

Git 사용 기본과

Redmine을 활용한 이슈 관리

Synetics

한동준 대표

dongjoon.han@synetics.kr
handongjoon@gmail.com

목표

◆ 목표

- Git
 - 소스코드 버전 관리 개념과 필요성을 이해한다.
 - Git의 버전 관리 방법을 이해한다.
 - Git의 기본 기능을 활용할 수 있다.
- Redmine
 - 이슈 관리의 필요성을 이해한다.
 - 이슈 관리와 버전 관리 도구의 연동 필요성을 이해한다.
 - 버전을 활용할 수 있다.

01 DevOps (요즘 핫한 개발 절차)

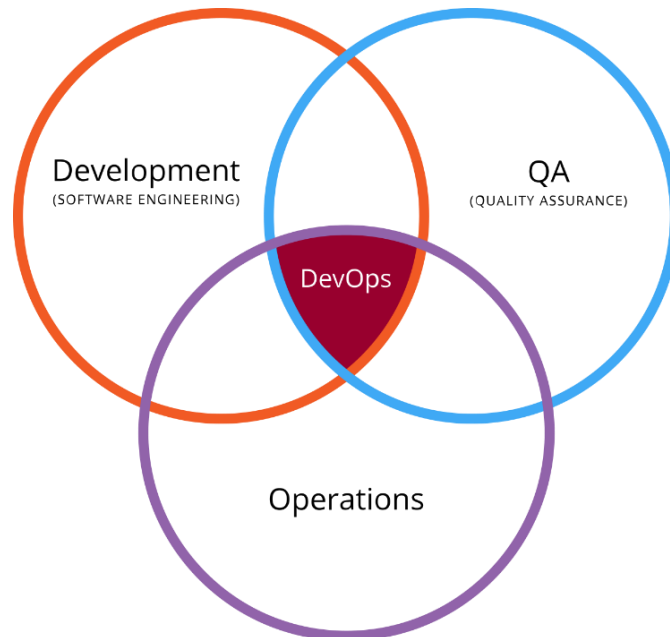
DevOps란

◆ 개요

- 소프트웨어의 개발(Development)과 운영(Operations)의 합성어
- 소프트웨어 개발자와 정보기술 전문가 간의 소통, 협업 및 통합을 강조하는 개발 환경이나 문화

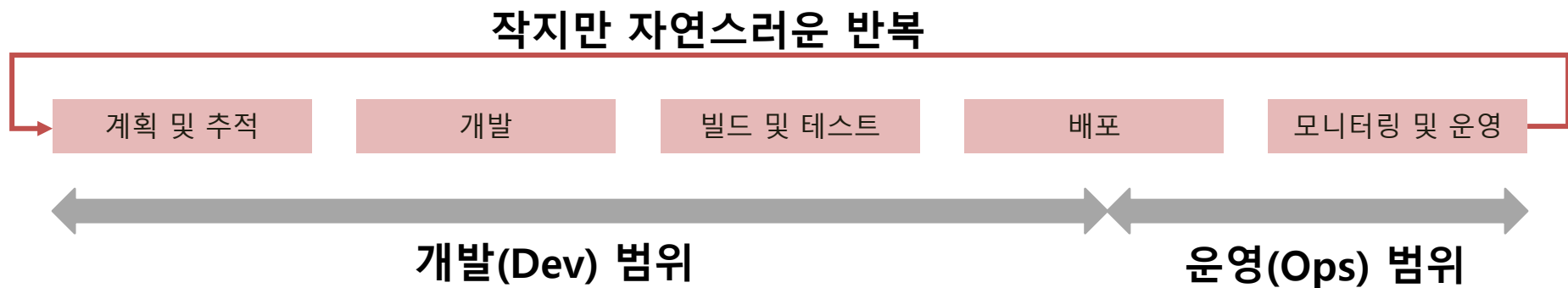
◆ 목적

- 제품 출시까지 걸리는 기간(time to market) 단축
- 새로운 판의 더 낮은 실패율
- 픽스 간 짧아진 리드 타임(상품 생산 시작부터 완성까지 걸리는 시간)
- 복구 시 더 빠른 평균 시간 (새로운 릴리스의 충돌 및 시스템을 비활성화하는 상황에서)



출처: 위키피디아

DevOps 파이프라인



DevOps를 운영하기 위해 Dev / Ops 모든 단계의 자동화가 필수

자동화

자동화에 투자하면 반복적인 수동 업무를 없애고, 반복 가능한 프로세스와 안정적인 시스템을 만들 수 있습니다.

자동화 빌드, 테스트, 배포 및 프로비저닝은 이러한 단계를 갖추고 있지 않은 팀이 일반적으로 시작해야 하는 과정입니다. 모두에게 도움이 되는 시스템을 구축하는 것보다 개발자, 테스터 및 운영자가 협력해야 하는 것이 더 중요한 이유는 무엇입니까?

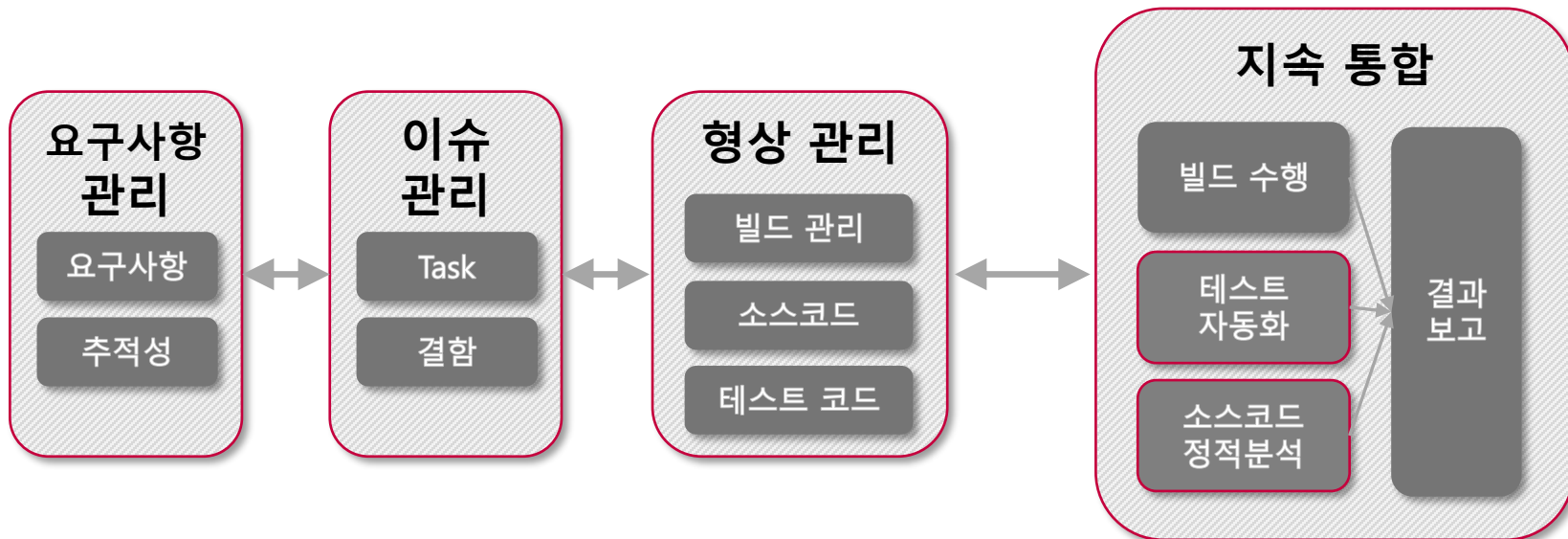
자동화를 처음 접하는 팀은 보통 지속적 배포로 시작합니다. 지속적 배포란 대부분 클라우드 기반 인프라로 촉진된 자동화 테스트를 통해 각 코드 변경 사항을 실행한 다음, 성공적인 빌드를 패키징하고, 자동화 배포를 사용하여 생산을 추진하는 과정입니다. 짐작할 수 있듯이 지속적 배포는 쉽고 빠르게 만들어낼 수는 없지만 ROI는 충분한 가치가 있습니다.

