

프로그래밍 역량 강화 전문기관, 민코딩

Git with Sourcetree



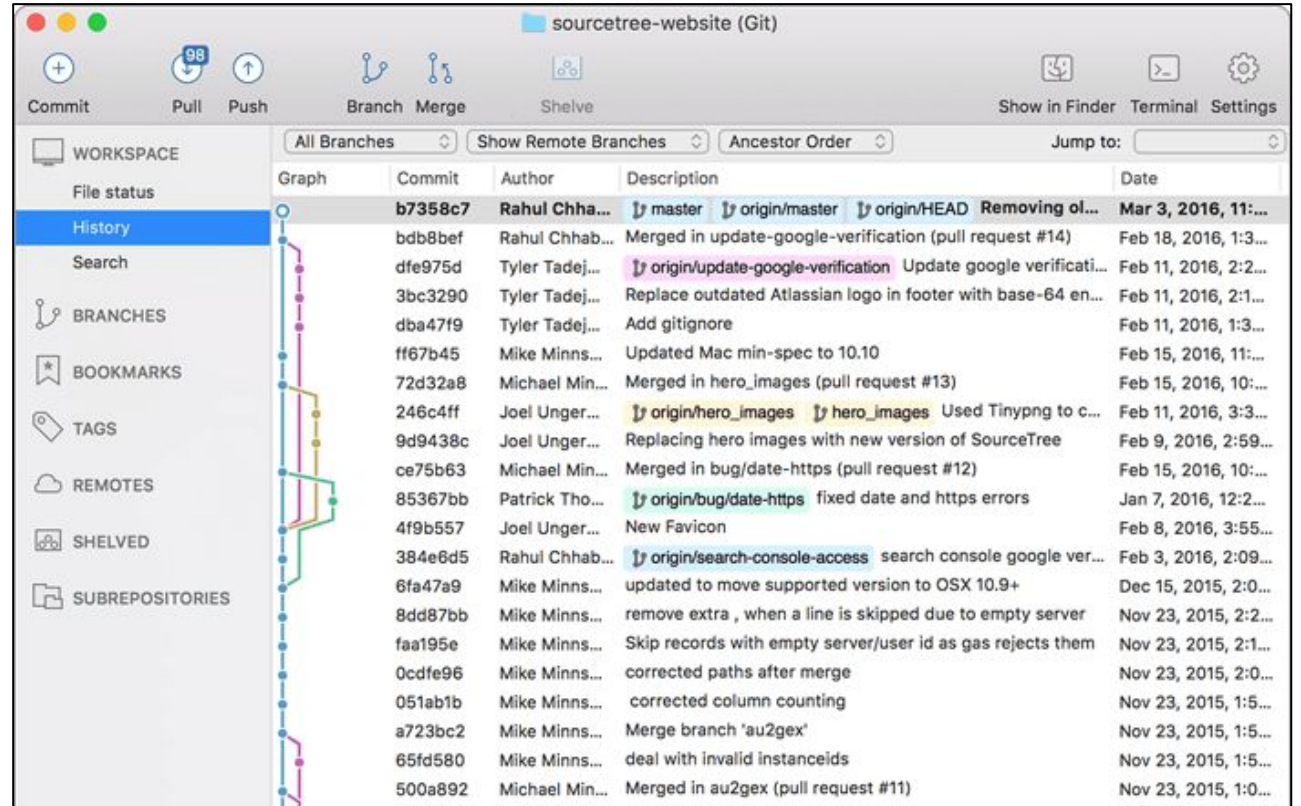
목차

1. Sourcetree 설치와 github 연결
2. 한글 깨짐 현상 해결하기
3. Clone
4. Visual Studio 에서 Git CLI 사용

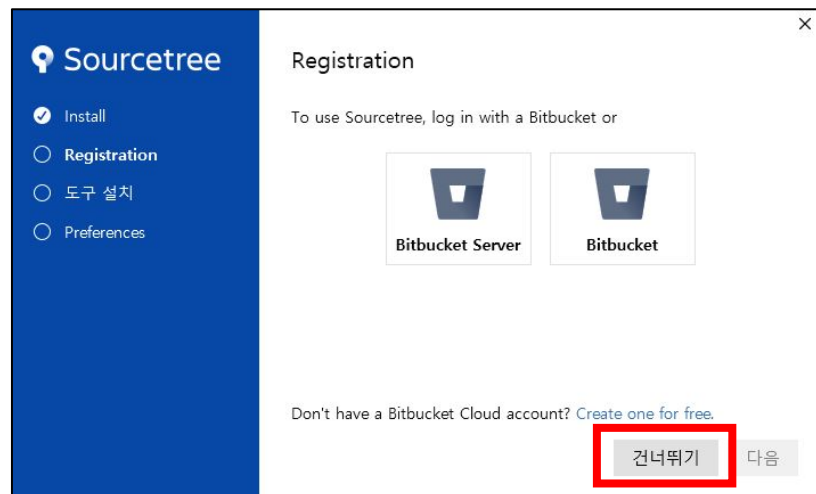
Sourcetree 설치와 github 연결

아틀란시안에서 만든 상용도 무료인 Git Client

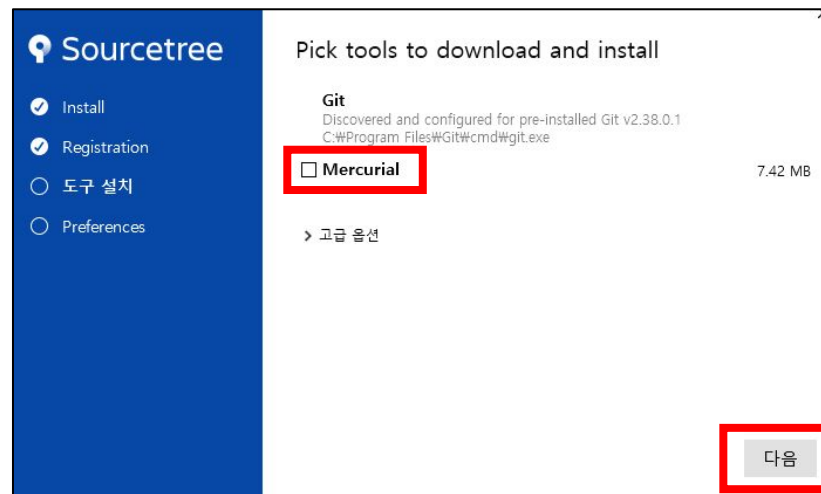
GUI 기반으로,
Repository 시각화 가능



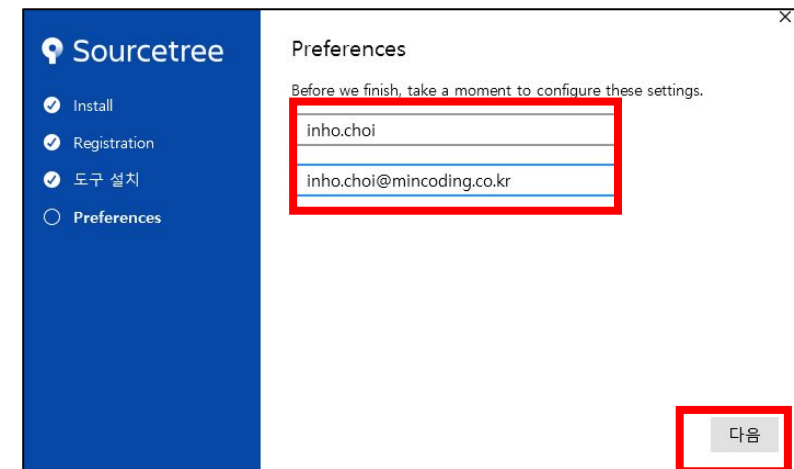
설치



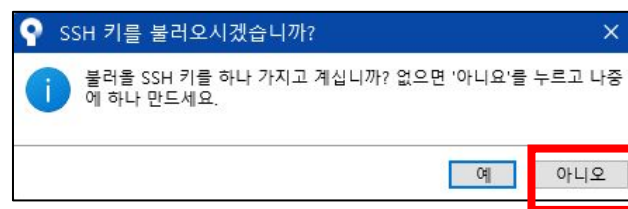
비트버켓을 안쓰고,
Github을 쓰므로, 건너뛰기



머큐리얼 안쓰고,
Git을 쓰므로, 체크 해제 후 다음



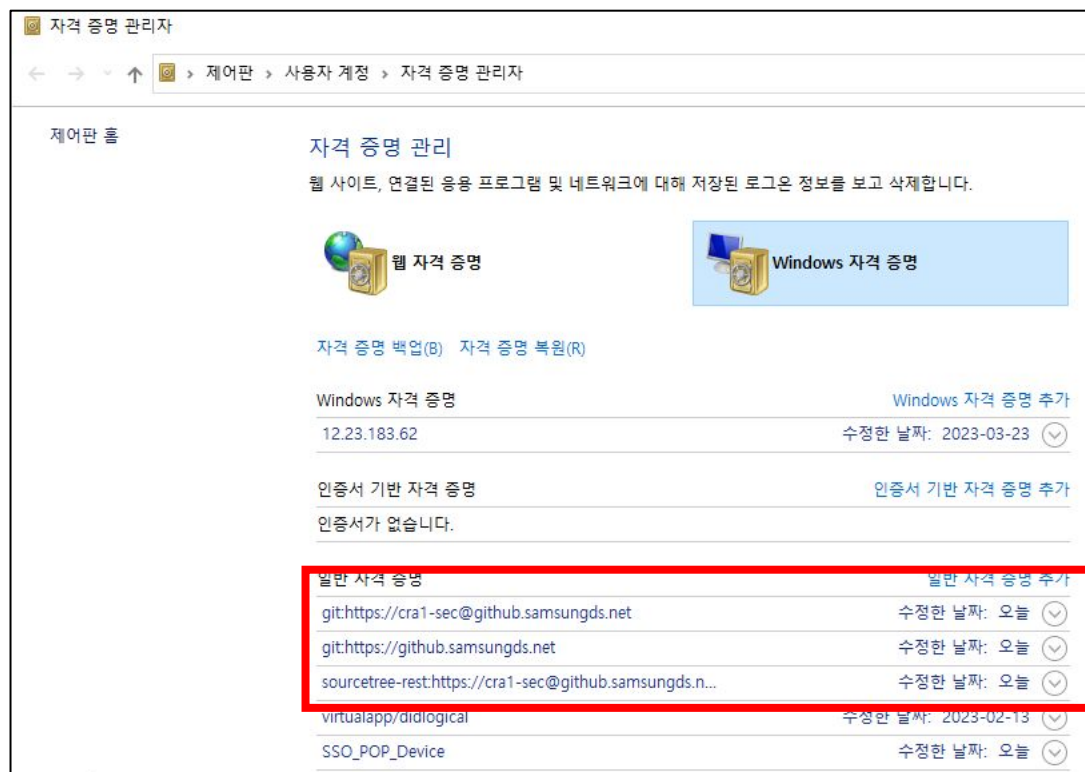
적절한 이름과 이메일 입력



SSH 키 안씀, 아니오 누름

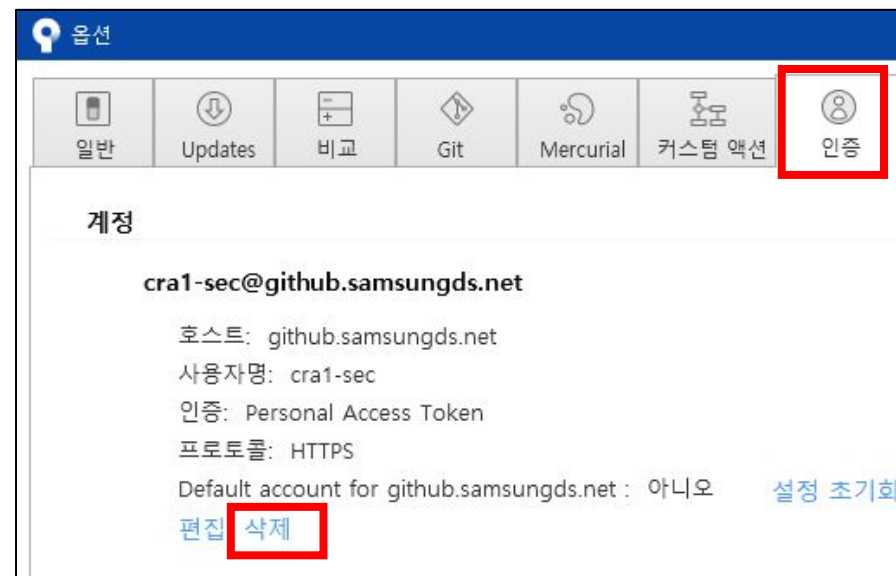
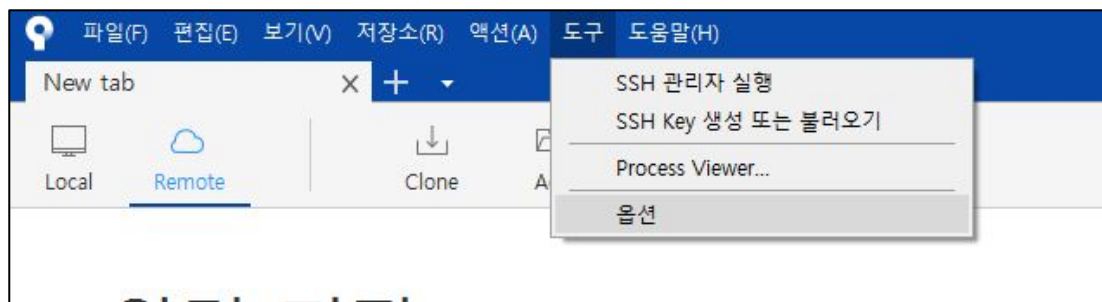
로그인 전 확인 1

