



Version Control System

GIT (깃)

한국전기연구원/전기의료기기 연구센터

INDEX

KOREA ELECTROTECHNOLOGY RESEARCH INSTITUTE

1. 배경 설명
2. 필요성
3. 중요 개념
4. 사용하기
5. GUI 클라이언트
6. 브랜치 관리 기법
7. 협업하기
8. 기타



- 의료 영상 처리 알고리즘 : 실시간 초음파/CT/MRI 영상 정합 (삼성전자 의료기기 S-fusion)
- 음성 신호 처리 알고리즘 : 보청기 알고리즘 개발 프레임워크 개발
- 인공지능 알고리즘 연구 (현재):
 - 무선전력전송용 안테나 인공지능 기술
 - 의료영상 망막 산소포화도 측정 인공지능 기술

서울대학교
전기공학부, 의용생체공학
(1999년~2011년)

삼성전자
종합기술원
(2011년~2015년)

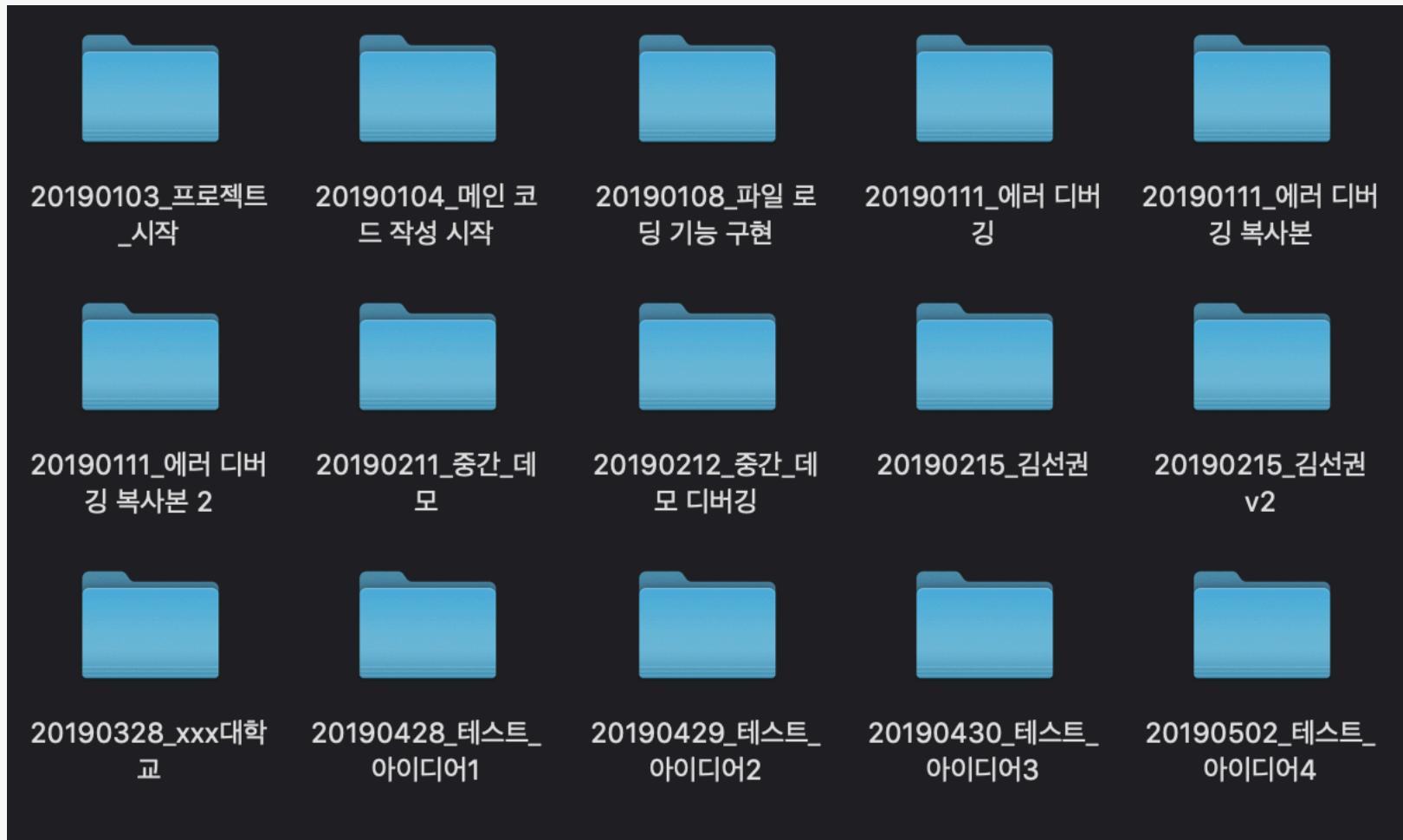
한국전기연구원
첨단의료기기연구본부
(2015년~현재)



Git이 탄생한

1. 배경 설명

1. 프로그램을 개발하다 보면...



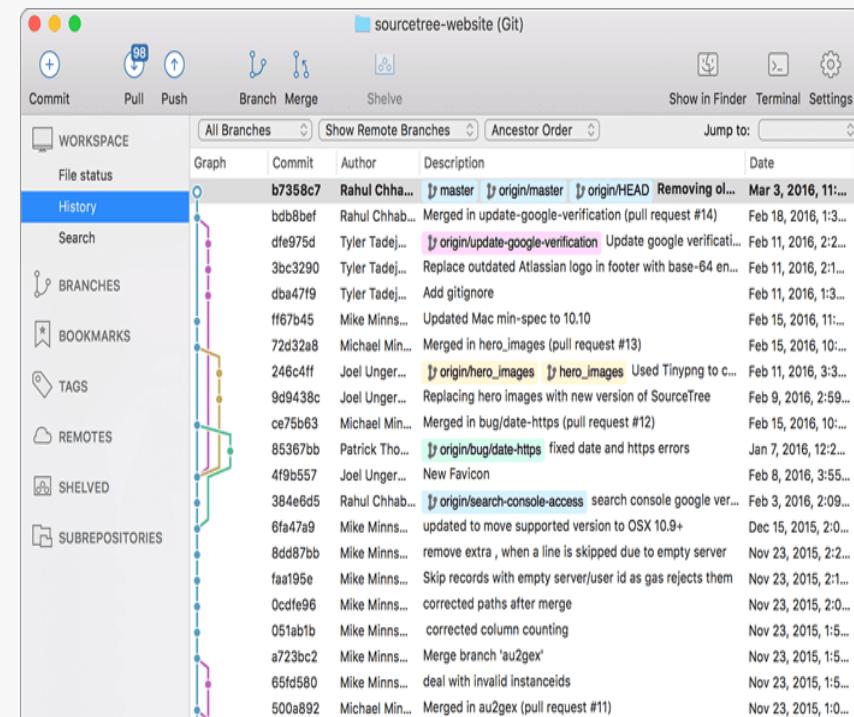
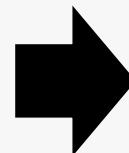
1. 프로그램을 개발하다 보면...



1. 프로그램을 개발하다 보면...

KERI

코드 관리를 도와줄 뭔가가 필요하다.



Version Control System

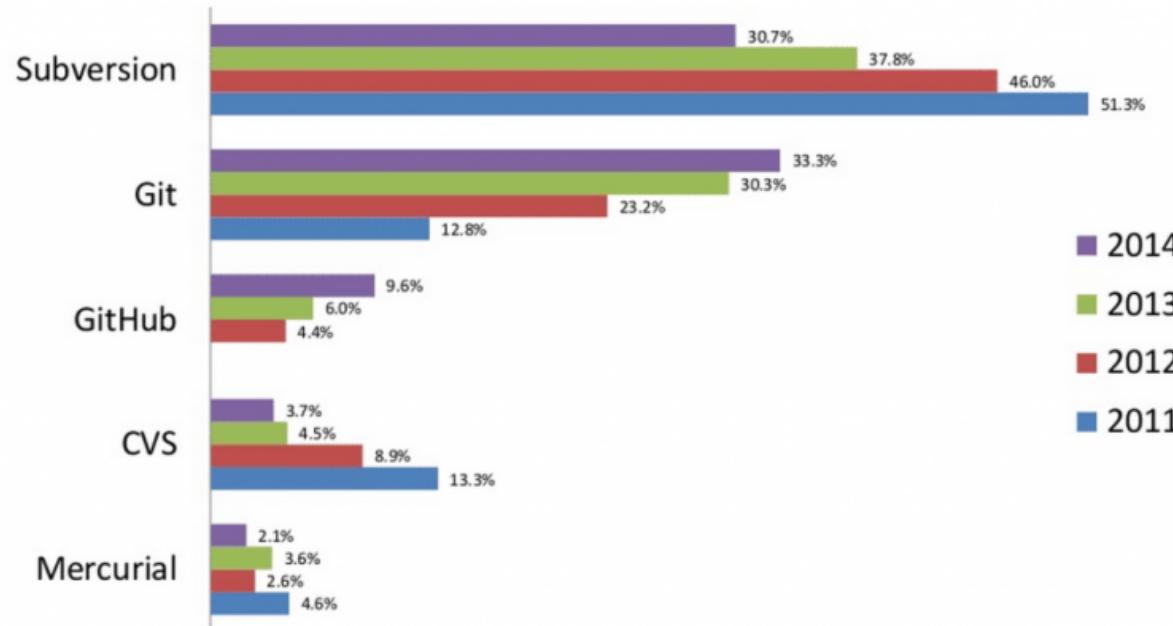
Local —————> Centralized —————> Distributed

- RCS
- CVS
- Subversion
- Perforce
- Git
- Mercurial
- Bazaar

Primary Code Management



What is the primary source code management system you typically use? (Choose one.)



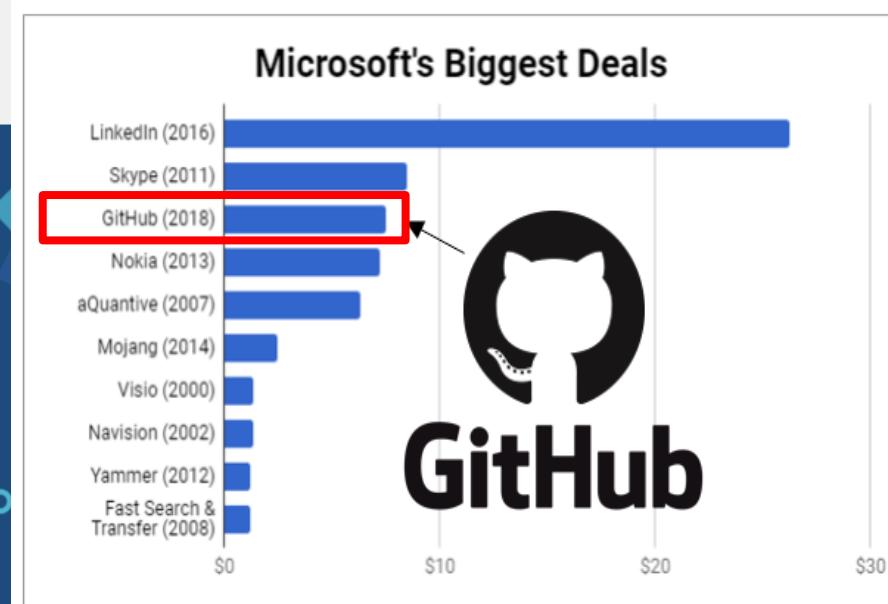
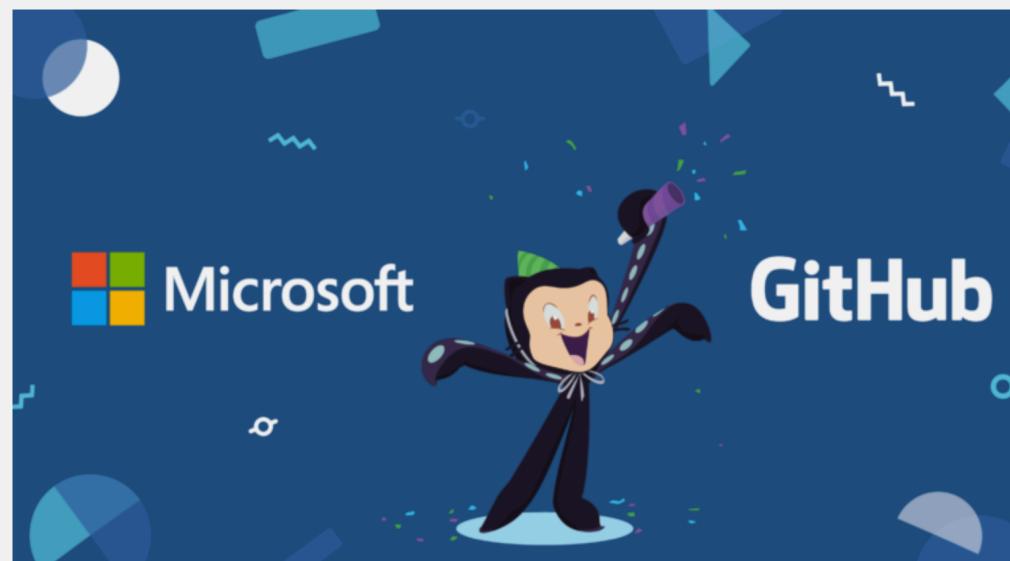
1. 배경설명 – Git의 현재

BEWARE OF OCTOCLIPPY —

Microsoft snaps up GitHub for \$7.5 billion

Biggest open source hosting service joins biggest proprietary software company.

PETER BRIGHT - 6/4/2018, 10:56 PM



GitHub

31 million
Users

57 million
repositories

Largest host
of source
codes



리누스 토발즈

리눅스 개발에 소스 코드 관리 프로그램인 BitKeeper를 사용하고 있었음

- BitKeeper가 무상 제공이 종료됨에 따라 리누스 토발즈가 직접 소스 코드 관리 프로그램(Git)을 직접 개발하여 사용함
- 당시 소스 컨트롤 프로그램들의 단점을 보완(극복)할 수 있도록 개발됨.

1. 배경설명 – Git의 창시자



Linus Torvalds

torvalds

Follow

Linux Foundation

Portland, OR

Block or report user

Organizations

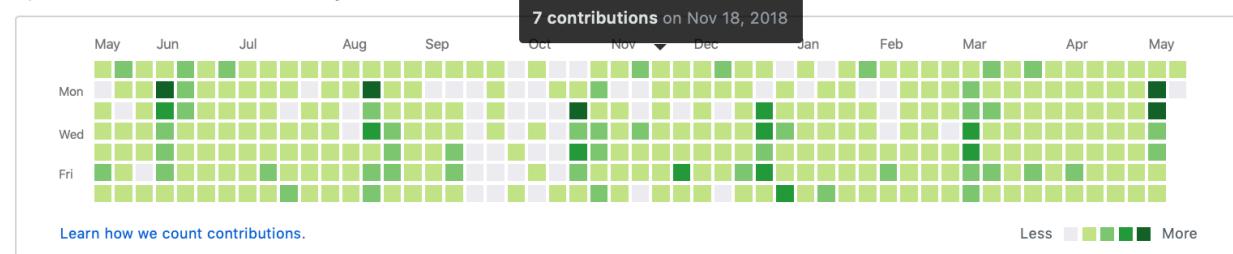


Overview Repositories 6 Projects 0 Stars 2 Followers 93.6k Following 0

Popular repositories

linux Linux kernel source tree ● C ★ 43.3k ⚡ 16.5k ● C ★ 74.8k ⚡ 26.2k	(2017.03)	uemacs Random version of microemacs with my private modifications ● C ★ 380 ⚡ 57
test-tlb Stupid memory latency and TLB tester ● C ★ 212 ⚡ 69	(2019.05)	pesconvert Brother PES file converter ● C ★ 112 ⚡ 12
subsurface-for-dirk Forked from Subsurface-divelog/subsurface Do not use - the real upstream is Subsurface-divelog/subsurface ● C++ ★ 85 ⚡ 31		libdc-for-dirk Forked from Subsurface-divelog/libdc Only use for syncing with Dirk, don't use for anything else ● C ★ 43 ⚡ 18

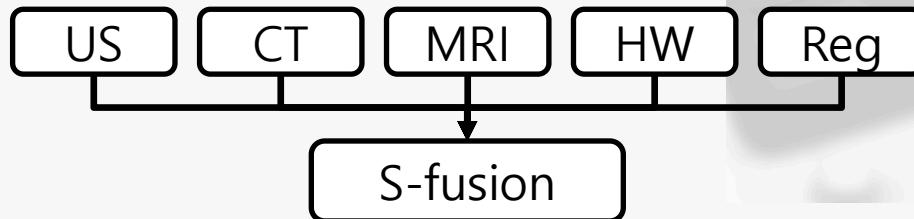
2,240 contributions in the last year



1. 배경설명 – Git을 이용한 연구/개발

삼성 메디슨 S-Fusion 개발

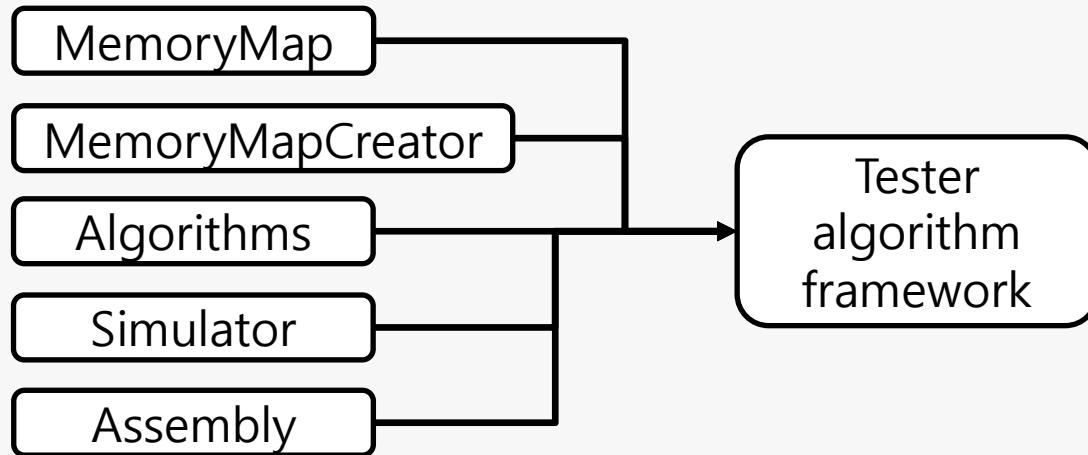
- 초음파, CT, MRI 실시간 정합
- 중재 시술용



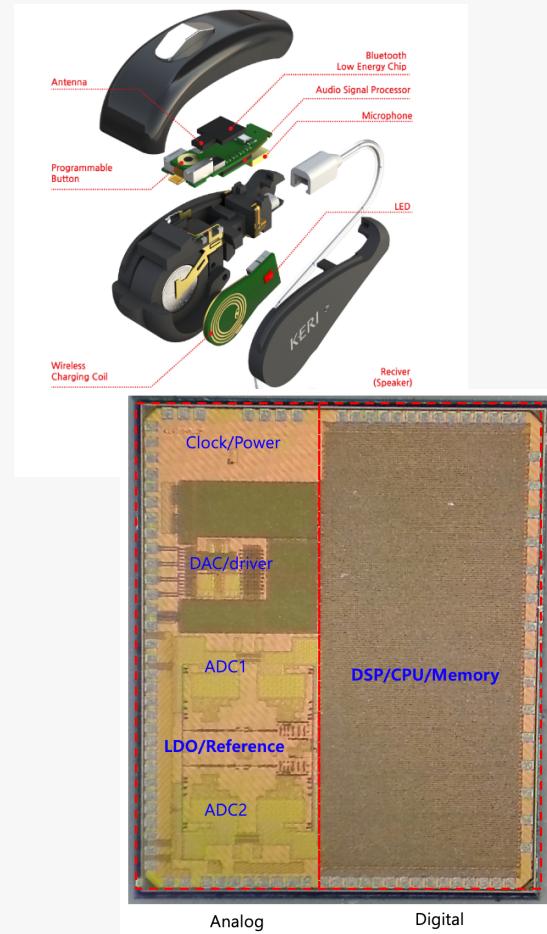
1. 배경설명 – Git을 이용한 연구/개발

KERI

보청기 알고리즘 연구/개발 알고리즘 테스터 프레임워크



Commit	Author	Date	Time
MemoryMap에 문자열 임포트 Global과 Internal 해모 주소를 조회하는 기능 추가			
faf13f4	sunkwonkim <sunk...	2018. 3. 27.	9:5 3:42
fa800e8	sunkwonkim <sunk...	2018. 3. 27.	9:5 2:45
b4015ef	sunkwonkim <sunk...	2018. 3. 28.	0:3 8:03
f4993f1	Kyunghean Cho <c...	2018. 3. 28.	0:3 10:03
2afe2bb	Kyunghean Cho <c...	2018. 3. 28.	0:3 8:17
e128466	sunkwonkim <sunk...	2018. 3. 27.	9:8 12:06
bf39558	sunkwonkim <sunk...	2018. 3. 27.	9:8 12:00
1716474	sunkwonkim <sunk...	2018. 3. 27.	2:51 11:55
2657c4d	sunkwonkim <sunk...	2018. 3. 27.	2:51 11:46
5404f4a	sunkwonkim <sunk...	2018. 3.	2:51 11:28
14150f7	sunkwonkim <sunk...	2018. 3. 27.	2:51 11:23
10372b	sunkwonkim <sunk...	2018. 3. 26.	9:8 3:51
2ddc006	sunkwonkim <sunk...	2018. 3. 26.	9:8 3:43
44c2cb8	Keedong Yang <dy...	2018. 3. 26.	9:8 3:41
f6343bd	sunkwonkim <sunk...	2018. 3. 26.	9:8 3:09
13ef6d9	Keedong Yang <dy...	2018. 3. 26.	9:8 10:27
6a27b80	sunkwonkim <sunk...	2018. 3. 26.	9:8 3:08
0180934	sunkwonkim <sunk...	2018. 3. 26.	9:8 1:19
819c9a9	Kyunghean Cho <c...	2018. 3. 28.	0:3 8:10
39d3a40	Kyunghean Cho <c...	2018. 3. 28.	0:3 8:06
MCRA / LogMMSE 오류 수정			
MemoryMap에 문자열 임포트 Global과 Internal 해모 주소를 조회하는 기능 추가			
MCRA.asm에서 단계적 수준 MCRA_alpha,d --> MCRA_alpha_d			
FDFV error fix (FDFV: O Slot 0槽 기자)			
Merge remote-tracking branch 'origin/sunkwonkim' into khcho			
refactoring : std namespace 사용을提倡 대체			
친환경 성능을 위하여 위해서는 정 표시가 바로 충족되도록 설정. (std::flush을 충족함.) Runner.cpp에서 std namespace 사용. Runner.h에 using namespace std; 구현이 있으므로.			
AsmModule (내부에서 Simulator 관련 코드를 충족해서 Debug을 관련 정보를 출력하지 않도록 수정			
Runner에서 주사율리미트 크고 저자 총 실행시간과 실행된 프레임 수를 표시하도록 함			
CMake 최소 버전을 3.6로 수정			
필터즈 스크립트 뛰어 수정			
ManagedCore의 Parser.cpp에서 바로 프로젝트를 빠져나와서 초기화 (ManagedCore.h.release, Stdafx.h.release) 추가된 해더 파일은 바로 프로젝트 스크립트 (release.sh)에서 각 -			
144cc5b0c942ea7be024bd180b7c4317a363f into sunkwonkim			
Merge commit '44cc5b0c942ea7be024bd180b7c4317a363f' into sunkwonkim			
Revert 'sed'을 사용 하면서 파일 path 조작하는 부분을 변경 (ffidfd 구문 및 #define 구문 사용)			
Merge commit '13ef6d9c8b98f648fb0a6e66d77ab78ca2c' into sunkwonkim			
sed을 사용 하면서 파일 path 조작하는 부분을 변경 (ffidfd 구문 및 #define 구문 사용)			
Merge commit '13444f169b2b32d3282d5e1682244eb4e9295f' into sunkwonkim			
unsigned long long -> int64_t long long -> int64_t 1 int -> int32_t			
Simulator C++ 관련 코드에서 unsigned long long을 uint64_t로 변경. unsigned long long 타입은 Operating OS가 32bit/64bit 차지 32bit 또는 64bit 데이터를 다뤄야 해서			
Logger Binary 기관			
MCRA / LogMMSE 오류 수정			