출처: Atlassian

Dev, 즉 개발과 테스트 중심의 자동화를 ALM 이라고도 함

ALM 구성

- ◆ ALM은 수행 절차를 정의한 프로세스와 이를 지원하는 도구로 구성되어야 함



요구사항 관리 프로세스

프로젝트 관리 프로세스

프로젝트 계획 프로세스

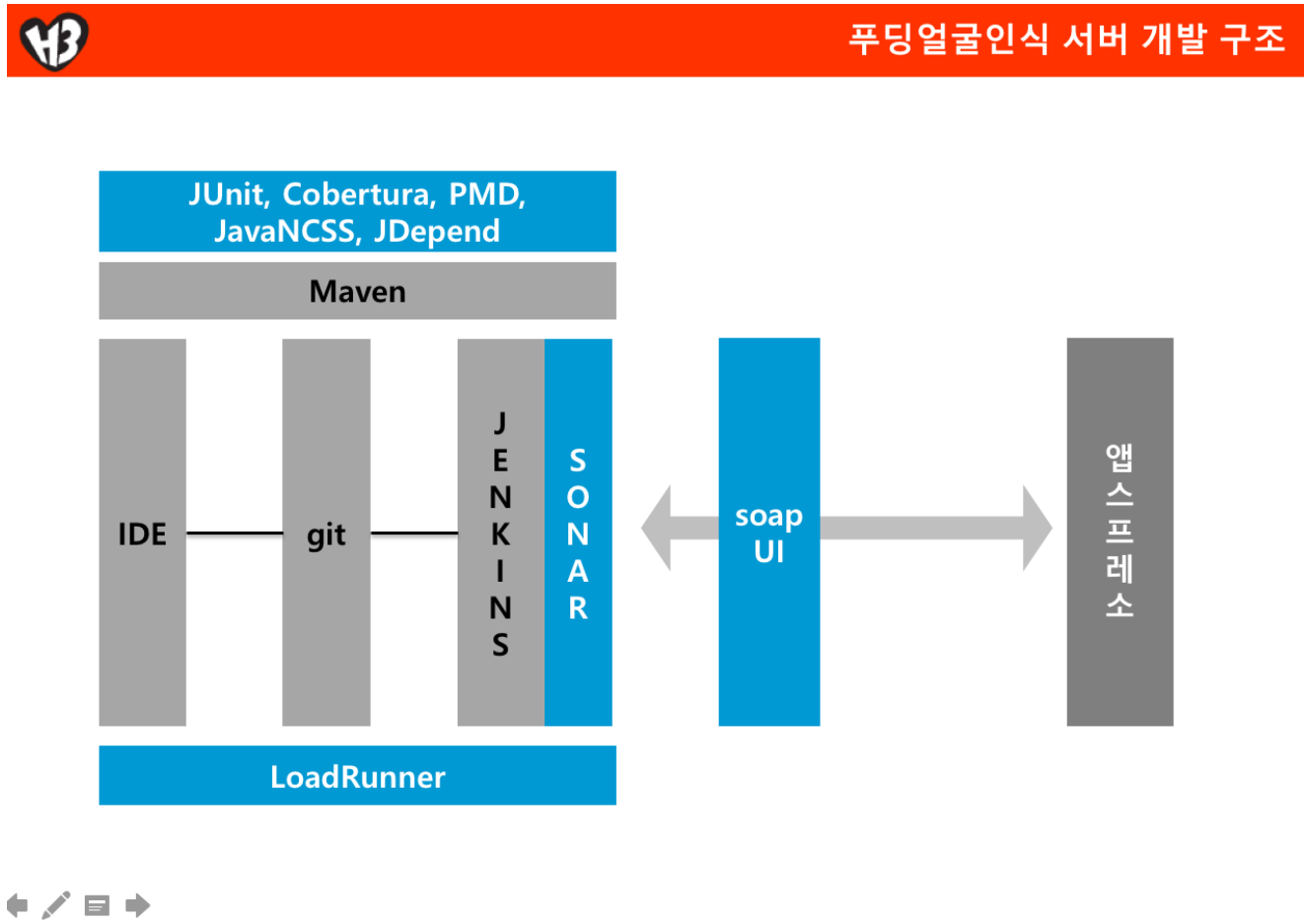
형상 관리 프로세스

SW 개발 프로세스

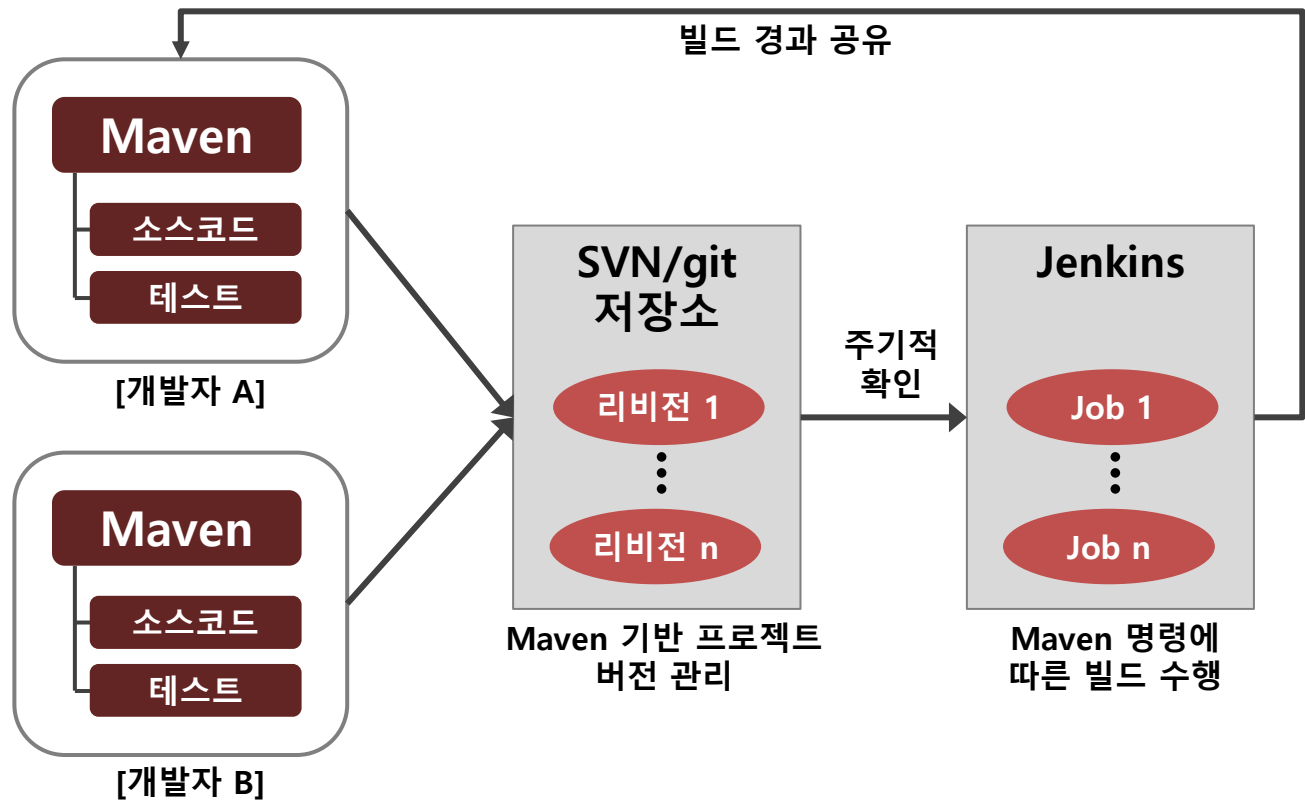
품질 관리 프로세스

이제는 많이 하는 소프트웨어 개발 활동

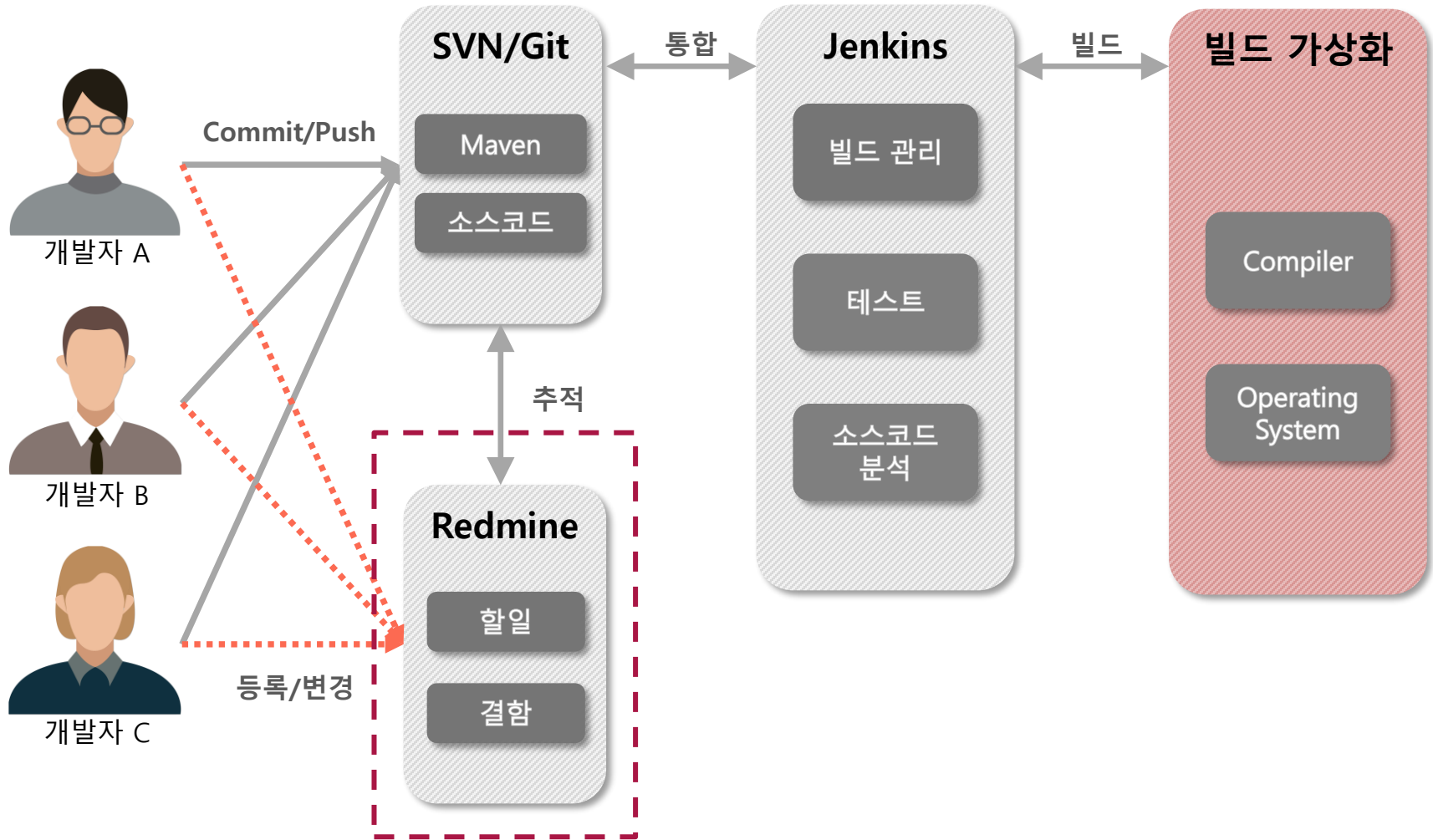
2012 H3 컨퍼런스 발표 – 이미 ALM 사용사례 발표



일반적인 오픈소스 기반 ALM 구성(Java 개발이라면)



보다 확대한 구성



02 소스코드 버전 관리 - Git과 Github

Git 소개

◆ 도구 개요

- 버전 관리의 대표 도구

◆ 목적 : 버전 관리와 협업 커뮤니케이션

- 소스코드 버전 관리
 - 변경 이력을 확인
 - 마음의 평화: 실수해도 이전 버전으로 빠르게 되돌리기
- 동료에게 변경 내역 공유하기

◆ 일반적인 사용 방법

- 별도의 저장소 운영 서버 사용
 - Git: GitHub, Bitbucket, Gitlab, Gblit
 - 통합 저장소: SCMManager
