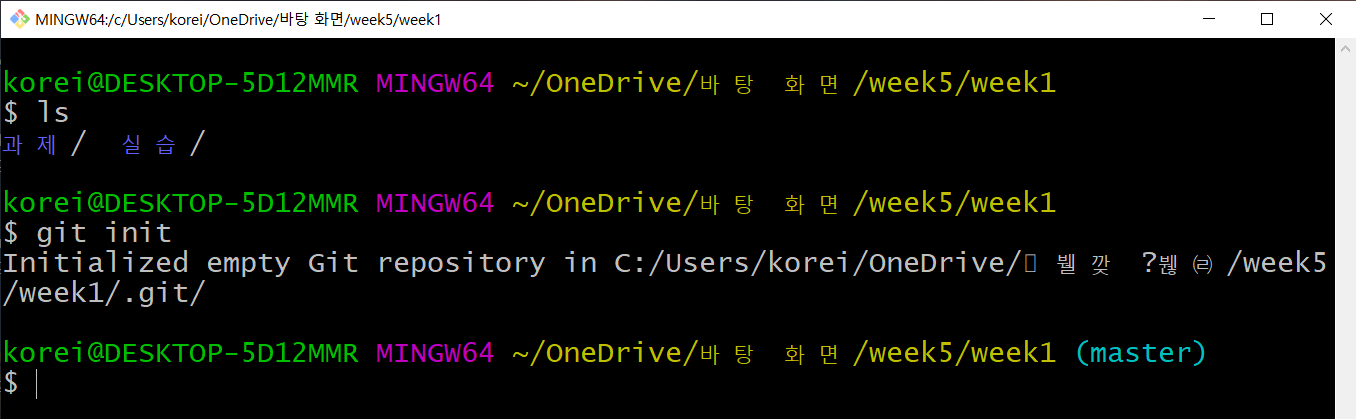
다음과 같이 Github 계정에 로그인해서 각 주차 실습 및 과제를 push할 repository를 만든다.

각 주차 실습과 과제를 push할 저장소의 이름은 comsil1\_week#(1주차는 comsil1\_week1)로 만든다. 인터넷 상의 다른 사람들도 볼 수 있도록 public으로 만든다.

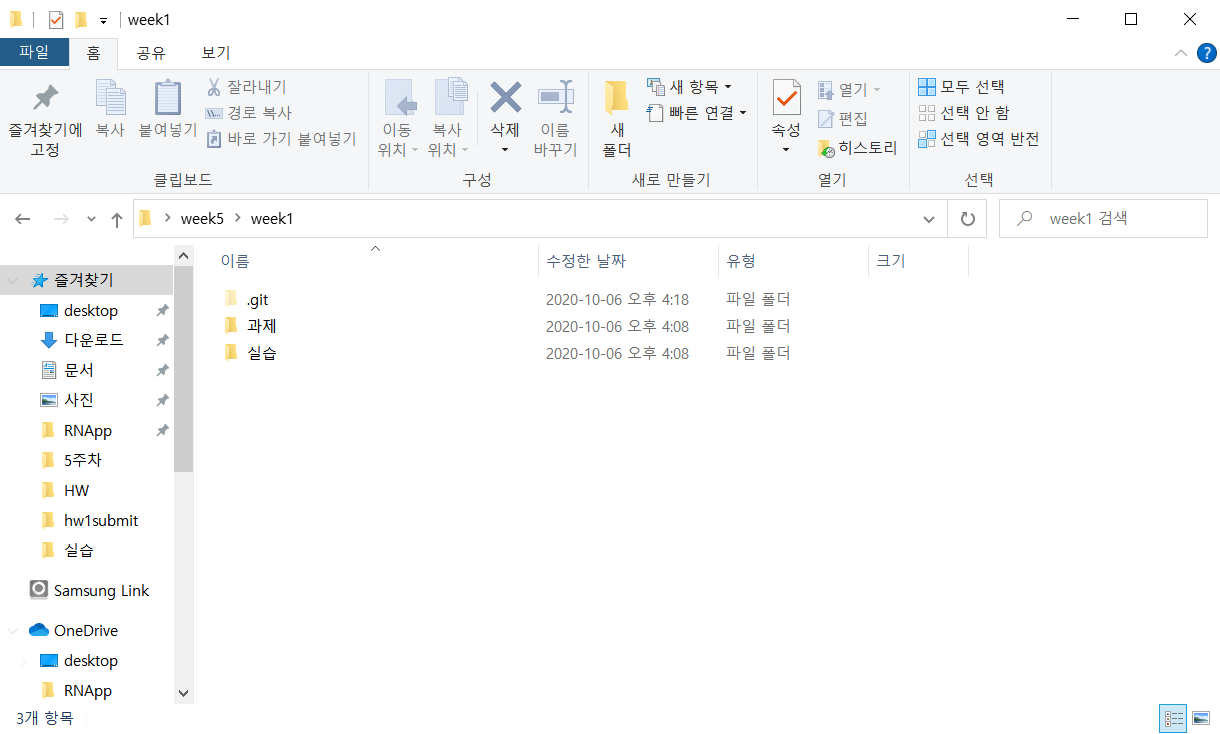
바탕화면의 week5 폴더에 1, 2, 3, 4주차 실습과 과제를 저장할 로컬 저장소를 각각 week1 ~ 4로 만든다.



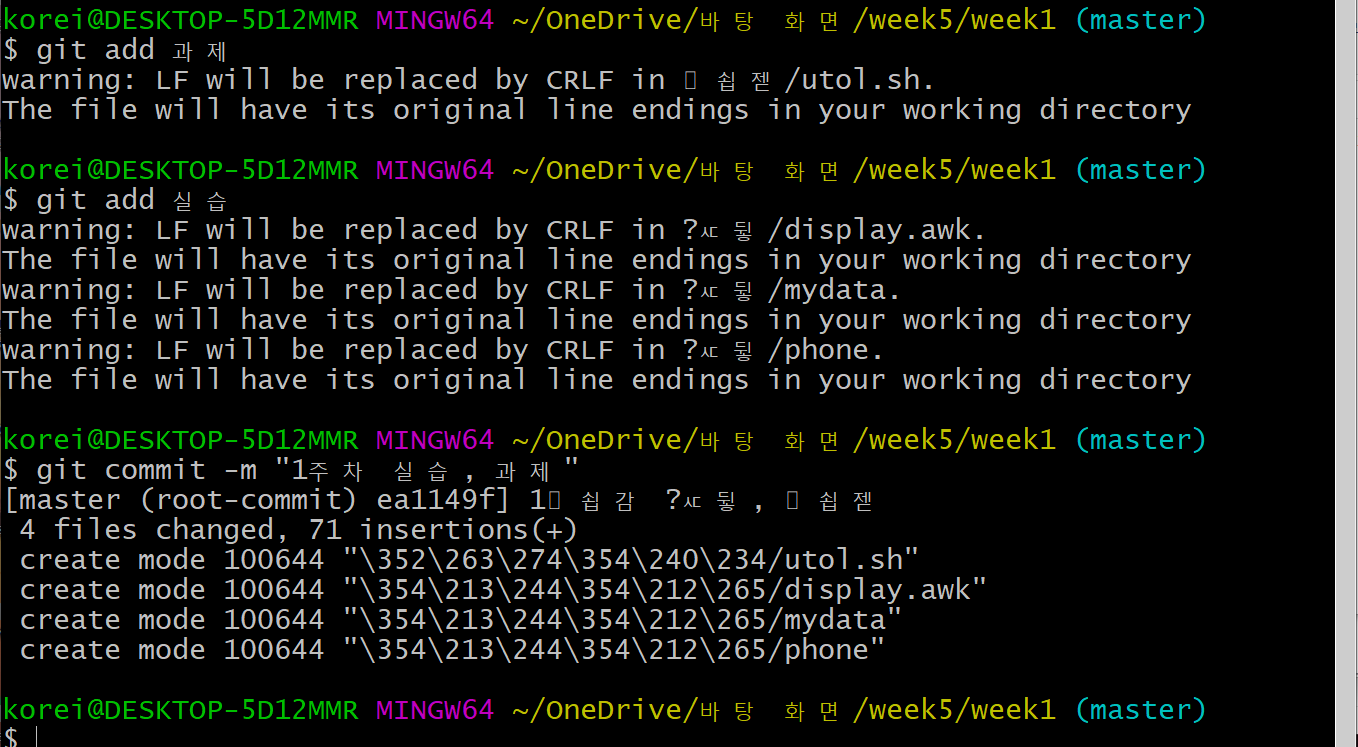
week1 폴더로 들어가서 1주차 실습 폴더와 과제 폴더를 옮겨 놓는다. 이 때 실습 폴더와 과제 폴더에는 어떤 종류의 파일이 있어야 한다. 저장할 파일을 week1로 옮겨 놓은 후 마우스 오른쪽 버튼을 클릭해 Git Bash here을 클릭한다.



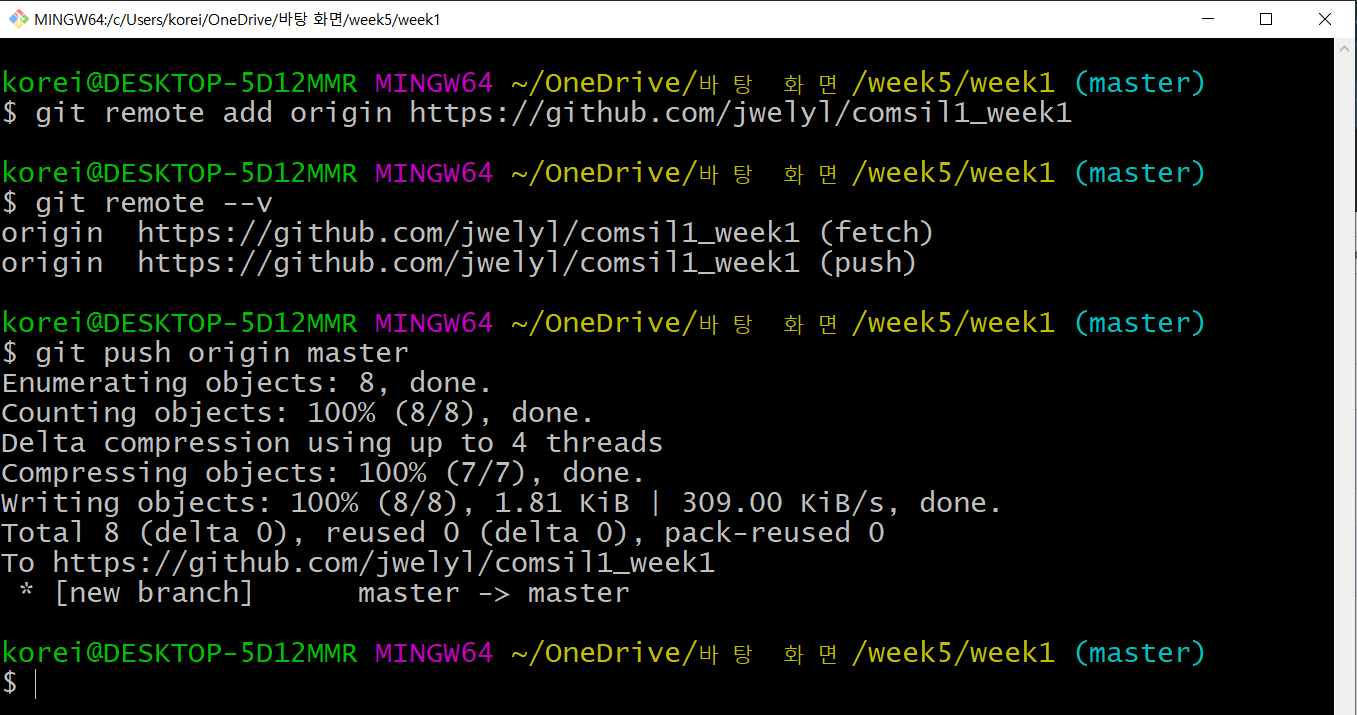
클릭하면 해당 디렉토리(week1) 경로로 Bash 창이 뜬다. week1을 로컬 저장소로 만들기 위해서 명령어로 git init을 입력한다. 입력하면 로컬 저장소가 초기화되었다고 나온다.



git init 명령어를 입력 후 week1 디렉토리를 보면 .git이라는 숨김폴더가 생긴 것을 알 수 있다.

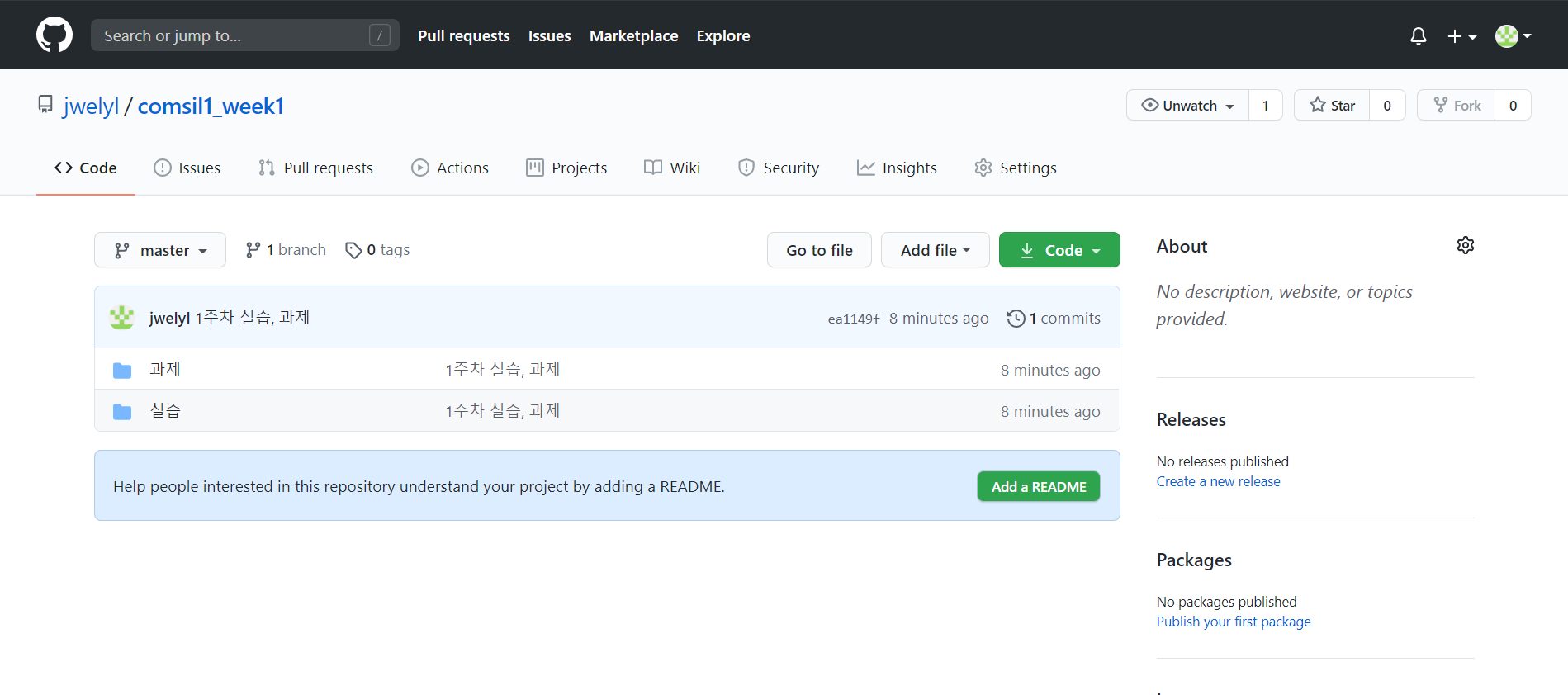


week1 디렉토리에 있는 실습 디렉토리와 과제 디렉토리를 원격저장소 comsil1\_week1에 push하기 위해 두 디렉토리를 add하고 (git add 실습, git add 과제) commit 한다. (git commit -m “1주차 실습, 과제”)



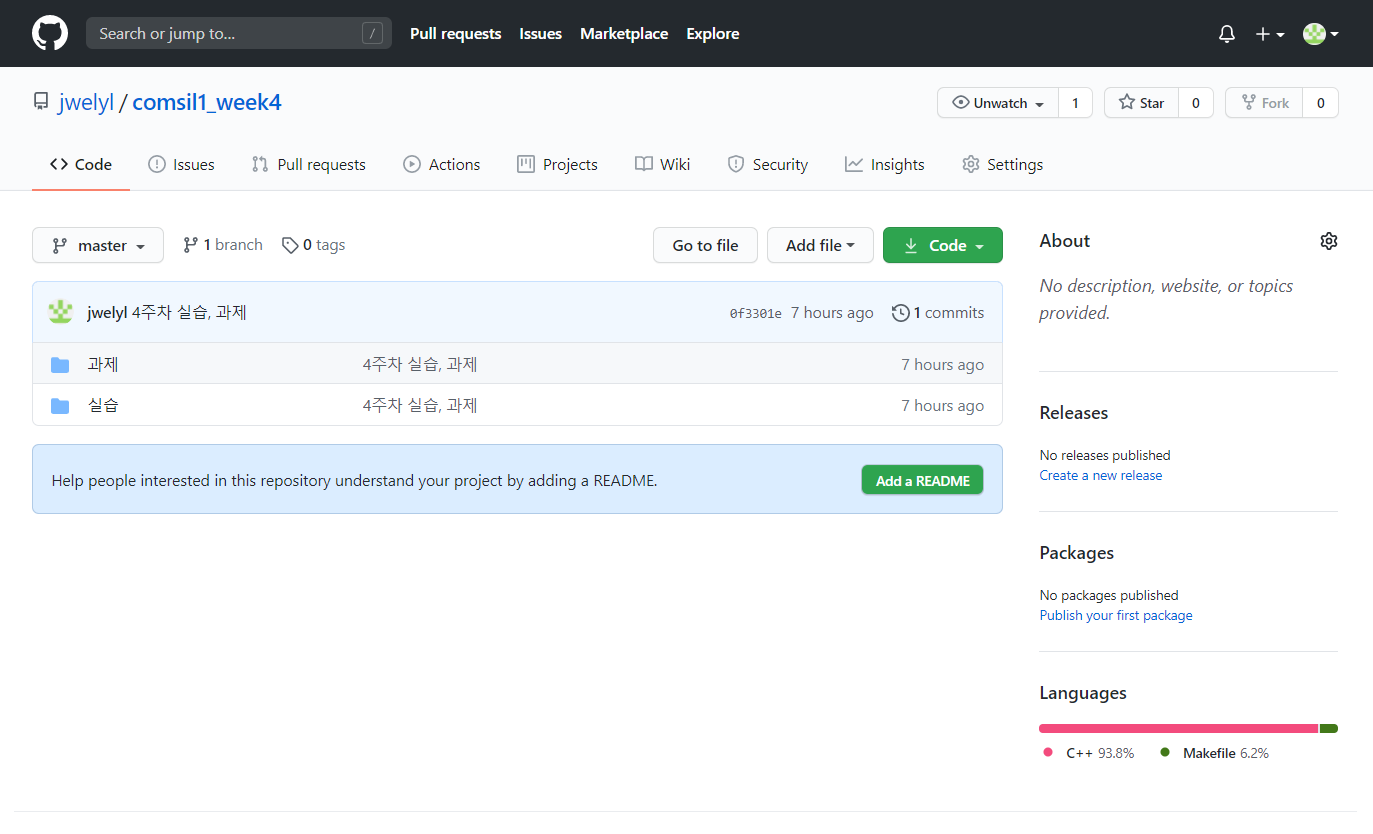
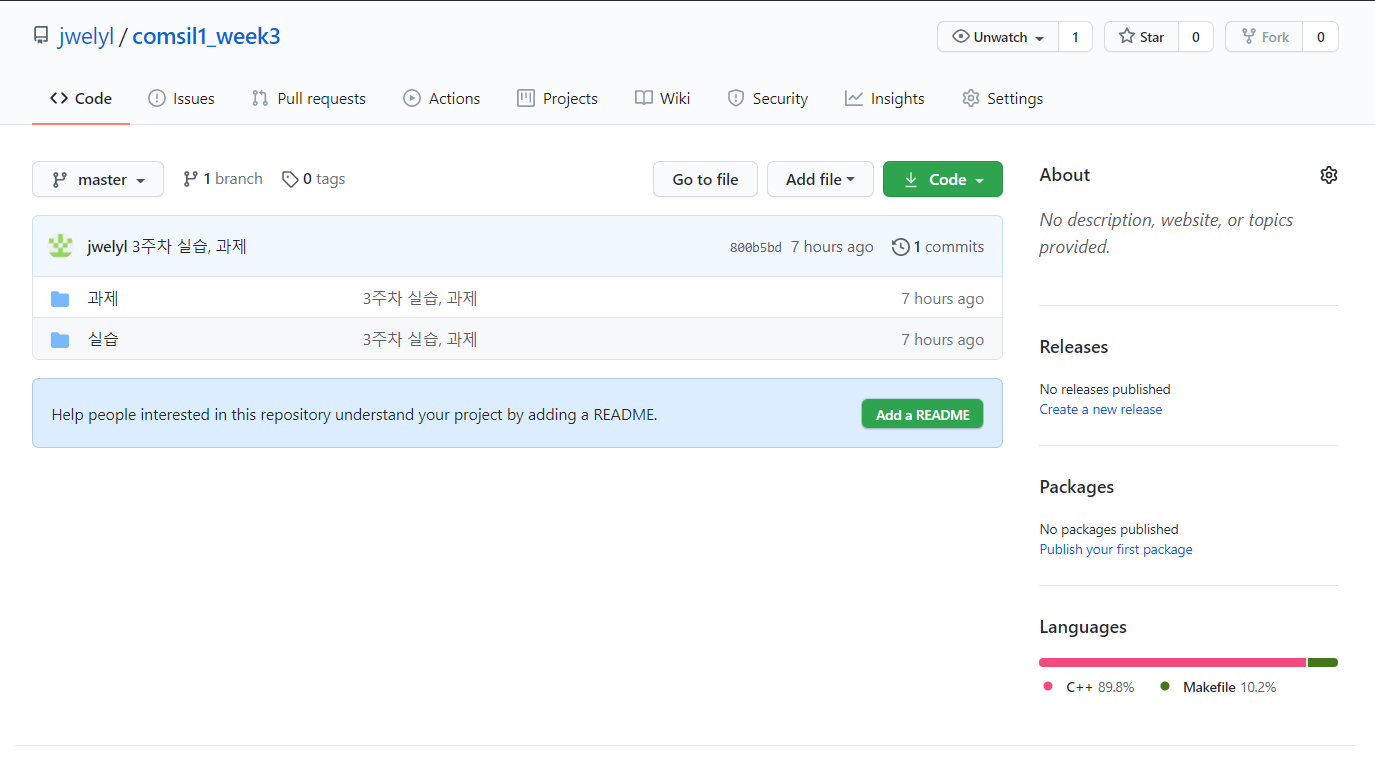
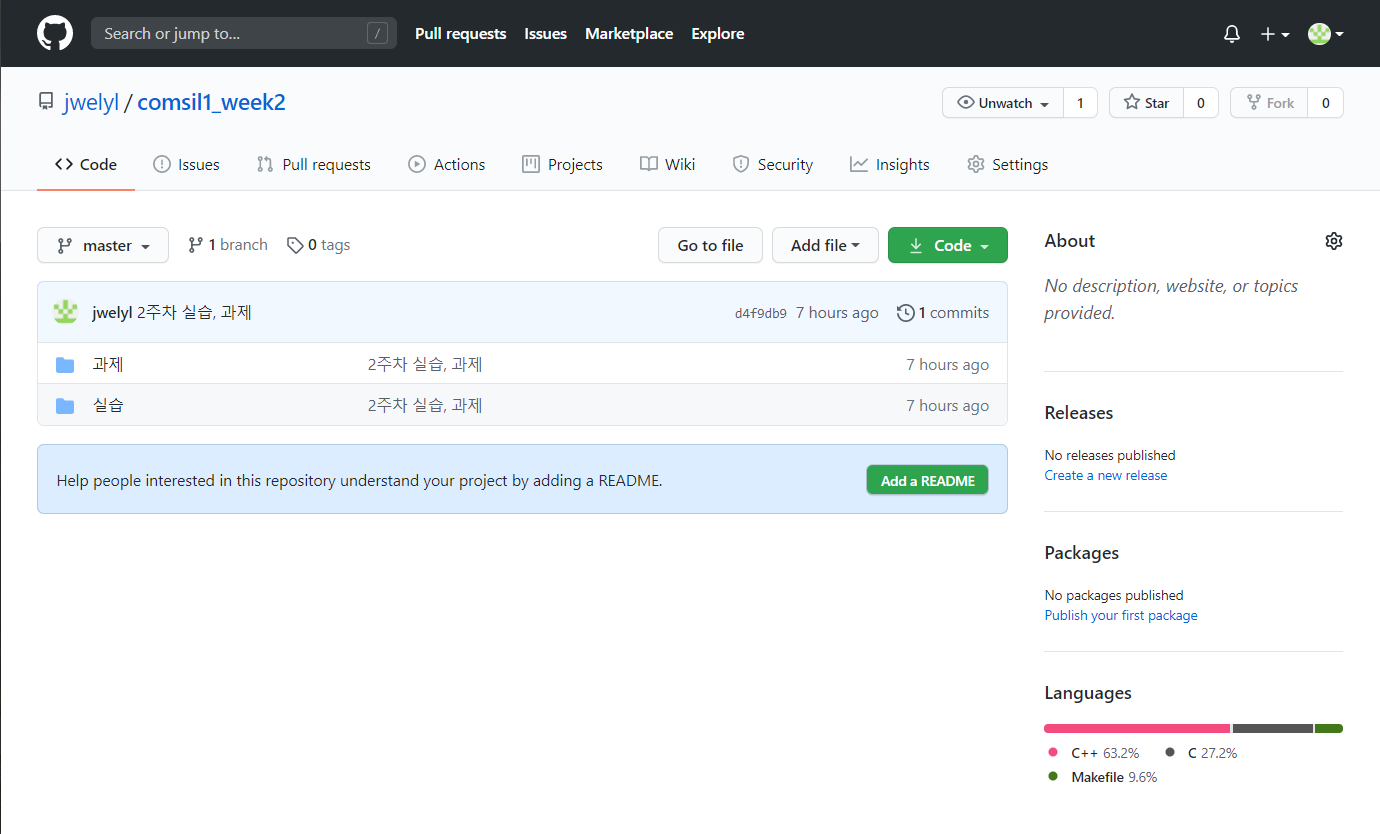
원격 저장소 comsil1\_week1와 로컬 저장소를 연결하기 위해 git remote add origin (원격 저장소 주소) 명령어를 입력한다. comsil1\_week1 저장소의 주소 <https://github.com/jwelyl/comsil1_week1>를 입력하여 명령어를 입력한다.