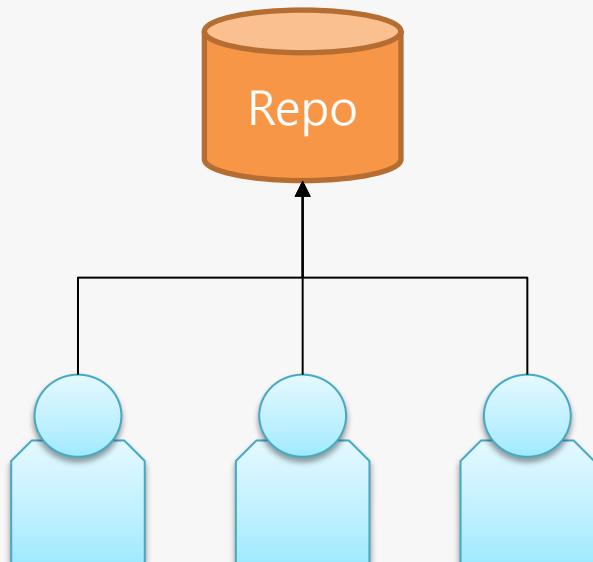
필요성

2. GIT이 왜 좋은가?

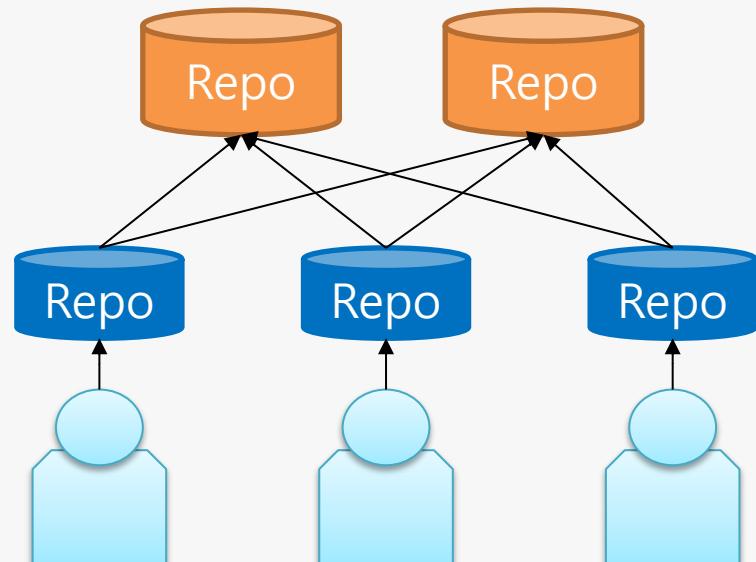
2. 기존의 툴과 비교

	CVS	SVN	Git
개발자	The CVS Team	콜랩넷	리누스 토발즈
종류	중앙집중식	중앙집중식	분산 버전 처리
운영체제	유닉스, 윈도우	크로스 플랫폼	크로스 플랫폼

CVS / SVN

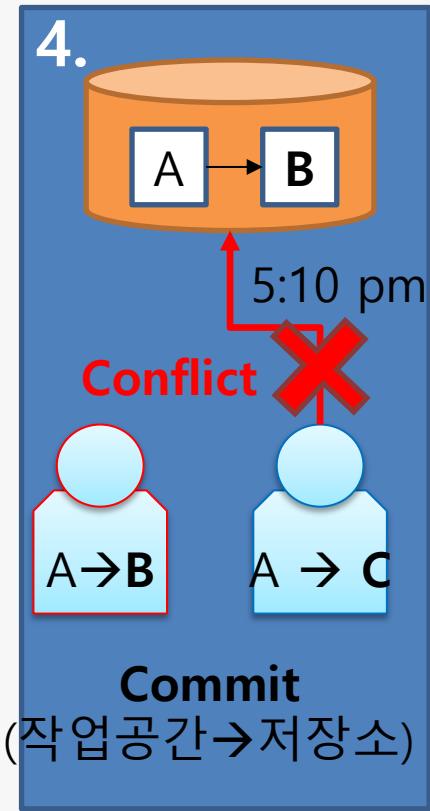
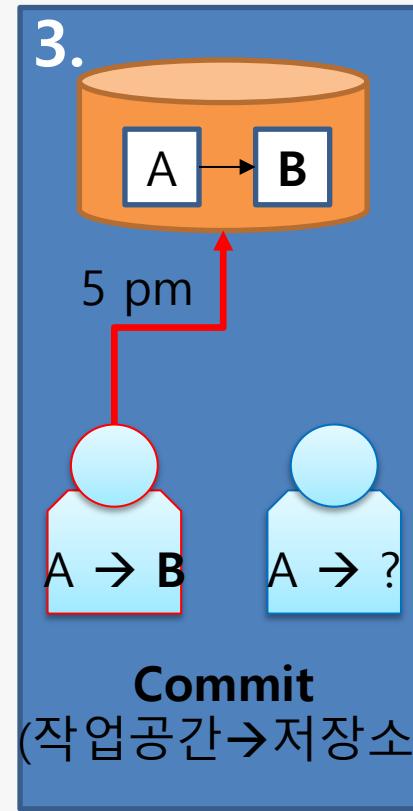
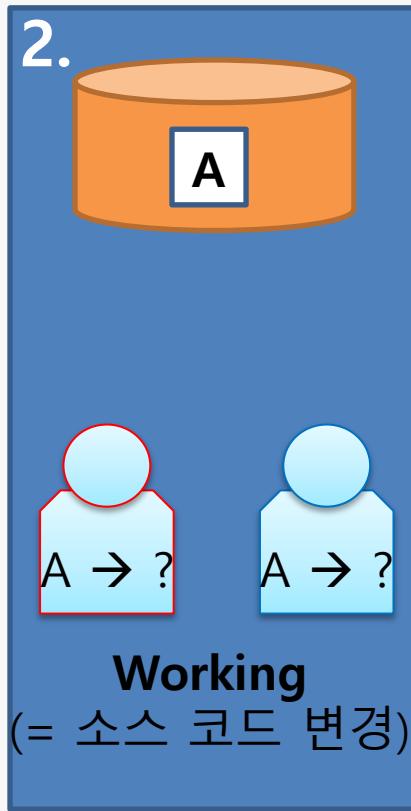
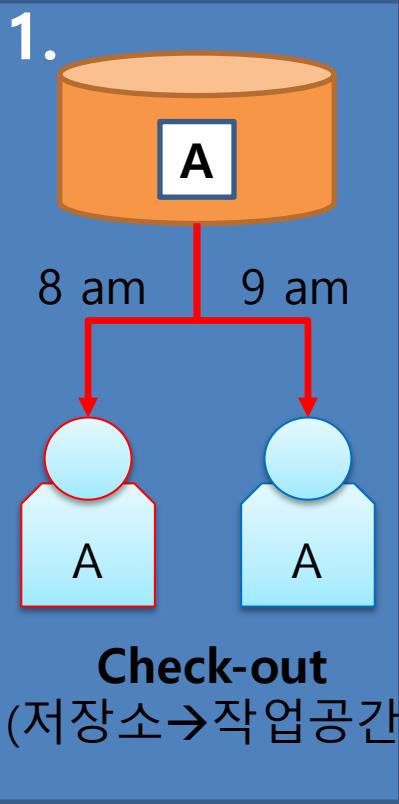


Git



2. 기존의 툴과 비교

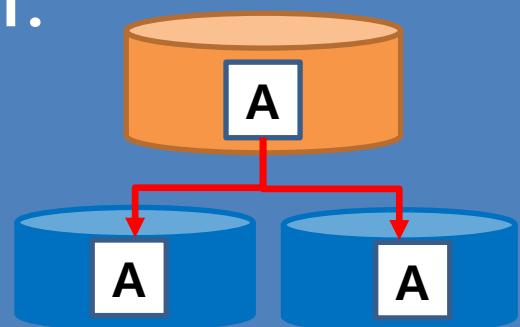
CVS / SVN



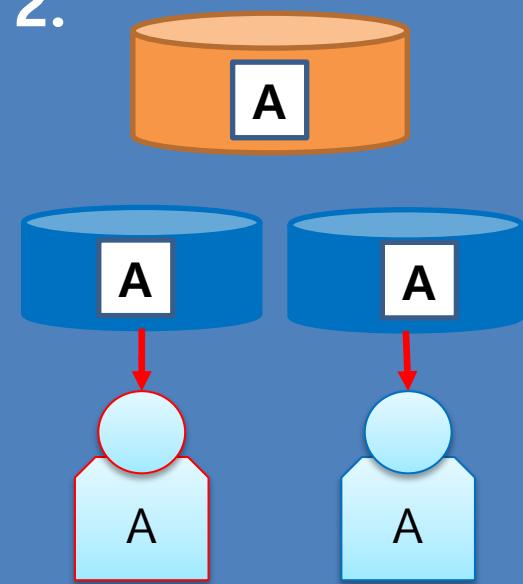
2. 기존의 툴과 비교

Git

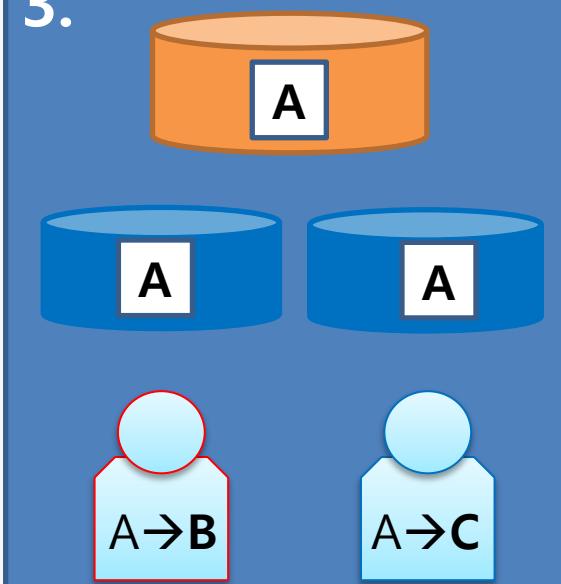
1.



2.



3.



Pull

(원격저장소→로컬저장소)

Check-out

(로컬저장소→작업공간)

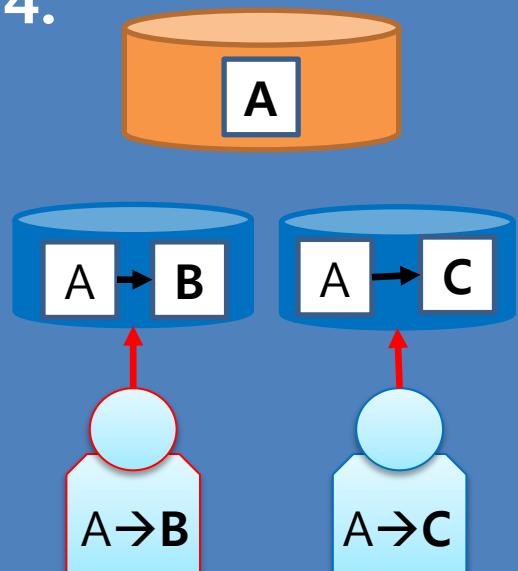
Working

(=소스 코드 변경)

2. 기존의 툴과 비교

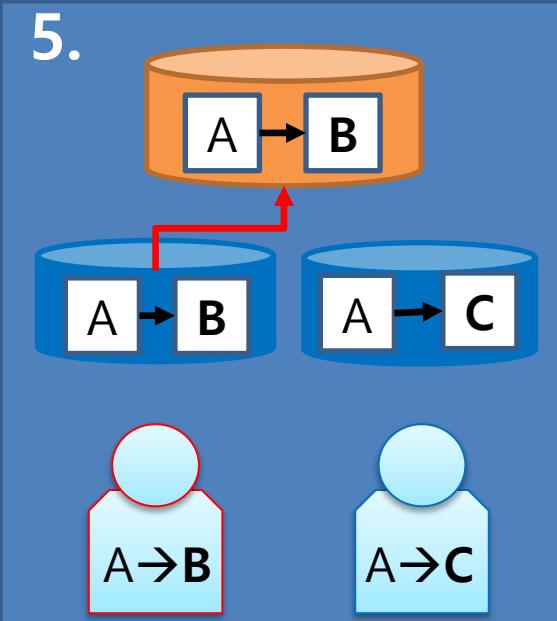
Git

4.



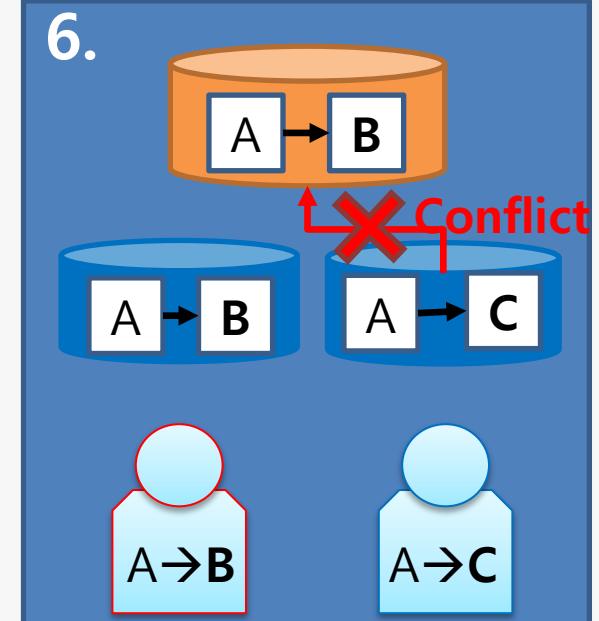
Commit
(작업공간→로컬저장소)

5.



Push
(로컬저장소→원격저장소)

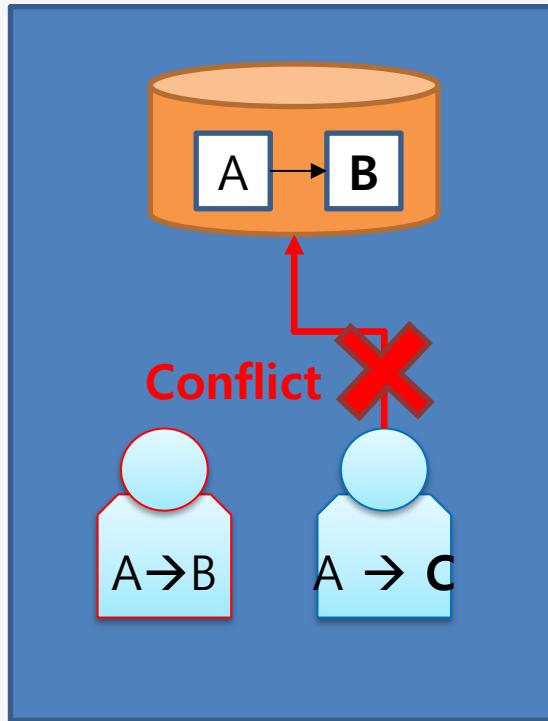
6.



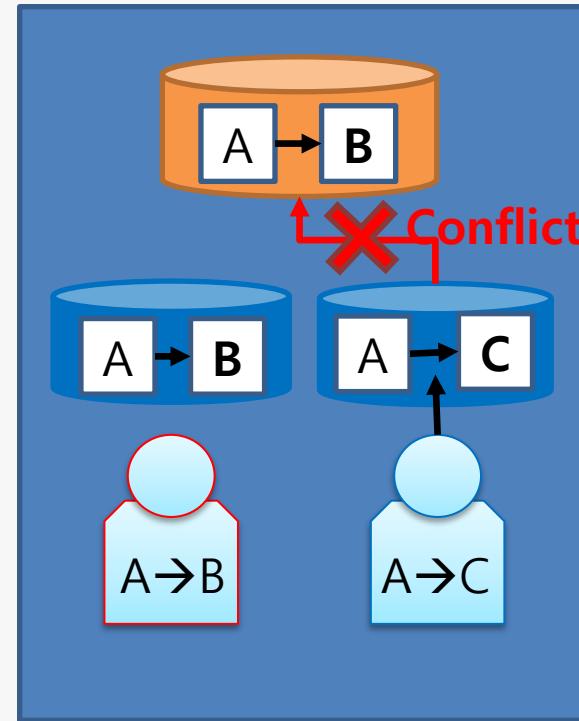
Push
(로컬저장소→원격저장소)

2. 장점 1

CVS / SVN



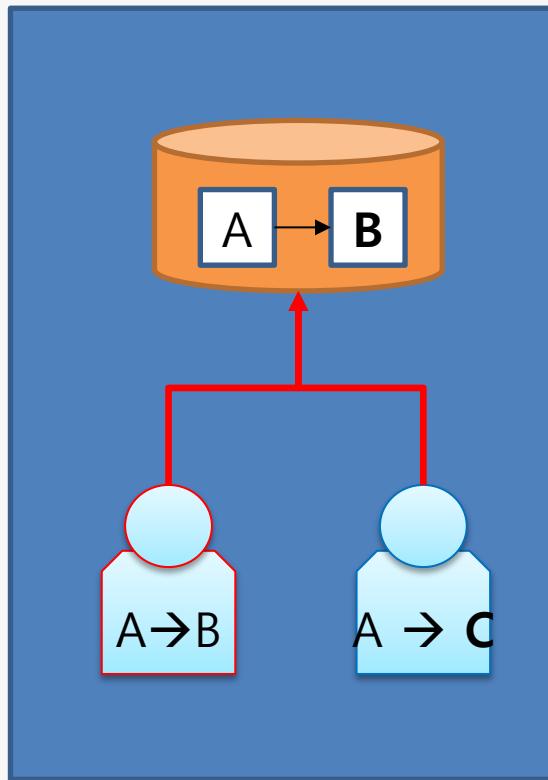
Git



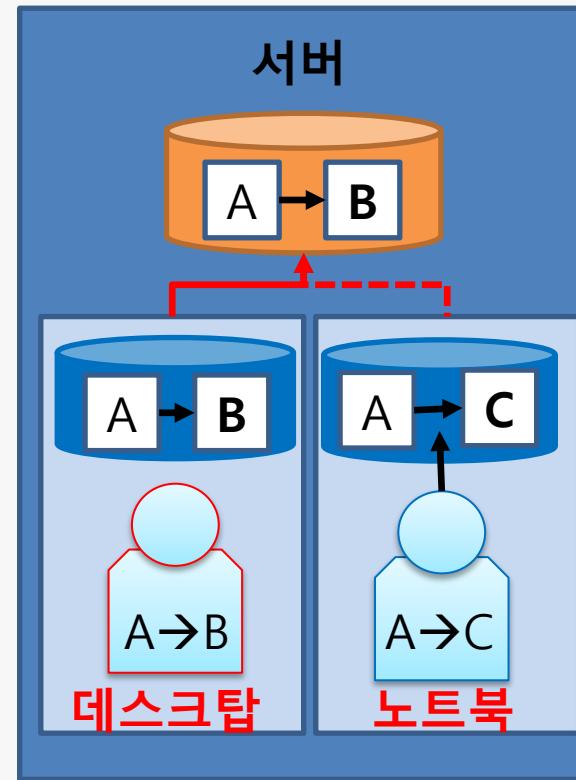
장점 1. Conflict을 해결하지 않으면 작업 내용을 버려야 한다. → 버리지 않아도 된다.

2. 장점 2

CVS / SVN



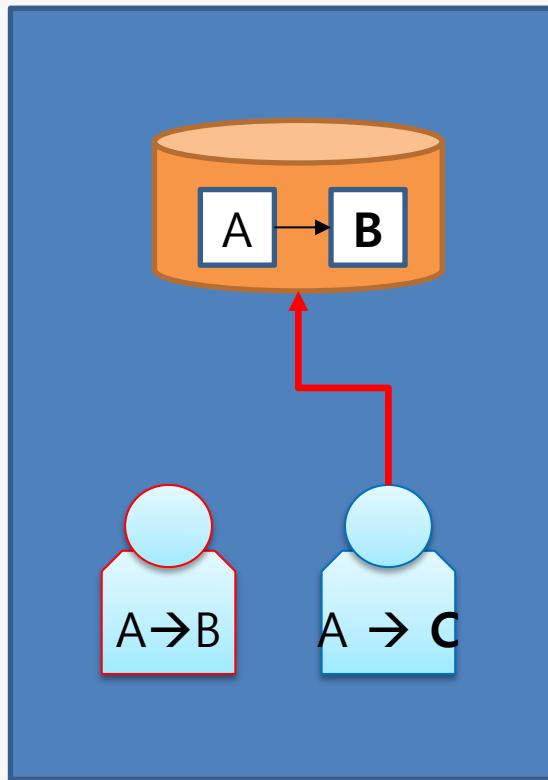
Git



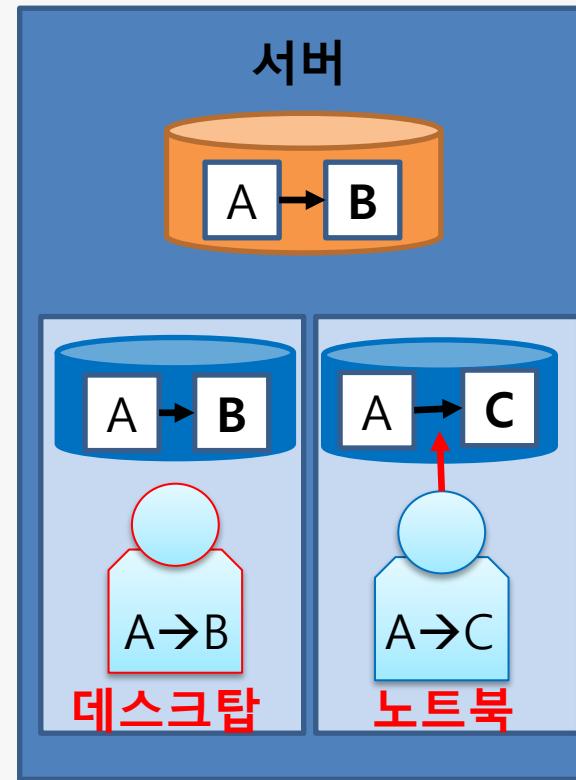
장점 2. 항상 저장소에 연결되어야 한다. → 오프라인에서도 작업이 가능하다.

2. 장점 3

CVS / SVN



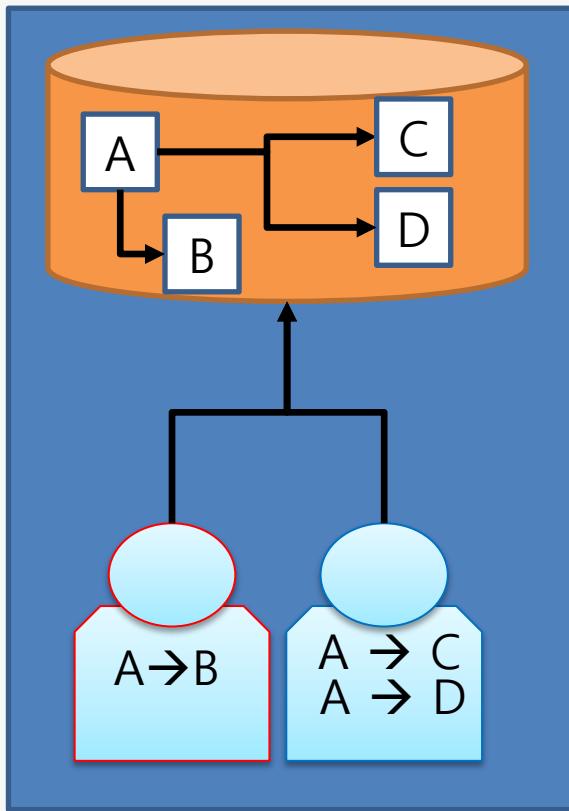
Git



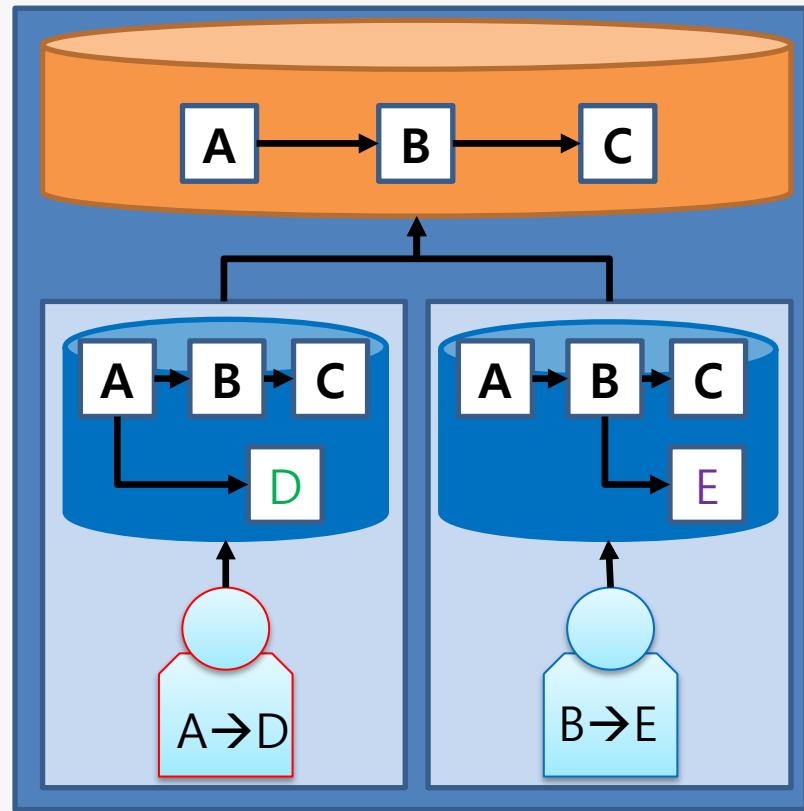
장점 3. 프로그램 규모와 인원이 늘어날수록 느리다 → 느려지지 않는다 (빠르다).