✓ 시작 > 자격증명 > Windows 자격 증명 > git / sourcetree 관련 모두 삭제



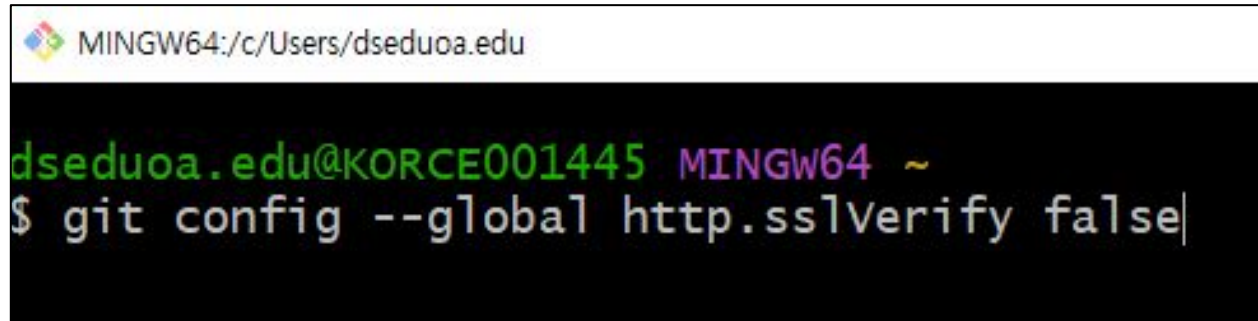
로그인 전 확인 2

✓ 이전 컴퓨터 사용자의 로그인 계정 기록 삭제



로그인 전 확인 3

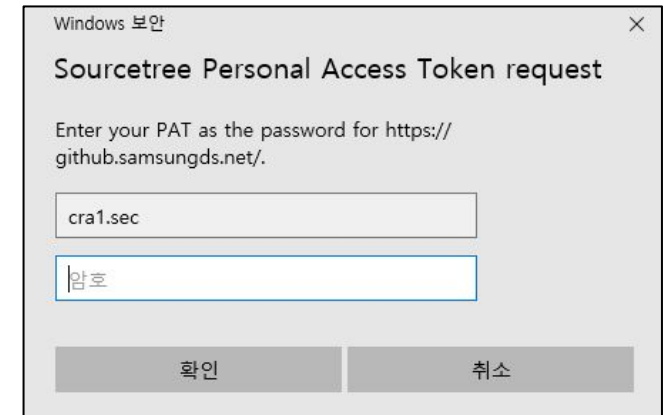
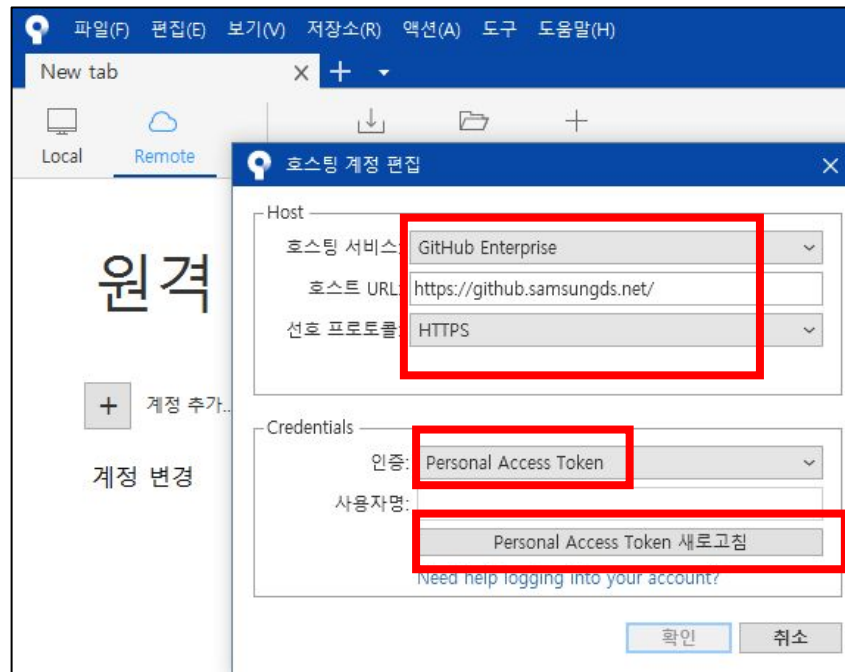
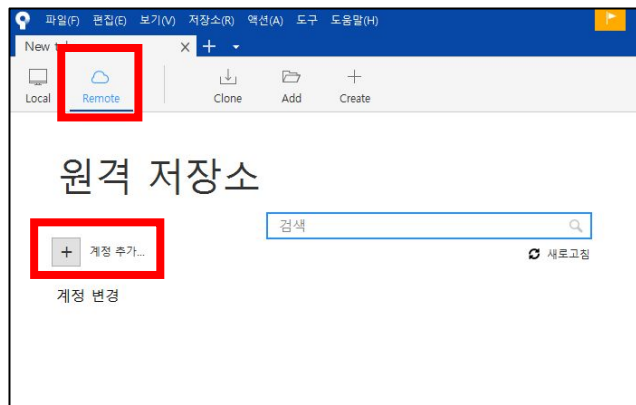
✓ git bash 실행 후, 다음 옵션 false 처리



```
MINGW64:/c/Users/dseduoa.edu  
dseduoa.edu@KORCE001445 MINGW64 ~  
$ git config --global http.sslVerify false|
```


Github과 연결하기 1

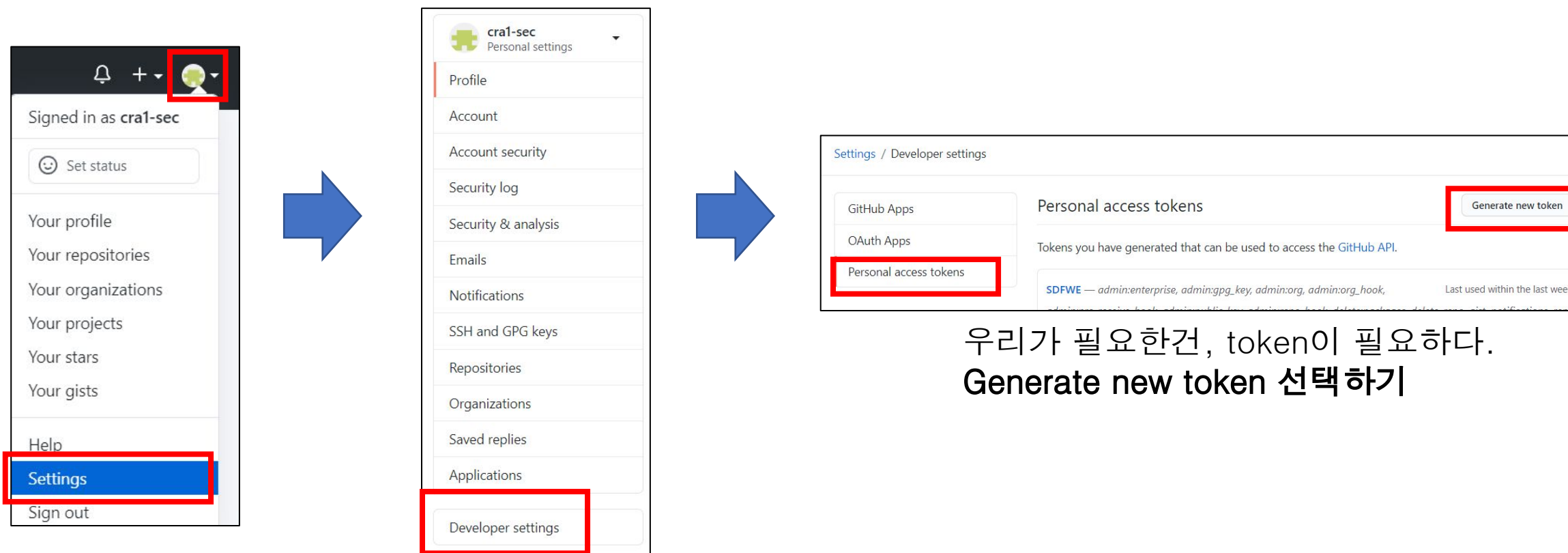
✓ Remote > 계정 추가하기



ID와 Password 가 아니다.
Personal Access Token을
발급받아서 입력해야한다.

Github과 연결하기 2

✓ github에 접속하고, Settings > Developer setting 클릭



우리가 필요한건, token이 필요하다.
Generate new token 선택하기

Github과 연결하기 3

New personal access token

Personal access tokens function like ordinary OAuth access tokens, but can be used to [authenticate to the API over Basic](#) or [HTTPS](#), or can be used to [authenticate to the API over Basic](#) or [HTTPS](#).

Note

KFC code reviewer

What's this token for?

Select scopes

Scopes define the access for personal tokens. [Read more about 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 |

토큰 설명 입력

전부 체크

| | |
|---|----------------------------|
| <input checked="" type="checkbox"/> admin:enterprise | Full control of enterprise |
| <input checked="" type="checkbox"/> manage_billing:enterprise | Read and write enterprise |
| <input checked="" type="checkbox"/> read:enterprise | Read enterprise profile |
| <input checked="" type="checkbox"/> admin:pre_receive_hook | Control enforcement |
| <input checked="" type="checkbox"/> admin:pgp_key | Full control of public |
| <input checked="" type="checkbox"/> write:pgp_key | Write public user pgp |
| <input checked="" type="checkbox"/> read:pgp_key | Read public user pgp |

Generate token Cancel



Settings / Developer settings

Personal access tokens

Tokens you have generated that can be used to access the GitHub API

Make sure to copy your new personal access token now. You will not be able to see it again.

af796a0d25c91e08472be36d0e702c5d57679cc

토큰이 생성되었다.
토큰을 복사한다.



Windows 보안

Sourcetree Personal Access Token request

Enter your PAT as the password for https://github.samsungds.net/.

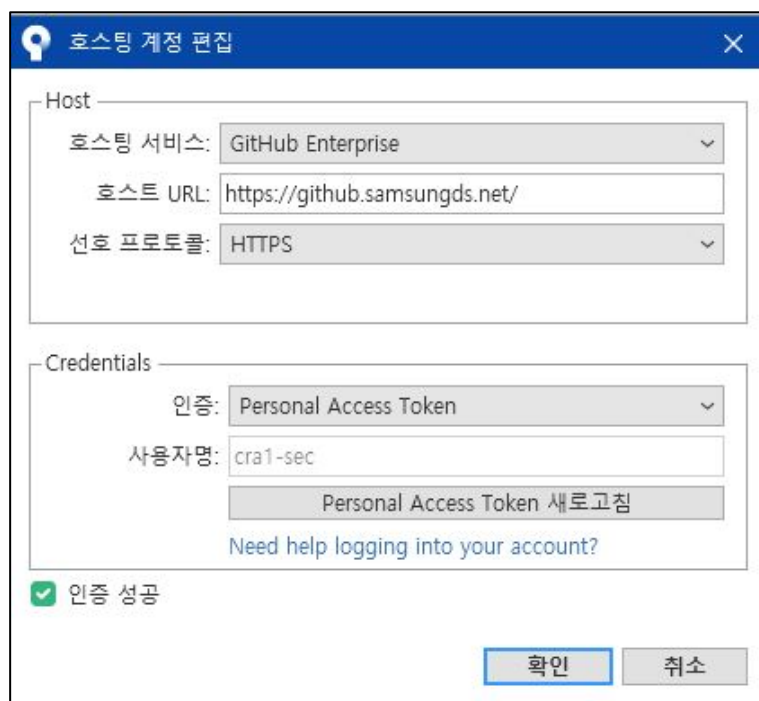
cra1.sec

.....

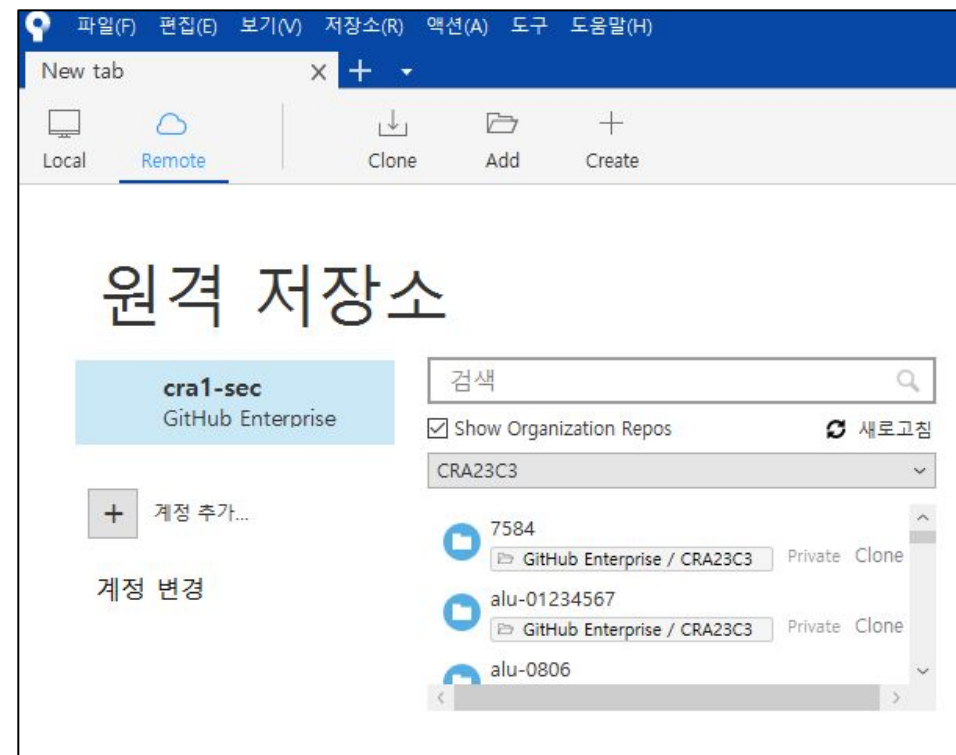
확인 취소

Personal Access Token
기입후 확인버튼

Github과 연결하기 4



인증 성공 확인 후,
확인 버튼



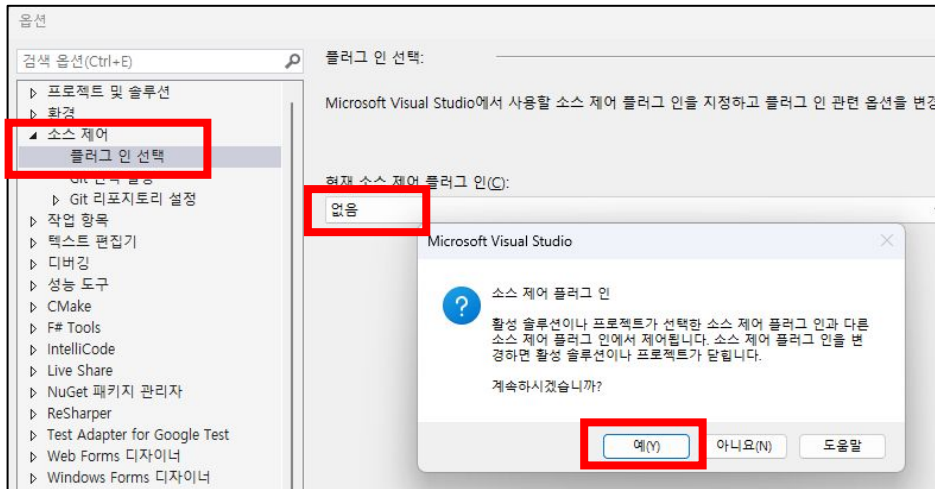
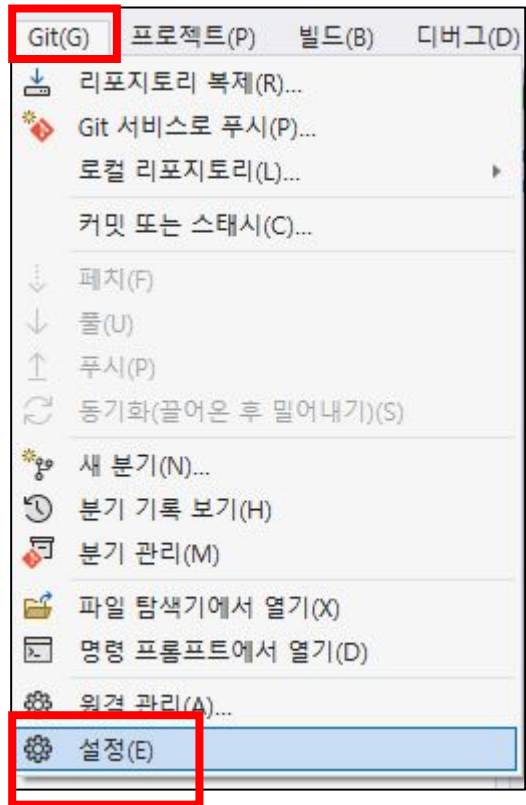
다음과 같이 뜨면
접속 완료이다.

Visual Studio

프로젝트 생성 부터, Push 까지

Visual Studio git 설정 끄기

✓ Visual Studio 내장 Git GUI 를 사용하지 않는다는 옵션.