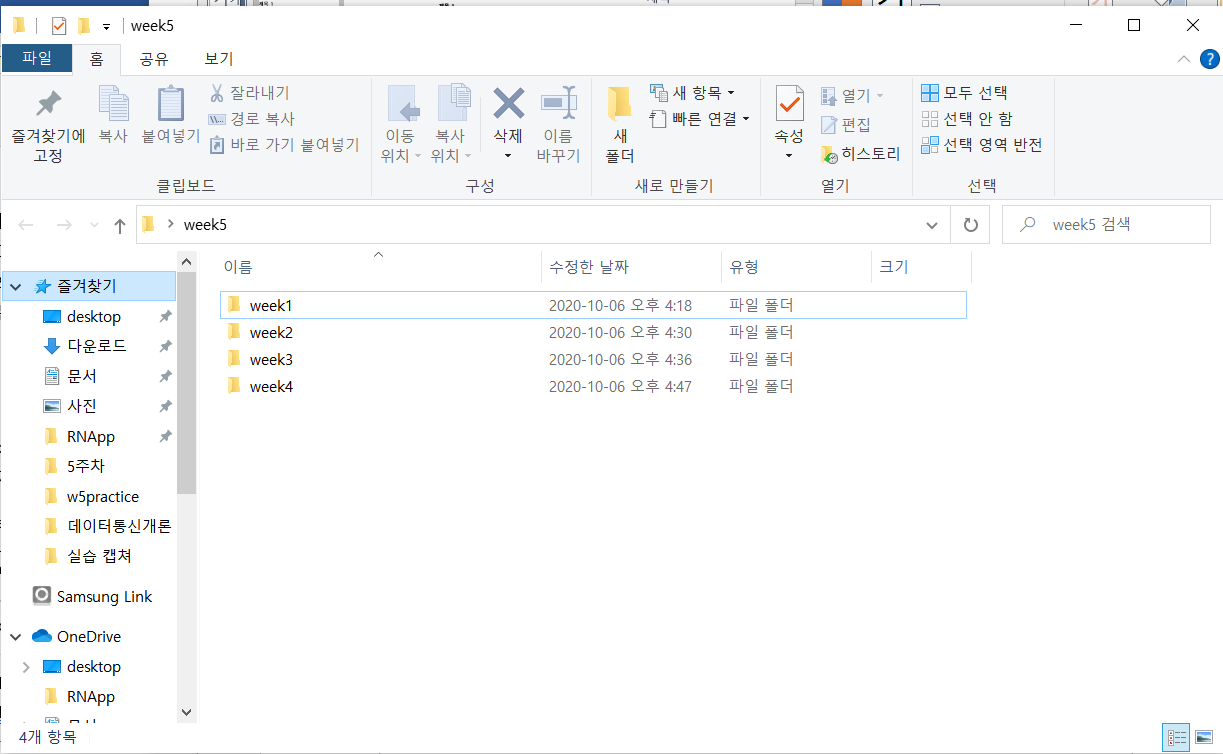
git remote --v 명령어로 연결된 원격 저장소를 확인할 수 있다. origin이라는 이름으로 comsil1\_week1 원격 저장소가 연결되었음을 확인하고 git push origin master 명령어를 입력하여 commit한 내용을 원격 저장소에 push한다.



comsil1\_week1 원격 저장소에 가보면 실습, 과제 폴더와 폴더 안의 파일들이 업로드된 것을 알 수 있다. 폴더 옆에 commit 할 때 입력한 메시지가 적혀 있는 것도 확인할 수 있다.

위와 같은 과정을 반복하면 comsil1\_week2, 3, 4 원격 저장소에 2,3,4주차 실습, 과제 소스 코드를 업로드할 수 있다. 다음은 완성된 2, 3, 4주차 원격 저장소를 캡쳐한 것이다.

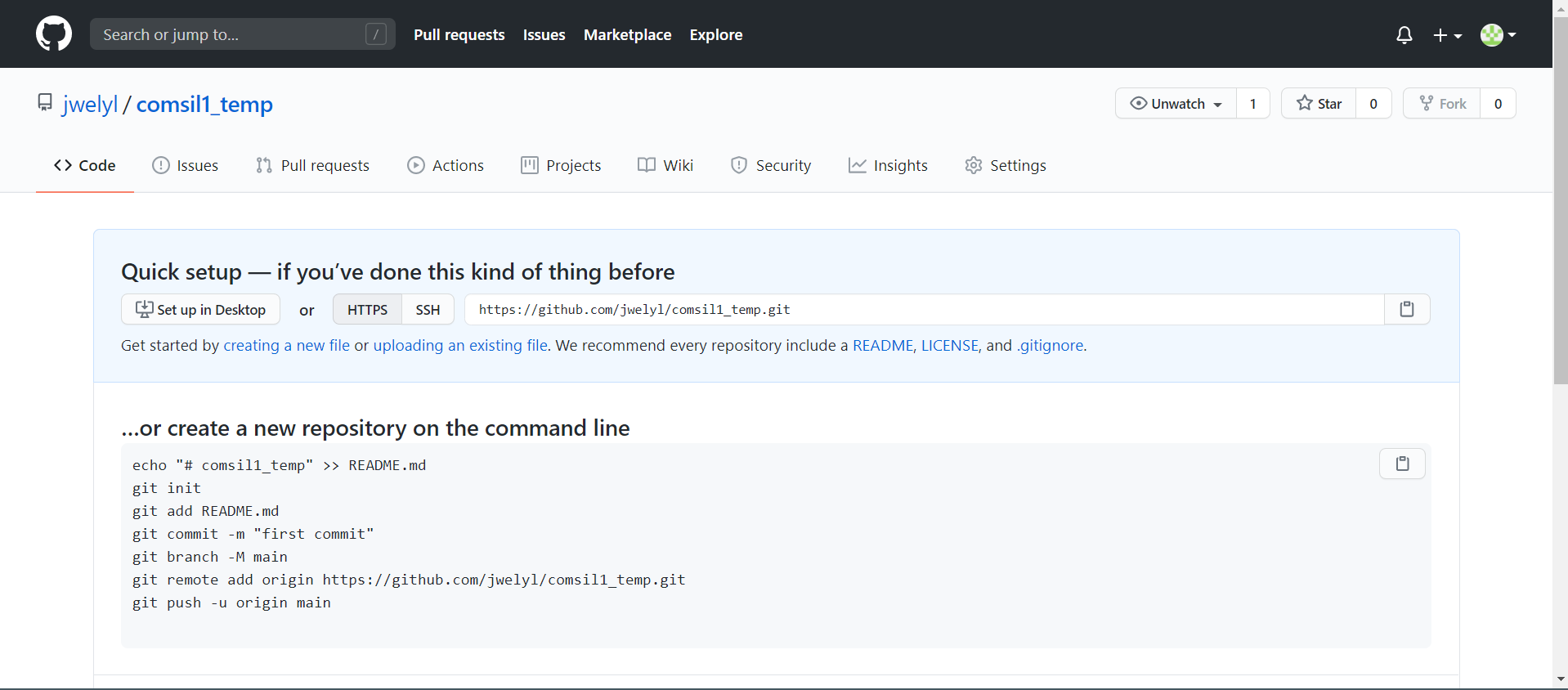


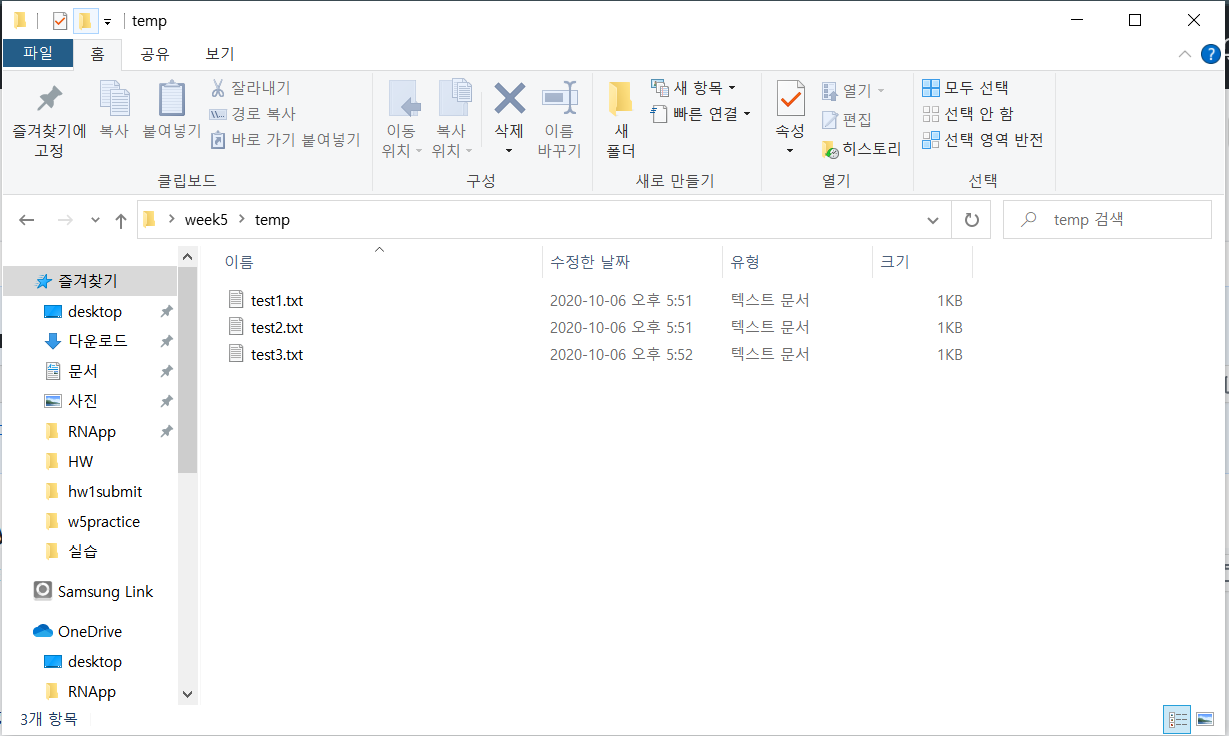


각 주차의 로컬 저장소는 위와 같다.

위에서 만든 comsil1\_week1~4 와는 다른 원격 저장소를 하나 만든 후, 2개 이상의 파일을 push한다. 그리고 해당 원격 저장소의 내용을 다른 로컬 저장소에 복제(clone)한다. 그 후 새로운 파일을 하나 추가하여 push한다.

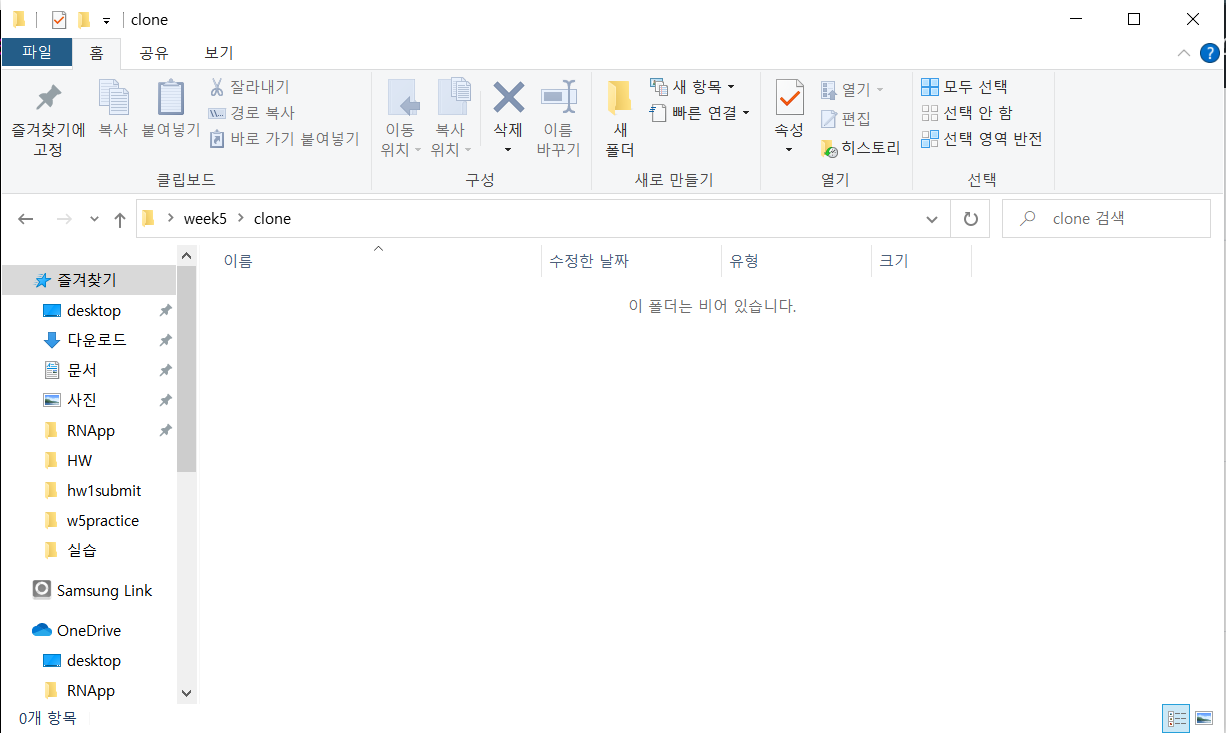
새로운 원격 저장소를 comsil1\_temp라고 하자.

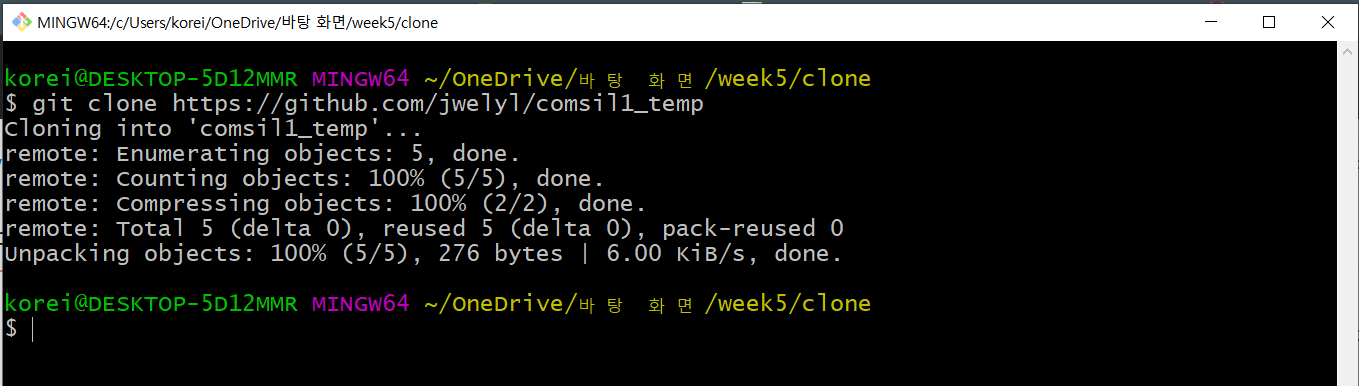




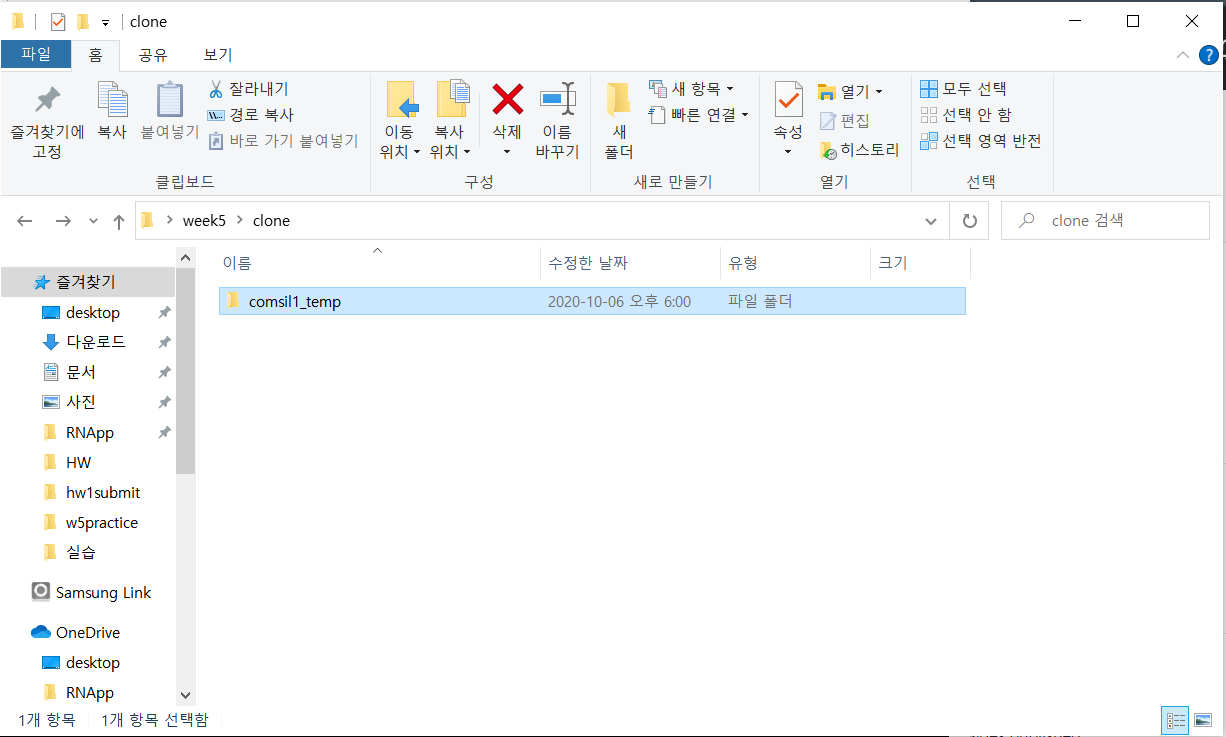
원격 저장소 comsil1\_temp와 연결할 로컬 저장소 temp를 바탕화면의 week5 폴더 안에 만든다. week1 폴더를 로컬 저장소로 초기화할 때와 마찬가지로 bash를 실행시켜 git init으로 초기화해주고 add, commit 명령어로 test1.txt, test2.txt, test3.txt 파일을 commit한다. Commit 메시지는 clone test라고 했다. 원격 저장소 comsil1\_temp와 연결 후 push한다.

comsil1\_temp 원격 저장소를 복제하기 위해 바탕화면의 week5 폴더에 clone 폴더를 만든 후 git bash를 실행한다.