2. 장점 4

CVS / SVN



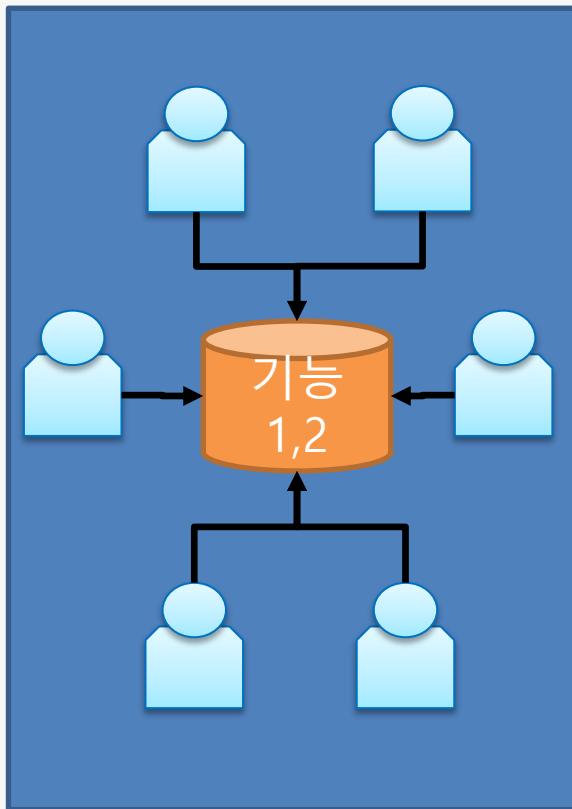
Git



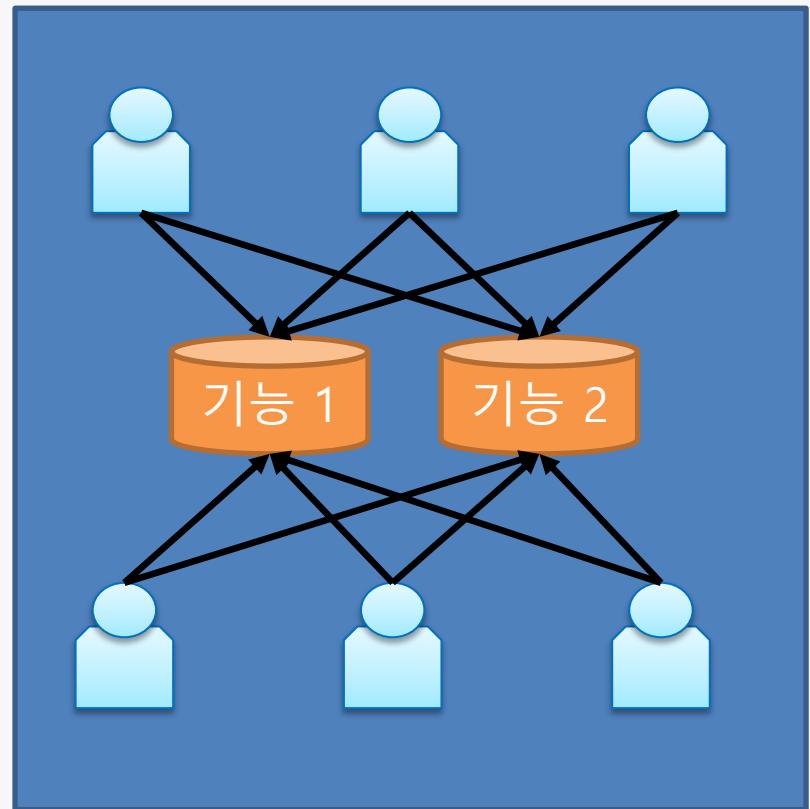
장점 4. 모든 작업 과정은 공개되어야 한다. → 여러 가지 아이디어를 테스트한 후 최종 안을 공개할 수 있다.

2. 장점 5

CVS / SVN



Git



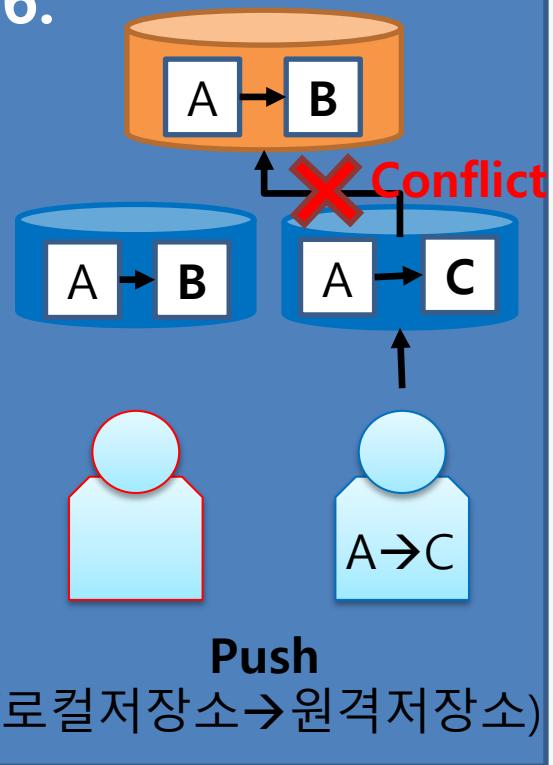
장점 5. 단일 저장소 → 다중 원격 저장소 (소스 코드의 분산 저장/관리가 가능)

2. 기존의 툴과 비교

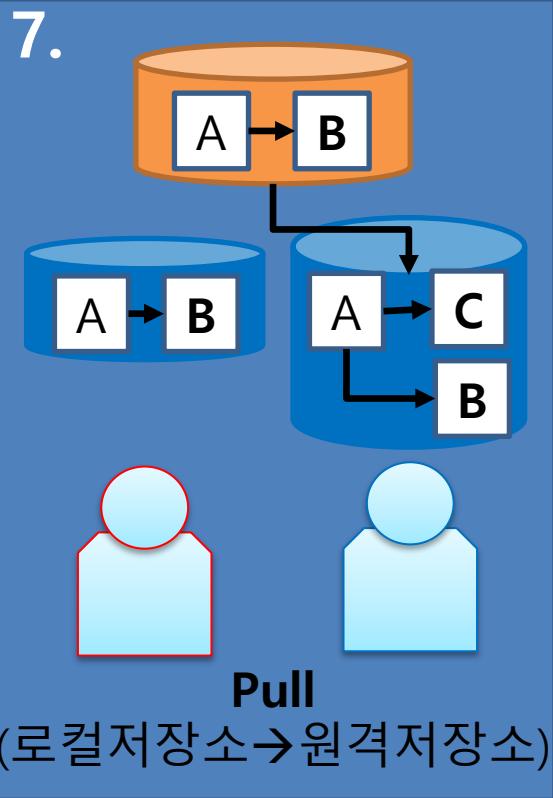
Git

* 다시 충돌(Conflict)이 발생했을 시점으로 돌아가 봅시다.

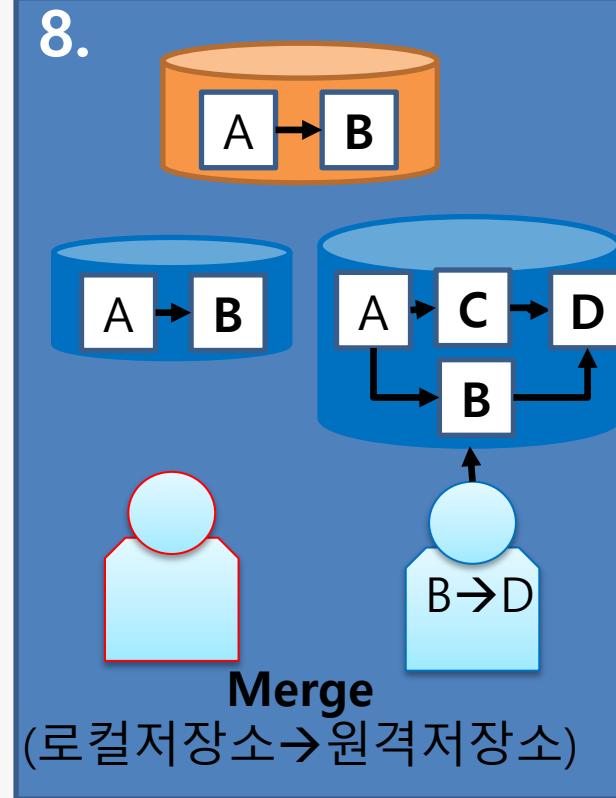
6.



7.



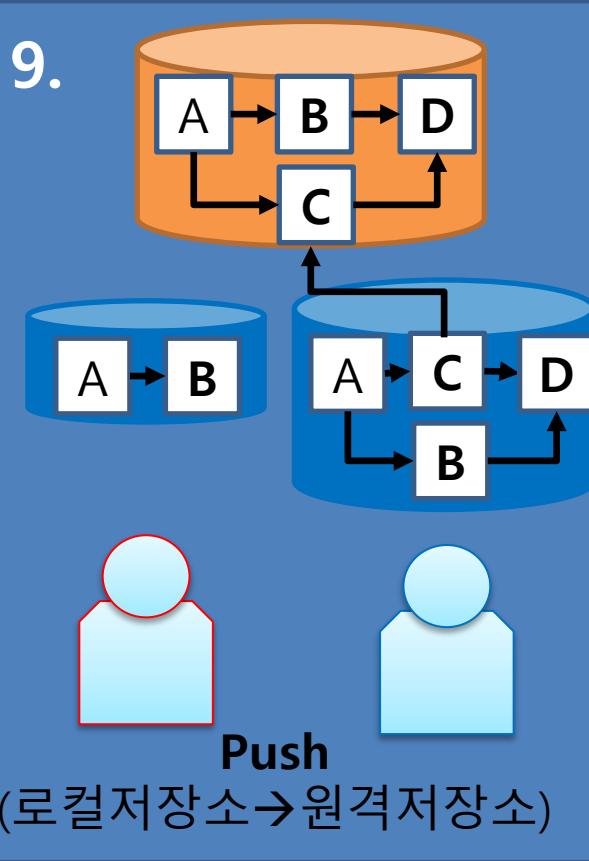
8.



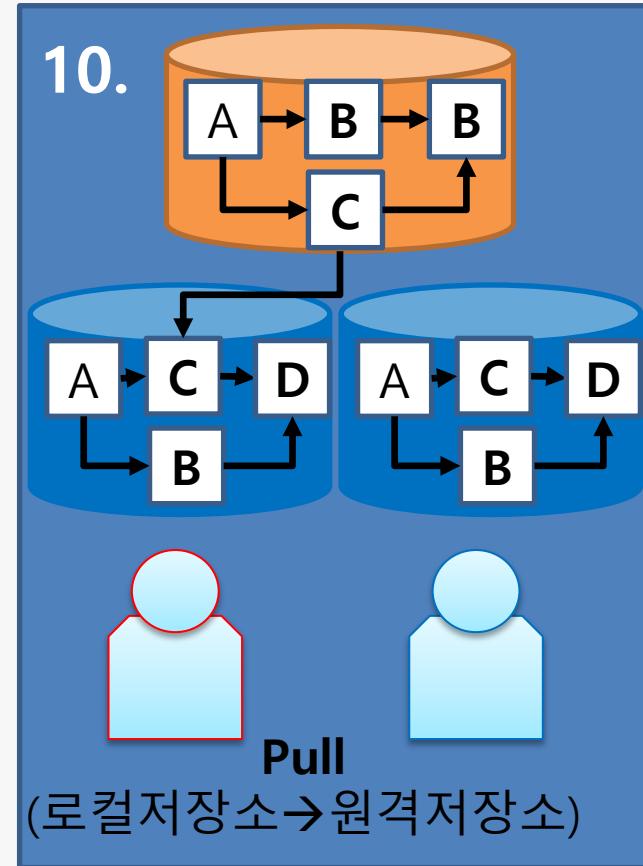
2. 기존의 툴과 비교

Git

9.



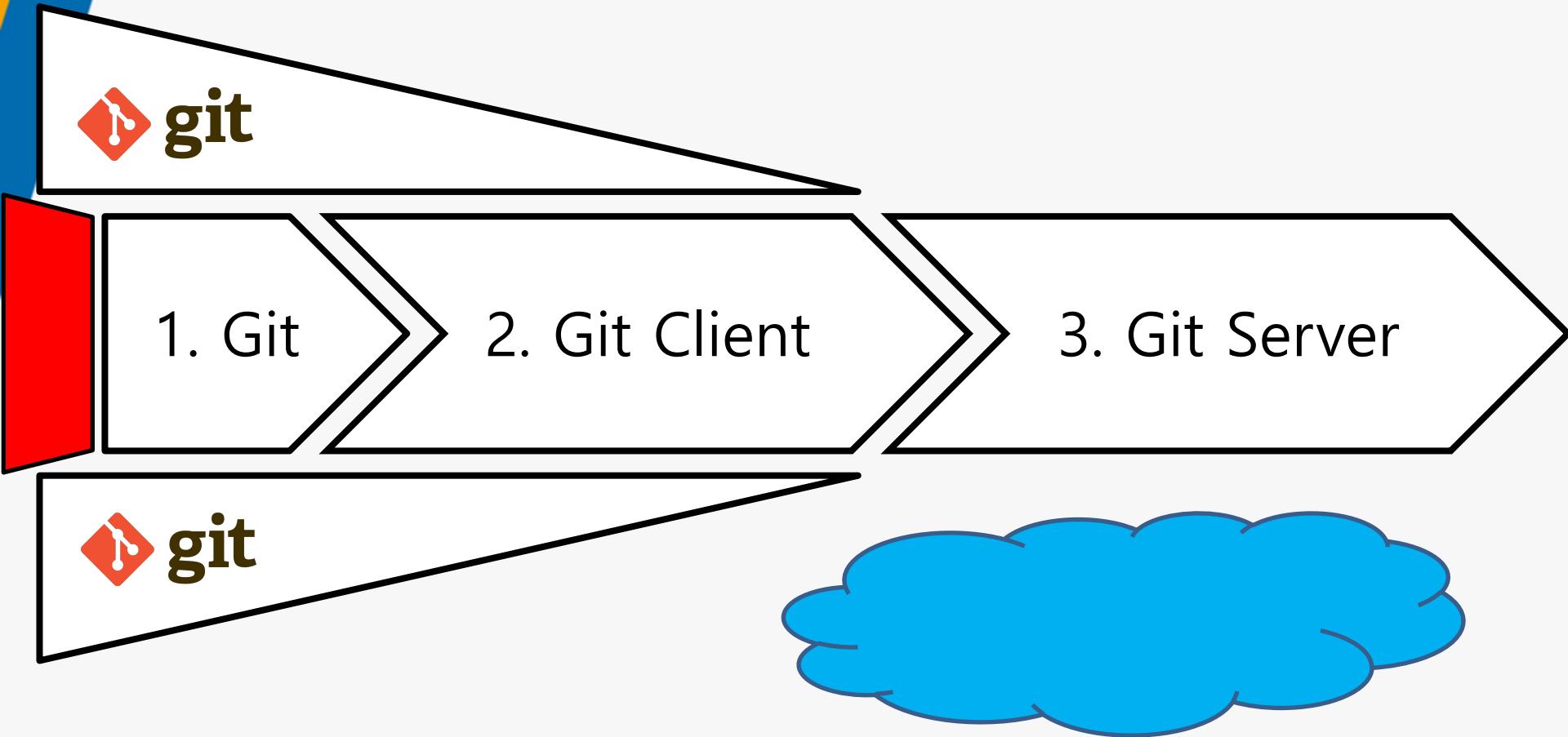
10.





3. 시작하기

3. 시작하기



3. Git

1.1 Git 인스톨러를 이용한 설치

- 구글 검색 → 다운로드 → 설치
- <https://git-scm.com>



1.2 터미널에서 설치 (패키지 관리 툴 사용)

Ubuntu: `apt install git ↵`

Red Hat: `yum install git ↵`

Mac: `brew install git ↵`

2. Git 초기 설정

`git config --global user.name "이름" ↵`

`git config --global user.email "메일주소" ↵`

3. Git client

1. Sourcetree (Windows/Mac)

- <https://www.sourcetreeapp.com/>
- Completely free for any use



The screenshot shows the Sourcetree application interface. On the left, there's a sidebar with tabs for WORKSPACE, HISTORY (which is selected), SEARCH, BRANCHES, BOOKMARKS, TAGS, REMOTES, SHELVED, and SUBREPOSITORIES. The main area displays a timeline graph of commits and a detailed list of commits. The commits are listed in chronological order from top to bottom, with columns for Commit ID, Author, Date, and Description. Some commit descriptions include links to pull requests or other details.

Commit	Author	Date	Description
b7358c7	Rahul Chhab...	Mar 3, 2016, 11:...	Removing ol...
bdb8bef	Rahul Chhab...	Feb 18, 2016, 1:3...	Merged in update-google-verification (pull request #14)
dfe975d	Tyler Tadej...	Feb 11, 2016, 2:2...	[origin/update-google-verification] Update google verificati...
3bc3290	Tyler Tadej...	Feb 11, 2016, 2:1...	Replace outdated Atlassian logo in footer with base-64 en...
dba47f9	Tyler Tadej...	Feb 11, 2016, 1:3...	Add gitignore
ff67b45	Mike Minns...	Feb 15, 2016, 11:...	Updated Mac min-spec to 10.10
72d32a8	Michael Min...	Feb 15, 2016, 10:...	Merged in hero_images (pull request #13)
246c4ff	Joel Unger...	Feb 11, 2016, 3:3...	[origin/hero_images] [hero_images] Used Tinypng to c...
9d9438c	Joel Unger...	Feb 9, 2016, 2:59...	Replacing hero images with new version of SourceTree
ce75b63	Michael Min...	Feb 15, 2016, 10:...	Merged in bug/date-https (pull request #12)
85367bb	Patrick Tho...	Jan 7, 2016, 12:2...	[origin/bug/date-https] fixed date and https errors
4f9b557	Joel Unger...	Feb 8, 2016, 3:55...	New Favicon
384e6d5	Rahul Chhab...	Feb 3, 2016, 2:09...	[origin/search-console-access] search console google ver...
6fa47a9	Mike Minns...	Dec 15, 2015, 2:0...	updated to move supported version to OSX 10.9+
8dd87bb	Mike Minns...	Nov 23, 2015, 2:2...	remove extra , when a line is skipped due to empty server
faa195e	Mike Minns...	Nov 23, 2015, 2:1...	Skip records with empty server/user id as gas rejects them
0cdfe96	Mike Minns...	Nov 23, 2015, 2:0...	corrected paths after merge
051ab1b	Mike Minns...	Nov 23, 2015, 1:5...	corrected column counting
a723bc2	Mike Minns...	Nov 23, 2015, 1:5...	Merge branch 'au2gex'
65fd580	Mike Minns...	Nov 23, 2015, 1:5...	deal with invalid instanceids
500a892	Michael Min...	Nov 23, 2015, 1:0...	Merged in au2gex (pull request #11)

3. Git client

2. Smart-git (Windows/Mac/Linux)

- <https://www.synteko.com/smartgit/>
- Free for non-commercial use



The screenshot shows the SmartGit application window. The menu bar includes Repository, Edit, View, Remote, Local, Branch, Query, Changes, Tools, Review, and Window. The toolbar contains icons for Pull, Push, Check Out, Add Tag, Add Branch, Delete, Merge, Cherry-Pick, Revert, Rebase, and Reset. On the left, a sidebar titled 'Branches' lists local branches like HEAD, master, Features, Local Branches (including deepgit-master, smartgit-17.1, smartgit-master), smartsvn, smartsync, origin, Tags, Stashes, Other Refs, and Recyclable Commits. The main area displays a 'Commits (1001)' list with detailed commit information, including author, date, message, and file changes. A commit for 'ScLogGraphLineBuilderLeftBranchDrawer: fixed size on Linux' is highlighted in blue.

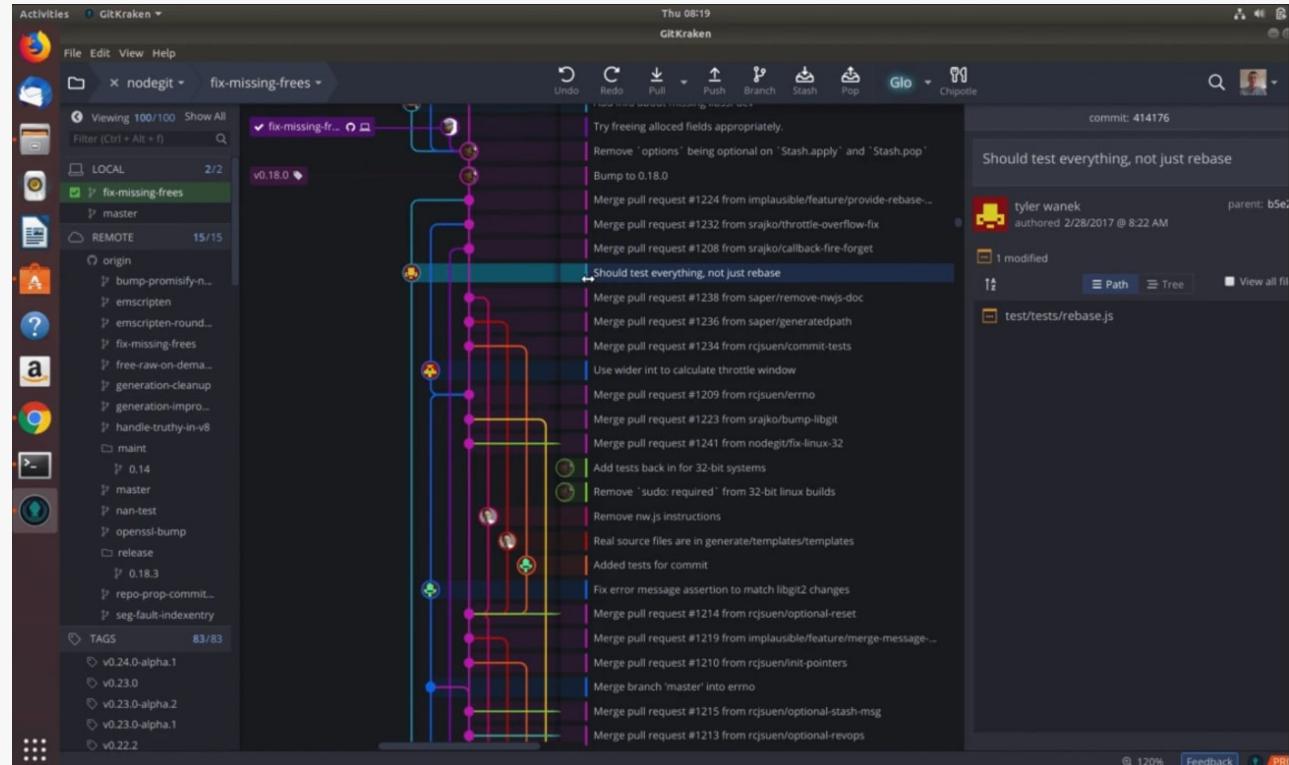
Commit Details	Date
origin[master] ScLogGraphLineBuilderLeftBranchDrawer: fixed size on Linux	TS at 2017-12-12 11:03
ScLogGraphLineBuilder*BranchDrawer: fixed cornerSize for lower resolutions	at 2017-12-12 11:03
dark and light theme: graph.connector.other	at 2017-12-12 11:03
SG-11311: Log Graph: layout optimization for single node merge-forks	at 2017-12-12 11:03
origin[deepgit-master] ScCommitGraphNodeLineRenderer fixes	MS at 2017-12-12 10:21
applied from DeepGit	at 2017-12-11 18:16
origin[smartsync] Merge branch 'master' into smartsync (r15463)	TS at 2017-12-11 15:51
Merge branch 'smartsync'	at 2017-12-11 14:46
origin[smartsync] r21021	at 2017-12-11 14:46
Commit wizard: progress text for detecting renamed files	at 2017-12-11 14:46
Merge branch 'master' into smartsync	at 2017-12-11 13:21
r20980	at 2017-12-11 12:01
SU-19230: Update: allow to be run without files selected (using directory...)	at 2017-12-11 12:01
SU-19279: Update toolbar button: add Update More to popup	at 2017-12-11 12:00
added smartsync-light-theme.properties	at 2017-12-11 11:46

3. Git client

KERI

3. Gitkraken (Windows/Mac/Linux)

- <https://www.gitkraken.com/>
- Free for non-commercial use



3. Git Server

1. GitHub

- Free for private repositories (< 3 users)
- <https://github.com>
- Microsoft.

2. Bitbucket

- Free for private repositories (< 5 users)
- <https://bitbucket.org>
- Atlassian

3. My server

- Local storage (HDD, SDD)
- Network attached storage (NAS)

3. Git Server 연결하기

1. Git Server 주소 얻기

The screenshot shows a GitHub repository page for 'aifrenz / membership'. At the top, there are buttons for 'Watch' (1), 'Star' (3), 'Fork' (63), and 'Code'. Below the header, there are tabs for 'Issues' (0), 'Pull requests' (2), 'Projects' (0), 'Wiki', and 'Insights'. A message says 'No description, website, or topics provided.' Below this, stats show '181 commits', '1 branch', '0 releases', and '54 contributors'. A dropdown for 'Branch: master' and a 'New pull request' button are visible. On the right, there's a 'Clone or download' button (circled with a red box and labeled 1) and a 'Clone with HTTPS' section (labeled 2). The URL 'https://github.com/aifrenz/membership' is shown in the 'Clone with HTTPS' field, with a red arrow pointing to it.

