
전자정부 SW 개발자를 위한
표준프레임워크
Git 적용 가이드

2021. 3

NIA 한국지능정보사회진흥원

표준프레임워크센터

▪ <제목 차례>	
▪ 제 1 장 개요	1
▪ 제1절 Git이란	1
▪ 제2절 Git의 역사	1
▪ 제3절 Git의 구조	2
▪ 제4절 Git SCM 설치	2
▪ 1. Windwos에 설치	2
▪ 2. Mac에 설치	3
▪ 3. Linux에 설치	3
▪ 제5절 Git 원격 저장소	3
▪ 1. GitHub	4
▪ 2. GitLab	4
▪ 3. Bitbucket	4
▪ 제6절 Git 기초	4
▪ 1. 사용자 정보 설정	4
▪ 2. 도움말 보기	5
▪ 3. Git 저장소	5
▪ 4. 파일 버전 관리	5
▪ 제 2 장 Git Clients 도구	7
▪ 제1절 Git Clients 도구	7
▪ 제2절 Git Bash 사용	7
▪ 제3절 표준프레임워크 개발환경에서 사용	7
▪ 제4절 3rd Party Git GUI Clients 사용	8
▪ 1. Gitk와 Git GUI	8
▪ 2. GitHub Desktop	9
▪ 3. SourceTree	10
▪ 3. TortoiseGit	11
▪ 제 3 장 Git 주요 기능	12
▪ 제1절 브랜치	12
▪ 1. 브랜치 관리	12
▪ 2. 브랜치 전환	12
▪ 3. 브랜치 병합	13
▪ 제2절 원격 저장소	14
▪ 1. Remote	14
▪ 2. Pull, Fetch	15
▪ 3. Push	15
▪ 4. Fork	15
▪ 5. Pull Request (Merge Request)	18
▪ 제3절 태그	21
▪ 1. 태그 조회	21
▪ 2. 태그 붙이기	21
▪ 3. 태그 공유	21
▪ 제 4 장 표준프레임워크 오픈소스 기여 하기	23
▪ 제1절 Git 사용 환경 설정	23

▪ 1. Git 설치	23
▪ 2. GitHub 계정 등록	23
▪ 3. 개발환경 eGovFrame Perspective	25
▪ 4. 개발환경에 Git 환경 설정	25
▪ 제2절 표준프레임워크 소스 복제	26
▪ 1. GitHub Fork	26
▪ 2. GitHub Clone	28
▪ 제3절 개발작업 후 내 원격저장소로 Push	30
▪ 1. 수정된 소스를 내 로컬저장소에 commit	30
▪ 2. 수정된 소스를 내 원격저장소로 Push	32
▪ 3. 원본 원격저장소에 반영 요청 (Pull Request)	34
▪ 제4절 Pull request 이후 검토내용 확인	35
▪ 1. 검토내용 확인	35
▪ 2. 원본 원격저장소 브랜치에 병합 확인	36
▪ 제5절 표준프레임워크 원격저장소와 최신버전 동기화	36
▪ 1. 개발환경에 표준프레임워크 원본 원격저장소 추가	36
▪ 2. 표준프레임워크 원격저장소로부터 최신 commit Pull	38
▪ 3. Conflict 해결	39
▪ 4. 내 원격저장소로 Push	40
▪	

제 1 장 개요

제1절 Git이란

Git은 파일의 변경사항을 추적하고 다수 개발자들 간에 해당 파일들의 작업을 조율하기 위한 분산 버전 관리 시스템이다. 원격저장소(Git 서버라고 하기도 함)를 전부 로컬 장비로 복제하기 때문에, 서버에 문제가 생겨도 어떠한 클라이언트에서라도 그 로컬에 복제된 것으로 다시 복원할 수 있다. Git은 프로젝트의 히스토리를 조회할 때 서버 없이 조회한다. 파일을 비교하기 위해 리모트에 있는 서버에 접근하지 않고, 로컬 데이터베이스에서 히스토리를 읽어서 보여주기 때문에 처리속도가 빠르며, 오프라인 상태일 때도 커밋을 할 수 있기에 장소에 구애받지 않고 협업이 가능하다.

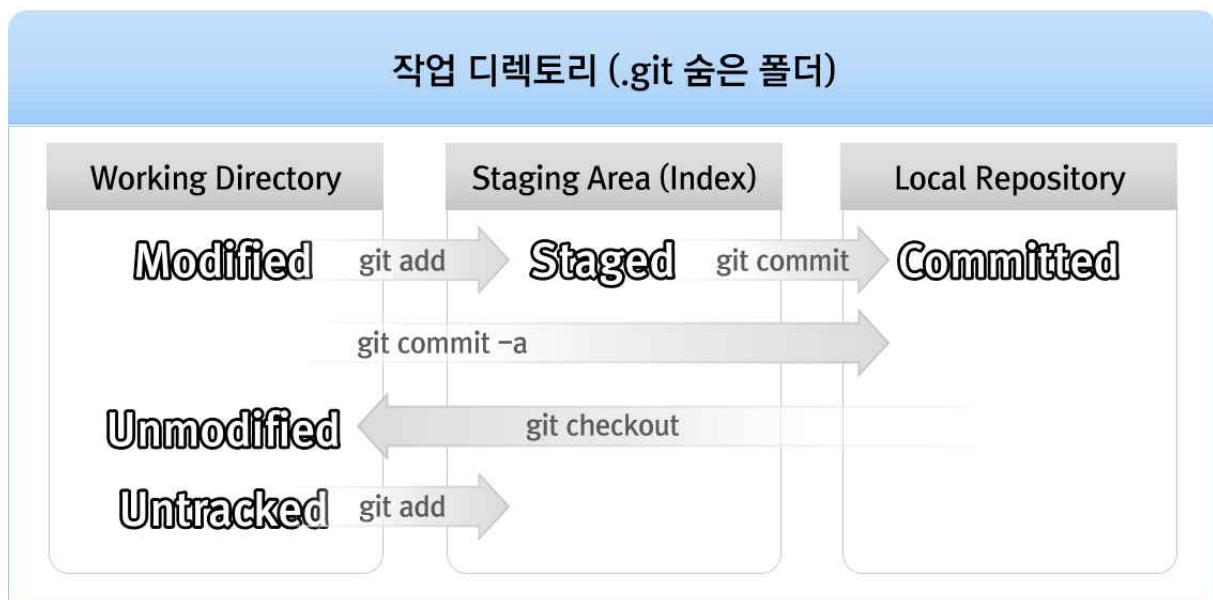
제2절 Git의 역사

Linux라는 운영체제를 만든 Linus Torvalds (핀란드)가 Linux 버전관리를 위해 BitKeeper라는 분산 버전 관리 툴을 사던 중 Linux 커뮤니티와 BitKeeper 측 사이의 불화가 생겨, BitKeeper를 대신할 다른 버전 관리 툴을 찾았으나, 찾지 못하자, 본인이 직접 버전 관리 프로그램을 만들었는데, 그것이 바로 2005년에 탄생한 Git이다. Git은 BitKeeper를 사용하면서 배운 교훈을 기초로 아래와 같은 목표를 갖고 설계 및 제작되었다.

- 빠른 속도
- 단순한 디자인
- 비선형적 개발 (수천 개의 동시 다발적인 브랜치)
- 완벽한 분산
- Linux 커널 같은 대형 프로젝트도 속도 저하 문제없이 관리

제3절 Git의 구조

Git은 파일을 Modified, Staged, Committed 이렇게 세가지 상태로 관리한다. Modified는 수정만 한 상태이고, Staged란 현재 수정한 파일을 곧 커밋할 것이라고 표시한 상태이고, Committed란 데이터가 로컬 데이터베이스에 안전하게 저장됐다는 것을 의미한다.



Git의 작업 디렉토리 즉, .git 폴더는 프로젝트의 메타데이터와 객체 데이터베이스를 저장하는 곳으로 Git의 핵심이다. Git 원격저장소로부터 기존 프로젝트를 복제할 때 'git clone' 명령을 실행하면 Git 디렉토리가 만들어진다. 또한 아직 버전관리를 하지 않는 로컬 디렉토리도 'git init' 명령을 실행하면 '.git'이라는 하위 디렉토리가 만들어 진다.

제4절 Git SCM 설치¹⁾

Git을 사용하려면 우선 Git SCM (Source Code Management)을 설치해야 한다.

1. Windwos에 설치

1) <https://git-scm.com/downloads> 참조

공식 배포판은 '<http://git-scm.com/download/win>'에서 다운로드하여 설치하면 된다.

2. Mac에 설치

Mac에 Xcode Command Line Tools이 설치되어 있다면 터미널에서 'git'을 실행하는 것만으로 간단히 설치할 수 있다. 좀 더 최신 버전의 인스톨러로 설치를 하려 한다면, Git 웹사이트에서 제공하는 공식 배포판 '<http://git-scm.com/download/mac>'에서 다운로드하여 설치하면 된다.

3. Linux에 설치

Linux에서 패키지로 Git을 설치할 때는 보통 각 배포판에서 사용하는 패키지 관리도구를 사용하여 설치한다.

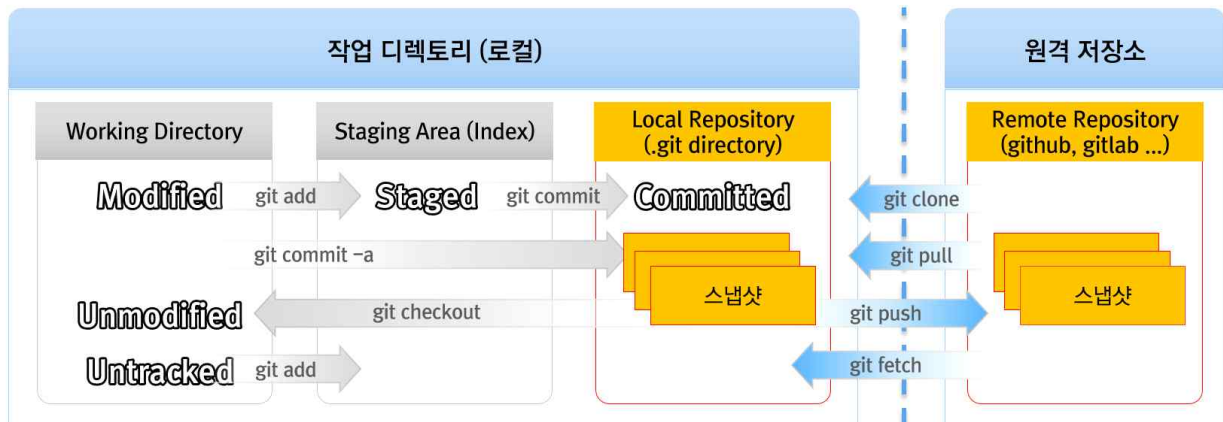
- Fedora에서 설치 '\$ sudo yum install git-all'
- Ubuntu에서 설치 '\$ sudo apt-get instal git-all'
- 그 외 다른 Unix 배포판에 설치하려면

<http://git-scm.com/download/linux> 에서 확인한다.

제5절 Git 원격 저장소

Git은 로컬 저장소에서만 운영해도 되지만, 로컬 저장소의 작업 결과물을 백업을 하고, 다른 사람들과 협업 및 공유하기 위해 인터넷이나 네트워크상에 있는 저장소인 원격 저장소를 사용한다. 원격 저장소는 일반적으로 작업 디렉토리가 없는 Bare 저장소이다. 이 저장소는 협업용이기 때문에 체크아웃이 필요 없다. 단지 Git 데이터를 포함하는 .git 디렉토리만 있는 저장소이다.

Git은 단순히 디스크의 다른 디렉토리에 있을 때나 NFS같은 파일시스템을 공유하고 있을 때 사용하는 Local을 비롯하여 HTTP, SSH, Git 이렇게 네가지의 프로토콜을 사용할 수 있다.



1. GitHub²⁾

GitHub은 Git을 지원하는 웹호스팅 서비스로 GitHub.com 계정으로 접근하여 웹에서 프로젝트 버전관리를 할 수 있다. GitHub 원격 저장소를 고객과 공유할 때 주로 사용된다. 공개를 원치 않는 개인 저장소로도 사용할 수 있는데 이는 유료이다.

2. GitLab³⁾

설치형 GitHub이라는 컨셉으로 시작된 MIT 라이선스 오픈소스로 소스코드의 보안이 중요한 프로젝트에 적합하다.

3. Bitbucket⁴⁾

Bitbucket은 Atlassian의 웹기반 호스팅 서비스로 5명 이하 구성원의 소규모팀은 무료로 사용할 수 있다.

제6절 Git 기초

1. 사용자 정보 설정

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
$ git config --list
```

2) <https://github.com/> 참조

3) <https://docs.gitlab.com/> 참조

4) <https://bitbucket.org/> 참조

Git은 커밋할 때마다 사용자정보(사용자명, 이메일)를 저장한다. 한 번 커밋한 후에는 정보를 변경할 수 없다.

2. 도움말 보기

```
$ git help <verb>
$ git <verb> --help
$ git help
```

3. Git 저장소

Git 저장소를 만드는 방법은 두 가지다. 기존 디렉토리를 Git 저장소로 만드는 방법과 원격 저장소로부터 Clone 하는 방법이 있다.

```
$ cd /c/user/your_repository
$ git init
```

기존 디렉토리로 이동하여 'git init' 명령을 내리면 .git 이라는 하위 디렉토리를 만든다.

```
$ git clone [가져올 원격 저장소 주소] (폴더명 지정)
$ git clone https://github.com/libgit2/libgit2
$ git clone https://github.com/libgit2/libgit2 mylibgit
```

git clone [url] 명령은 원격 저장소를 로컬 저장소로 복제한다. 위의 명령은 'libgit2'라는 디렉토리를 만들고 그 안에 .git 디렉토리를 만든다. 그리고 원격 저장소의 데이터를 모두 가져와서 자동으로 가장 최신 버전을 Checkout 해 놓는다. 나만의 디렉토리 이름으로 저장할 수도 있다.

4. 파일 버전 관리

파일을 최초로 생성하면 Untracked 상태이다. 새로 만든 파일을 추적하기 위해 'git add' 명령으로 Staged 상태로 만든다.


```
$ git add README
$ git status
On branch master
No commits yet
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   README
```

Staged 상태의 파일을 커밋한다. -m 옵션으로 인라인 메시지로 첨부할 수 있고, 옵션을 적지 않으면 지정된 편집기가 열린다. 'git add' 명령으로 Staged 상태로 만든 파일들만 커밋할 수 있음에 주의한다.

```
$ git commit
$ git commit -m "fixed bug #2"
```

매번 'git add'를 사용하여 Staged 상태로 만들기 번거로울 때 추적가능한 파일에 한하여 -a 옵션으로 Modified 상태의 파일을 커밋할 수 있다. 편리하긴 하지만, 불필요한 파일까지 커밋되지 않도록 주의한다.

```
$ git commit -a
```

git diff 명령을 실행하면 Modified 상태의 파일의 변경 내용을 볼 수 있다. --staged 옵션을 주면 Staged 상태의 파일 변경 내용을 볼 수 있다. 비교 대상은 Committed 이다.