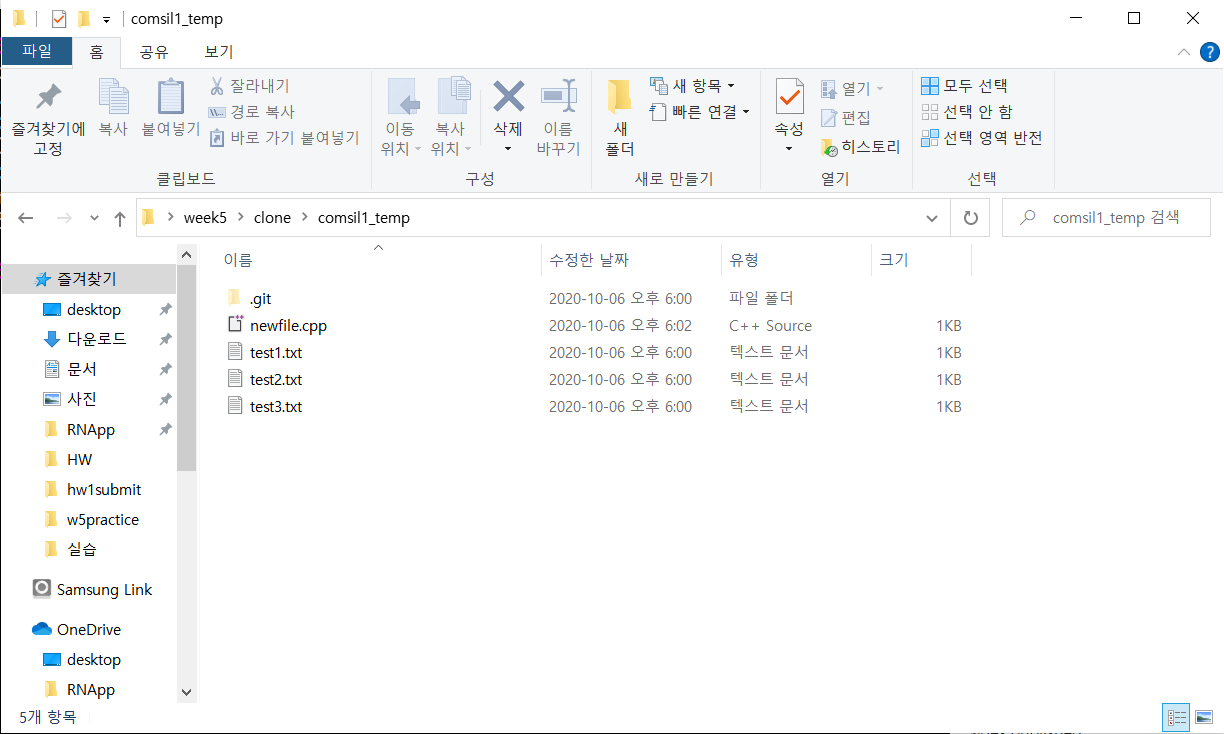




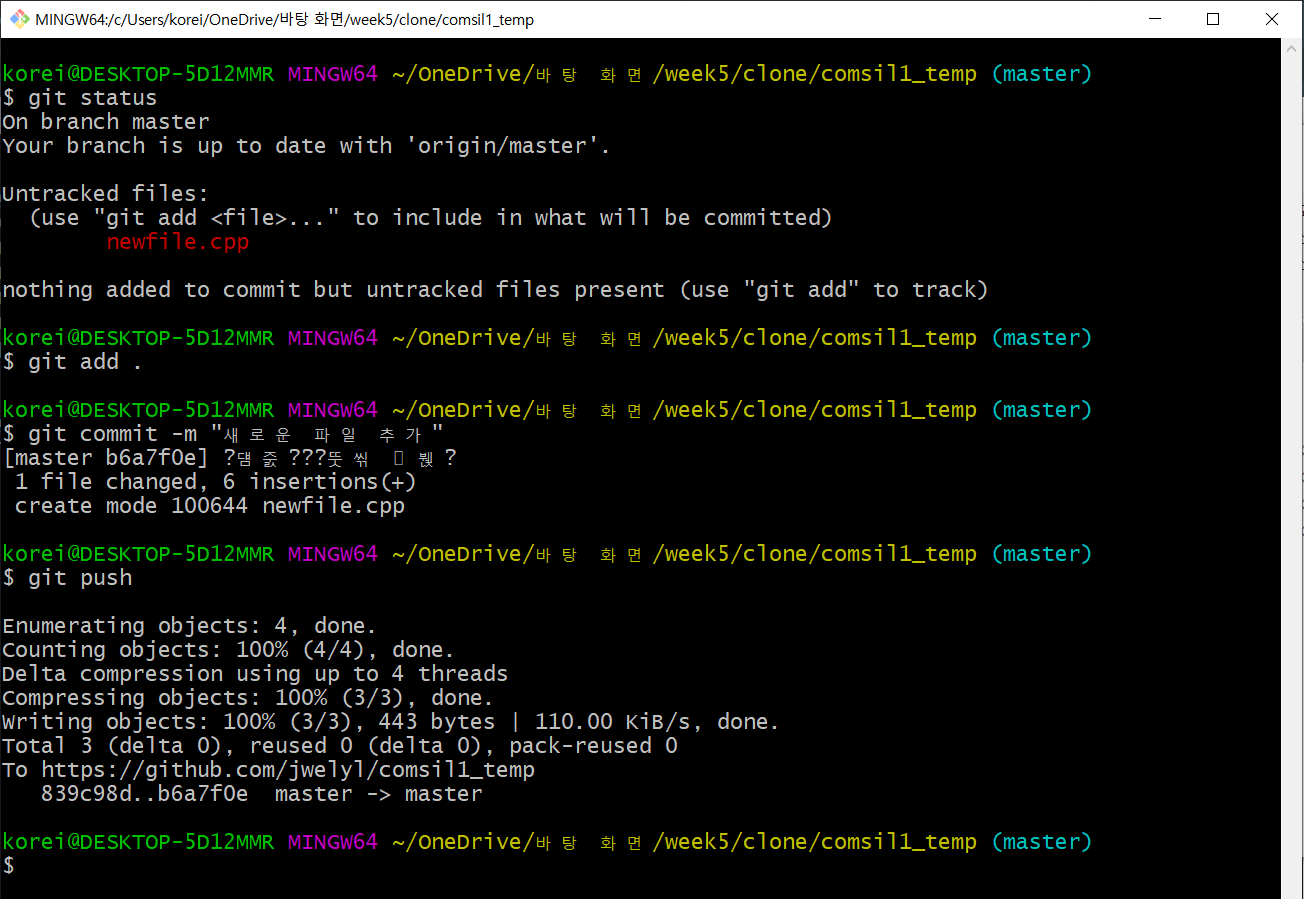
명령어 git clone https://github.com/jwelyl/comsil1\_temp를 입력하면 clone 폴더에 comsil\_temp 저장소가 복제된다.



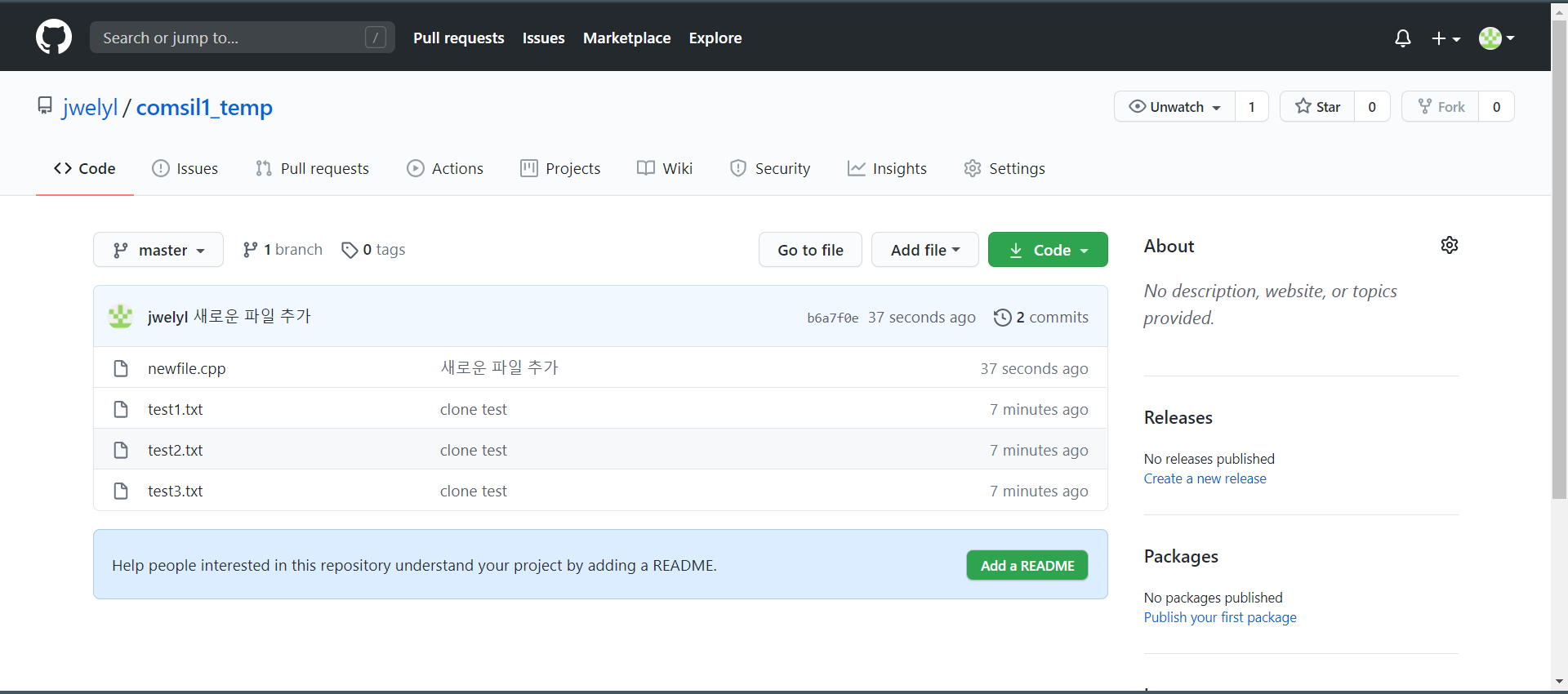
clone 디렉토리를 보면 comsil1\_temp 원격 저장소가 복제된 것을 알 수 있다. comsil1\_temp 폴더로 들어간다.



새로운파일 newfile.cpp를 추가한다.



Git bash를 실행 후 git status 명령어를 입력하면 새로운 파일 newfile.cpp가 생성되었지만 아직 commit 되지 않을 것을 알 수 있다. Push할 수 있도록 add와 commit 해준다. Commit 메시지는 “새로운 파일 추가”라고 한다. 복제된 저장소는 이미 원격 저장소 comsil1\_temp와 연결되어 있으므로 따로 연결하는 명령어를 입력할 필요는 없다. push 명령어를 입력하면 바로 원격 저장소로 새로 생성한 newfile.cpp가 업로드 된다.



newfile.cpp가 추가된 것을 알 수 있다. 변화가 없는 test1.txt~test3.txt는 commit 메시지가 처음 commit했을 때와 같고 새로 생성 후 commit한 newfile.cpp만 commit 메시지가 다른 것도 확인할 수 있다.

3. 실습 환경

OS : Windows 10

CPU : Intel® Core(TM) i3-5005U CPU @ 2.00GHz

RAM : 8.00GB

4. 실습 결과 및 분석

Git bash로 로컬 PC의 디렉토리를 로컬 저장소로 만들면 온라인에 연결되어 있지 않아도 백업, 버전 관리가 가능하다. 변경 사항이 생겼을 때 변경 사항이 생긴 파일을 일일이 새로 만들고 이름을 다시 짓는 수고를 할 필요가 없고, 그렇게 했을 때 생기는 혼란을 방지할 수 있다. 로컬 저장소에서의 백업 및 버전 관리는 오프라인에서도 가능하다는 장점이 있지만 PC 사용자 외에는 접근이 불가능하기 때문에 협업은 불가능하고 로컬 저장소, 더 나아가 PC 자체에 문제가 생기면 백업의 의미가 사라진다.

Git으로 생성한 로컬 저장소를 Github의 원격 저장소와 연결하여 이러한 단점을 해결할 수 있다. 로컬 저장소에 저장한 내용을 Github의 서버에 저장하여 로컬 PC에 문제가 생겨도 Github 원격 저장소에 백업되어 있는 소스 파일을 불러올 수 있다. 또한 다른 사람들도 Github 원격 저장소에 접근 가능하게 설정하면 다른 사람들과의 협업도 가능해진다. 온라인으로 연결만 되어 있다면 Clone 명령어를 이용하여 원격 저장소에 있는 데이터를 쉽게 로컬 저장소로 옮길 수도 있어서, 어떠한 로컬 저장소에 있는 데이터를 USB, 외장 하드와 같은 저장 매체 없이도 쉽게 다른 로컬 저장소로 옮길 수 있다.

사용한 명령어

git init : 해당 디렉토리를 로컬 저장소(repository)로 초기화한다.

git add : 변경된 내용을 commit하기 전에 staging area로 올린다. add 뒤에 add 할 파일이나 디렉토리를 입력하면 해당 파일, 디렉토리만 add 되고, \*이나 .을 입력하면 해당 저장소의 모든 파일, 디렉토리가 add 된다.

git commit : add로 올린 내용을 버전으로 확장시킨다. -m 옵션을 줘서 바로 메시지를 입력할 수도 있다.

git status : 변경 사항을 확인한다. 변경되었지만 add되지 않았을 경우 빨간 색으로 표시된다. Add된 내용은 초록색으로 표시된다.

git log : commit한 버전들을 확인한다. Commit message를 학인할 수 있다.

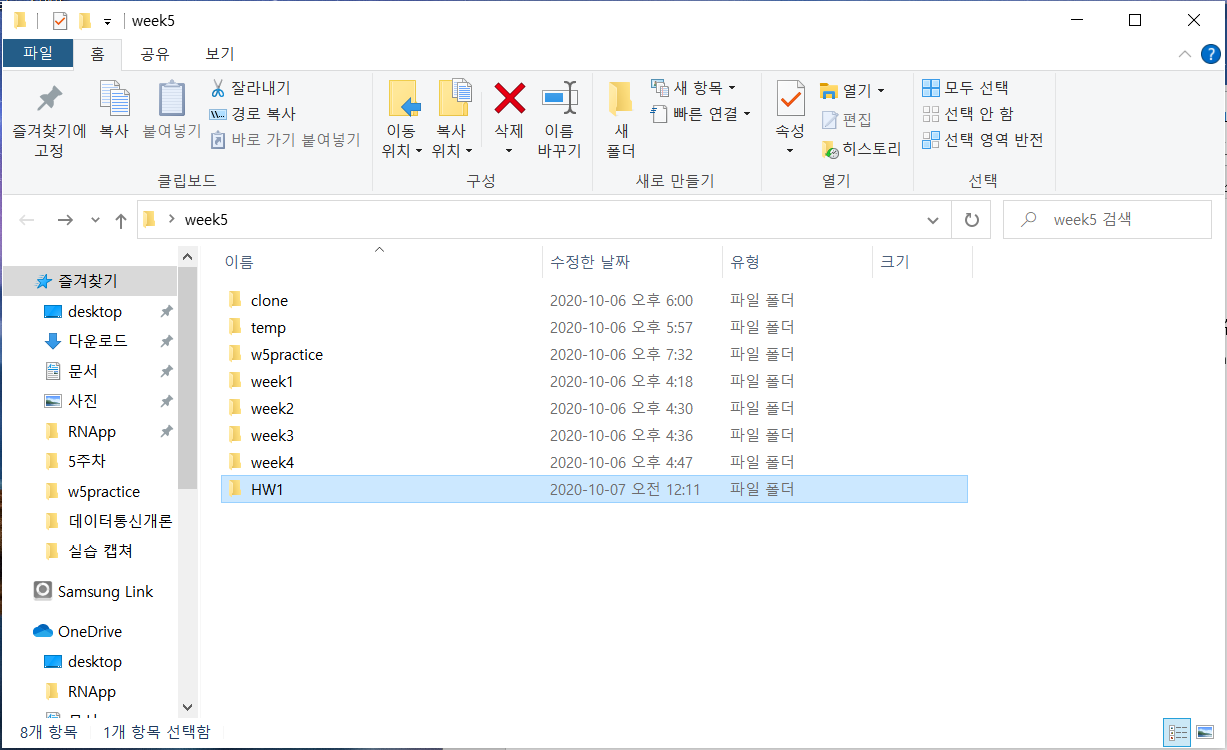
git remote add (주소 이름) (원격 저장소 주소) : 원격 저장소와 로컬 저장소를 연결한다. 주소 이름에는 원격 저장소 주소를 대표할 이름을 설정한다. 보통 origin으로 설정한다. 해당 명령어를 통해 설정한 주소 이름으로 원격 저장소에 push가 가능하다.

git push (주소 이름) (branchname) : 주소 이름에 해당하는 원격 저장소에 branchname에 해당하는 버전을 push한다. 주소 이름을 origin이라 했고, 따로 branch를 만들지 않았다면 git push origin master라고 입력하면 된다.

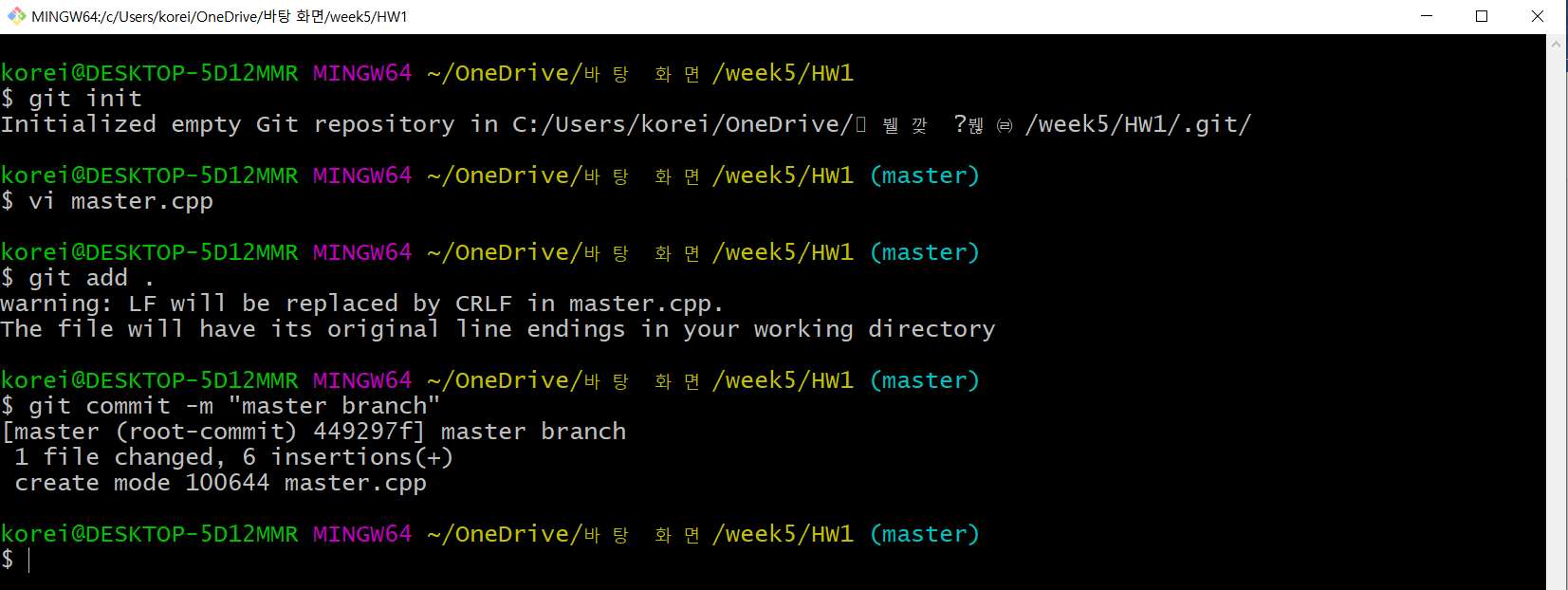
git clone (원격 저장소 주소) : 해당 주소의 원격 저장소의 내용을 해당 디렉토리에 복제한다.

5. 과제

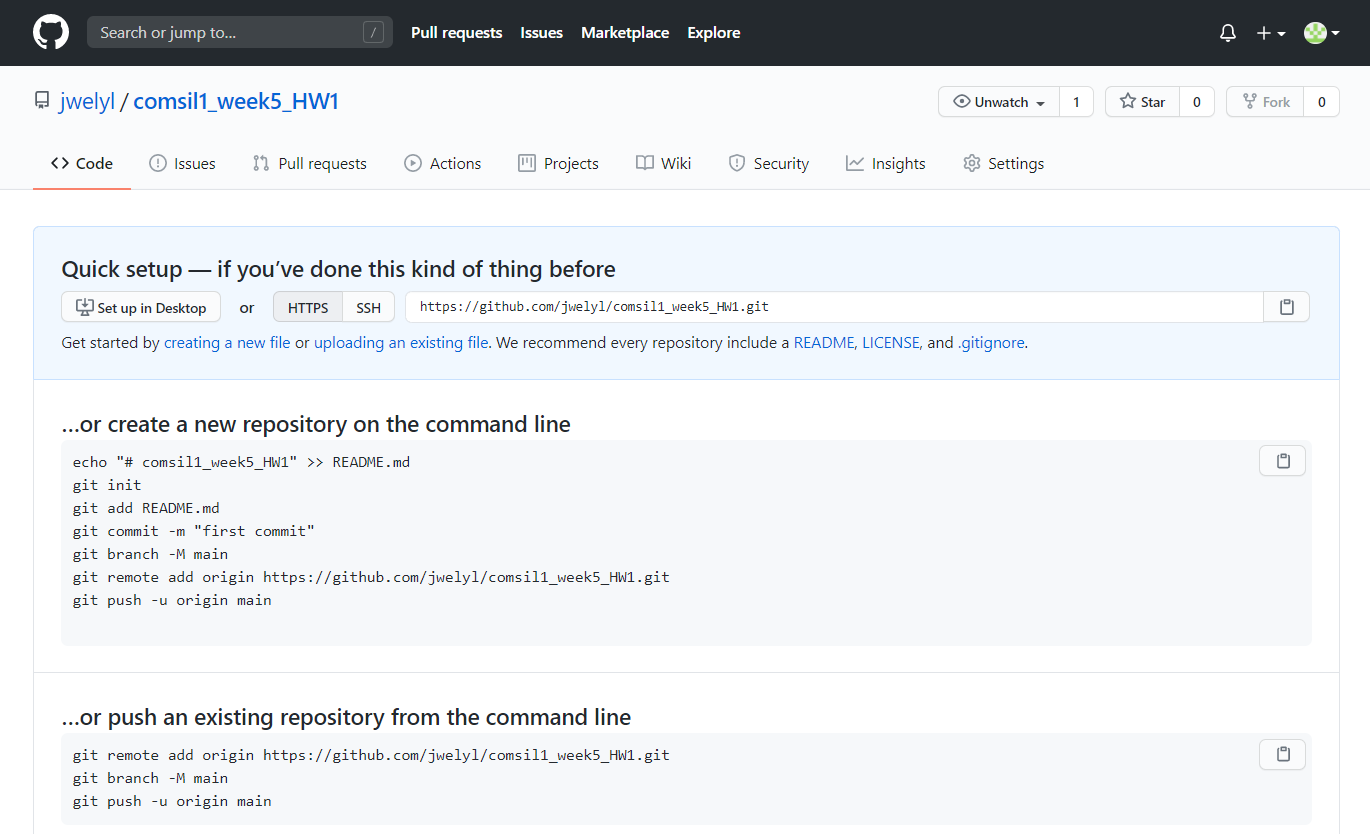
첫 번째 과제로 3개 이상의 Branch를 만들고, 각 Branch의 내용을 다르게 수정하여 commit 후 원격저장소에 push한다. 모든 branch를 master branch에 merge 한다. 또한 merge 과정에서 충돌이 일어날 수 있는 상황을 만들어보고 이를 해결해본다.