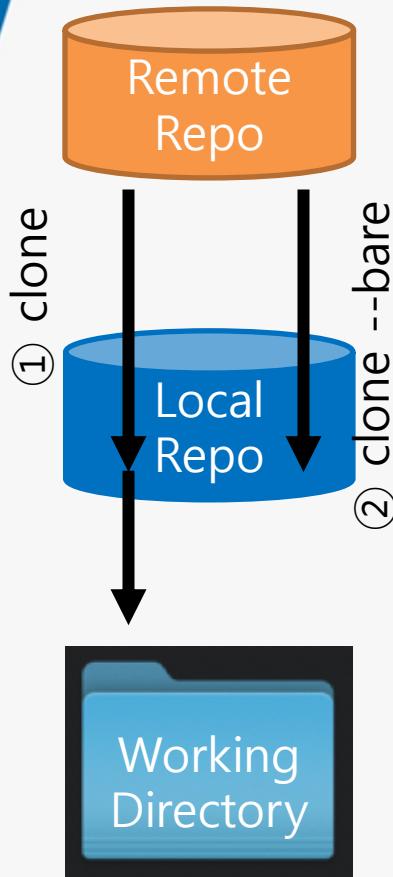
2. Git Server와 연결하기

③ git clone https://github.com/aifrenz/membership.git ↵

※ 원격 저장소를 로컬 저장 장치에 복사하는 방식으로 로컬 저장소를 생성

4. 사용하기

4. 사용하기



① 원격 저장소를 복사하면서 지역 저장소와 작업공간을 생성하기

```
git clone (원격 저장소 주소) ↪
```

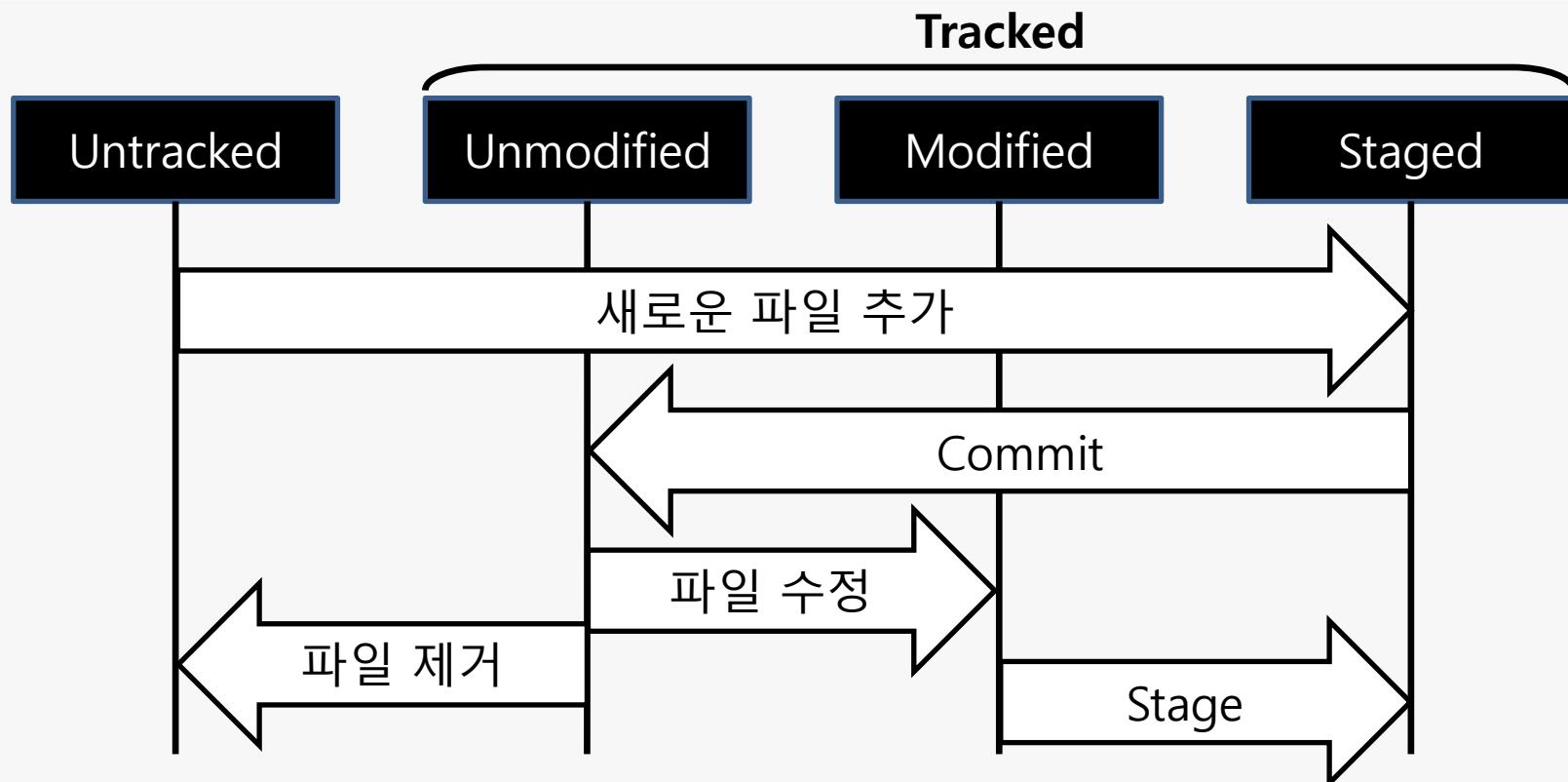
```
git clone https://github.com/aifrenz/membership.git ↪
```

② 원격 저장소를 복사해오면서 지역저장소만 생성하기

```
git clone (원격 저장소 주소) --bare ↪
```

```
git clone --bare https://github.com/aifrenz/membership.git ↪
```

4. 사용하기



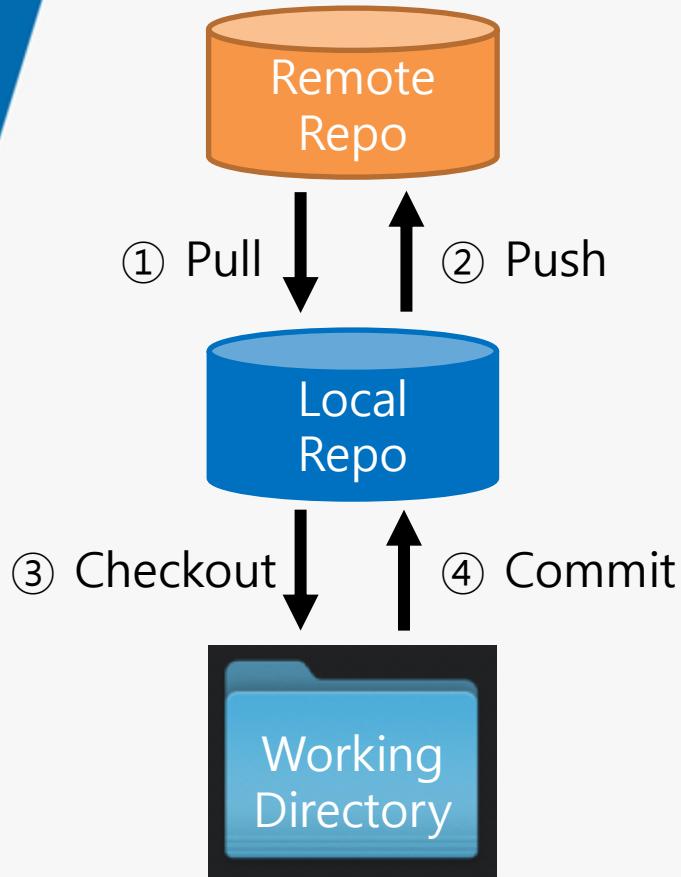
상태 출력: `git status ↵`

파일 추가: `git add 파일명 ↵`

파일 제거: `git rm 파일명 ↵`

파일 이동: `git mv 파일명1 파일명2 ↵`

4. 사용하기



① 원격 저장소 → 지역 저장소

git pull ↳

git pull (remote) ↳

git pull (remote) (branch) ↳

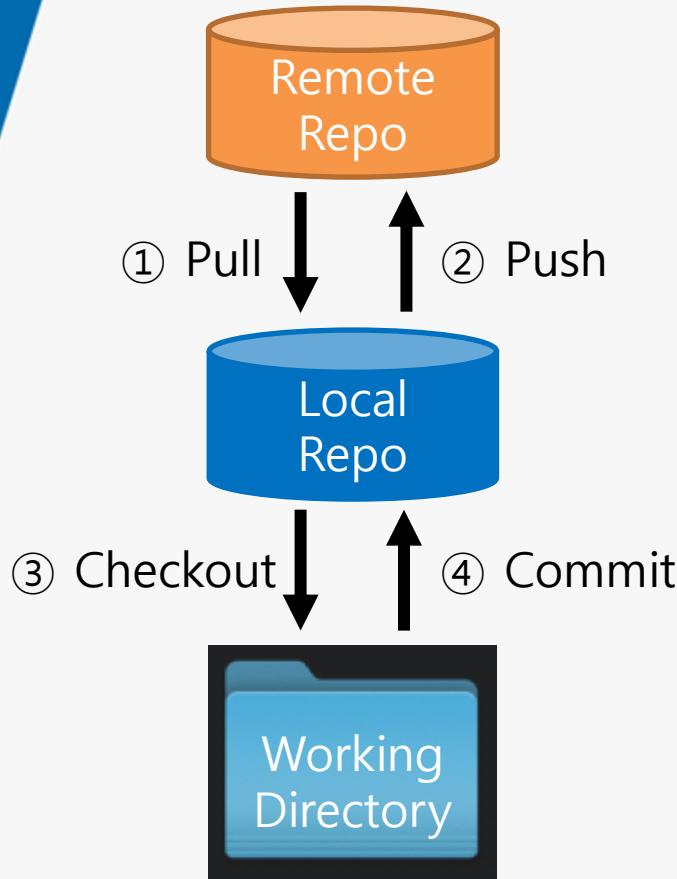
② 지역 저장소 → 원격 저장소

git push ↳

git push (remote) ↳

git push (remote) (branch) ↳

4. 사용하기



③ 지역 저장소 → 작업 공간

브랜치 이동: `git checkout (branch) ↵`

커밋 히스토리 이동: `git checkout (hashcode) ↵`

그래프	커밋	작성자	날짜	설명
○	*	*	오늘 오전 8:08	Uncommitted changes
●	02b14de	sunkwon <sunkwon...>	2019. 5. 13. 오후 1:42	↳ develop ↳ origin/deve

해시코드

sourcetree

④ 작업 공간 → 지역 저장소

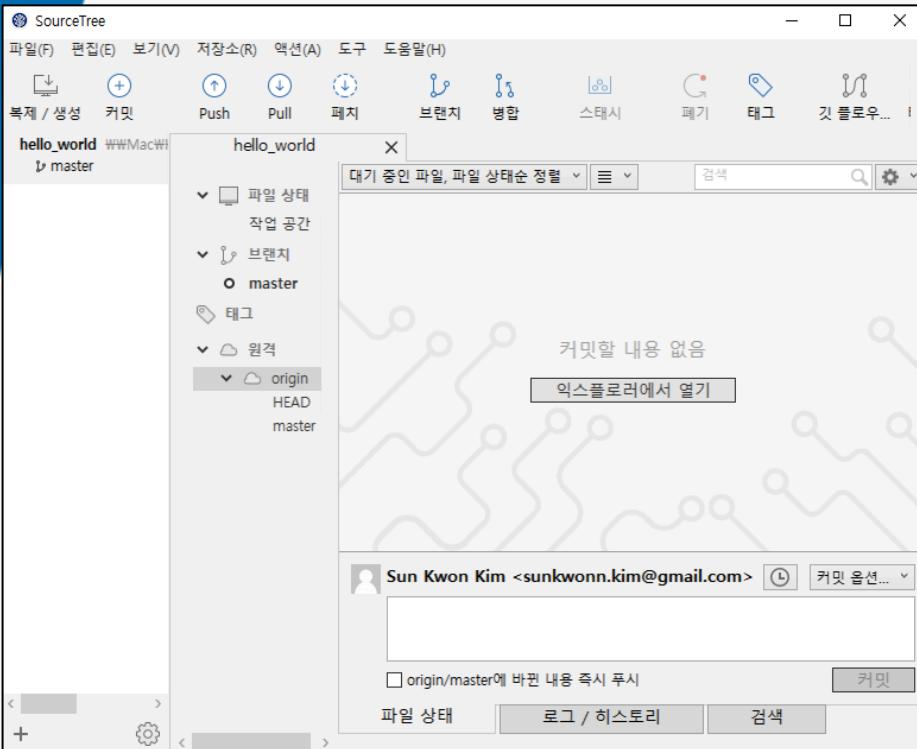
`git commit -m "설명/기록" ↵`

Source Tree (Git)의

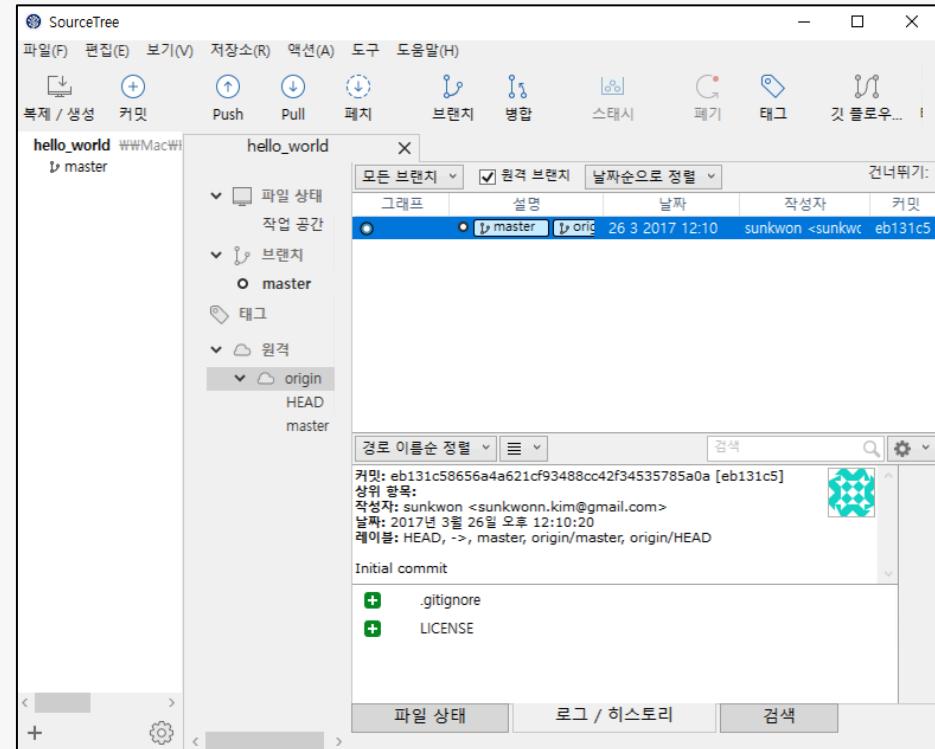
5. GUI 클라이언트

5. GUI 클라이언트

파일 상태 뷰(CTRL + 1)



로그 보기 뷰(CTRL + 2)

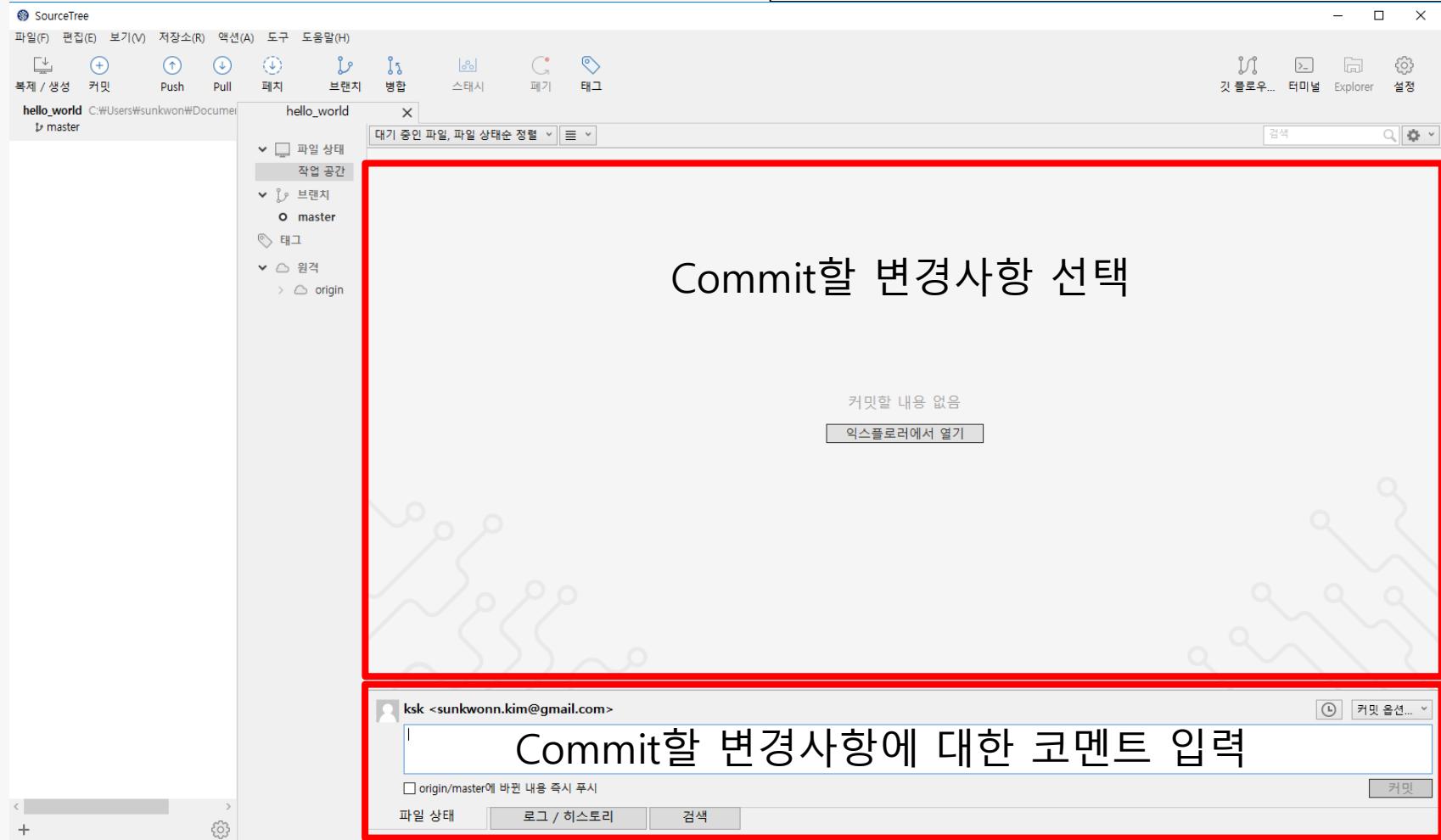


- Commit하기 위해 필요한 기능이 모여 있음
- Commit할 내용 선택 및 코マン트 입력
- 소스 코드 충돌 (Conflict) 시 해결하는 기능

- 소스 코드 히스토리를 조작하는 모든 기능
- 히스토리(로그) 조회/이동
- 브랜치 생성 / 합치기 (Merge)

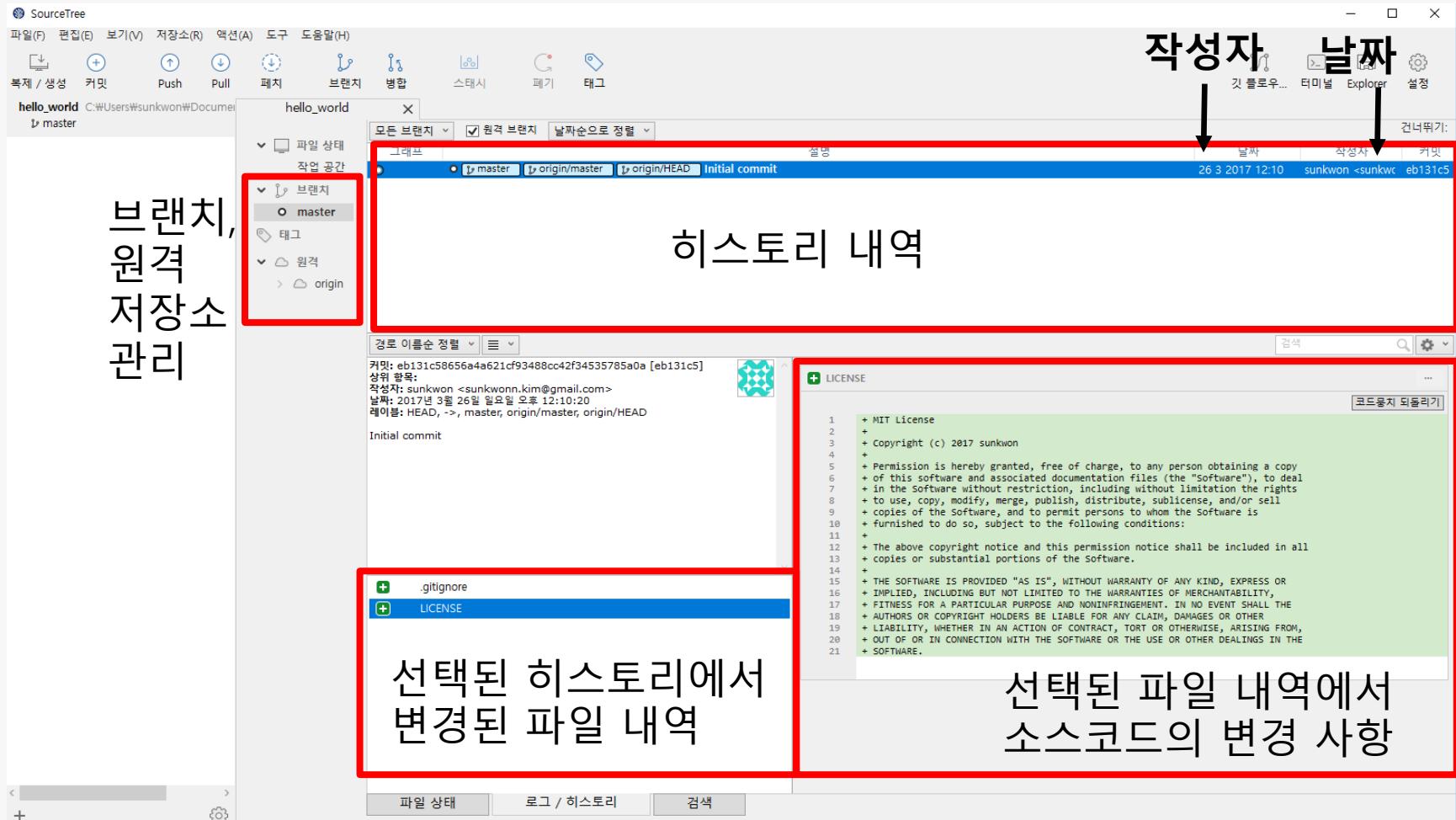
5. GUI 클라이언트

1. 파일 상태 뷰(CTRL + 1)



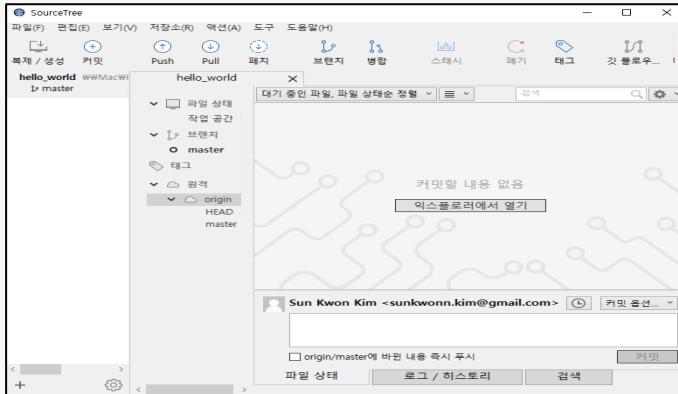
5. GUI 클라이언트

히스토리 뷰 (단축키: CTRL + 2)



