

Best Reviewer 양성과정



1. C/C++ 실습환경 구축

Best Reviewer 양성과정

실습 환경 요약



형상관리

Git

C++

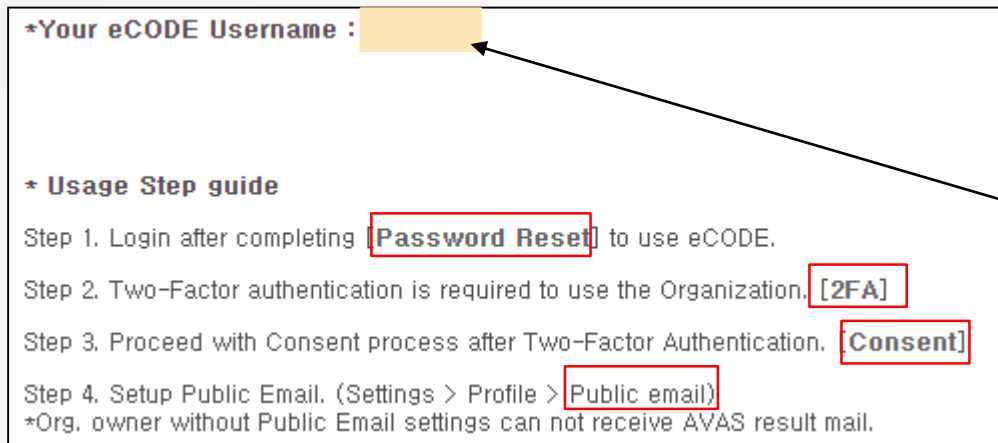
cygwin (compile, make, cmake)
Visual Code, C/C++ extension

Test 관련

gTest/gMock

GitHub(eCODE) 계정 활성화

- eCODE 초대 메일([Notice] Welcome to eCODE) 확인



eCODE 로그인 username
eCODE 초대메일상의 사용자명 변경 불가.
자세한 내용은 메일 참조.

- 이미 eCode 가입된 경우 메일 전송되지 않음.
: 계정 활성화, Token 생성, ORG 확인 절차들
체크 필요

- GitHub(eCode) 계정 활성화

- 메일상의 step 1 ~ step 4 수행

https://github.ecodesamsung.com/pages/Service/guide/page/en/user_process/1_user_config

https://github.ecodesamsung.com/pages/Service/guide/page/en/user_process/2_otp_config

https://github.ecodesamsung.com/pages/Service/guide/page/en/user_process/3_consent

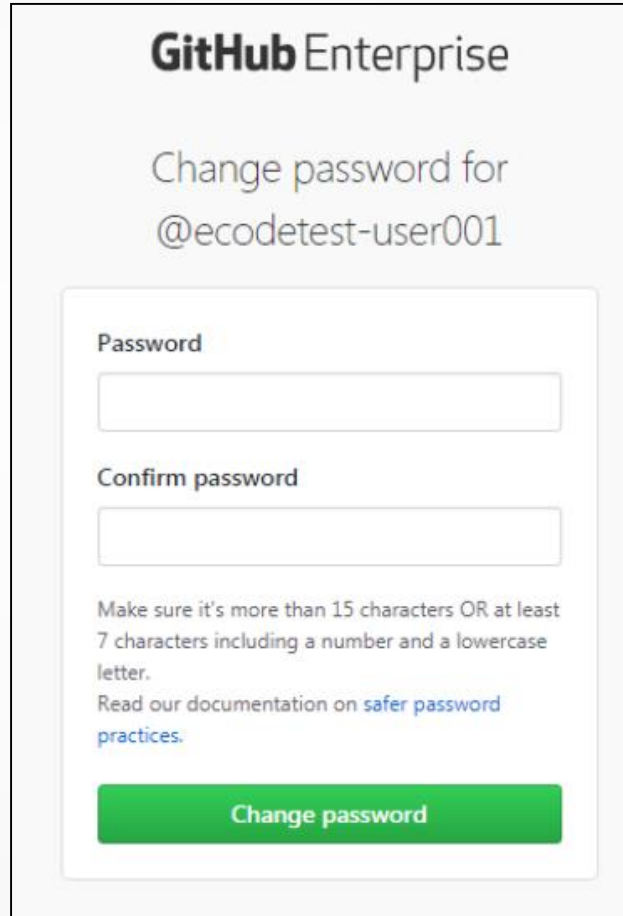
<https://github.ecodesamsung.com/settings/profile> -> Public email 등록

GitHub(eCODE) 계정 활성화 Step 1

- 사용자 Setup

https://github.ecodesamsung.com/pages/Service/guide/page/ko/user_process/1_user_config

email 의 링크 클릭하여 패스워드 설정



The screenshot shows the GitHub Enterprise password change page. At the top, it says "GitHub Enterprise". Below that, it says "Change password for @ecodetest-user001". There are two input fields: "Password" and "Confirm password". Below the input fields, there is a note: "Make sure it's more than 15 characters OR at least 7 characters including a number and a lowercase letter." and a link: "Read our documentation on [safer password practices](#)." At the bottom, there is a green button labeled "Change password".

GitHub(eCODE) 계정 활성화 Step 2

• 2FA 인증 설정

https://github.ecodesamsung.com/pages/Service/guide/page/ko/user_process/2_otp_config

1. [구글 OTP 설치](#)
2. [2FA 인증 활성화](#)
3. [Recovery codes 생성](#)
 - › recovery code 는 스마트폰 분실, 변경시 OTP 복구를 위해 필요.
4. [OTP 앱 등록](#)
 - › [바코드 스캔](#), [직접 입력](#) 중 선택하여 실행

GitHub(eCODE) 계정 활성화 Step 3

• 동의 프로세스

https://github.ecodesamsung.com/pages/Service/guide/page/ko/user_process/3_consent

eCODE 서비스 사용을 위해서

이용약관 , 개인정보처리방침 , 개인정보수집이용 에 대한 동의가 필요합니다.

사용자를 생성하고 **Two-Factor** 인증을 완료한 이후,

동의 페이지 에서 아래 그림에 따라 동의 프로세스를 완료하세요.

동의페이지 클릭하여
동의 프로세스 수행

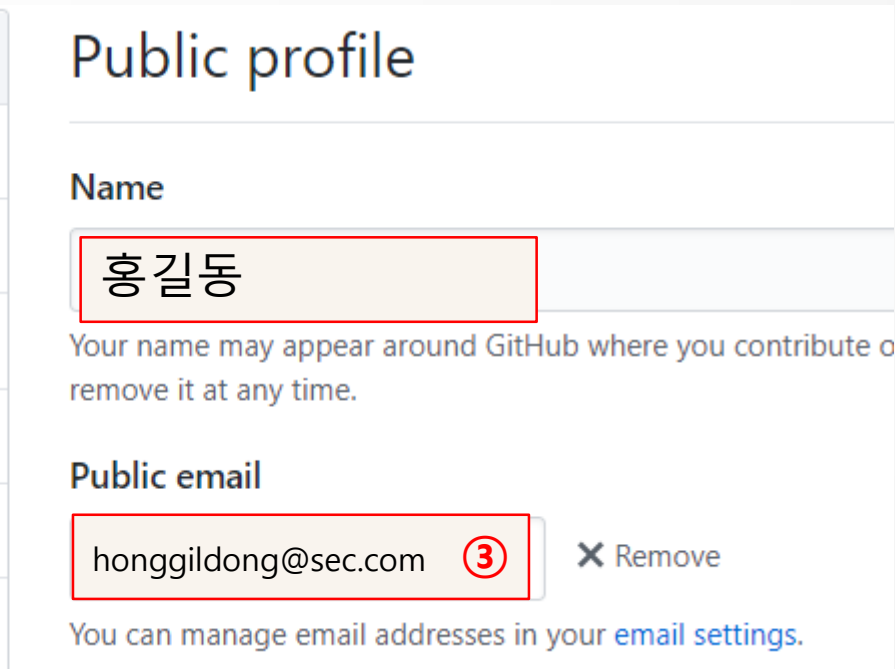
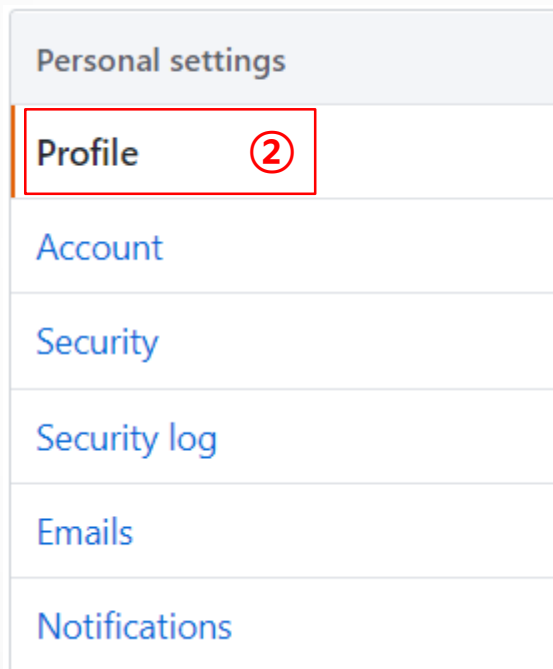
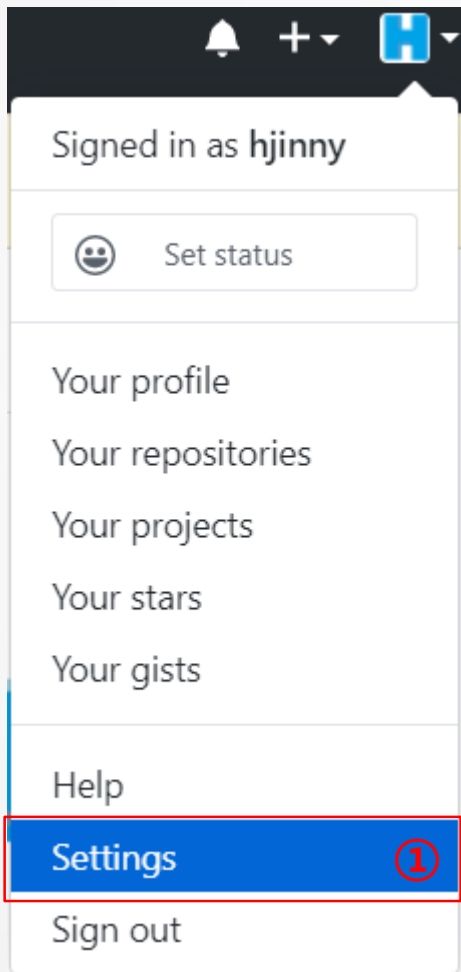
* 주의 : 상단 1~ Complete 모두
클릭하여 동의해야 정상 처리됨.

1. eCODE 로그인 : id, password, OTP
2. Enable GitHub OAuth Authorize
3. Nationality 선택
4. Process Consent as Steps
5. Click Ready button
6. Consent completed

GitHub(eCODE) 계정 활성화 Step 4

- public email 설정

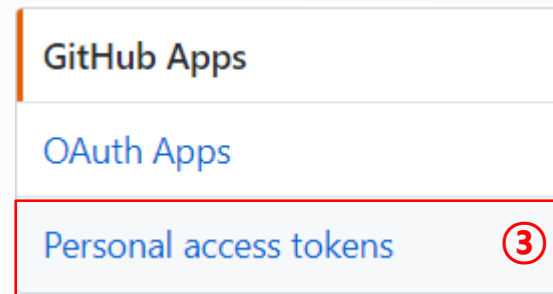
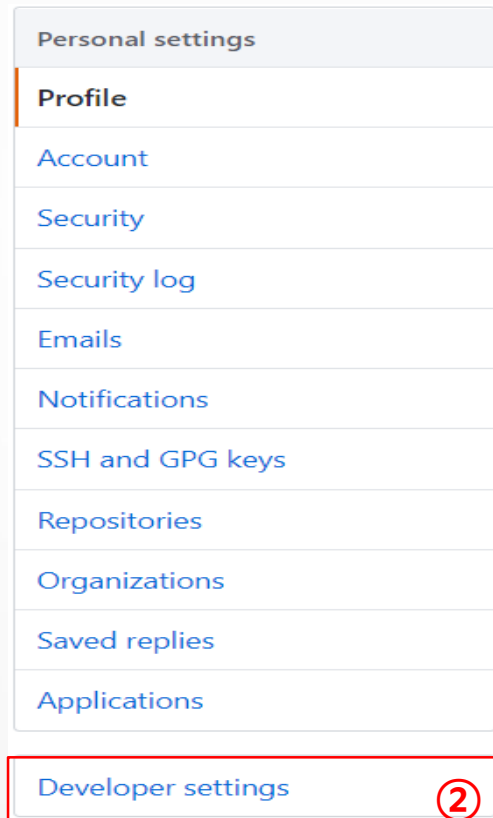
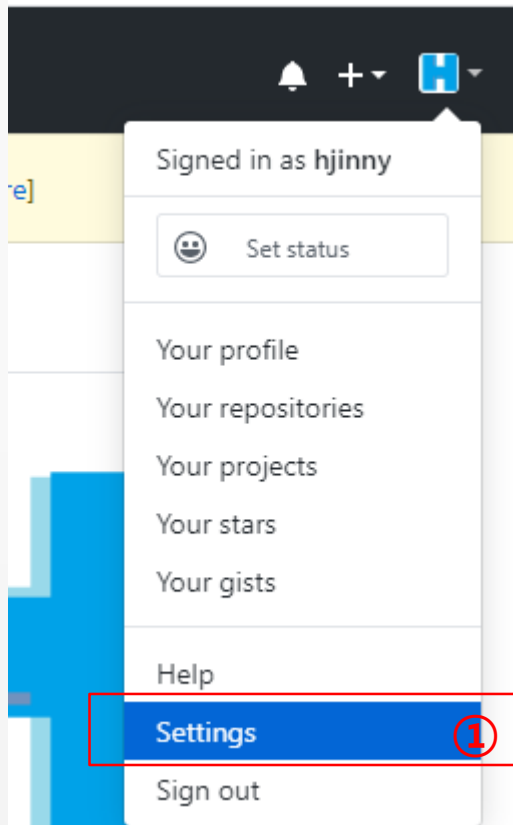
<https://github.ecodesamsung.com/settings/profile>



GitHub(eCODE) Token 생성 (1/2)

• GitHub(eCODE) Personal Access Token

- Local Clone(*git bash/visual code clone*)시, 자격증명이 요구되는 경우, Token정보 필요
- **생성된 Token은, 자격 증명의 password 로 사용** : 반드시 저장
- https://github.ecodesamsung.com/pages/Service/guide/page/ko/user_process/4_clone



GitHub(eCODE) Token 생성 (2/2)

- <https://github.ecodesamsung.com/settings/tokens>
 - Personal access tokens → Generate new token → 생성된 token 복사/저장
 - Generate new token시, Token description을 작성
: repo 부분 체크. (사용에 따라 추가로 체크)

Enterprise Search or jump to... Pull requests Issues Explore

Settings / Developer settings

OAuth Apps
GitHub Apps
Personal access tokens

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Token description

MyToken

What's this token for?

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations

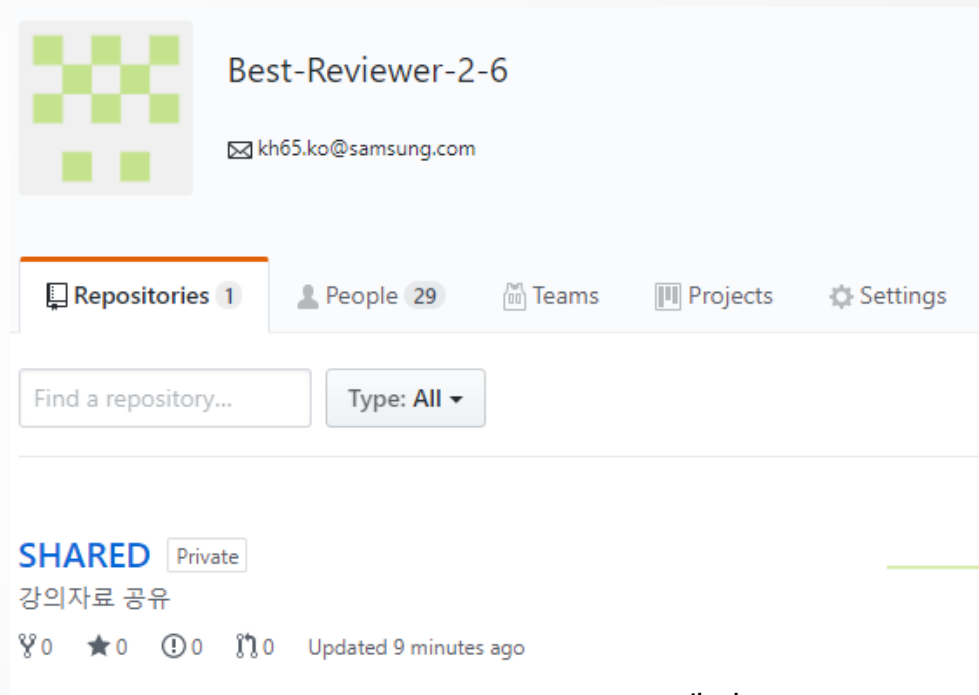
GitHub(eCODE) ORG 확인

- GitHub 내 실습공간

- <https://github.ecodesamsung.com/{개인계정}> : 개인 Repository
- <https://github.ecodesamsung.com/Best-Reviewer-3-12> : ORG Repository
 - › 계정 활성화 후 10분 정도 지나야 ORG 멤버 등록 확인 가능

- ORG 멤버 등록 확인

- : <https://github.ecodesamsung.com/Best-Reviewer-3-12>
- SHARED : 강의 자료 공유 repository

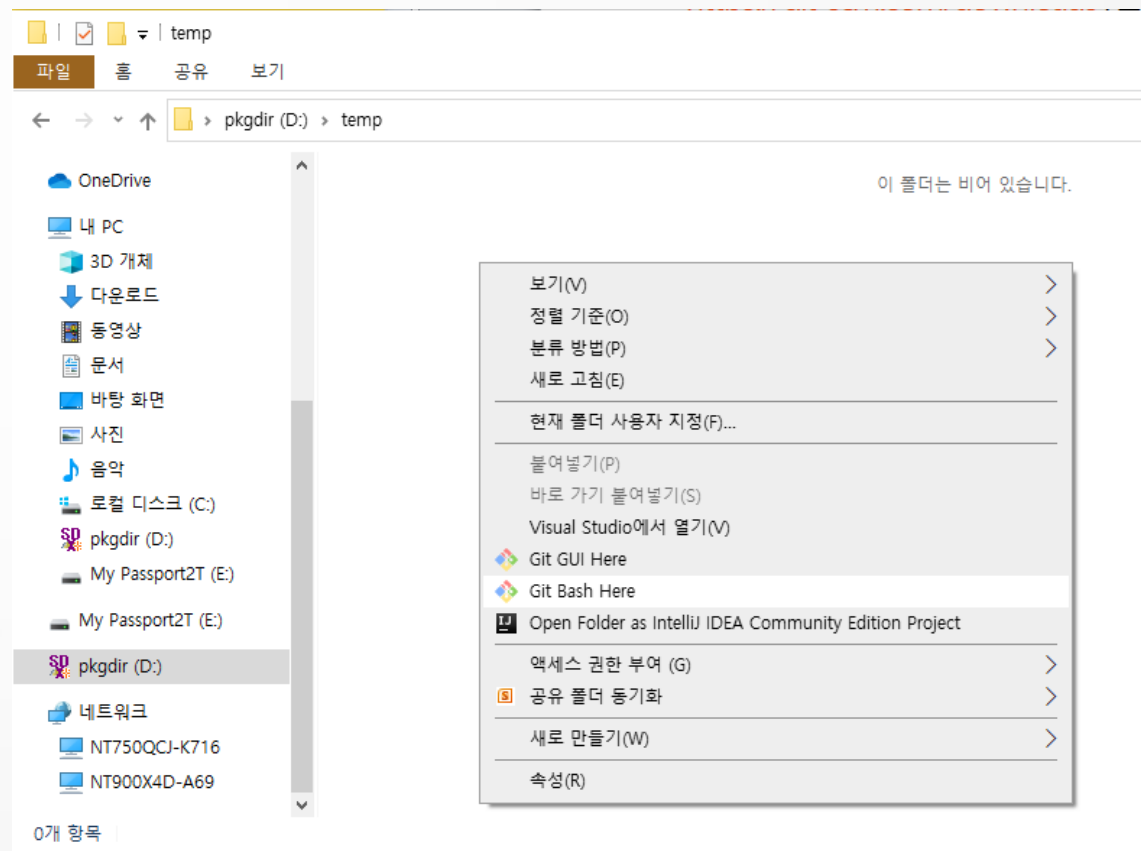


Best-Reviewer-2-6 ORG 예시
ORG명은 메일에서 명시된 이름으로 나타나며,
기본repository는 동일합니다.