첫 번째 과제를 위한 로컬 저장소를 만들기 위해 week5 폴더에 HW1 폴더를 만든다.

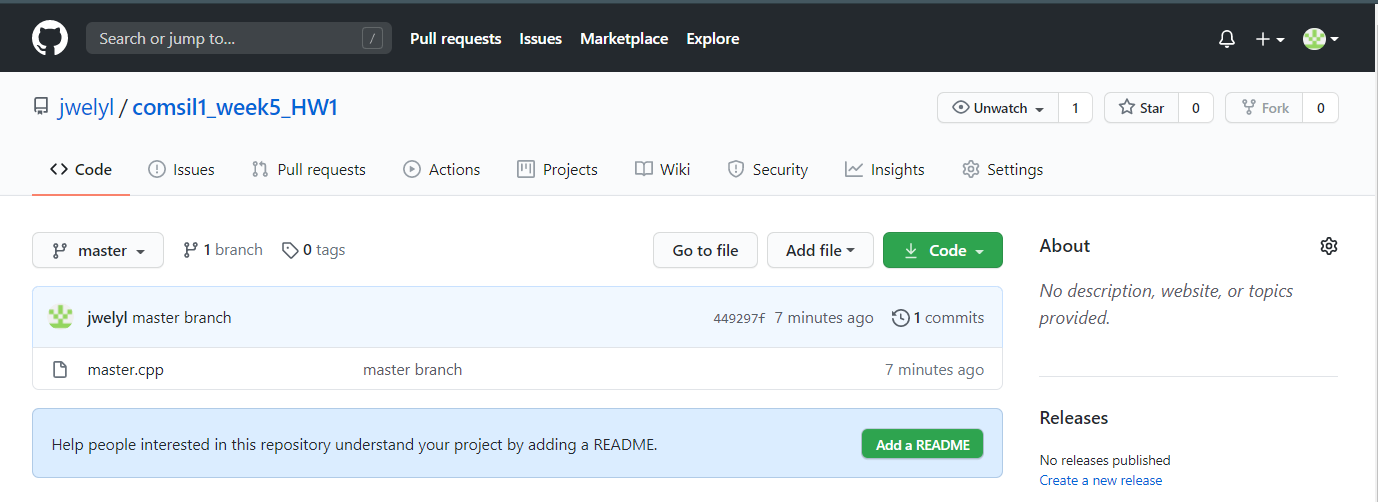


HW1 디렉토리를 로컬 저장소로 초기화하고 master.cpp 파일을 만들어 master branch라는 메시지로 commit한다.

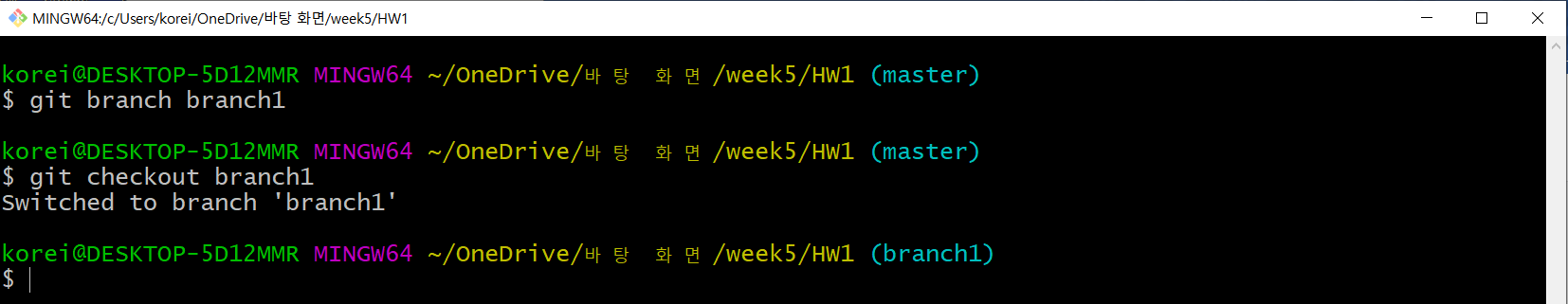
HW1 로컬 저장소를 저장할 원격 저장소 comsil1\_week5\_HW1을 만들고



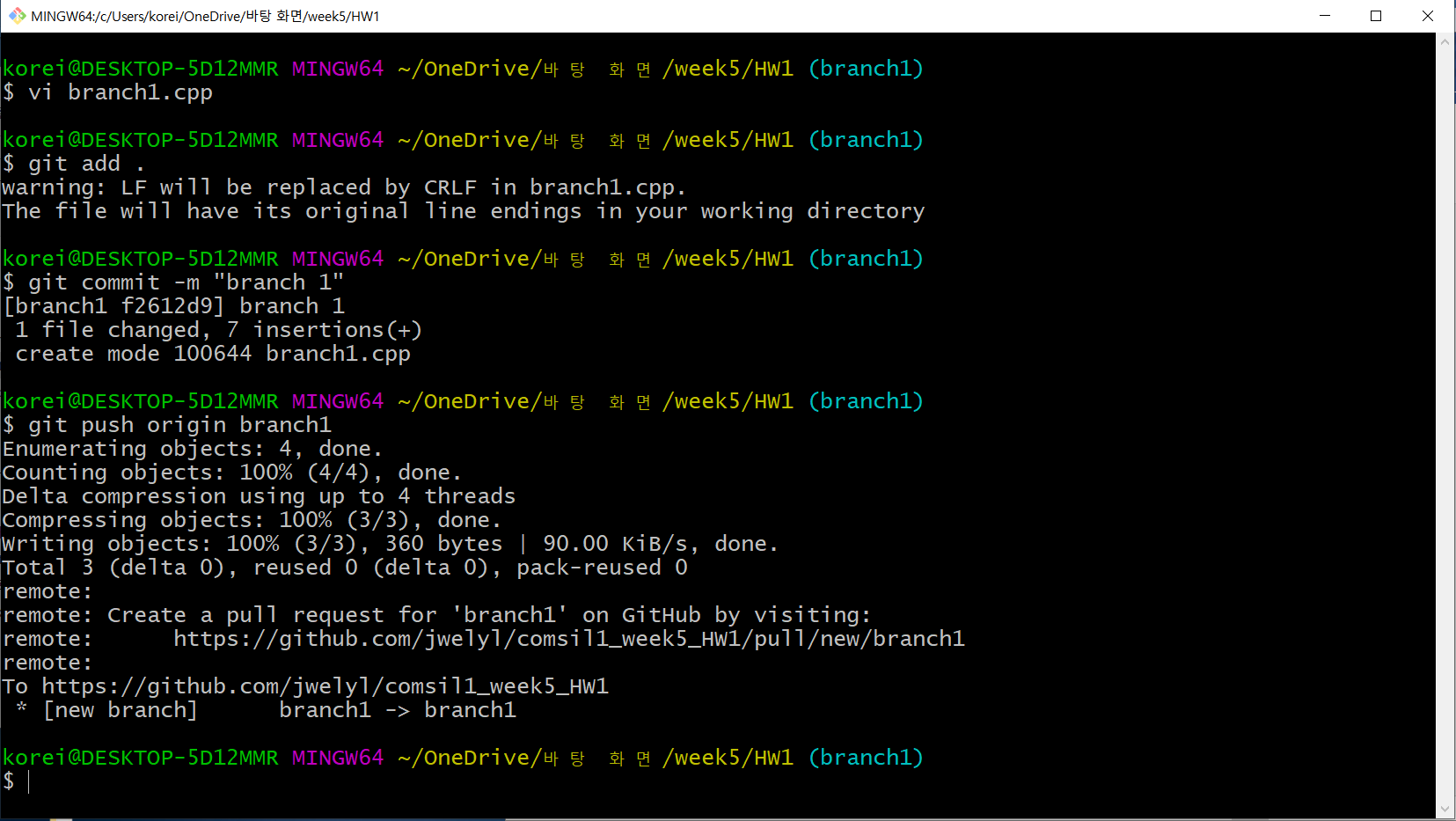
로컬 저장소 HW1을 comsil1\_week5\_HW1 원격 저장소와 연결 후 push한다. 명령어 git push origin master에서 origin은 원격 저장소의 이름이고, master는 master branch를 의미한다. 다른 branch를 push할 때는 해당 branch 이름을 입력하면 된다.

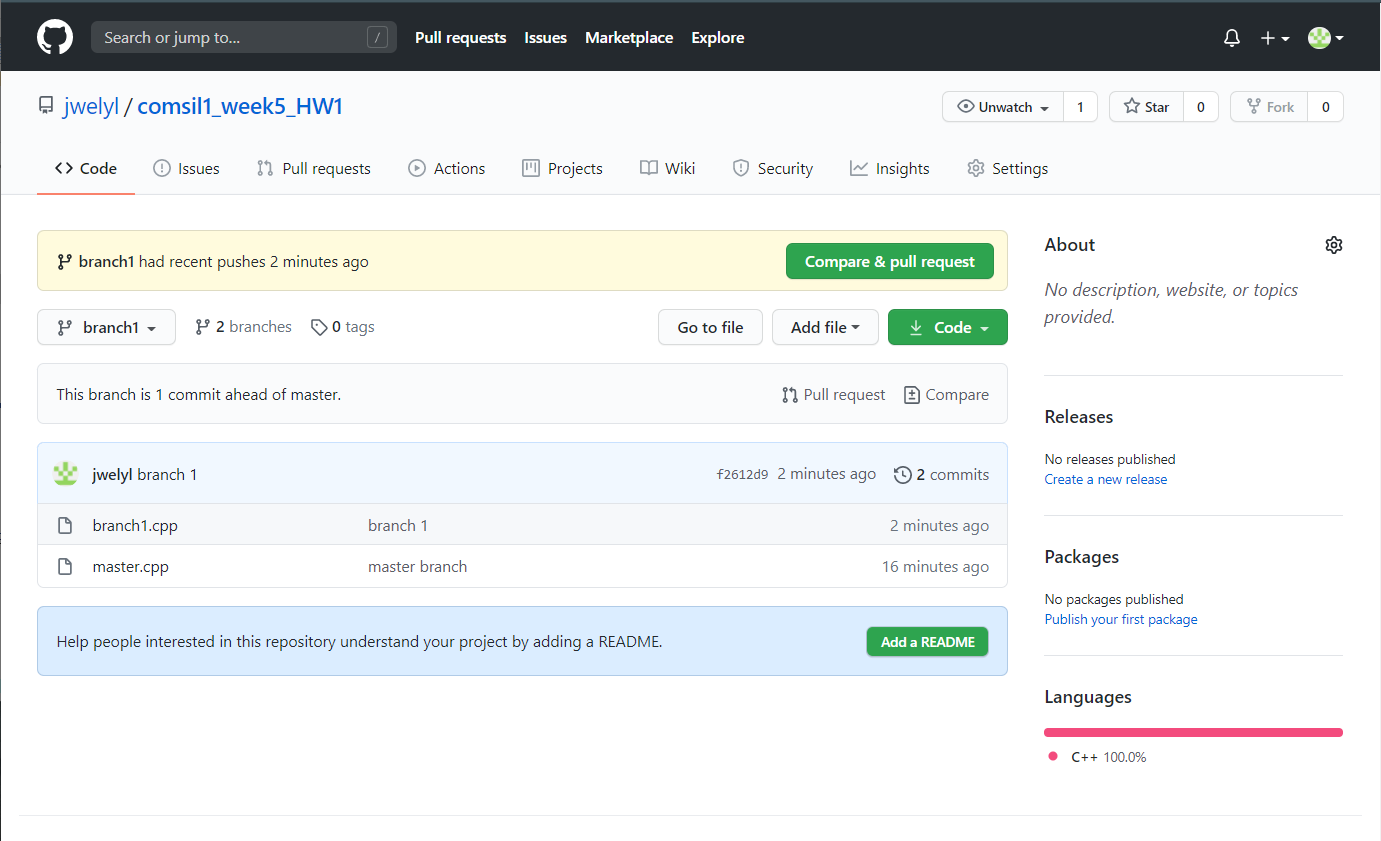


push된 원격 저장소이다. Master branch라는 commit 메시지와 master.cpp 파일을 확인할 수 있다.



새로운 branch를 만들기 위해 git branch branch1 명령어를 입력하면 branch1이 만들어진다. Bash의 주소 옆의 하늘색 글자가 현재 branch를 의미하며 현재 master branch에 있음을 알 수 있다. git checkout branch1을 입력하면 branch1 branch로 이동할 수 있다.

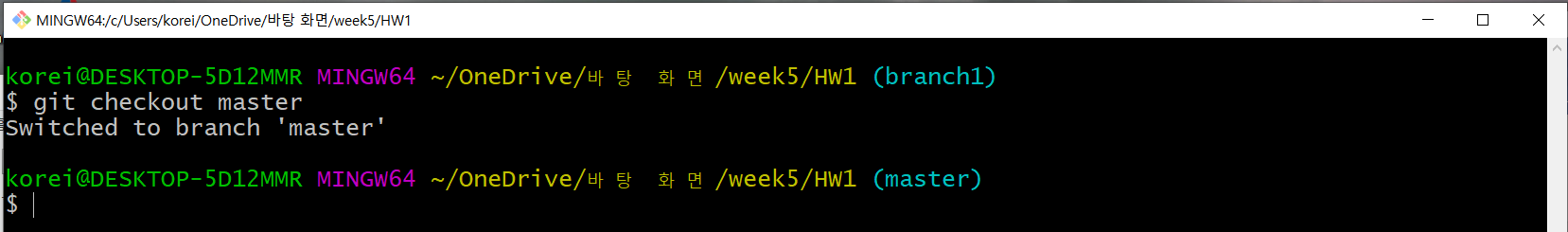
branch1에 소스 파일 branch1.cpp를 만들고 add와 commit을 한다. Commit 메시지는 branch 1이다. Push 할 때 master가 아닌 branch1으로 push한다.



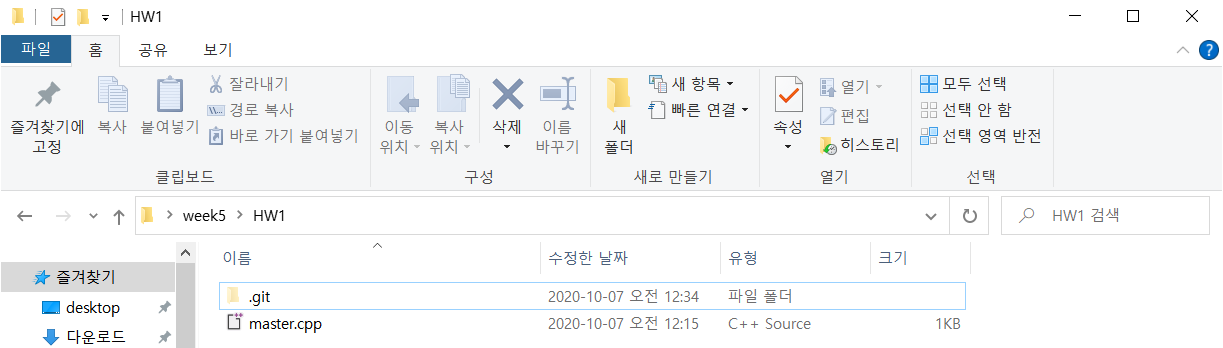
branch1 원격 저장소에 branch1이 push 된 것을 알 수 있다.



로컬 저장소의 branch1.cpp 파일을 확인할 수 있다.

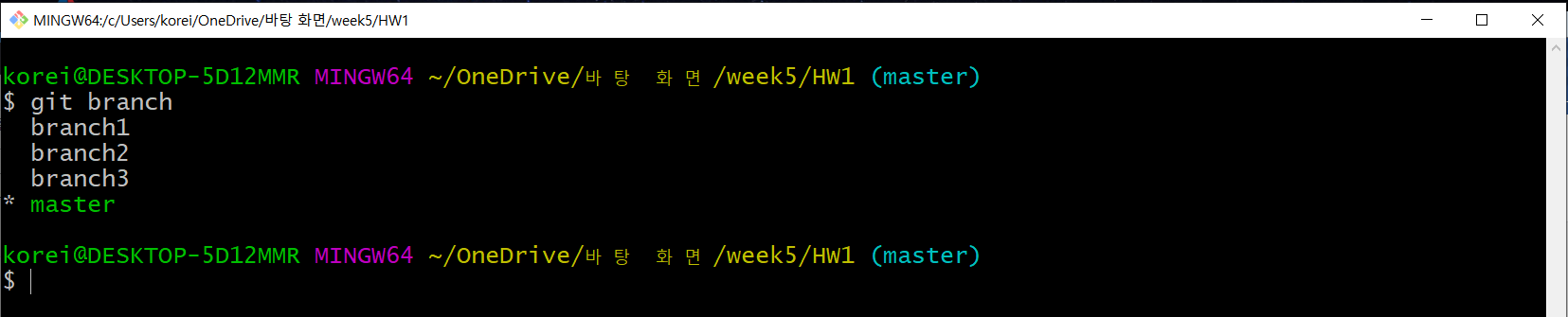


Master branch에 branch2를 만들기 위해 master branch로 다시 돌아오면

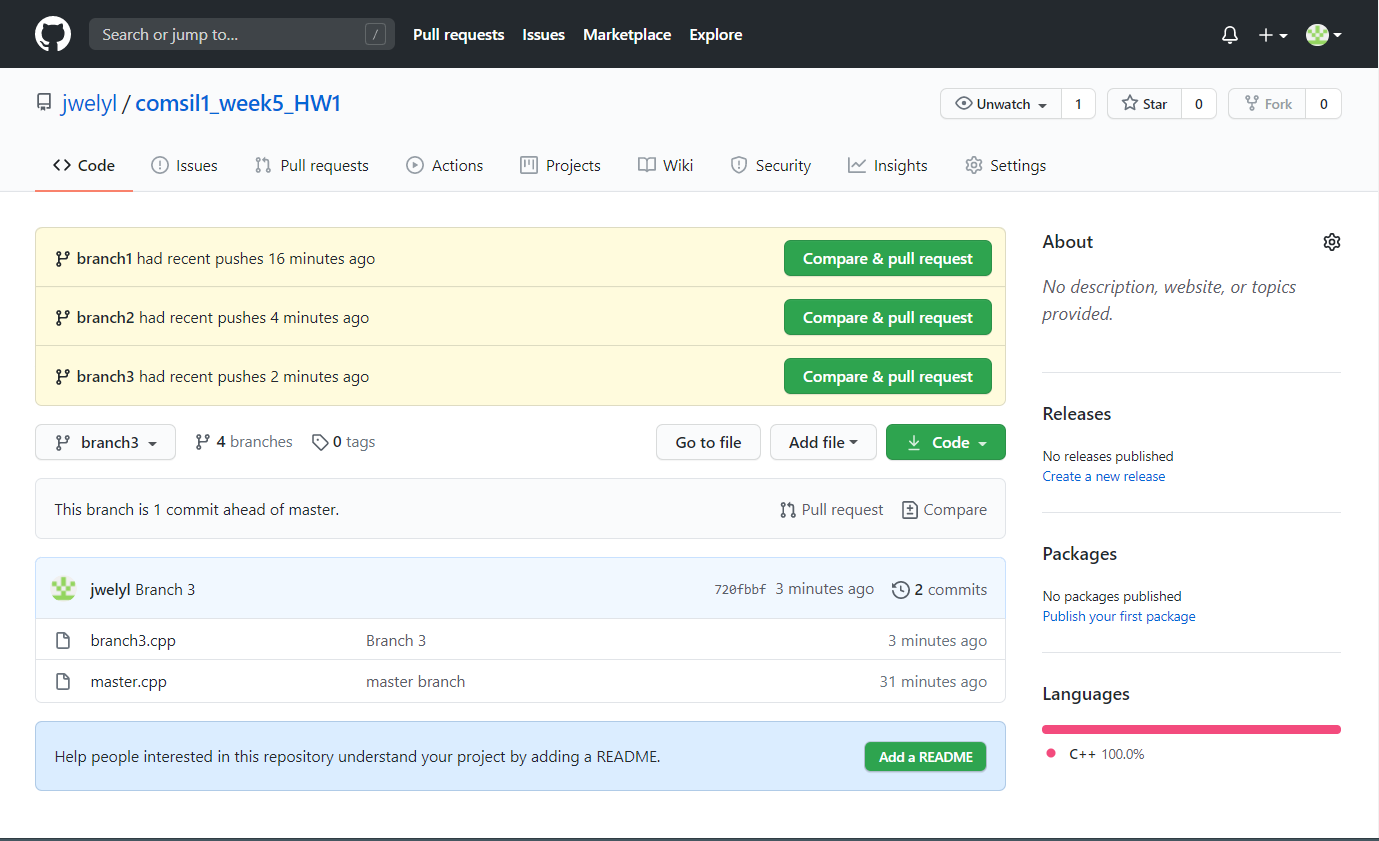
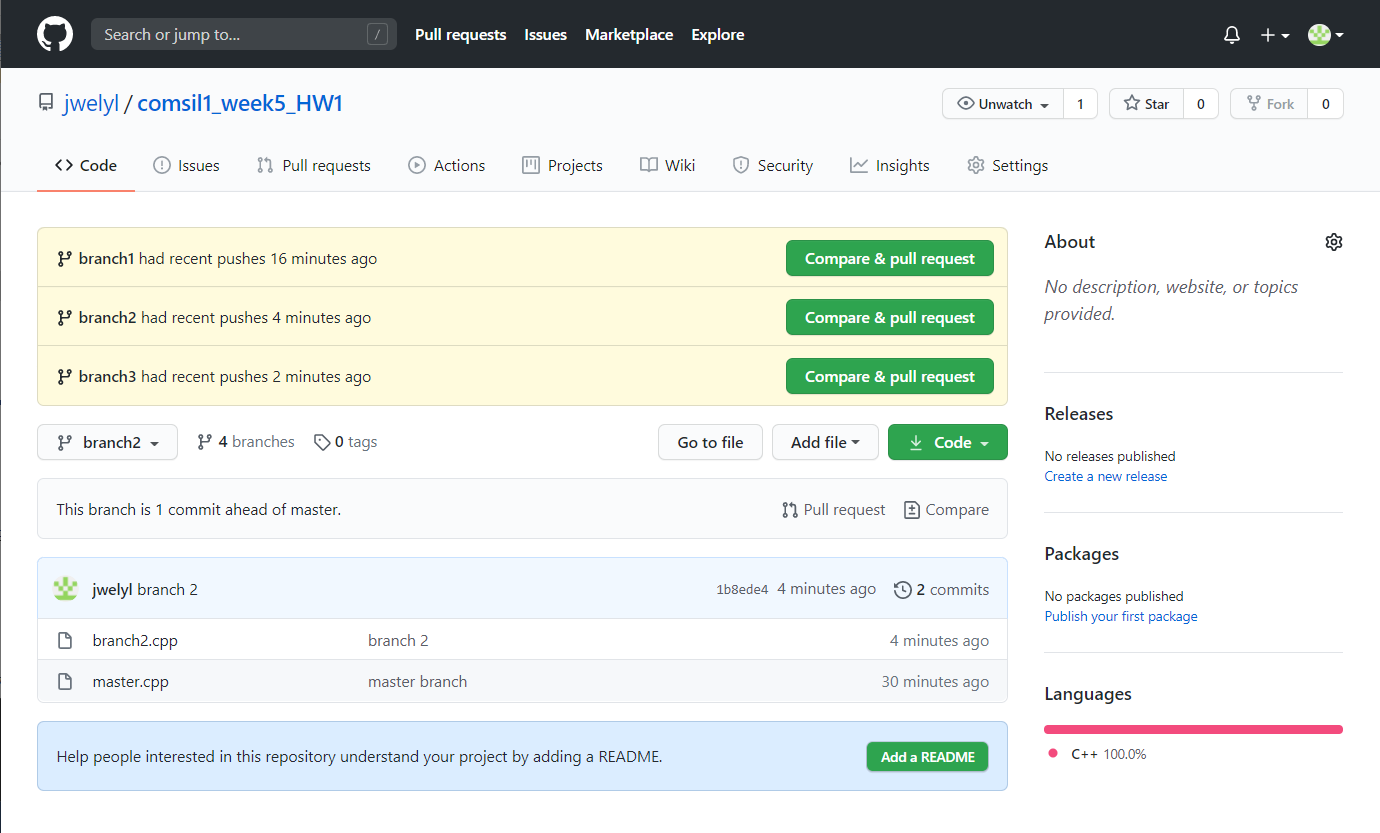