5. GUI 클라이언트

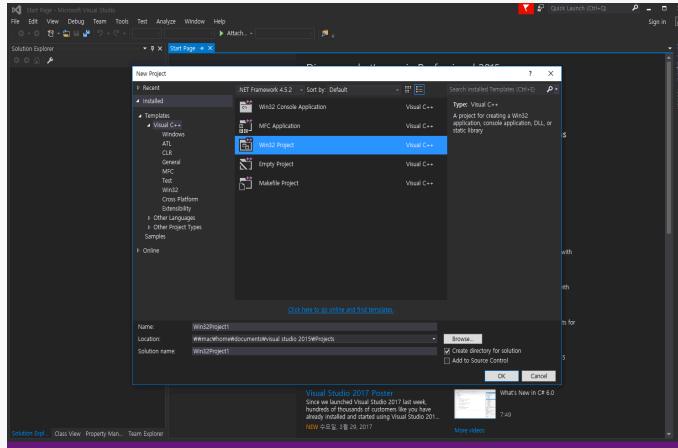
1. 익스플로어에서 열기 클릭



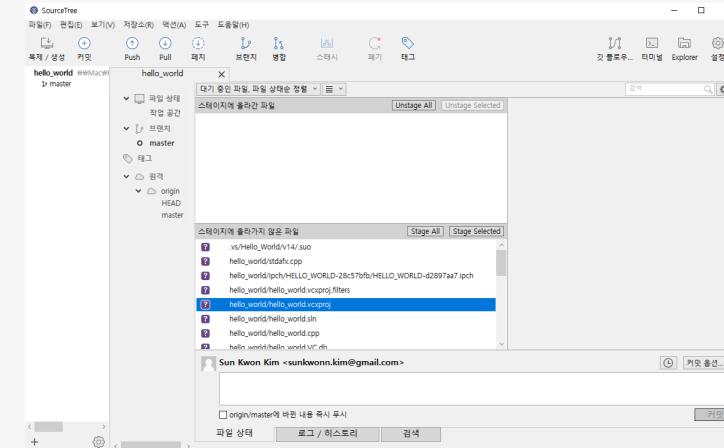
2. 작업 공간



3. 작업 공간에 프로젝트 생성



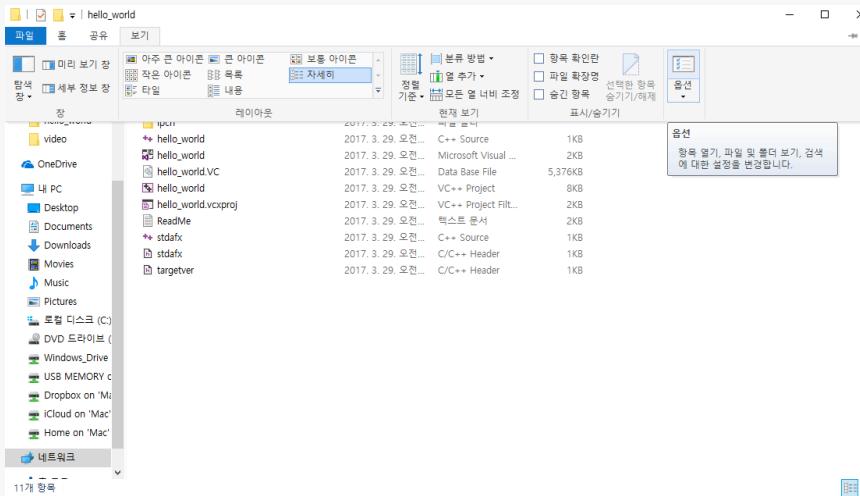
4. 파일 상태 뷰에서 변경사항 체크



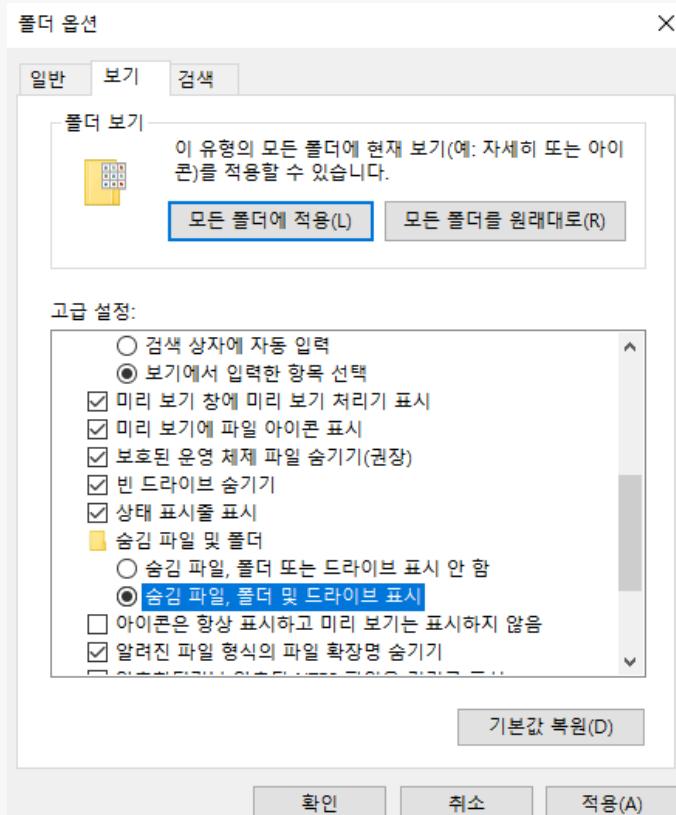
5. GUI 클라이언트

KERI

5. 탐색기 → 보기 → 옵션 선택



6. 숨김파일이 보이도록 변경

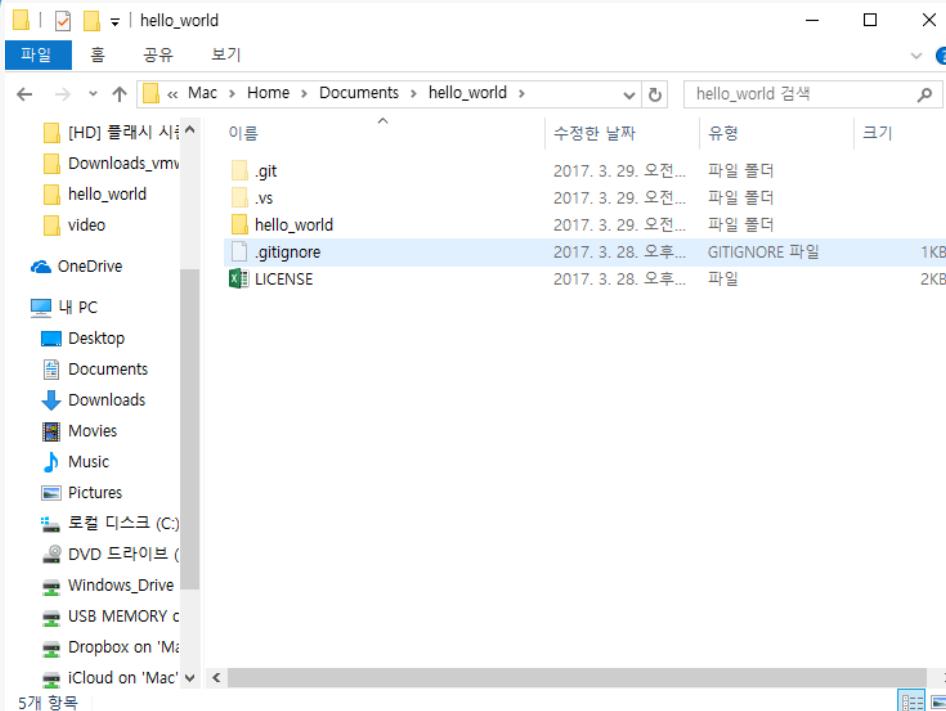


KERI

KOREA ELECTROTECHNOLOGY RESEARCH INSTITUTE

5. GUI 클라이언트

7. ".gitignore" 파일을 텍스트 편집기로 열기



.gitignore - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
# Compiled Object files
*.slo
*.lo
*.o
*.obj

# Precompiled Headers
*.gch
*.pch

# Compiled Dynamic Libraries
*.so
*.dylib
*.dll

# Fortran module files
*.mod
*.smod

# Compiled Static Libraries
*.lai
*.la
*.a
*.lib

# Executables
*.exe
*.out
*.app
```

5. GUI 클라이언트

8. 구글에서 ".gitignore + 사용 프로그래밍 환경"을 검색

Google search results for ".gitignore visual studio 2015". The results page shows a red box highlighting the top result, which is a GitHub link to a .gitignore file for Visual Studio 2015.

.gitignore visual studio 2015

전체 동영상 이미지 뉴스 지도 더보기 설정 도구

검색결과 약 355,000개 (0.37초)

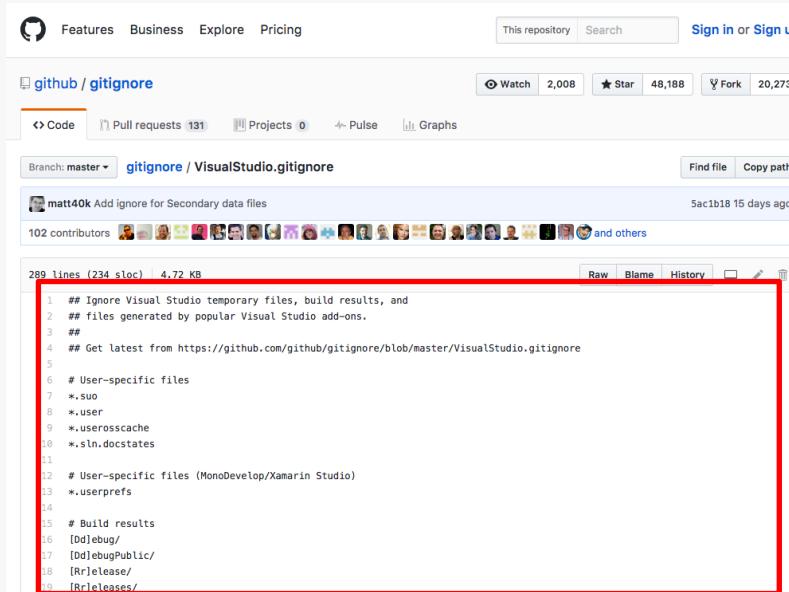
gitignore/VisualStudio.gitignore at master · github/gitignore · GitHub
<https://github.com/github/gitignore/blob/.../VisualStudio.gitignore> ▾ 이 페이지 번역하기
Get latest from <https://github.com/github/gitignore/blob/master/VisualStudio.gitignore>. # User-specific ...
[L]og/. # Visual Studio 2015 cache/options directory .vs/.

Example .gitignore file I use for C# projects · GitHub
<https://gist.github.com/kmorcinek/2710267> ▾ 이 페이지 번역하기
Example .gitignore file I use for C# projects. Raw .gitignore ... -OutFile .gitignore **/*.sln,ide, # Visual Studio temp something .vs/, # VS 2015+, *.vc,vc.opendb.

Ignore files in your Git repo | Team Services & TFS - Visual Studio
<https://www.visualstudio.com/en-us/docs/git/tutorial/ignore-files> ▾ 이 페이지 번역하기
Customize your .gitignore(으)로 이동 - Git starts ignoring these files as soon as the .gitignore is updated, but be sure to ... Visual Studio 2015 & 2017.

5. GUI 클라이언트

9. 검색된 내용을 ".gitignore" 파일에 붙여넣기

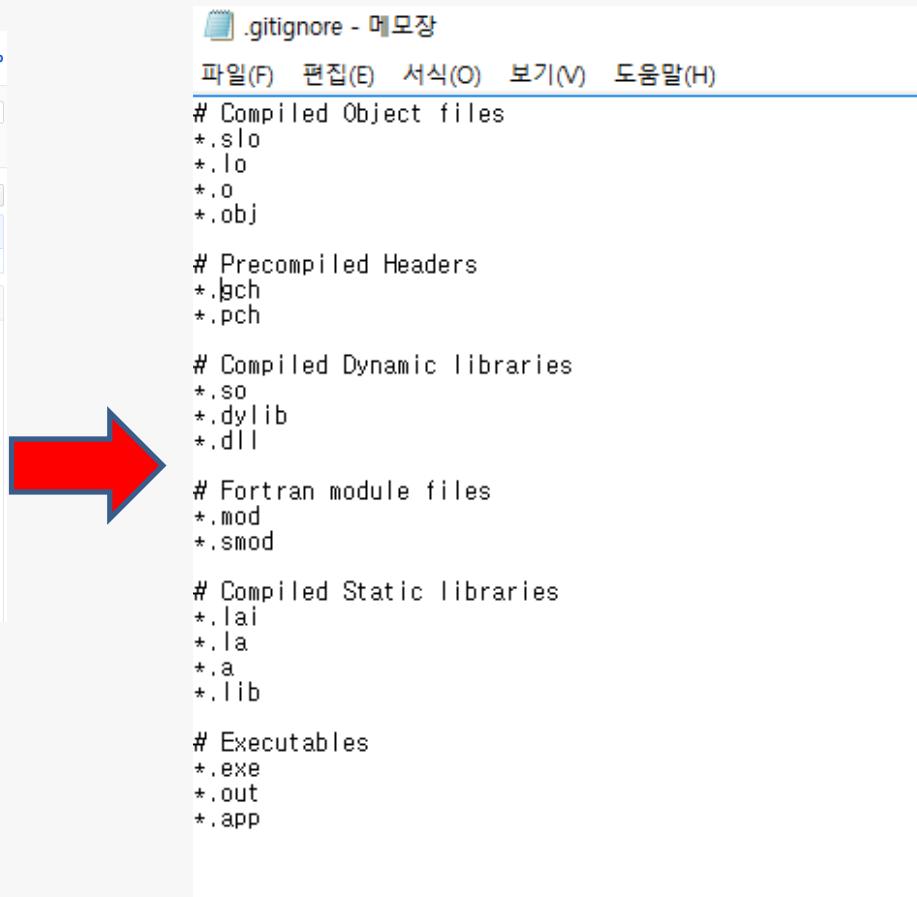


A screenshot of a GitHub repository page for ".gitignore". The repository has 2,008 stars and 20,273 forks. The "Code" tab is selected, showing the "VisualStudio.gitignore" file. A red box highlights the first few lines of the code, which are standard ignore patterns for Visual Studio temporary files. A large red arrow points from this highlighted area to the right-hand side where the clipboard content is being pasted.

```
## Ignore Visual Studio temporary files, build results, and
## files generated by popular Visual Studio add-ons.
##
## Get latest from https://github.com/github/gitignore/blob/master/VisualStudio.gitignore
#
# User-specific files
*.suo
*.user
*.userosscache
*.sln.docstamps

# User-specific files (MonoDevelop/Xamarin Studio)
*.userprefs

# Build results
[D]ebug/
[D]ebugPublic/
[R]elease/
[R]eleases/
```



A screenshot of a Windows clipboard titled ".gitignore - 메모장". The clipboard contains the same .gitignore code as the GitHub repository, including the header and various ignore patterns for different file types like object files, precompiled headers, dynamic libraries, and executables. A large red arrow points from the GitHub screenshot to this clipboard view, indicating the transfer of the code.