Git 메뉴가 사라짐

Production 프로젝트 생성 + Google Test 프로젝트

✓ 실습을 위한 기본 코드 작성

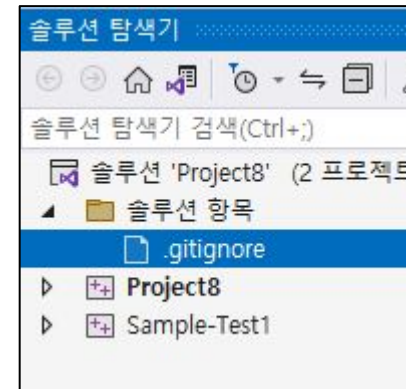
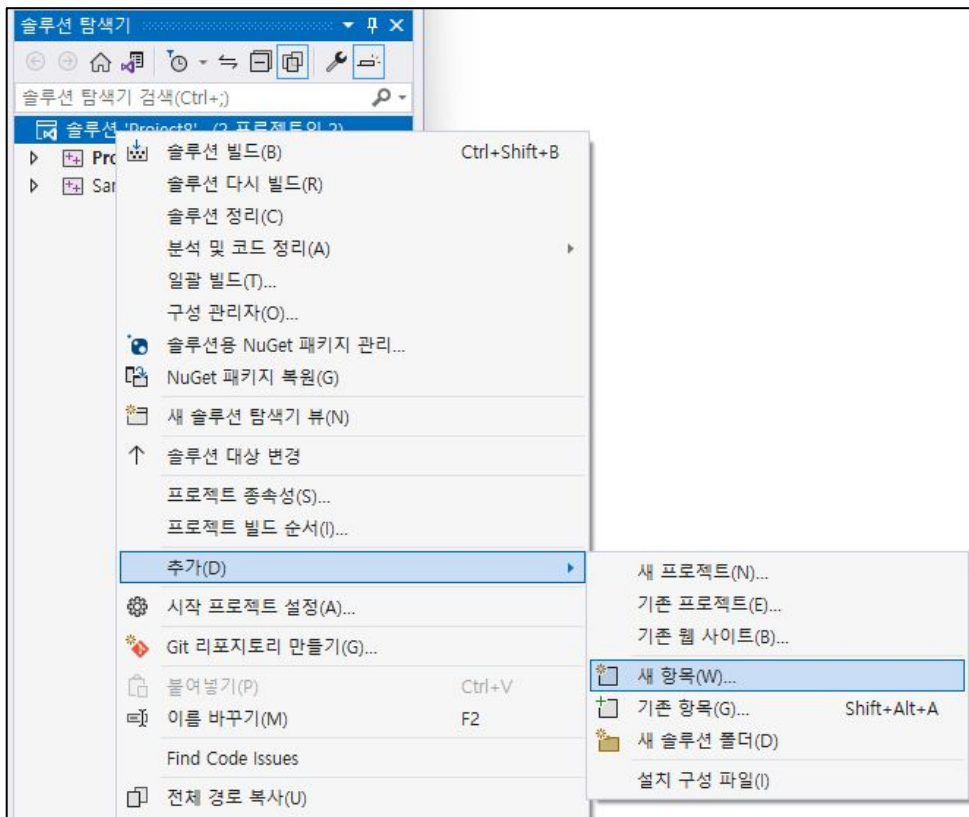


```
1 int main(int argc, char* argv[]) {
2
3 }
4

1 #include "pch.h"
2
3 TEST(TestCaseName, TestName) {
4     EXPECT_EQ(1, 1);
5     EXPECT_TRUE(true);
6 }
```

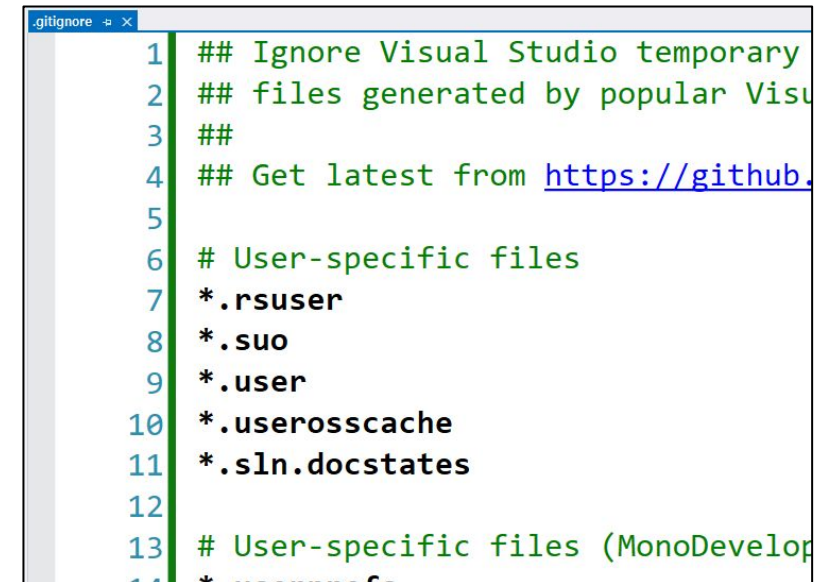
.gitignore 파일

- ✓ Visual Studio 환경설정 파일, 임시 파일들을 Push 할 필요 없다.
- ✓ Git이 무시해야할 파일들을 .gitignore에 기입한다.



.gitignore 템플릿 사용

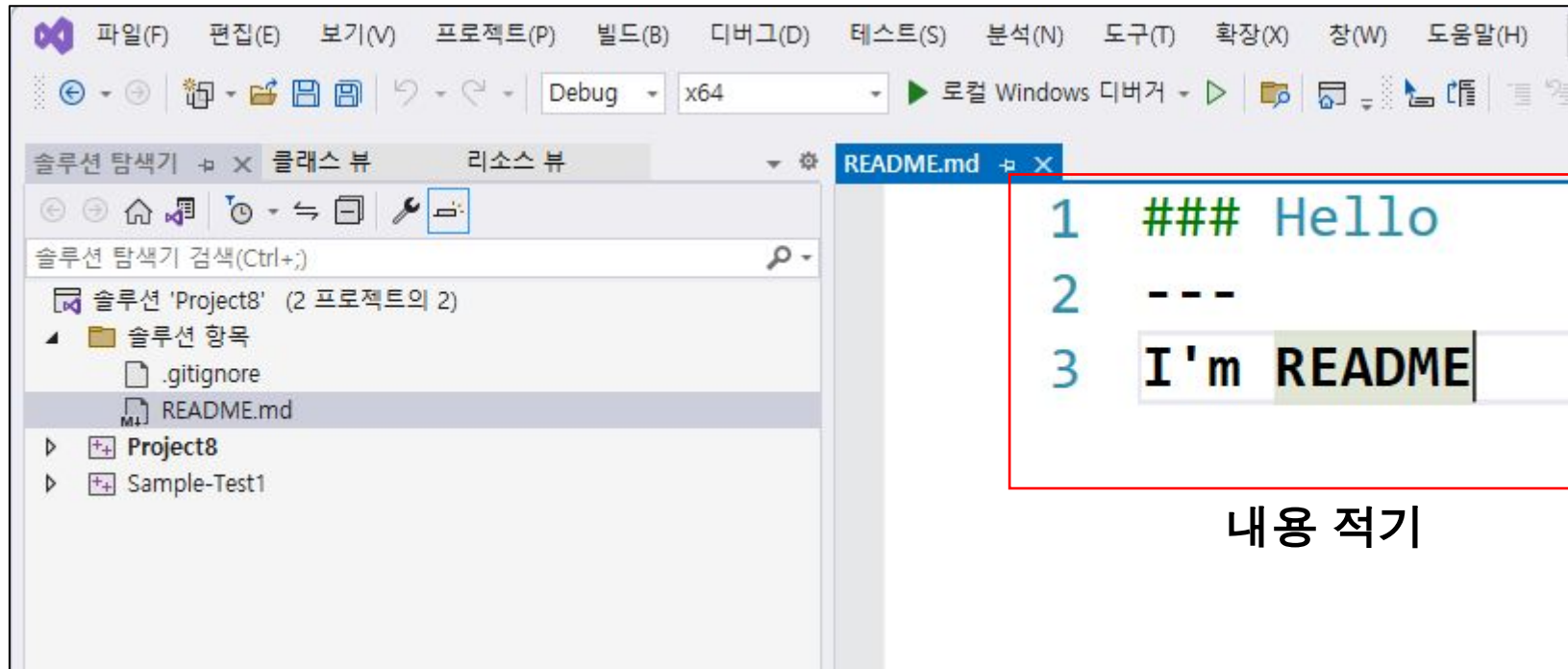
- ✓ Visual Studio 권장 .gitignore 파일 set
 - Github 공식 gitignore 템플릿 모음
 - <https://github.com/github/gitignore>
 - 위 링크에서 **VisualStudio.gitignore** 파일을 찾는다.
 - 전체 내용을 복사하여, 생성한 .gitignore에 붙여넣기



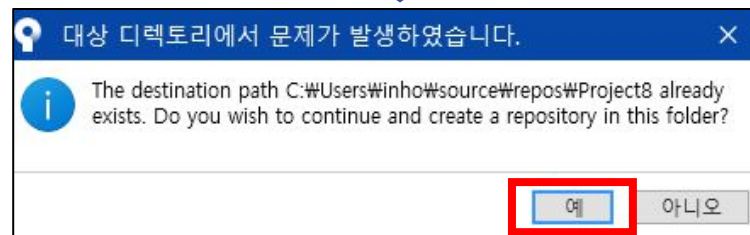
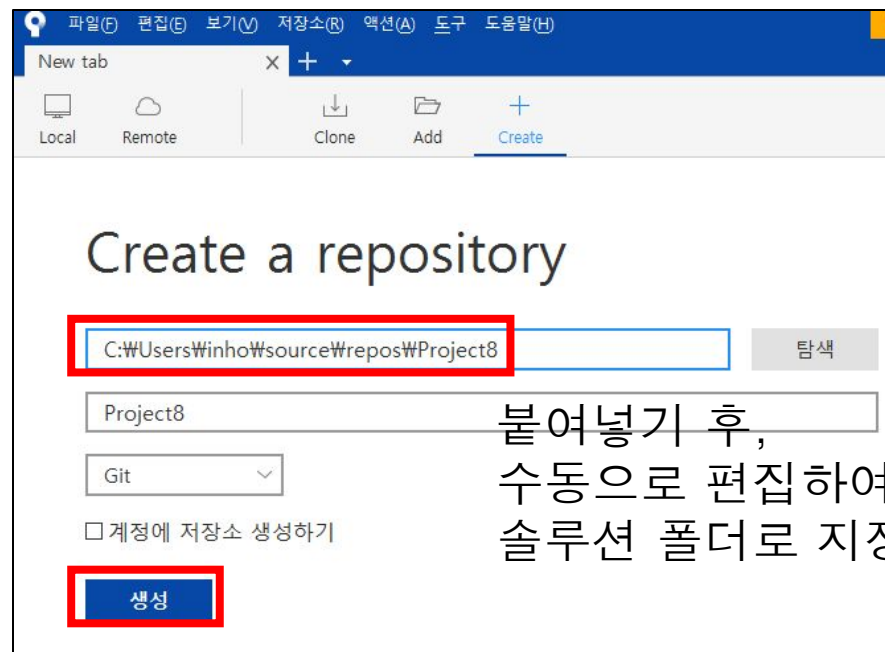
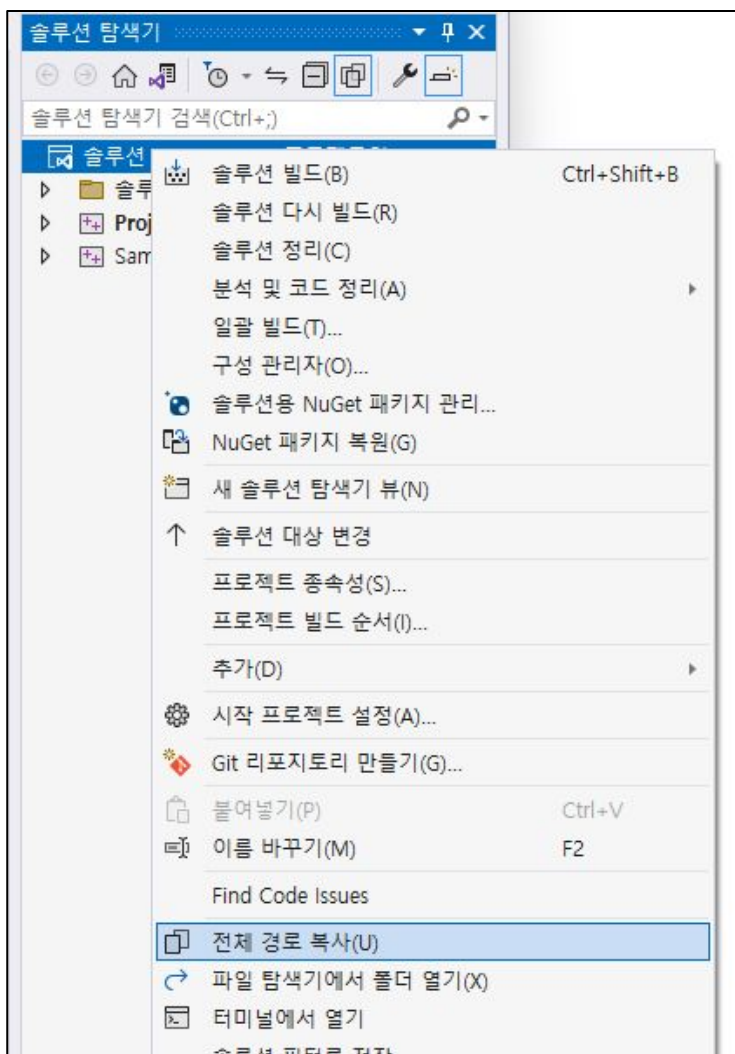
```
.gitignore
1  ## Ignore Visual Studio temporary
2  ## files generated by popular Visu
3  ##
4  ## Get latest from https://github.com
5
6  # User-specific files
7  *.rsuser
8  *.suo
9  *.user
10 *.userosscache
11 *.sln.docstates
12
13 # User-specific files (MonoDevelop
14 *.userprefs
```