- 클라이언트는 Core나 별도의 확장 도구 사용
 - Eclipse/Intelli J 계열/VS 계열: 기본 포함
 - Tortoise 시리즈
 - Sourcetree

Git 개요

◆ 역사

- 2005년 리눅스 토발즈를 중심으로 개발
- 현재 GitHub에서 운영

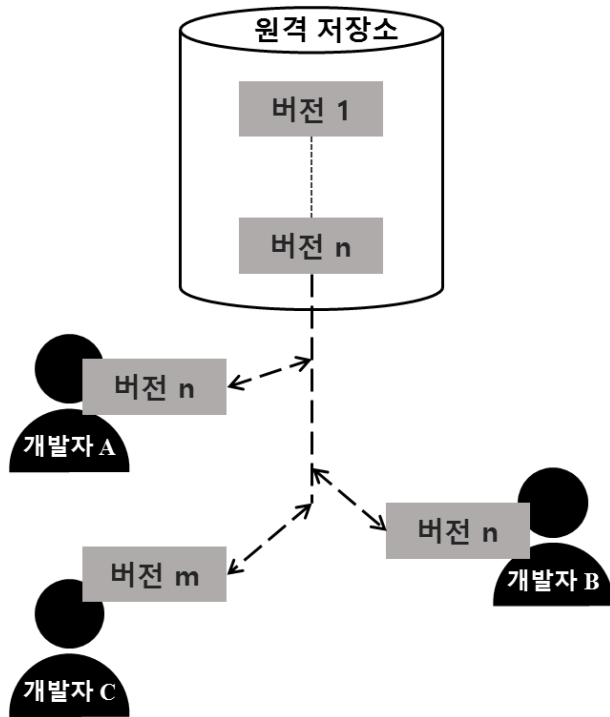
◆ 사용 추세

- 리눅스, Git을 포함하여 주요 오픈소스는 Git(GitHub) 사용
- 회사의 신규 프로젝트는 Git을 많이 사용하는 추세

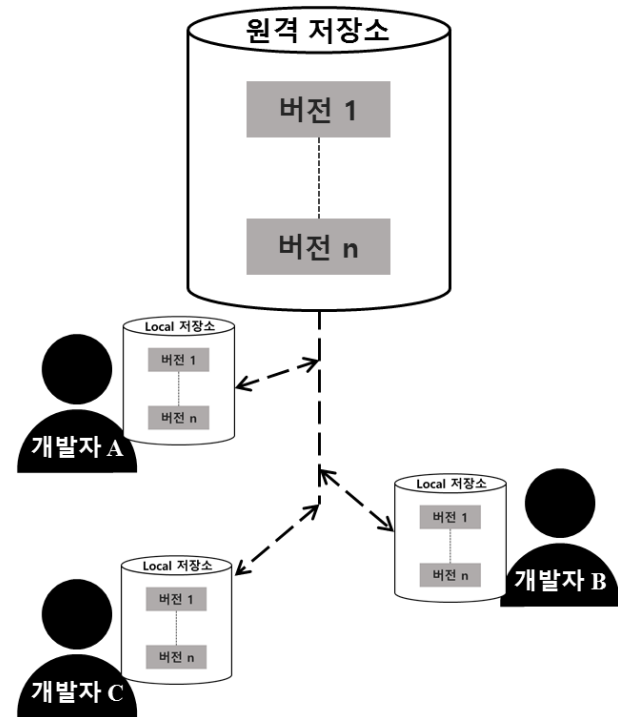
버전 관리 방식

◆ 분산 시스템 기반

- 원격 저장소와 동일한 로컬 저장소를 각 개발자가 유지
 - SVN은 최종 리비전만 각 개발자가 유지



Subversion



Git

기본 용어

영문	설명
Init	기존 폴더에 저장소를 생성
Clone	원격 저장소를 내 로컬에 복사
Push	내 저장소의 변경사항을 원격 저장소에 반영
Pull	원격 저장소의 변경사항을 내 저장소에 반영
Add	수정된 파일을 Staging Area에 등록하여 Staged 상태 만들
Check-out	작업할 브랜치를 지정
Commit	Staged 상태의 파일을 로컬 저장소에 반영
Remote Repository	공유를 위해 사용하는 저장소, 서버의 역할