```
# Compiled Object files
*.slo
*.lo
*.o
*.obj

# Precompiled Headers
*.gch
*.pch

# Compiled Dynamic Libraries
*.so
*.dylib
*.dll

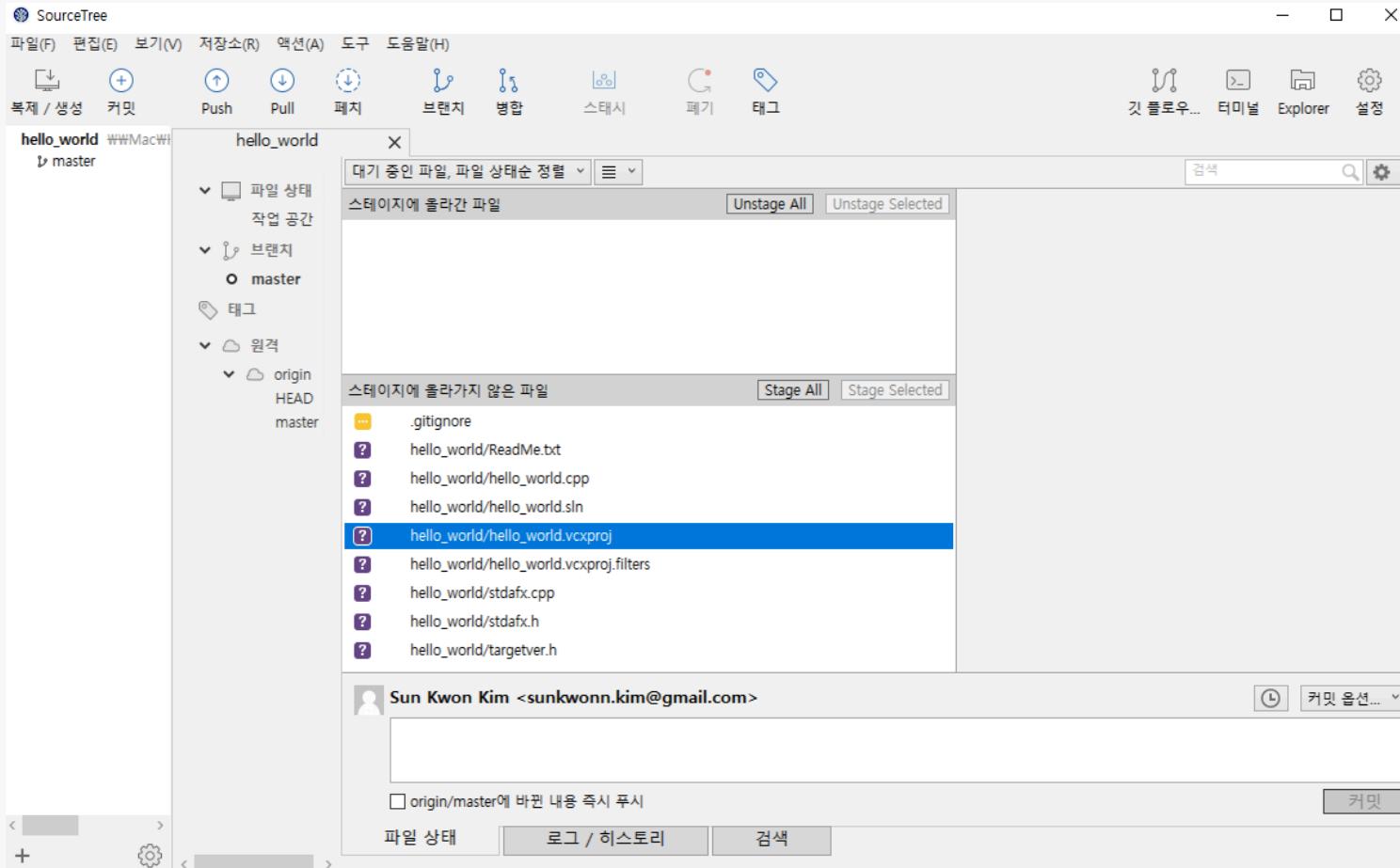
# Fortran module files
*.mod
*.smod

# Compiled Static Libraries
*.la
*.la
*.a
*.lib

# Executables
*.exe
*.out
*.app
```

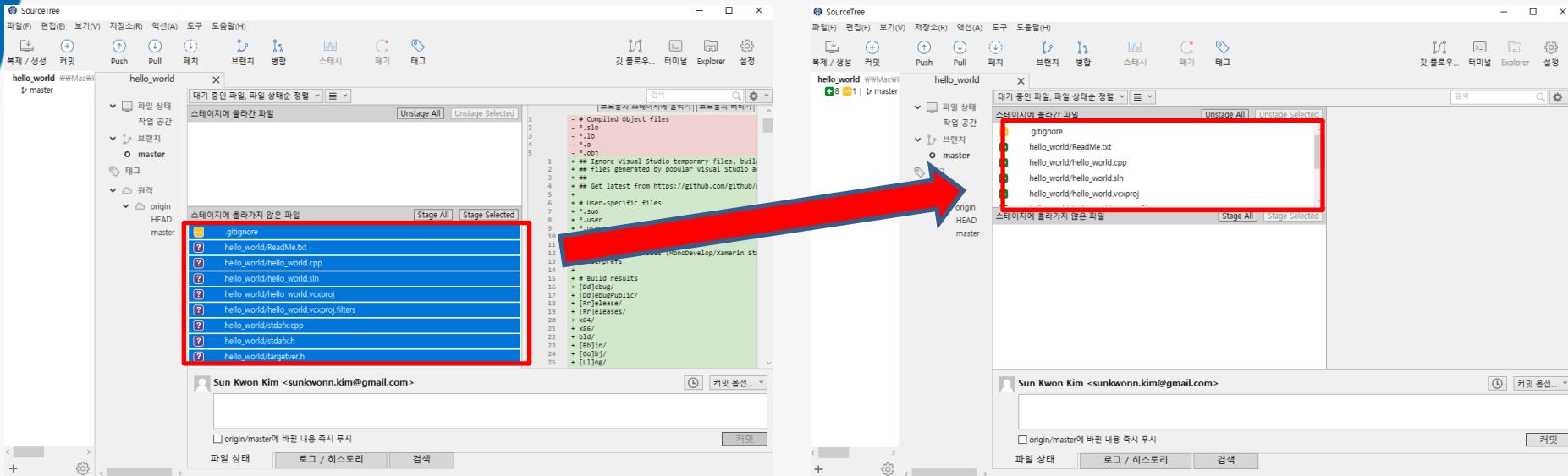
5. GUI 클라이언트

10. 저장소에 저장할 필요가 없는 불필요한 파일은 자동 무시됨



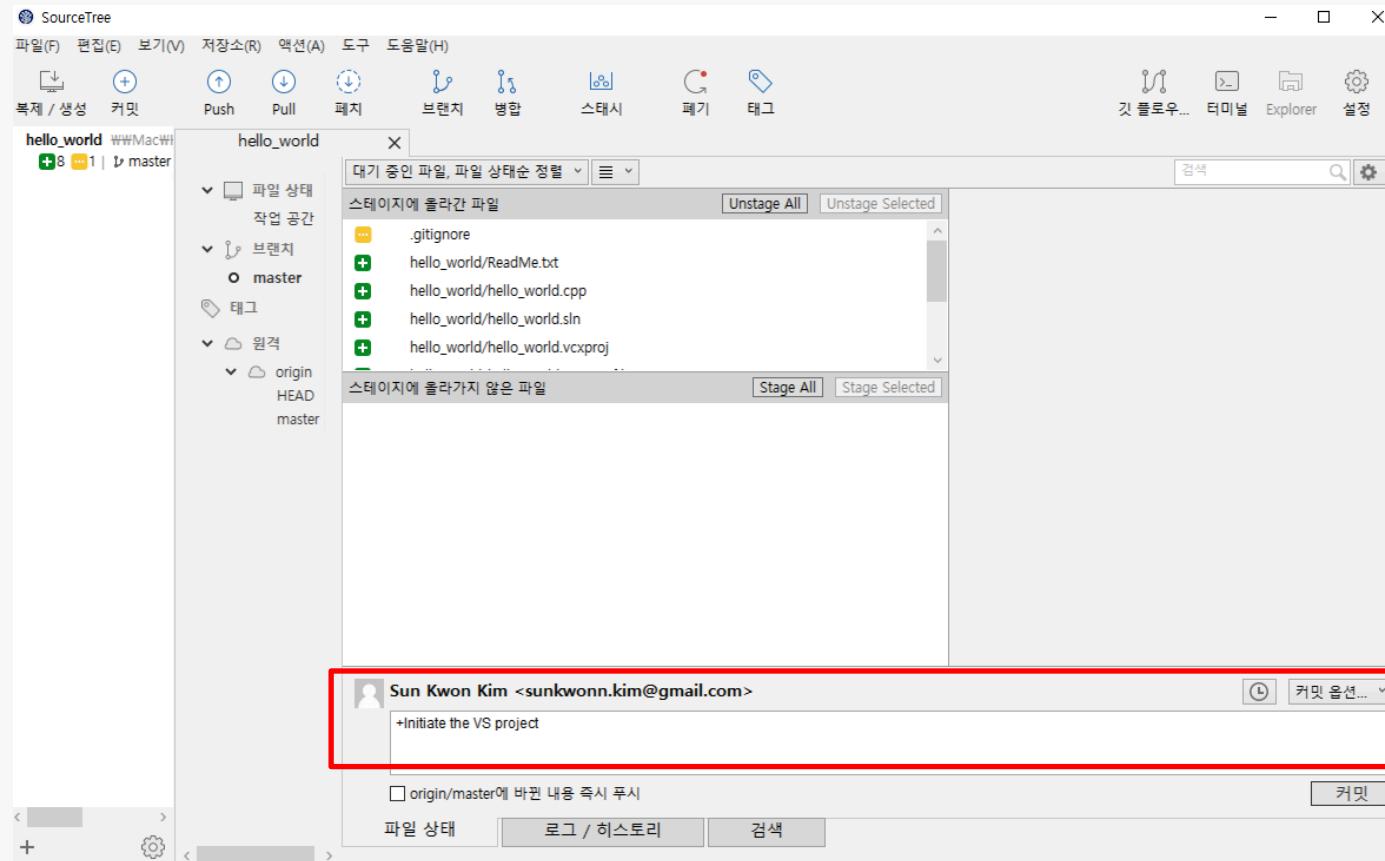
5. GUI 클라이언트

11. "Stage all" 또는 "Stage Selected" 버튼을 눌러서 로컬 저장소에 저장할 파일 상태(스테이지)로 넣음



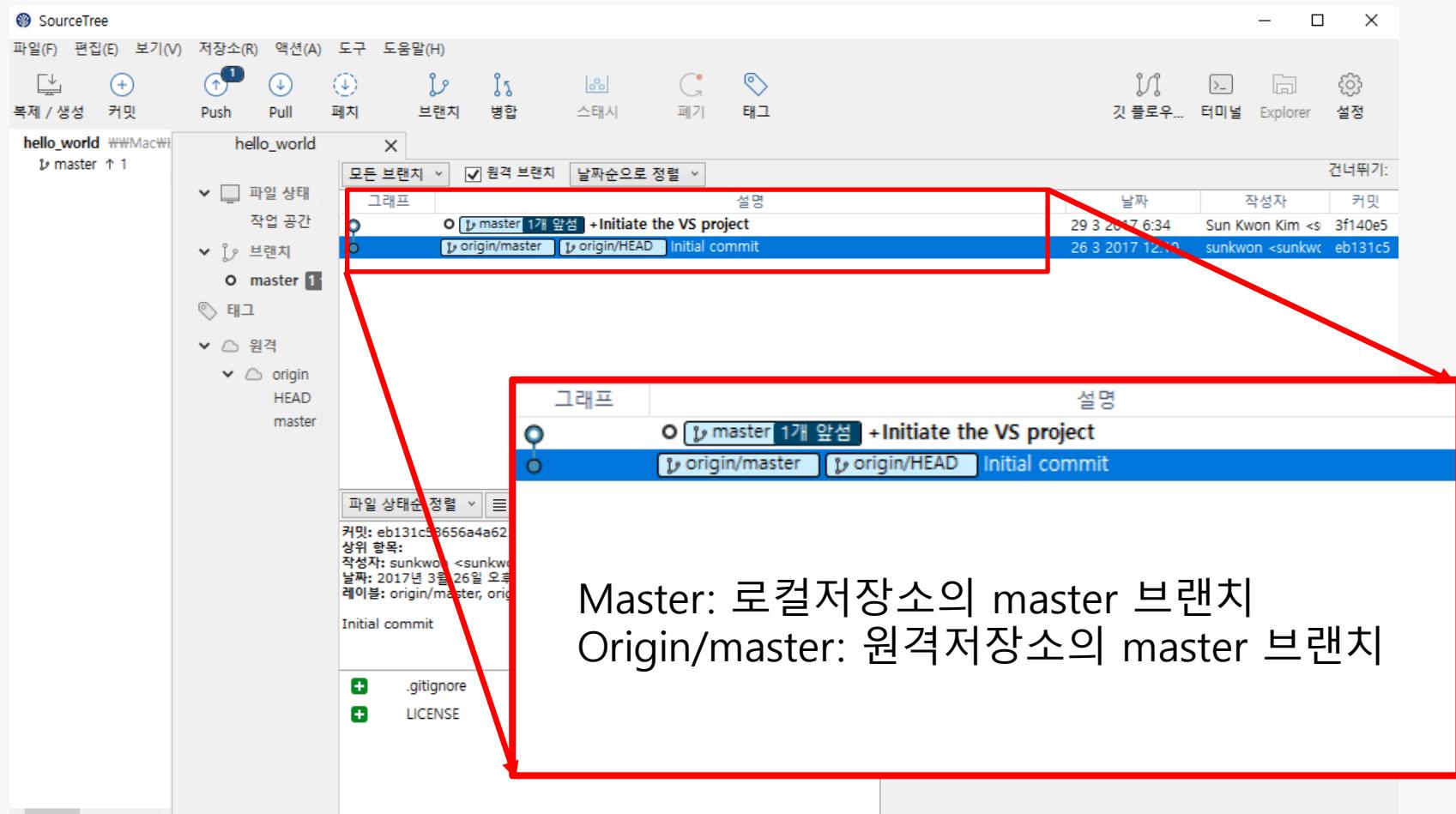
5. GUI 클라이언트

12. Commit할 내용에 대해서 기술한 다음 "Commit (커밋)" 버튼을 클릭



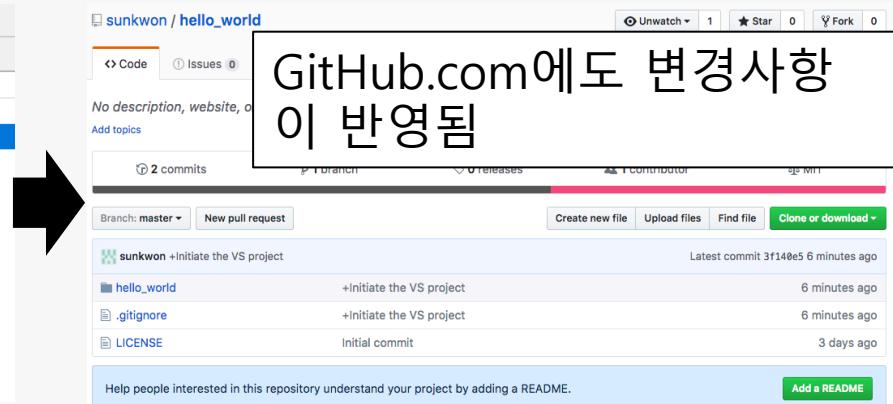
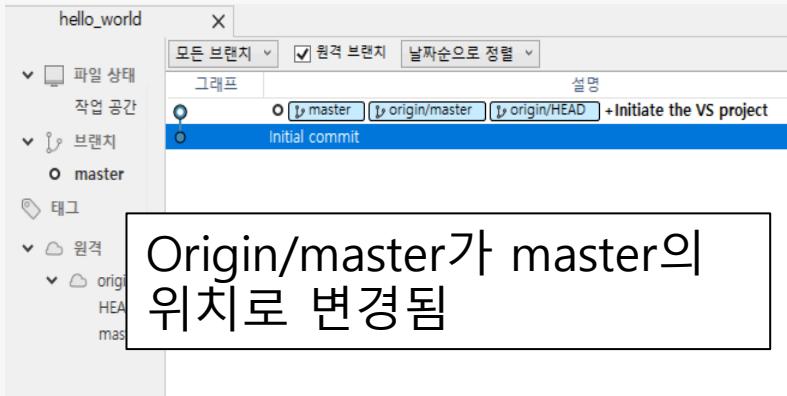
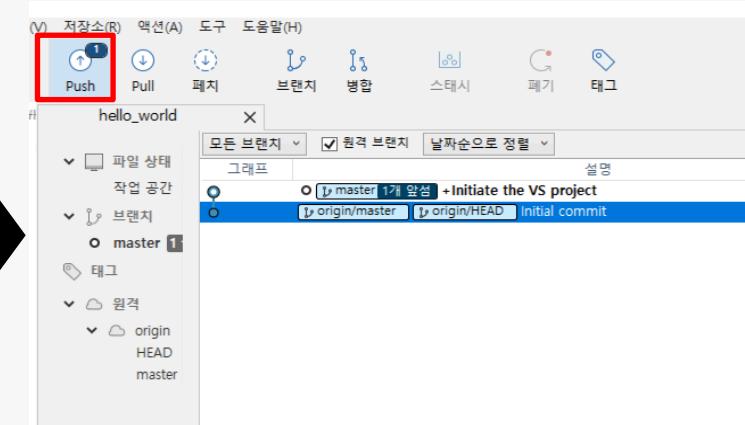
5. GUI 클라이언트

13. Commit할 내용에 대해서 기술한 다음 "Commit (커밋)" 버튼을 클릭



5. GUI 클라이언트

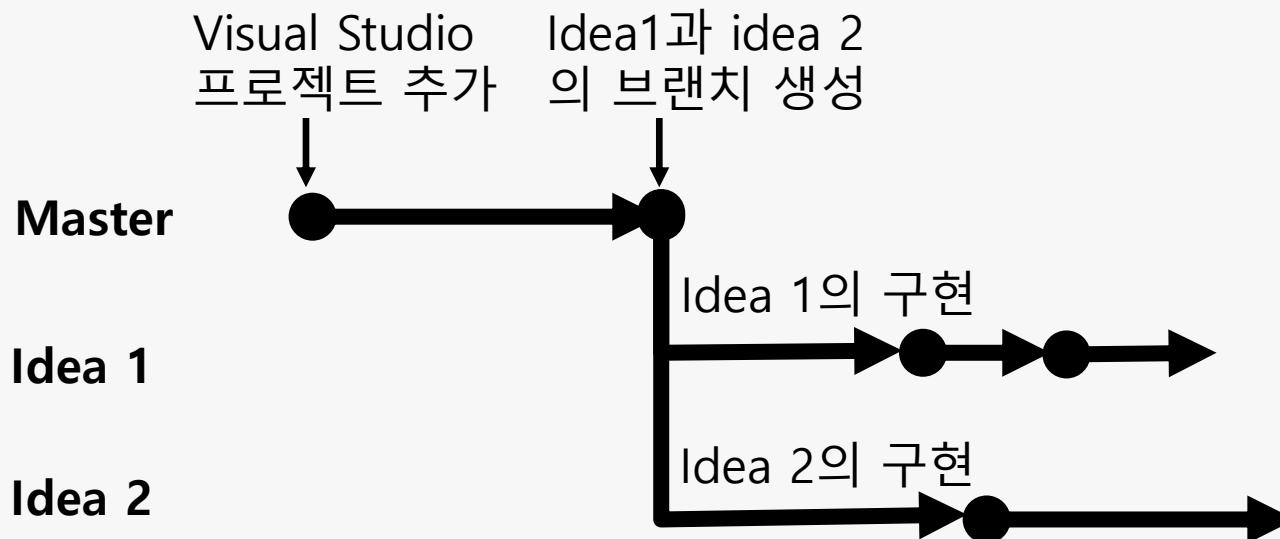
14. "Push (푸시)" 버튼을 눌러서 원격저장소 (GitHub)에 변경사항을 업로드



5. GUI 클라이언트: 브랜치 생성 (Branch)

- 브랜치 == 하나의 대표되는 흐름
- 새로운 아이디어나 기능을 구현해야 할 때에는 별도의 브랜치를 생성하여 작업을 하자.

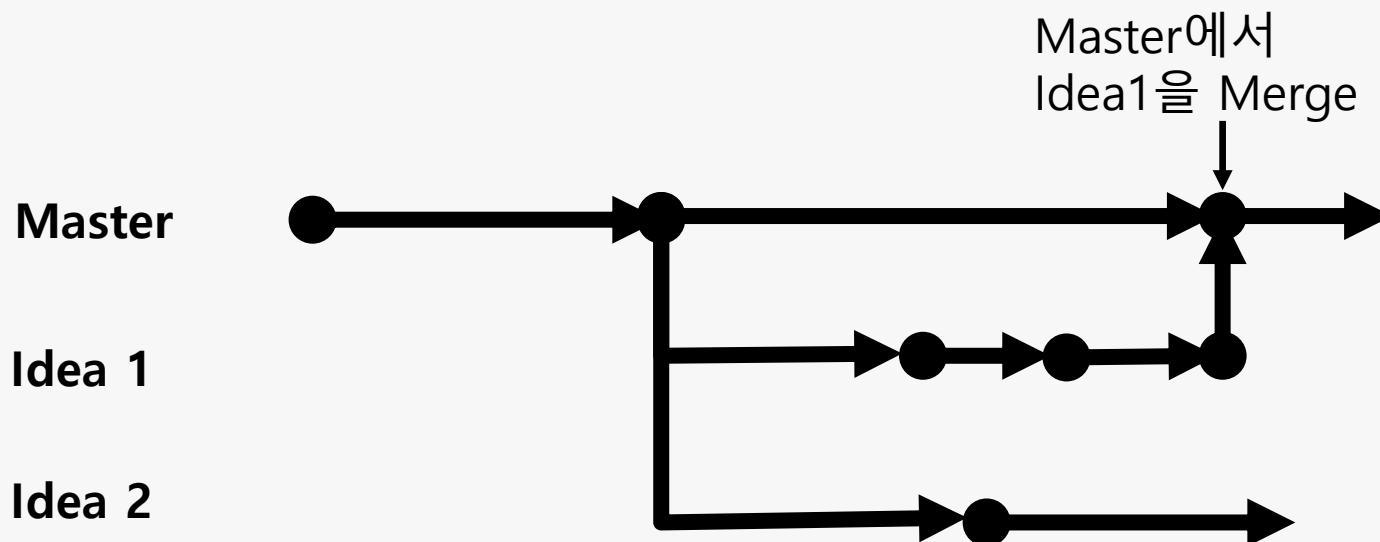
(예시) 어떤 문제를 해결하기 위하여 2가지의 아이디어가 도출되었고, 각각의 아이디어를 구현/검증해 보기로 결정함



5. GUI 클라이언트: 브랜치 합치기 (Merge)

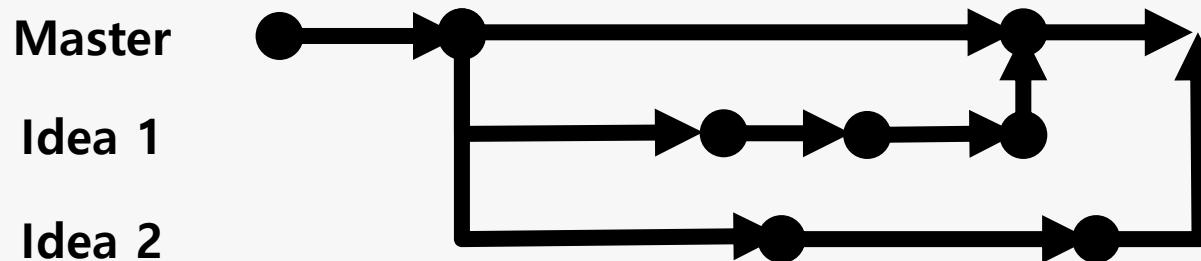
- 새로운 아이디어가 타당하여 본 프로젝트에 정식으로 편입시키고자 할 때
브랜치 합치기 (Merge)를 한다

(예시) Idea 1의 구현 결과가 성능이 좋아 본 프로젝트에 편입시키기로 결정함.
Master 브랜치의 최신 히스토리로 이동(Check-out)한 후, Idea1을 Merge 한다.

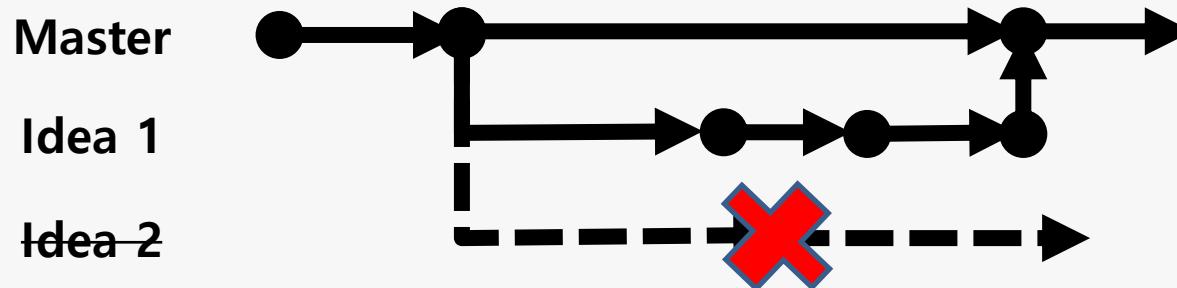


5. GUI 클라이언트: 브랜치 삭제 (Delete)

(예시) Idea 2의 결과 역시 만족스럽거나 별도의 기능이어서, 후에 본 프로젝트에 합치고자 하면 역시 같은 방법으로 Merge 하거나,



삭제할 수 있다. (Idea 2의 내용만 깔끔하게 삭제된다.)



5. GUI 클라이언트: 충돌 해결 (Resolve Conflicts)

(예시) 작업자 A와 작업자 B가 동시에 동일한 부분(파일)을 수정함.

작업자 A

작업자 A의 화면은 Git의 커밋 히스토리와 파일 내용을 보여주는 인터페이스입니다. 커밋 히스토리에는 여러 번의 커밋이 등록되어 있으며, 각 커밋마다 저작자는 Sun Kwon Kim입니다. 파일 내용은 hello_world.cpp이며, 코드에 '+ The Function is implemented.'라는 메시지가 포함되어 있습니다.

작업자 B

작업자 B의 화면은 코드 편집기로, hello_world.cpp 파일의 내용을 보여줍니다. 코드는 다음과 같습니다:

```
// hello_world.cpp : Defines the entry point for the console application.

#include "stdafx.h"

int main()
{
    printf_s("hello world\n");
    printf_s("My name is Sun Kwon Kim.\n");
    printf_s("Function 1.\n");
    printf_s("Function A. worked by Person 2");

    return 0;
}
```

작업자 A가 추가한 코드는 녹색으로 표시되었고, 작업자 B가 추가한 코드는 빨간색으로 표시되었습니다. 특히, 13번 줄과 14번 줄은 둘 다 빨간색으로 강조되어 충돌된 부분임을 시사합니다.

5. GUI 클라이언트: 충돌 해결 (Resolve Conflicts)

KERI

작업자 A

작업자 A의 환경은 다음과 같습니다.

- Git 허브에서 작업자 A의 커밋 목록을 볼 수 있습니다.
- 커밋 88268...에 대한 상세 내용은 '+ the function A is implemented'입니다.
- 커밋 내용에는 'New Function 1', 'new print_f is added', '+Initiate the VS project', 'Initial commit' 등이 포함되어 있습니다.
- 커밋 저자는 Sun Kwon Kim <sunkwonn.kim@gmail.com>입니다.
- 커밋 번호는 d466ca3, e13510c, 81b51a6, 3f140e5, eb131c5입니다.
- 작업자 A는 hello_world.cpp 파일을 수정했습니다.
- 수정 내용은 다음과 같습니다:

```
1 부분 : 8-14 줄
8 8     printf_s("hello world\
9 9     printf_s("My name is S
10 10    +
11 11 +printf_s("Function A.
12 12    return 0;
13 13 }
14 }
```

Commit과 Push 성공

작업자 B

작업자 B의 환경은 다음과 같습니다.

- Pushing 화면에서 출력 전부 보기 옵션을 선택했습니다.
- Push 명령어는 git -c diff.mnemonicprefix=false -c core.quotepath=false -c credential.helper=manager-st push로 실행되었습니다.
- Push 대상은 https://github.com/sunkwon/hello_world.git입니다.
- Push 과정에서 오류가 발생했습니다. [rejected] master -> master (fetch first)입니다.
- 에러 메시지: error: failed to push some refs to 'https://github.com/sunkwon/hello_world.git'
- 힌트: Updates were rejected because the remote contains work that you do not have locally. This is usually caused by another repository pushing to the same ref. You may want to first integrate the remote changes (e.g., 'git pull ...') before pushing again.
- 힌트: See the 'Note about fast-forwards' in 'git push --help' for details.

오류가 나면서 완료됨.

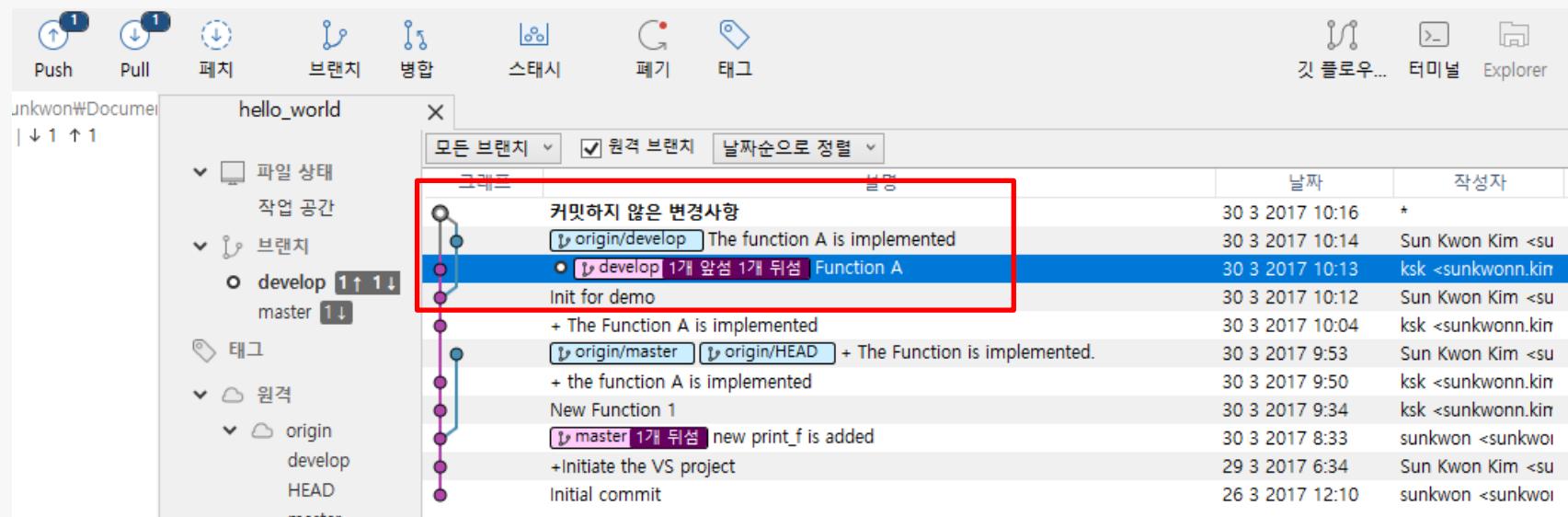
Commit은 성공하지만,
Push에서는 에러 발생

KERI

KOREA ELECTROTECHNOLOGY RESEARCH INSTITUTE

5. 사용 방법 : 충돌 해결 (Resolve Conflicts)

작업자 B



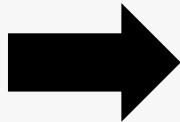
Pull 버튼을 눌러서 원격 저장소의 정보를 가져와 보면, 원격 저장소에 저장된 내용(작업자 A)와 현재 작업한 내용(작업자B)가 충돌이 발생했다는 것을 확인할 수 있음

5. GUI 클라이언트: 충돌 해결 (Resolve Conflicts)

KERI

사용하는 프로그래밍툴을 이용하여 충돌이 일어난 부분을 수정함

```
1 // hello_world.cpp : Defines the entry point for the console application.
2 [
3
4 #include "stdafx.h"
5
6
7 int main()
8 {
9     printf_s("hello world\n");
10    printf_s("My name is Sun Kwon Kim.\n");
11    <<<<< HEAD
12    printf_s("The Function A. implemented by Person B.\n");
13    =====
14    printf_s("The function is implemented. implemented by person A.\n");
15    >>>>> 201248c2d832fa4bc2f0bc724d6ef70f3a404631
16    return 0;
17 }
18
```