Git Client 설치

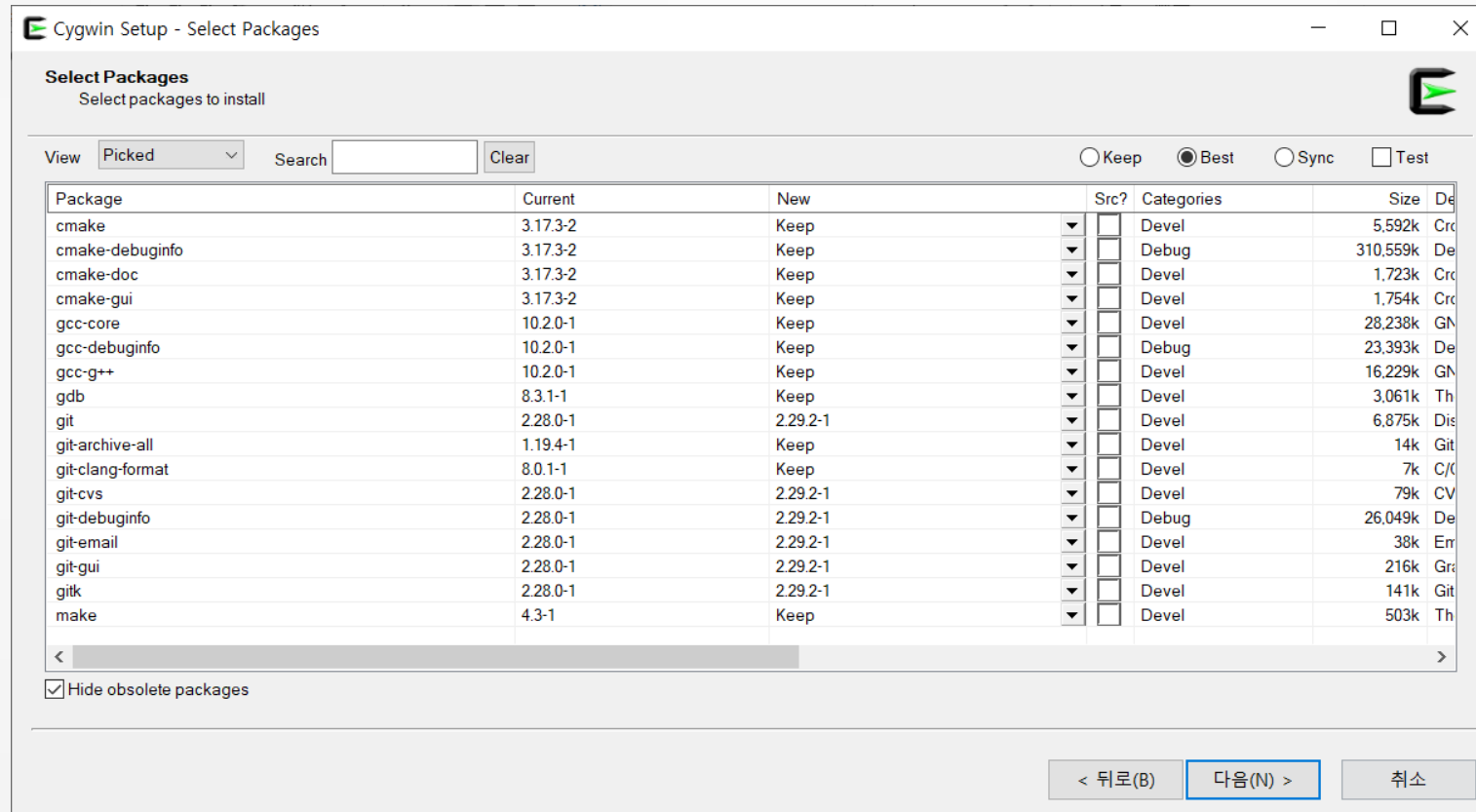
• CLI Git client 프로그램 설치

- <https://git-scm.com/downloads> (윈도우용)
 - 설치확인 방법 : Git bash 혹은 power shell에서, git 명령어 인식
 - > 윈도우 메뉴 -> Git -> Git Bash 또는 powershell, cmd 창
 - > 윈도우 탐색기 -> 우클릭 -> Git Bash Here -> Terminal 창
- \$ git

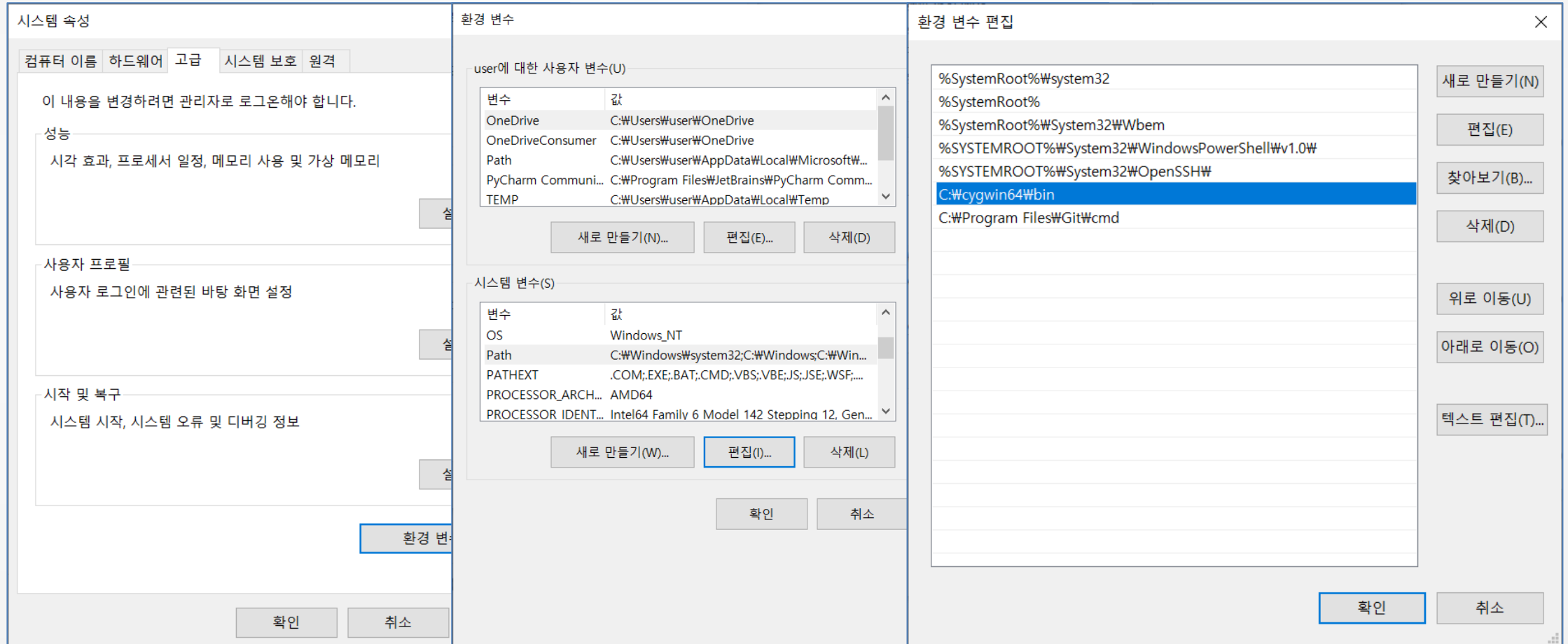


Cygwin 설치 (1/2)

- compile, make를 위한 Cygwin 설치
 - https://cygwin.com/setup-x86_64.exe
 - › gcc, cmake, make, 외 개인적으로 필요한 기능 설치



- 시스템 환경변수 편집
 - Cygwin\bin 을 path에 추가



Google test 설치

- google test download

<https://github.com/google/googletest>

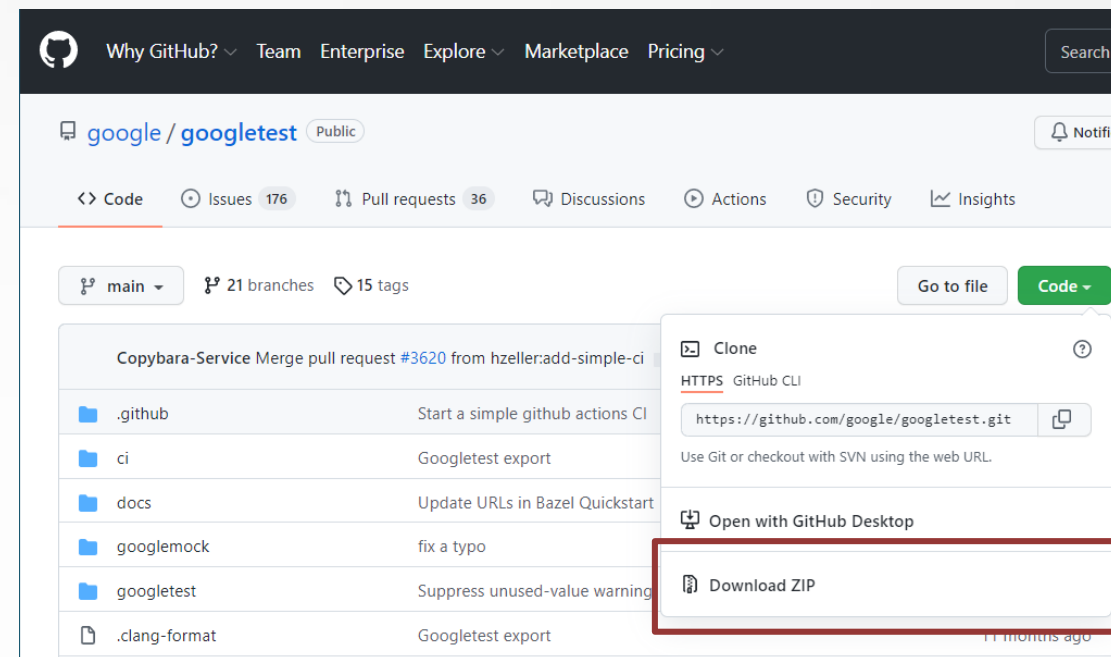
- git접근이 불가능한 경우 파일을 직접 다운로드

<https://github.com/google/googletest/archive/refs/heads/main.zip>

- 압축 해제

- 라이브러리 생성

- Cygwin64 Terminal 실행
- 소스 코드 위치에서 compile 및 실행(다음 페이지 참조)



Install Gtest for Cygwin

```
CORP+andy.shlee@andy-shlee01 /cygdrive/d/test/gtest-master  
$ mkdir build
```

```
CORP+andy.shlee@andy-shlee01 /cygdrive/d/test/gtest-master  
$ cd build
```

```
CORP+andy.shlee@andy-shlee01 /cygdrive/d/test/gtest-master/build  
$ cmake .. -D gtest_disable_pthreads=ON  
-- The C compiler identification is GNU 7.4.0  
:
```

```
CORP+andy.shlee@andy-shlee01 /cygdrive/d/test/gtest-master/build  
$ make  
Scanning dependencies of target gtest  
[ 12%] Building CXX object CMakeFiles/gtest.dir/src/gtest-all.cc.o  
:
```

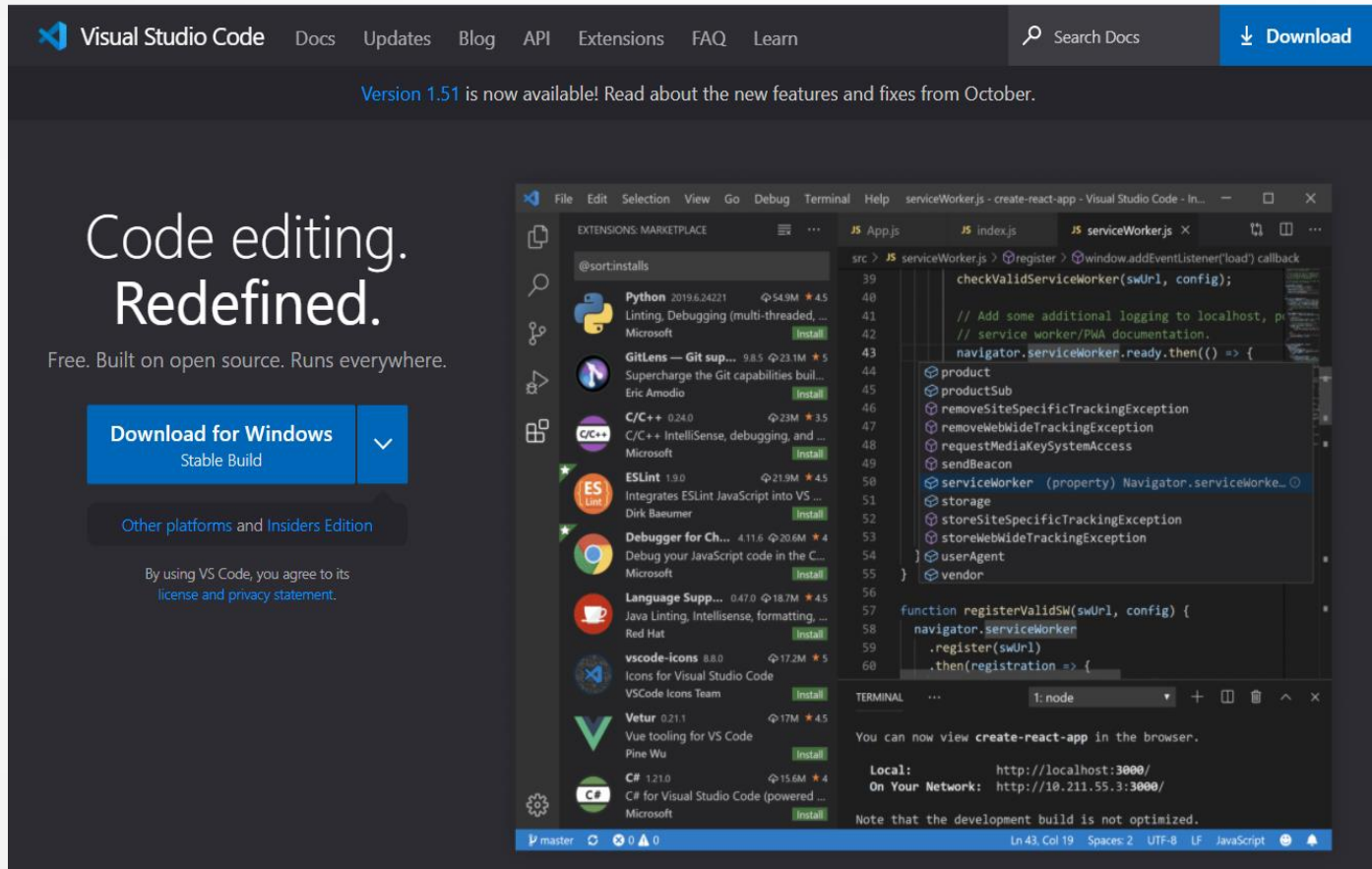
```
CORP+andy.shlee@andy-shlee01 /cygdrive/d/test/gtest-master/build  
$ make install
```

```
$ cp /usr/local/lib/lib* /lib/
```


IDE 설치 (1/4)

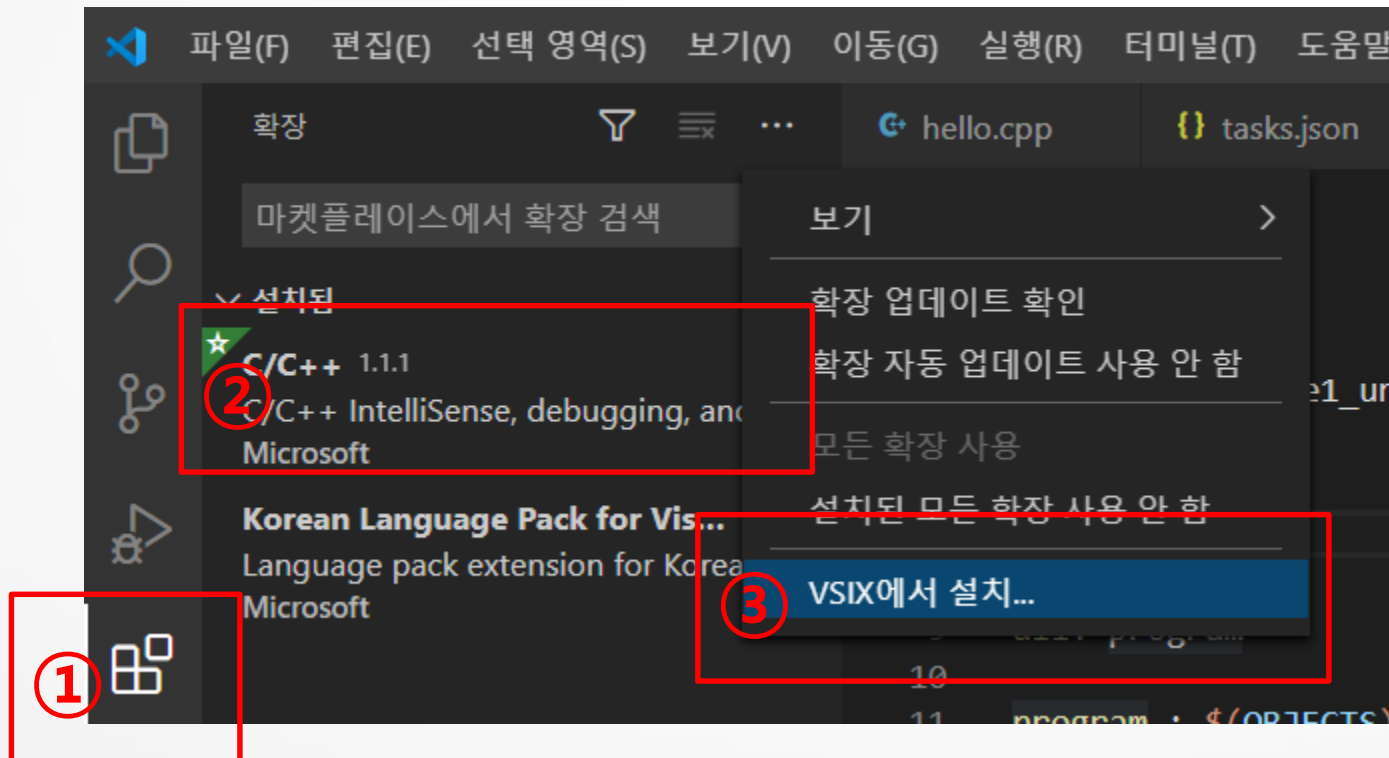
- CPP 실습 및 팀 project 진행을 위한 IDE

- Visual Studio Code : <https://code.visualstudio.com/> (free)