README.md 파일 생성

✓ 웰컴페이지에 등장할 README 파일 추가하기



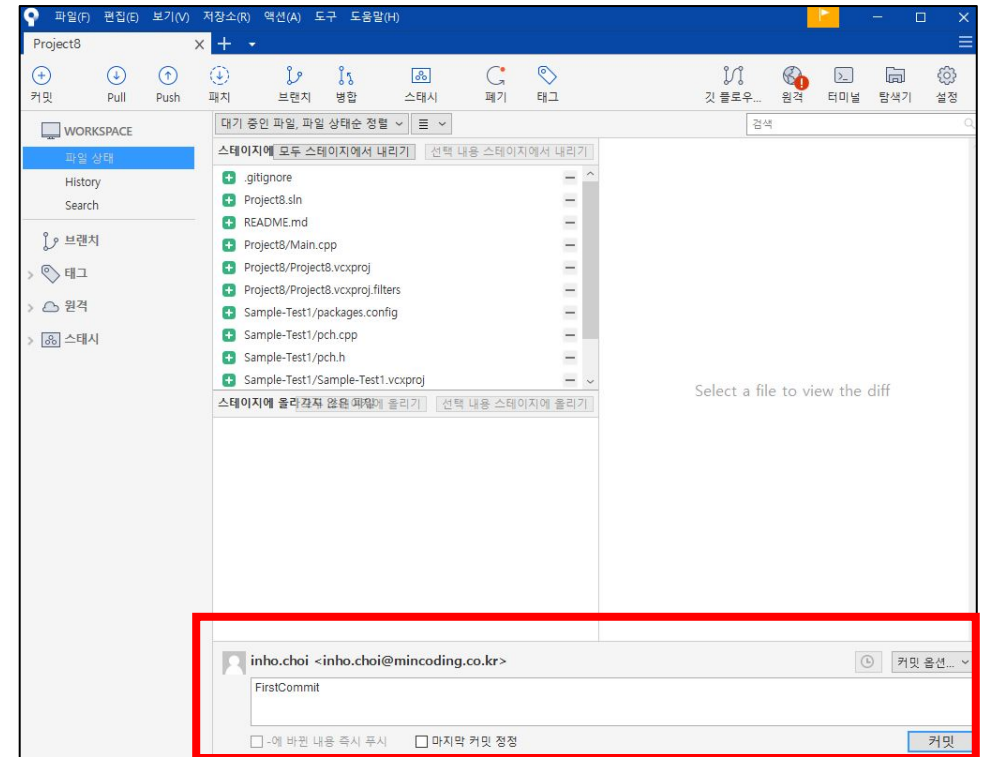
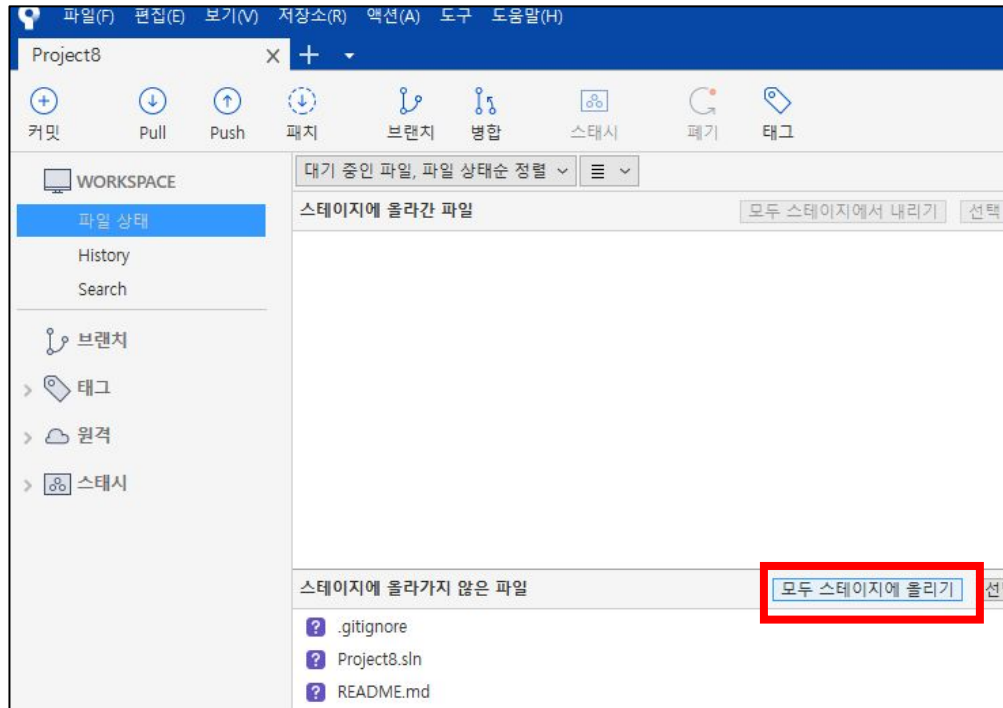
GUI 도구로 **git init** 수행하기



이미 존재하는 폴더 사용? OK

모두 Staging 후 Commit

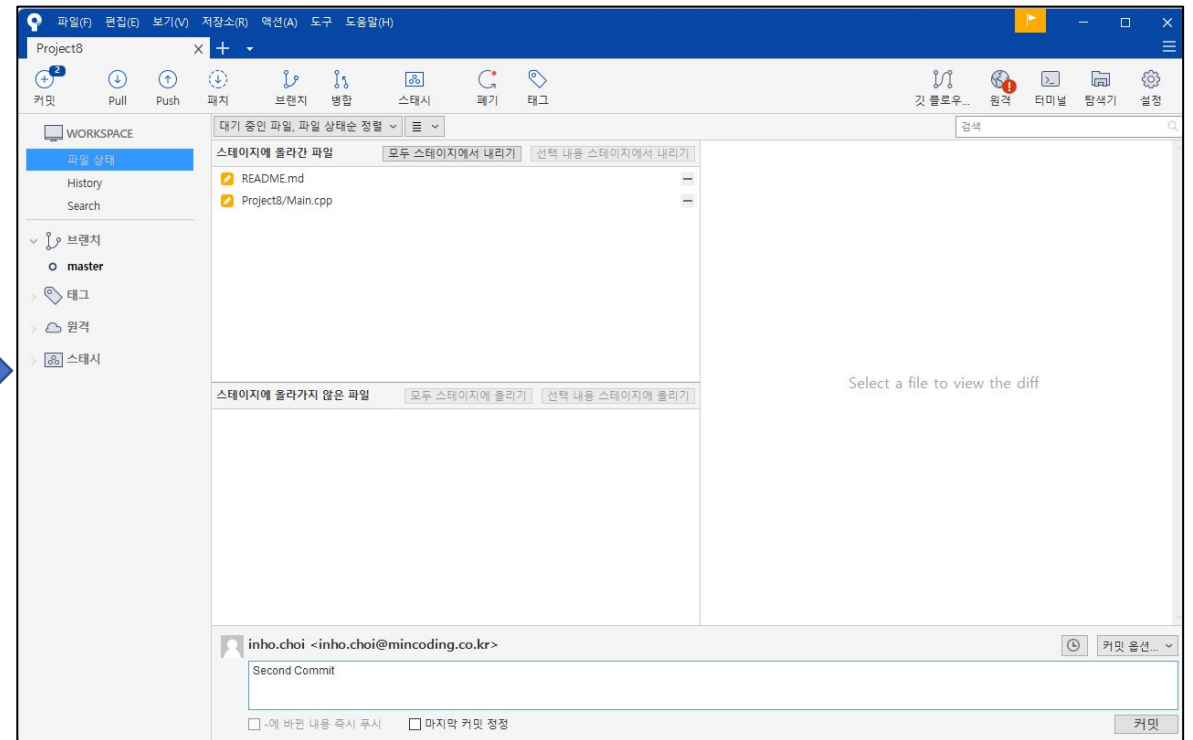
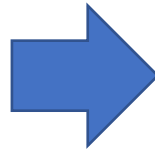
✓ 첫 번째 커밋을 진행한다.



두 번째 Commit 진행

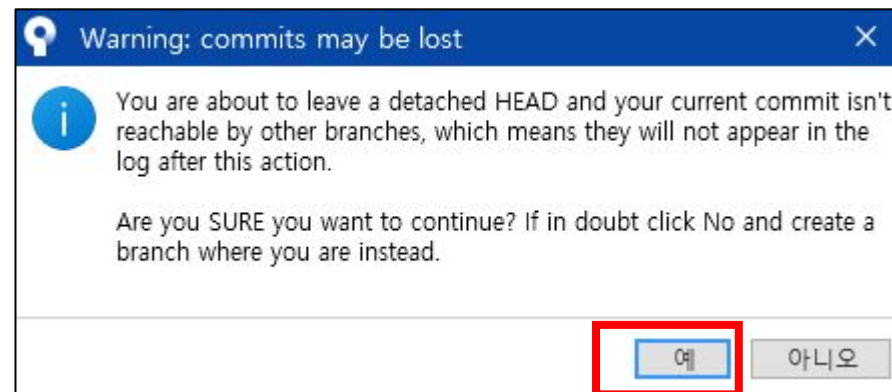
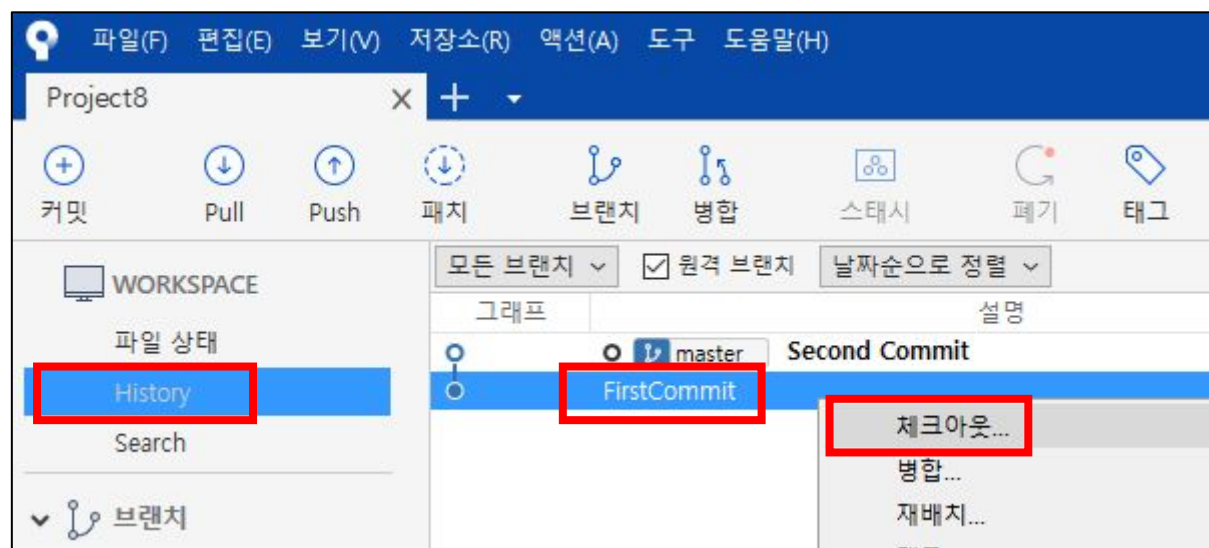
✓ main 코드 일부 수정 후, Commit

```
Main.cpp x
Project8
1 #include <iostream>
2 int main(int argc, char* argv[]) {
3     std::cout << "HI";
4 }
5
```



체크아웃 테스트

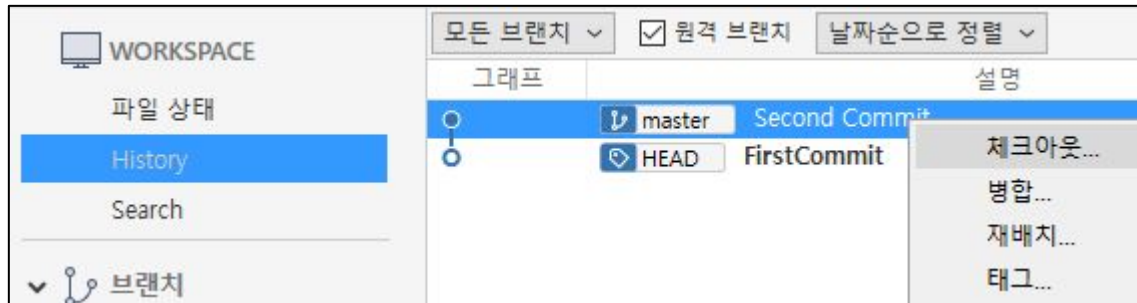
History > First Commit 체크아웃하고, 소스코드 반영 결과 확인



```
int main(int argc, char* argv[]) {  
  
}
```

체크아웃 테스트

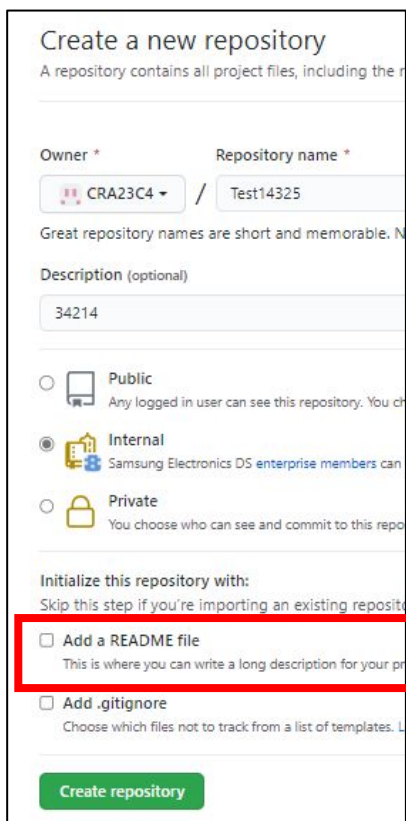
다시 최근 Commit으로 Checkout하기



```
#include <iostream>
int main(int argc, char* argv[]) {
    std::cout << "HI";
}
```

push 준비하기

✓ remote repo 추가하기



Create a new repository

A repository contains all project files, including the source code.

Owner * CRA23C4 / Repository name * Test14325

Great repository names are short and memorable. No spaces, please.

Description (optional)

34214

☐ Public
Any logged in user can see this repository. You can't restrict access.

☒ Internal
Samsung Electronics DS enterprise members can see this repository.

☐ Private
You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ Add a README file
This is where you can write a long description for your project.

☐ Add .gitignore
Choose which files not to track from a list of templates.

Create repository



Quick setup — if you've done this kind of thing before

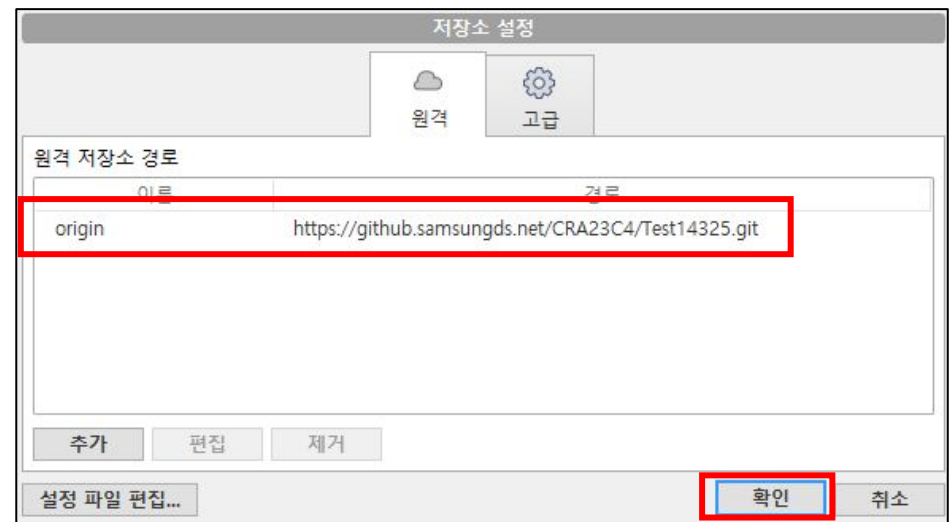
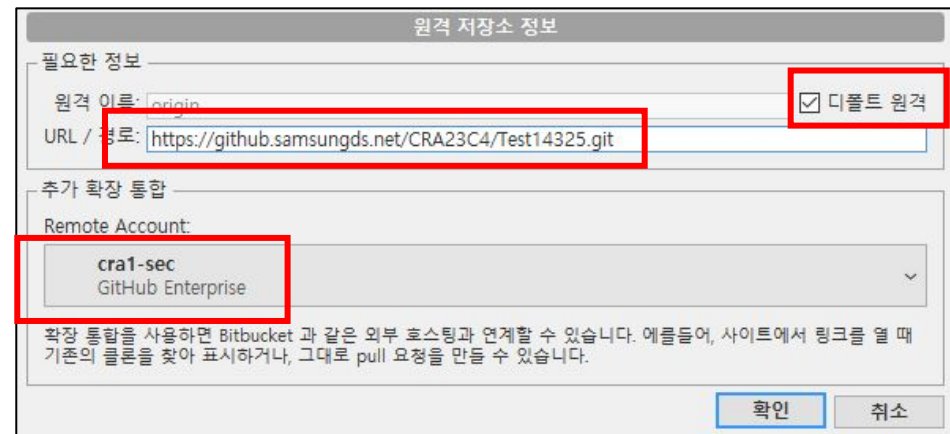
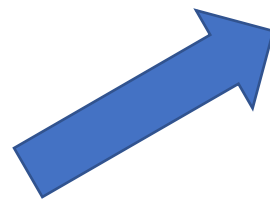
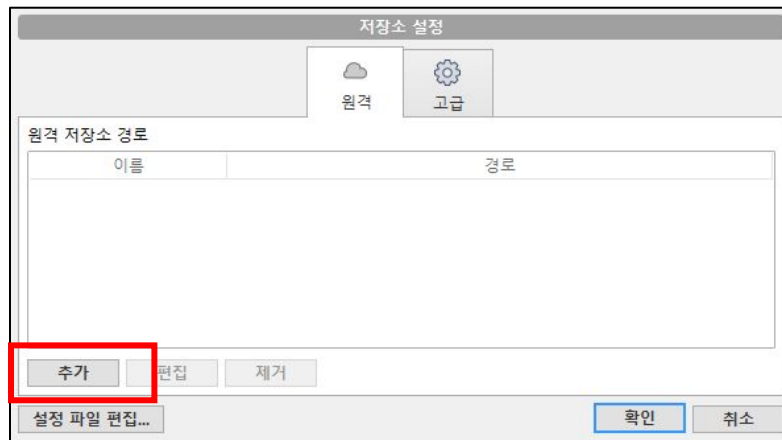
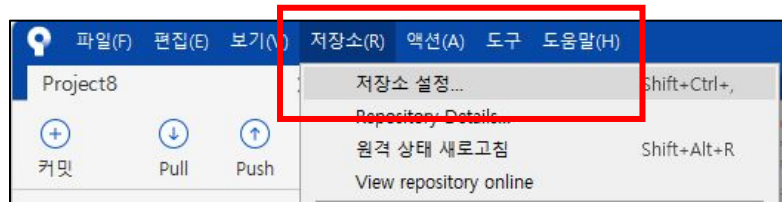
Set up in Desktop or HTTPS SSH https://github.samsungds.net/CRA23C4/Test14325.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

리드미 생성안함!