로컬 저장소에서 branch1에 만들어 줬던 branch1.cpp 파일이 사라진 것을 알 수 있다. branch1.cpp 파일은 branch1 branch에 생성했기 때문에 master branch로 돌아오면 사라진다.

branch1을 만든 방법과 같이 master branch에 branch2, branch3 branch를 추가하고 각각 branch2.cpp, branch3.cpp 파일을 추가하여 commit과 push를 한다.

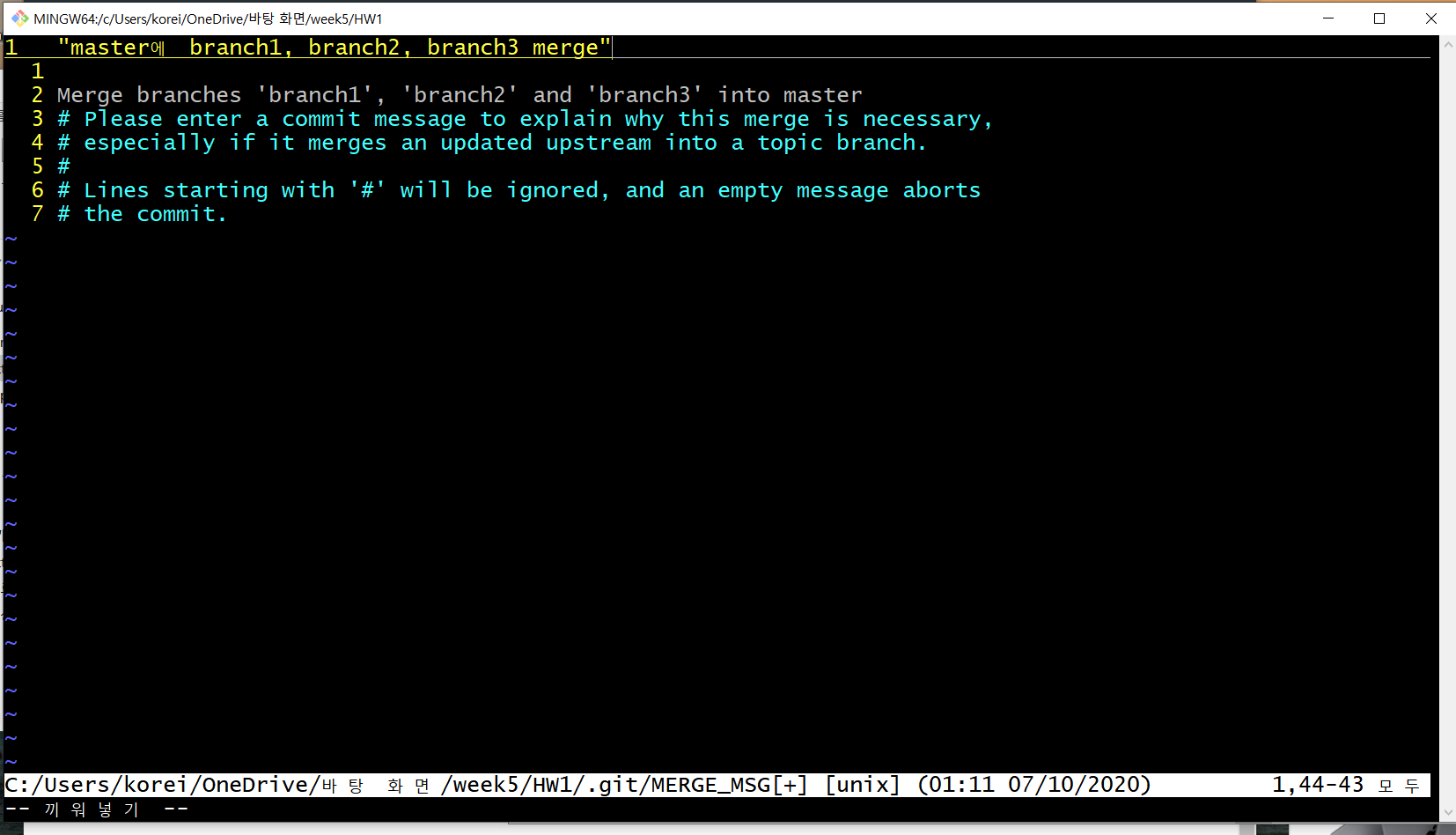


git branch 명령어로 master, branch1, branch2, branch3 branch를 확인한다.



원격 저장소에 branch2, branch3가 저장된 것을 확인할 수 있다.

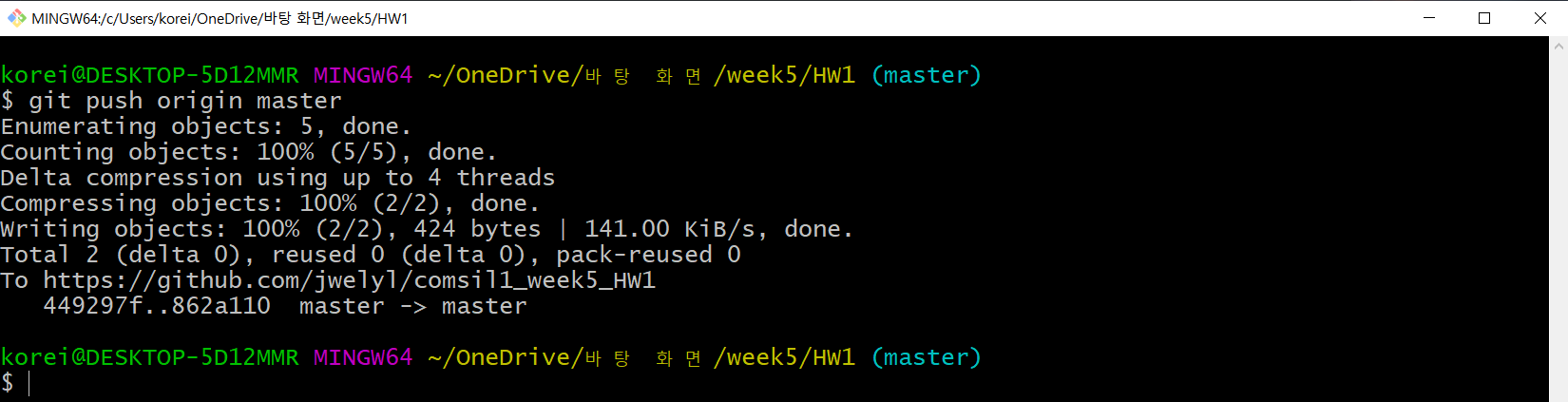
Master branch에 branch1, 2, 3을 merge하기 위해 master branch로 이동 후, git merge branch1 branch2 branch3 명령어를 입력한다.

.

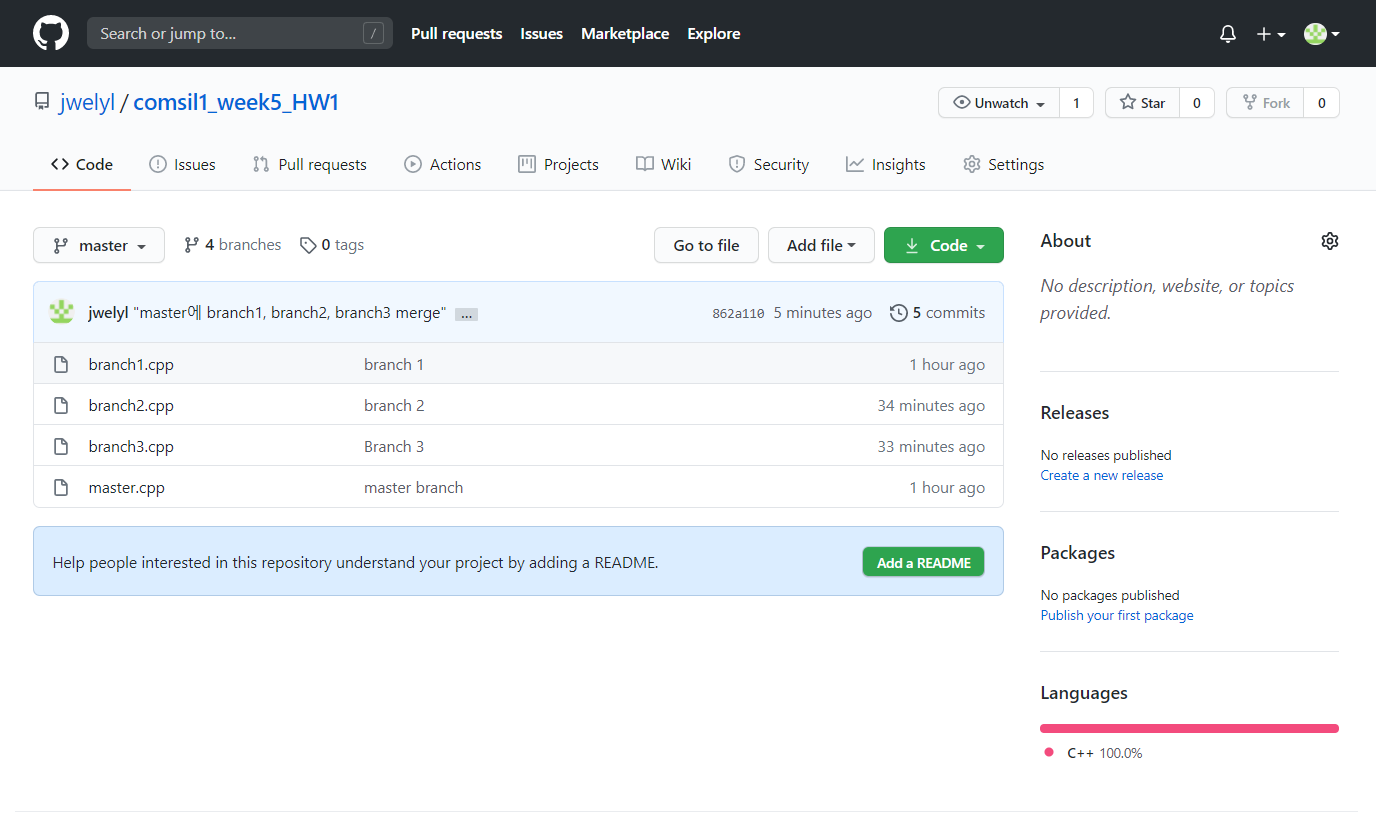
Vim 창이 뜨면 메시지를 입력 후 저장한다.



merge 후 ls 명령어로 로컬 저장소의 파일을 확인해 보면 master branch에 branch1, 2, 3에 있던 branch1.cpp, branch2.cpp, branch3.cpp가 생긴 것을 알 수 있다. 별 다른 충돌 없이 merge가 성공적으로 수행된 것을 알 수 있다.

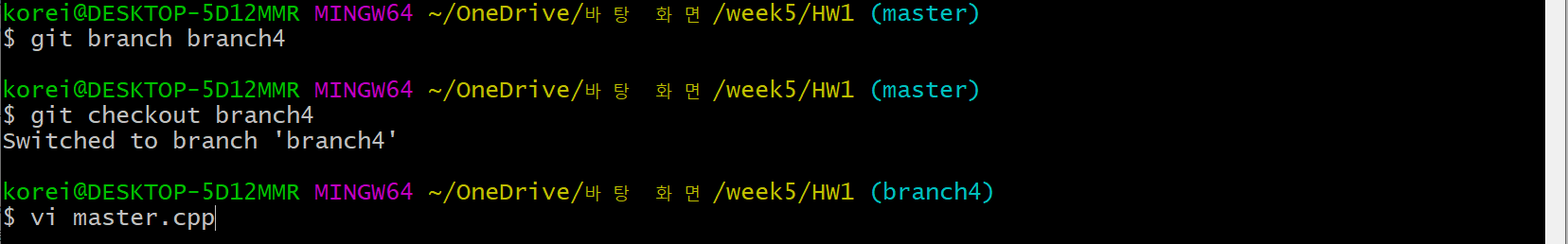


원격 저장소에 push 후,

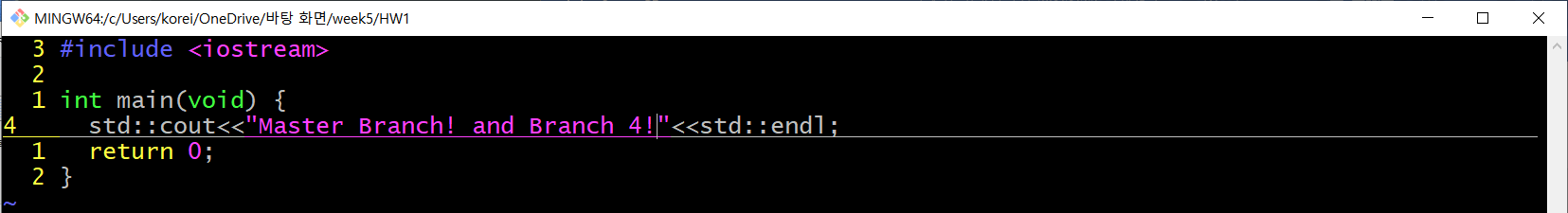


원격 저장소의 master branch를 확인하면 merge된 것을 알 수 있다.

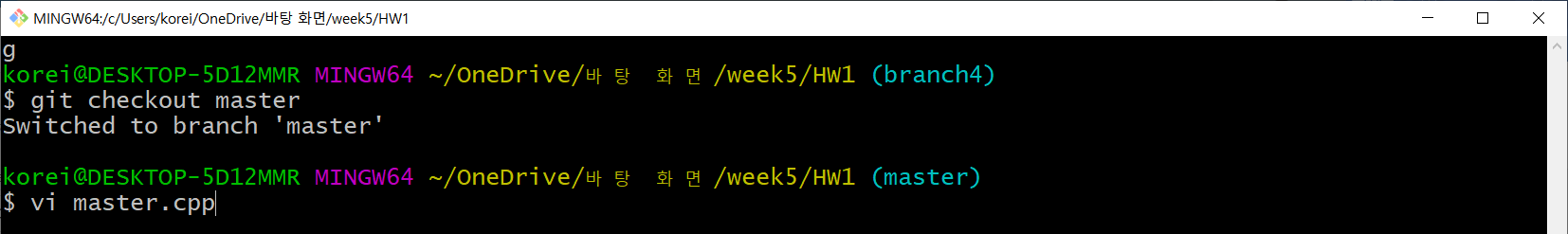
Branch를 merge할 때 충돌이 발생하도록 새로운 branch4를 만들고 merge해 보겠다.



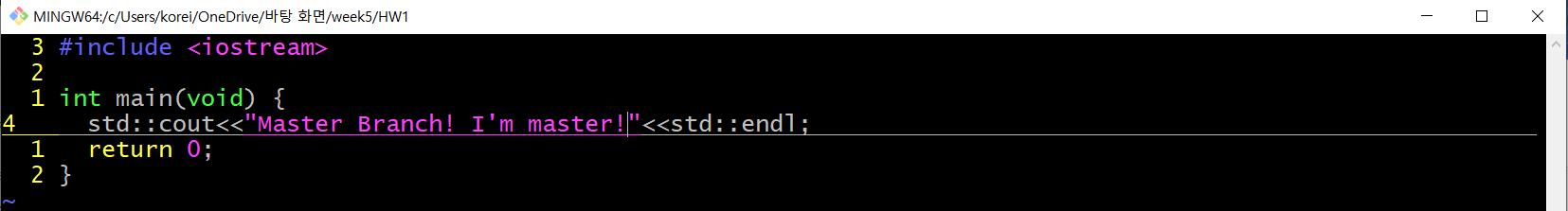
branch4를 만들고 이동하여 master.cpp 파일을 실행한다.



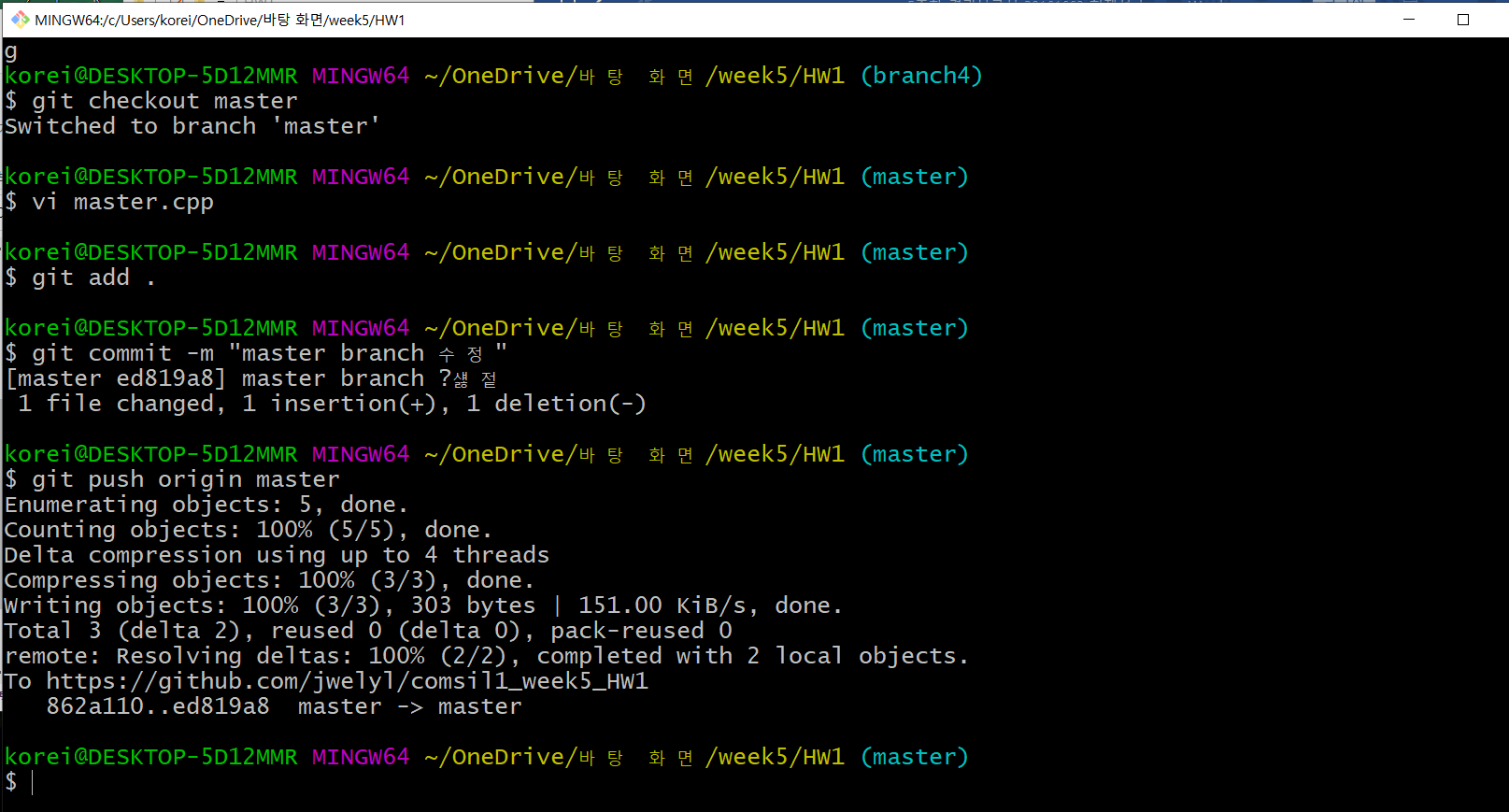
4번째 줄을 수정 후 저장하고 add, commit, push한다.