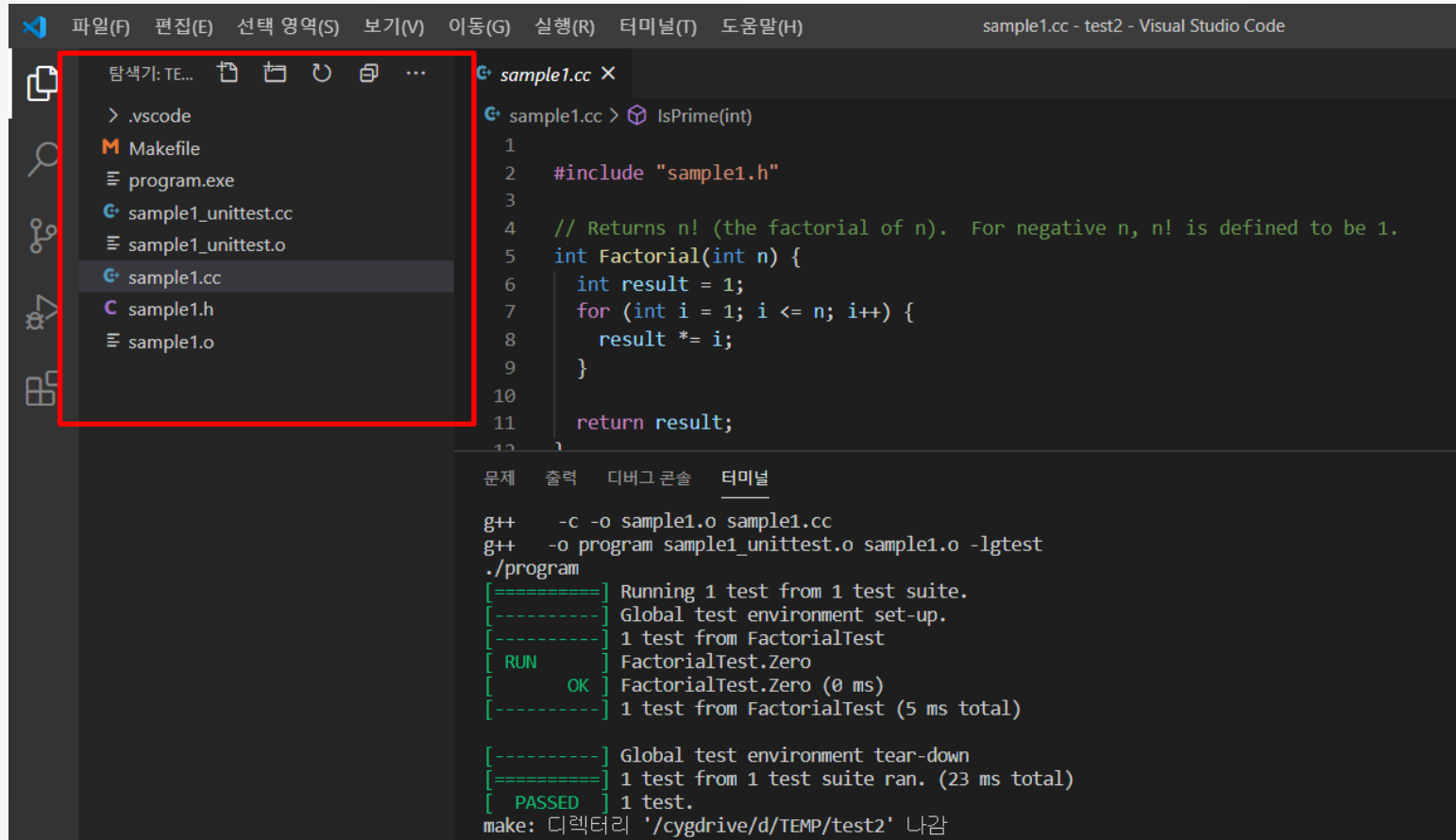


- C/C++ 확장 설치(C/C++ IntelliSense, debugging, and code browsing)

- Visual Studio Code 실행
- 온라인 환경에서는 ① 확장, ② C/C++ 설치
- 오프라인 환경에서는 ① 확장, ③ VSIX에서 설치(cpptools-win32.vsix 설치)



- 테스트 경로에 압축파일(test.zip) 해제



The screenshot shows the Visual Studio Code interface. The file explorer on the left is open, showing a list of files: .vscode, Makefile, program.exe, sample1_unittest.cc, sample1_unittest.o, sample1.cc, sample1.h, and sample1.o. The file sample1.cc is selected. The main editor shows the code for sample1.cc, which includes a header file sample1.h and defines a factorial function. The terminal at the bottom shows the output of the compilation and execution of the test suite.

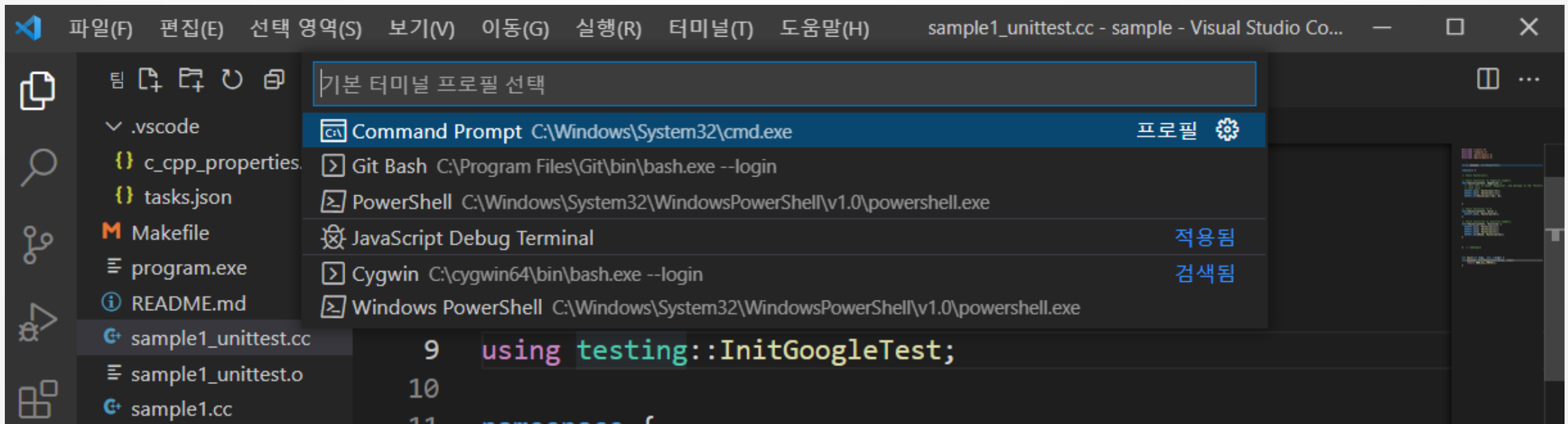
```
sample1.cc ×
sample1.cc > IsPrime(int)
1
2 #include "sample1.h"
3
4 // Returns n! (the factorial of n). For negative n, n! is defined to be 1.
5 int Factorial(int n) {
6     int result = 1;
7     for (int i = 1; i <= n; i++) {
8         result *= i;
9     }
10
11     return result;
12 }

문제 출력 디버그 콘솔 터미널
g++ -c -o sample1.o sample1.cc
g++ -o program sample1_unittest.o sample1.o -lgtest
./program
[=====] Running 1 test from 1 test suite.
[-----] Global test environment set-up.
[-----] 1 test from FactorialTest
[ RUN ] FactorialTest.Zero
[ OK ] FactorialTest.Zero (0 ms)
[-----] 1 test from FactorialTest (5 ms total)

[-----] Global test environment tear-down
[=====] 1 test from 1 test suite ran. (23 ms total)
[ PASSED ] 1 test.
make: 디렉터리 '/cygdrive/d/TEMP/test2' 나감
```

- Cmd를 default로 지정

- Ctrl+Shift+P
- default 검색
- Terminal: Select Default Profile 선택
- Command Prompt 선택



- Ctrl-Shift-B 로 실행 확인

- 실행환경설정

- 터미널 - 기본빌드작업구성 - 템플릿에서 tasks.json 만들기 - others 에 아래 내용 복사

```
{
  "version": "2.0.0",
  "tasks": [
    {
      "label": "build",
      "type": "shell",
      "command": "make",
      "args": ["-C",
        "${fileDirname}"
      ],
      "group": {
        "kind": "build",
        "isDefault": true
      }
    }
  ]
}
```

- 실행환경설정(앞페이지와 동일)

- Ctrl + Shift + p - C/C++ 구성편집(JSON)

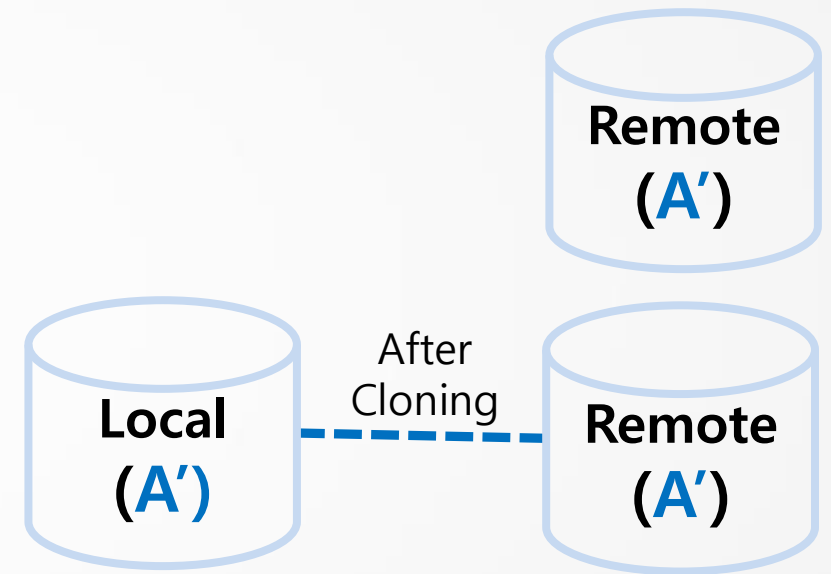
```
{
  "configurations": [
    {
      "name": "Win32",
      "includePath": [
        "${workspaceFolder}/**"
      ],
      "defines": [
        "_DEBUG",
        "UNICODE",
        "_UNICODE"
      ],
      "compilerPath": "C:\\\\cygwin64\\\\bin\\\\gcc.exe",
      "cStandard": "c17",
      "cppStandard": "c++17",
      "intelliSenseMode": "gcc-x64"
    }
  ],
  "version": 4
}
```

2. 실습대비 모의연습 (git-workflow)

Best Reviewer 양성과정

실습 순서

1. GitHub Organization 내, 개인 실습용 Remote Repository 준비
2. Repository clone (Local – Remote)
3. 제공소스 실행(c/c++코드, test코드)
4. Create Branch
5. 코드 수정 및 시험, commit, push
6. Create Pull Request (* 1 PR, PR당 1 commit으로 진행)
7. Code Review(*As Reviewer*) 및 Merge Pull Request



1. 실습용 Remote Repository

- 소스 Repository 이름 (제공)
 - › <https://github.ecodesamsung.com/{Your-Organization}/HelloWorld.git>
- 개인 실습 Repository 이름
 - › HelloWorld_00 (개인번호 01~99)

2. 임시 pair 구성 (author-협업자[reviewer]) *(모든 참가자는 author임과 동시에 다른 누군가의 reviewer)*

3. GitHub Organization 내, 개인 실습용 Remote Repository 준비 [[GitHub](#)]

- 실습용 Repository 생성 : private, README, .gitignore(c++)체크, **Description**, **협업자등록**
 - › Description예) 1조, Author:김나님, Reviewer:박그분 // (1조 혹은 1번)
- 소스 Repository를 복사해 옴 (Download ZIP, UploadFiles 메뉴 활용)

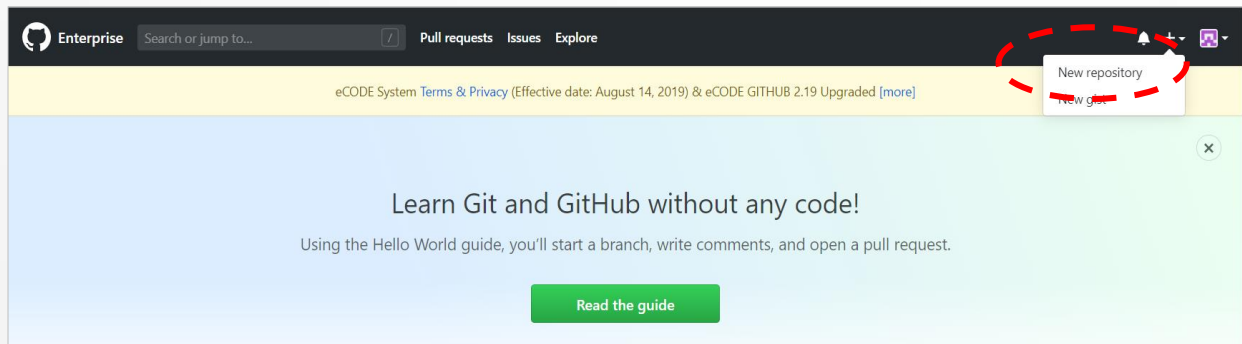
4. 이후, 실습 순서로 진행 (이후 슬라이드 참조)

- **merge-PR까지** 완료 필요

1) 실습용 Remote Repository 준비 (1/3)

• 실습용 Repository 생성

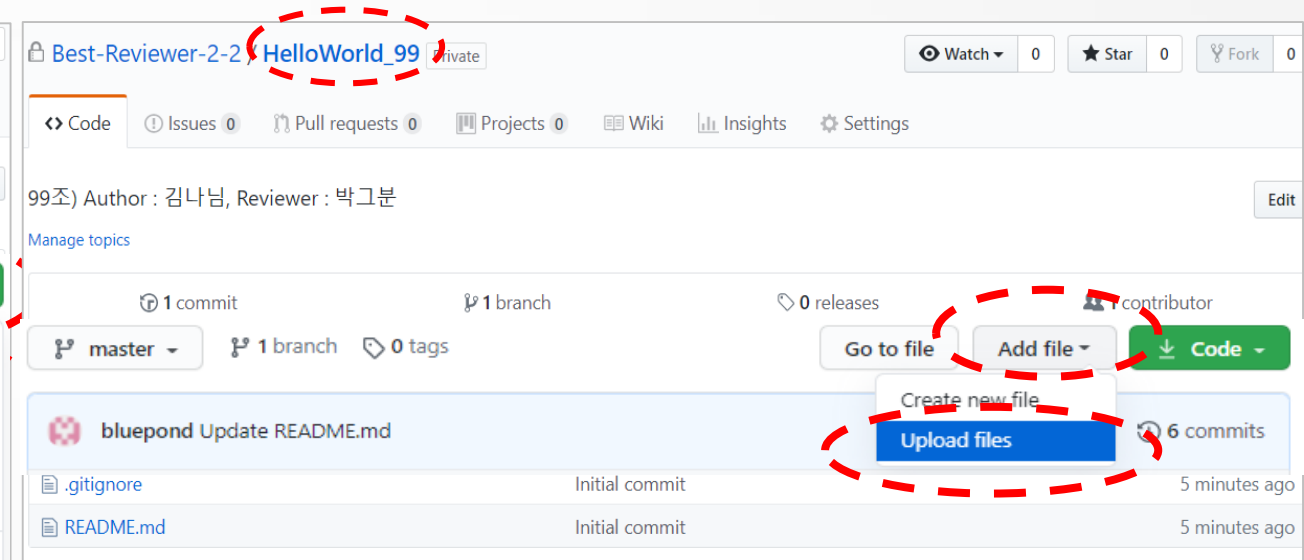
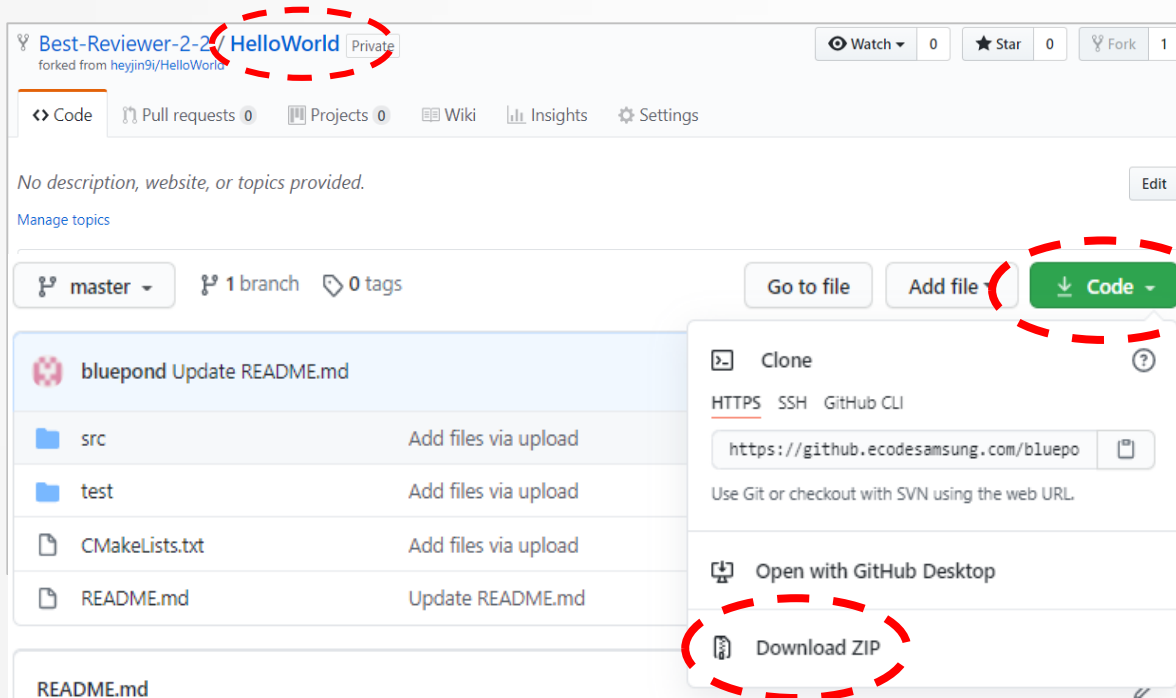
- New Repository 클릭
- repository 위치 확인(**Your-Organization**)
- repository 이름 입력 및 내용 확인
 - › Description, private여부, README화일, .gitignore (c++ 선택)
 - › **Create Repository** 클릭

A screenshot of the 'Create a new repository' form. The form has a white background with a light gray border. At the top, it says 'Create a new repository' and 'A repository contains all project files, including the revision history.' Below this, there are two input fields: 'Owner' with a dropdown menu showing 'Best-Reviewer-3-12' and 'Repository name' with a text input 'HwllloWorld_99' and a green checkmark. A red dashed circle highlights the 'Owner' dropdown and the 'Repository name' input. Below these fields, there's a text input for 'Description (optional)' with the text '99조) Author: 김나님, Reviewer: 박그분'. A red dashed circle highlights the description input. Below the description, there are three radio button options: 'Public' (selected), 'Internal', and 'Private'. A red dashed circle highlights the 'Private' option. Below the options, there's a section 'Initialize this repository with:' with two checked checkboxes: 'Add a README file' and 'Add .gitignore'. A red dashed circle highlights the 'Add .gitignore' checkbox and the '.gitignore template: C++' dropdown. At the bottom, there's a green button labeled 'Create repository'. A red dashed circle highlights the 'Create repository' button.