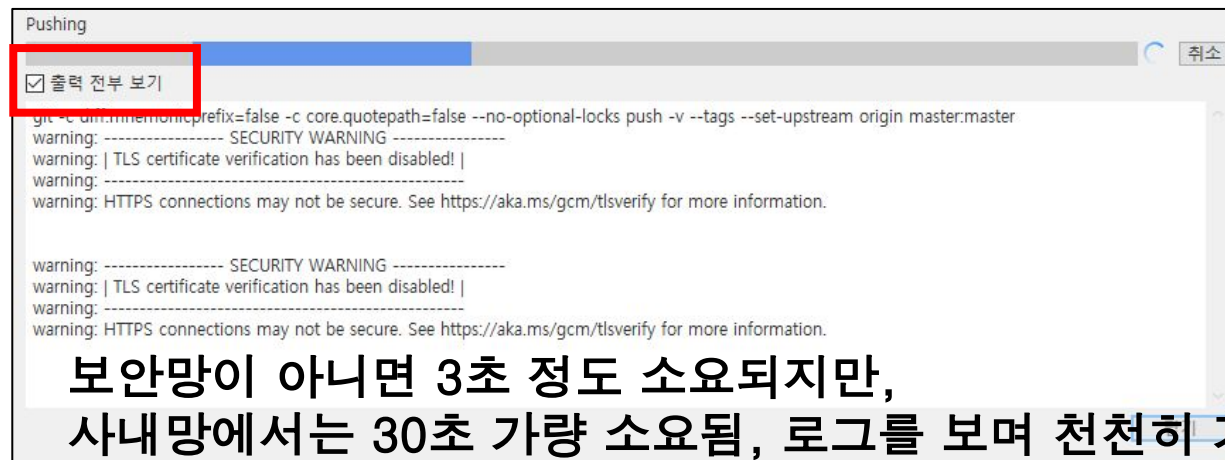
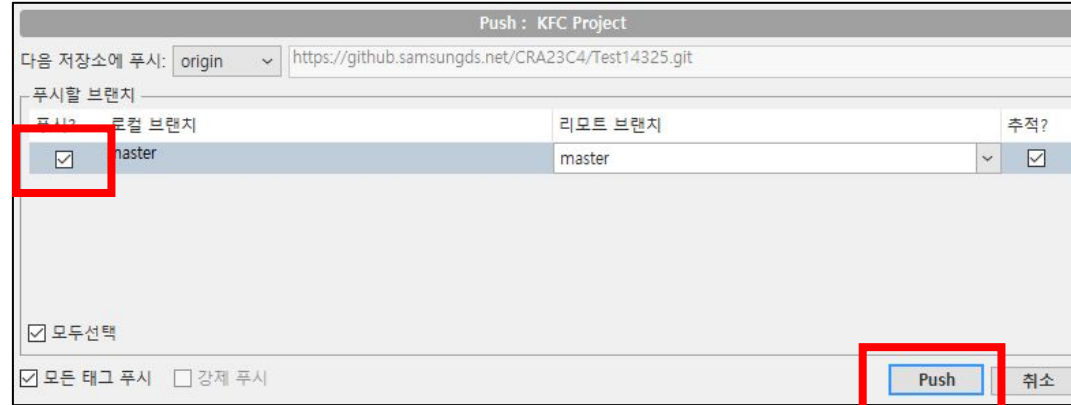
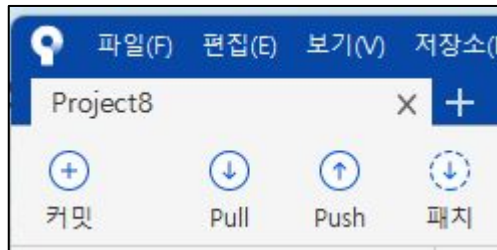
리드미 생성시에는 Push 시 충돌발생으로,
기존 Branch 변경해주고 다시 Push 해줘야하는 번거로움 발생

remote 등록 □ Push 준비 끝



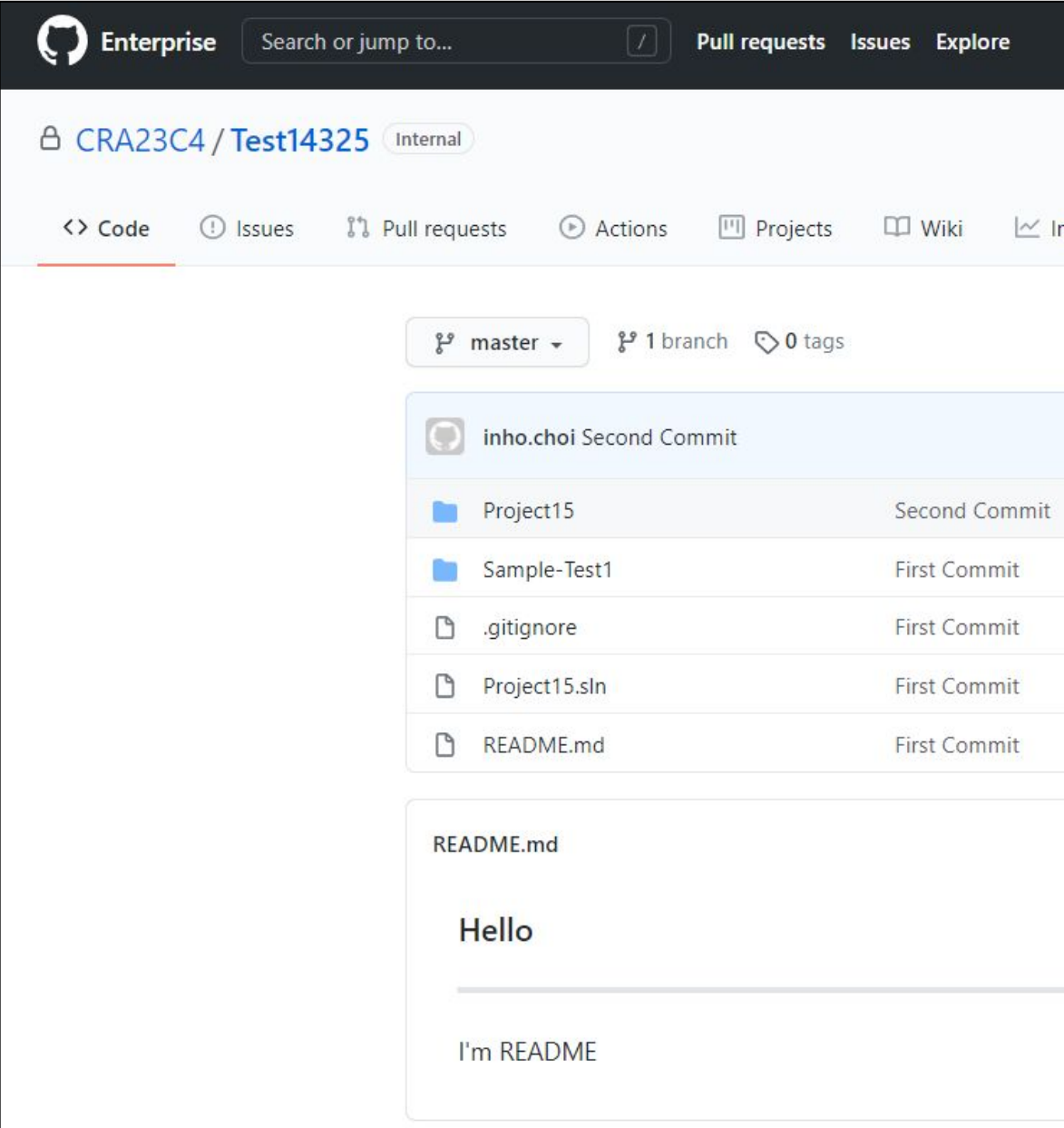
Push 진행

- ✓ Local : Master Branch를
Remote : Master Branch 로 Push 진행



Push 확인하기

✓ 정상적으로 Push 됨을 확인하자.



The screenshot shows the GitHub interface for a repository named 'CRA23C4 / Test14325' (Internal). The 'Code' tab is selected. The repository has 1 branch (master) and 0 tags. A commit by 'inho.choi' is shown as the 'Second Commit'. The commit details for 'Project15' are visible, showing a file tree with 'Sample-Test1', '.gitignore', 'Project15.sln', and 'README.md'. The 'README.md' file content is displayed below, showing 'Hello' and 'I'm README'.

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

CRA23C4 / Test14325 Internal

<> Code ! Issues 🔗 Pull requests ⏮ Actions 📁 Projects 📖 Wiki 📈 Insights

🔑 master 1 branch 0 tags

inho.choi Second Commit

| File | Commit |
|---------------|---------------|
| Project15 | Second Commit |
| Sample-Test1 | First Commit |
| .gitignore | First Commit |
| Project15.sln | First Commit |
| README.md | First Commit |

README.md

Hello

I'm README

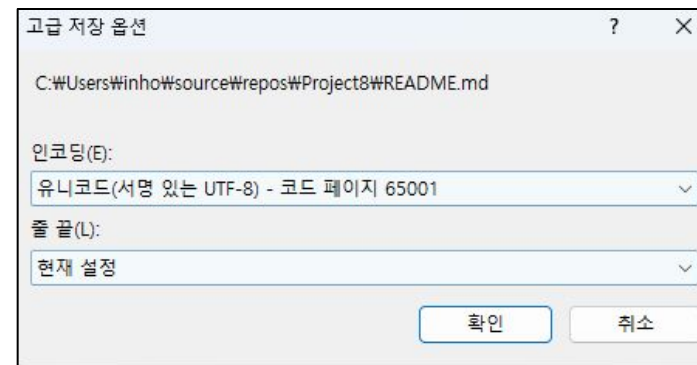
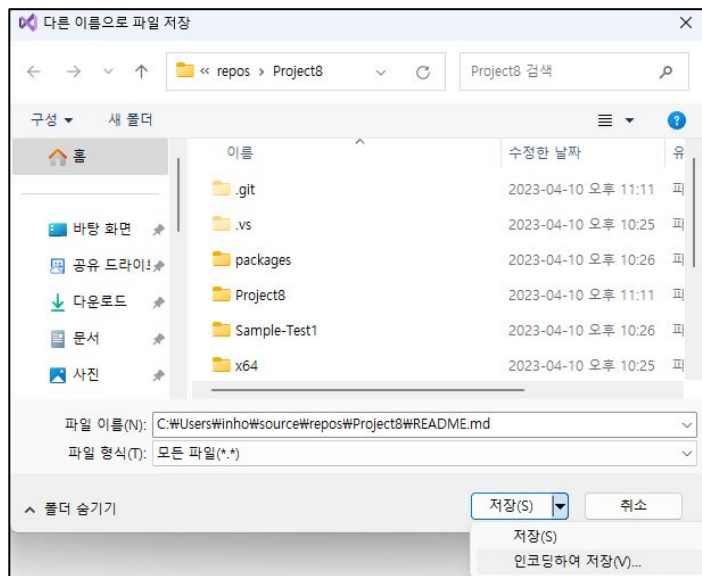
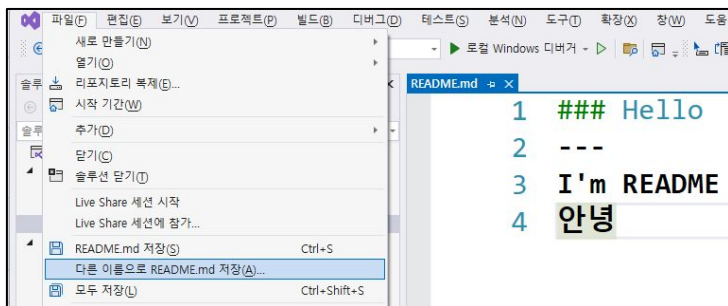
한글 깨짐 현상 해결하기

한글을 사용하여 Push 진행 시

- ✓ 한글 부분이 깨진다.
- ✓ 이유
 - Visual Studio 에서 유니코드가 아닌, EUC-KR 으로 저장하기 때문

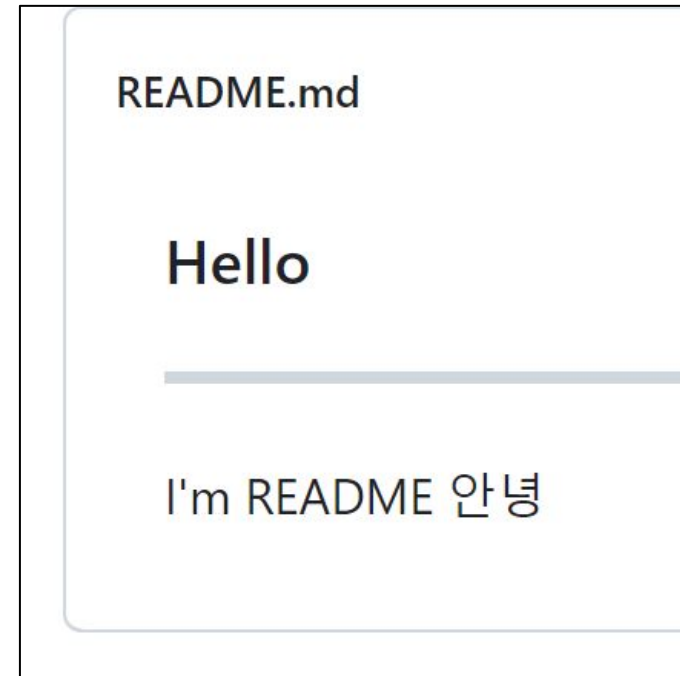
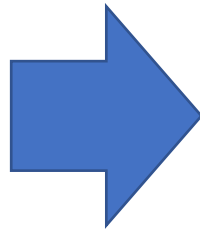
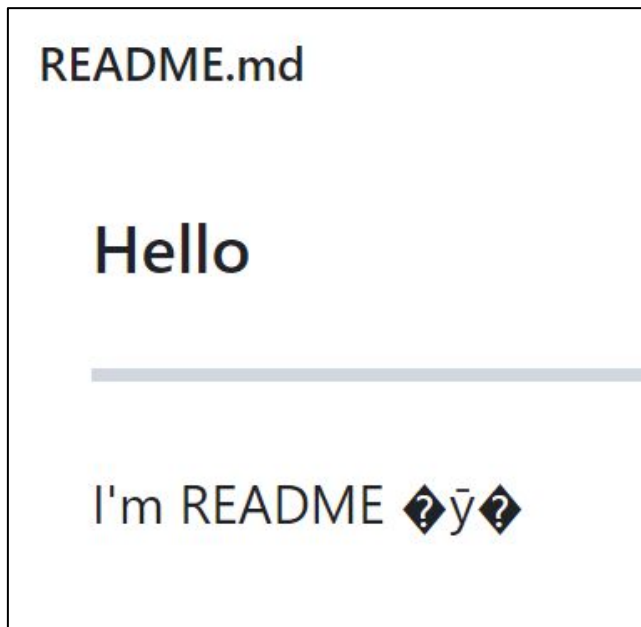