```
$ git diff
$ git diff --staged
```

제 2 장 Git Clients 도구

제1절 Git Clients 도구

Git을 사용하는 방법은 많다. CLI(Command Line Interface)로 사용할 수도 있고 GUI(Graphic User Interface)를 사용할 수도 있지만, CLI를 사용해야만 Git의 모든 기능을 사용할 수 있다. Git 클라이언트 툴로는 CLI의 Git Bash 또는 GUI의 SourceTree, Github Desktop, TortoiseGit 등과 표준프레임워크 개발환경에 탑재되어 있는 Egit 플러그인이 있다.

Git의 모든 기능을 사용할 수 있는 CLI 사용을 기본으로 하고, 프로젝트 히스토리를 시각화해서 무슨 일이 있었는지 살펴볼 때나 기본적인 기능을 쉽게 작업하려 한다면, 추가적으로 GUI 도구를 사용한다.

제2절 Git Bash 사용⁵⁾

Git SCM (Source Code Management)을 설치할 때 구성 설정에서 Windows Explorer Integration > Git Bash Here를 선택하면 Git Bash를 사용할 수 있다. 설치 시 환경 설정에서 'Git from the command line and also from 3rd-party software'를 선택하면 윈도우 명령창 cmd 또는 파워셸등에서도 git명령어를 사용할 수 있다.

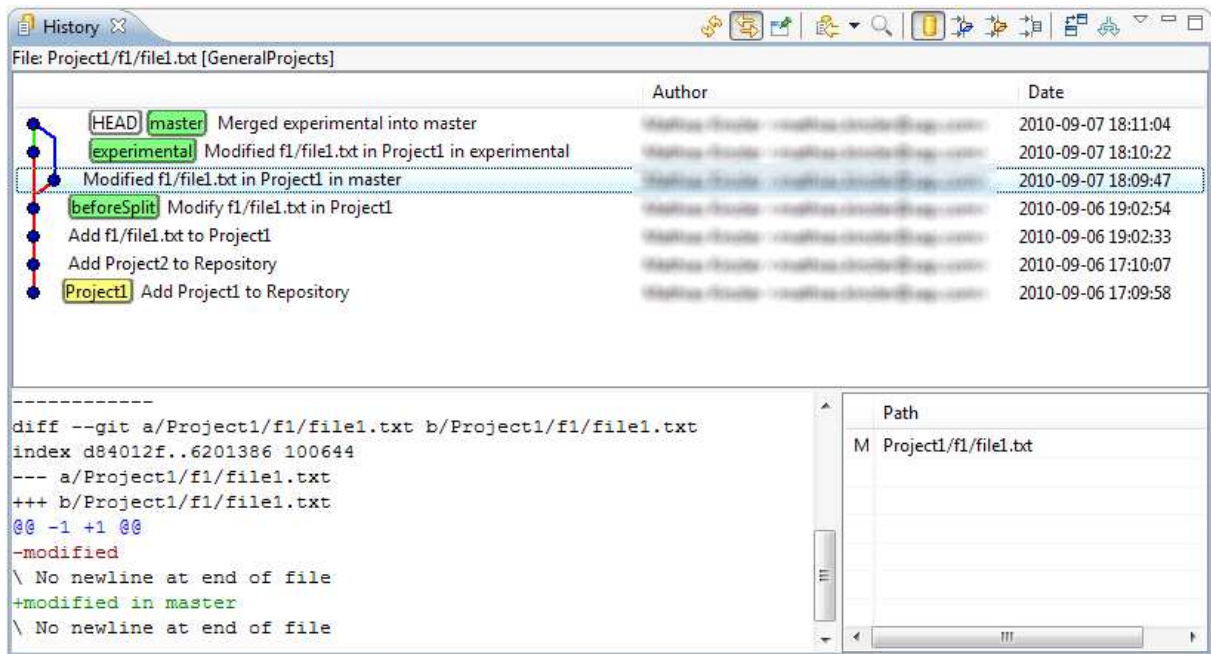
제3절 표준프레임워크 개발환경에서 사용

프로젝트에 Git 로컬 저장소가 존재하면, 프로젝트를 선택하여 마우스 우클릭하여 팝업되는 컨텍스트 메뉴에서 Team 메뉴에 Git⁶⁾의 기능이 있다. 프로젝트에 아직 Git 로컬 저장소가 존재하

5) <https://git-scm.com/book/en/v2/Appendix-A%3A-Git-in-Other-Environments-Git-in-Bash> 참조

지 않으면 'Share Project...' 선택 후 Repository Type을 Git으로 선택하여 Git 로컬 저장소를 설정하면 된다. 자세한 사용법은 Egit의 사용자 가이드 https://wiki.eclipse.org/EGit/User_Guide를 참조한다.

Windows > Open Perspective에서 Git을 선택하여 열면 Git Repositories 탐색창과 History View로 화면이 재구성된다.



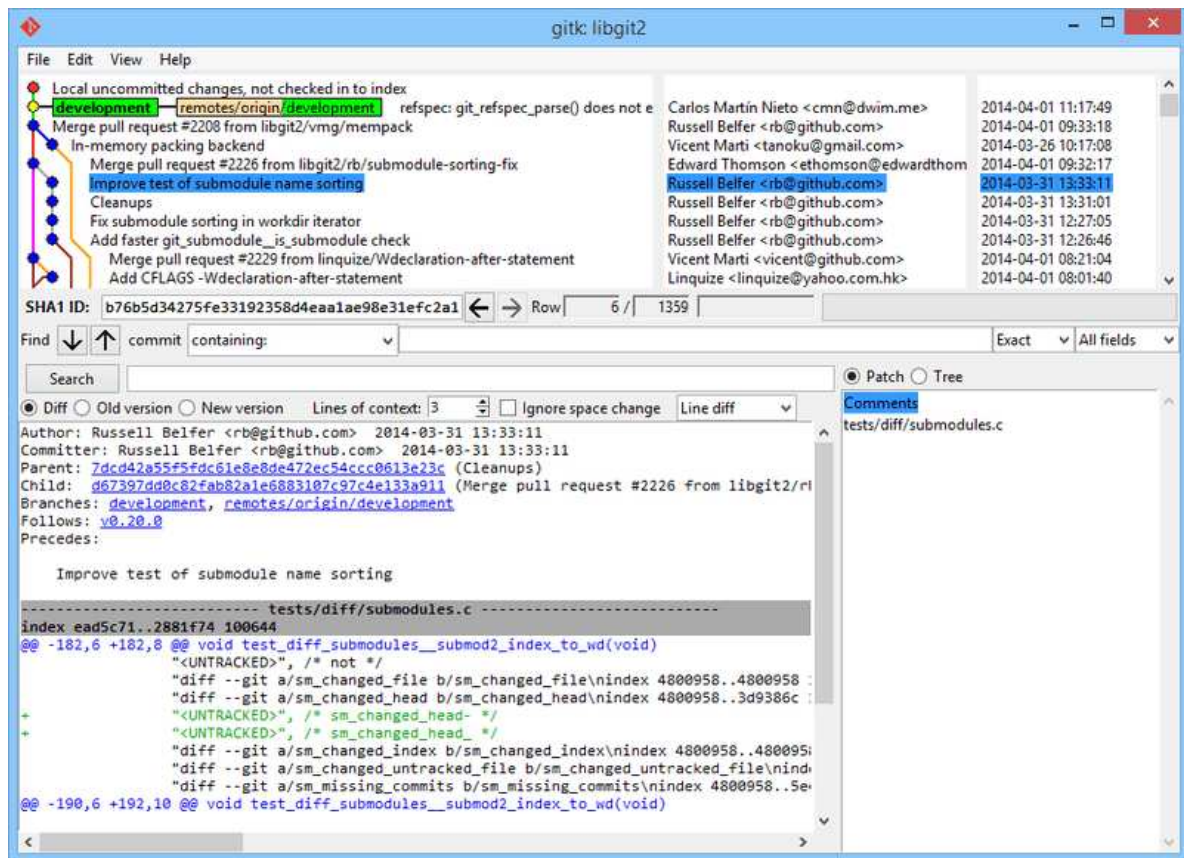
제4절 3rd Party Git GUI Clients 사용

1. Gitk와 Git GUI⁷⁾

Git SCM 설치 시 gitk와 git-gui는 함께 설치된다. gitk는 히스토리 표시 도구이다.

6) https://wiki.eclipse.org/EGit/User_Guide 참조

7) <https://git-scm.com/book/en/v2/Appendix-A%3A-Git-in-Other-Environments-Graphical-Interfaces> 참조

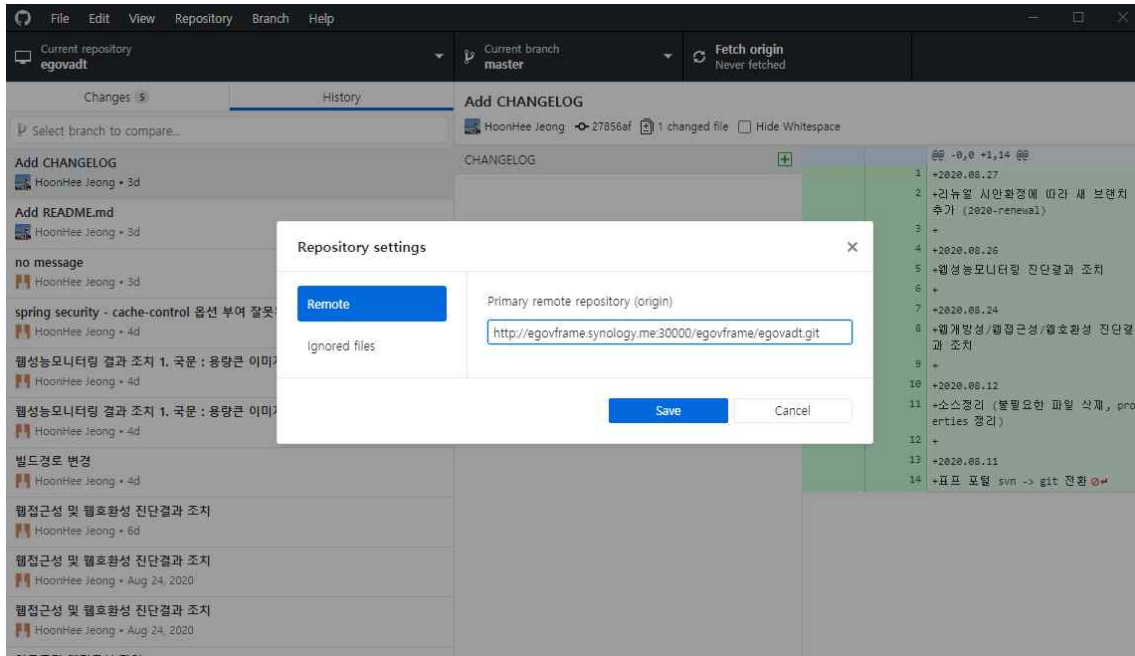


2. GitHub Desktop⁸⁾

GitHub가 배포하는 공식 클라이언트로 GitHub를 원격저장소

8) <https://desktop.github.com/> 참조

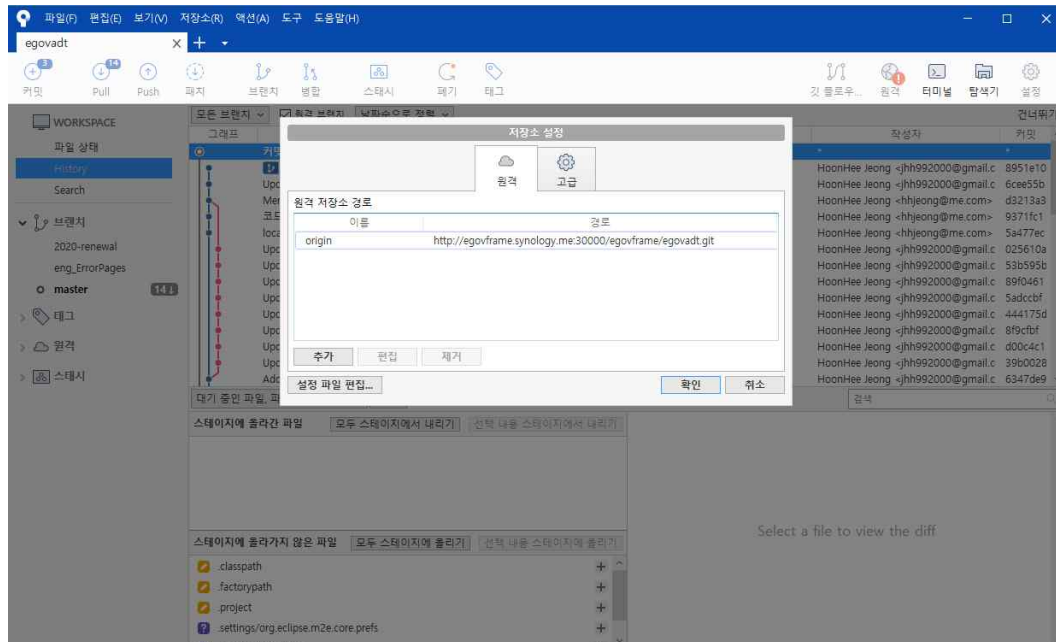
로 사용할 때 유용한 도구이다. 단 하나의 원격저장소만 설정할 수 있고, 커밋 히스토리 그래프 없고 단순한 기능들만 지원하는 단점이 있다.



3. SourceTree⁹⁾

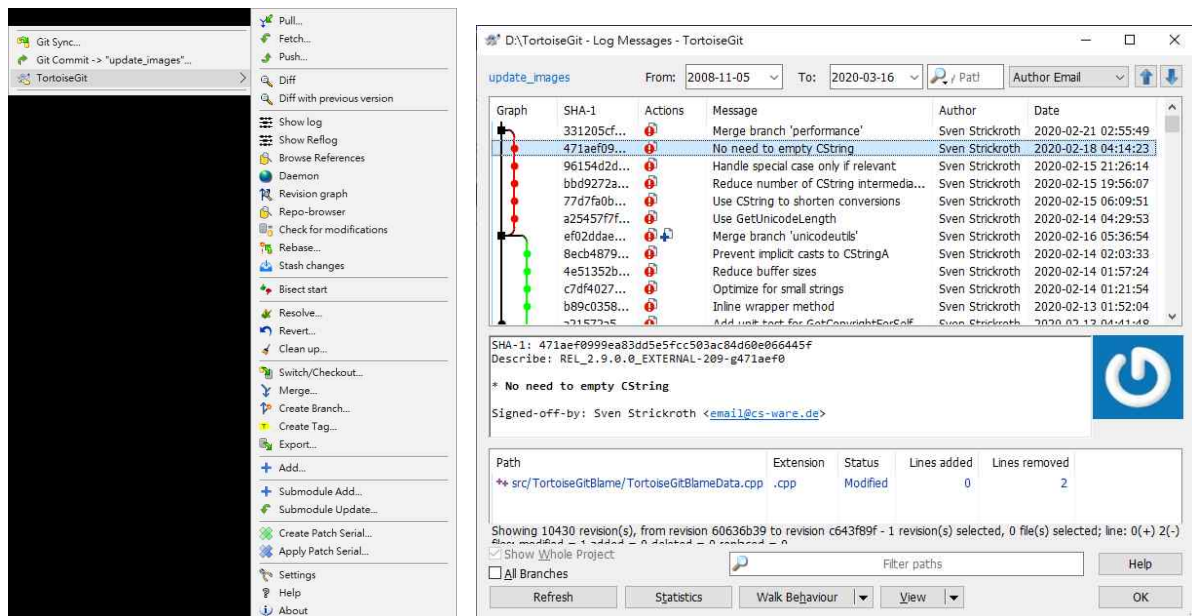
SourceTree는 JIRA와 Bitbucket 개발사 Atlassian이 만든 무료 엔터프라이즈급 Git Client이다. UI에서 GitHub Desktop보다 더 많은 기능을 제공한다. SourceTree를 사용하기 전에 Atlassian 계정을 만들어야 한다.

9) <https://www.sourcetreeapp.com/> 참조



3. TortoiseGit¹⁰⁾

TortoiseGit은 윈도우 셸 인터페이스로 TortoiseSVN 기반으로 한 오픈소스이다. 윈도우 탐색기의 컨텍스트 메뉴로 주로 사용된다.



10) <https://tortoisegit.org/> 참조

제 3 장 Git 주요 기능

제1절 브랜치¹¹⁾

개발을 하다 보면 코드를 여러 개로 복사해야 하는 일이 자주 생긴다. 코드를 통째로 복사하고 나서 원래 코드와는 상관없이 독립적으로 개발을 진행할 수 있는데, 이렇게 독립적으로 개발하는 것이 브랜치다. Git은 데이터를 Change Set이나 변경사항으로 기록하지 않고 일련의 스냅샷으로 기록한다. 커밋하면 Git은 Staging Area에 있는 데이터의 스냅샷에 대한 포인터가, 저자나 커밋 메시지 같은 메타데이터, 이전 커밋에 대한 포인터 등을 포함하는 커밋 개체를 저장한다. 브랜치를 위한 디렉토리를 만들고 파일들을 복사하는 과정을 거치는 SVN과는 다르게 Git은 'git branch' 명령만으로 마지막 스냅샷을 이용하여 브랜치를 만들어내기 때문에, 빠르고 편리한 브랜치와 병합이 가능하다. 따라서, 수정량이 작은 브랜치를 빠르게 만들어 병합하기가 가능하기 때문에 충돌 시 해결해야 할 일도 줄일 수 있고, 브랜치별로 다양한 테스트를 과감하게 진행할 수 있게 해 준다.

1. 브랜치 관리

```
$ git branch [브랜치 이름] ▶ 브랜치 생성  
$ git branch ▶ 브랜치 목록 조회  
$ git branch -d [브랜치 이름] ▶ 브랜치 삭제
```

2. 브랜치 전환

```
$ git checkout [브랜치 이름]  
$ git checkout -b [브랜치 이름] -- 브랜치 작성과 전환을 한번에 실행
```

브랜치를 전환하면 HEAD¹²⁾ 포인터가 전환된 브랜치를 가리키게 된다. commit 하지 않은 변경 내용이나 새롭게 추가한 파일이 인덱스와 작업 트리에 남아 있는 채로 다른 브랜치로 전환

11) Pro Git Second Edition, Scott Chacon and Ben Straub 55쪽

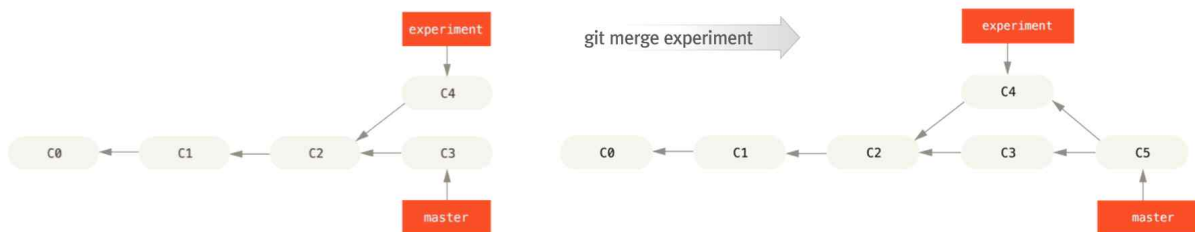
12) 현재 작업하는 로컬 브랜치를 가리키는 Git의 특수한 포인터

하면, 그 변경 내용은 기존 브랜치가 아닌 전환된 브랜치에서 commit 할 수 있다. 단, commit 가능한 변경 내용 중 전환하려는 브랜치에서도 한 차례 변경이 되어 있는 경우에는 전환이 실패할 수 있다. 이 경우 이전 브랜치에서 commit하지 않은 변경 내용을 commit 하거나, stash를 이용해 일시적으로 변경 내용을 다른 곳에 저장하여 충돌을 피하게 한 뒤 전환해야 한다.

3. 브랜치 병합

merge를 사용하여 여러 브랜치를 하나로 병합할 수 있다.