1) 실습용 Remote Repository 준비 (2/3)

• Repository 복사 (1/2)

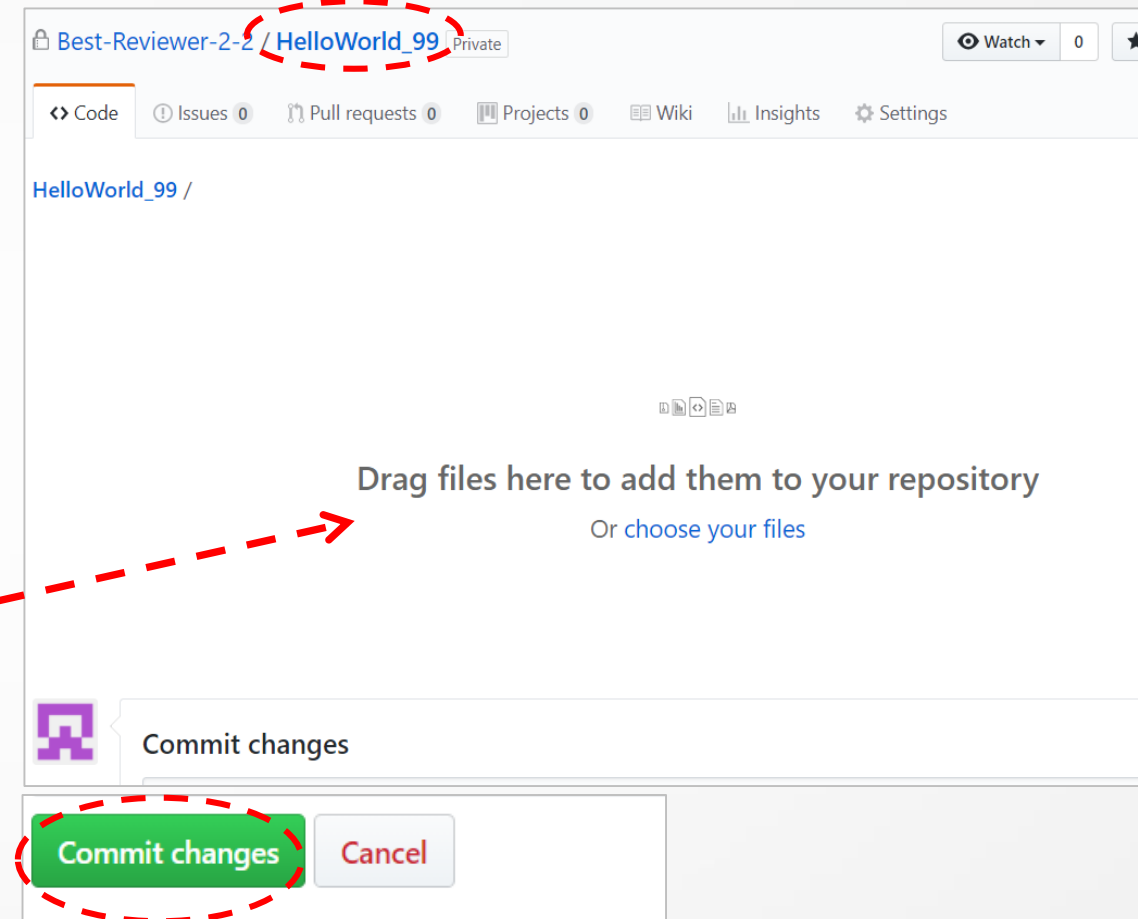
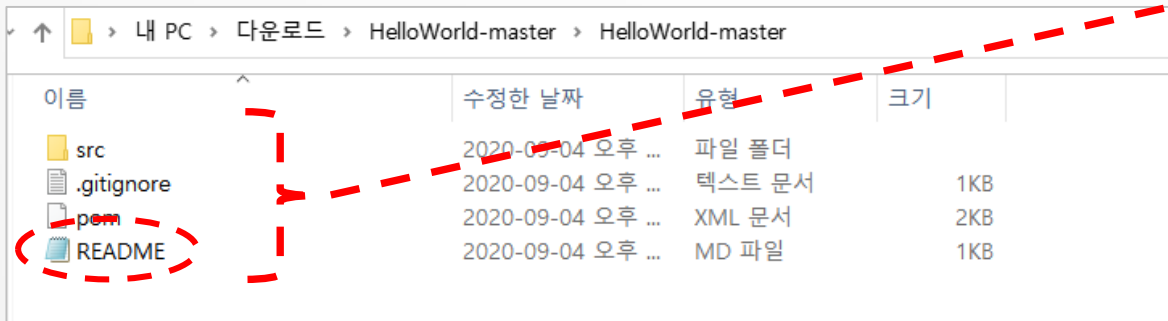
- **소스** Repository에서 **Copy** 클릭 - **DownloadZIP** 클릭
(PC로 다운로드됨 -> 압축해제 필요)
- **실습용** Repository에서 **Add file** 클릭 - **upload files** 클릭



1) 실습용 Remote Repository 준비 (3/3)

• Repository 복사 (2/2)

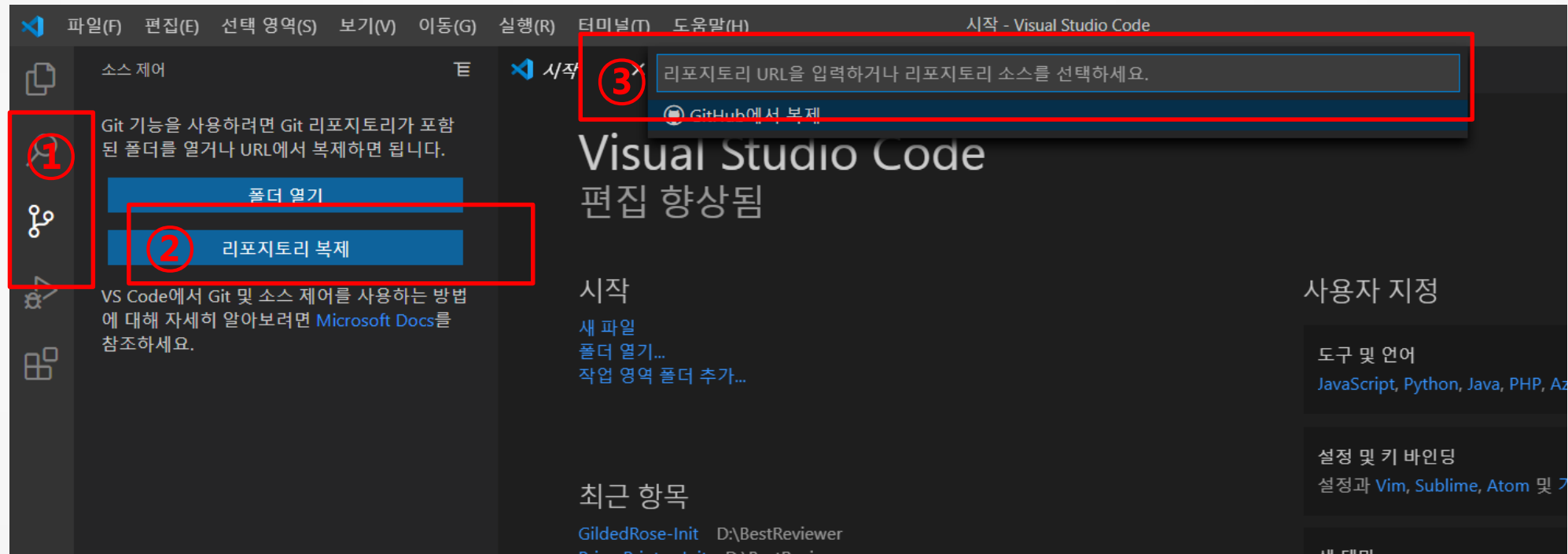
- 나타난 화면 창에(upload Files 실행결과),
앞서 Download ZIP(압축해제) 한
(README화일이 있는 위치에서) 화일/dir.들을 끌어 옴
- **commit changes** 클릭



2) Visual Studio Code Git-Clone (프로젝트 생성) (1/3)

- 실습용 remote Repository와 Clone(복제)

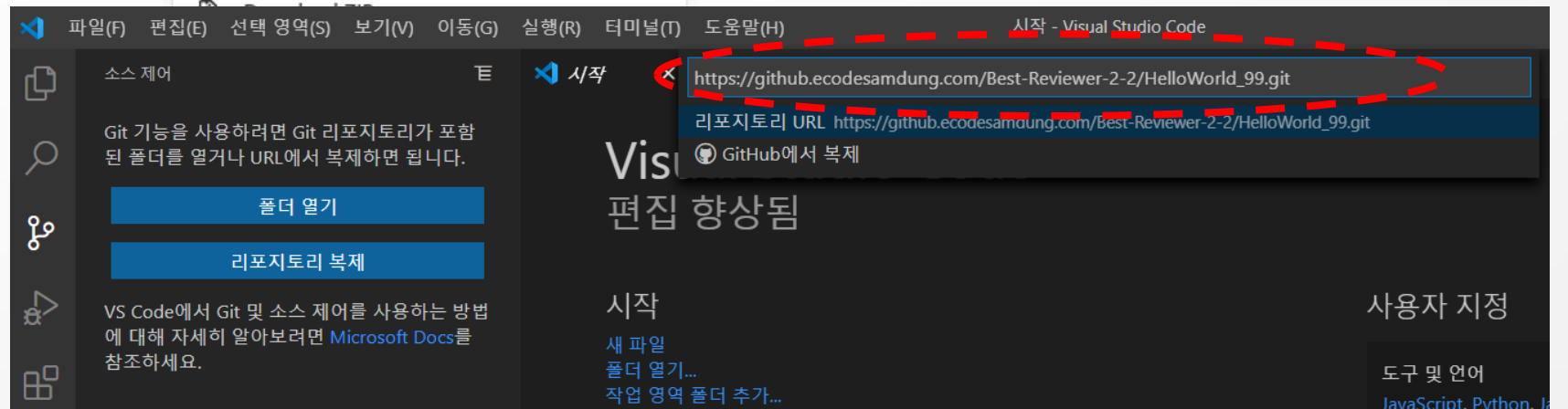
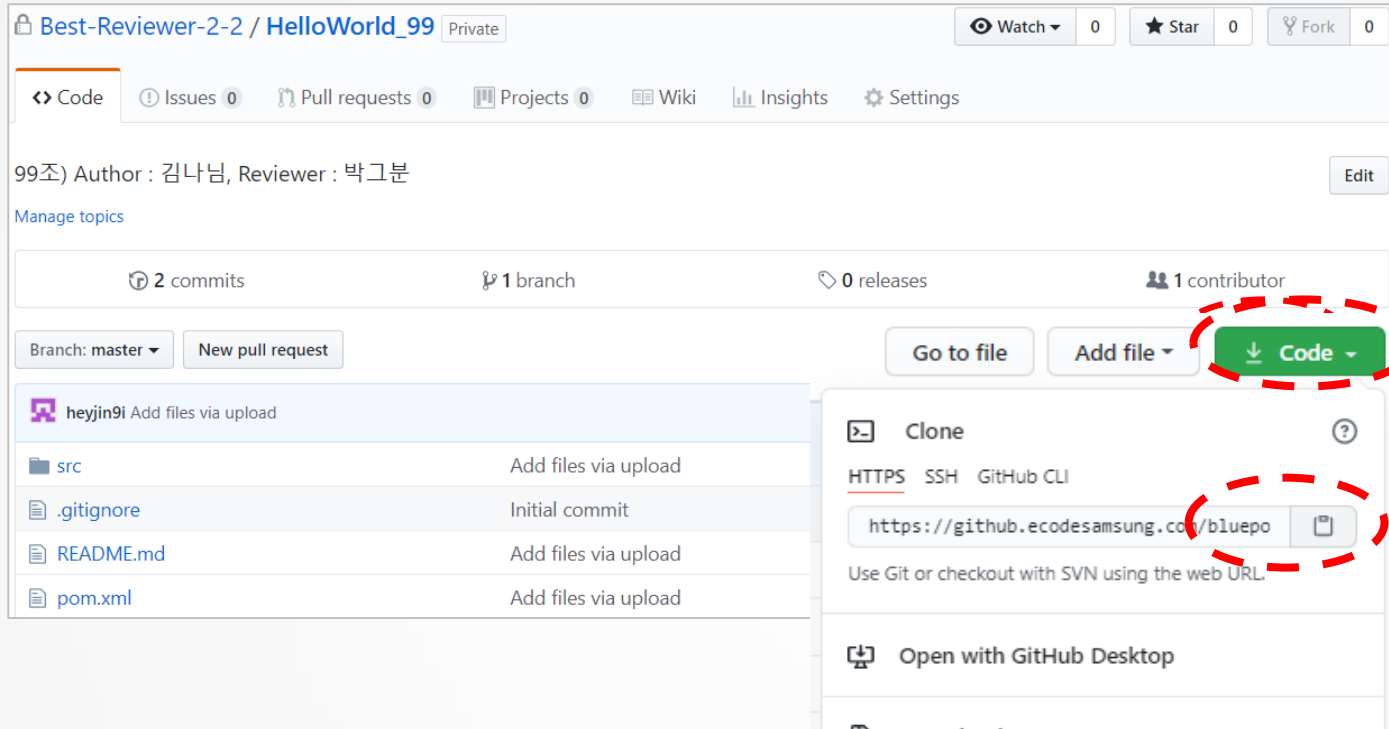
- ①소스제어 ② 리포지토리복제 ③리포지토리 주소 입력
(폴더 생성, local repository 생성 및 cloning)



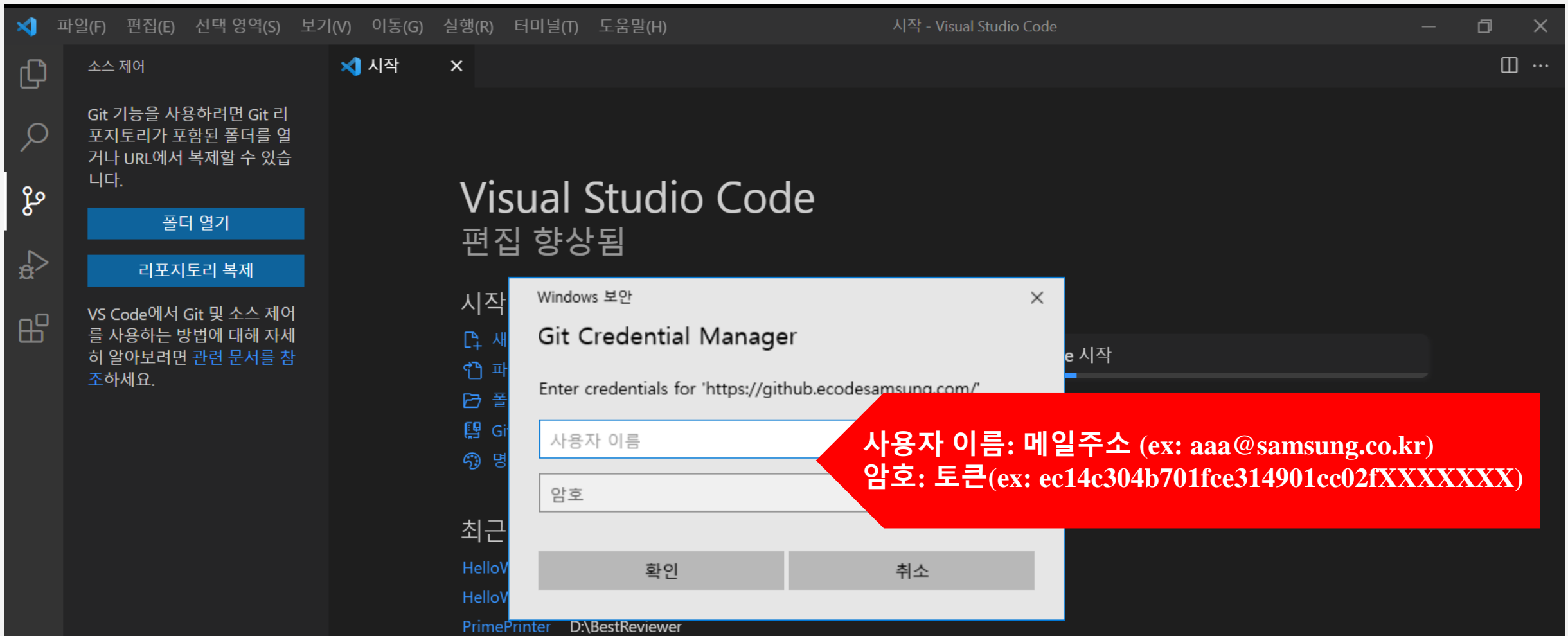
※ 이미 remote repository와 Clone 되어 있는 경우, 폴더열기

※ 새로운 리포지토리 연결을 위하여, 폴더닫기 후 적용

2) Visual Studio Code Git-Clone (프로젝트 생성) (2/3)