UTF-8 으로 변경하여 저장하기



다시 Commit 후 Push 하기

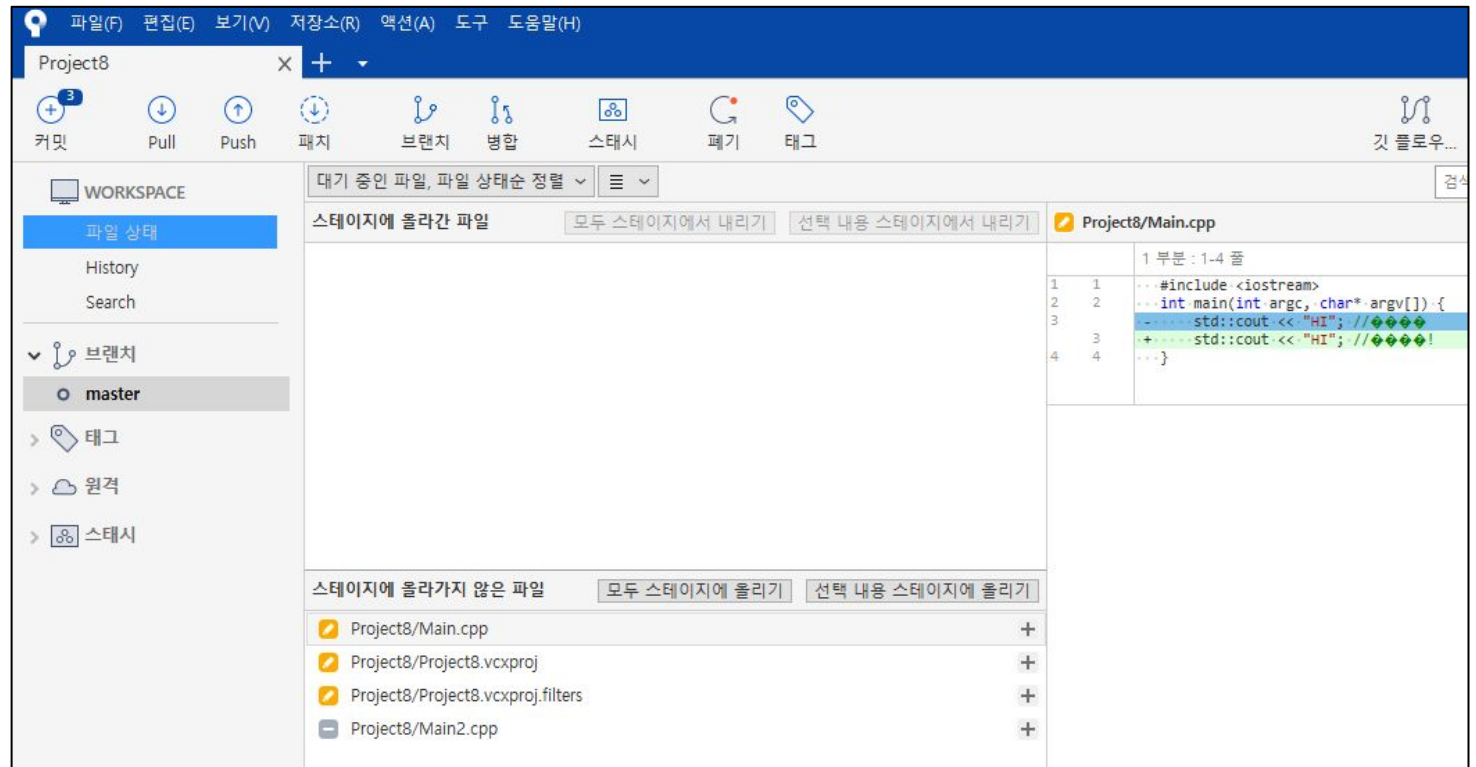
✓ 한글 적용 완료



cpp 소스코드 살펴보기

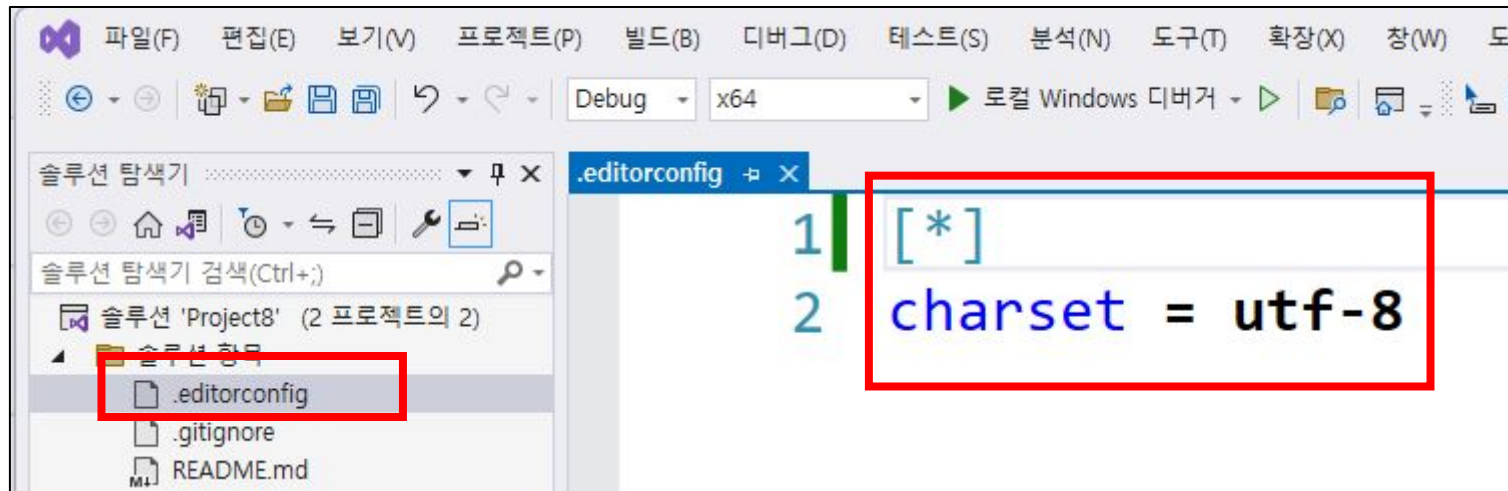
✓ 유니코드 vs ECU-KR?

```
1 #include <iostream>
2 int main(int argc, char* argv[])
3 {
4     std::cout << "HI"; //후후
5 }
```



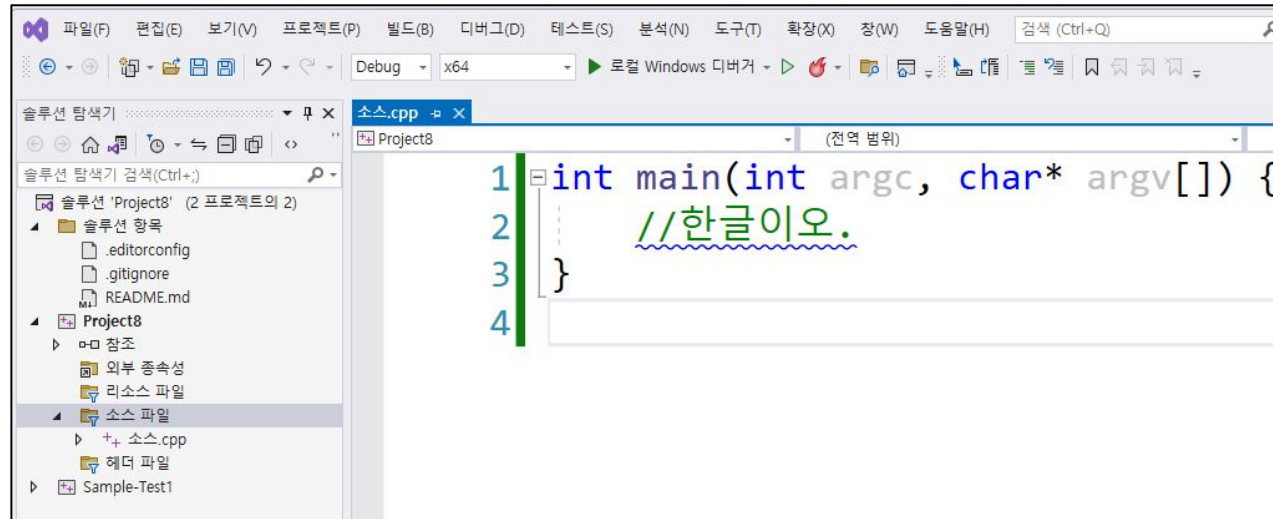
앞으로 생성되는 파일은 모두 UTF-8로 생성

- ✓ 이미 만든 파일들은 하나씩 수동으로 UTF-8로 저장해야한다.
- ✓ 앞으로 만들 파일들을 UTF-8로 자동 설정이 되도록 지정해보자.
 - .editorconfig 파일 생성 후 기입하기



다시 파일 생성 후, 한글 테스트

✓ 정상동작함.



The screenshot shows the Visual Studio Code editor interface. The main editor window displays the file '소스.cpp' with the following code:

```
1 int main(int argc, char* argv[]) {  
2     //한글이오.  
3 }  
4
```

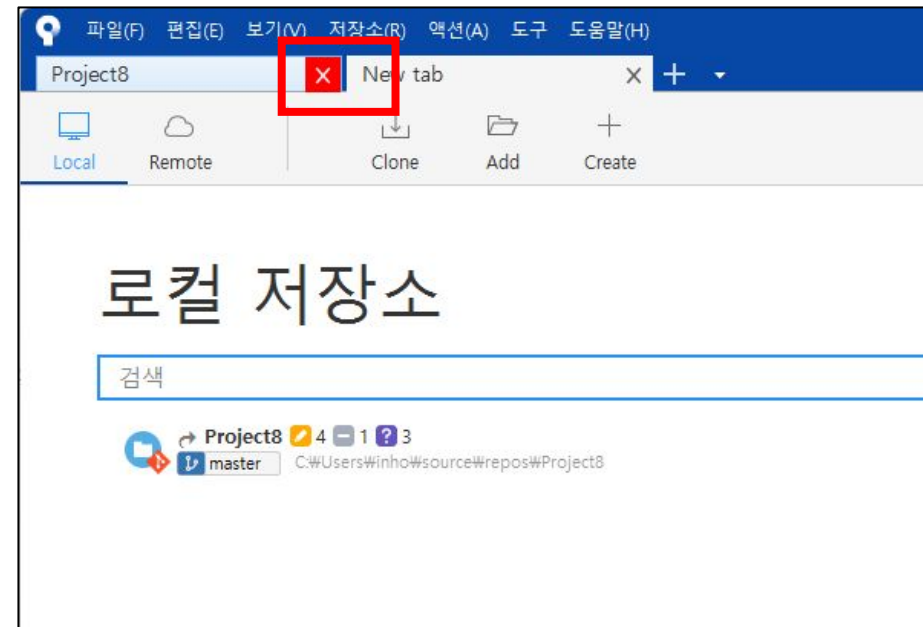
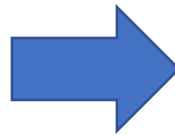
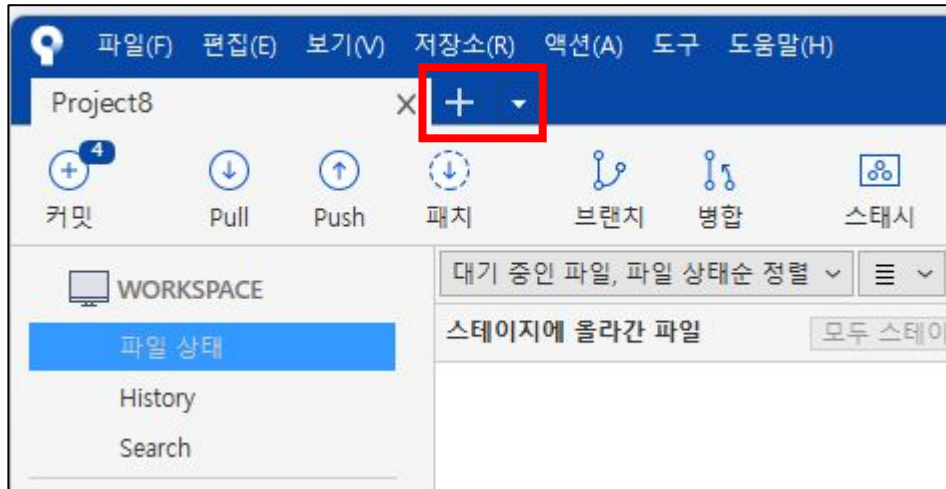
The left sidebar shows the project structure for 'Project8', including files like '.editorconfig', '.gitignore', 'README.md', and a folder '소스 파일' containing '소스.cpp'.

| Project8/소스.cpp | |
|-----------------|-----------------------------|
| | 파일 내용 |
| 1 | int main(int argc, char* ar |
| 2 | //한글이오. |
| 3 | } |

Clone

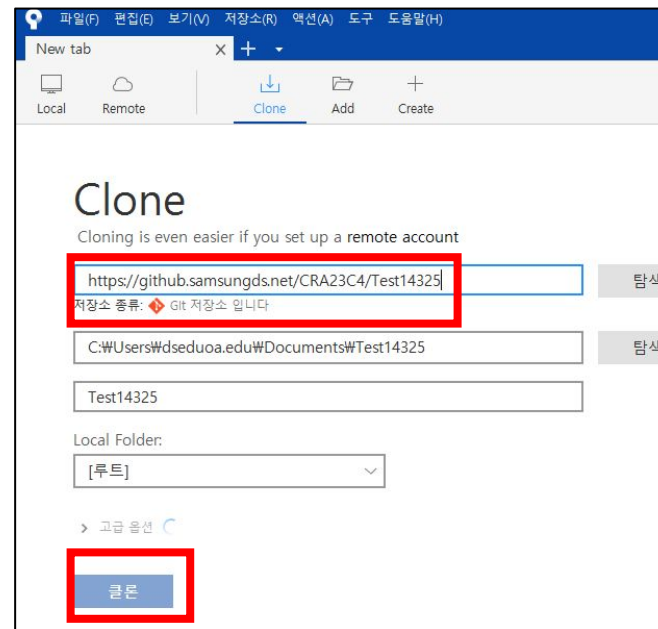
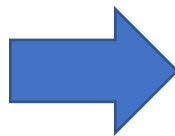
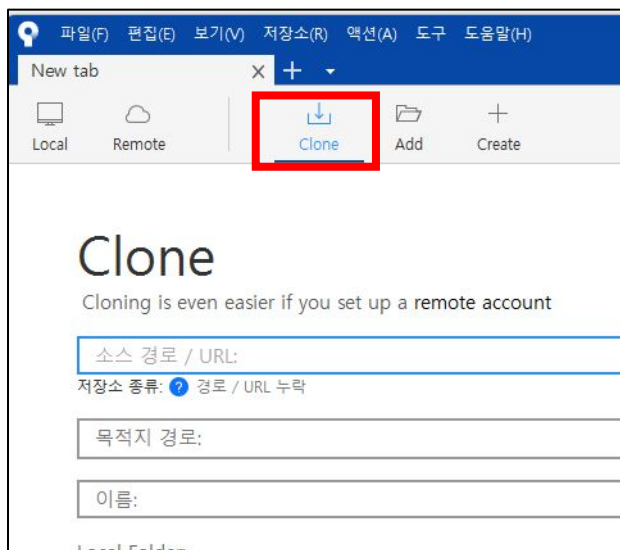
Clone을 위한 신규 Project 생성