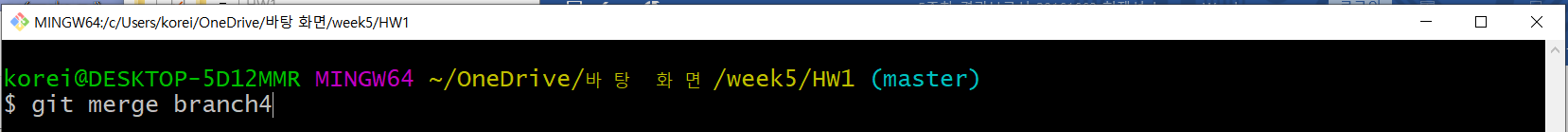
master branch로 돌아가서 master.cpp를 실행하고



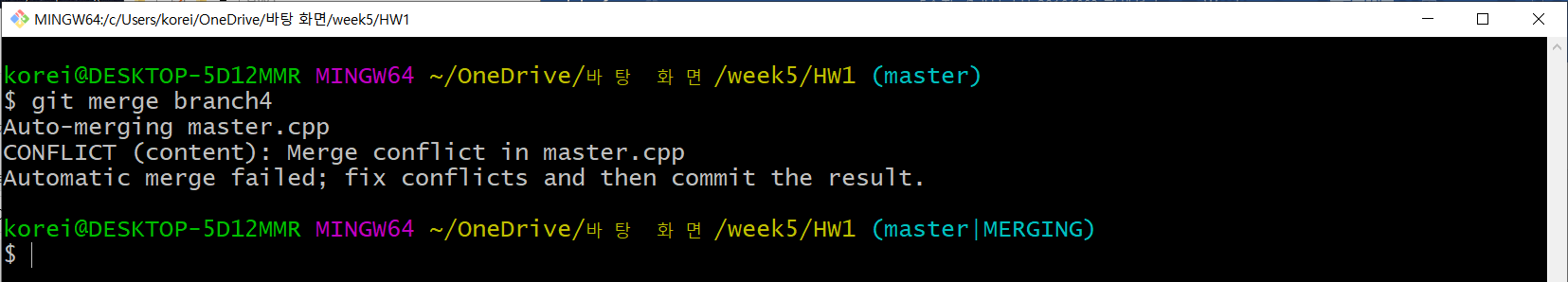
branch4 때와 같이 4번째 줄을 수정하여 충돌이 일어나도록 하고 저장 후 push한다.



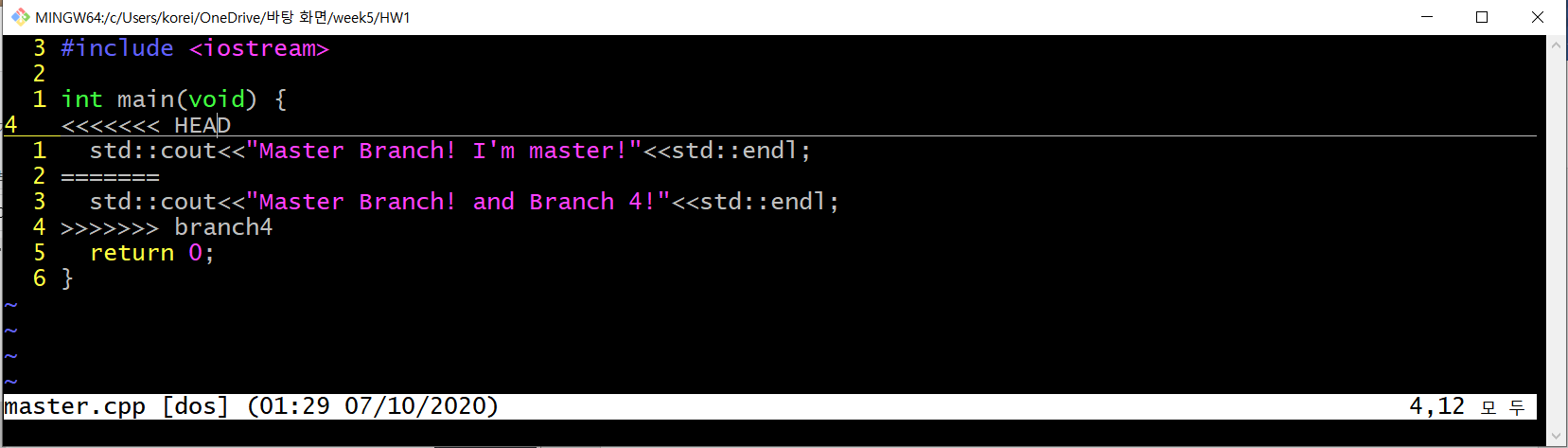
master branch에서 branch4와 merge를 하면



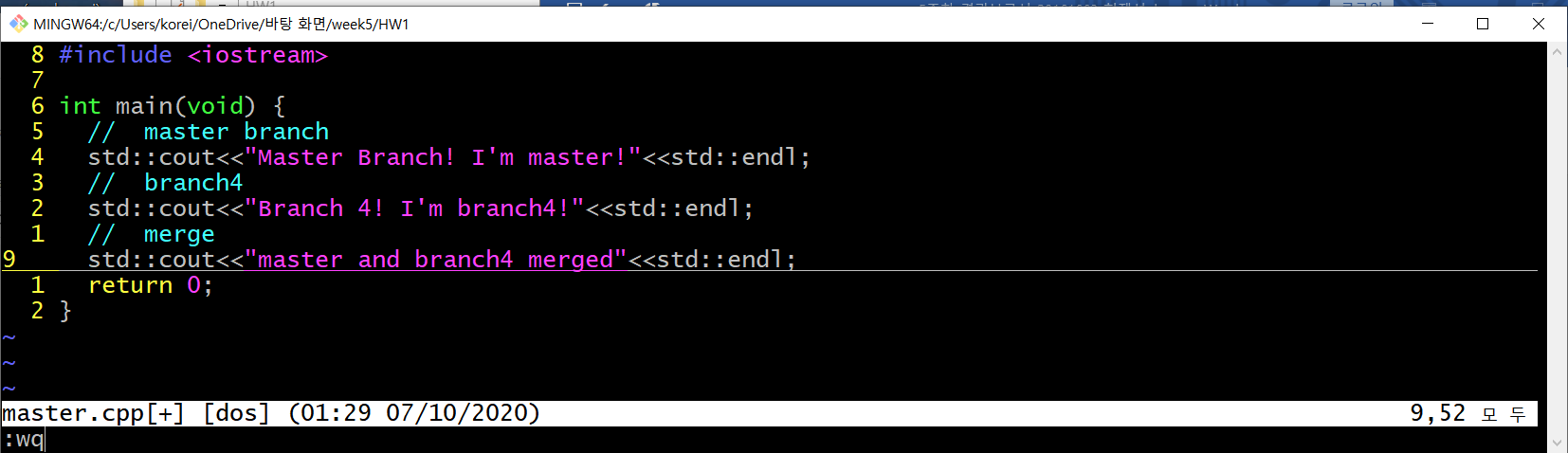
다음과 같이 master.cpp 파일에서 충돌이 발생해서 자동적으로 merge하는 것이 실패했다는 경고 메시지가 출력된다.



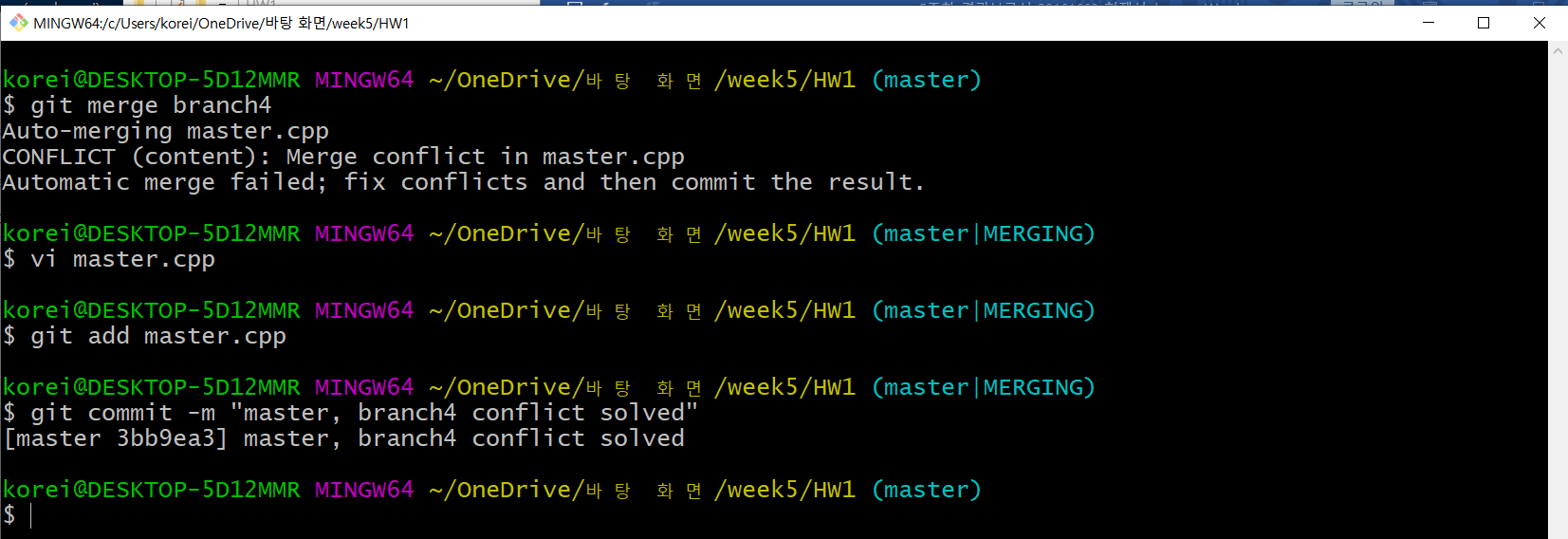
master.cpp 파일을 열어보면 아래와 같이 충돌 내용을 확인할 수 있다.



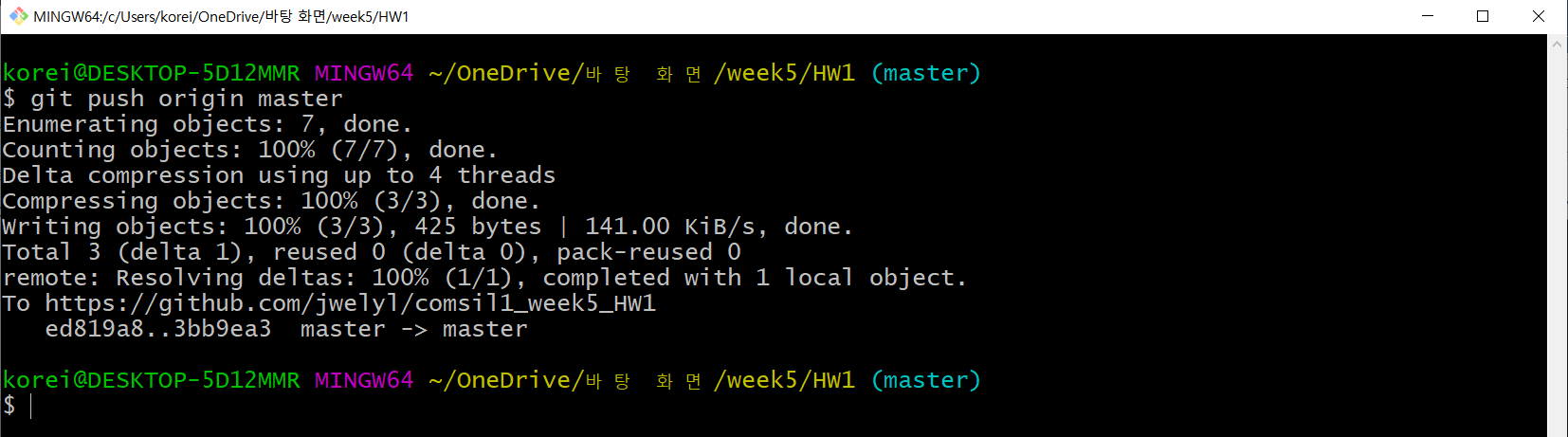
충돌로 자동 병합이 안되므로 원하는 대로 내용을 수정하고 저장한다.



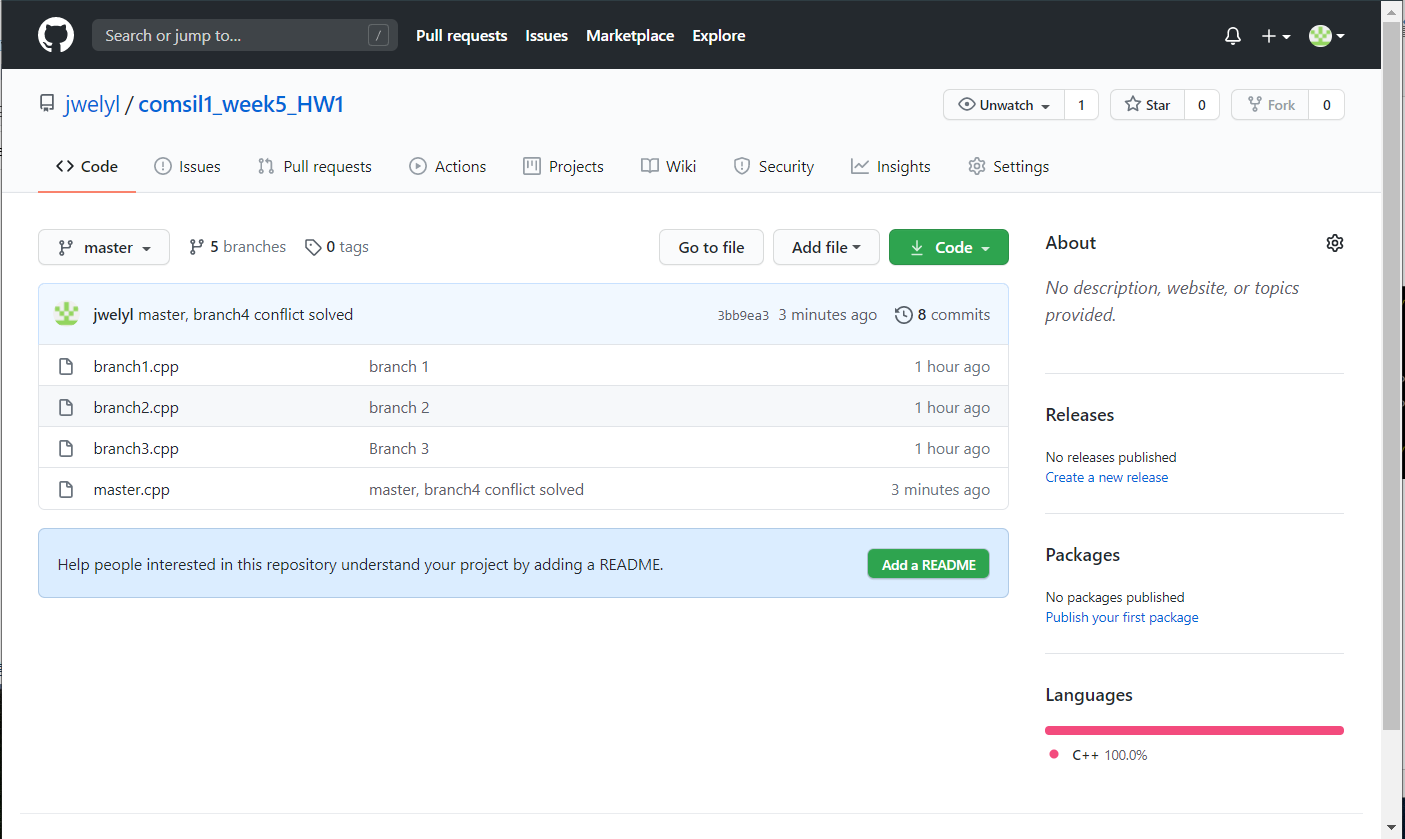
충돌이 해결되지 않았을 때에는 branch 이름이 master|MERGING으로 아직 merge가 되지 않았음을 알 수 있다. 파일을 수정 후, commit까지 마치면 충돌이 해결되어 branch 이름이 master로 바뀐다.

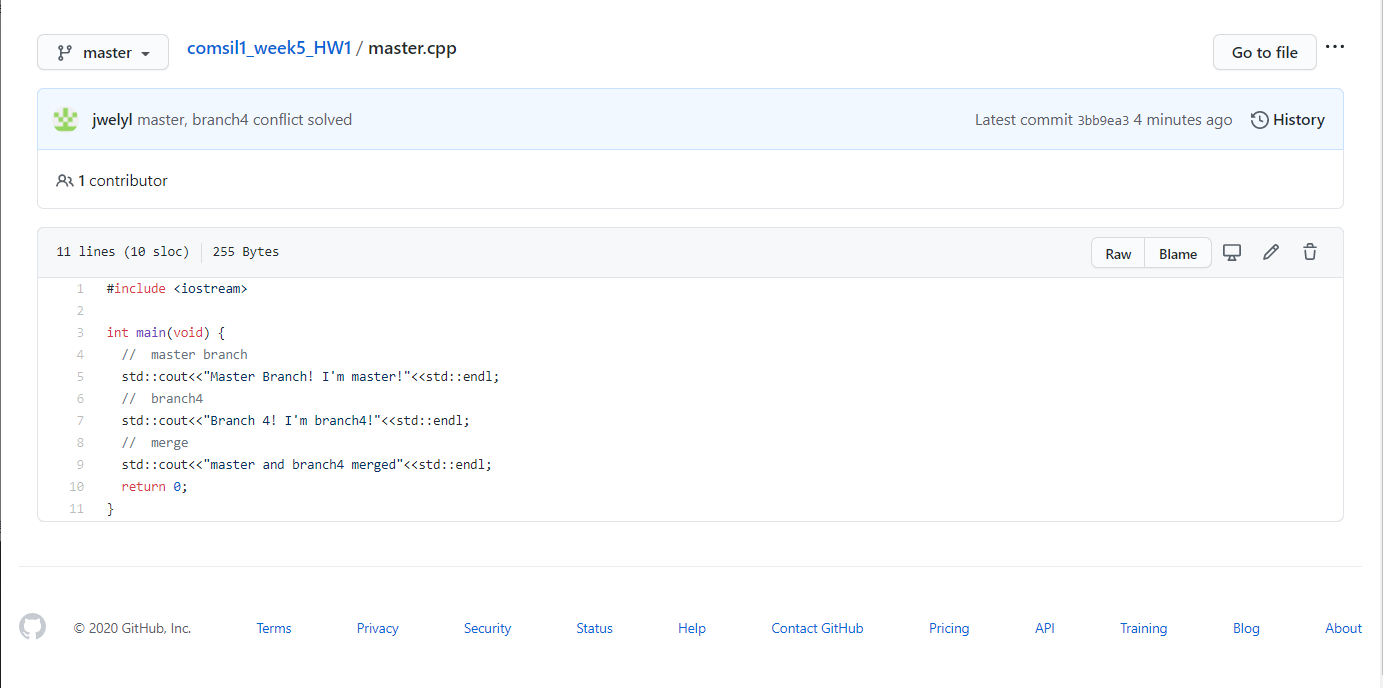


원격 저장소로 master branch를 push한다.



원격 저장소를 확인하면 branch4가 master에 merge된 것을 알 수 있다.



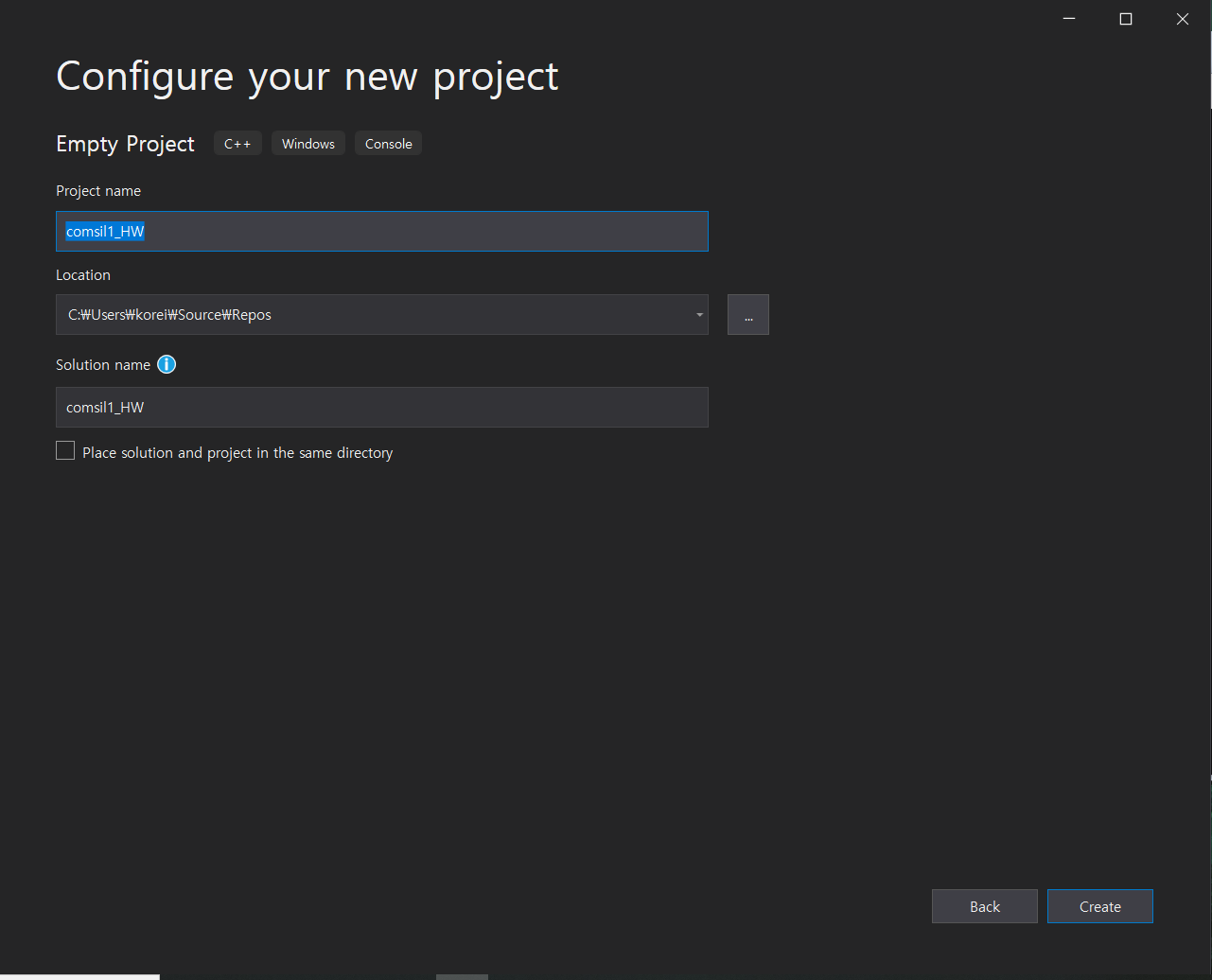


해당 원격 저장소의 주소는 다음과 같다.