```
$ git merge [브랜치 이름]
```

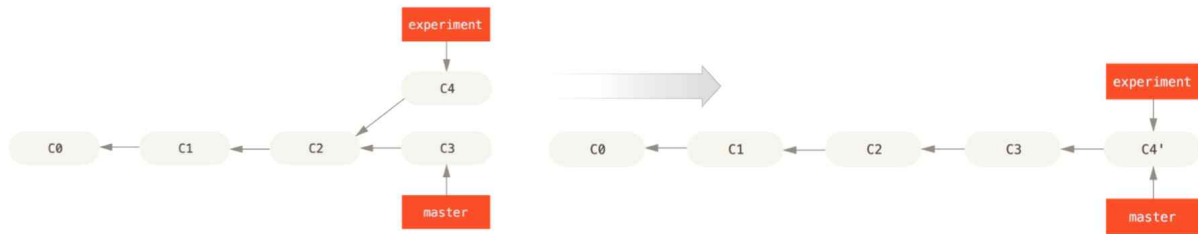


두 브랜치의 마지막 commit 두 개 (C3, C4)와 공통 조상(C2, base라 함)을 사용하는 3-way Merge로 새로운 commit(C5)을 만든다. 병합하는 브랜치가 많아질수록 히스토리가 복잡하여 찾기 어려워지는 단점이 있다. 병합 후에는 더 이상 experiment 브랜치가 필요 없다. 'git branch -d experiment'로 삭제한다.

rebase¹³⁾를 사용하면 불필요한 merge 이력을 남기지 않고 좀 더 깨끗한 히스토리로 병합할 수 있다. 브랜치가 많아도 Rebase로 병합하고 나면 모든 작업이 선형으로 차례대로 수행된 것처럼 보인다.

```
$ git checkout experiment
$ git rebase master
$ git checkout master
$ git merge experiment
```

13) Rebase 하기 : Pro Git Second Edition, Scott Chacon and Ben Straub 85쪽



master C2와 C4의 변경사항을 Patch로 저장해 두고 Rebase (조상을 재설정하여 일렬로 정렬) 한다. 아직은 master 브랜치에 변경사항이 적용되지 않은 상태이다. 그리고 나서 merge를 실행하여 master 브랜치를 Fast-forward 시킨다. 즉, 병합하려는 브랜치 experiment의 임시로 저장해둔 모든 diff(Patch)를 master 가 가리키는 C3 뒤에 차례대로 diff를 적용한다. Fast-forward 병합이란 서로 다른 상태를 병합하는 것이 아니라 master를 experiment 위치로 이동만 해도 되는 상태를 말한다.

Checkout 하지 않고 바로 experiment 브랜치를 master 브랜치로 다음과 같이 Rebase할 수 있다.

```
$ git rebase master experiment
```

결국 Merge를 하나 Rebase를 하나 최종 결과물은 같고 커밋 히스토리만 다르다. Rebase는 experiment 브랜치의 변경사항을 순서대로 적용하면서 병합하고 Merge는 두 브랜치의 최종결과만을 가지고 병합한다. 로컬 브랜치에서 작업할 때는 히스토리를 정리하기 위해서나 Rebase 해도 되지만, 원격 저장소의 rebase 대상 브랜치에 타인의 변경 커밋이 있을 경우에는 Merge를 이용하는 것이 좋다.

제2절 원격 저장소

1. Remote

git remote 명령으로 원격 저장소를 관리할 수 있다.

```
$ git remote -v ▶ 원격 저장소 목록 조회 (URL포함)
$ git remote show [원격 저장소 이름] ▶ 구체적인 정보 조회
$ git remote rename [원격 저장소 이름] [원격 저장소 바꿀 이름] ▶ 원격 저장소 이름 변경
$ git remote remove [원격 저장소 이름] ▶ 원격 저장소 삭제
$ git remote add [원격 저장소 이름] [원격 저장소 URL] ▶ 별칭으로 원격 저장소 연결
```

2. Pull, Fetch

pull은 원격 저장소로부터 다운로드하여 로컬 저장소와 merge 한다. fetch는 원격 저장소로부터 로컬에 없는 데이터를 모두 가져오지만, 병합은 하지 않기 때문에 개발자가 별도로 merge를 해야 한다. 따라서, HEAD 포인터는 pull 완료 후에는 로컬 브랜치와 원격 저장소 origin/master 가 같은 위치를 가리키지만, fetch 후에는 로컬 브랜치의 최근 커밋 위치를 가리키고 있다.

```
$ git fetch [원격 저장소 이름]
$ git pull [원격 저장소 이름]
```

3. Push

프로젝트를 공유하려면 원격 저장소로 Push 할 수 있다. 단, 원격 저장소에 쓰기 권한 있어야 하며, clone 한 이후로 아무도 원격 저장소에 Push하지 않았을 경우에만 할 수 있다. 이미 다른 사람이 Push한 후에는 다른 사람이 작업한 것을 가져와서 Merge한 후에 Push를 할 수 있다.

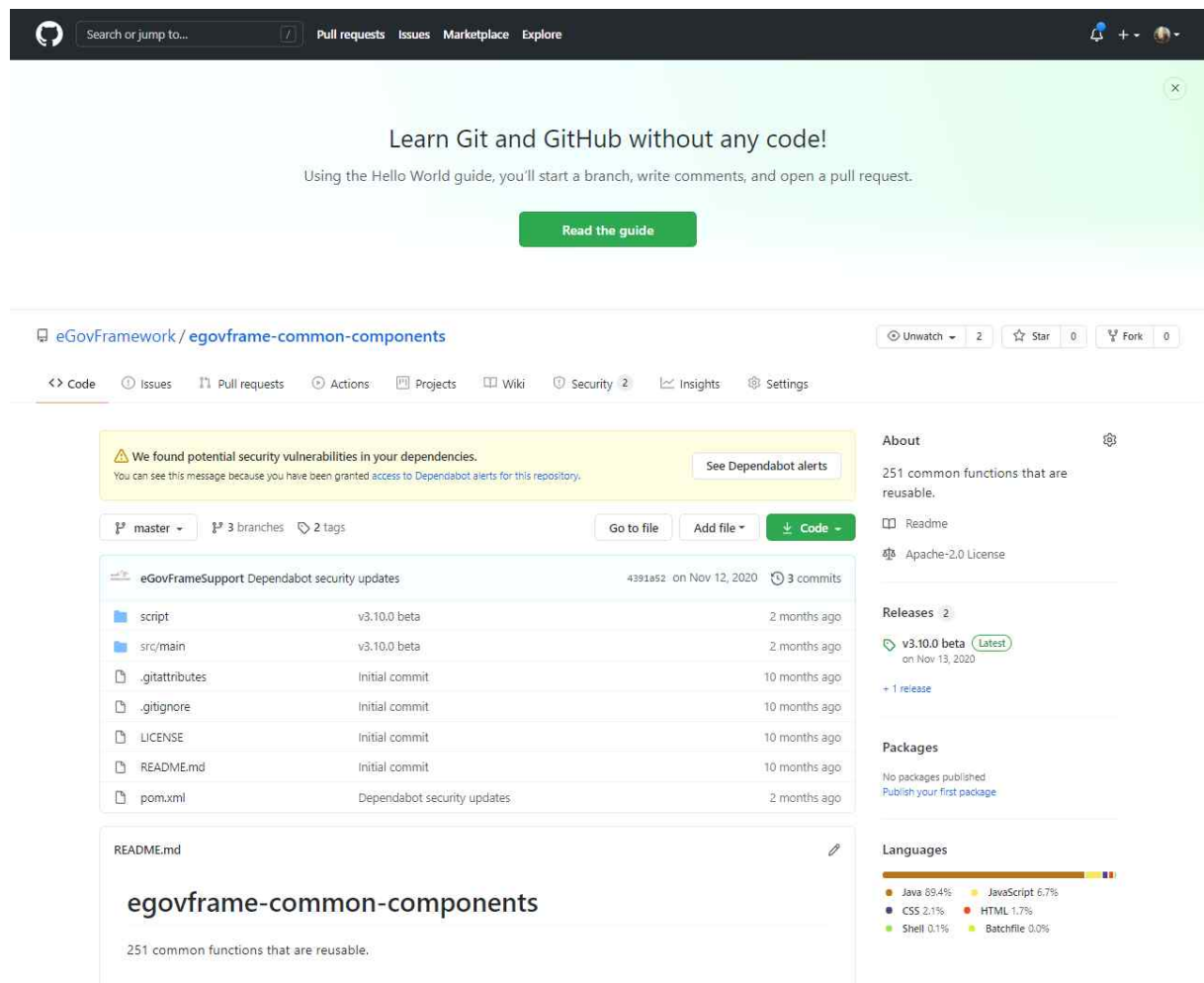
```
$ git push [원격 저장소 이름] [브랜치 이름]
$ git push [원격 저장소 이름] --delete [브랜치 이름]
```

git push 명령어에 --delete 옵션을 사용하여 원격 저장소의 브랜치를 삭제할 수 있다.

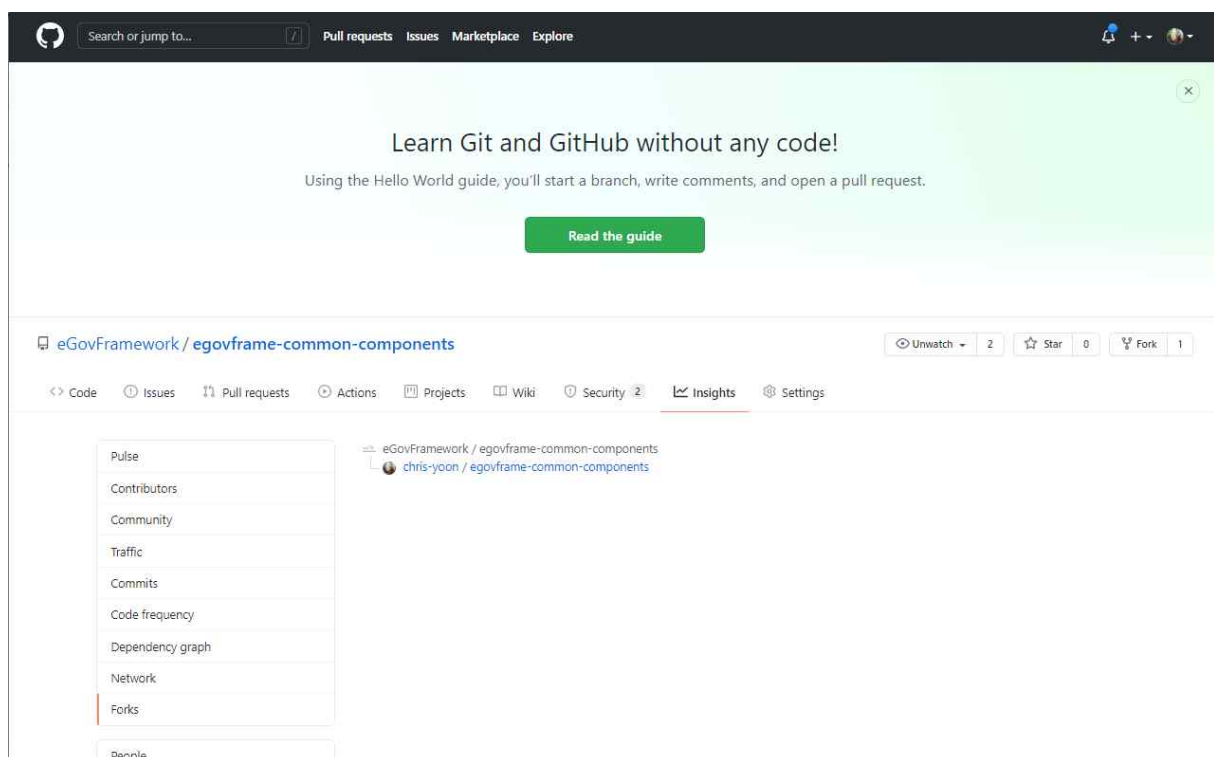
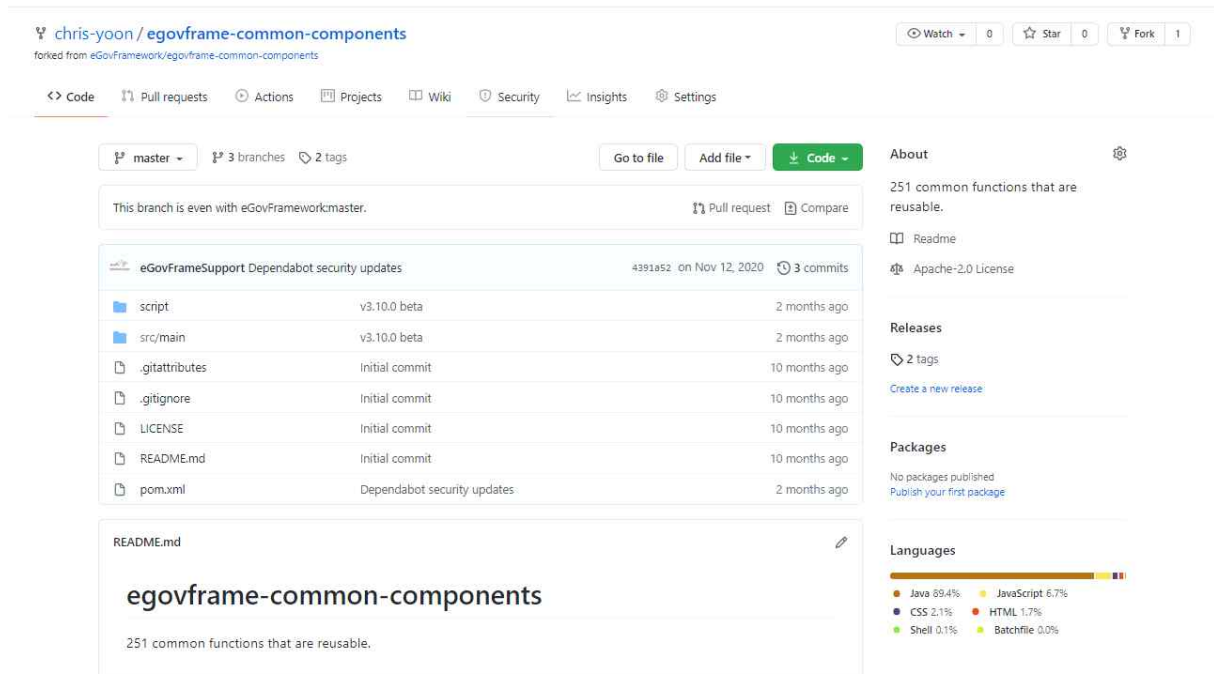
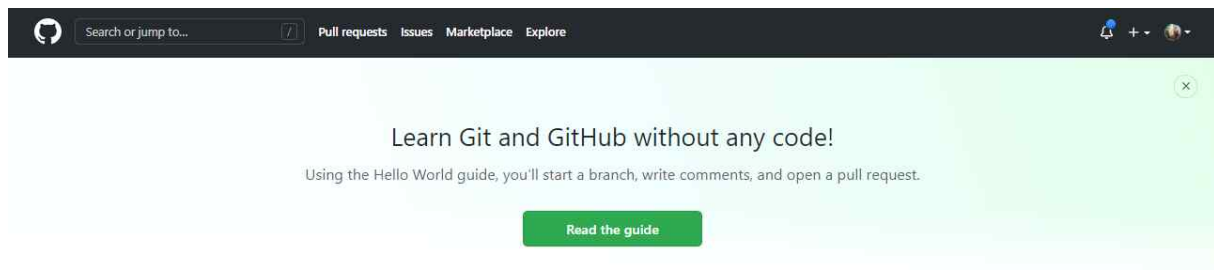
4. Fork

Fork는 Git의 기능이 아니라, Github 또는 Gitlab 등의 원격 저장소에서 Git의 기능을 추상화해서 제공하는 기능으로 원격 저장소로 로그인해서 사용할 수 있다. Fork는 타인의 원격 저장소 (upstream¹⁴)로부터 자신의 원격 저장소로 통째로 복제하는 기

능이다. Fork한 저장소는 원본 원격 저장소와 연결되어 있어 Forked 원격 저장소로 반영된다. 이때 로컬 저장소로 Fetch, Rebase 과정이 필요하다. 타인의 원격 저장소로 바로 Push할 수 없기 때문에 Fork 하여 코드를 수정하고 반영해 본 후 원래의 원격 저장소로 Pull Request하는 것이다.



위의 GitHub 표준프레임워크의 공통컴포넌트 프로젝트에서 우측 상단의 Fork 버튼을 클릭하면, 아래 화면처럼 개발자의 GitHub 저장소로 복사가 된다.



Fork 버튼 우측 숫자를 클릭하면, 표준프레임워크의 공통컴포

년트 소스가 어느 저장소에 Fork 되었는지 알 수 있다. 이제부터는 개발자의 GitHub 저장소에 복사되었기에 개발자 로컬 저장소로 Clone하여 소스 수정 및 Commit을 할 수 있다.

5. Pull Request (Merge Request)

Forked 원격 저장소를 로컬 저장소로 clone¹⁵⁾하여 내려받은 프로젝트를 수정하여 add, commit 그리고 push 까지 하여 forked 원격 저장소까지 반영할 수 있지만, 원본이 되는 원격 저장소(upstream)에는 권한이 없다면 Push를 할 수 없기 때문에 대신 반영해 달라고 요청하는 Pull Request를 할 수 있다. Pull Request는 Git의 기능이라기보다 원격 저장소의 기능이라 할 수 있다. GitLab에서는 Merge Request라는 용어를 사용하고 있다.

Learn Git and GitHub without any code!

Using the Hello World guide, you'll start a branch, write comments, and open a pull request.

[Read the guide](#)

chris-yoon / egovframe-common-components

forked from eGovFramework/egovframe-common-components

Watch 0 Star 0 Fork 1

Code Pull requests Actions Projects Wiki Security Insights Settings

master 3 branches 2 tags

Go to file Add file Code

This branch is 2 commits ahead, 3 commits behind eGovFramework/master. Pull request Compare

chris-yoon Pull Request를 하기 위한 테스트입니다. aceb3af · 5 minutes ago 5 commits

File	Commit	Time
script	v3.10.0 beta	2 months ago
src/main	v3.10.0 beta	2 months ago
.gitattributes	Initial commit	10 months ago
.gitignore	Initial commit	10 months ago
LICENSE	Initial commit	10 months ago
README.md	Pull Request를 하기 위한 테스트입니다.	5 minutes ago
pom.xml	Dependabot security updates	2 months ago

README.md

egovframe-common-components

251 common functions that are reusable.

개발자가 소스를 수정 후 Pull Request(Merge Request)를 하는 테스트입니다.

About: 251 common functions that are reusable. Readme Apache-2.0 License

Releases: 2 tags Create a new release

Packages: No packages published Publish your first package

Languages: Java 88.4%, JavaScript 6.7%, CSS 2.1%, HTML 1.7%, Shell 0.1%, Batchfile 0.0%

15) 나의 원격 저장소를 origin이라 함

개발자가 소스를 수정 후 Code 녹색버튼 아래 Pull Request 버튼을 클릭하면 아래와 같이 표준프레임워크 저장소의 Create pull request 화면으로 이동된다.

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

base repository: eGovFramework/egovframe-co... base: master ← head repository: chris-yoon/egovframe-commo... compare: master

✖ Can't automatically merge. Don't worry, you can still create the pull request.

Discuss and review the changes in this comparison with others. [Learn about pull requests](#)

Create pull request

2 commits 1 file changed 0 comments 1 contributor

Commits on Jan 14, 2021

- Update README.md Verified 828af2a
- Pull Request를 하기 위한 테스트입니다. Verified ac8b3af

Showing 1 changed file with 3 additions and 0 deletions.

Unified Split

3 README.md