Git을 사용합시다

◆ 모든 코딩 프로젝트에 Git 사용이 필수

- 이유: 소프트웨어 개발의 기본 (혼자/팀/여러 팀 모두)
- 소스코드는 문서가 아닙니다.
 - 계획서_최종_최종_최종_마지막_....docx

◆ Git을 사용하려면 팀원이 공유하기 위한 저장소(서버)가 필요함

- 회사에서 일반적인 방법 1: 내부 서버 설치, Git 서버용 설치...
- 회사에서 일반적인 방법 2: 외부 서비스 이용

◆ 이번 실습은 GitHub 이용

- MS가 약 7조원에 인수
- 최근 유명한 대부분의 오픈소스는 여기에서 운영

03 Git 기본 사용

Git 설치

◆ Git 다운로드

- <https://git-scm.com/>

◆ Git 설치

- 별도의 옵션 변경 없이 설치

◆ 정상 설치 확인

- 명령 프롬프트 실행 후, `git --version` 실행
- Git 버전 정보 확인되면 정상 설치 완료

C:\Windows\system32\cmd.exe

```
C:\Users\wdjhan>git --version  
git version 2.39.0.windows.2
```

```
C:\Users\wdjhan>
```



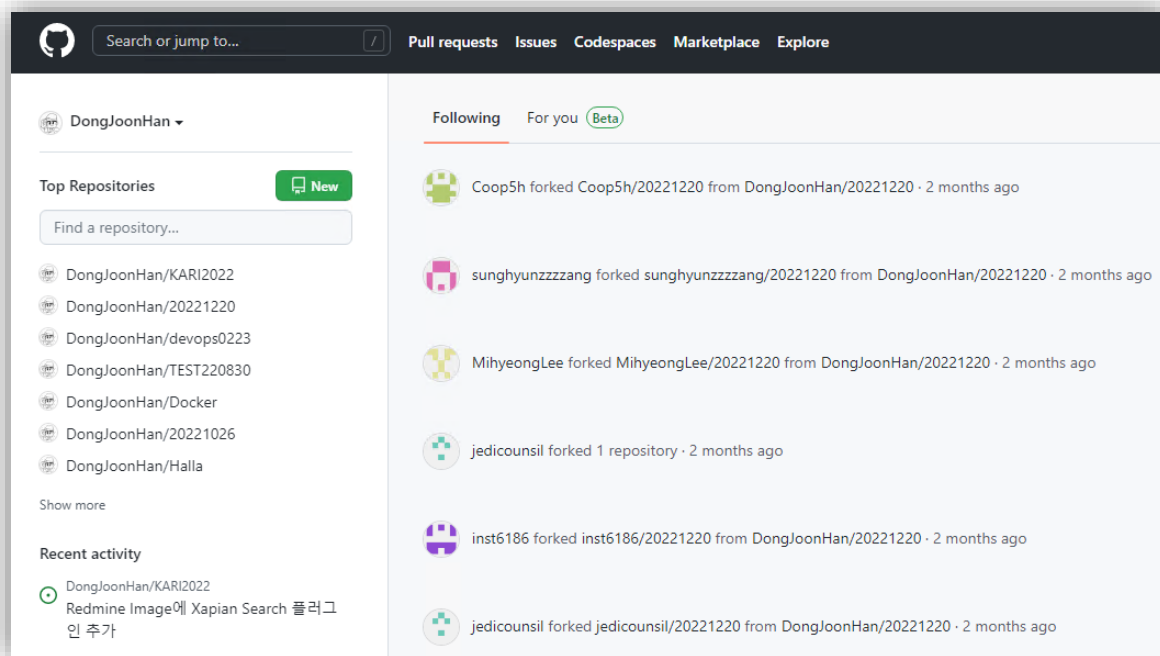
GitHub 회원 가입

◆ 목적

- GitHub에 실습용 저장소를 만들고 활용하기 위함

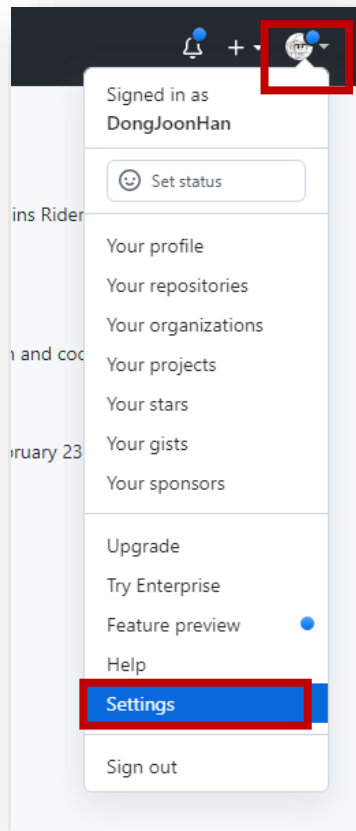
◆ 방법

- GitHub 가입 (가입 계정으로 개인 메일 권장)
- 인증 메일 클릭
- 정상 로그인 확인



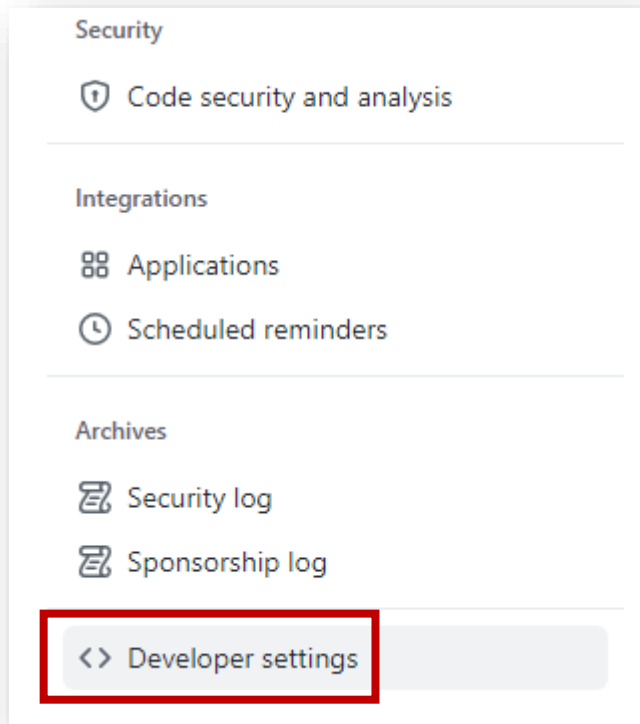
Github Personal Access Token 생성 - 1

- ◆ GitHub에서 Push 시 더 이상 ID/PW 방식을 지원하지 않음
 - Personal Access Token을 사용해서 인증해야 함
- ◆ 개인- Settings 이동



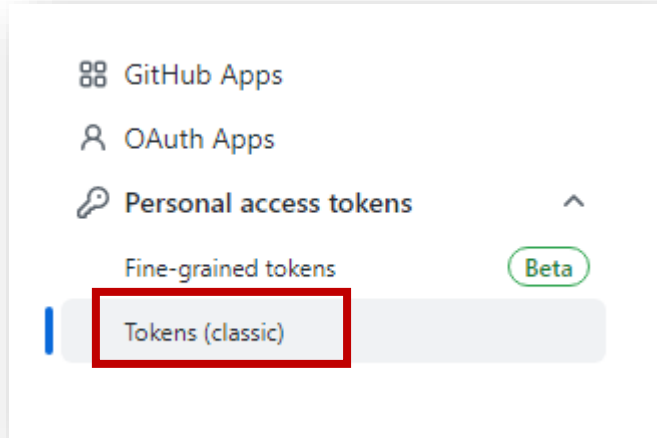
Github Personal Access Token 생성 - 2

- ◆ 제일 하단 Developer Settings 이동

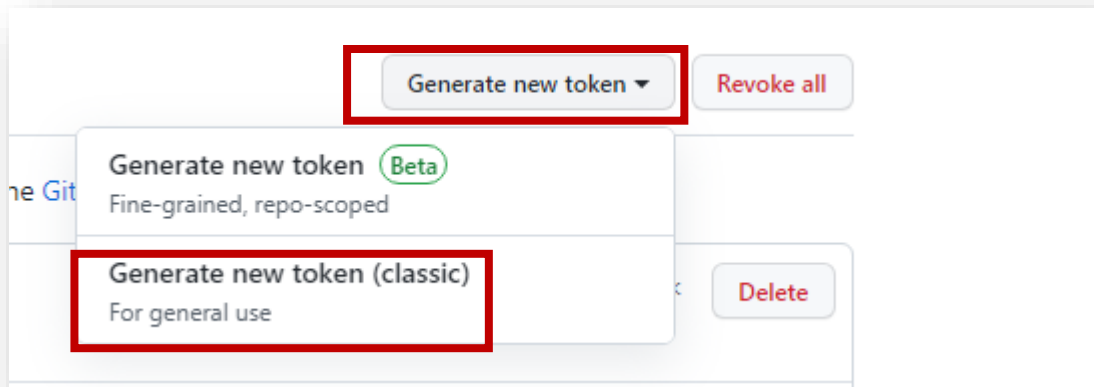


Github Personal Access Token 생성 - 3

◆ Personal access tokens – Tokens 이동



◆ Generate new token - Generate new token (classic) 이동



Github Personal Access Token 생성 - 4

◆ 기본 정보 입력

- Note: Token 사용 대상 등 정보 작성
- Expiration: Token 유효 기간
- Select Scopes: repo 선택

New personal access token (classic)

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

Note

LD15

What's this token for?