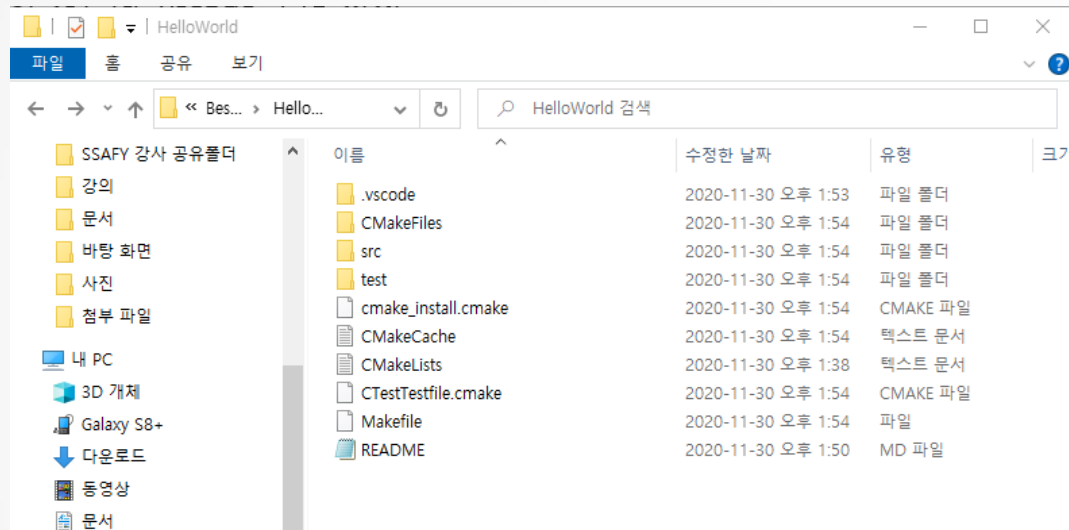
2) Visual Studio Code Git-Clone (프로젝트 생성) (2/3)



3) 제공소스 실행

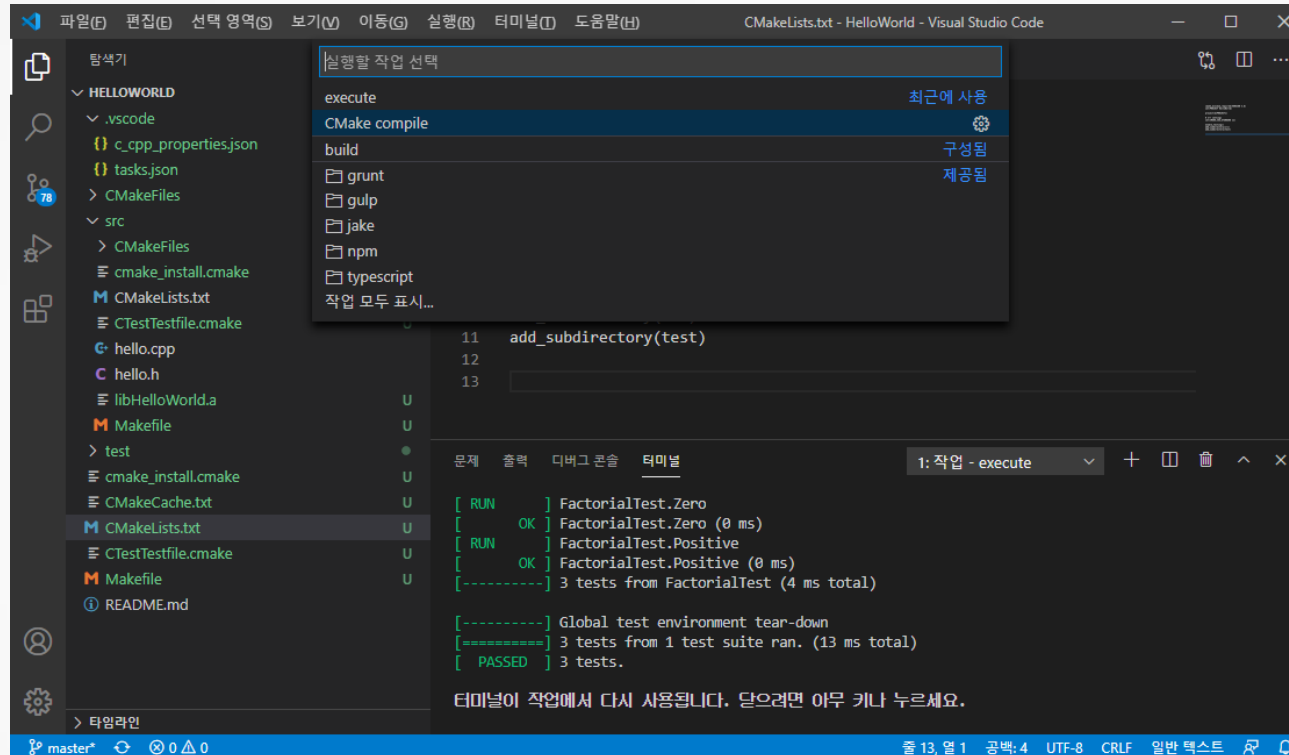
- 셋팅환경 파일 복사

- Git 의 shared repository 에서 vscode.zip 파일을 다운받아 Local 경호에 .vscode 폴더 생성(두 개의 json파일)



3) 제공소스 실행

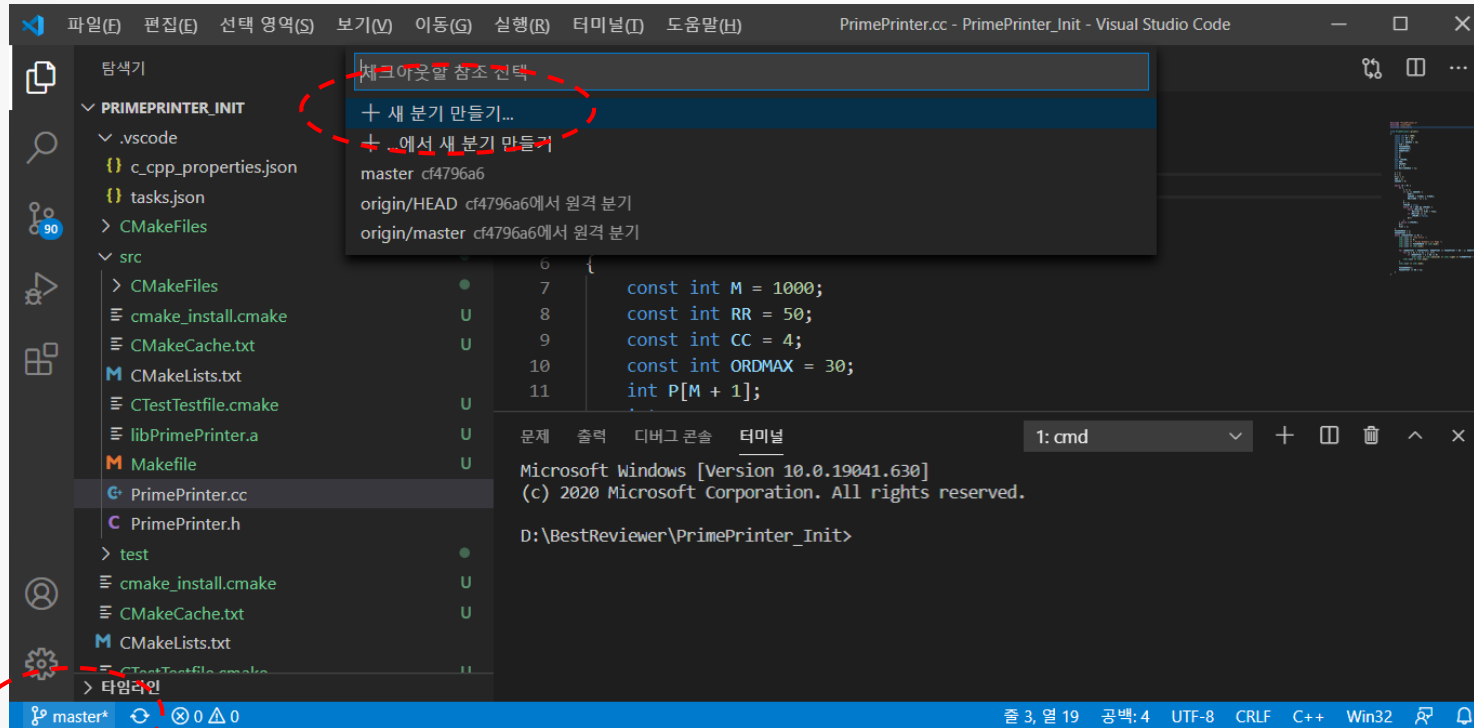
- build: 가장 밖에 있는 cmakeLists.txt 를 선택하고 터미널/작업실행/Cmake Compile



- execute: test 폴더에 생성된 exe 파일을 선택하고 터미널/작업실행/execute

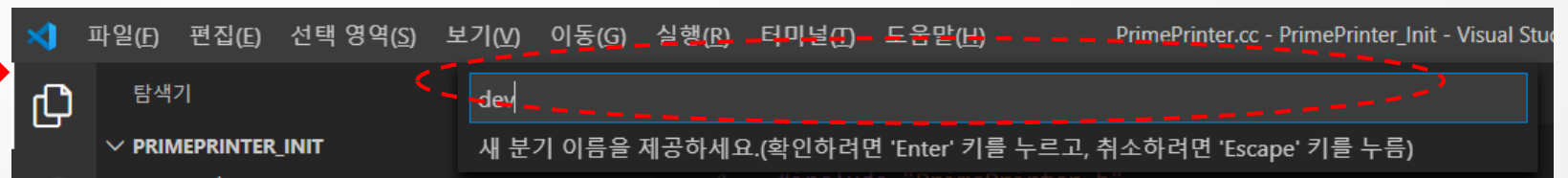
4) 작업 브랜치 추가

• 작업 브랜치 dev 추가



브랜치
이름

새 브랜치
이름 입력



5) Git-Commit/Push (1/4)

- 작업 브랜치에서 코드 수정/실행

- (hello.cpp)

- Commit 수행

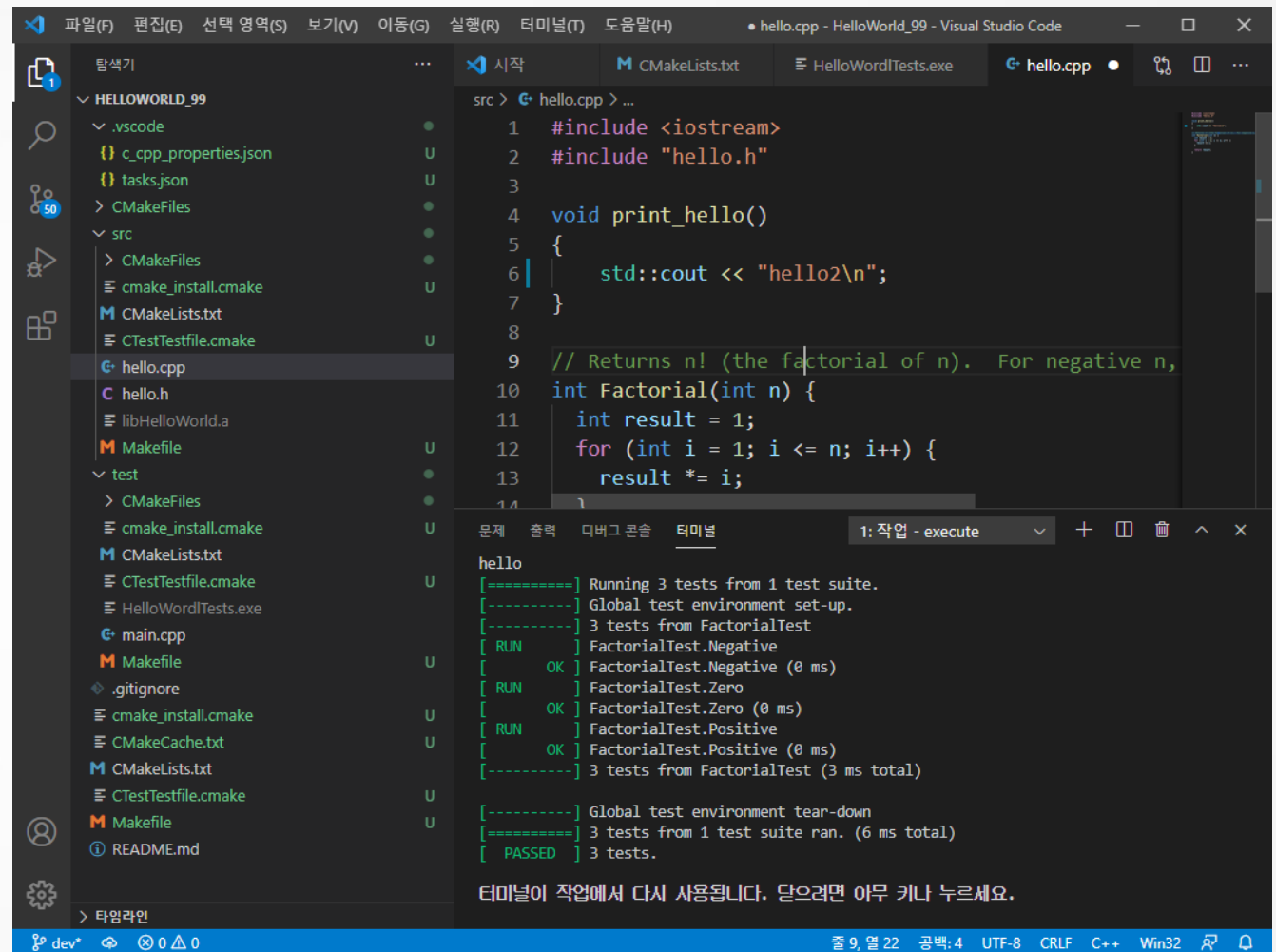
- 소스제어 -> 변경 내용 스테이징

- > 수정한 파일 목록(M) 확인

- 커밋 - 스테이징된 항목 커밋

- 커밋메시지 입력

- 푸시 수행



```
src > G hello.cpp > ...
1  #include <iostream>
2  #include "hello.h"
3
4  void print_hello()
5  {
6      std::cout << "hello2\n";
7  }
8
9  // Returns n! (the factorial of n). For negative n,
10 int Factorial(int n) {
11     int result = 1;
12     for (int i = 1; i <= n; i++) {
13         result *= i;
14     }
15 }
```

문제 출력 디버그 콘솔 터미널 1: 작업 - execute

```
hello
[=====] Running 3 tests from 1 test suite.
[-----] Global test environment set-up.
[-----] 3 tests from FactorialTest
[ RUN    ] FactorialTest.Negative
[ OK     ] FactorialTest.Negative (0 ms)
[ RUN    ] FactorialTest.Zero
[ OK     ] FactorialTest.Zero (0 ms)
[ RUN    ] FactorialTest.Positive
[ OK     ] FactorialTest.Positive (0 ms)
[-----] 3 tests from FactorialTest (3 ms total)

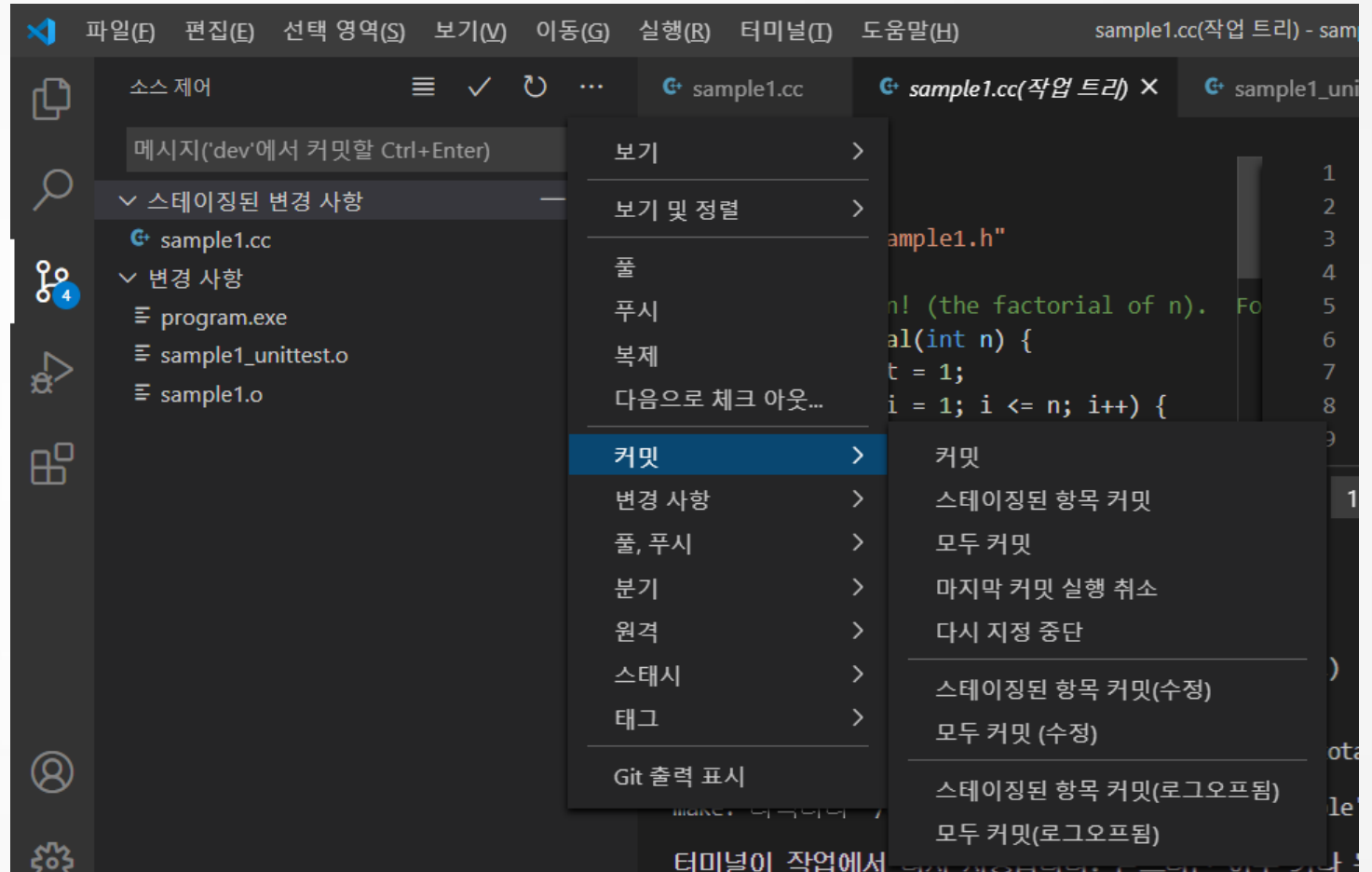
[-----] Global test environment tear-down
[=====] 3 tests from 1 test suite ran. (6 ms total)
[ PASSED ] 3 tests.

터미널이 작업에서 다시 사용됩니다. 닫으려면 아무 키나 누르세요.
```