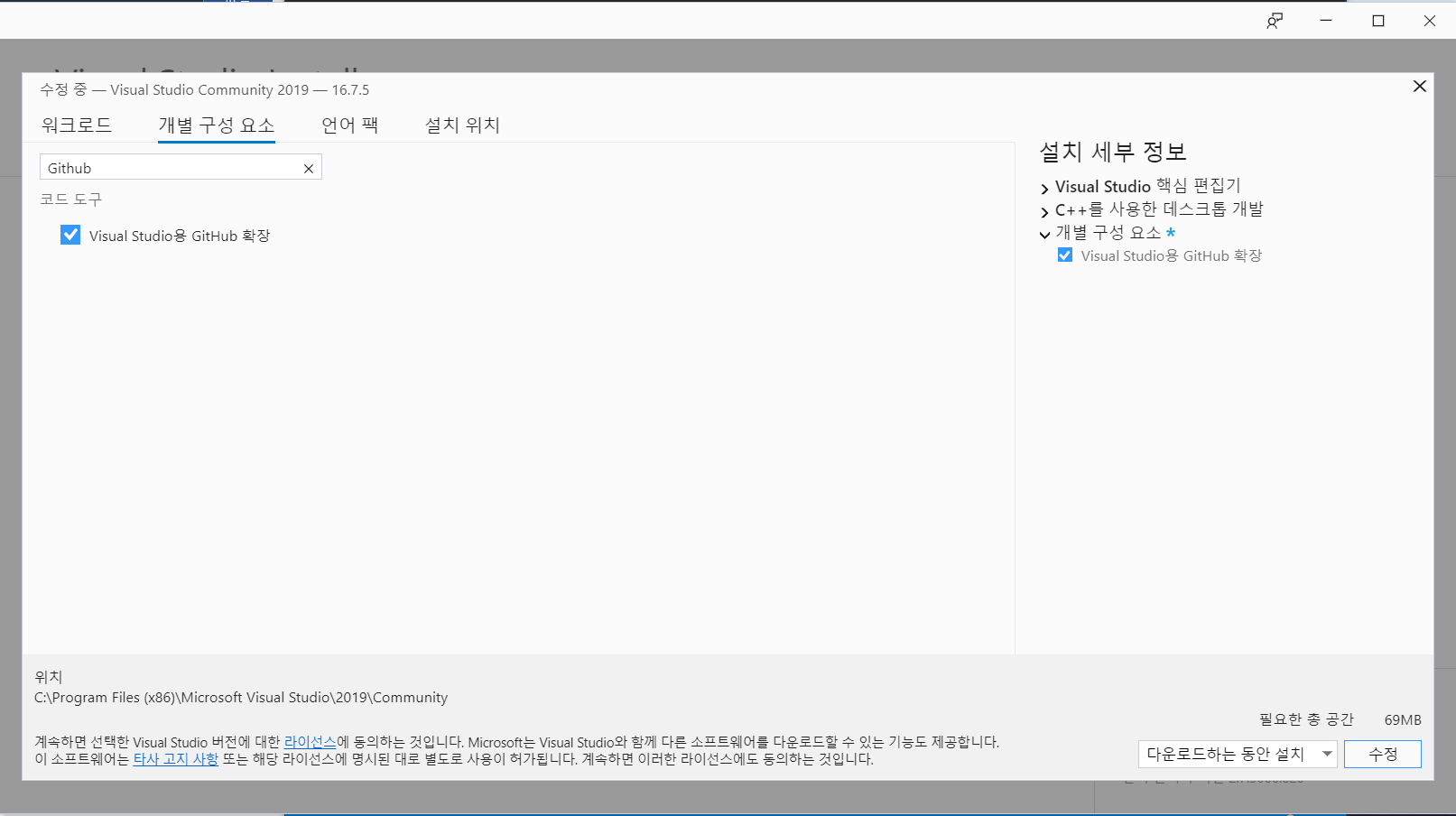
<https://github.com/jwelyl/comsil1_week5_HW1>

두 번째 과제로 Visual Studio에서 Github를 사용해본다. Visual Studio 2019에서 Github Extension for Visual Studio를 추가하고 Github 계정을 연결 후, 임의의 프로젝트를 만들어서 원격 저장소에 추가한다.

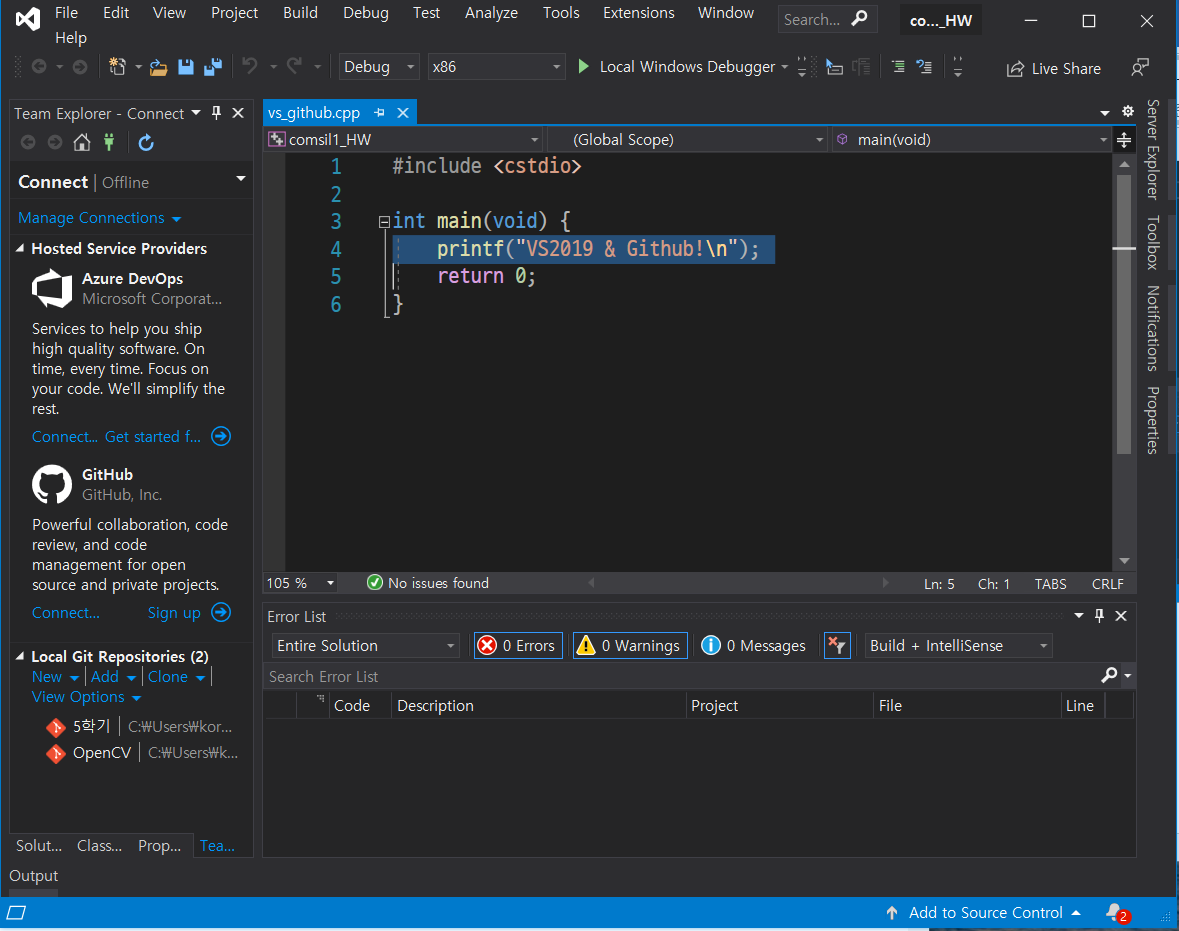
VS2019를 실행해 새로운 프로젝트를 만든다.



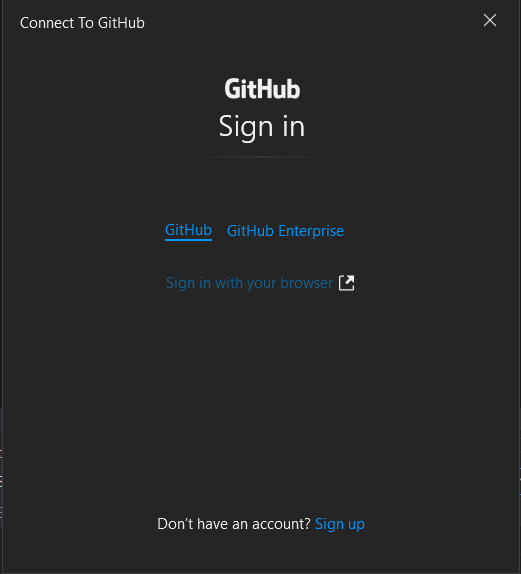
도구 – 도구 및 가져오기 -개별 구성 요소 로 이동하여 Visual Studio용 Github 확장을 선택 후 설치한다.



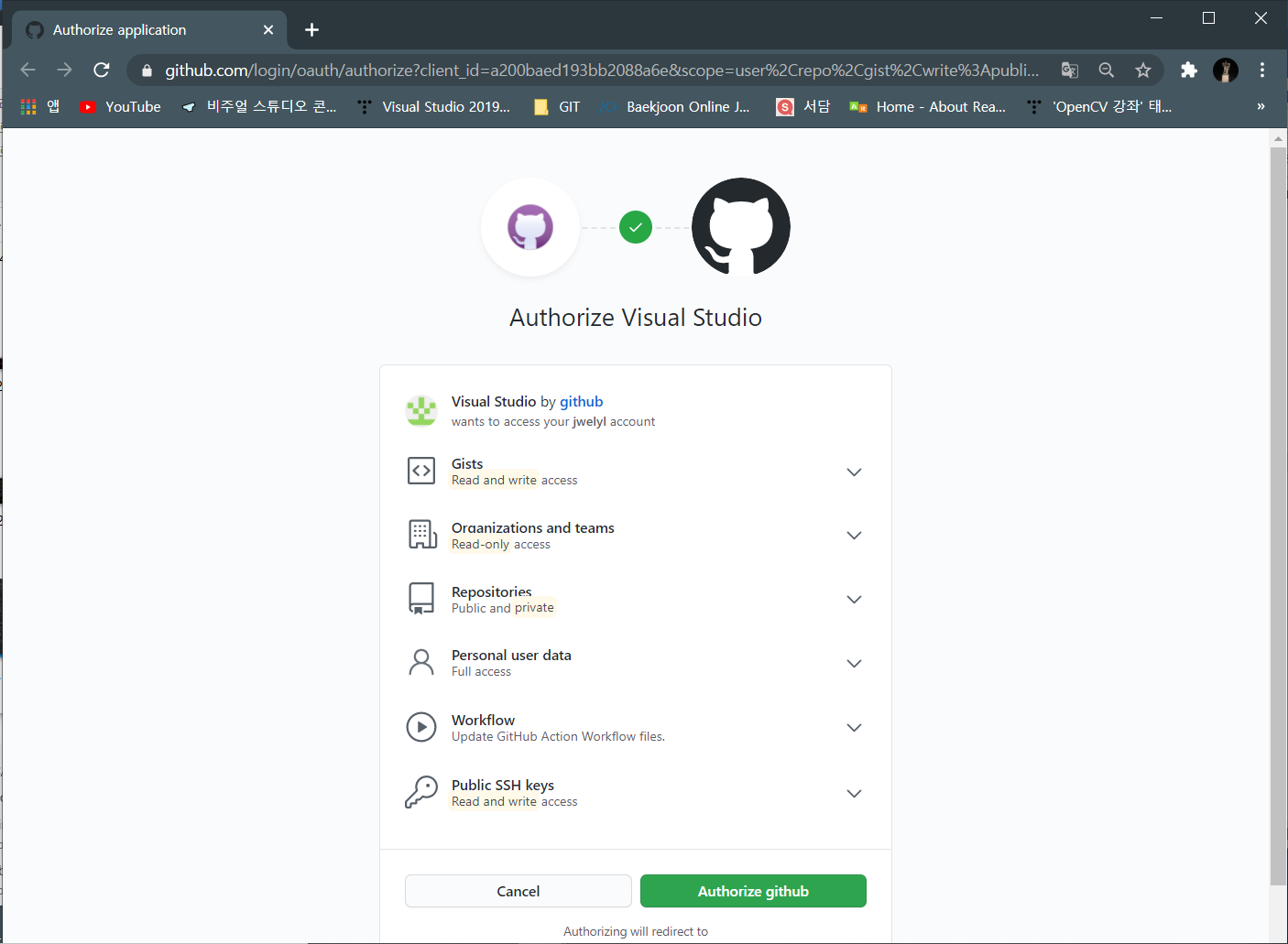
팀 탐색기를 실행하여 GitHub의 Connect를 클릭한다.



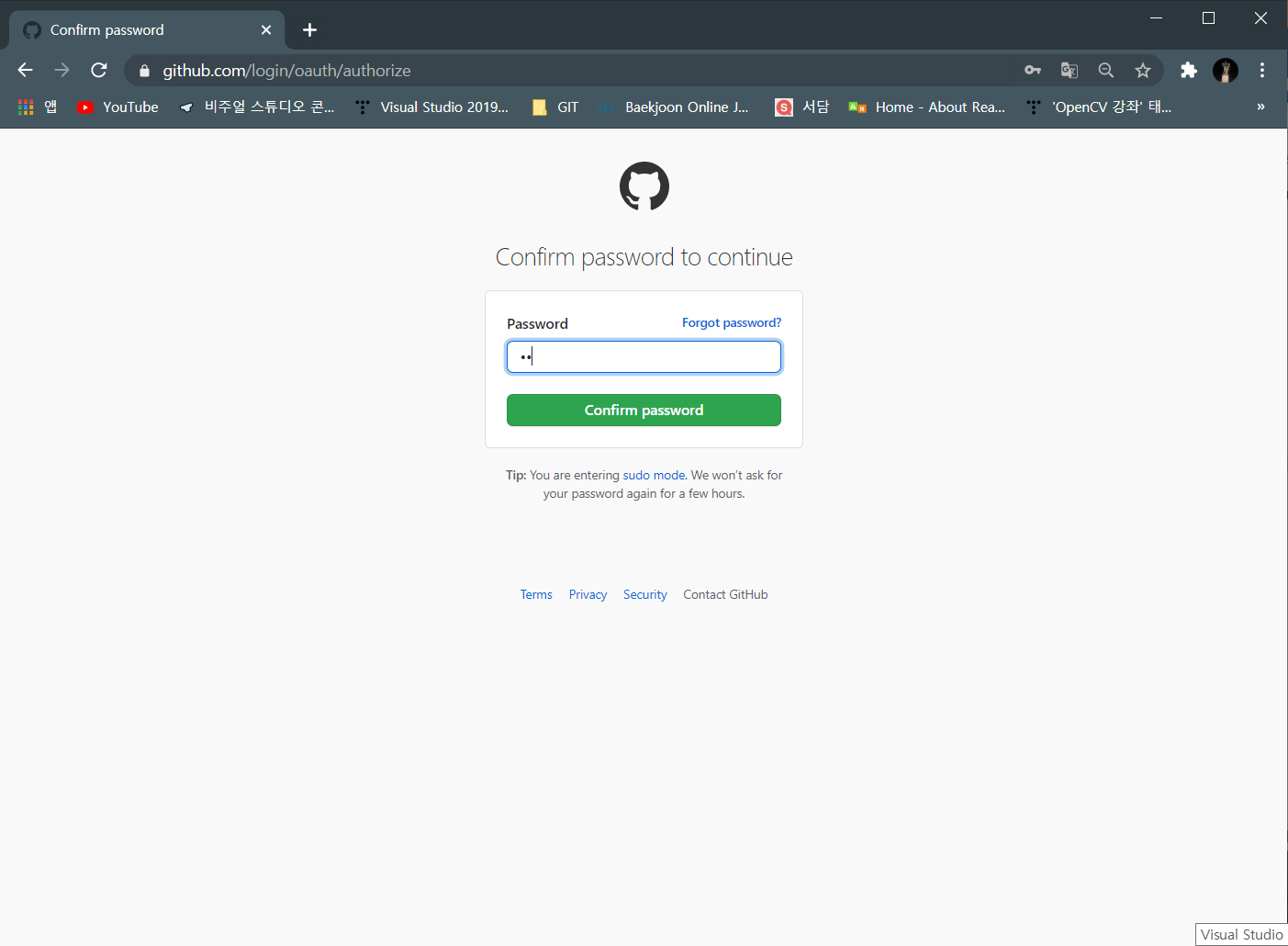
Sign in with your browser를 클릭한다.



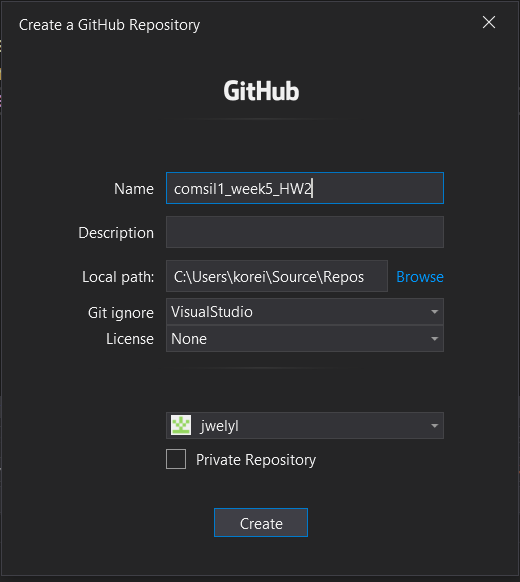
Authorize github를 클릭하고

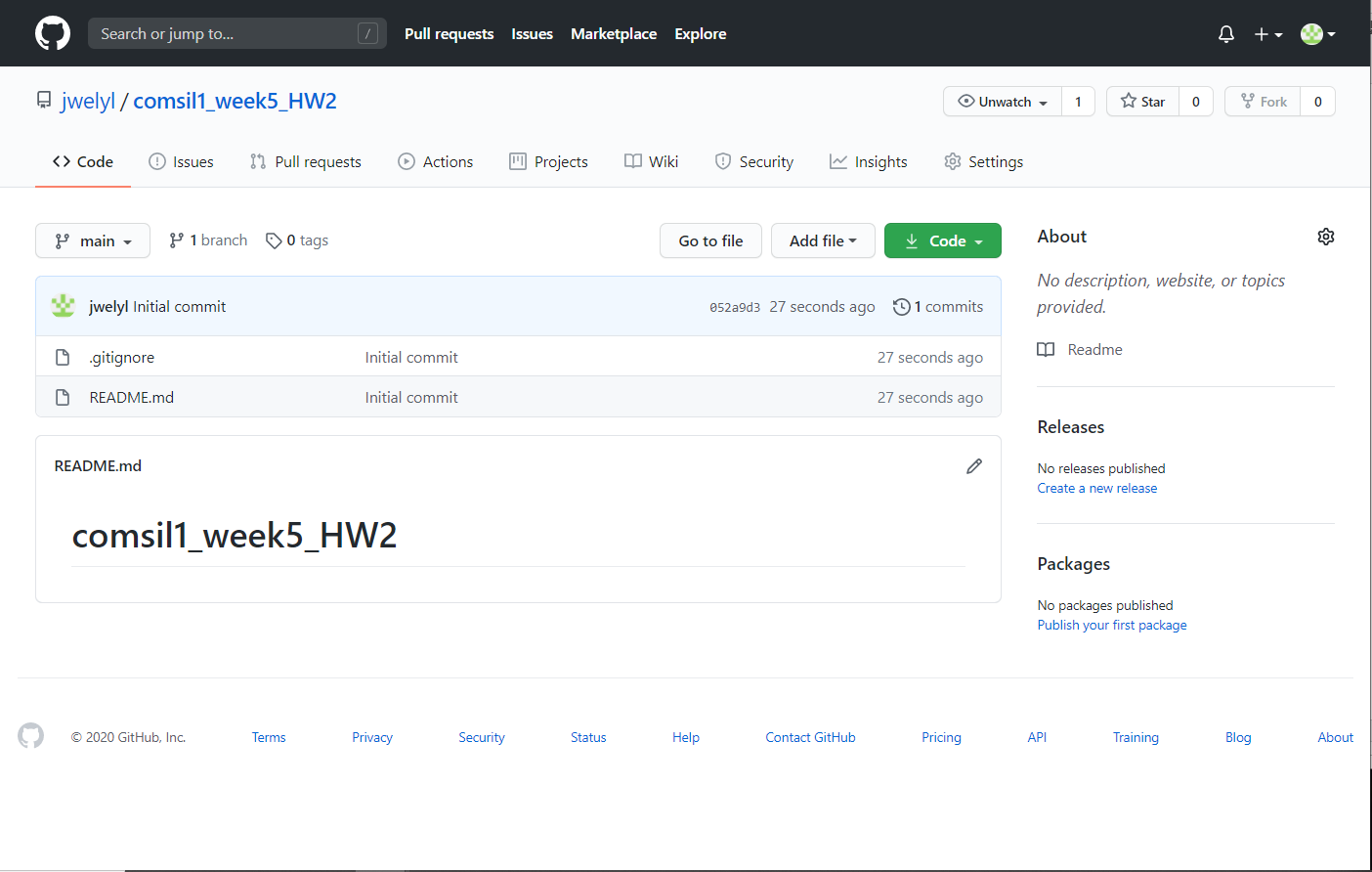


비밀번호를 입력한다.