Expiration *

30 days

The token will expire on Tue, Mar 28 2023

Select scopes

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



repo

Full control of private repositories



repo:status

Access commit status



repo_deployment

Access deployment status



public_repo

Access public repositories



repo:invite

Access repository invitations



security_events

Read and write security events

Github Personal Access Token 생성 - 5

◆ 생성된 토큰 확인

Personal access tokens (classic)

Generate new token ▼ Revoke all

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your personal access token now. You won't be able to see it again!

✓ ghp_TTc

비밀내용

Delete

Github 저장소 생성

◆ 메인화면 우상단 + -> New repository 선택


◆ 저장소 정보 입력

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *

Repository name *


 DongJoonHan ▾

 /


basic ✓

Great repository names are short and memorable. Need inspiration? How about [fantastic-octo-funicular?](#)

Description (optional)

☐  Public

Anyone on the internet can see this repository. You choose who can commit.

☒  Private

You choose who can see and commit to this repository.

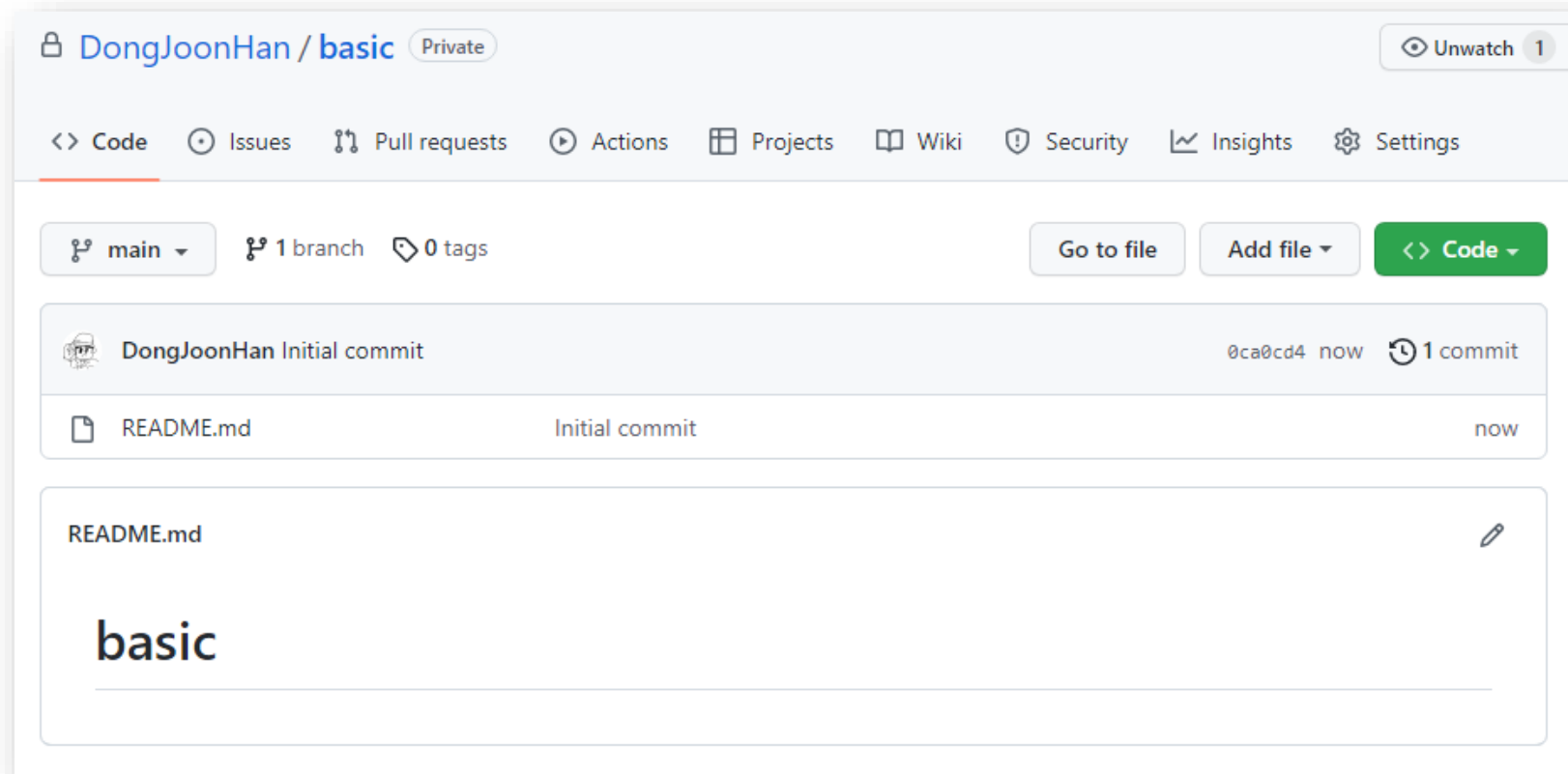
Initialize this repository with:

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

☒ Add a README file

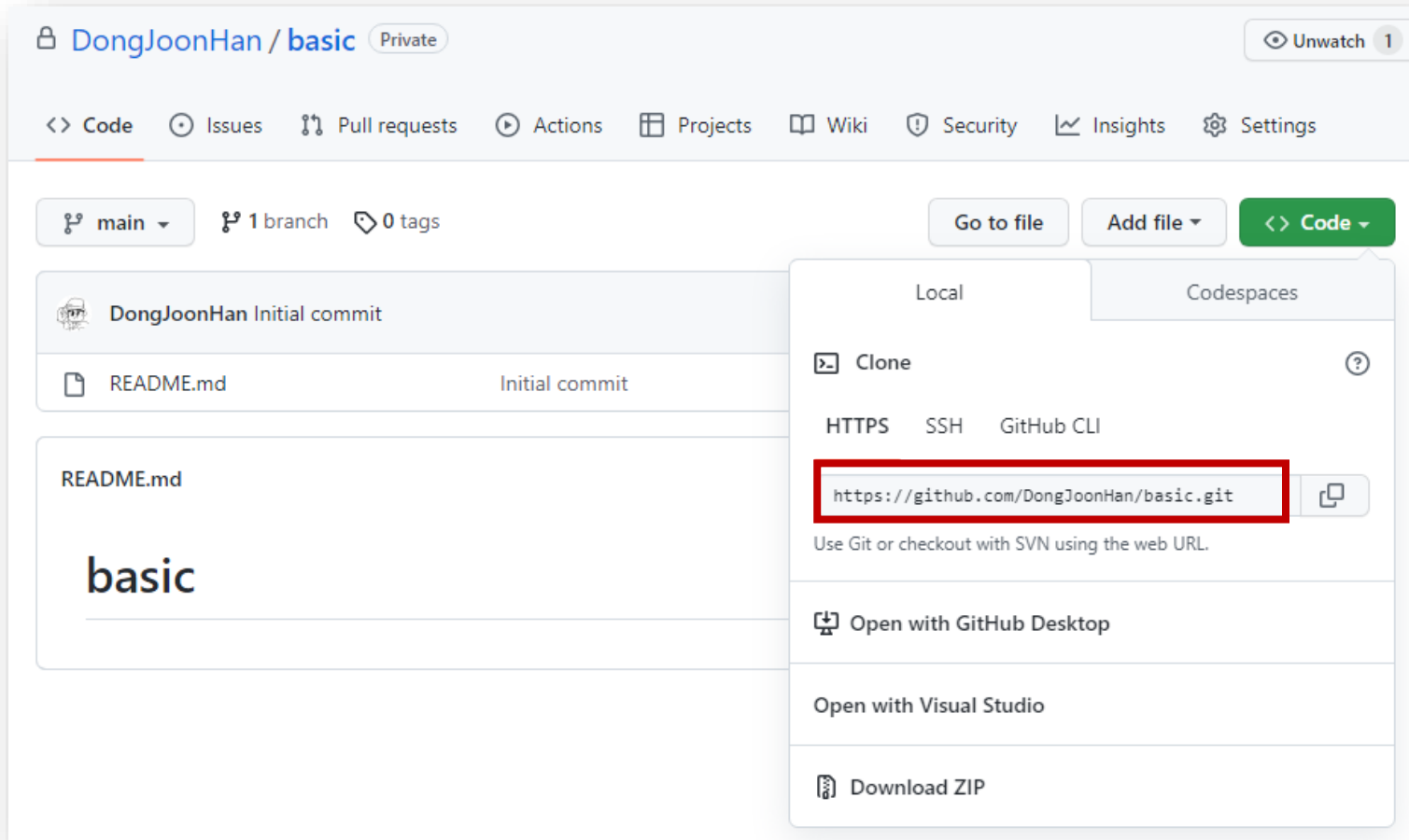
This is where you can write a long description for your project. [Learn more.](#)