5) Git-Commit/Push (2/4)

• Commit 수행

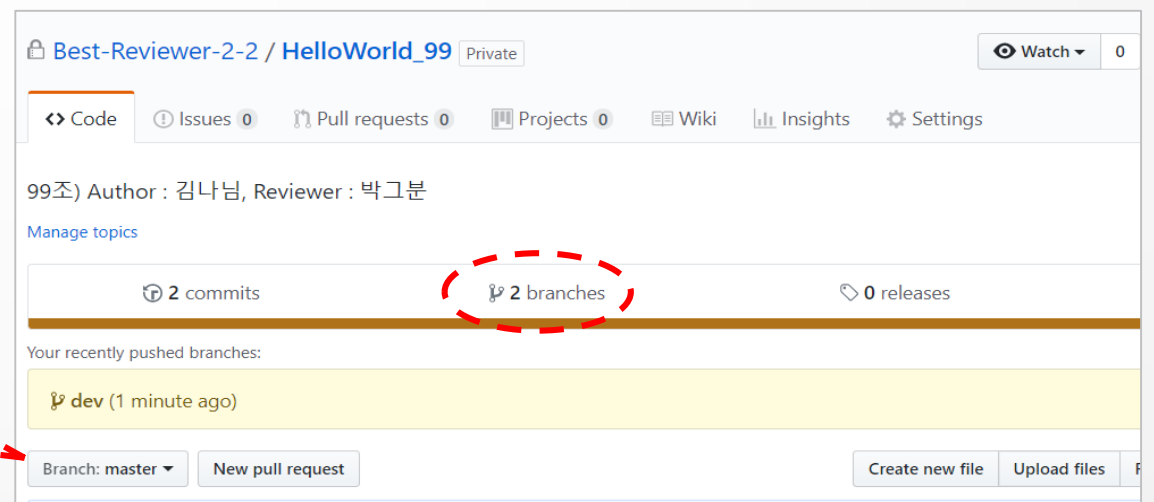
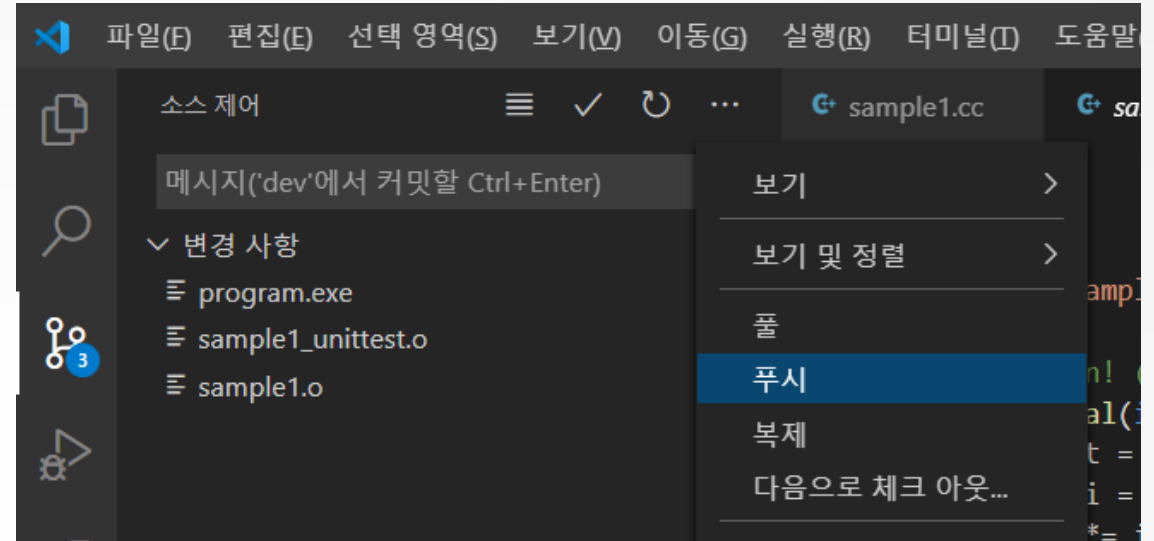
- 소스제어 -> 변경 내용 스테이징
 - › 수정한 파일 목록(M) 확인
- 커밋 - 스테이징된 항목 커밋
 - 커밋메시지 입력
- 푸시 수행



5) Git-Commit/Push (3/4)

• Commit 수행

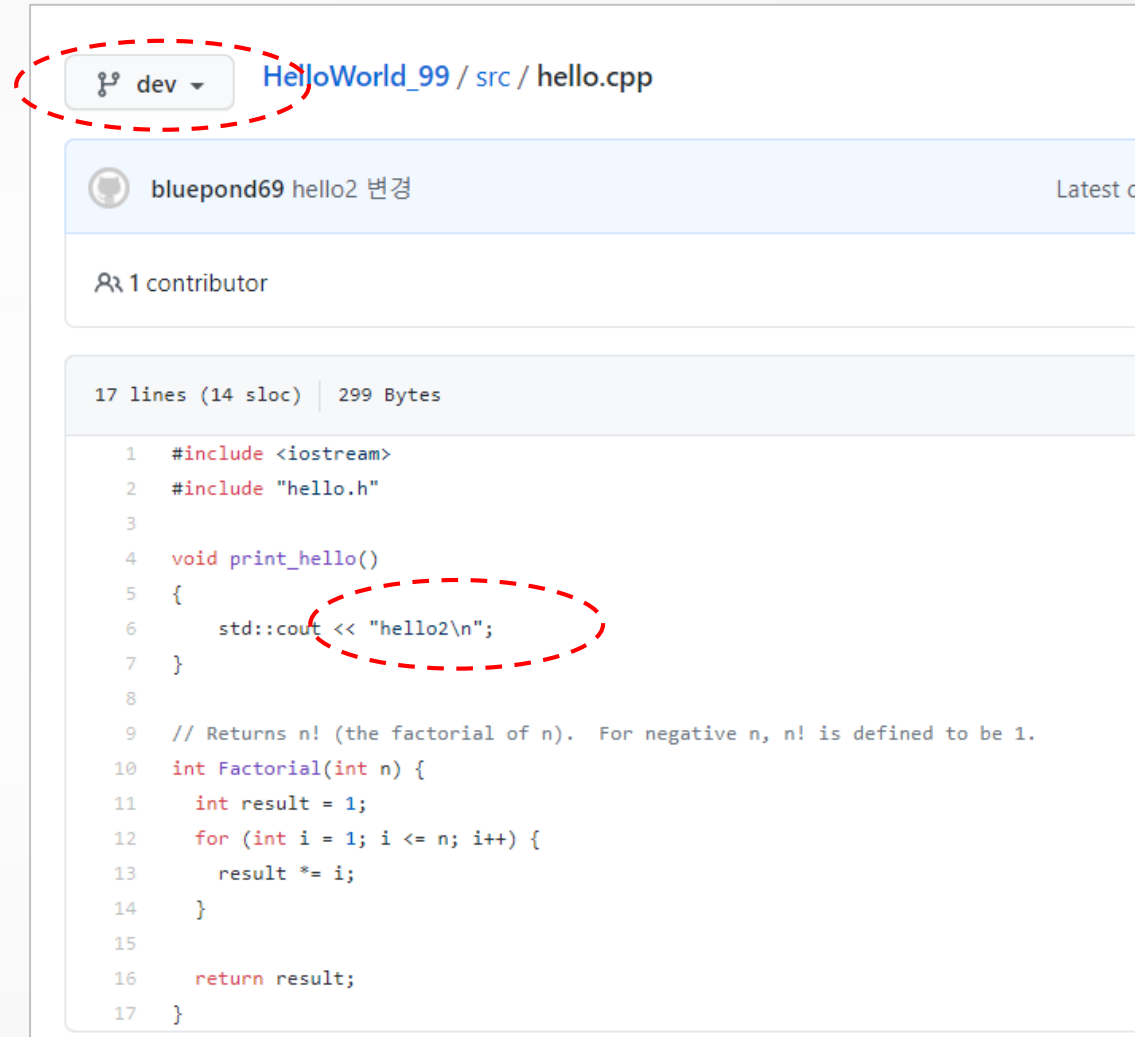
- 소스제어 -> 변경 내용 스테이징
 - › 수정한 파일 목록(M) 확인
- 커밋 - 스테이징된 항목 커밋
커밋메시지 입력
- 푸시 수행
 - * 분기 생성 후 첫 push의 경우
“‘dev’ 분기에는 상향 분기가 없습니다.
이 분기를 게시하겠습니까?” 메시지 출력 - 확인 클릭
- Repository 확인



5) Git-Commit/Push (4/4)

• Commit 수행

- 소스제어 -> 변경 내용 스테이징
 - › 수정한 파일 목록(M) 확인
- 커밋 – 스테이징된 항목 커밋
커밋메시지 입력
- 푸시 수행
- Repository 확인



The screenshot shows a GitHub interface for a repository named 'HelloWorld_99'. The file path is 'src / hello.cpp'. A commit by 'bluepond69' with the message 'hello2 변경' is highlighted. Below the commit, the code diff is shown. The diff indicates 17 lines (14 sloc) and 299 Bytes. The code is as follows:

```
1  #include <iostream>
2  #include "hello.h"
3
4  void print_hello()
5  {
6      std::cout << "hello2\n";
7  }
8
9  // Returns n! (the factorial of n). For negative n, n! is defined to be 1.
10 int Factorial(int n) {
11     int result = 1;
12     for (int i = 1; i <= n; i++) {
13         result *= i;
14     }
15
16     return result;
17 }
```

In the screenshot, the 'dev' branch selector and the line of code 'std::cout << "hello2\n";' are circled with red dashed lines.

6) [Author] Pull Request 생성 (1/2)

(Author가, 자신이 push 한 commit들에 대해 PR을 생성)

- 브랜치 확인후, Compare&pullrequest 클릭

The screenshot shows the GitHub interface for a repository named 'Best-Reviewer-2-2 / HelloWorld_99'. The repository is private. The top navigation bar includes 'Code', 'Issues 0', 'Pull requests 0', 'Projects 0', 'Wiki', 'Insights', and 'Settings'. Below this, the repository name and 'Private' status are shown, along with 'Watch 0', 'Star 0', and 'Fork 0' buttons. The main content area shows the repository description '99조) Author : 김나님, Reviewer : 박그분' and a 'Manage topics' link. A summary bar displays '3 commits', '2 branches', '0 releases', and '1 contributor'. Under the heading 'Your recently pushed branches:', a yellow box highlights the 'dev' branch, pushed '11 minutes ago'. To the right of this box, a green button labeled 'Compare & pull request' is visible. Both the 'dev' branch entry and the 'Compare & pull request' button are circled with red dashed lines.

6) [Author] Pull Request 생성 (2/2)

- PR의 대상 브랜치 확인

- commit이 들어있는 브랜치
- Merge-into 브랜치 (base 브랜치)

- PR 내용 작성

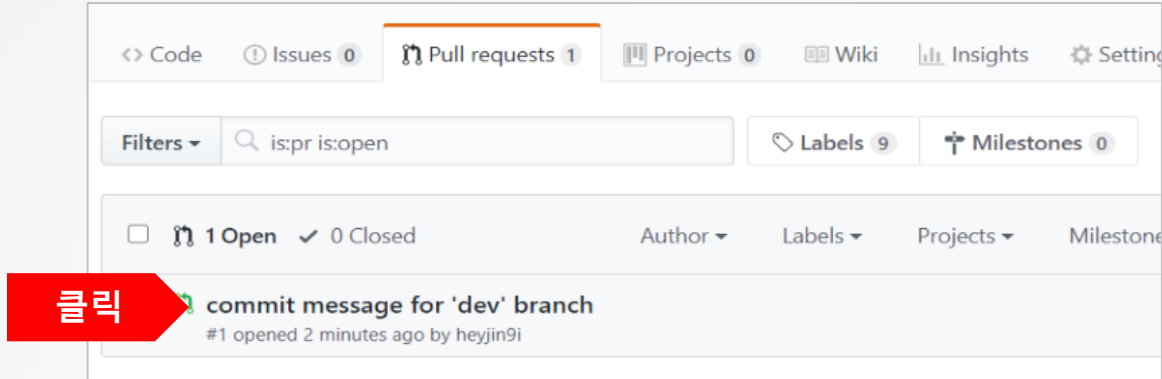
- PR Title, Message 작성
- PR을 보낼 **Reviewer** 지정
- **Create pull request** 클릭

The screenshot shows the GitHub 'Open a pull request' page. Red dashed circles highlight the following elements:

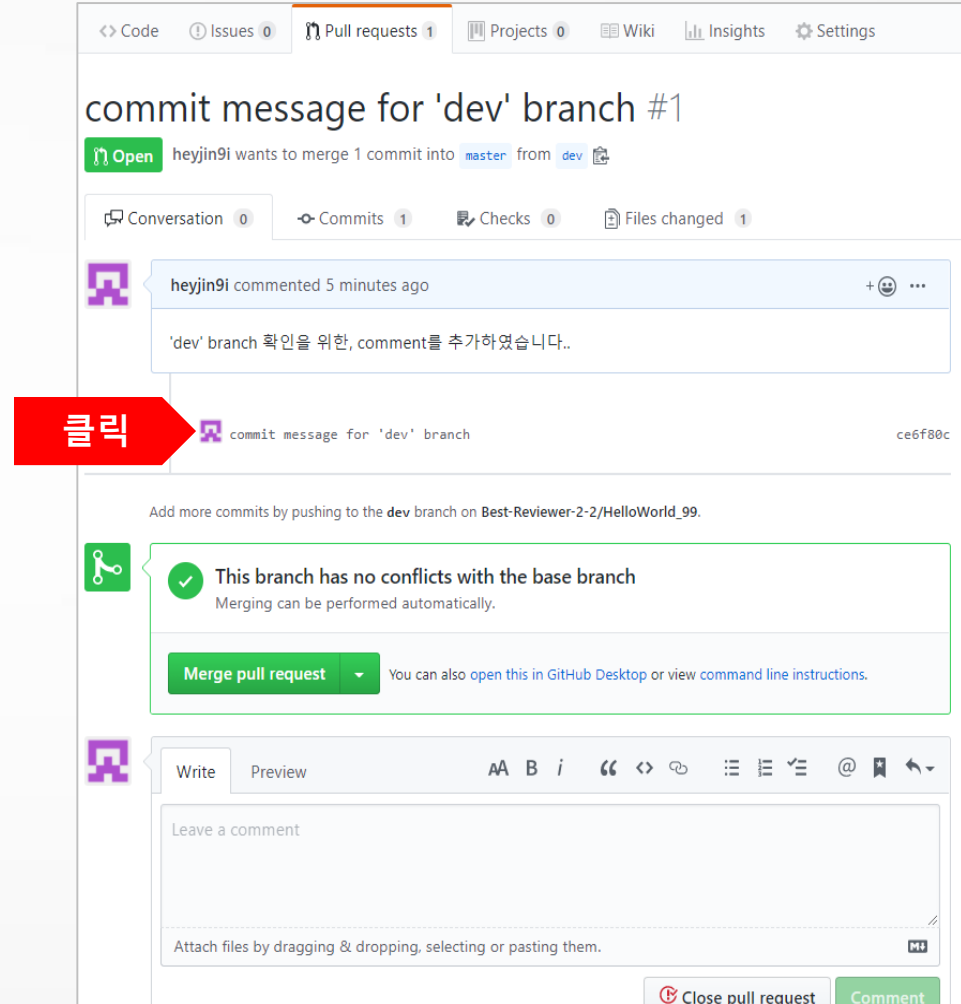
- The title 'Open a pull request' at the top.
- The 'base: master' and 'compare: dev' dropdown menus, along with the 'Able to merge' status.
- The 'commit message for 'dev' branch' text input field.
- The text area containing the message: 'dev' branch 확인을 위한, comment를 추가하였습니다..
- The 'Create pull request' button at the bottom right.
- The 'Reviewers' section on the right sidebar, which currently shows 'No reviews'.

7-1) [Reviewer] Code Review & Communication (1/4)