```
@@ -1,2 +1,5 @@
1 1 # egovframe-common-components
2 2 251 common functions that are reusable.
3 3 +
4 4 +
5 5 + 개발자가 소스를 수정 후 Pull Request(Merge Request)를 하는 테스트입니다.
```

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base repository: eGovFramework/egovframe-co... base: master ← head repository: chris-yoon/egovframe-commo... compare: master

✖ Can't automatically merge. Don't worry, you can still create the pull request.

수정사항 반영요청

Write Preview

표준프레임워크의 공통컴포넌트의 수정사항을 반영해 주시면 감사하겠습니다.

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

☒ Allow edits by maintainers

Create pull request

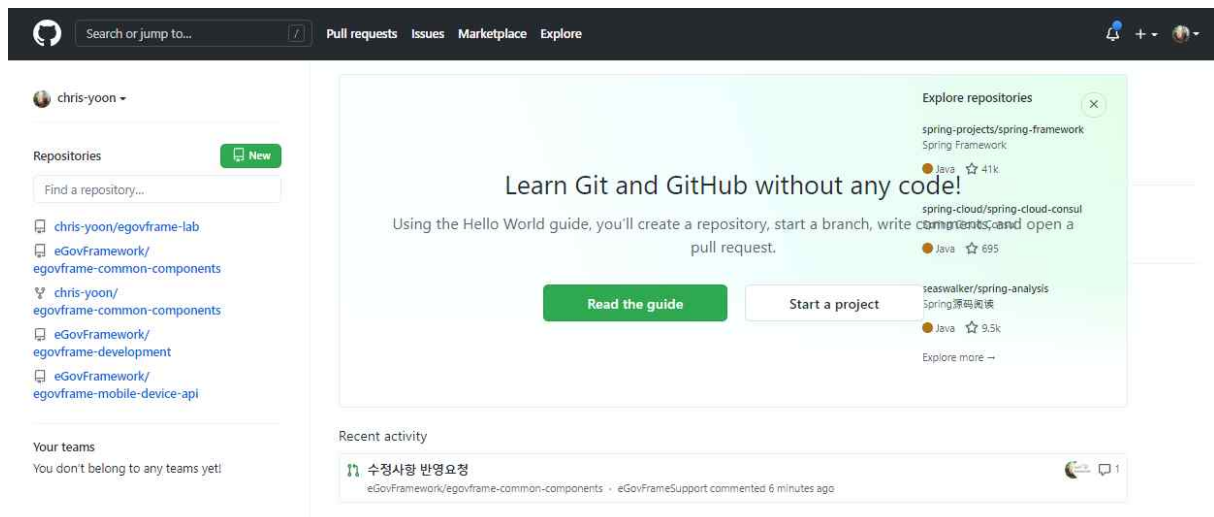
Reviewers: No reviews

Assignees: No one—assign yourself

Labels: None yet

Projects: None yet

Milestone: No milestone



개발자 GitHub 데시보드에 최근 활동내역이 표시되며, 검토내용을 확인할 수 있으며, 이메일로도 자동으로 전송된다.

제3절 태그16)

태그는 보통 릴리즈 버전을 매길 때 사용한다.

1. 태그 조회

```
$ git tag
```

2. 태그 붙이기

```
$ git tag -a v3.10 -m "eGovFrame version 3.10"
$ git tag v3.10-alpha
```

-a 옵션을 붙이면 Git 데이터베이스에 태그를 만든 사람의 이름, 이메일과 태그를 만든 날짜, 그리고 태그 메시지도 저장한다. 하지만, 임시로 생성하는 태그이거나 태그 상세정보를 유지할 필요가 없는 경우는 -a 나 -m 옵션을 주지 않는다.

3. 태그 공유

```
$ git push [원격 저장소 이름] [태그 이름]
$ git push [원격 저장소 이름] --tags
```

git push 만으로는 원격 저장소에 태그를 전송하지 않고, 태

16) Pro Git Second Edition, Scott Chacon and Ben Straub 49쪽

그 이름을 붙여주어야 한다. 한 번에 태그 여러개를 Push하려 한다면, --tags 옵션을 추가한다.

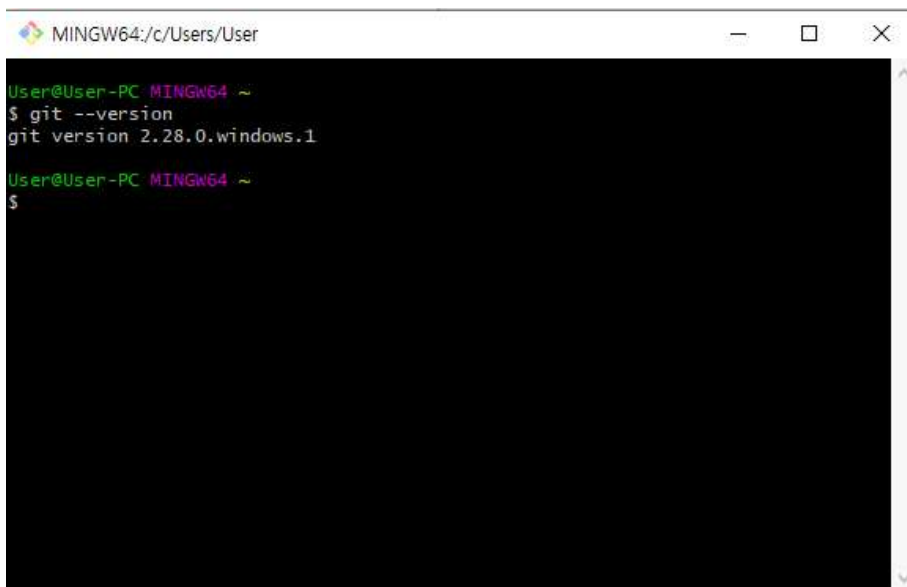
제 4 장 표준프레임워크 오픈소스 기여 하기

제1절 Git 사용 환경 설정

1. Git 설치

표준프레임워크 개발환경을 설치¹⁷⁾한 후, 본 가이드 '[제1장 4절 Git SCM 설치](#)'를 참조하여 Git을 설치한다.

제대로 설치되었는지 확인하기 위해 '윈도우키'+ 'Q'를 누르고 GitBash라고 검색한 후 실행하면 다음과 같은 윈도우가 뜨며, 'git --version' 입력 후 엔터를 쳐서 버전이 확인되면 정상적으로 설치된 것이다. 윈도우 명령 프롬프트 cmd 창에서도 동일하게 확인 가능하다.

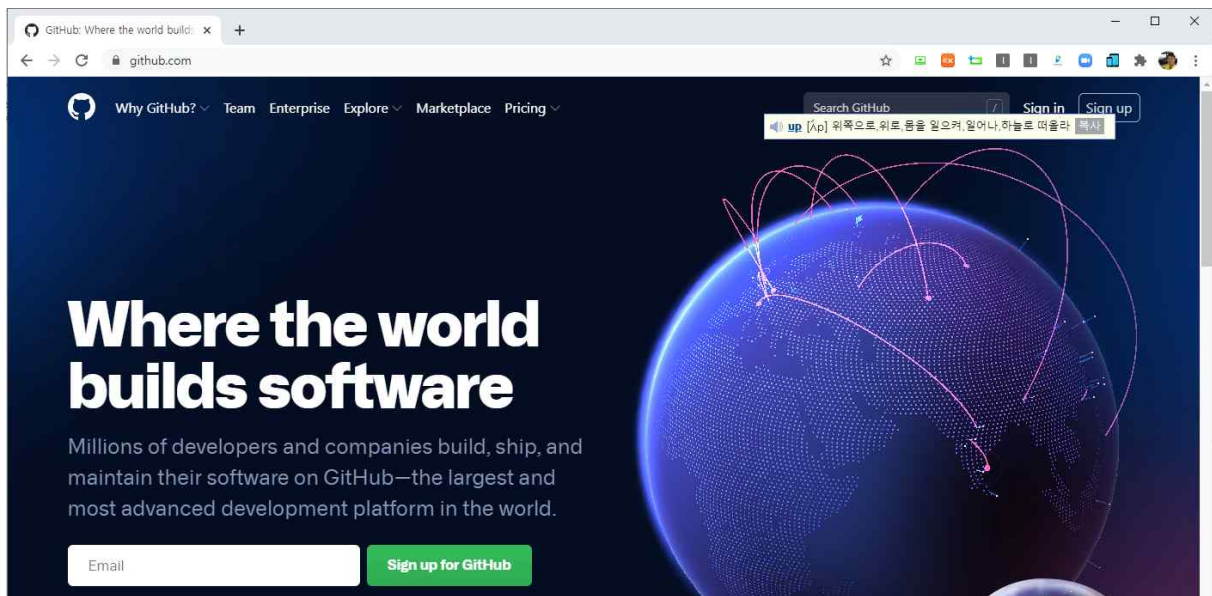