- ✓ 신규 프로젝트 생성 후,
기존 프로젝트 닫기



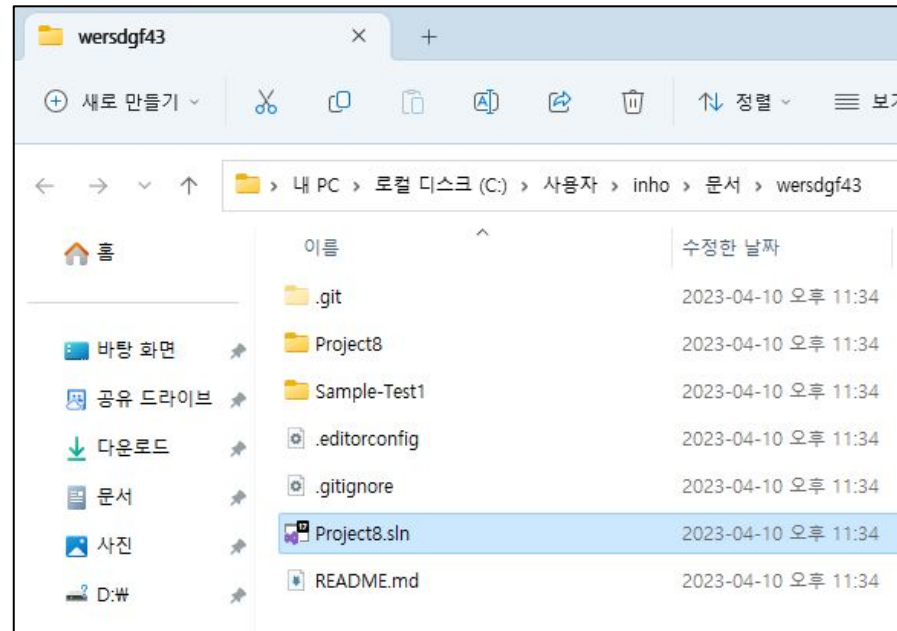
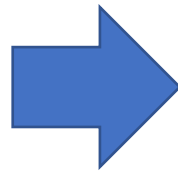
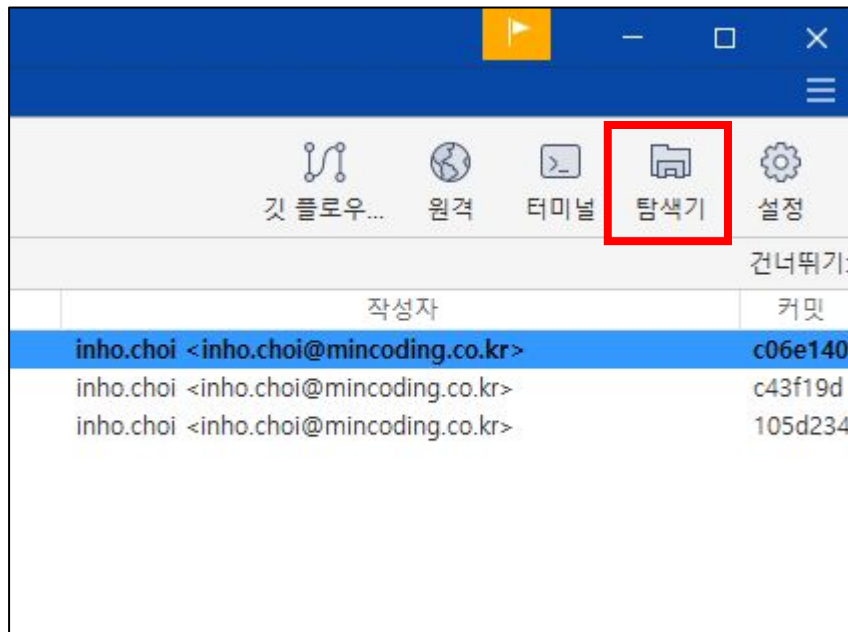
Clone 받을 Remote Repo 붙여넣기

- ✓ 1 ~ 3초 걸려야 정상이지만,
보안사업장에서는 90초 정도 소요된다.
- 저장소 종류 파악하는데 30초
 - 클론 버튼 활성화 되는데 60초



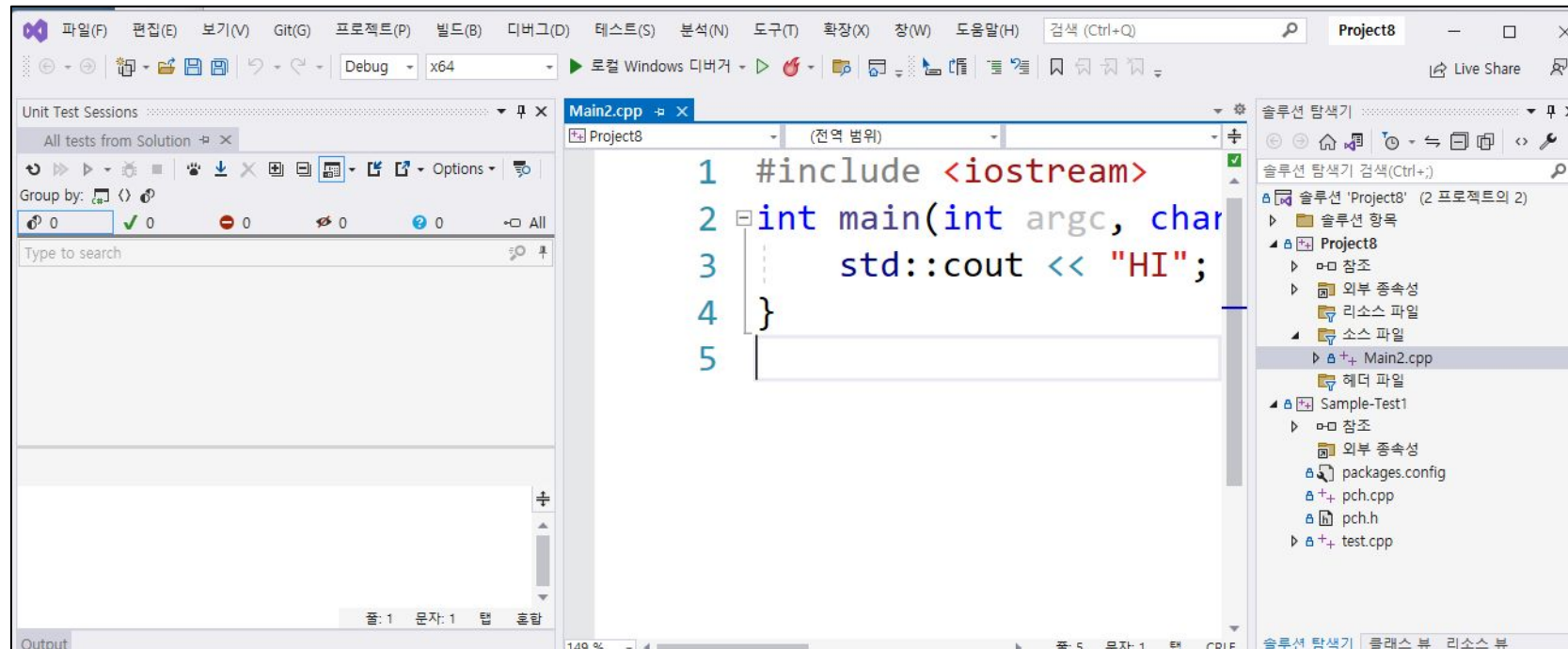
Clone 받은 프로젝트 Visual Studio로 열기

✓ 솔루션 파일 (.sln) 열기



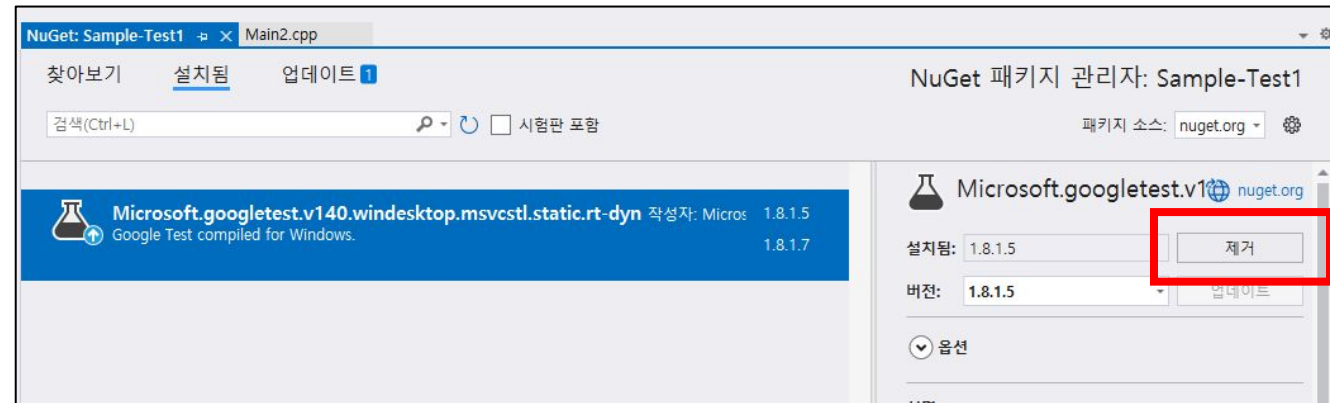
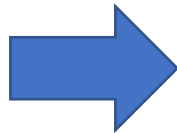
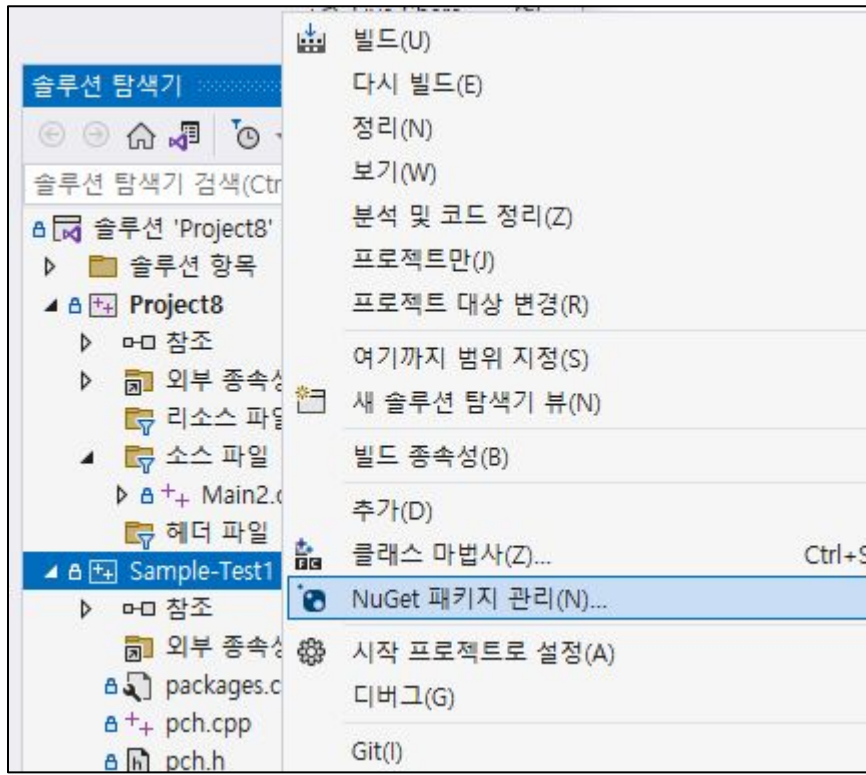
신규로 받은 프로젝트, 테스트 수행해보기

- ✓ 테스트가 실행이 안된다면
Google Test를 다시 설치해야한다. (다음 페이지 참조)

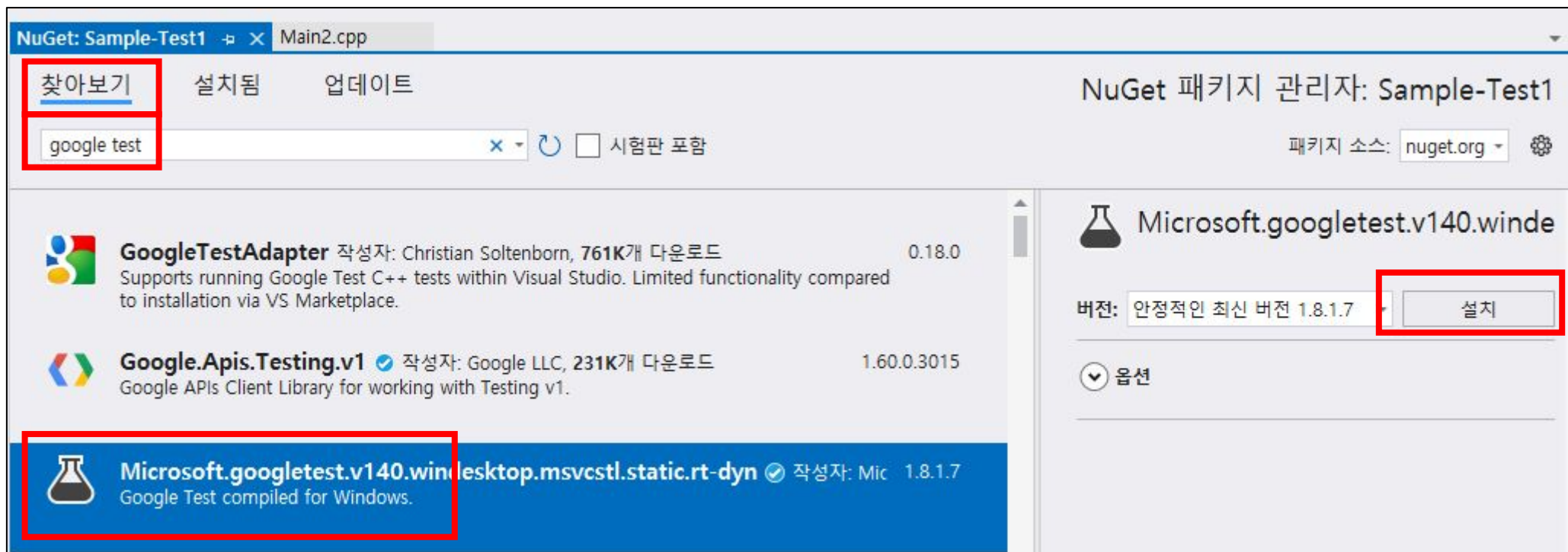


Google Test 다시 설치하기 1

✓ NuGet 패키지 관리도구 실행 후, Google Test 제거한다.