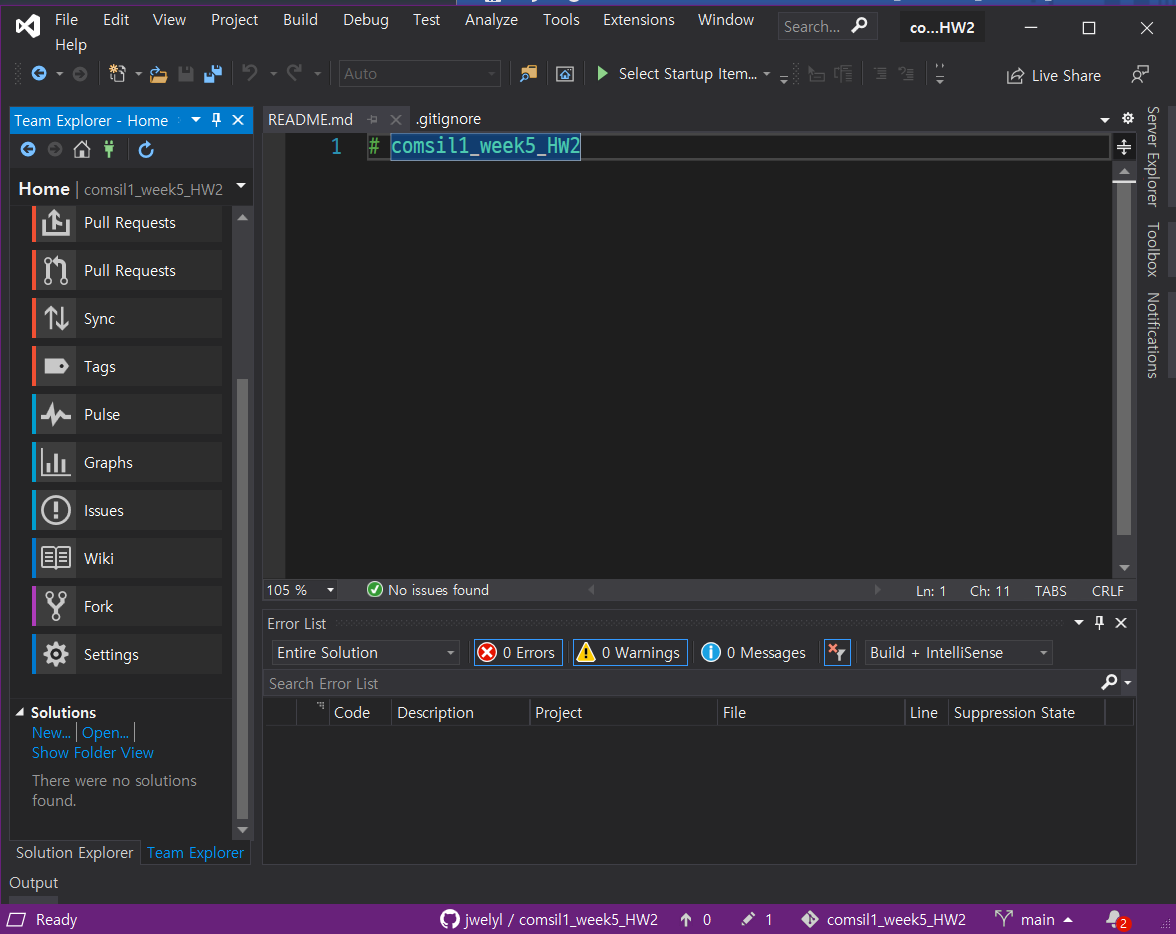
GitHub와 연동하여 Repository를 생성한다.



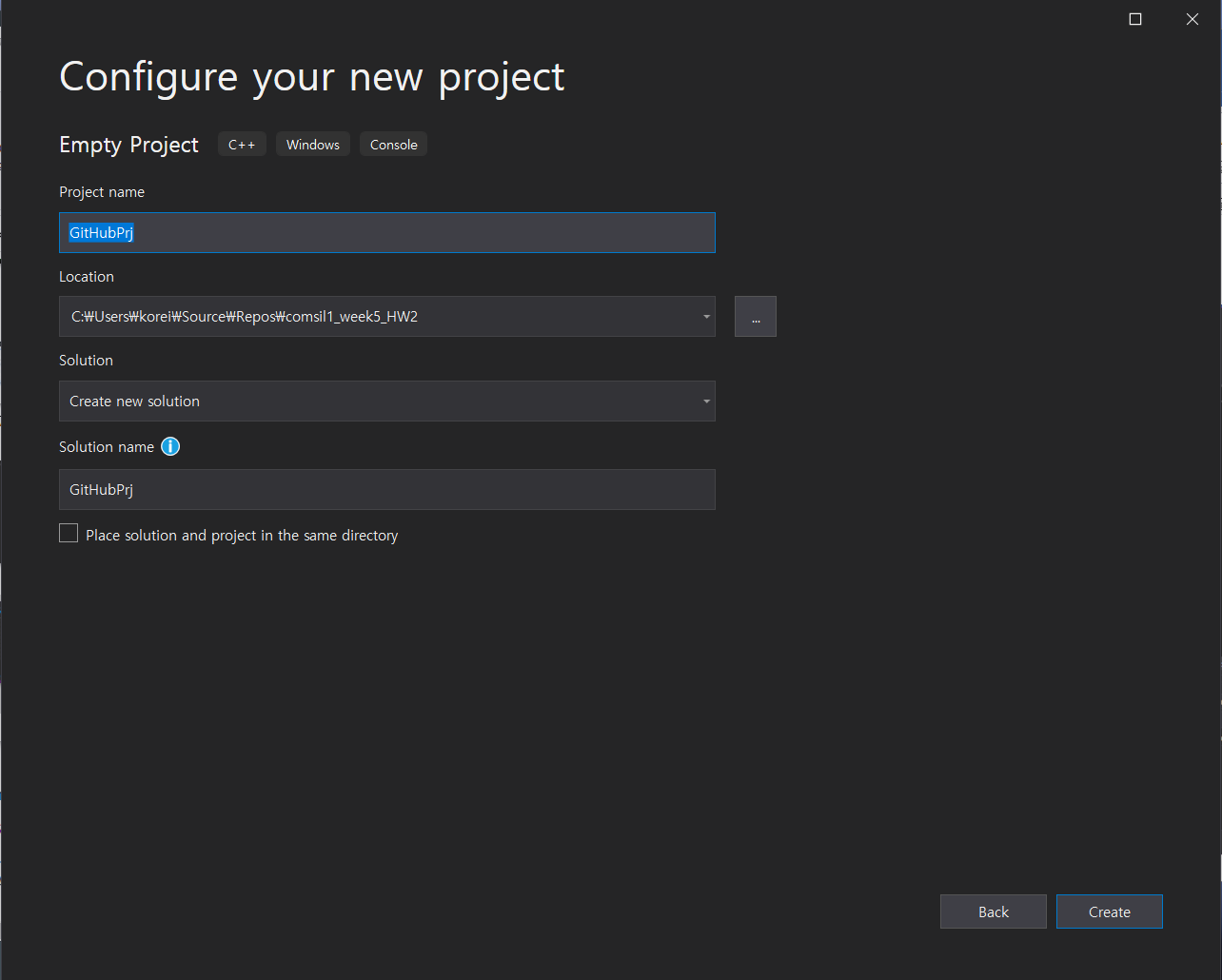


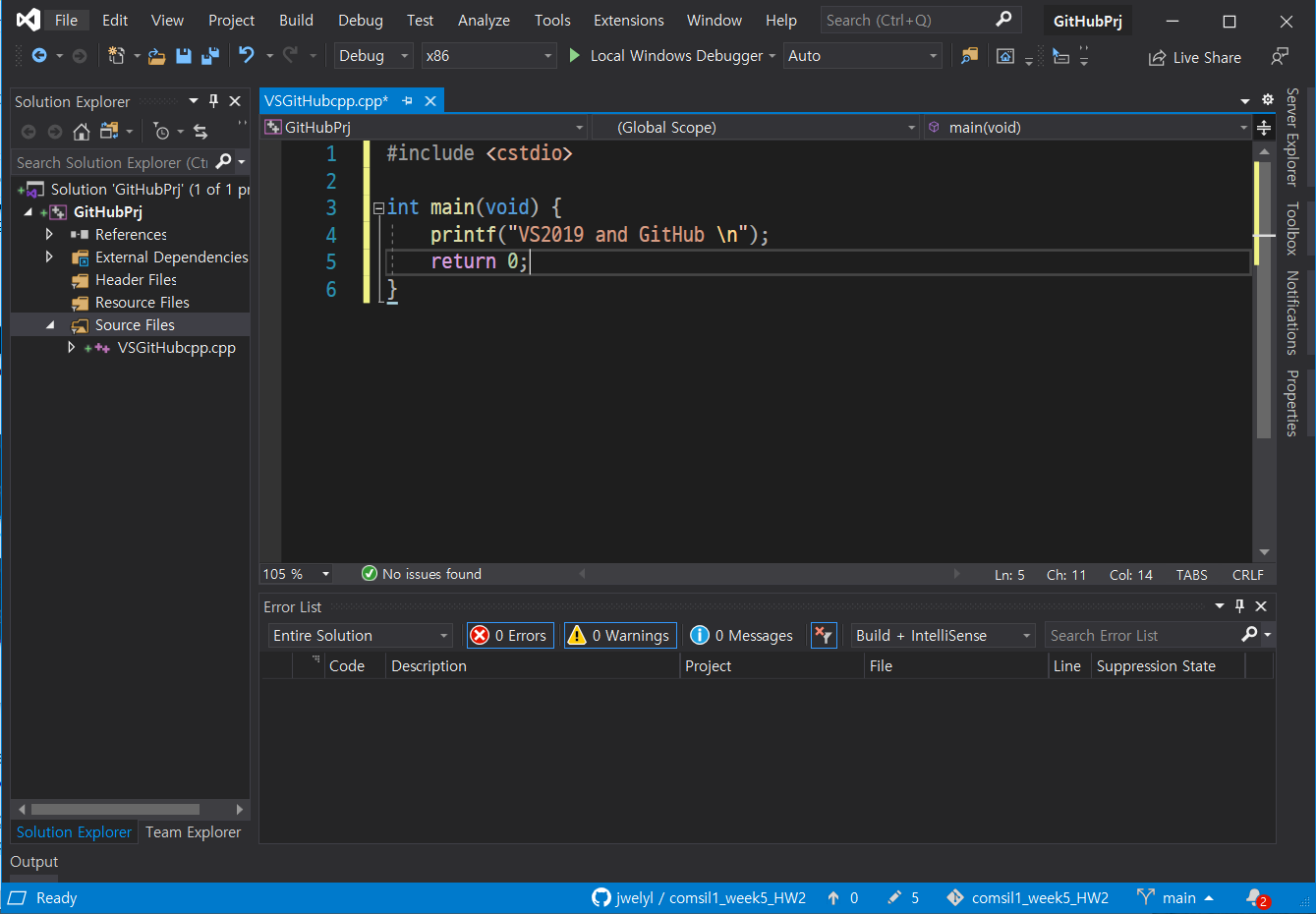
원격 저장소 comsil1\_week5\_HW2가 생성된 것을 확인한다.

왼쪽 아래의 New…를 클릭해 새로운 프로젝트를 만든다.

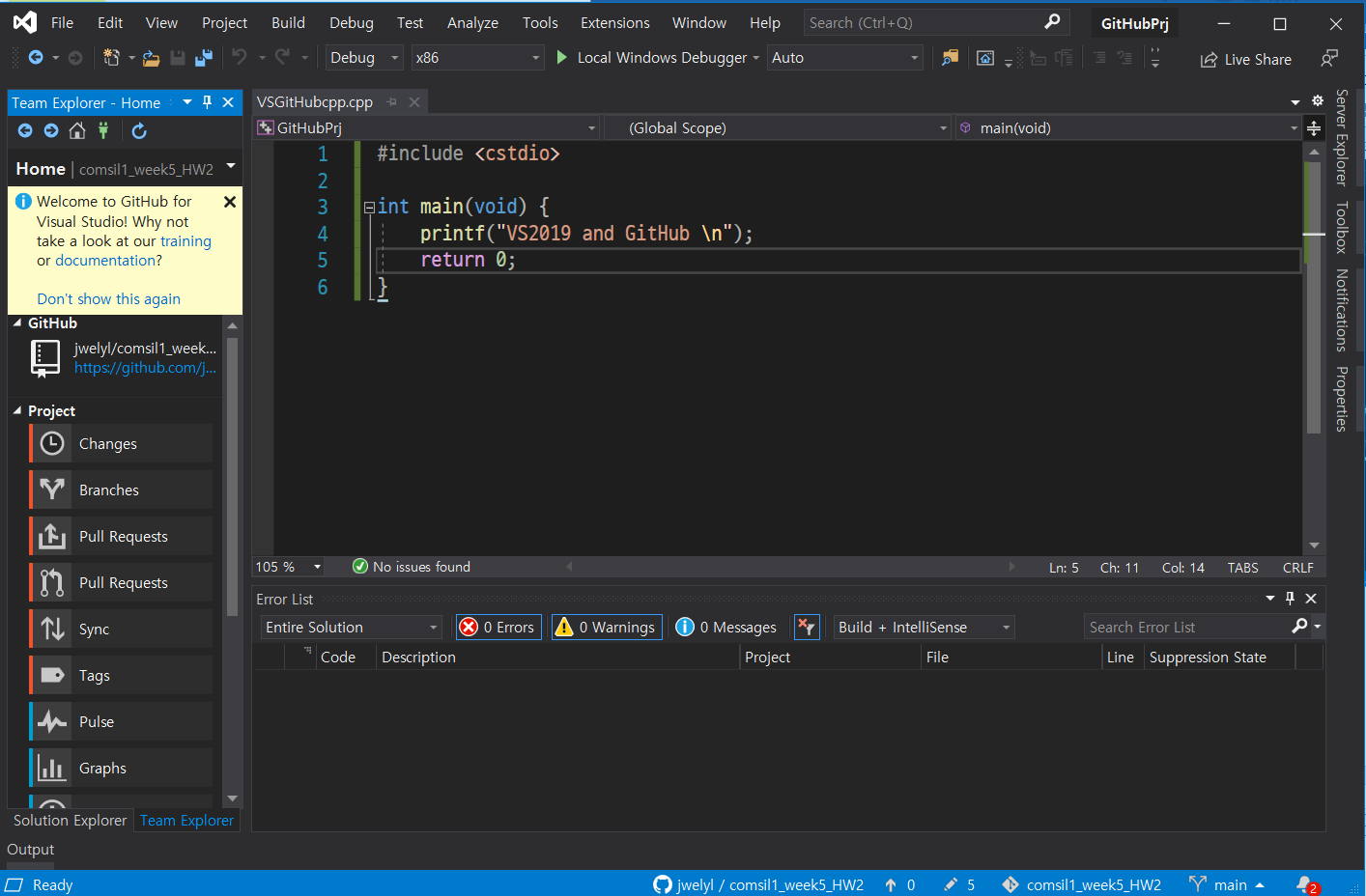


새로운 프로젝트를 만든다.

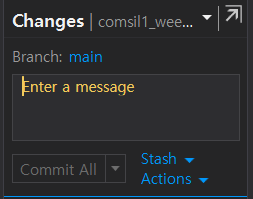
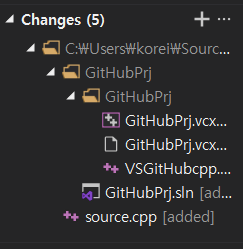


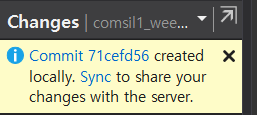


임의의 코드를 작성 후 빌드한다. 소스 파일 옆에 초록색 +는 add 되었음을 의미한다.

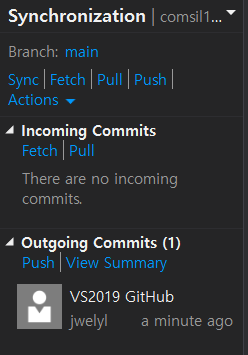


변경 내용을 확인 후 commit 하기 위해 팀 탐색기의 Changes를 클릭한다.

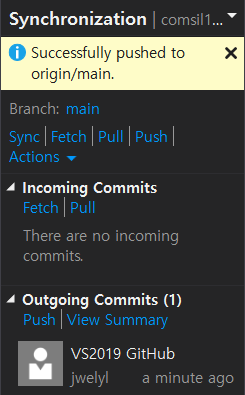


변경 내용을 확인 후, Commit 메시지를 입력 후 commit all을 클릭한다. 

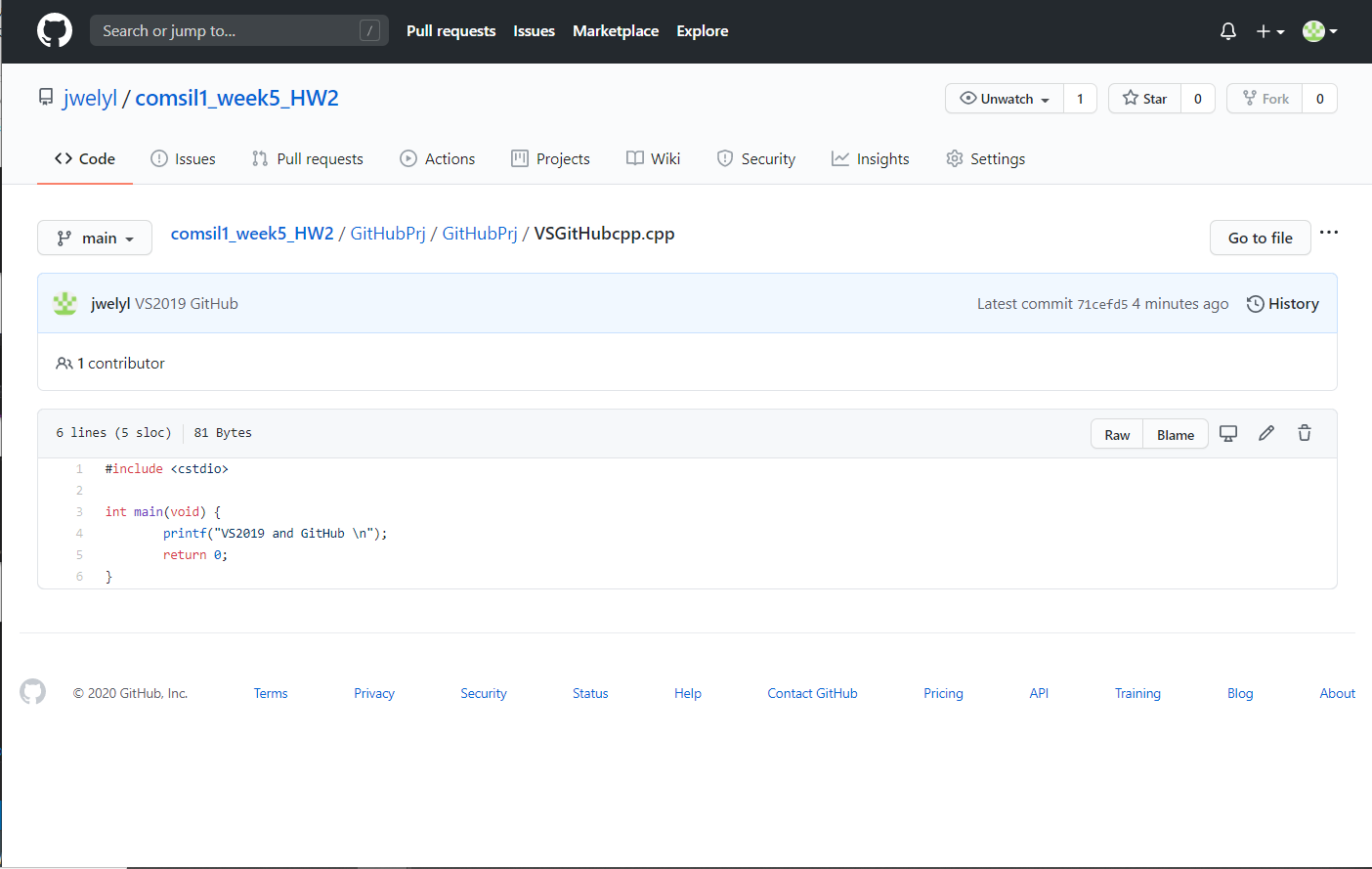
Commit이 완료되었다. 원격 저장소로 push하기 위해 Sync(동기화)를 클릭한다.

.

push 버튼을 클릭해 push한다. 아래에 commit 메시지를 확인할 수 있다.



Push가 완료되었다.



원격 저장소에 push된 것을 확인한다.

사용한 명령어

git branch : 현재 만들어진 branch 목록을 본다. 따로 branch를 만들지 않았다면 master branch만 있으므로 master만 출력된다.

git branch branchname : branchname이라는 이름의 branch를 만든다.

git checkout branchname : branchname branch로 이동한다. 이동함에 따라 로컬 저장소의 파일, 디렉토리도 해당 branch의 내용으로 변경된다.

git branch -d branchname : branchname branch를 제거한다.

git merge b1 b2 … bn : 현재 위치한 branch에 b1, b2, … bn branch를 병합한다. 위의 과제 1에서git checkout master 명령어로 master branch로 이동하고 git merge branch1 branch2 branch3 명령어를 입력하면 master branch에 branch1, branch2, branch3이 병합된다. 충돌이 발생할 경우 자동으로 merge되지 않으므로 충돌이 발생한 부분을 수동으로 처리해줘야 한다.

6. 결론

Branch merge 원격 저장소 주소

<https://github.com/jwelyl/comsil1_week5_HW>1

VS2019 프로젝트 원격 저장소 주소

<https://github.com/jwelyl/comsil1_week5_HW2>

버전 관리를 위해서 Branch와 Merge를 이용할 수 있다. 여러 경우의 Branch를 나누어 각각의 Branch에 구현하고 Branch들을 merge하여 하나의 프로그램을 만들 수 있다. 이 과정에서 충돌이 생길 수 있는데 충돌이 생길 경우 제작하려는 프로그램의 의도에 맞게 소스를 수정한다. Git은 push로 대표되는 원격 저장소로의 백업 기능을 제공하고 commit으로 버전 관리가 가능하다. 또한 branch로 나누어 여러 사람들이 협업하기에 용이하며 충돌이 발생했을 경우에도 의도에 맞게 유연하게 대처가 가능하다. clone으로 다른 사람들이 소스를 쉽게 가져와서 사용할 수도 있다. Git이 제공하는 이러한 기능은 이전의 비효율적인 백업, 버전 관리, 협업 과정을 한결 더 간편하게 해준다.

Git bash로 CLI 환경으로 git을 다룰 수 있다. VS2019를 사용할 경우 GitHub와 연동하여 GUI 환경에서 Git을 다룰 수도 있다.