```
MINGW64: c:/Users/User
User@User-PC MINGW64 ~
$ git --version
git version 2.28.0.windows.1
User@User-PC MINGW64 ~
$
```

2. GitHub 계정 등록

<https://github.com/>에 방문해서 [Sign up] 버튼을 클릭 후, 사용자명, 이메일, 비밀번호를 입력하고, [Create account]를 클릭한다.

17) <https://www.egovframe.go.kr/wiki/doku.php?id=egovframework:dev3.10:clntinstall> 참조



Join GitHub

Create your account

Username *

Email address *

Password *

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more](#)

Email preferences

☐ Send me occasional product updates, announcements, and offers.

Verify your account

다음에 보이는 'Welcome to GitHub' 페이지에서 질문에 대한 답변을 한 다음, [Complete setup] 버튼을 클릭한다.

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

Selected plan: Free

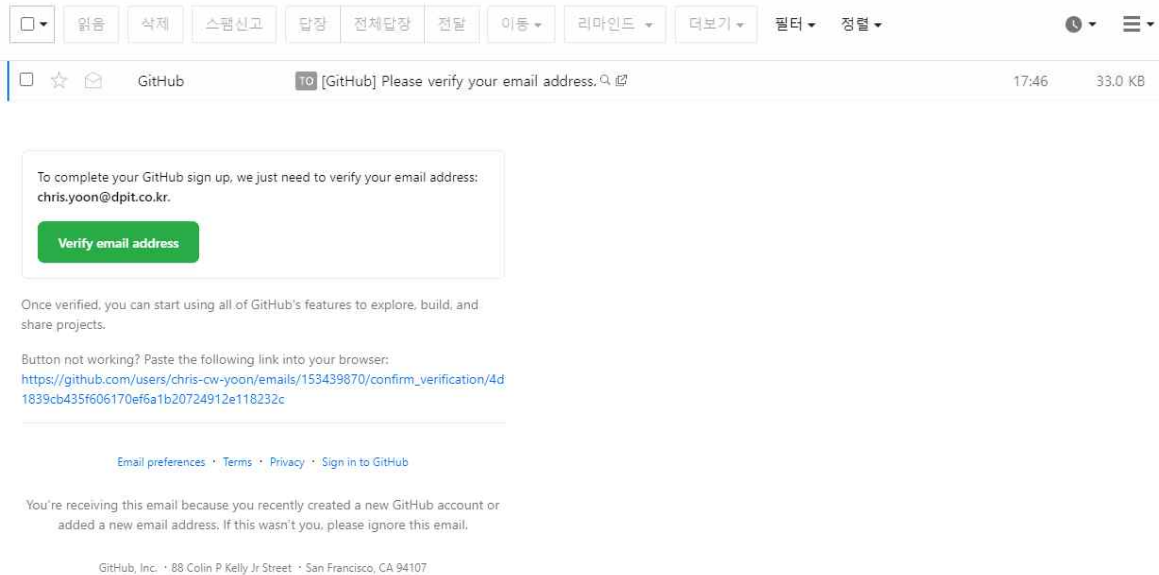
Welcome to GitHub

Woohoo! You've joined millions of developers who are doing their best work on GitHub. Tell us what you're interested in. We'll help you get there.

What kind of work do you do, mainly?

Software Engineer I write code	Student I go to school
Product Manager I write specs	UX & Design I draw interfaces
Data & Analytics I write queries	Marketing & Sales I look at charts
Teacher	Other

받은메일함에 다음과 같은 도착한 이메일을 클릭하여 [Verify email address]를 클릭하면, GitHub을 사용할 수 있는 상태가 된다.

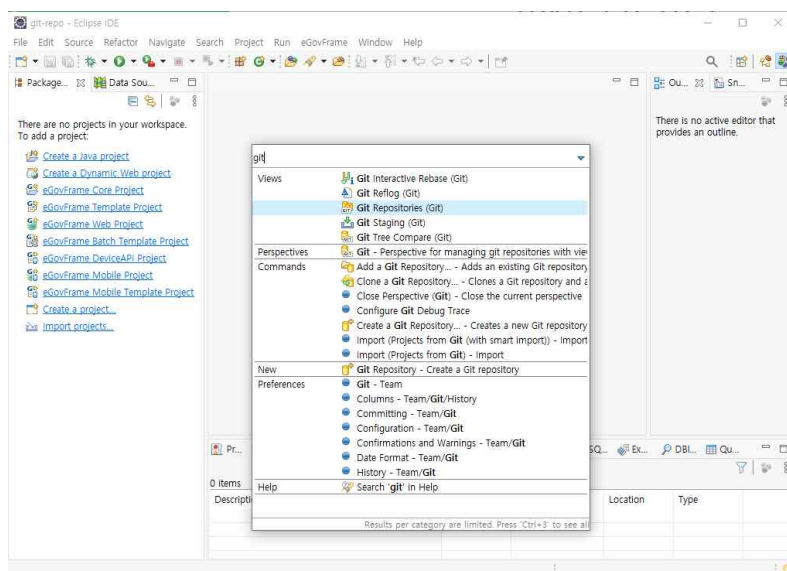


3. 개발환경 eGovFrame Perspective

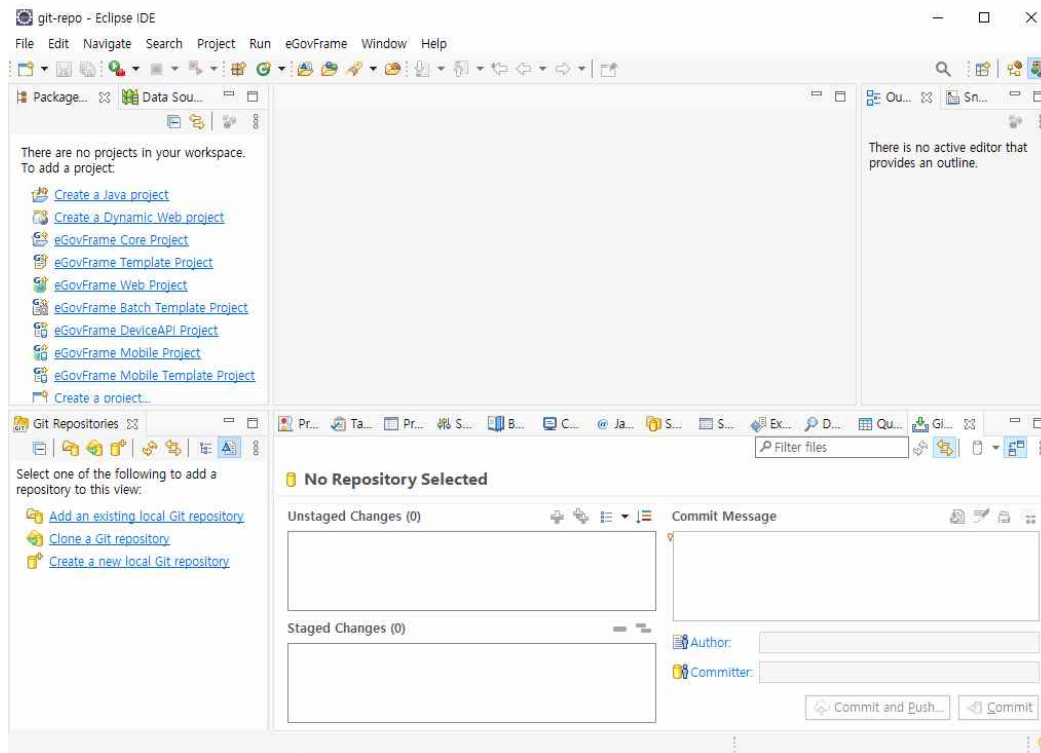
개발환경에서 Window > Perspective > Open Perspective > Other > eGovFrame을 선택한 후 [Open] 버튼을 클릭한다.

4. 개발환경에 Git 환경 설정

개발환경에서 'Ctrl+3' 버튼 (맥OS는 'Command+3')을 클릭한 후 'Git'으로 검색, 'Git Repositories'와 'Git Staging'을 선택한다.



개발환경에 다음과 같이 Git 관련 View를 확인한다.

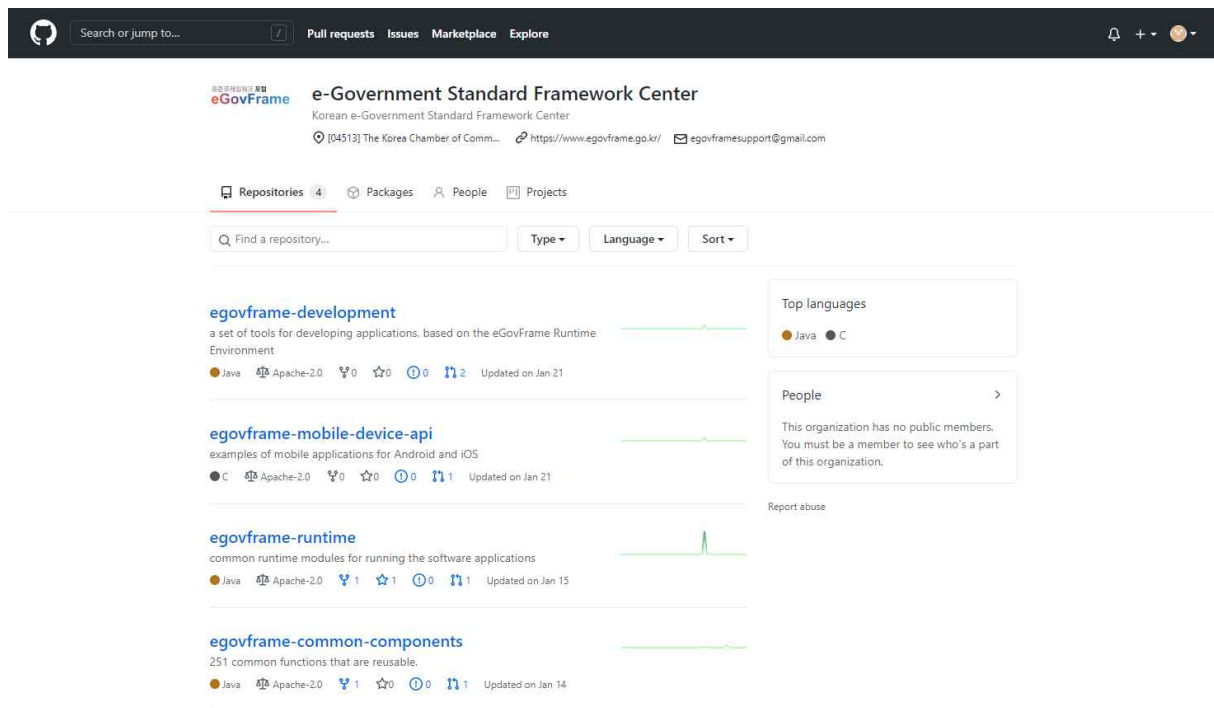


제2절 표준프레임워크 소스 복제

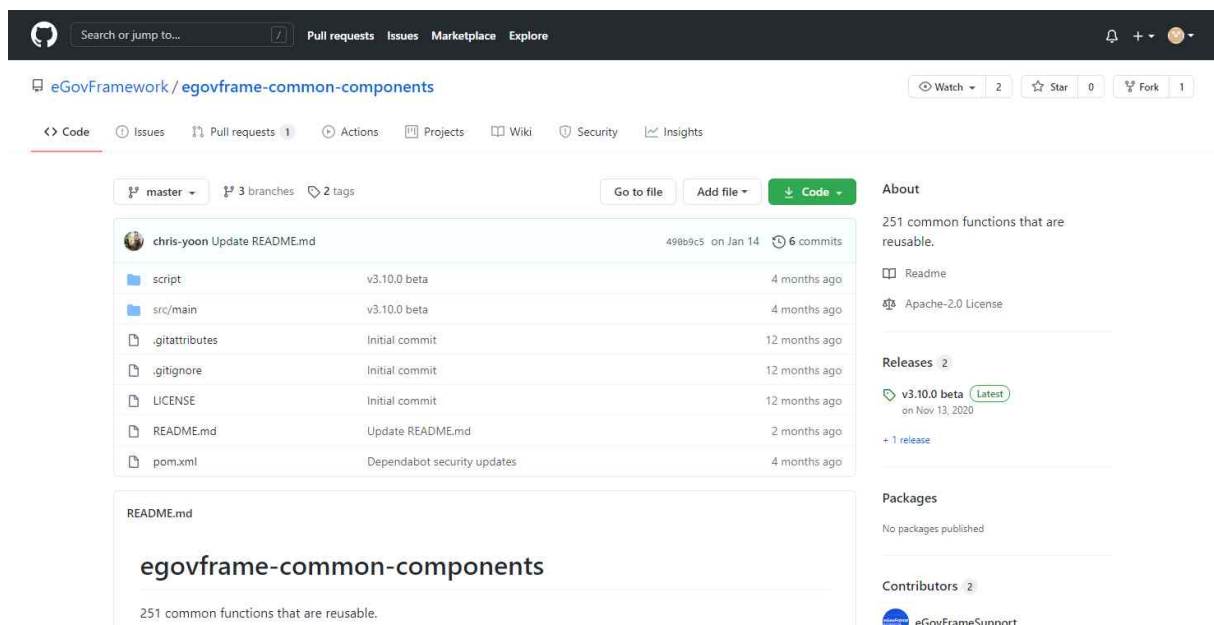
표준프레임워크 원격저장소의 Push 권한이 없기 때문에 나의 원격저장소로 Fork 한 후 나의 원격저장소를 Clone 하여 소스를 복제할 것이다.