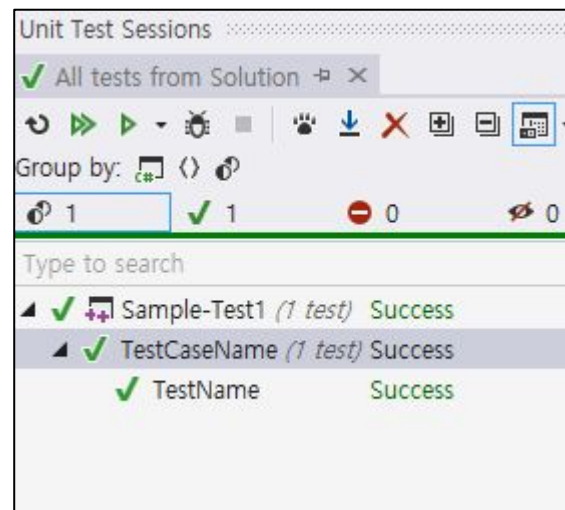
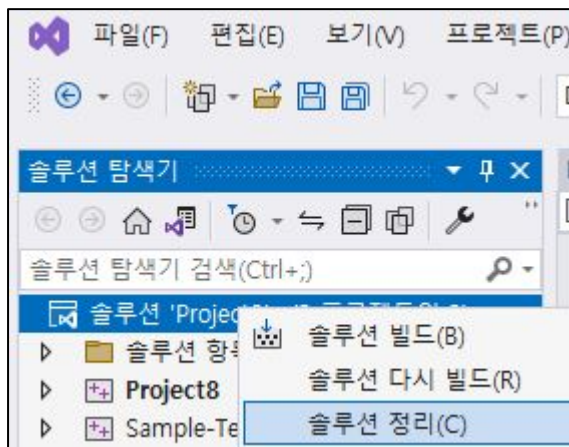


다시 설치한다.



솔루션 정리 후 다시 빌드

✓ 정리 후 다시 빌드를 해본다.

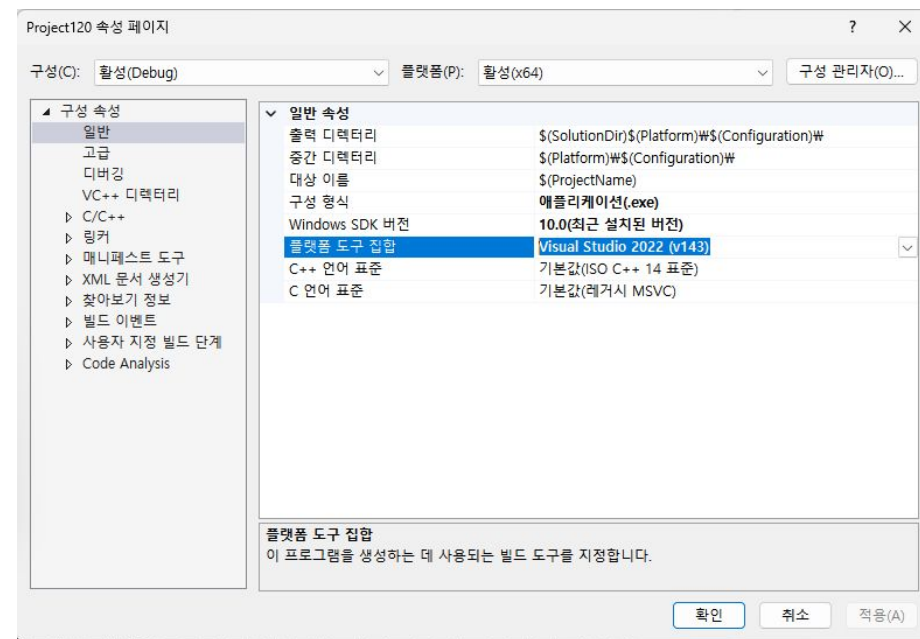


Trouble Shooting

- ✓ v 1 4 3 을 찾을 수 없습니다. 오류 발생시
 - 프로젝트 자체는 Visual Studio 2022로 만들었지만, Clone 을 받은 본인은 해당 버전을 사용하지 않을 때 발생

✓ 해결방법

- 프로젝트 > 속성 > 플랫폼 도구 집합
에서 본인의 버전으로 선택



Visual Studio 에서 Git CLI 사용

Visual Studio 에서 Git Bash (CLI)를 쓰는 이유

✓ GUI 장점 : 가독성

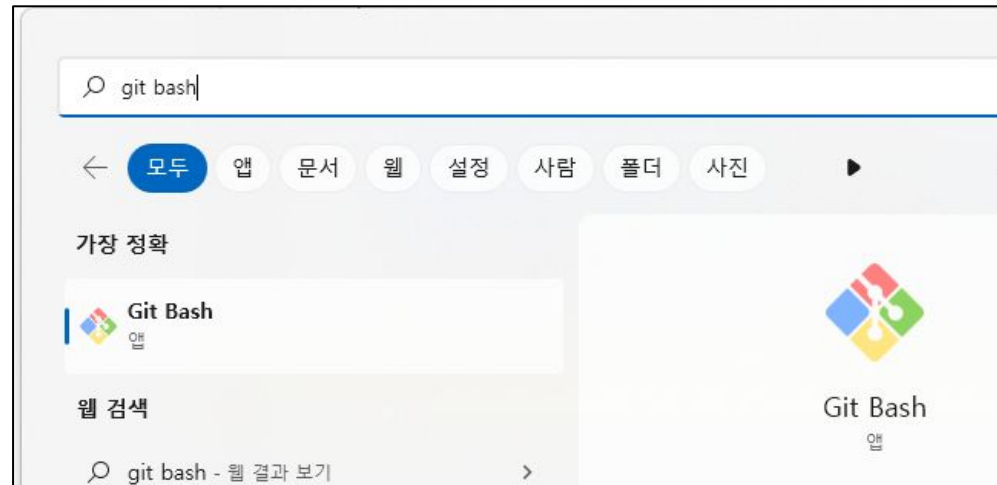
- Git History 을 GUI로 가독성 있게 확인 가능.
- 두 개의 파일 내용 비교 (Diff)가 가독성

✓ GUI 단점 : 느린 동작속도

- Commit & Push 를 할 때마다, 마우스 클릭 해줘야 한다.
- 속도가 CLI보다 느리다.

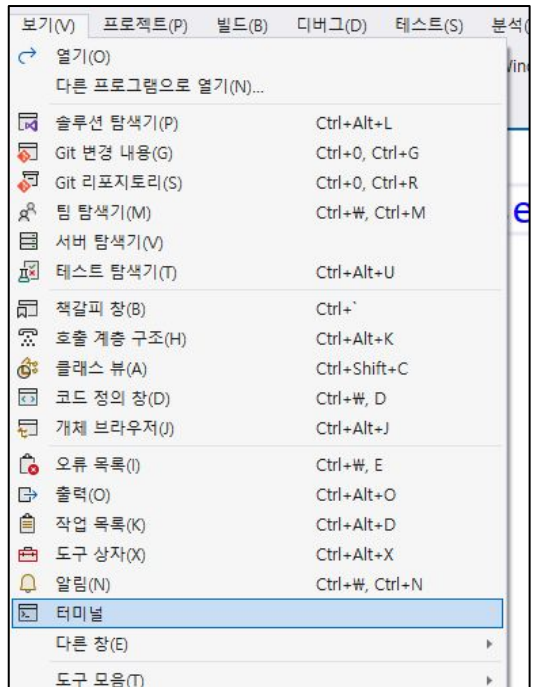
환경세팅

로컬 또는 개발 PC에 git bash가 설치 필수



Visual Studio 하단, Terminal 사용하기

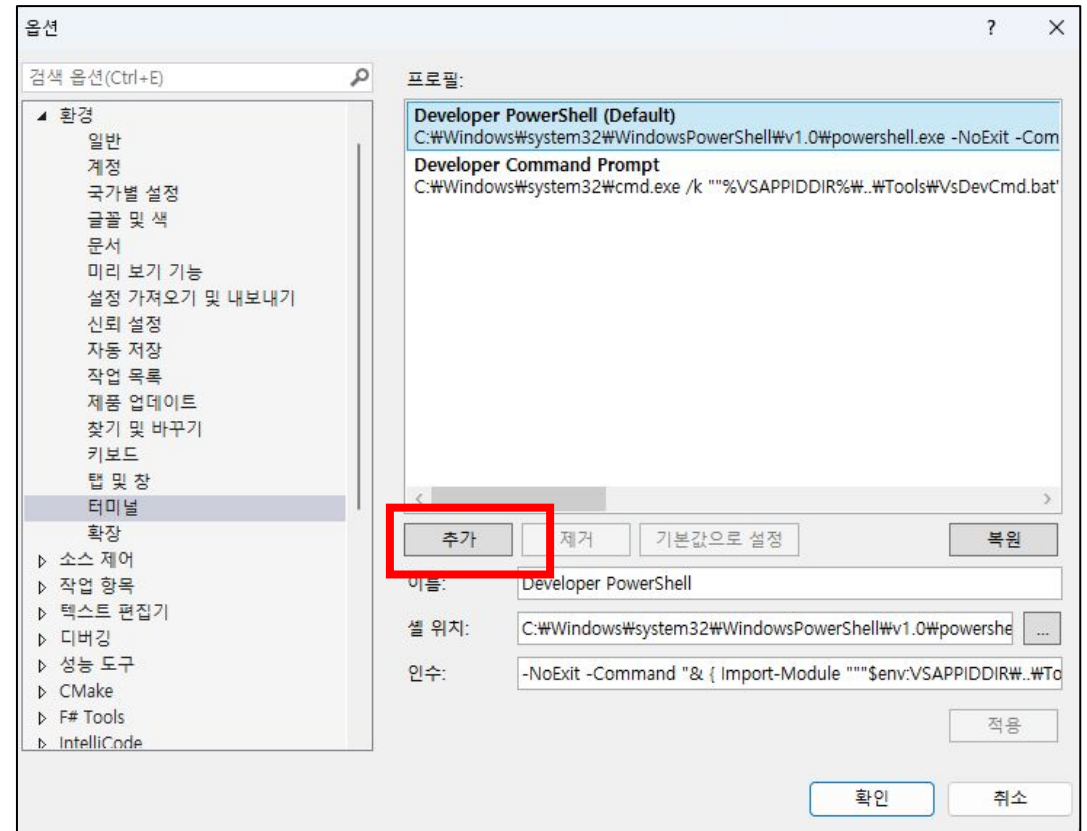
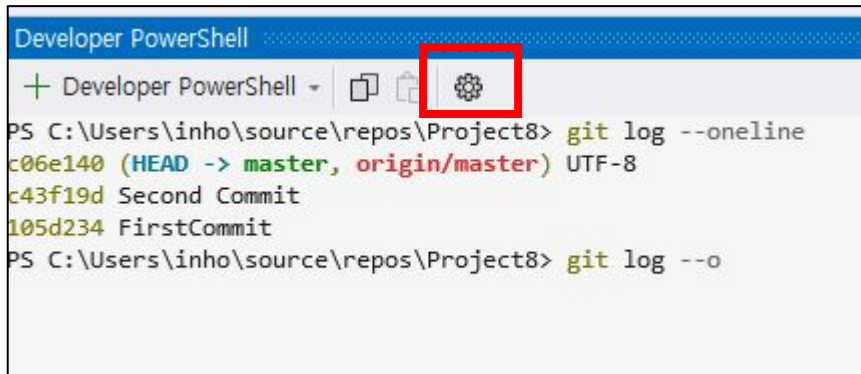
CLI 사용 환경 제공



```
Developer PowerShell
+ Developer PowerShell
PS C:\Users\inho\source\repos\Project8> git log --oneline
c06e140 (HEAD -> master, origin/master) UTF-8
c43f19d Second Commit
105d234 FirstCommit
PS C:\Users\inho\source\repos\Project8> git log --o
```

단점 : 자동완성이 Tab 키로 불가하다.

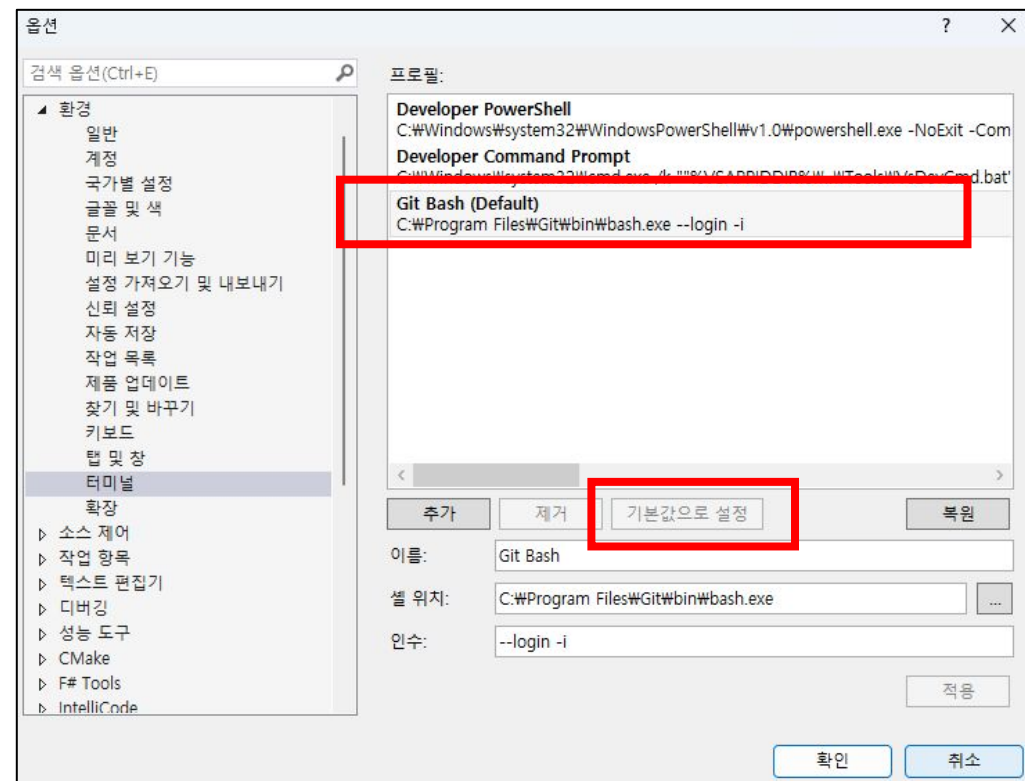
터미널 환경설정 누르기



Git Bash 경로 추가

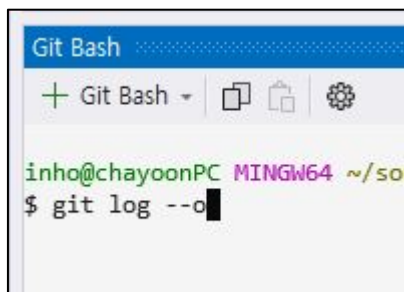
✓ 기본값으로 설정하기

| | |
|---------------|-----------------------------------|
| 이름: | Git Bash |
| 셸 위치: | C:\Program Files\Git\bin\bash.exe |
| 인수: | --login -i |
| <div>적용</div> | |



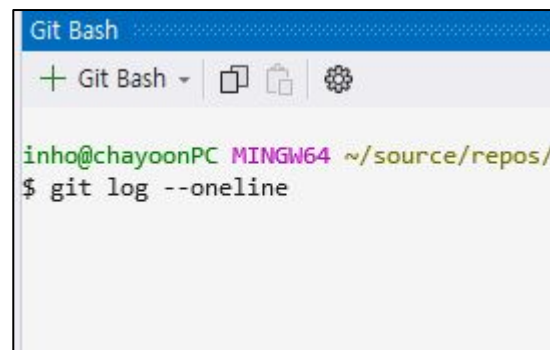
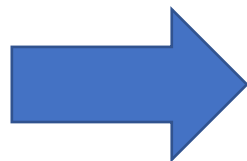
완성

✓ 자동완성이 잘 된다.



```
Git Bash
+ Git Bash
inho@chayoonPC MINGW64 ~/so
$ git log --o
```

git log --o 누른 후 TAB 키



```
Git Bash
+ Git Bash
inho@chayoonPC MINGW64 ~/source/repos/
$ git log --oneline
```