- 메일 수신후, 요청된 PR 클릭



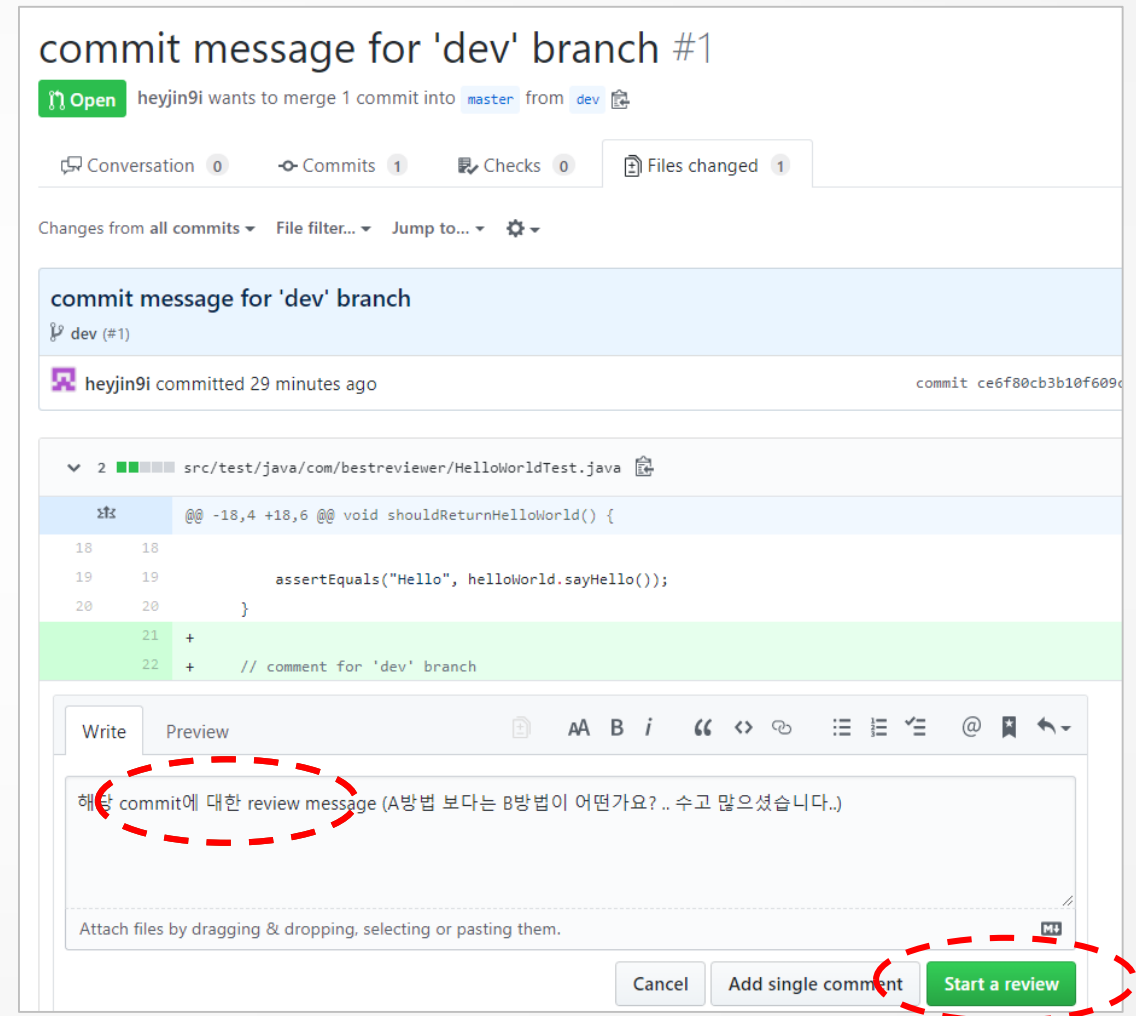
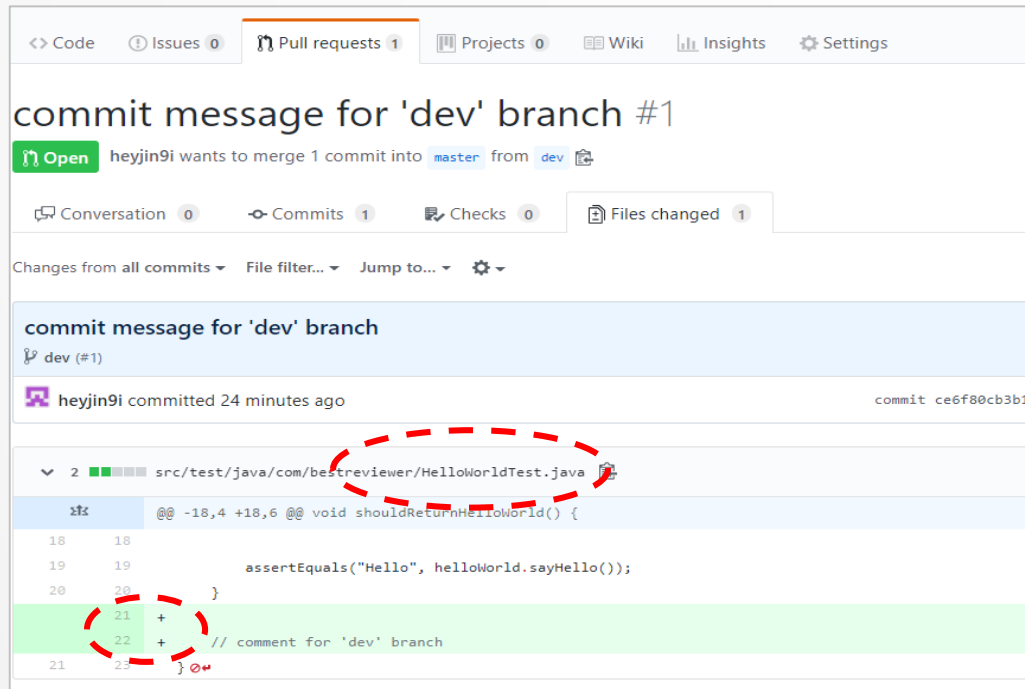
PR에 포함된 commit



7-1) [Reviewer] Code Review & Communication (2/4)

• Review내용 작성

- 해당 Commit의 코드 변경 부분 확인
- + 부분을 클릭후, Review 내용 작성
 - > (혹은 +부분들을 묶어서 클릭)
- **Start review** 혹은 Add review comment 클릭
- PR에 포함된 다른 commit에 대해서도 Review작성



7-1) [Reviewer] Code Review & Communication (3/4)

• Submit Review

- PR/commit에 대한 Review들을 완료한 후,
Finish your Review/Review changes 클릭
- 작성한 Review들에 대한, 전체 comment를 입력
- Review의 성격을 체크
 - › (Comment, Approve, Request)
- **Submit review** 클릭

Write Preview AA B i “ <> @

해당 PR(혹은 commit)에 대한 리뷰의견을 작성하였습니다..

Attach files by dragging & dropping, selecting or pasting them.

☒ **Comment**
Submit general feedback without explicit approval.

☐ **Approve**
Submit feedback and approve merging these changes.

☐ **Request changes**
Submit feedback that must be addressed before merging.

Submit review Cancel review 3 pending comments

commit message for 'dev' branch #1

Open heyjin9i wants to merge 1 commit into master from dev

Conversation 0 Commits 1 Checks 0 Files changed 1 +2 -0

Changes from all commits File filter... Jump to... Finish your review 3

commit message for 'dev' branch

dev (#1)

heyjin9i committed 41 minutes ago commit ce6f80cb3b10f609c5c29ba2a46c9c047a4e46d3

src/test/java/com/bestreviewer/HelloWorldTest.java

```
@@ -18,4 +18,6 @@ void shouldReturnHelloWorld() {
18 18
19 19     assertEquals("Hello", helloWorld.sayHello());
20 20 }
21 +
22 + // comment for 'dev' branch
```

heyjin9i Author Pending
해당 commit에 대한 review message (A방법 보다는 B방법이 어떤가요? .. 수고 많으셨습니다..)

Reply...

heyjin9i Author Pending
2번째 review message

7-1) [Reviewer] Code Review & Communication (4/4)

(Author는, Reviewer의 Code Review에 대해 Reply)

• Communication

- Reviewer의 Review 내용을 확인
- Review마다 Reply 작성
- **Resolve Conversation** 클릭
- Review 내용을 코드에 반영 후, Commit/Push
 - › (Local – **Visual Code**)
 - › 새로운 commit은 기존 PR에 포함됨

The screenshot shows a GitHub code review interface. At the top, a user named 'heyjin9i' has reviewed 1 minute ago. Below this, a comment box shows 'heyjin9i left a comment' with an 'Author' button and a '+ 😊 ...' menu. The comment text reads: '해당 PR(혹은 commit)에 대한 리뷰의견을 작성하였습니다..'. Below the comment is a code diff for 'src/test/java/com/bestreviewer/HelloWorldTest.java'. The diff shows lines 18-20 unchanged, and lines 21-22 added: '+ // comment for \'dev\' branch'. Below the code diff, another comment from 'heyjin9i' is shown, stating: '해당 commit에 대한 review message (A방법 보다는 B방법이 어떤가요? .. 수고 많으셨습니다..)'. At the bottom, there is a 'Reply...' input field and a 'Resolve conversation' button, both of which are circled with a red dashed line. A red arrow points from the text 'Review에 대한 댓글 작성' to the 'Resolve conversation' button.

heyjin9i reviewed 1 minute ago [View changes](#)

heyjin9i left a comment [Author](#) + 😊 ...

해당 PR(혹은 commit)에 대한 리뷰의견을 작성하였습니다..

```
src/test/java/com/bestreviewer/HelloWorldTest.java
...    ...    @@ -18,4 +18,6 @@ void shouldReturnHelloWorld() {
18      18
19      19          assertEquals("Hello", helloWorld.sayHello());
20      20      }
21      +
22      +      // comment for 'dev' branch
```

heyjin9i 1 minute ago [Author](#) + 😊 ...

해당 commit에 대한 review message (A방법 보다는 B방법이 어떤가요? .. 수고 많으셨습니다..)

[Resolve conversation](#)

Review에 대한
댓글 작성

7-2) [Author] Merge Pull-Request (1/2)

(Author가 PR을 Merge)

- **Merge pull request**

- Merge pull request 클릭
- Confirm merge 클릭
- Merge 수행됨
(PR생성시, 지정한 base 브랜치로)

The screenshot displays a GitHub pull request interface. At the top, a review comment by 'heyjin9i' is shown, stating '해당 PR(혹은 commit)에 대한 리뷰의견을 작성하였습니다..'. Below the comment, three code snippets are listed, each with a 'Show resolved' link. A green box with a checkmark icon indicates 'This branch has no conflicts with the base branch' and 'Merging can be performed automatically.' A red dashed circle highlights the 'Merge pull request' button. Below this, a message states 'Pull request successfully merged and closed' and 'You're all set—the dev branch can be safely deleted.', with a 'Delete branch' button.

heyjin9i reviewed 2 minutes ago [View changes](#)

heyjin9i left a comment [Author](#) + 😊 ...

해당 PR(혹은 commit)에 대한 리뷰의견을 작성하였습니다..

src/test/java/com/bestreviewer/HelloWorldTest.java [Show resolved](#)

src/test/java/com/bestreviewer/HelloWorldTest.java [Show resolved](#)

src/test/java/com/bestreviewer/HelloWorldTest.java [Show resolved](#)

Add more commits by pushing to the **dev** branch on **Best-Reviewer-2-2/HelloWorld_99**.

✓ This branch has no conflicts with the base branch
Merging can be performed automatically.

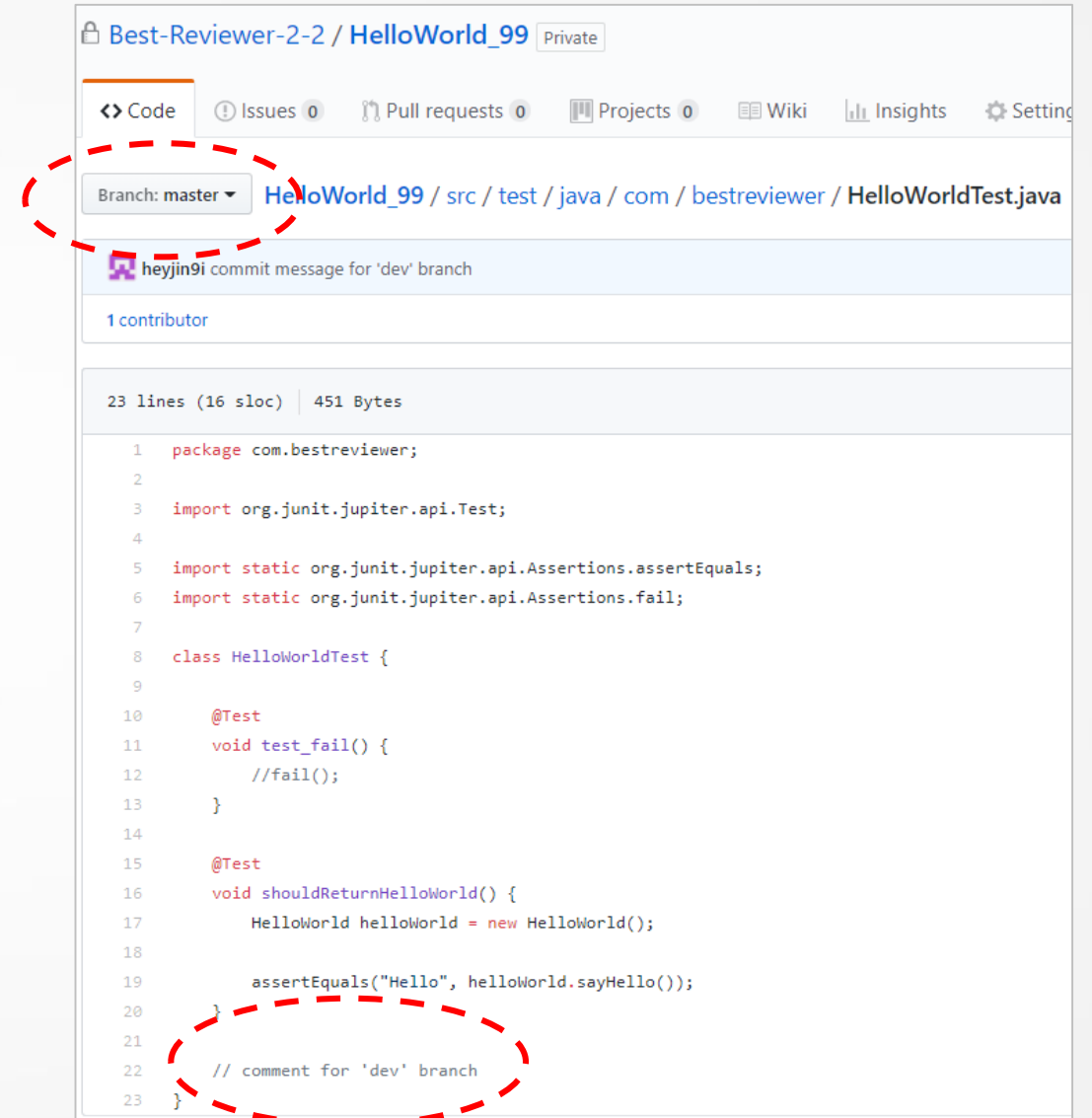
Merge pull request You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Pull request successfully merged and closed [Delete branch](#)
You're all set—the **dev** branch can be safely deleted.

7-2) [Author] Merge Pull-Request (2/2)

- Merge 된 코드 확인

- PR에서 지정된 base 브랜치 확인
- 코드 수정 내용 확인



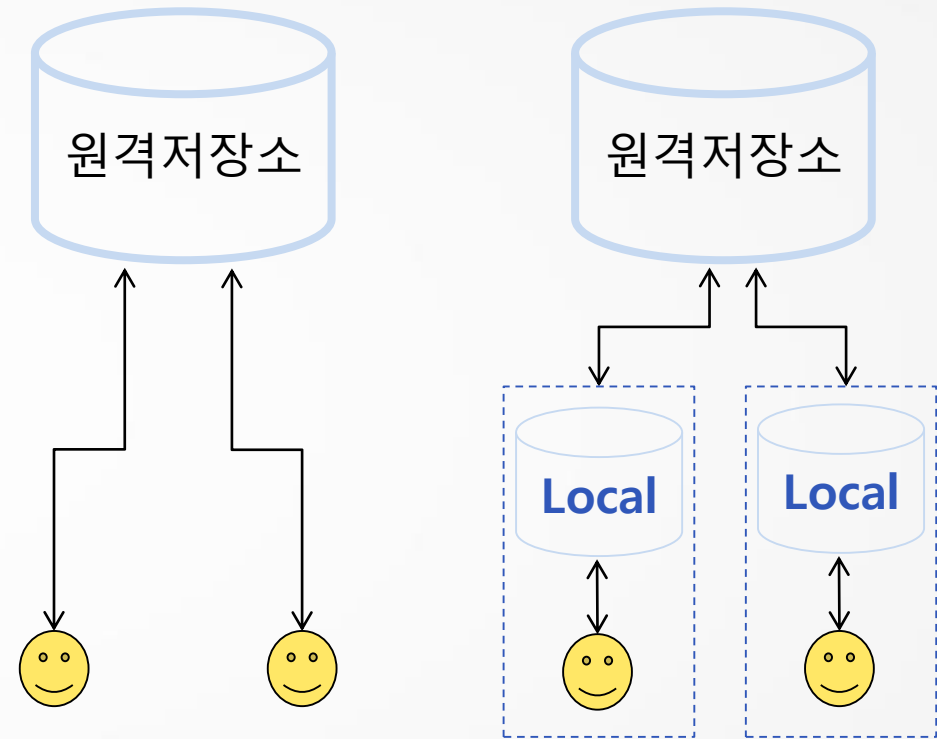
The screenshot shows a GitHub pull request page for the repository 'Best-Reviewer-2-2 / HelloWorld_99'. The page is titled 'Branch: master' and shows the file path 'HelloWorld_99 / src / test / java / com / bestreviewer / HelloWorldTest.java'. The pull request is from user 'heyjin9i' and is for the 'dev' branch. The code content is displayed, showing a Java class 'HelloWorldTest' with two test methods: 'test_fail()' and 'shouldReturnHelloWorld()'. The 'test_fail()' method is circled in red, and the 'shouldReturnHelloWorld()' method is also circled in red. The code is as follows:

```
1 package com.bestreviewer;
2
3 import org.junit.jupiter.api.Test;
4
5 import static org.junit.jupiter.api.Assertions.assertEquals;
6 import static org.junit.jupiter.api.Assertions.fail;
7
8 class HelloWorldTest {
9
10     @Test
11     void test_fail() {
12         //fail();
13     }
14
15     @Test
16     void shouldReturnHelloWorld() {
17         HelloWorld helloWorld = new HelloWorld();
18
19         assertEquals("Hello", helloWorld.sayHello());
20     }
21
22     // comment for 'dev' branch
23 }
```

***참고) Git 기본**

Best Reviewer 양성과정

- 분산 소스 버전 관리시스템
- Geometric Invariant Theory를 바탕으로 설계
- 메인 서버나 통신에 문제가 있어도 개발 진행
(원격저장소, 지역저장소)



Centralized Version Control Vs. Distributed Version Control

Repository Clone (Local - Remote)

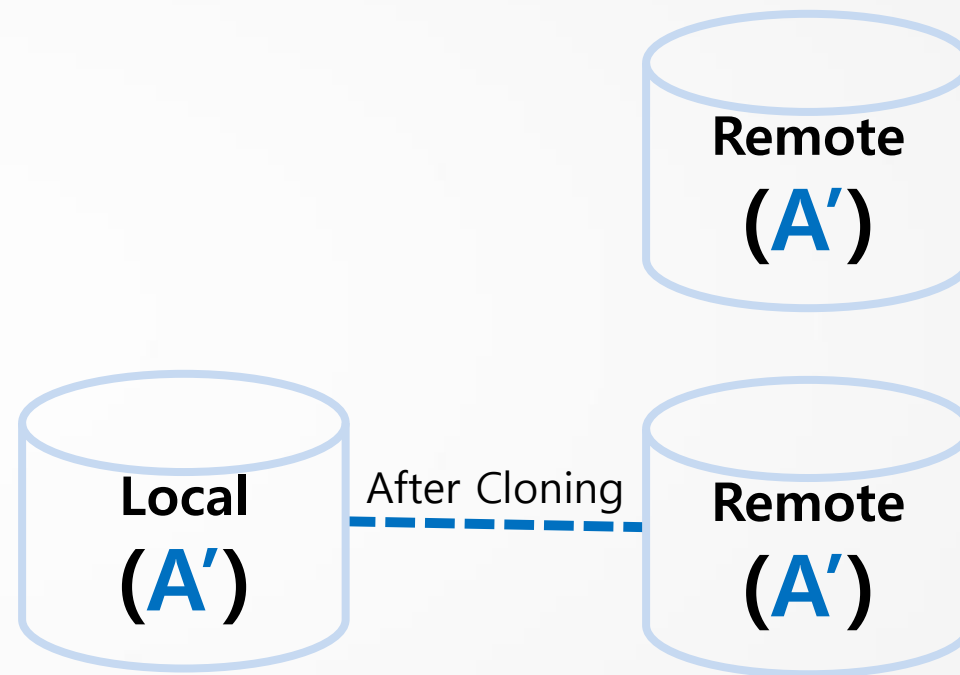
• CLI를 이용한 Clone(복제)

- 원격저장소 지정하기
 - › 협업자로 등록된 원격저장소 혹은 Public 원격저장소
- 지역저장소 생성하기
- 지역저장소와 복제하기

```
$ git remote add origin 원격저장소-URL  
$ git clone 원격저장소-URL // 지역저장소 생성 및 복제
```

• IDEA를 통한 Clone (주로 사용)

- (Visual Studio Code, IntelliJ)