1. GitHub Fork

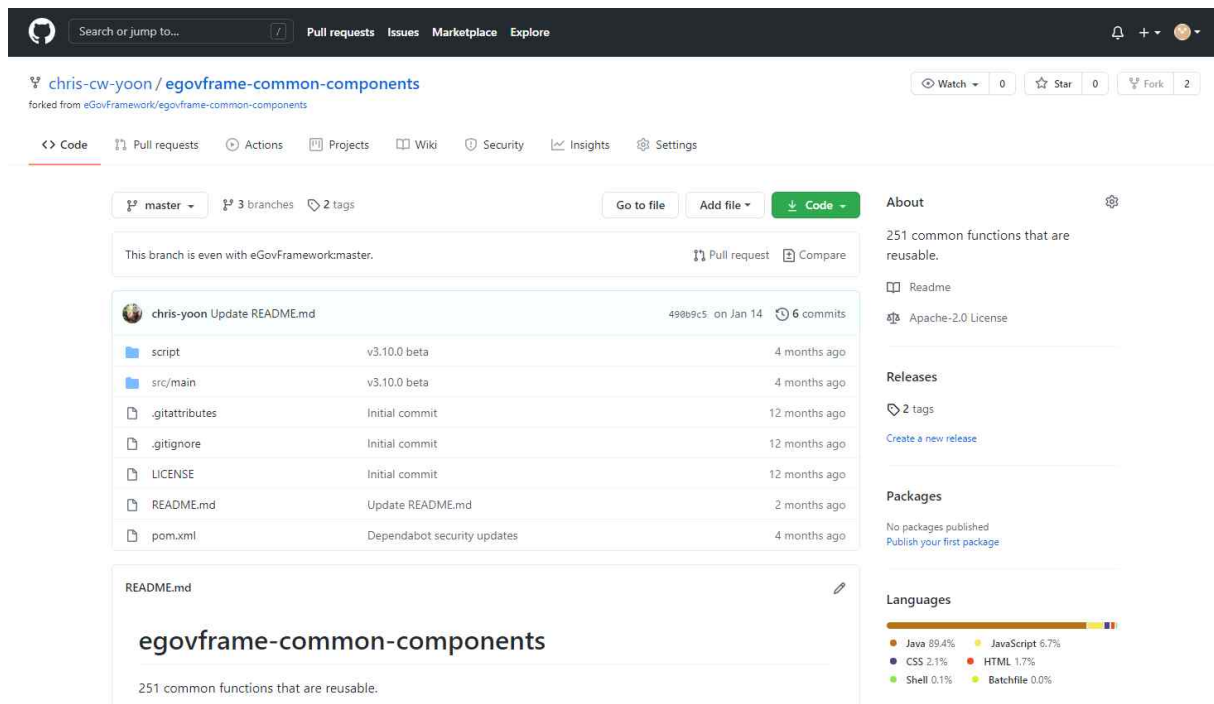
<https://github.com/>에 로그인 한 후, 표준프레임워크 원격저장소인 <https://github.com/eGovFramework> 에 접속한다.



복제하려는 프로젝트를 선택한 후, 우측 상단의 [Fork]버튼을 클릭한다.

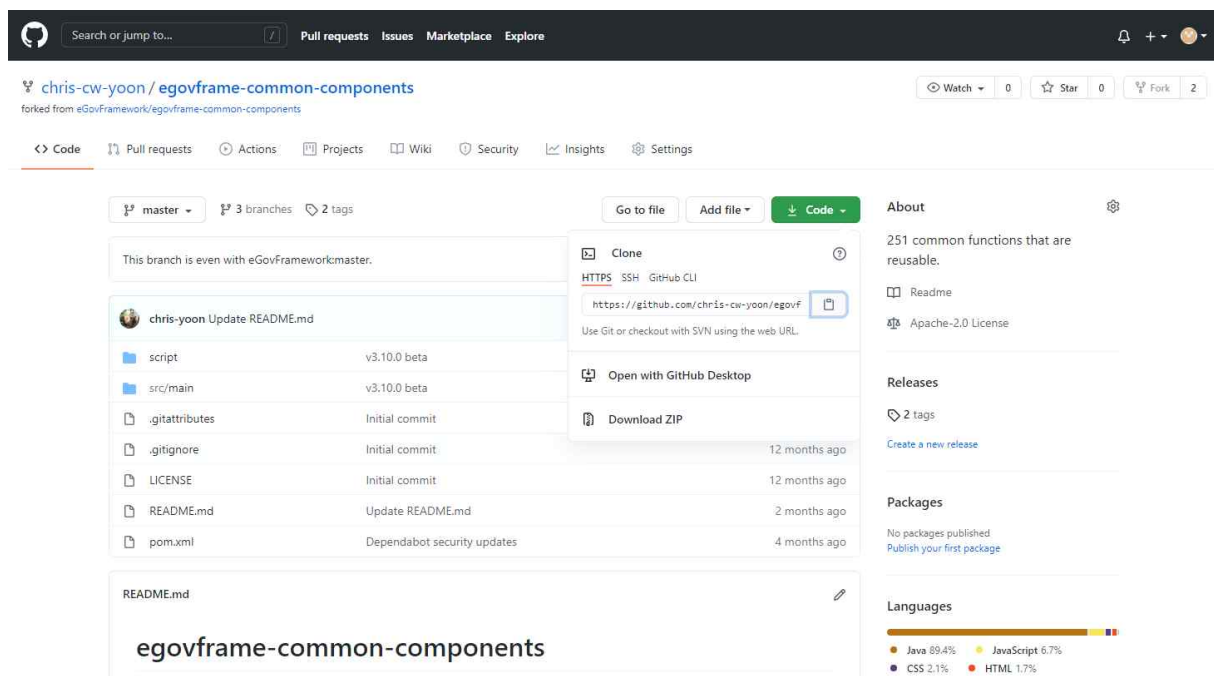


나의 Repository에 그대로 복사된 것을 확인 할 수 있다.




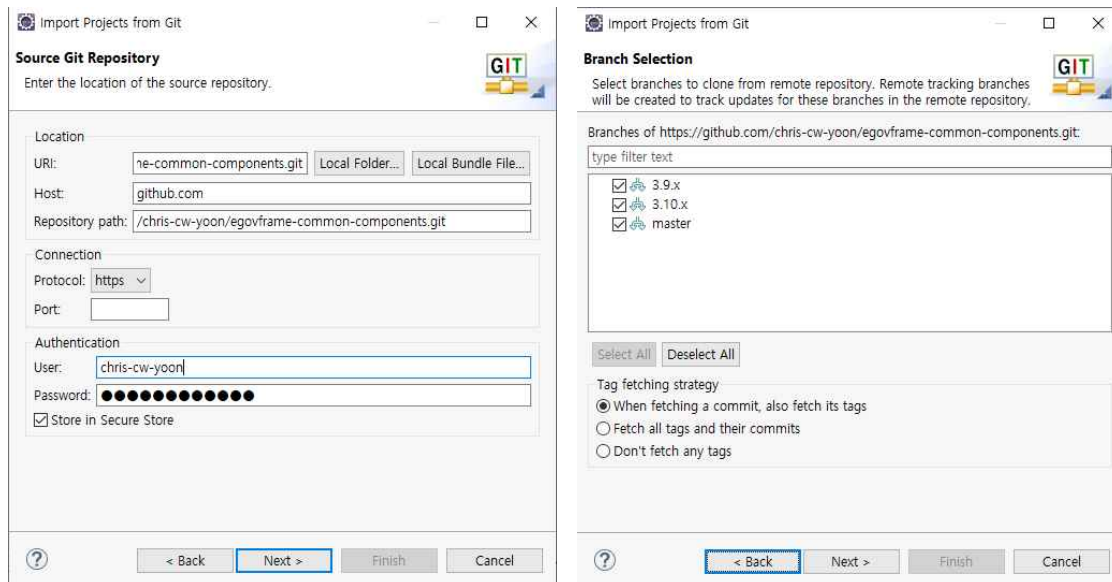
2. GitHub Clone

나의 Repository로 Fork된 프로젝트의 우측 상단 녹색 버튼 [Code]를 클릭해 보면, 다음과 같이 표시되는데 Clone URL 우측 아이콘을 클릭하여 Clone할 주소를 클립보드에 저장한다.

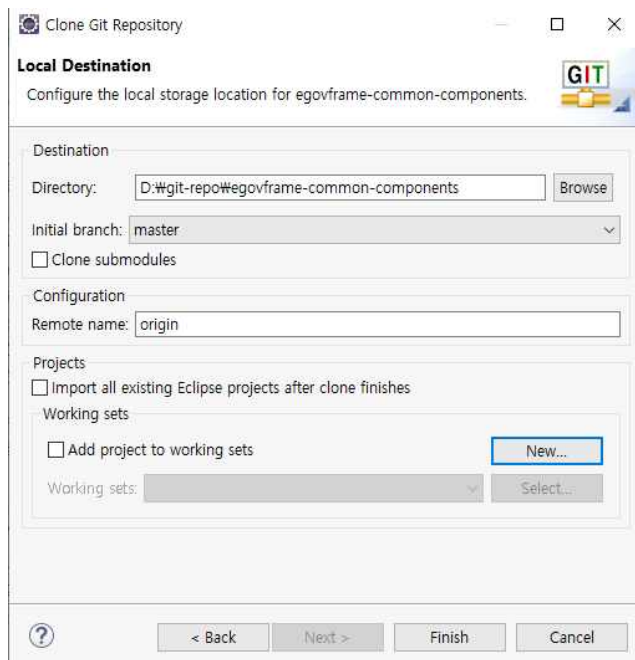


표준프레임워크 개발환경에서 Git Repositories View에서

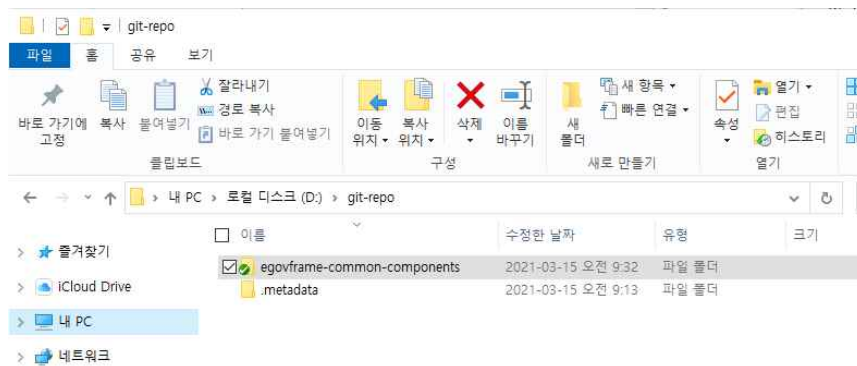
 [Clone a Git repository](#) 를 클릭하여 Source Git Repository 대화창의 URI 란에 복사했던 원격저장소의 URL을 붙여 넣으면, Host와 Repository path는 자동으로 입력된다. Authentication에서 User, Password 입력하고, Store in Secure Store를 선택한 후 [Next]를 클릭한다. Branch Selection에서 브랜치를 선택 후 [Next]를 클릭한다.



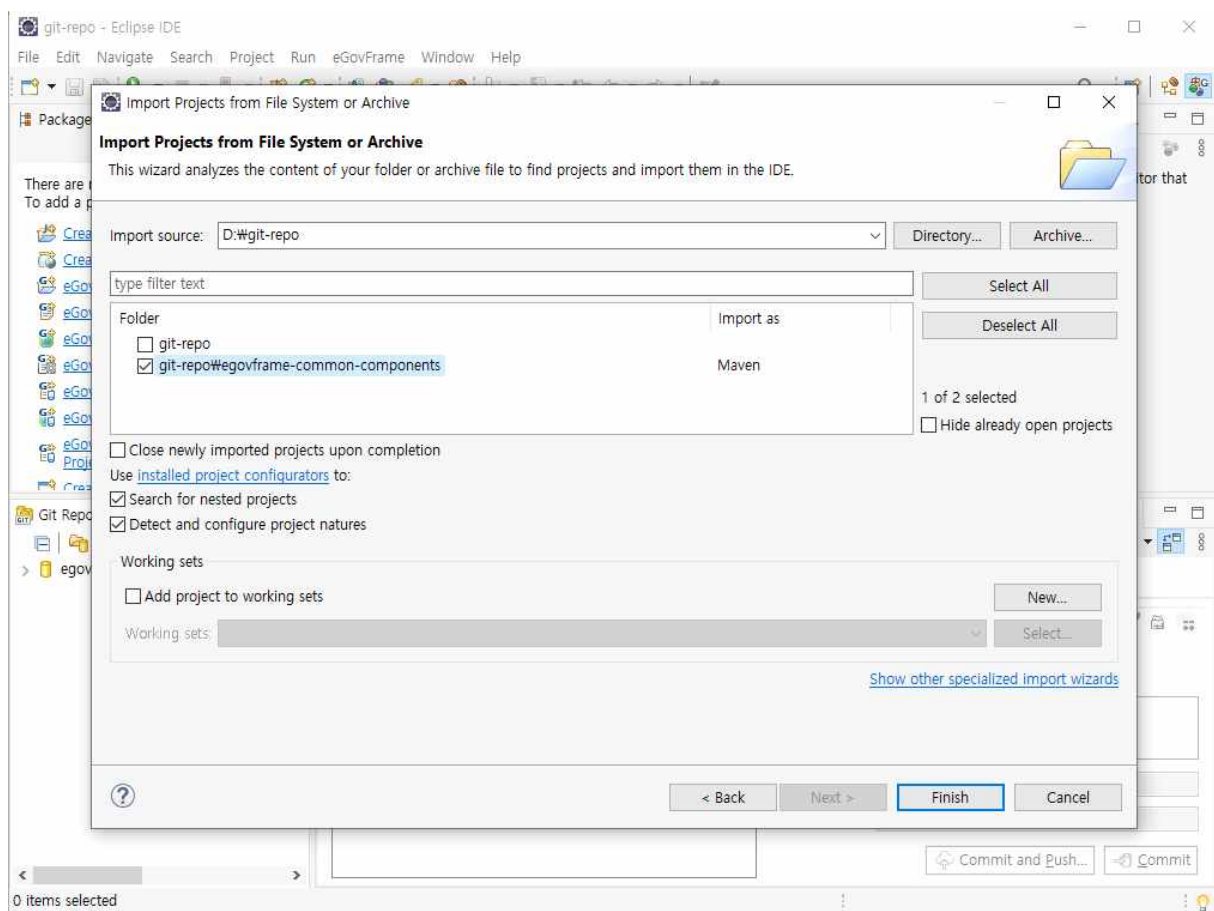
Local Destination 대화창에서 작업할 디렉토리를 선택 후 [Next]를 클릭한다.



다음과 같이 작업폴더에 복제된 것을 확인할 수 있다.




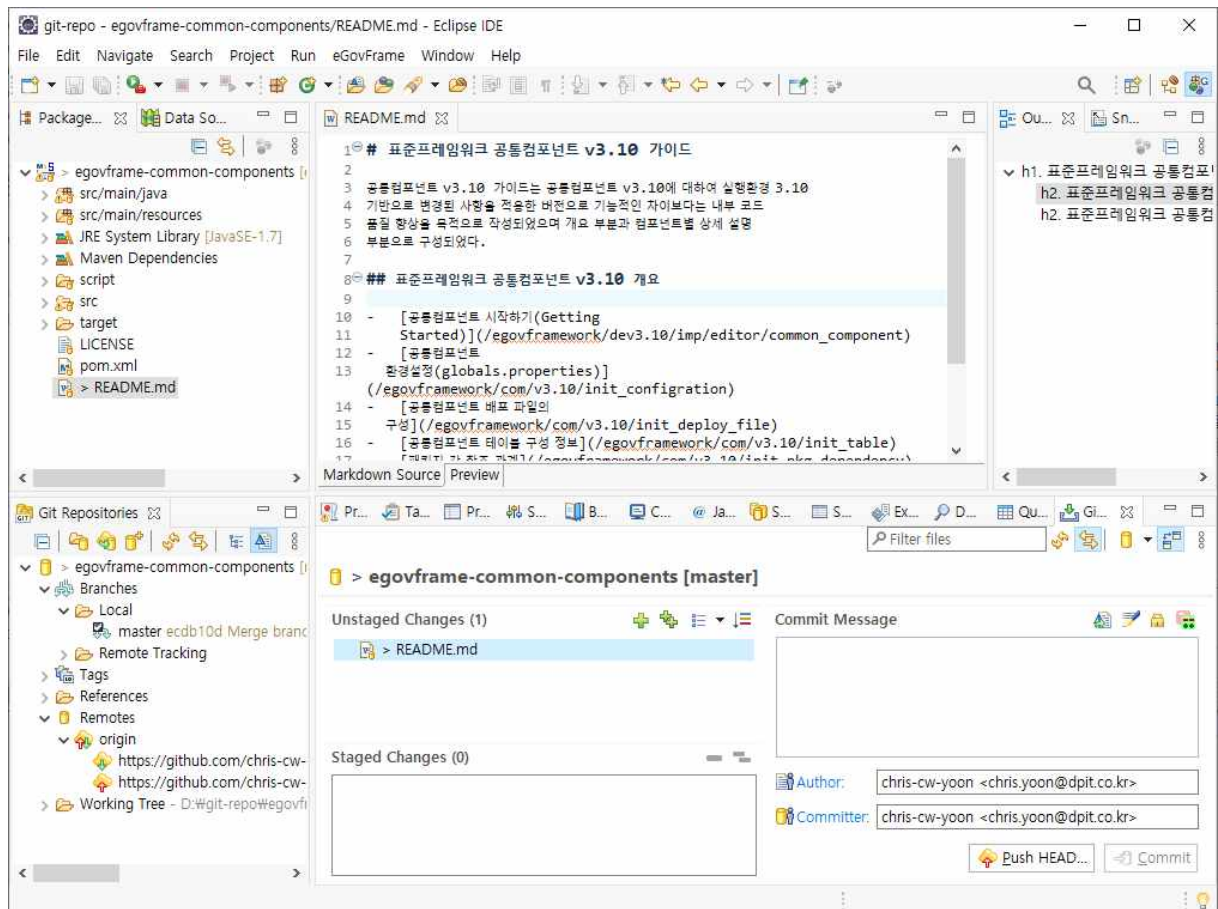
개발환경에서 File > Import... > General > Projects from Folder or Archive를 선택하여, 아래와 같은 대화창에서 복제된 프로젝트를 선택 후 [Finish]버튼을 클릭한다.




제3절 개발작업 후 내 원격저장소로 Push

1. 수정된 소스를 내 로컬저장소에 commit

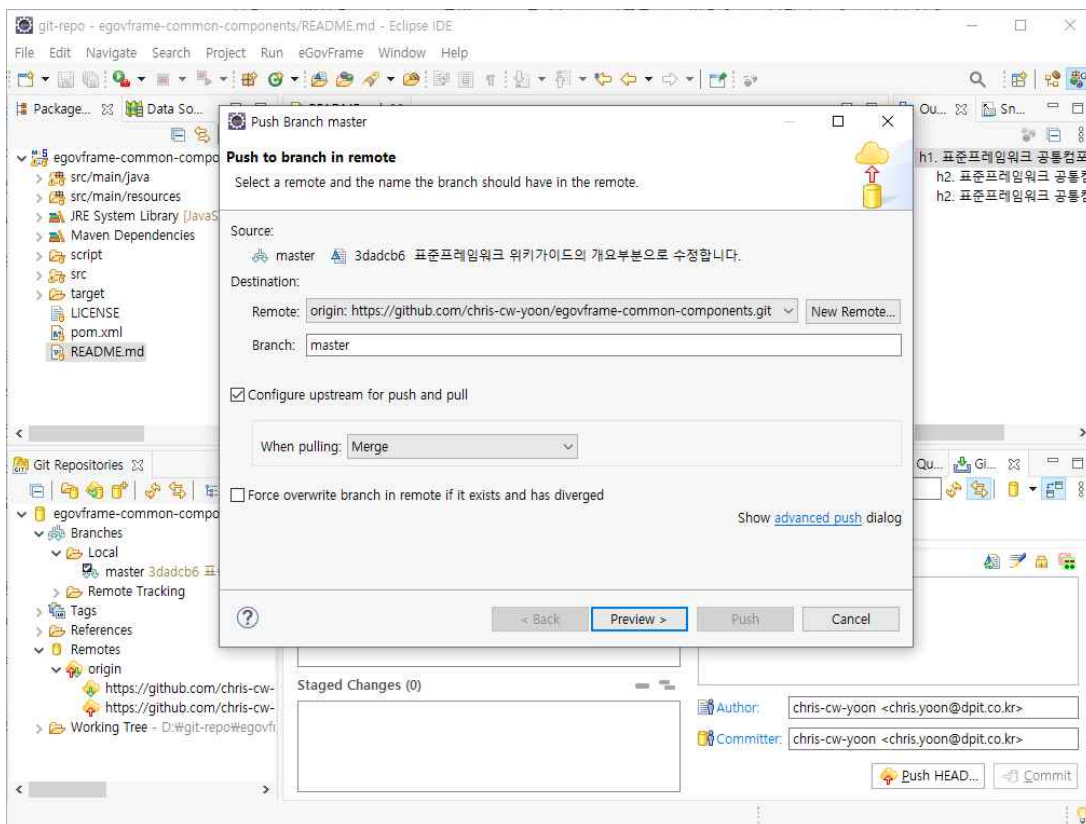
표준프레임워크 개발환경에서 소스를 수정하면, Package Explorer 의 수정된 파일명 좌측에  > README.md 처럼 ‘>’ 문자가 표시된다. Git Staging View의 Unstaged Changes에도 표시된다.



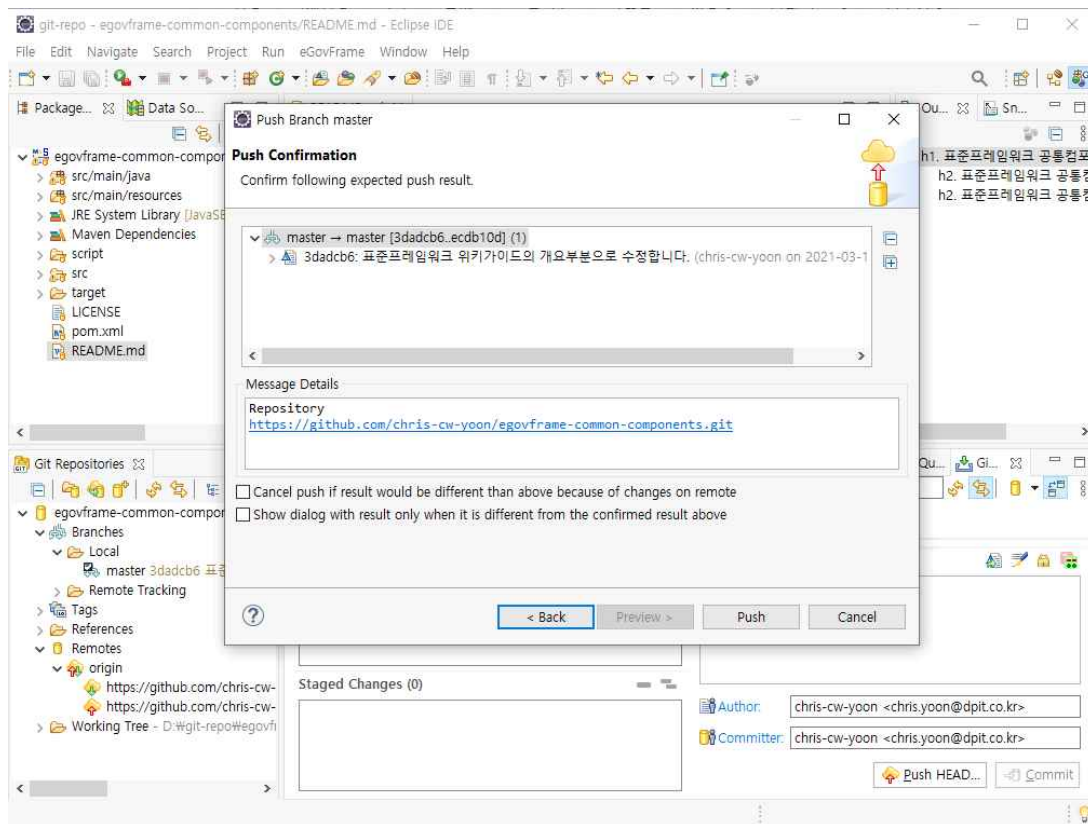
내 로컬저장소로 commit 하기 위해 Git Staging View의 Unstaged Changes 의 수정된 파일을 선택 후  클릭 또는 Staged Changes로 드래그 앤 드롭 하면, 다음과 같이 선택한 파일명 앞의 아이콘이 바뀌며, Staged Changes 에 추가된 것을 알 수 있다. Commit Message 란에 작업 내용을 작성 후 [Commit] 또는 [Commit and Push...] 버튼을 클릭한다. Commit 은 내 로컬저장소로 반영하는 것이고, Commit and Push는 내 로컬저장소 와 내 원격저장소에 동시에 반영하는 것이다. 내 로컬저장소로 Commit만 해 놓았다가, 내 원격저장소로 한번에 Push할 수도 있다.