```
hello_world (Global Scope) main()
1 // hello_world.cpp : Defines the entry point for the console application.
2 [
3
4 #include "stdafx.h"
5
6
7 int main()
8 {
9     printf_s("hello world\n");
10    printf_s("My name is Sun Kwon Kim.\n");
11    printf_s("The Function A. implemented by Person A and B.\n");
12    return 0;
13 }
14
15
```

5. GUI 클라이언트: 충돌 해결 (Resolve Conflicts)

충돌 부분을 수정한 후 Commit과 Push 버튼을 누름

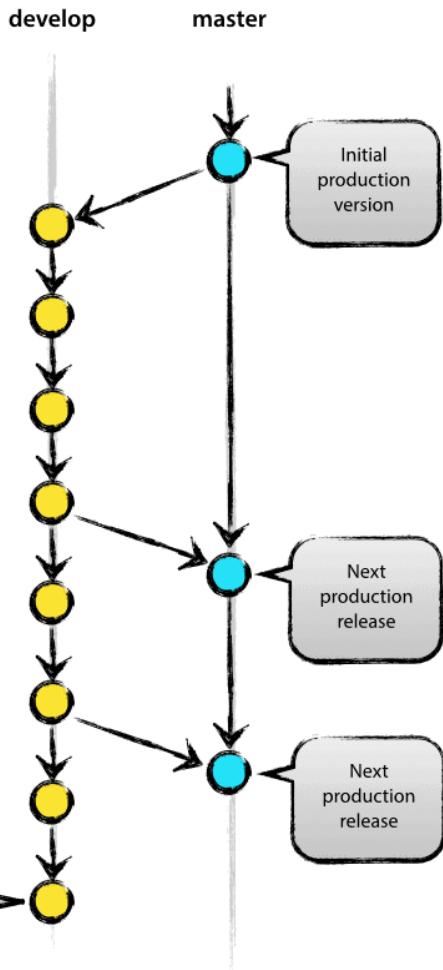
그림표	설명	날짜
	develop origin/develop Merge branch 'develop' of https://github.com/sunkwon/hello_world into develop	30 3 2017 10:24
	The function A is implemented	30 3 2017 10:14
	Function A	30 3 2017 10:13
	Init for demo	30 3 2017 10:12
	+ The Function A is implemented	30 3 2017 10:04
	origin/master origin/HEAD + The Function is implemented.	30 3 2017 9:53
	+ the function A is implemented	30 3 2017 9:50
	New Function 1	30 3 2017 9:34
	master 1개 뒤섬 new print_f is added	30 3 2017 8:33
	+Initiate the VS project	29 3 2017 6:34
	Initial commit	26 3 2017 12:10

충돌이 해결되었음을 확인할 수 있음

Git을 잘 사용하기 위한

6. 브랜치 관리 기법

6. 브랜치 관리 기법 : Simple

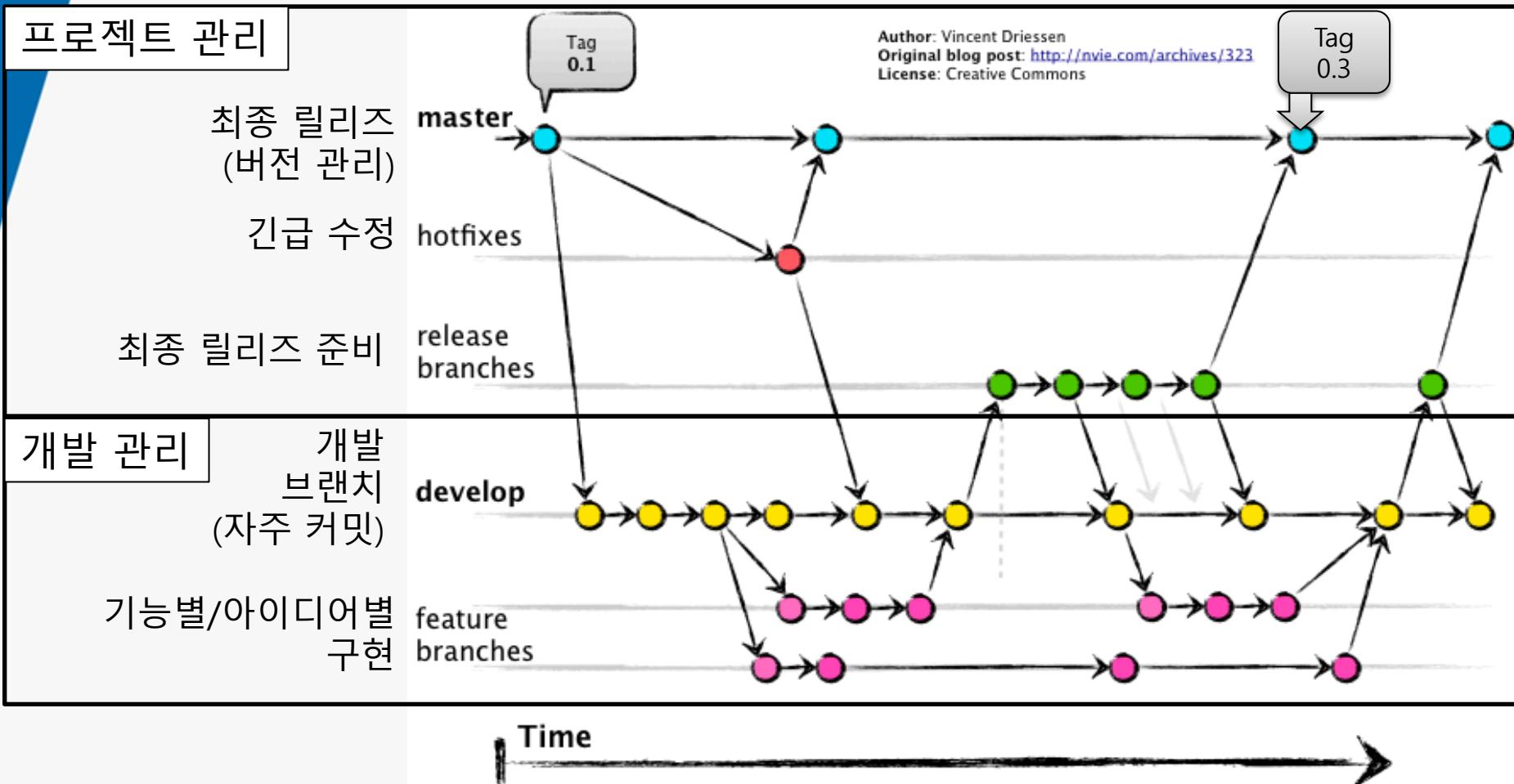


가장 간단한 브랜치 관리법

- Develop : 개발 브랜치
 - 평소에 커밋하는 브랜치
 - 여러 사람들의 작업들이 합쳐지는 브랜치
- Master : 릴리즈 브랜치
 - 릴리즈할 때마다 Develop 브랜치를 Merge
 - Release Version
- Tip:
 - Develop 브랜치는 자주 커밋을 해서 중간중간에 롤백이 가능한 포인트를 만드는 것이 좋음
 - Master 브랜치는 결과물이 출시되거나 팀 외부에 나갈 때마다 포인트를 만드는 것이 좋음

6. 브랜치 관리 기법 : Git-flow

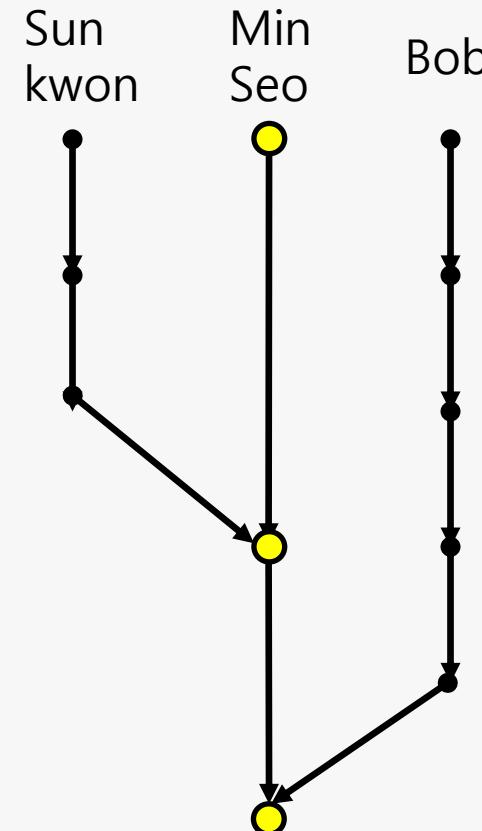
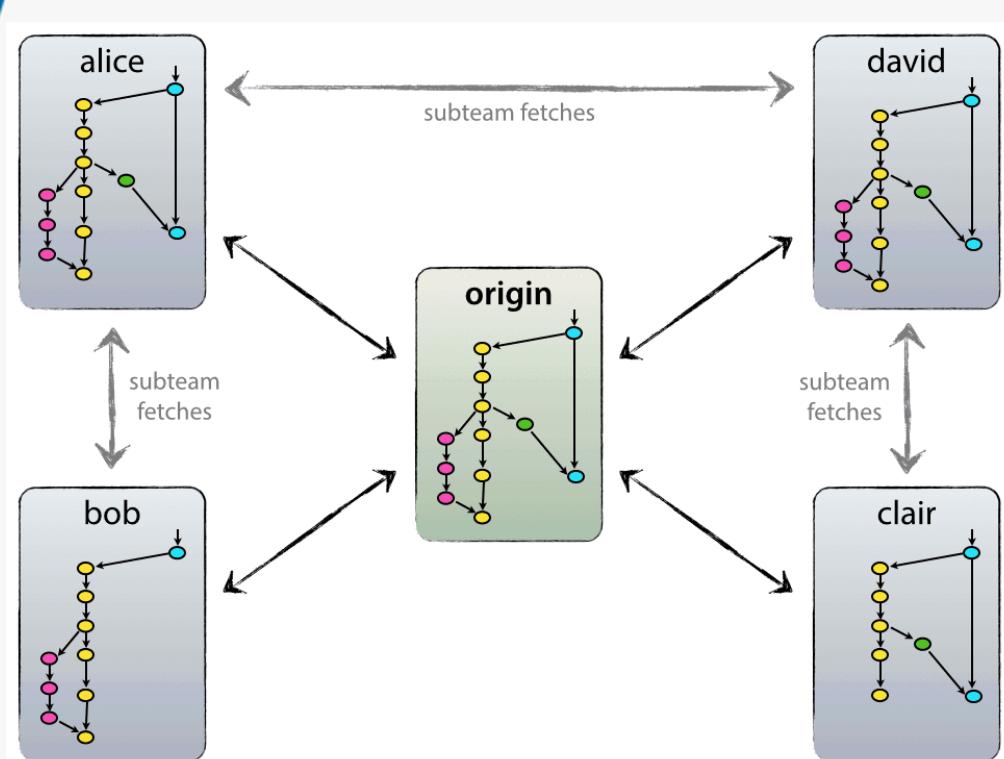
효과적인 브랜치 관리 기법으로 받아들여진 형태 : Git-flow



6. 브랜치 관리 기법 : Git-flow

Decentralized but centralized

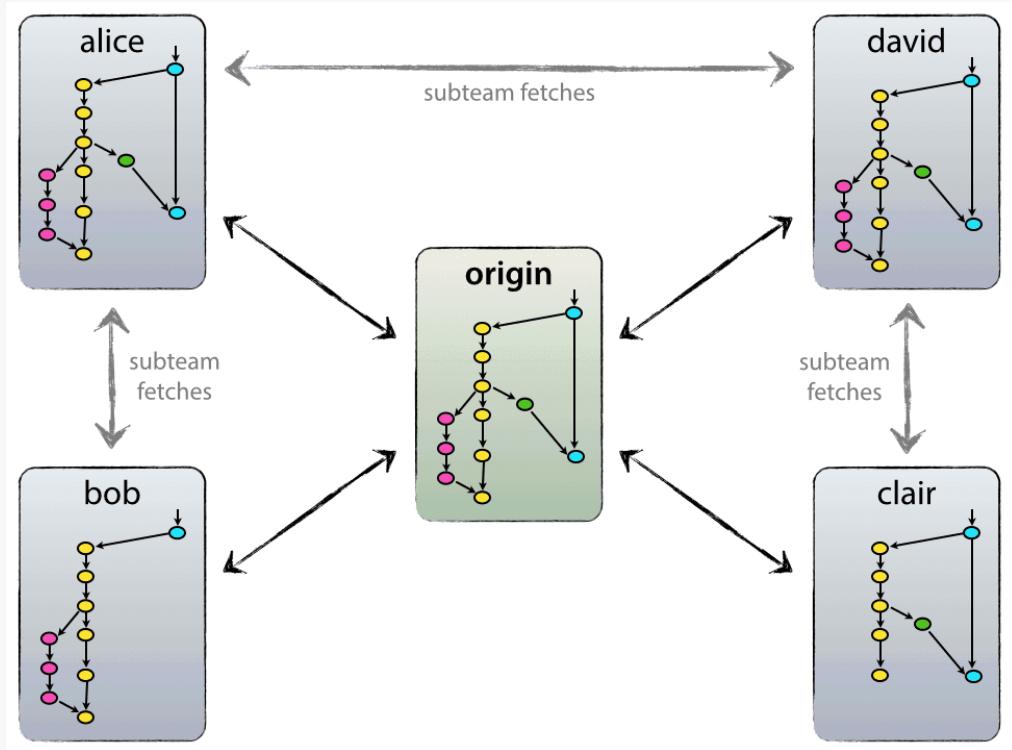
(예시) Git-flow의 정석은 아니지만...



출처: A successful Git branching model (<http://nvie.com/posts/a-successful-git-branching-model/>)

6. 브랜치 관리 기법 : Git-flow

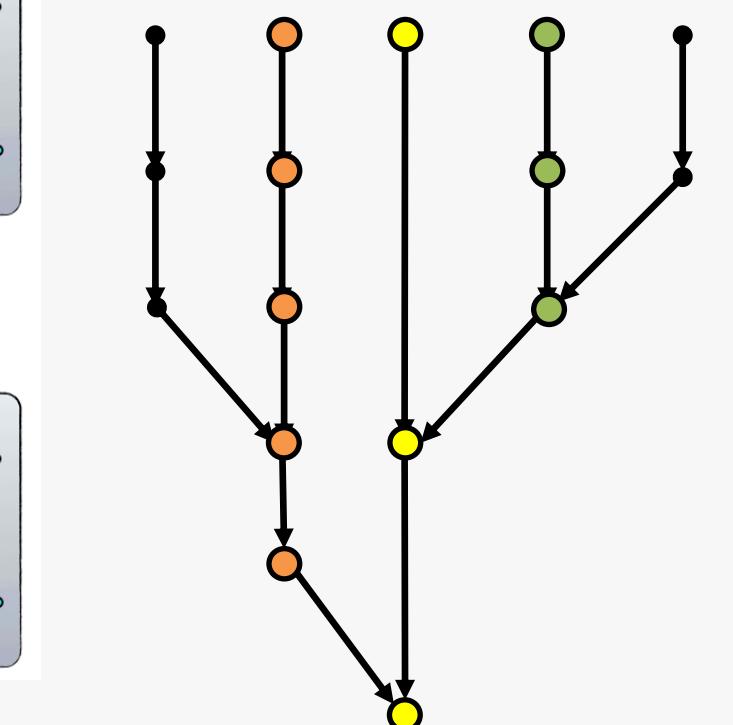
Decentralized but centralized



Team A
Sun
Min
kwon
Seo

PL
Bob

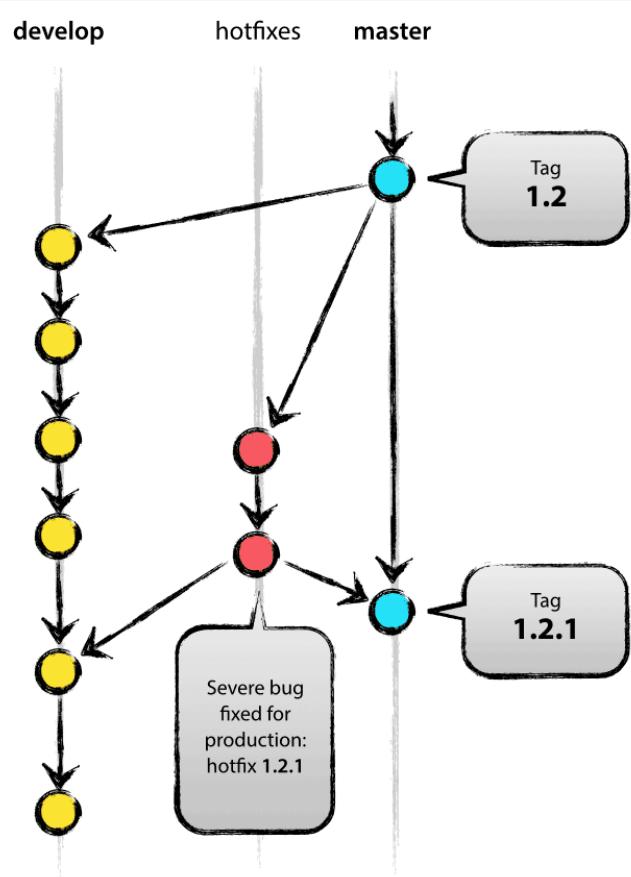
Team B
John
Alice



출처: A successful Git branching model (<http://nvie.com/posts/a-successful-git-branching-model/>)

6. 브랜치 관리 기법 : Git-flow

Hotfixes branches



버그를 수정한 내용들은 Develop 브랜치로 합치는 것이 일반적임

하지만 매우 긴급한 버그 수정 사항들(Hotfixes)은 Develop과 master 2개의 브랜치로 Merge 시키고 바로 공개(릴리즈)함

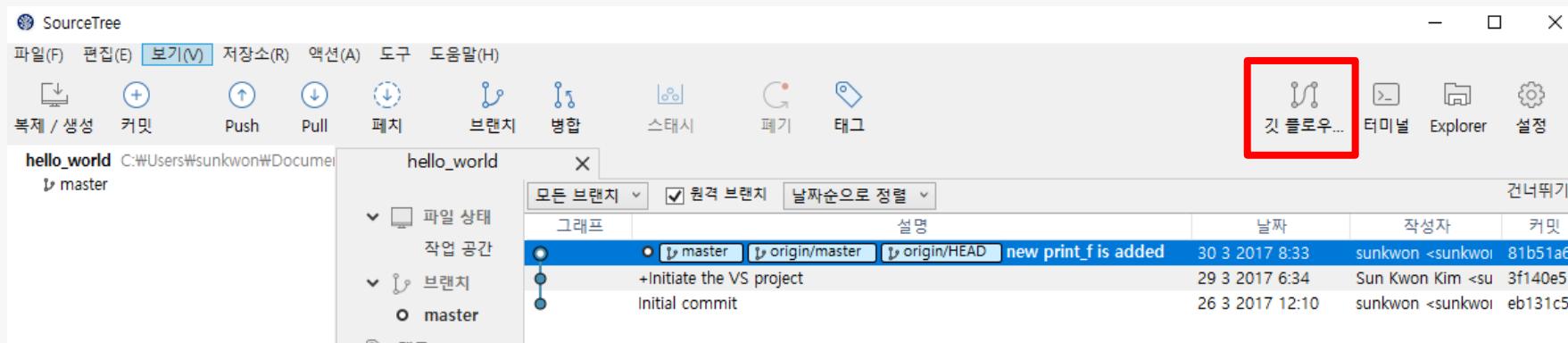
Tip. 릴리즈 버전을 매기는 규칙은 강제성이 없음. 편한대로 하면 좋지만, 일반적으로 (MajorRelease, MinorRelease, HotfixRelease)의 형태를 많이 사용함

출처: A successful Git branching model (<http://nvie.com/posts/a-successful-git-branching-model/>)

6. 브랜치 관리 기법 : Git-flow

대부분의 (모든) Git Client에서 Git-flow 관리를 도와주는 메뉴가 존재함.

1. 깃 플로우 아이콘을 선택한다.
(또는 메뉴 → 저장소 → 깃 플로우 → 저장소 초기화를 선택한다.)



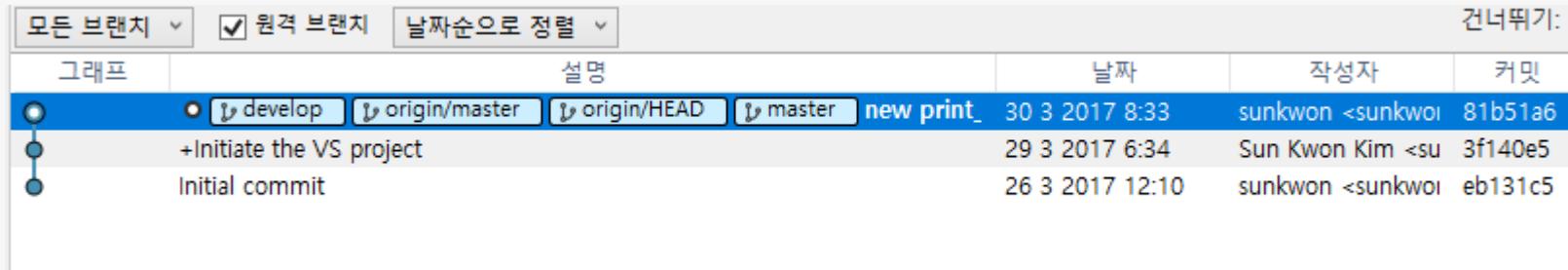
2. 확인을 버튼을 누른다.



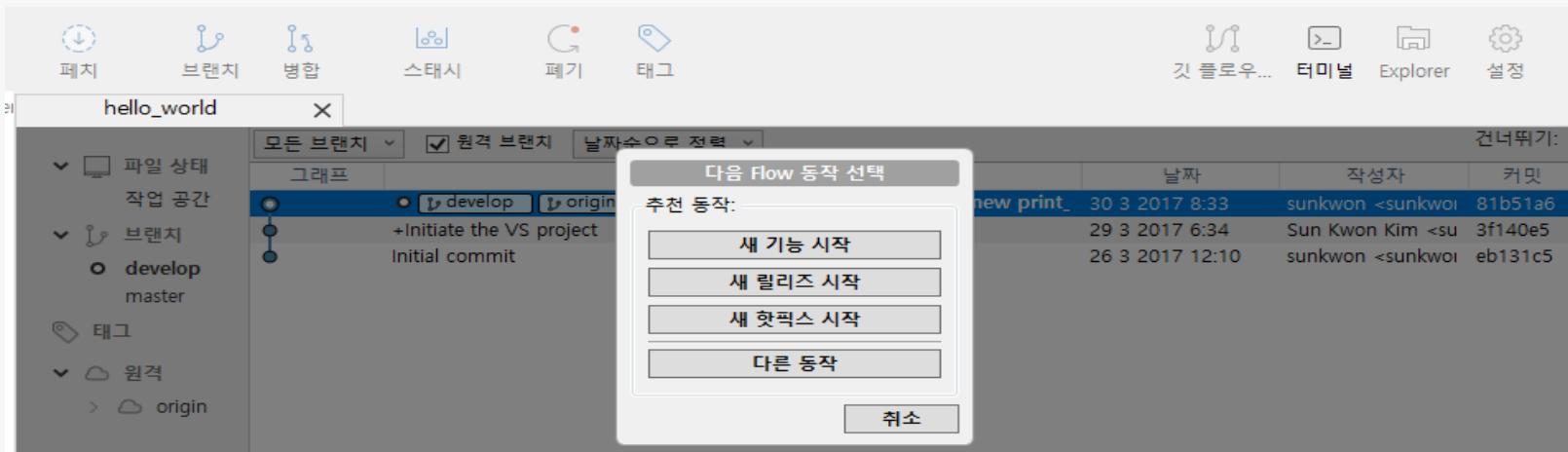
6. 브랜치 관리 기법 : Git-flow

대부분의 (모든) Git Client에서 Git-flow 관리를 도와주는 메뉴가 존재함.

3. 자동으로 Develop가 생성되었음

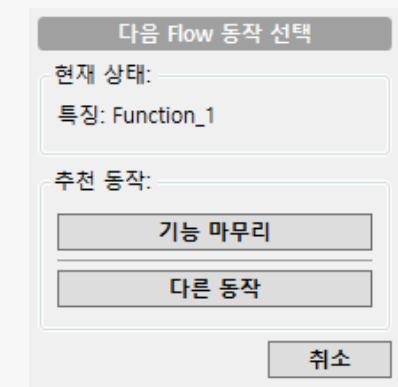
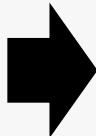
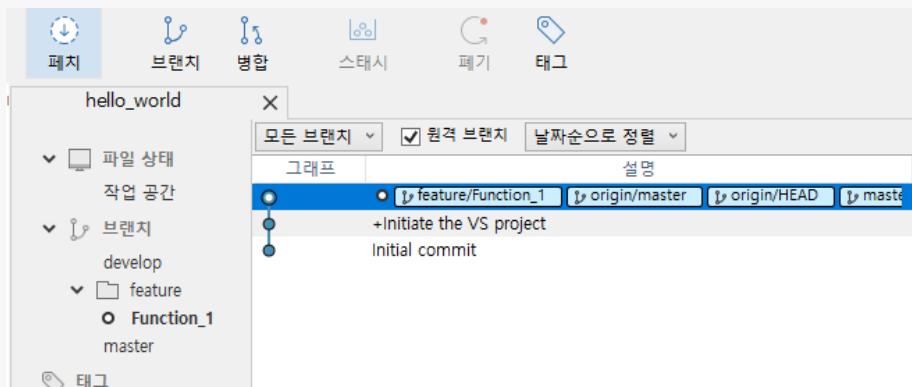
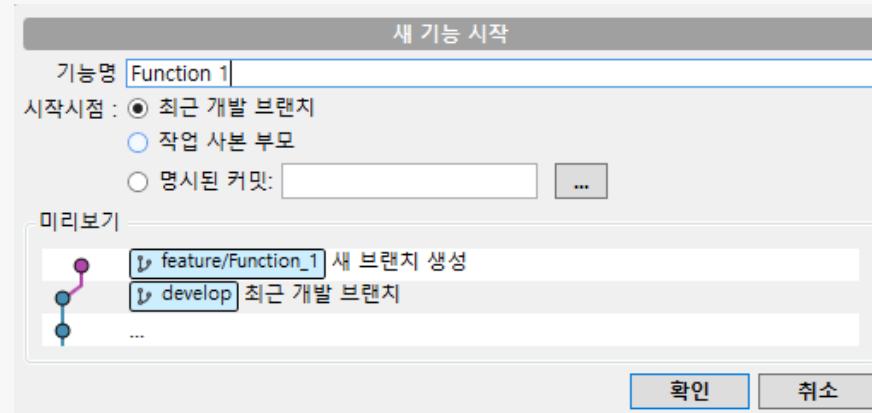
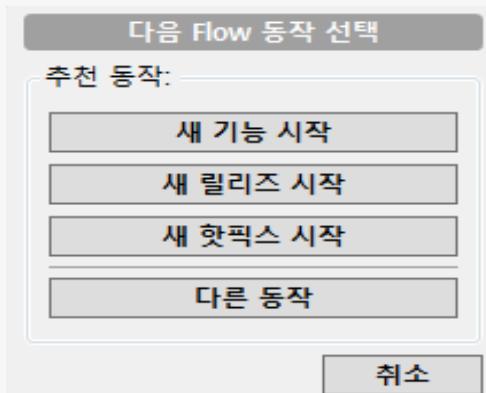


4. 다시 깃 플로우 버튼을 누르면, Git-flow 정책에 따른 메뉴가 나타남



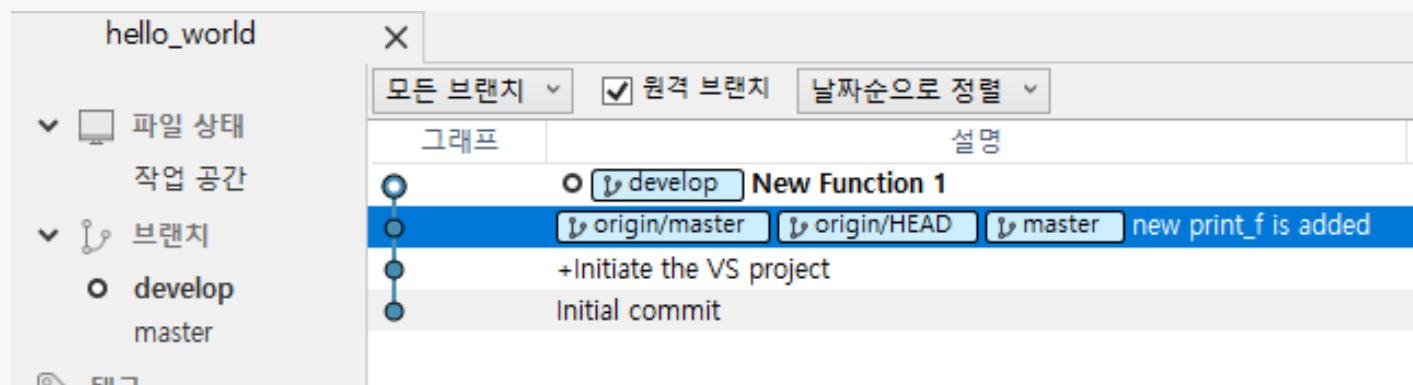
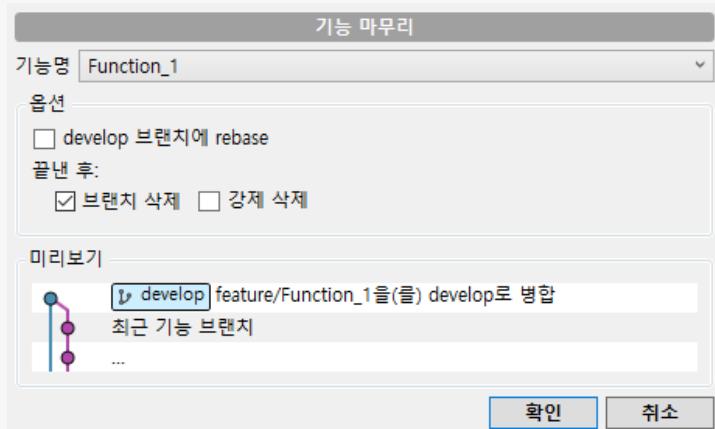
6. 브랜치 관리 기법 : Git-flow

Feature branches



6. 브랜치 관리 기법 : Git-flow

Feature branches



7. GITHUB

7. GitHub : 협업하기

1. 생성한 프로젝트 → Settings → Collaborators → Add collaborator

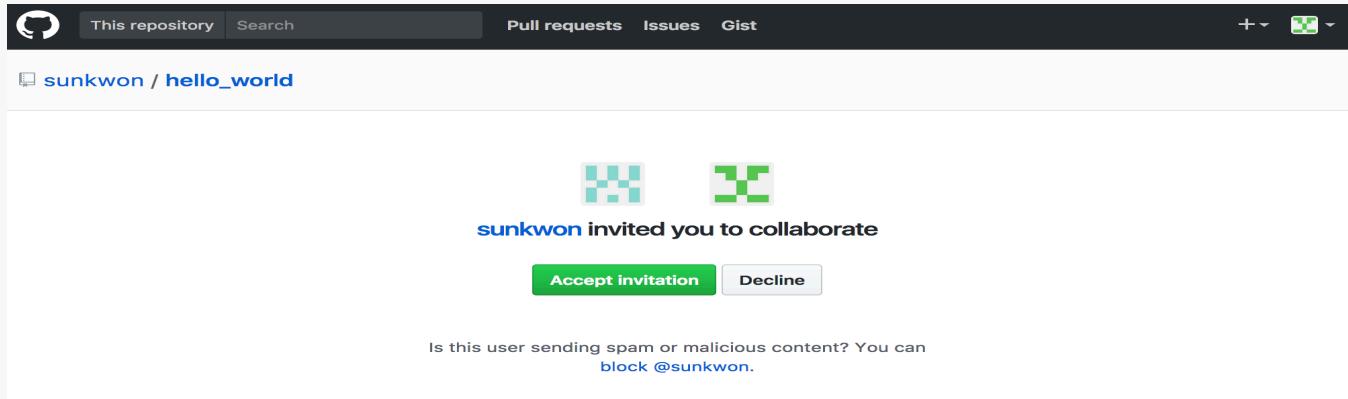
This screenshot shows the GitHub repository settings page for 'sunkwon / hello_world'. The 'Collaborators' tab is selected in the sidebar. A red arrow points to the 'Settings' button in the top right corner. Below the sidebar, there is a search bar and an 'Add collaborator' button, which is also highlighted with a red box.

2. Invite Link 주소를 초청하고자 하는 사람에게 보내기

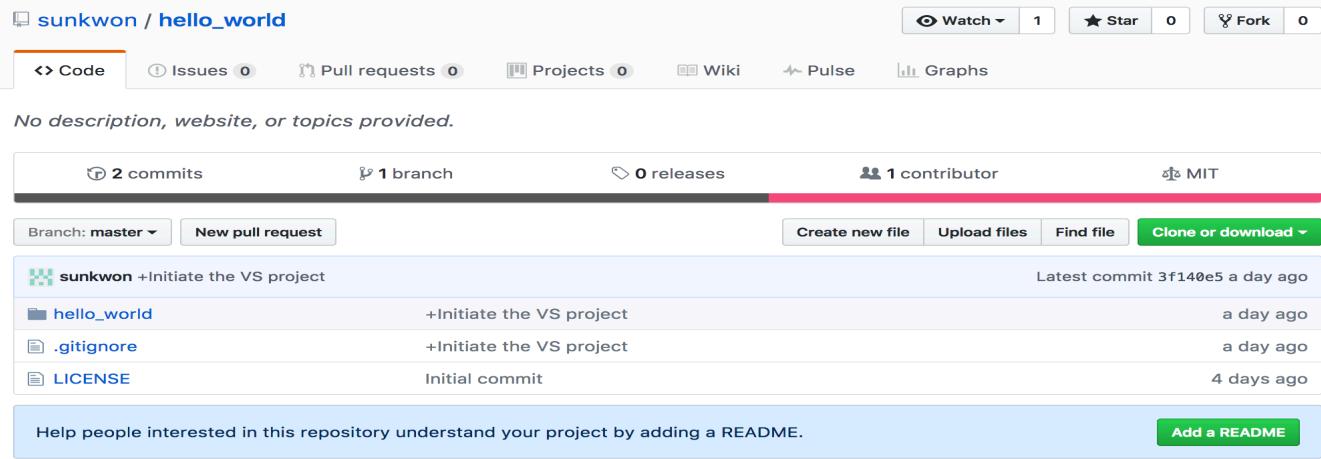
This screenshot shows the GitHub repository settings page for 'sunkwon / hello_world'. The 'Collaborators' tab is selected in the sidebar. A green icon indicates an awaiting response from 'sunkwonkim'. A 'Copy invite link' button is visible, and a tooltip shows the copied URL: https://github.com/sunkwon/hello_world/collaborators/add. The 'Add collaborator' button is also visible.

7. GitHub : 협업하기

3. 초청받은 사람이 Link 주소에 접속해서 "Accept Invitation" 버튼을 누른다.



4. 프로젝트에 합류 완료



7. GitHub : 협업하기

8. 초청받은 사람이 Git 저장소로부터 소스코드를 다운받기 (다른 환경을 가정: 리눅스 Ubuntu, Command 작업환경)