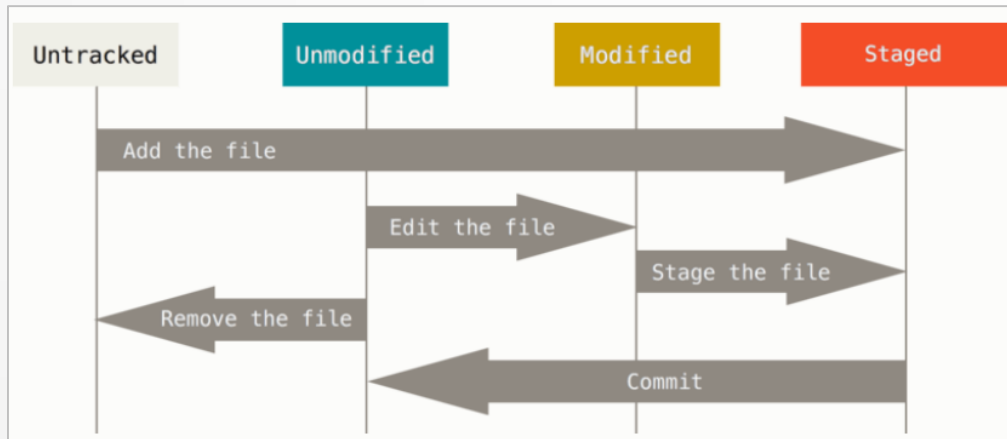


File Status in 'Local'

• File Status

- *tracked*: Git에서 Version이 관리 중인상태 (형상관리시스템에 화일이 존재)
- *untracked*: 한 번도 Version을 만들지 않은 상태
- *unmodified*: commit 후 수정 없는 상태
- *modified*: 수정한 후 아직 Stage에 저장하지 않은 상태
- *staged*: Stage에 있고 아직 commit 하지 않은 상태

• Life Cycle

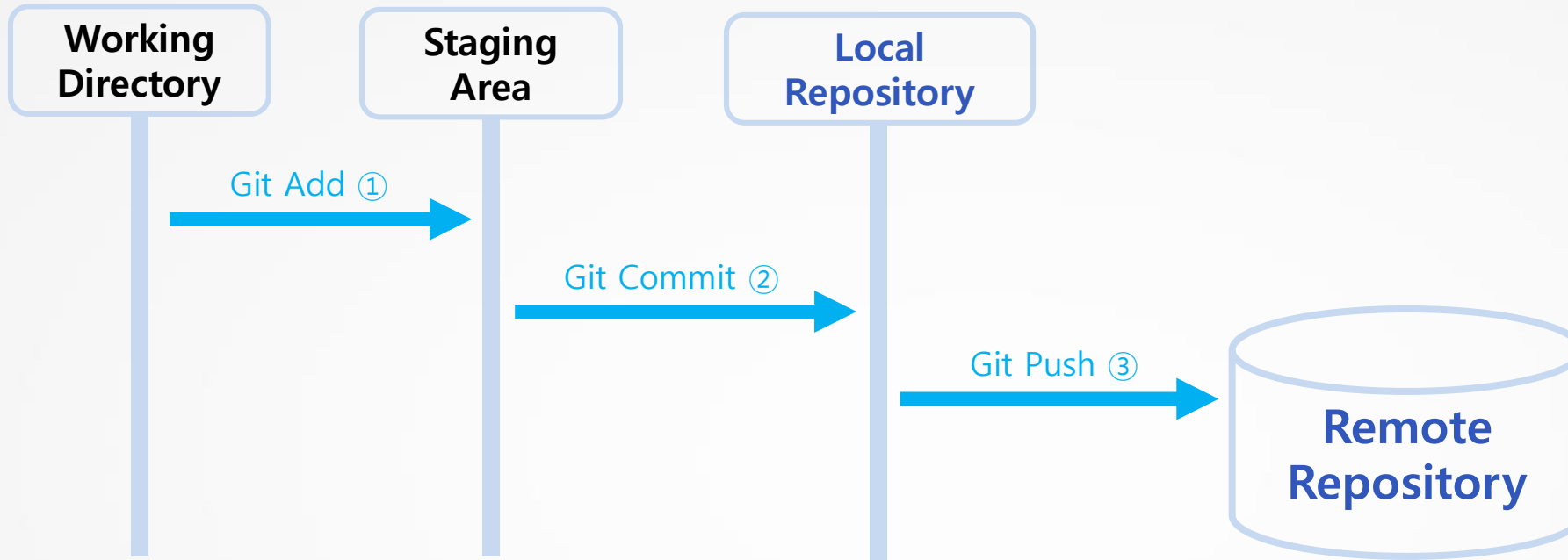


* <https://git-scm.com/book/en/v2/>

※ stash 영역

- 아직 add/commit 되지 않은 내용(unstaged)을 저장하는 임시 클립보드

파일 등록 Flow (Local -> Remote)



※ 매뉴얼

- Quick Guide : <https://github.github.com/training-kit/downloads/kr/github-git-cheat-sheet/>
- Tutorial : <https://git-scm.com/docs/gittutorial>
- Git workflow : <https://git-scm.com/docs/gitworkflows>
- 기타 문헌 : <https://git-scm.com/book/en/v2>

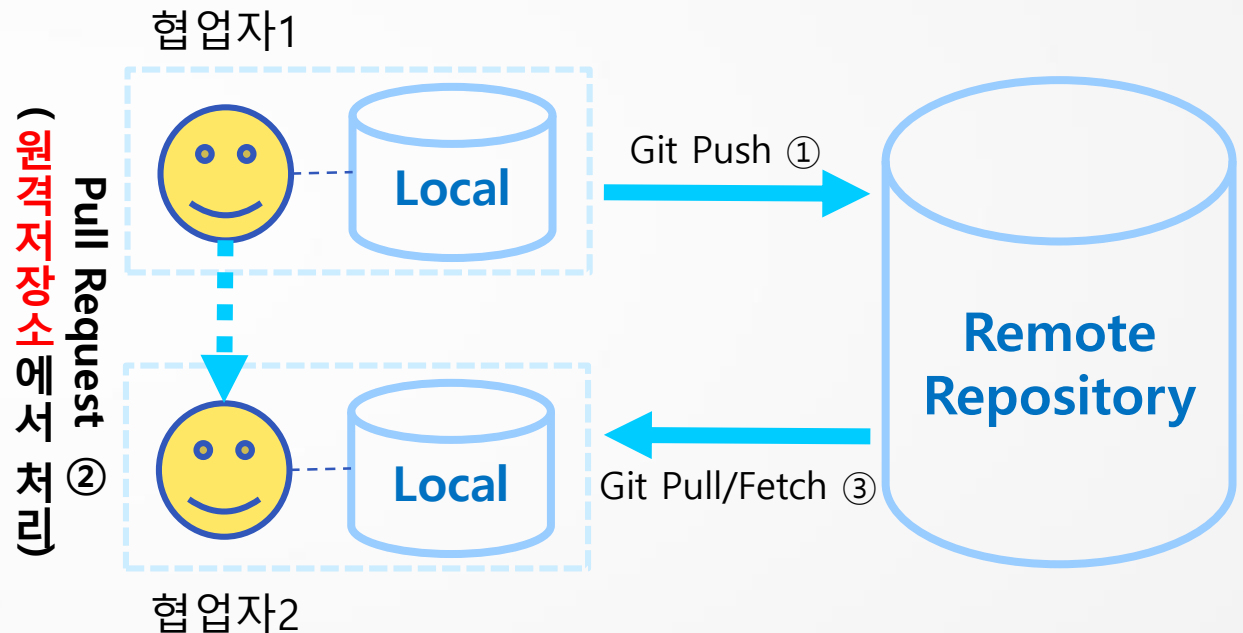
Git에 의한 협업 (1/2)

• 같은 원격저장소 상에서의 협업

- 브랜치 기반 개발을 통한 협업 (Git Workflow)

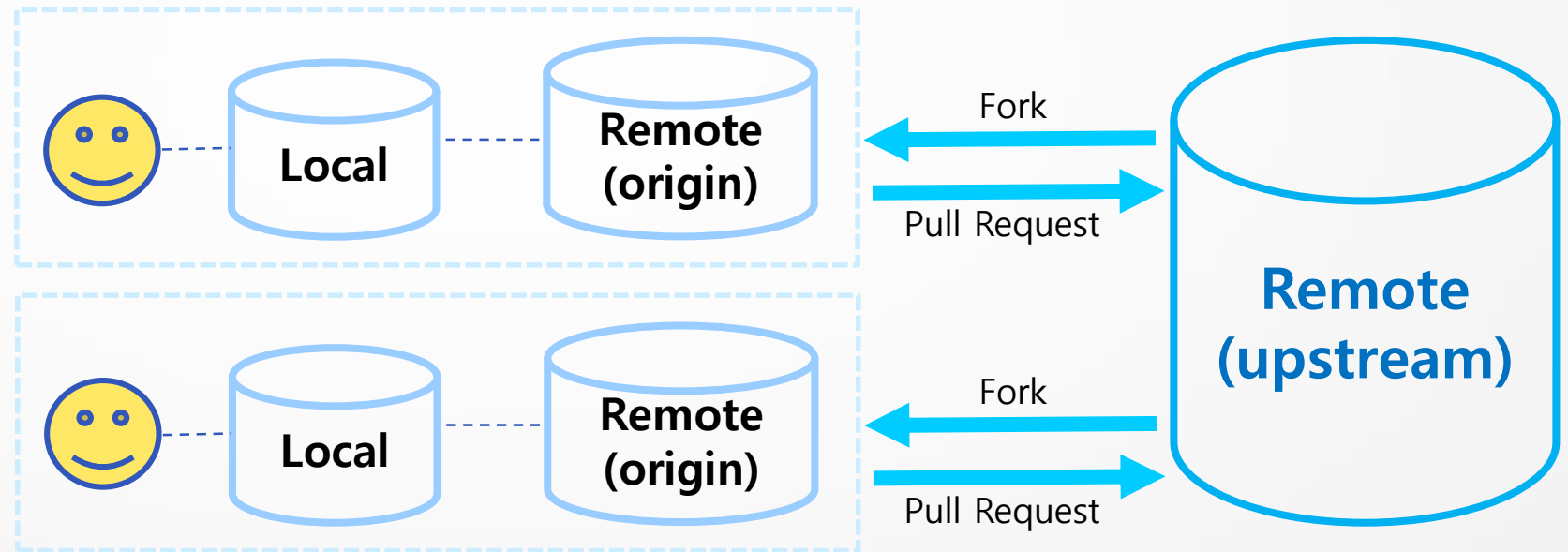
[원격저장소]

- 협업자 등록필요
- (협업자1 코드 수정 후)
 Pull Request를 통한 협업자 간 Communication
 (**Code Review** -> Reply -> ...)
- Communication 후, **Merge PR** 실행
 - (브랜치 merge)
- 다른 협업자들의 지역저장소 Refresh(**Pull**)



• 오픈 소스 개발모델

- 협업자로 등록되지 않은 경우도, 협업 가능
- 원격의 원격저장소(target 원격저장소)
- 원격의 원격저장소 <-> 원격저장소 (**fork**를 통한 복제)
- 원격저장소 <-> 지역저장소 (**clone**을 통한 복제)



***참고) Trouble Shooting**

Best Reviewer 양성과정

git clone 시 인증 에러가 발생합니다

- local clone 을 위한 토큰 생성 확인

https://github.ecodesamsung.com/pages/Service/guide/page/ko/user_process/4_clone

- 제어판>사용자 계정>자격 증명 관리자 에서
github.ecodesamsung.com 과 관련한 항목을 모두 지우고
다시 실행 해보시기 바랍니다


Windows 자격증명 삭제(GitHub 관련)


- 1 PC에서 GitHub관련 다른 자격증명이 있는 경우, 삭제 필요
(혹은 자격증명에 계정/암호[토큰]를 잘못 입력한 경우, 편집 또는 삭제 필요)
- 기존의 자격 증명서(GitHub 관련 부분) 제거 ([검색-자격증명 관리자])

제어판 홈

자격 증명 관리

웹 사이트, 연결된 응용 프로그램 및 네트워크에 대해 저장된 로그인 정보를 보고 삭제합니다.

 웹 자격 증명

 Windows 자격 증명

[자격 증명 백업\(B\)](#) [자격 증명 복원\(R\)](#)

일반 자격 증명

일반 자격 증명 추가

GitHub - https://[redacted]	수정한 날짜: 2020-05-26	▼
git:https://[redacted]	수정한 날짜: 2020-06-17	▼
MicrosoftAccount:user=[redacted]	수정한 날짜: 오늘	▼
virtualapp/didlogical	수정한 날짜: 2020-06-27	▼

Could not resolve proxy 오류

- 오류 메시지

Unable to access 'https://github. ecodesamsung.com/...': Could not resolve proxy:....

- 해결책 : 사업장의 Proxy 확인

- 참고 :

<http://mosaic.sec.samsung.net/kms/mosaicLayout.do?method=link&type=question&id=12037289229#93B302>

SSL certificate problem 오류

- 오류 메시지

SSL certificate problem: unable to get local issuer certificate

- 해결책 : **git configuration 수정**

- git config --global http.sslVerify false
- git clone <https://github.ecodesamsung.com/Best-Reviewer-3-12/repository>
- git config --global http.sslVerify true

- ❖ git config 파일을 열어 직접 수정하여도 동일함.
 - ❖ /사용자/{사용자명}/.gitconfig 파일 open 후 추가 및 수정

```
[http]
    sslVerify = false
```

3. Sample cmake project (hello world)

Best Reviewer 양성과정

Create main.cpp

```
#include <iostream>

using namespace std;

int main(int argc, char **argv) {
    cout << "Hello World" << endl;
}
```

Compile and run by Cygwin terminal

- 첫번째 수준

```
CORP+andy.shlee@andy-shlee01 /cygdrive/d/test/hello2  
$ c++ main.cpp  
  
CORP+andy.shlee@andy-shlee01 /cygdrive/d/test/hello2  
$ ./a.exe  
Hello world
```

Create CMakeLists.txt in same folder

```
cmake_minimum_required(VERSION 3.0.1)
set(PROJECT_NAME hello)
project(${PROJECT_NAME})

set(SRCS
    main.cpp
)

add_executable(${PROJECT_NAME}
    ${SRCS}
)
```

Compile and run by Cygwin terminal

```
CORP+andy.shlee@andy-shlee01 /cygdrive/d/test/hello2
$ mkdir build

CORP+andy.shlee@andy-shlee01 /cygdrive/d/test/hello2
$ cd build

CORP+andy.shlee@andy-shlee01 /cygdrive/d/test/hello2/build
$ cmake ..
-- The C compiler identification is GNU 7.4.0
:
-- Build files have been written to: /cygdrive/d/test/hello2/build






CORP+andy.shlee@andy-shlee01 /cygdrive/d/test/hello2/build
$ make
Scanning dependencies of target hello
[ 50%] Building CXX object CMakeFiles/hello.dir/hello.cpp.o
[100%] Linking CXX executable hello.exe
[100%] Built target hello

CORP+andy.shlee@andy-shlee01 /cygdrive/d/test/hello2/build
$ ./hello.exe
Hello world
```

4. Sample google test project (gtest_hello 콘솔 작성)

Best Reviewer 양성과정

폴더 구성

 src	2020-10-05 오후 4:28	파일 폴더	
 test	2020-10-05 오후 4:28	파일 폴더	
 .gitignore	2020-10-05 오후 4:28	텍스트 문서	1KB
 CMakeLists.txt	2020-10-08 오후 3:35	텍스트 문서	1KB
 README.md	2020-10-07 오후 2:46	Markdown 원본 ...	1KB

```
CMAKE_MINIMUM_REQUIRED(VERSION 3.8)
```

```
SET(PROJECT hello)
```

```
# C++ settings
```

```
PROJECT(${PROJECT})
```

```
SET(CMAKE_CXX_STANDARD 11)
```

```
enable_testing()
```

```
add_subdirectory(src)
```

```
add_subdirectory(test)
```

```
add_library(hello STATIC hello.cc hello.h)
```

```
#ifndef __HELLO_H_  
#define __HELLO_H_  
const char* sayHello();  
#endif
```

```
const char* sayHello()  
{  
    return "hello";  
}
```

```
cmake_minimum_required(VERSION 3.8)
set(PROJECT hello_test)
set(SOURCE
    hello_test.cc
)
add_executable(${PROJECT} ${SOURCE})
target_link_libraries(${PROJECT}
    PUBLIC
        gtest
        gtest_main
        hello
)
add_test(NAME ${PROJECT} COMMAND ${PROJECT})
```

```
#include "../src/hello.h"
#include <gtest/gtest.h>

TEST(TestA, test1){
    EXPECT_EQ(0,0);
}

TEST(TestA, test2){
    EXPECT_EQ(true, true);
}

TEST(helloTest, test3){
    EXPECT_STREQ("hello", sayHello());
}
```

build by Cygwin terminal

```
noogi@noogi-PC ~/gtest_hello
```

```
$ ls
```

```
CMakeLists.txt  README.md  src  test
```

```
noogi@noogi-PC ~/gtest_hello
```

```
$ mkdir build
```

```
noogi@noogi-PC ~/gtest_hello
```

```
$ cd build
```

build by Cygwin terminal

```
noogi@noogi-PC ~/gtest_hello/build
```

```
$ cmake ..
```

```
-- The C compiler identification is GNU 10.2.0
```

```
:
```

```
-- Build files have been written to: /home/noogi/gtest_hello/build
```

```
noogi@noogi-PC ~/gtest_hello/build
```

```
noogi@noogi-PC ~/gtest_hello/build
```

```
$ make
```

```
Scanning dependencies of target hello
```

```
[ 25%] Building CXX object src/CMakeFiles/hello.dir/hello.cc.o
```

```
[ 50%] Linking CXX static library libhello.a
```

```
[ 50%] Built target hello
```

```
Scanning dependencies of target hello_test
```

```
[ 75%] Building CXX object test/CMakeFiles/hello_test.dir/hello_test.cc.o
```

```
[100%] Linking CXX executable hello_test.exe
```

```
[100%] Built target hello_test
```


run by Cygwin terminal

```
noogi@noogi-PC ~/gtest_hello/build
$ cd test
noogi@noogi-PC ~/gtest_hello/build/test
$ ls
cmake_install.cmake  CMakeFiles  CTestTestfile.cmake  hello_test.exe  Makefile
noogi@noogi-PC ~/gtest_hello/build/test
$ ./hello_test.exe
Running main() from /home/noogi/googletest-master/googletest/src/gtest_main.cc
[=====] Running 3 tests from 2 test suites.
[-----] Global test environment set-up.
[-----] 2 tests from TestA
[ RUN    ] TestA.test1
[      OK ] TestA.test1 (0 ms)
[ RUN    ] TestA.test2
[      OK ] TestA.test2 (0 ms)
[-----] 2 tests from TestA (1 ms total)

...
[-----] Global test environment tear-down
[=====] 3 tests from 2 test suites ran. (6 ms total)
[ PASSED ] 3 tests.
```

Q&A