2. 수정된 소스를 내 원격저장소로 Push

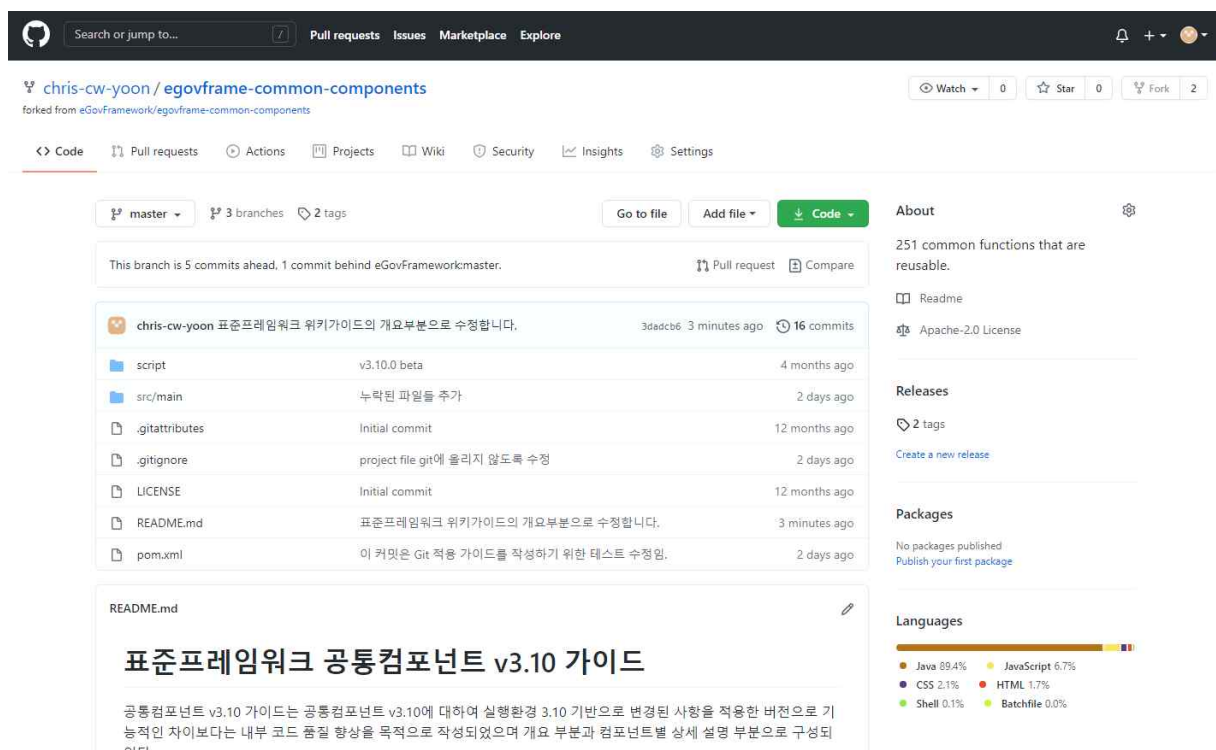
수정된 소스를 로컬저장소로 Commit 한 후 바로 내 원격저장소로 Push 하려면 Commit Message를 입력한 후 [Commit and Push...] 버튼을 클릭한다. 다음과 같이 원격저장소를 확인 하는 대화창이 표시되며, [Preview >]버튼을 클릭한다.



Push Confirmation 창에서 내용을 한번 더 확인하고, [Push]버튼을 클릭한다.

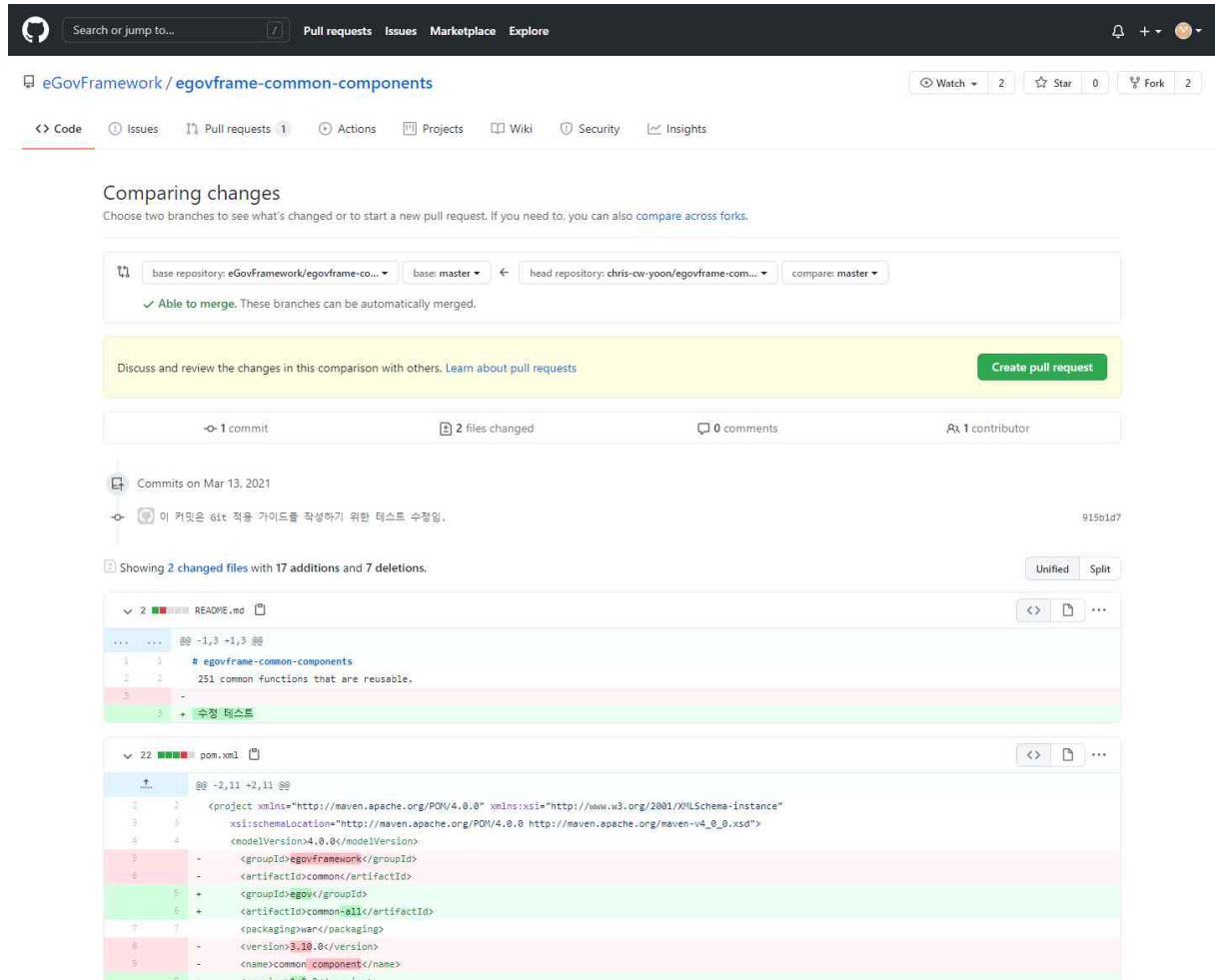


GitHub 사이트의 내 원격저장소에 commit 한 내용을 확인할 수 있다.

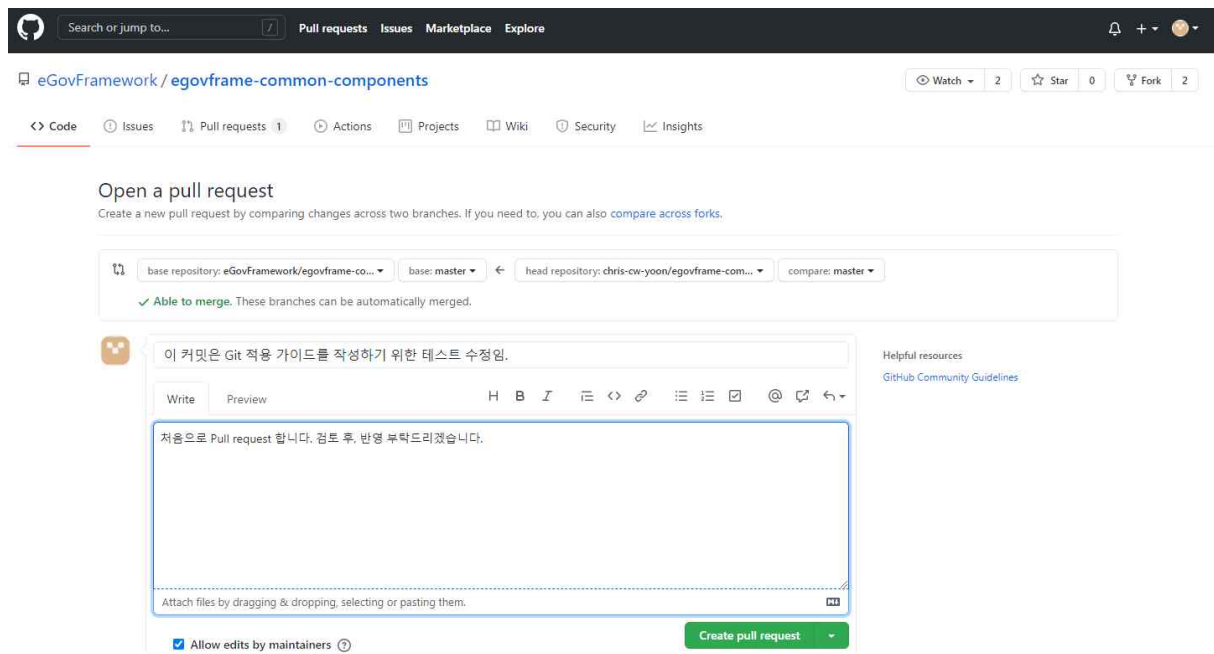


3. 원본 원격저장소에 반영 요청 (Pull Request)

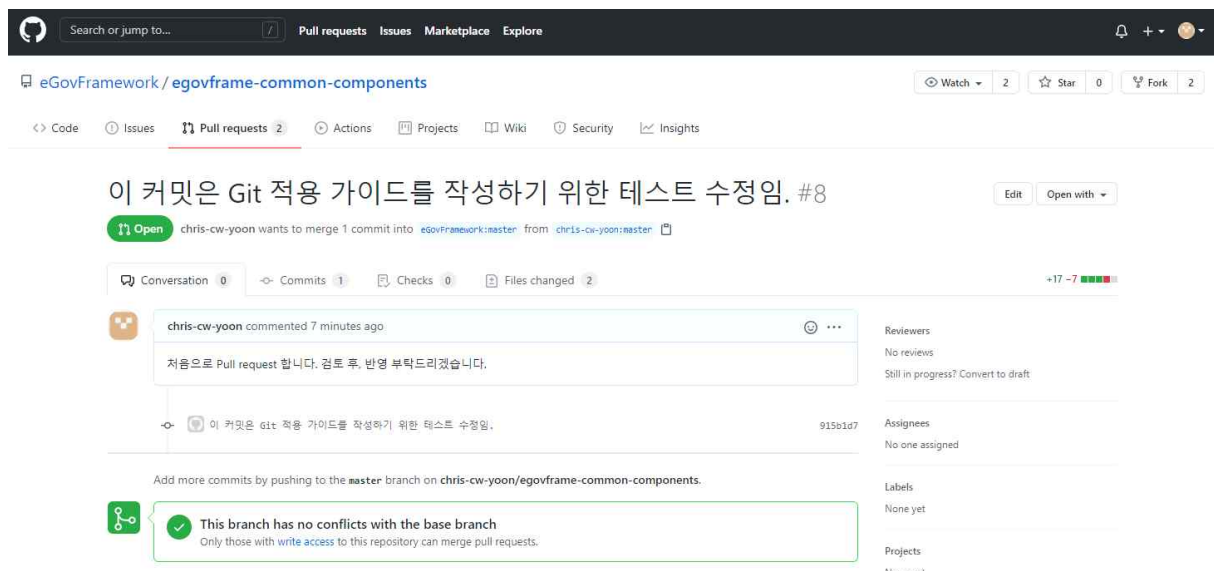
표준프레임워크 원격저장소(Upstream)에 나의 수정사항을 반영 요청하기 위해, 내 원격저장소의 **Pull request** 를 클릭한다. 다음과 같이 내가 수정한 내용을 확인 후 [Create pull request] 녹색 버튼을 클릭한다.



Pull request 요청 내용을 작성 후 [Create pull request] 버튼을 클릭한다.



표준프레임워크 원격저장소로 나의 Pull request가 요청되었음을 확인 할 수 있다.



제4절 Pull request 이후 검토내용 확인

1. 검토내용 확인

표준프레임워크 센터에서 작성한 검토내용은 나의 받은 이메일함에서 확인 할 수 있다.

Q 검색어를 입력해주세요. 상세 ▾ 받은메일함 0 / 4738 안읽은 메일 모두 읽음 표시

답장 전체답장 전달 삭제 스팸신고 안읽음 이동 ▾ 리마인드 ▾ 더보기 ▾

☆ Re: [eGovFramework/egovframe-common-components] 이 커밋은 Git 적용 가이드를 작성하기 위한 테스트 수정임. (#8)

2021. 3. 13. (토) 13:22

보낸사람 eGovFrameSupport<notifications@github.com> 메시지 약속초대

받는사람 eGovFramework/egovframe-common-components<egovframe-common-components@noreply.github.com>

참조 chris-cw-yoon<chris.yoon@dpit.co.kr>, Author<author@noreply.github.com>

@eGovFrameSupport approved this pull request.
바로 반영해도 될 듯 합니다.

—

You are receiving this because you authored the thread.
Reply to this email directly, [view it on GitHub](#), or [unsubscribe](#).

2. 원본 원격저장소 브랜치에 병합 확인

표준프레임워크 원격저장소 브랜치에 병합되면, 다음과 같이 수신된 이메일로 확인할 수 있다.

Q 검색어를 입력해주세요. 상세 ▾ 받은메일함 0 / 4738 안읽은 메일 모두 읽음 표시

답장 전체답장 전달 삭제 스팸신고 안읽음 이동 ▾ 리마인드 ▾ 더보기 ▾

☆ Re: [eGovFramework/egovframe-common-components] 이 커밋은 Git 적용 가이드를 작성하기 위한 테스트 수정임. (#8)

2021. 3. 13. (토) 13:22

보낸사람 eGovFrameSupport<notifications@github.com> 메시지 약속초대