```
sunkwon@HP-Z840-Workstation: ~/Documents
sunkwon@HP-Z840-Workstation:~/Documents$ git clone https://github.com/sunkwon/hello_world.git
```

9. 소스코드 다운로드 완료

```
sunkwon@HP-Z840-Workstation: ~/Documents
sunkwon@HP-Z840-Workstation:~/Documents$ git clone https://github.com/sunkwon/hello_world.git
Cloning into 'hello_world'...
remote: Counting objects: 16, done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 16 (delta 0), reused 12 (delta 0), pack-reused 0
Unpacking objects: 100% (16/16), done.
Checking connectivity... done.
sunkwon@HP-Z840-Workstation:~/Documents$
```

*위 작업은 source tree에서도 앞서 설명한 방식으로 할 수 있습니다. 이 예시는 다른 OS 환경에서 다른 툴로 작업하는 것을 예시로 보여드리는 것입니다.

7. GitHub : 협업하기

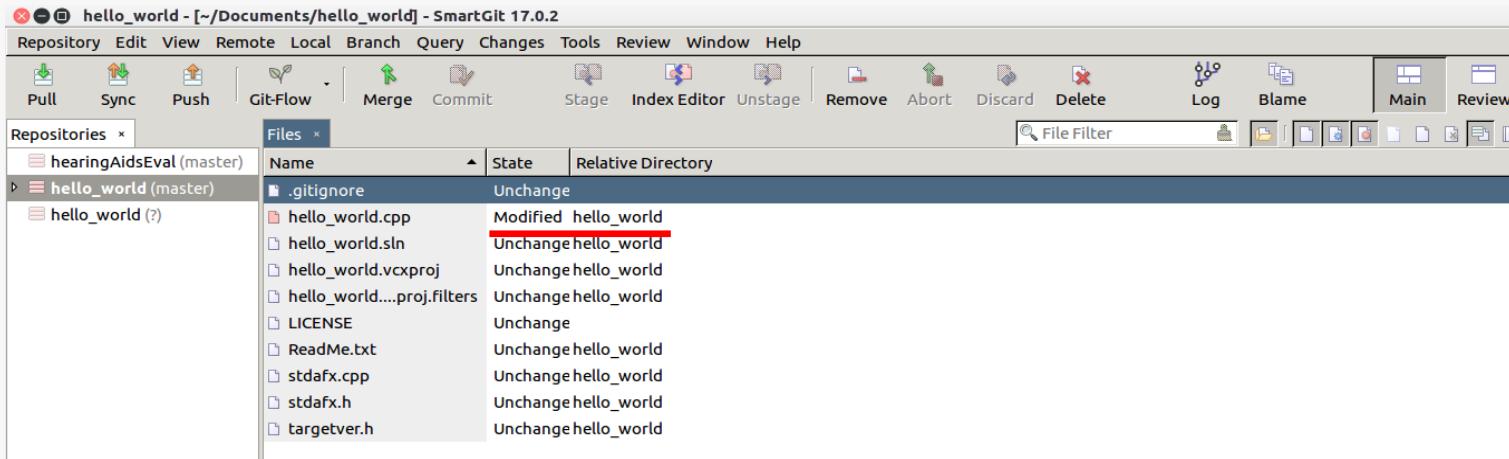
10. 메모장에서 소스 코드 수정



```
// hello_world.cpp : Defines the entry point for the console application.
// ...
#include "stdafx.h"

int main()
{
    printf_s("hello world\n");
    printf_s("My name is Sun_Kwon_Kim.\n");
    return 0;
}
```

11. Git Client에서 소스 변경을 확인



hello_world - [~/Documents/hello_world] - SmartGit 17.0.2

Repository Edit View Remote Local Branch Query Changes Tools Review Window Help

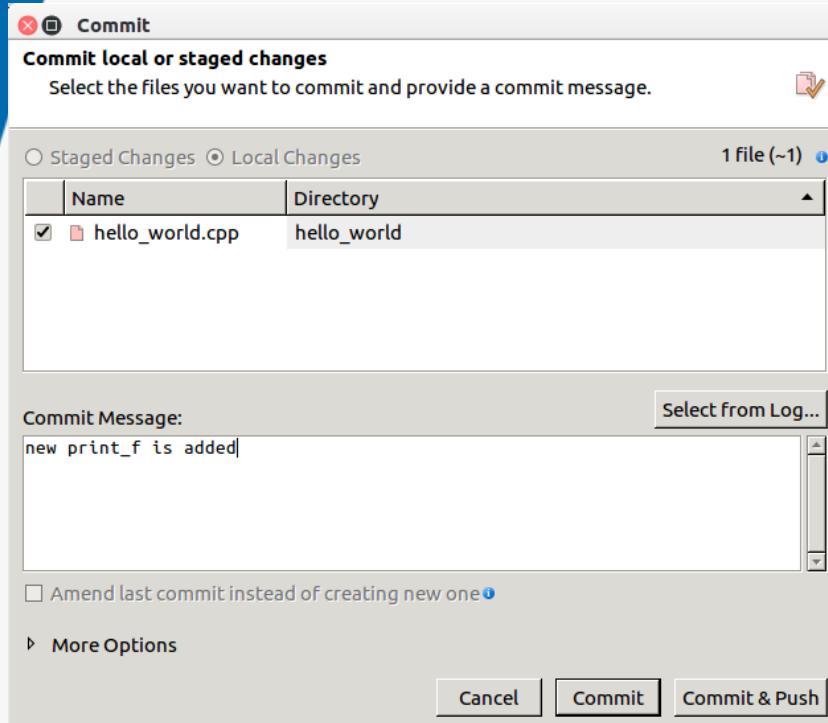
Pull Sync Push | Git-Flow Merge Commit Stage Index Editor Unstage Remove Abort Discard Delete Log Blame Main Review

Repositories x Files x

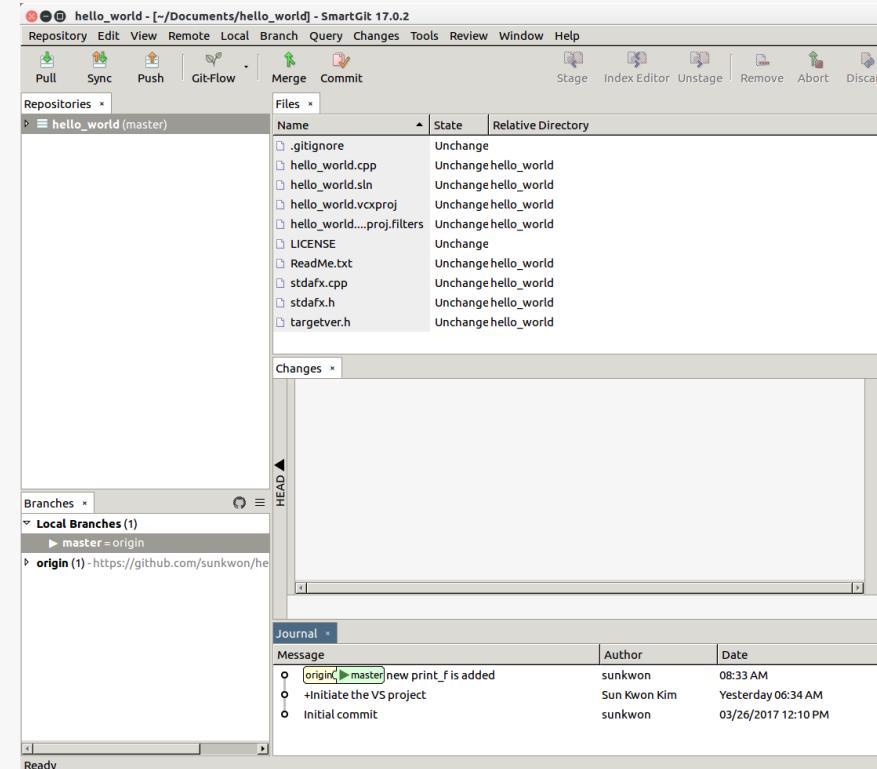
Name	State	Relative Directory
.gitignore	Unchange	
hello_world.cpp	Modified	hello_world
hello_world.sln	Unchange	hello_world
hello_world.vcxproj	Unchange	hello_world
hello_world....proj.filters	Unchange	hello_world
LICENSE	Unchange	
ReadMe.txt	Unchange	hello_world
stdafx.cpp	Unchange	hello_world
stdafx.h	Unchange	hello_world
targetver.h	Unchange	hello_world

7. GitHub : 협업하기

11. Commit과 Push 하기

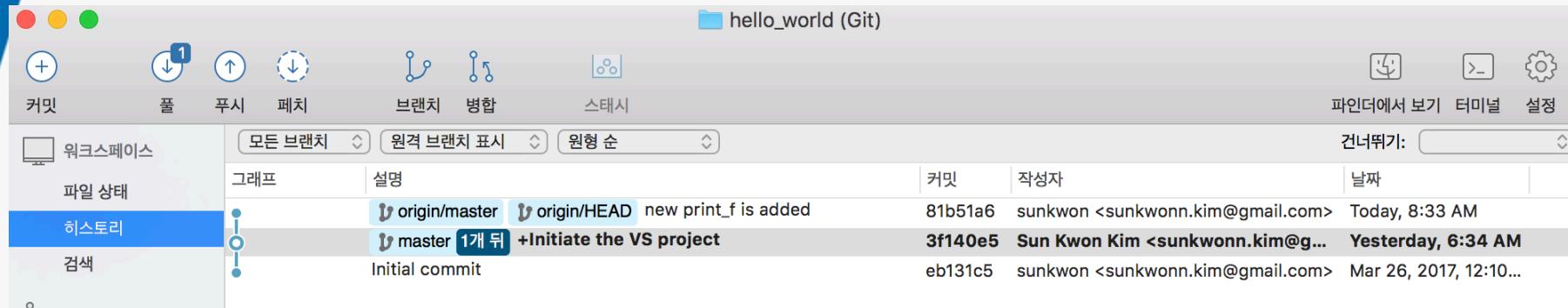


12. 새로운 history (log)가 생김

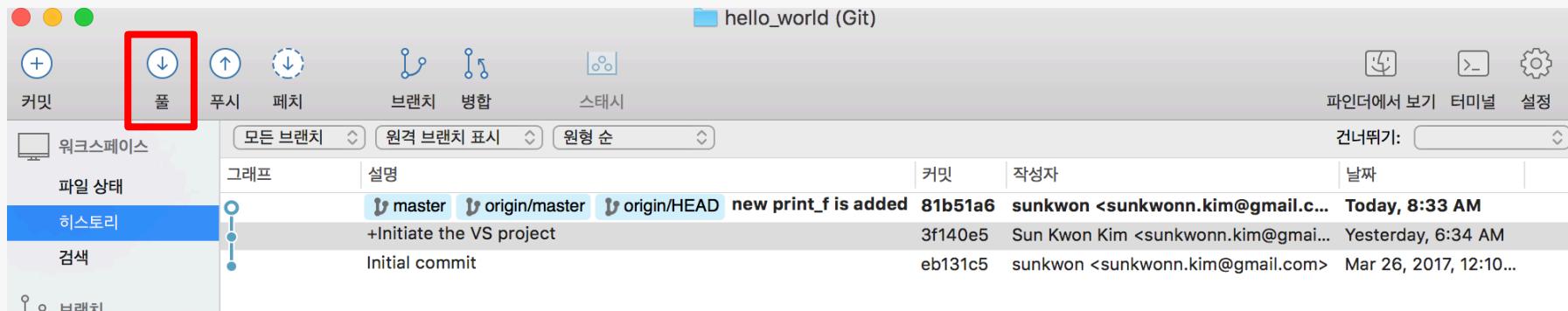


7. GitHub : 협업하기

13. 프로젝트 생성자의 Source Tree 화면에 원격 저장소("origin", GitHub)에 새로운 작업 히스토리가 생겼음을 볼 수 있음.



14. Pull 버튼을 눌러서 원격저장소의 새로운 내용을 다운받음

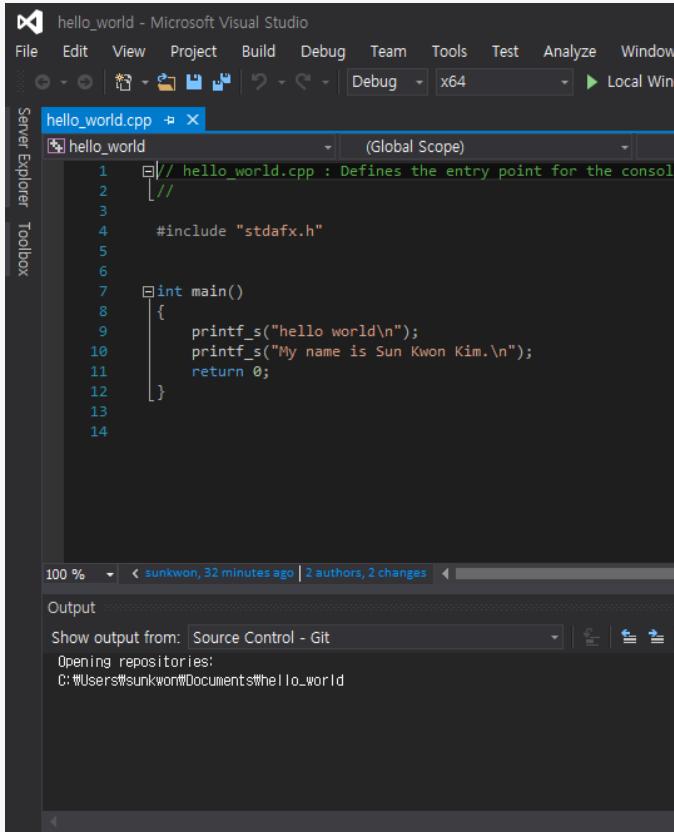


7. GitHub : 협업하기

KERI

15. 프로젝트 생성자의 작업 환경에 소스코드 변경 사항이 적용 되었음.

작업환경 (예: Visual Studio)

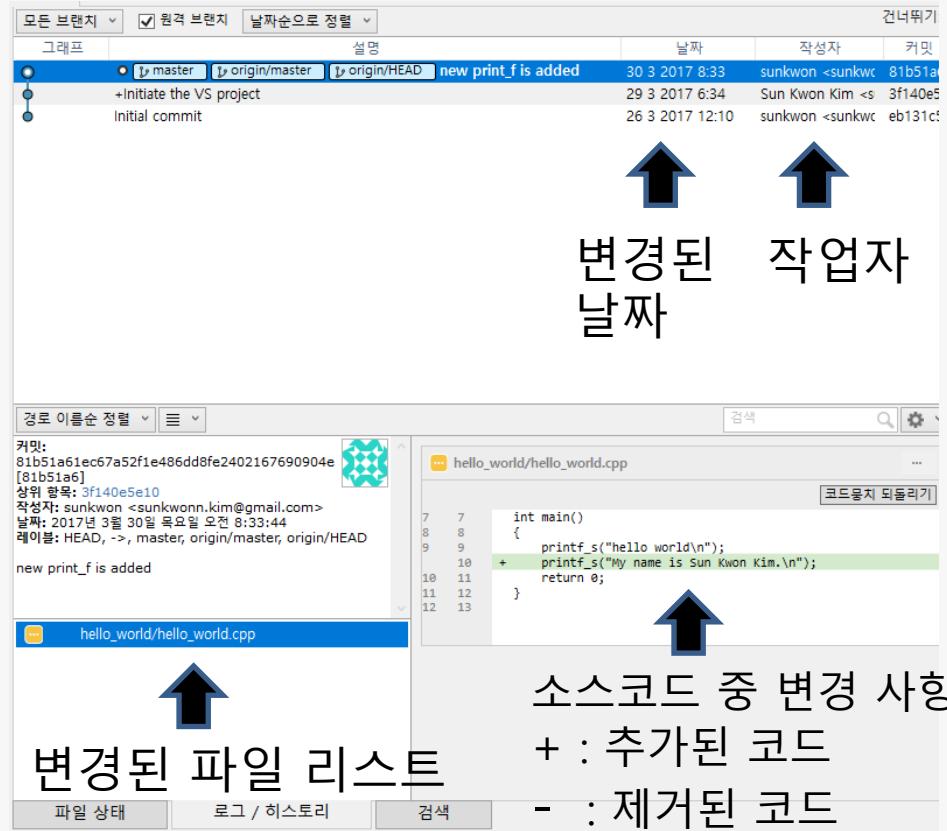


```
// hello_world.cpp : Defines the entry point for the console application.
#include "stdafx.h"

int main()
{
    printf_s("hello world\n");
    printf_s("My name is Sun Kwon Kim.\n");
    return 0;
}
```

Output
Show output from: Source Control - Git
Opening repositories:
C:\Users\sunkwon\Documents\hello_world

Git Client (예: Source Tree)



모든 브랜치 ✓ 원격 브랜치 날짜순으로 정렬 설명 날짜 작성자 커밋

	master	origin/master	origin/HEAD	new print_f is added	30 3 2017 8:33	sunkwon <sunkwon.kim@gmail.com>	81b51a6
+	Initiate the VS project				29 3 2017 6:34	Sun Kwon Kim <sunkwon.kim@gmail.com>	3f140e5
+	Initial commit				26 3 2017 12:10	sunkwon <sunkwon.kim@gmail.com>	eb131c5

변경된 작업자 날짜

경로 이름순 정렬 검색

커밋:
81b51a61ec67a52f1e486dd8fe2402167690904e [81b51a6]
상위 항목: 3f140e5e10
작성자: sunkwon <sunkwon.kim@gmail.com>
날짜: 2017년 3월 30일 목요일 오전 8:33:44
레이블: HEAD, -, master, origin/master, origin/HEAD

new print_f is added

hello_world/hello_world.cpp

7 7 int main()
8 8 {
9 9 printf_s("hello world\n");
10 + printf_s("My name is Sun Kwon Kim.\n");
11 11 return 0;
12 12 }

변경된 파일 리스트

소스코드 중 변경 사항
+ : 추가된 코드
- : 제거된 코드

지금까지 자세히 설명하지 않은

8. 기타

- 스탠시 (Stash)
 - 임시로 저장하기
- 리베이스 (Rebase)
 - 결과는 병합 (Merge)과 비슷함.
 - Merge는 합치는 2개의 작업 플로우가 그대로 남지만, Rebase는 하나의 플로우로 시간순으로 재배치하여 정리함.

매일 자동으로 프로젝트 실행 여부를 검사하기

Linux 예약 실행 스케줄러: crontab

1. 스케줄 목록 확인 : crontab -l
2. 스케줄 등록/수정/확인: crontab -e

```
.----- 분 - 범위 : 0-59  
| .----- 시 - 범위 : 0-23 (자정이 0)  
| | .----- 일 - 범위 : 1-31  
| | | .---- 달 - 범위 : 1-12 또는 jan,feb, mar, apr ... 축약형 영문 월 표시  
| | | | .--- 요일 - 범위 : 0-6 (일요일은 0 또는 7) 또는 sun, mon, tue, wed thu, fri, sat  
| | | ||  
# * * * * [실행될 명령어]
```

3. 스케줄 서비스 시작(재실행): service cron restart

8. 기타

/etc/test_project.sh 파일 생성

```
cd ~/Git 작업 디렉토리  
Git pull  
make  
.run_test
```

Crontab 서비스에 매일 오전 1시에 test_project.sh 파일을 실행하도록 등록

```
crontab -e  
* 1 * * * sh ~/test_project.sh
```



감사합니다.

가치있는 대형 연구성과 창출의 산실

KERI

KOREA ELECTROTECHNOLOGY RESEARCH INSTITUTE