저장소 생성 완료



저장소 Clone - 1

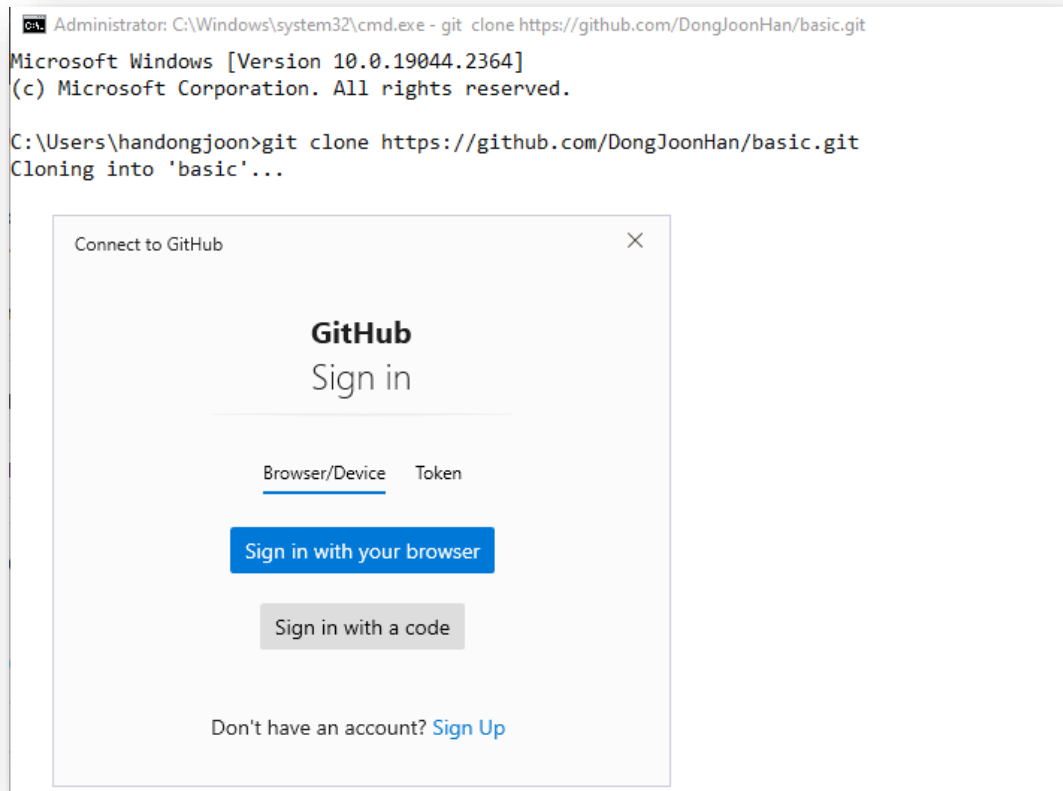
- ◆ 원격 저장소(Github)에서 내 로컬(PC)로 저장소를 복제함
 - 예시 개념) 네이버에 있는 카페 하나를 통째로 내 PC에 복제
- ◆ GitHub에서 생성한 저장소에서 원격 저장소 정보 확인



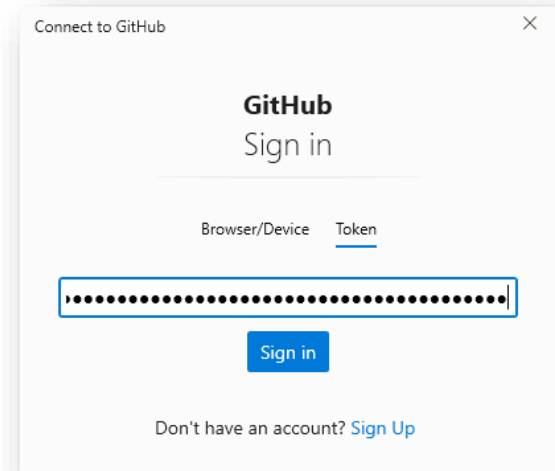
저장소 Clone - 2

◆ git clone <<Github에서 복사한 저장소 주소.git>>

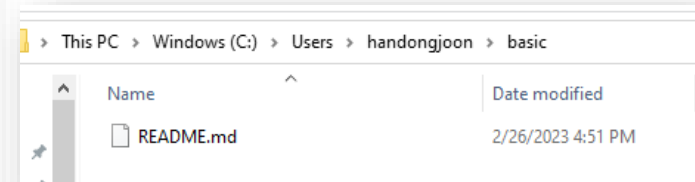
- Private의 경우 Github 로그인 정보 확인 -> Token으로 로그인



[계정 정보 확인]



[Token으로 로그인]

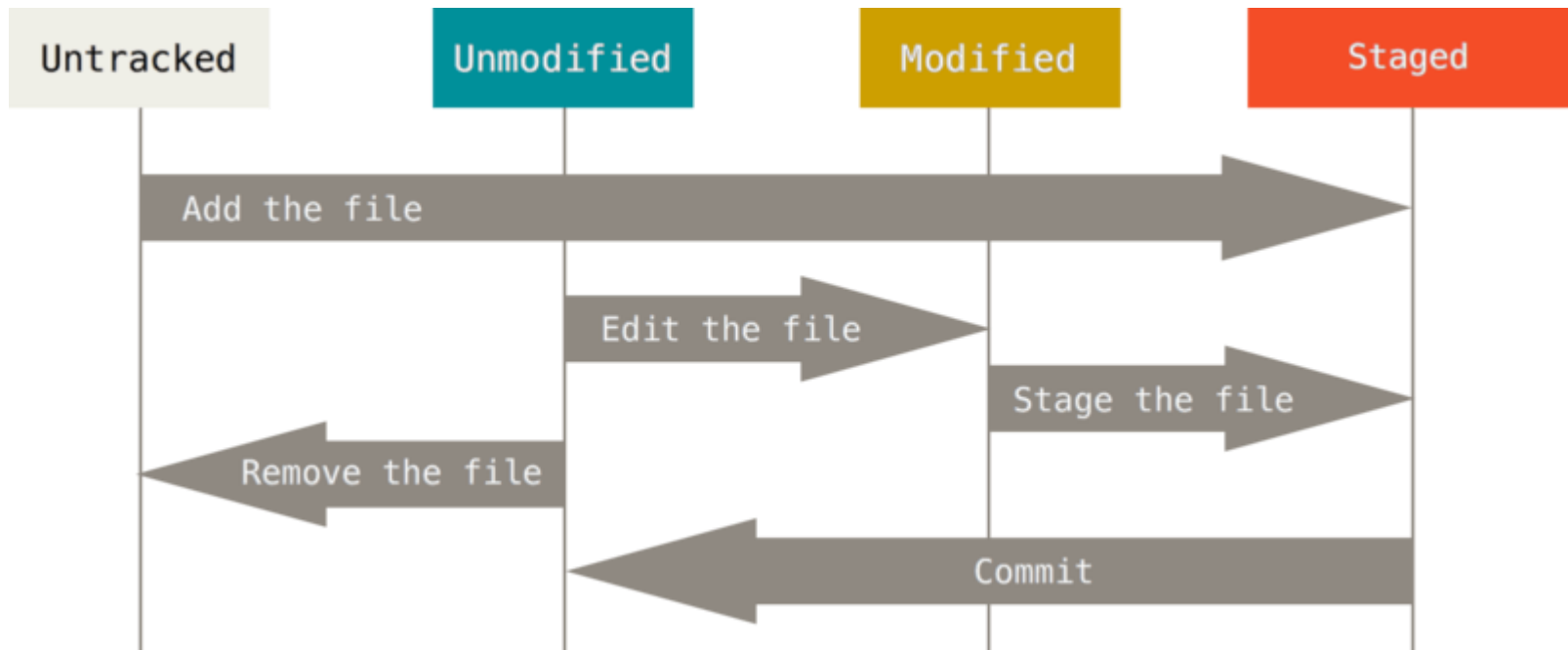


[Clone이 완료된 상태]