받는사람 eGovFramework/egovframe-common-components<egovframe-common-components@noreply.github.com>

참조 chris-cw-yoon<chris.yoon@dpit.co.kr>, Author<author@noreply.github.com>

Merged #8 into master.

—

You are receiving this because you authored the thread.
Reply to this email directly, [view it on GitHub](#), or [unsubscribe](#).

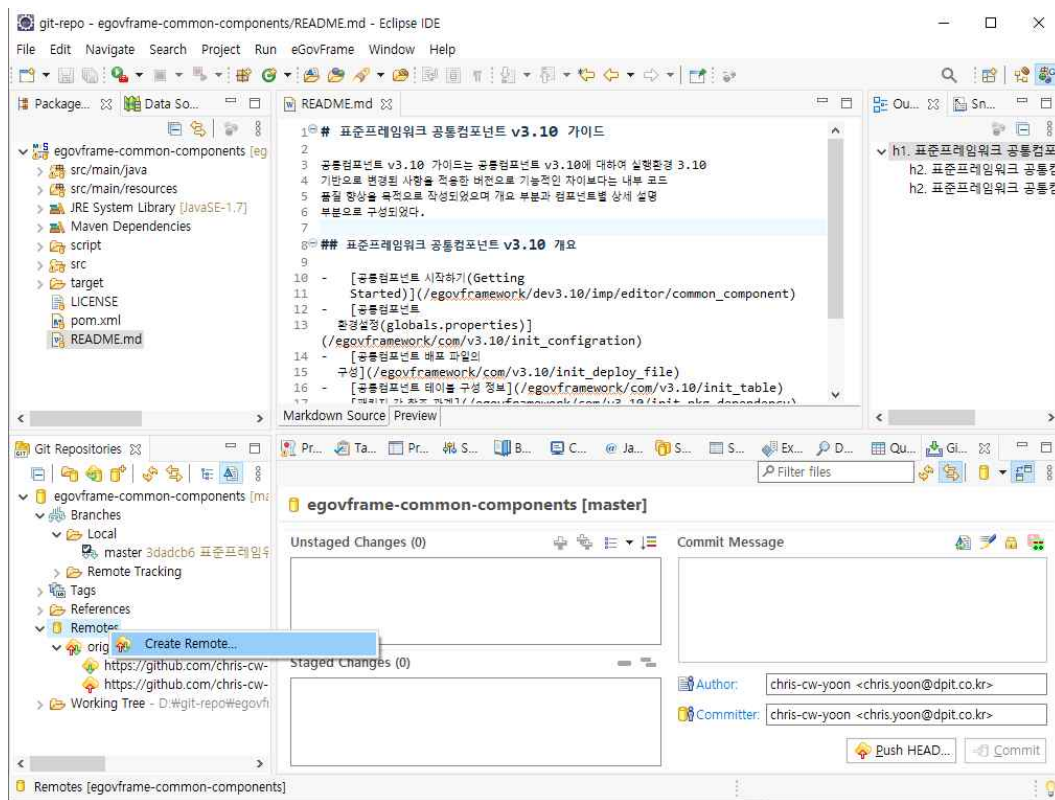
제5절 표준프레임워크 원격저장소와 최신버전 동기화

시간이 흐르면서 원본 원격저장소는 여러 개발자들로부터 Pull Request를 받아 브랜치에 병합하게 되면, 내 로컬저장소의 소스들과 달라지기 때문에 원본 원격저장소의 최신 commit을 내 로컬저장소에 반영해야 한다.

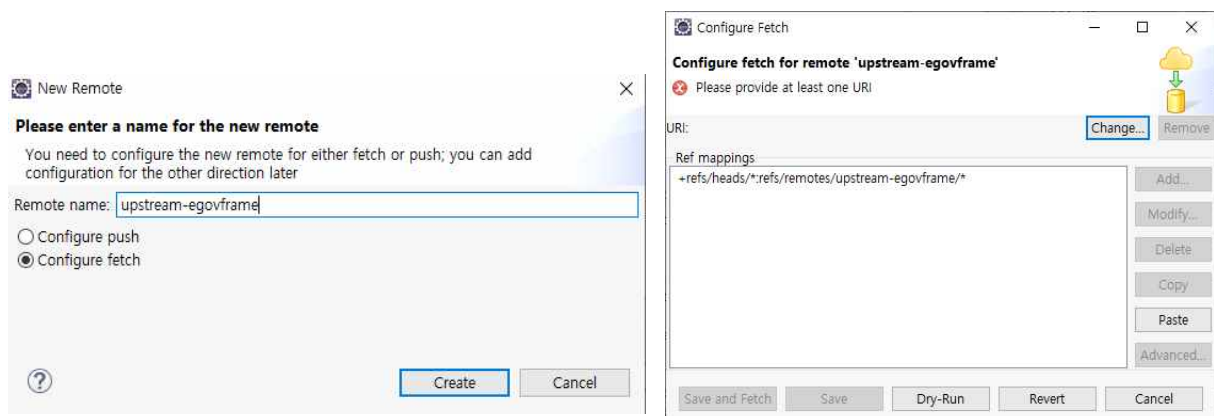
1. 개발환경에 표준프레임워크 원본 원격저장소 추가

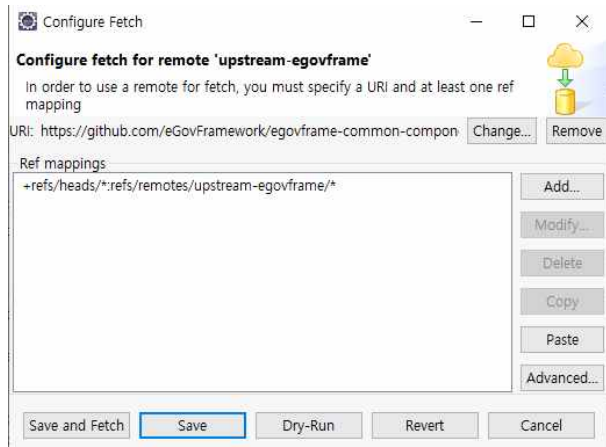
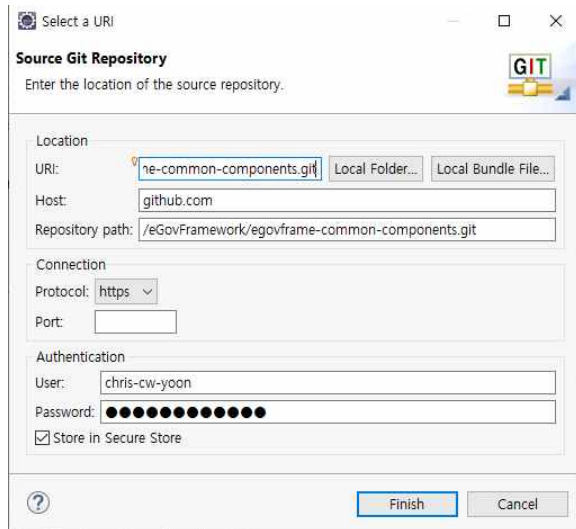
개발환경에 표준프레임워크 원본 원격저장소를 추가하기 위해 Git Repositories 창에서 Remote를 선택하고, 마우스 우클릭

하여 'Create Remote...' 선택한다.



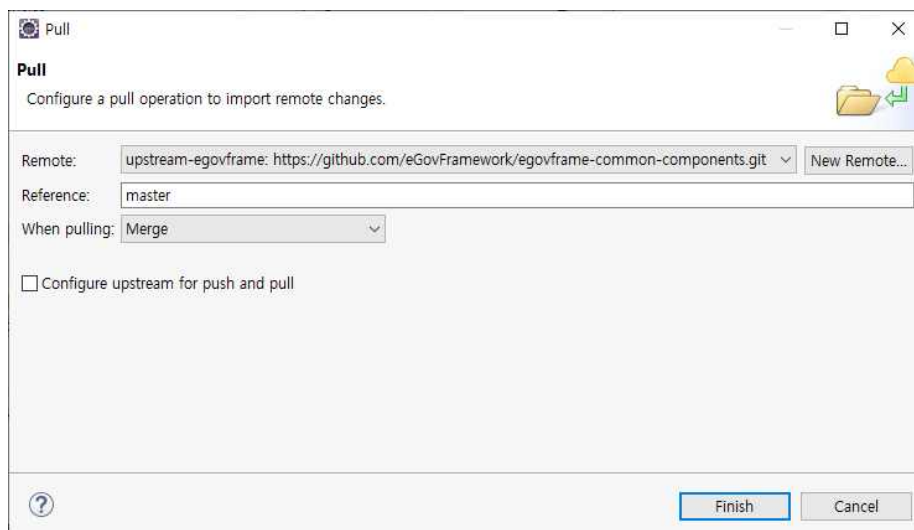
Remote name을 입력, Configure fetch 선택 후 [Create] 버튼을 클릭한다. Configure Fetch 다이얼로그창에서 [Change...] 클릭 후 표준프레임워크 원본 원격저장소 URL을 입력하고 [Finish]를 클릭한다. Configure Fetch 창에서 [Save]를 클릭한다.

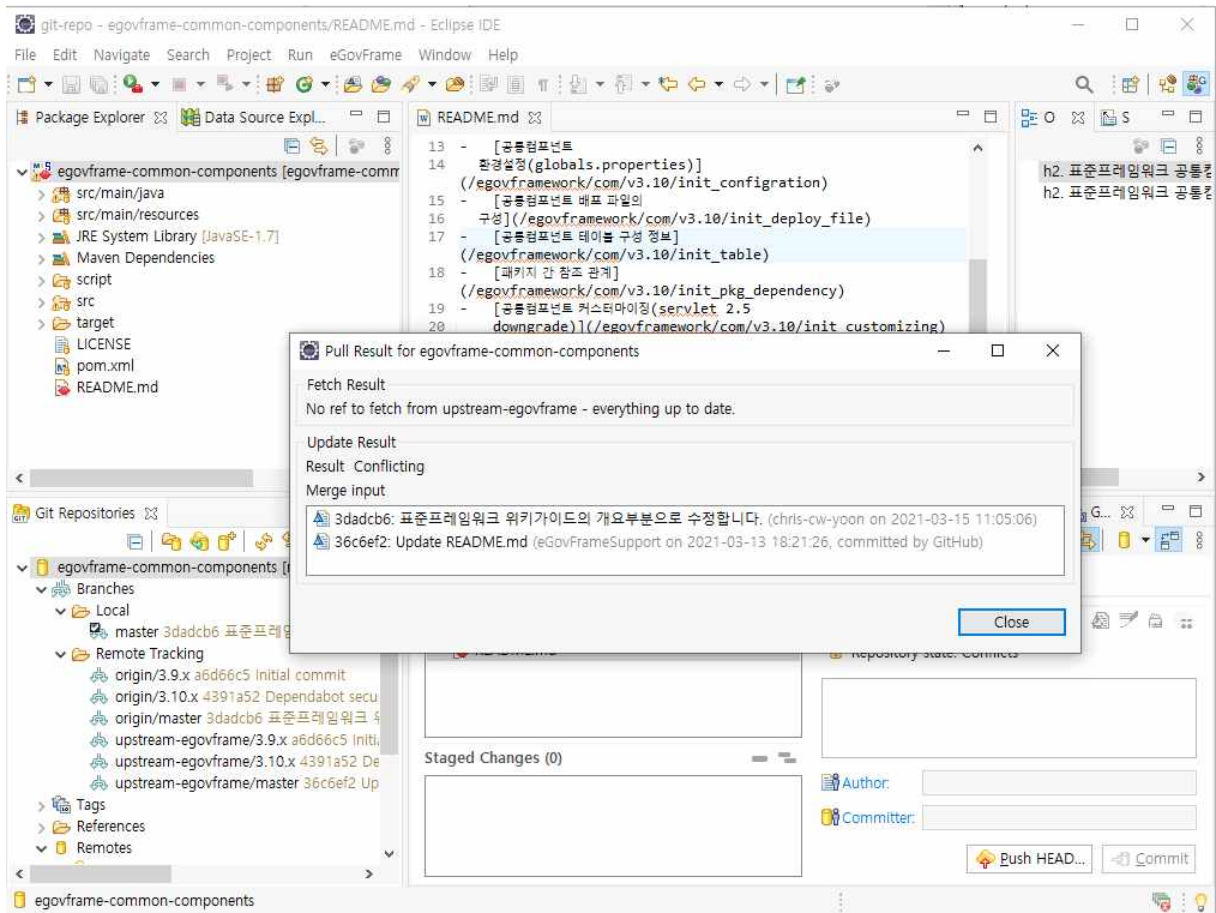





2. 표준프레임워크 원격저장소로부터 최신 commit Pull

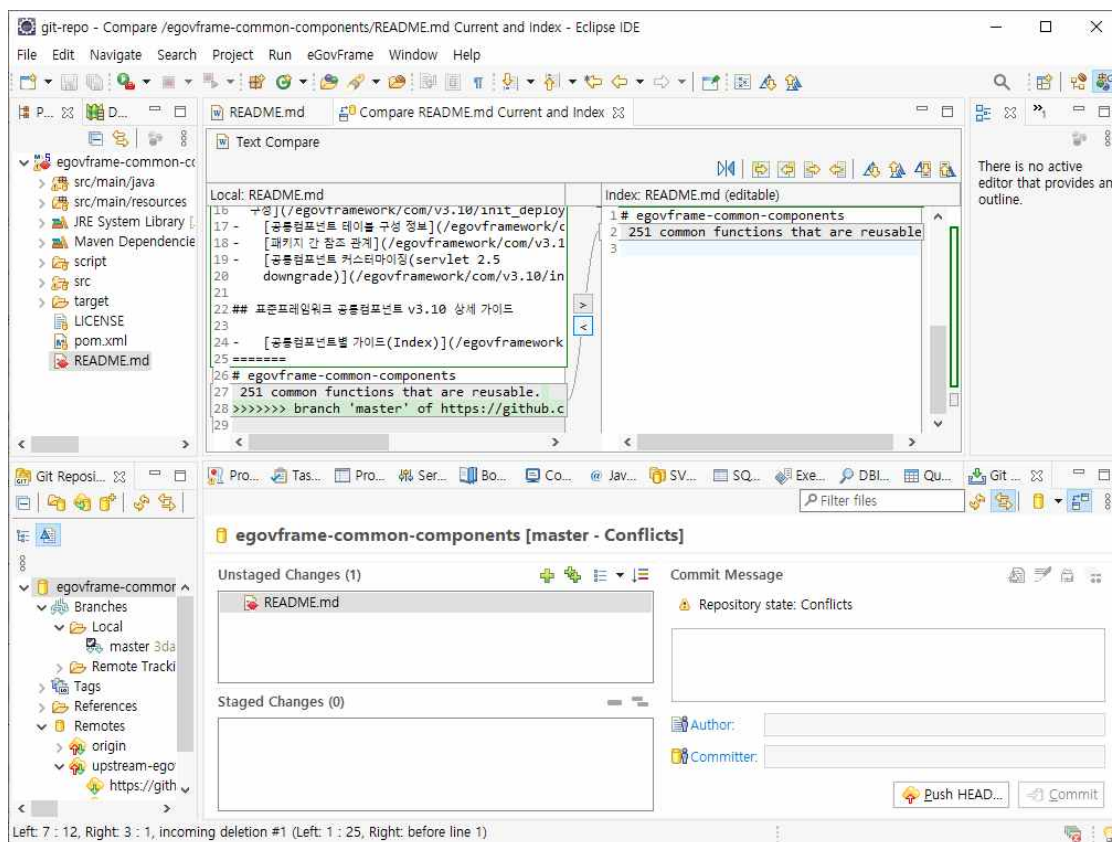
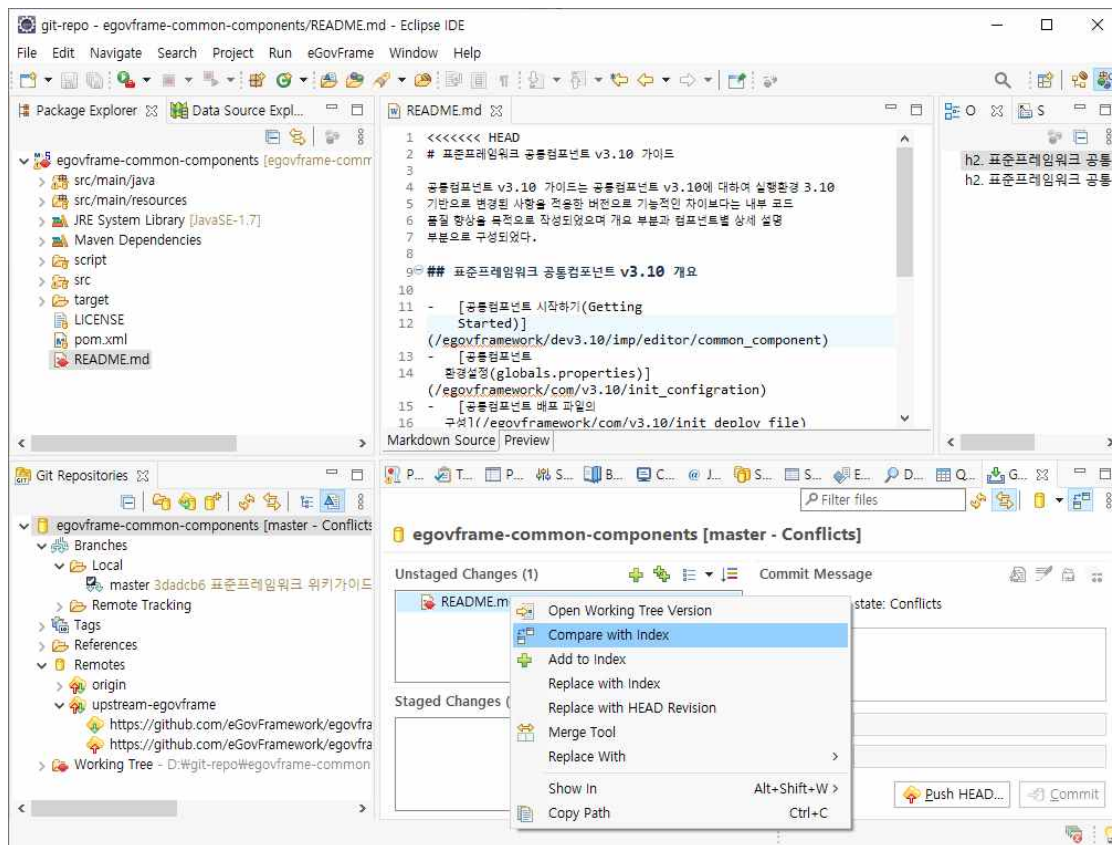
표준프레임워크 원격저장소로부터 최신 commit를 가져오기 위해서 Package Explorer에서 프로젝트를 선택 후 마우스 우클릭하여 Context 메뉴에서 Team > Pull... 메뉴를 선택한다. Remote를 표준프레임워크 원본 원격저장소로 선택한 후 [Finish]를 클릭한다.





3. Conflict 해결

표준프레임워크 원격저장소로부터 최신 commit을 가져오기 (Pull) 한 후 Conflict가 발생하는 경우엔  README.md 처럼 빨간색 아이콘이 표시된다. Git Staging View에서 Conflict 발생 파일을 선택 후 마우스 우클릭, Compare with index 메뉴를 선택한다. Staged 파일 내용과 로컬 파일을 비교할 수 있도록 에디터 영역이 구성되며, 이를 참조하여 로컬 파일을 수정 및 테스트를 거쳐 가며, Conflict를 해결한다.



4. 내 원격저장소로 Push

내 로컬저장소는 표준프레임워크 원격저장소와 동기화 되었지만, 내 원격저장소는 아직 반영이 되지 않은 상태이기 때문에 Push 하게 되면, 원본 원격저장소와 Forked 된 내 원격저장소와 일치 된다. 이후 개발 작업도 제 3절, 4절을 반복하면 된다.