git status: The lifecycle of the status - 1

◆ git status

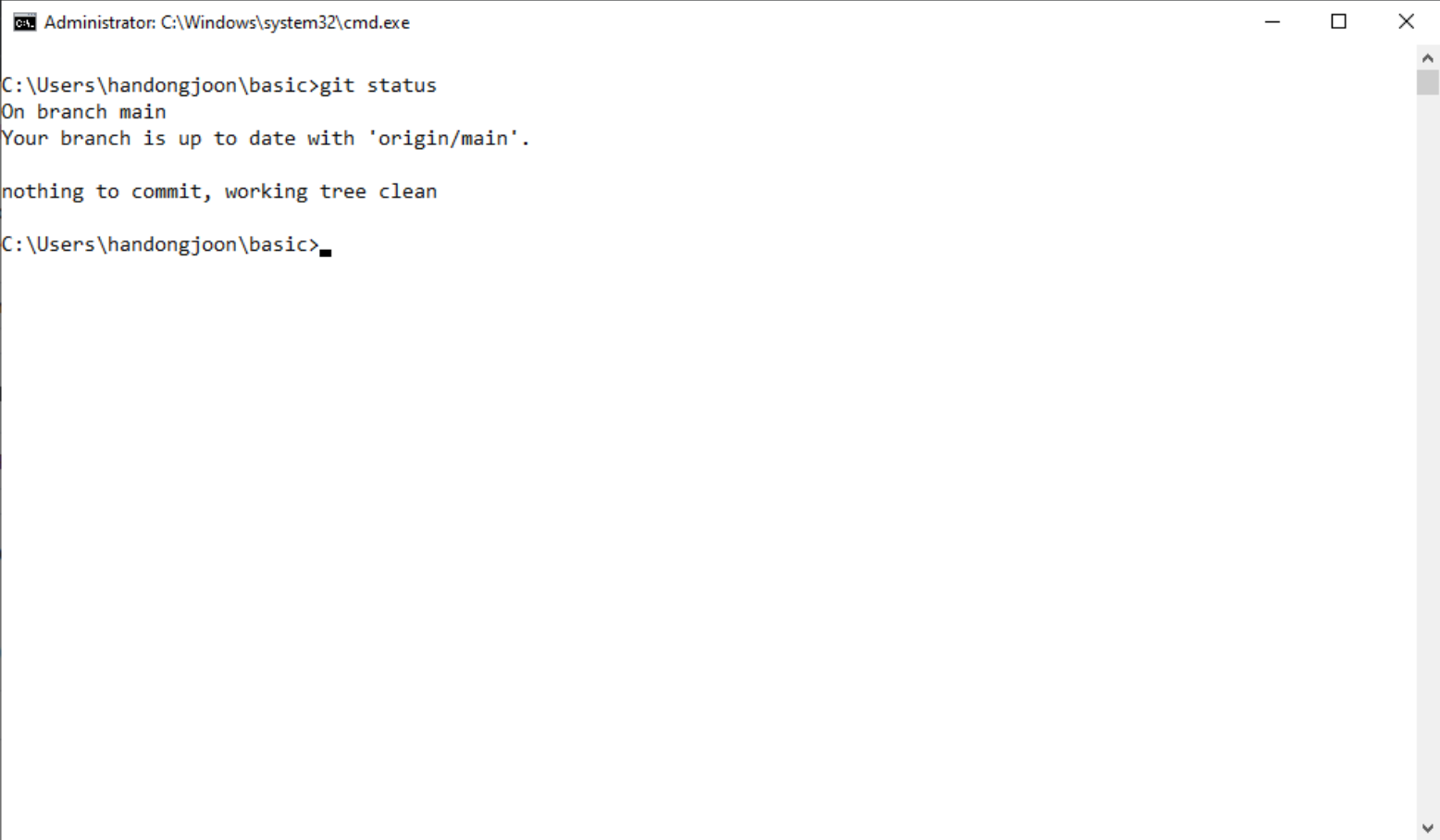
- 현재 상태를 확인하는 명령어
- 예시 개념) 내가 복제해온 카페에 글 등록되었나? -> 내가 안하면 될리가 없음



출처: <https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository>

git status: The lifecycle of the status - 2

- ◆ 현재 상태는 아무 것도 커밋할 것이 없음



```
Administrator: C:\Windows\system32\cmd.exe

C:\Users\handongjoon\basic>git status
On branch main
Your branch is up to date with 'origin/main'.

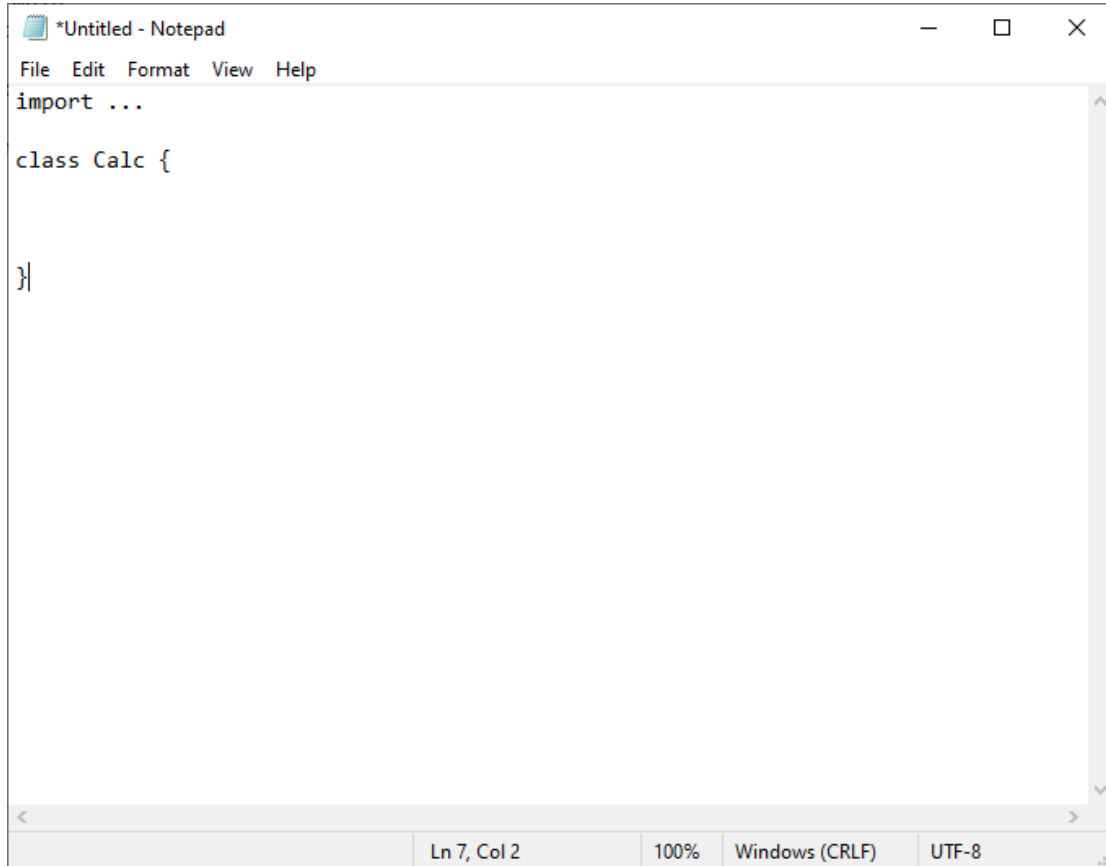
nothing to commit, working tree clean

C:\Users\handongjoon\basic>
```

소스코드 생성

◆ Calc.java 생성

- 메모장에서 간단히 생성
- <예시 개념> 카페에 올릴 글을 메모장에 적어본다.



```
*Untitled - Notepad
File Edit Format View Help
import ...

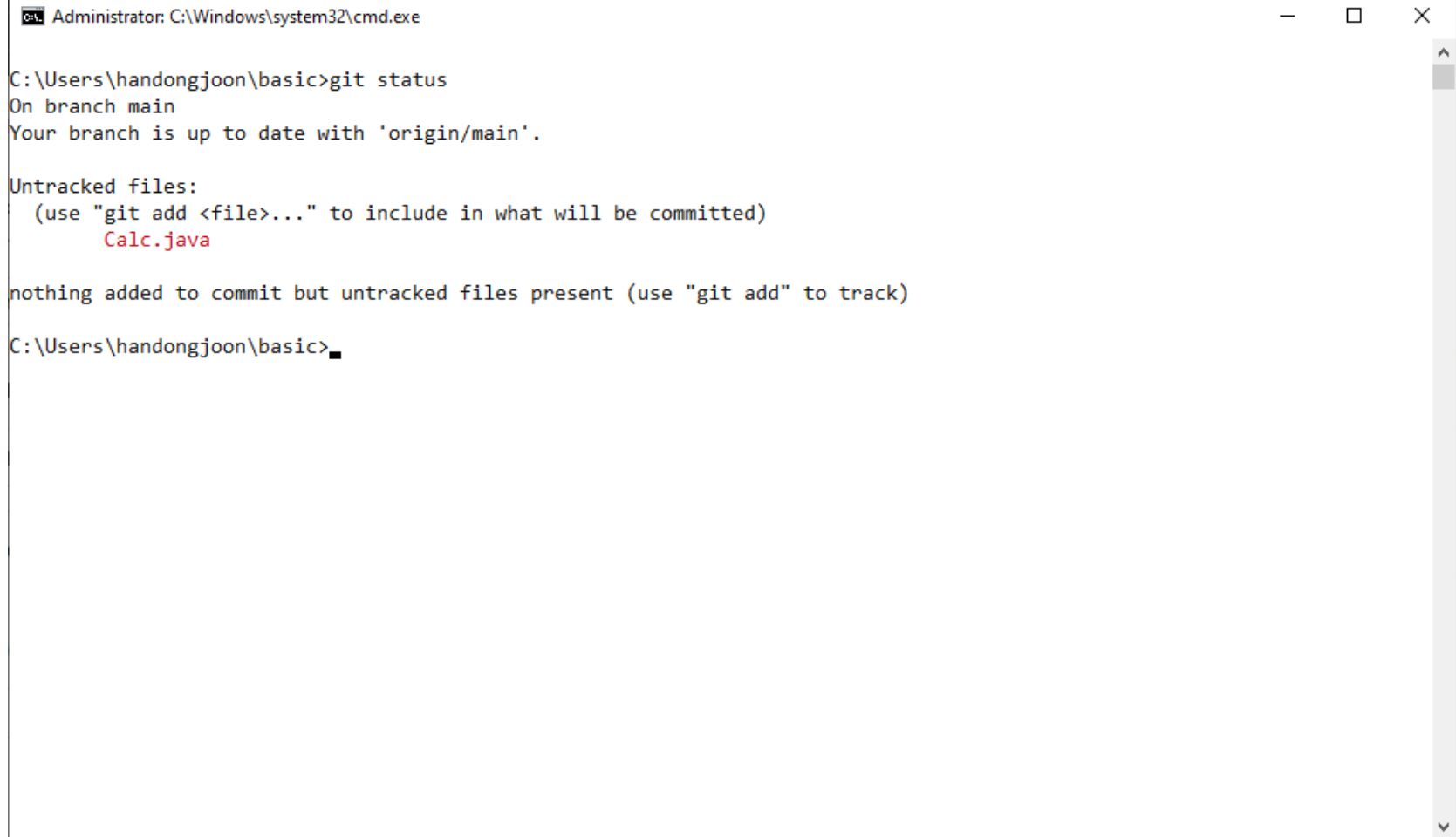
class Calc {

}

Ln 7, Col 2    100%    Windows (CRLF)    UTF-8
```

git status

- ◆ 추적되지 않는 파일이 있으니, 커밋하려면 git add를 먼저 하라는 메시지



```
Administrator: C:\Windows\system32\cmd.exe

C:\Users\handongjoon\basic>git status
On branch main
Your branch is up to date with 'origin/main'.

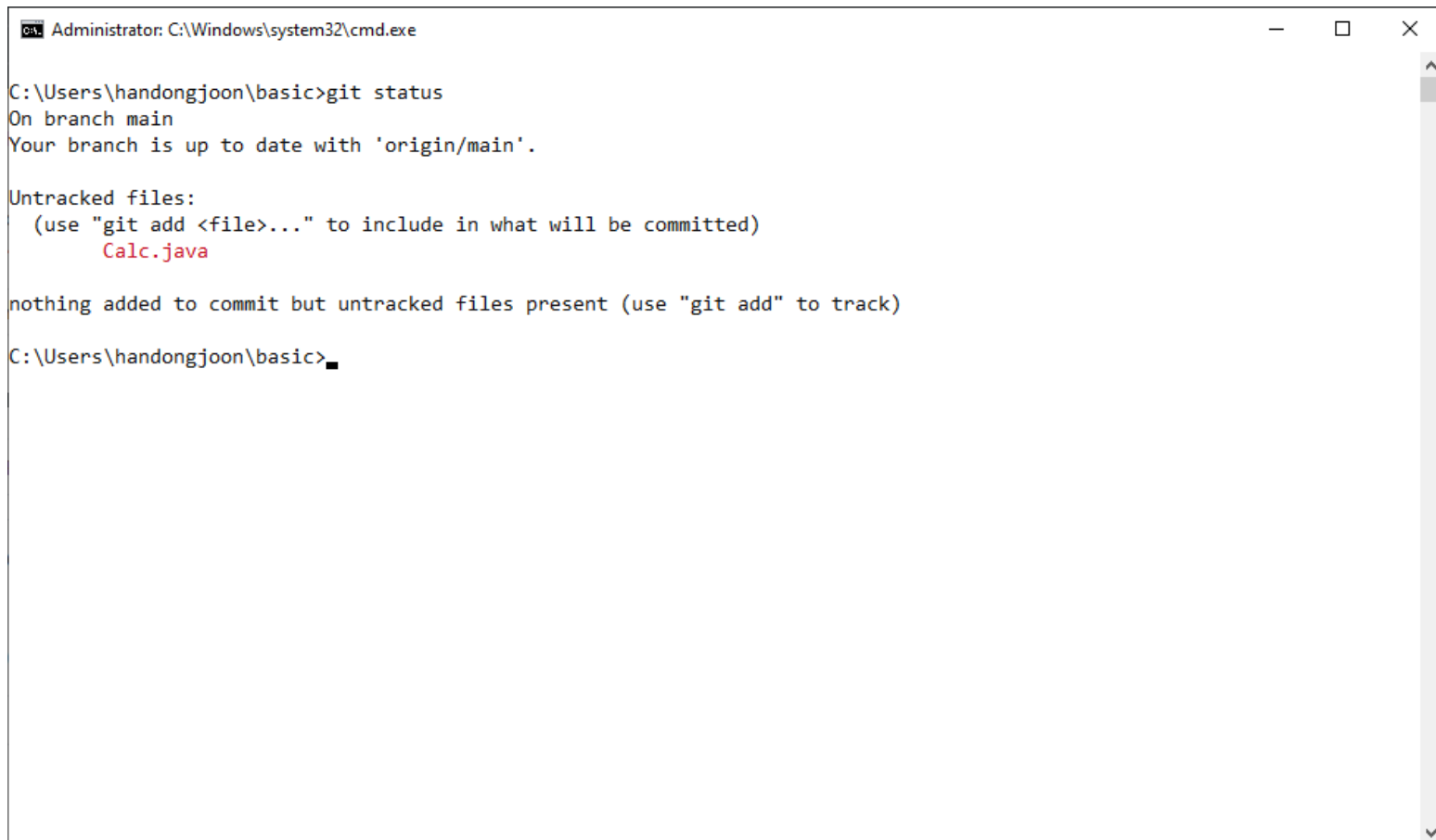
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Calc.java

nothing added to commit but untracked files present (use "git add" to track)

C:\Users\handongjoon\basic>
```

git add

- ◆ git add Calc.java
- ◆ Calc.java를 git의 관리 대상이자 커밋 대상(staged 상태)으로!
 - <예시 개념> 메모장에 올린 글을 게시판에 작성 하고 임시 저장



```
Administrator: C:\Windows\system32\cmd.exe

C:\Users\handongjoon\basic>git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
       Calc.java

nothing added to commit but untracked files present (use "git add" to track)

C:\Users\handongjoon\basic>
```

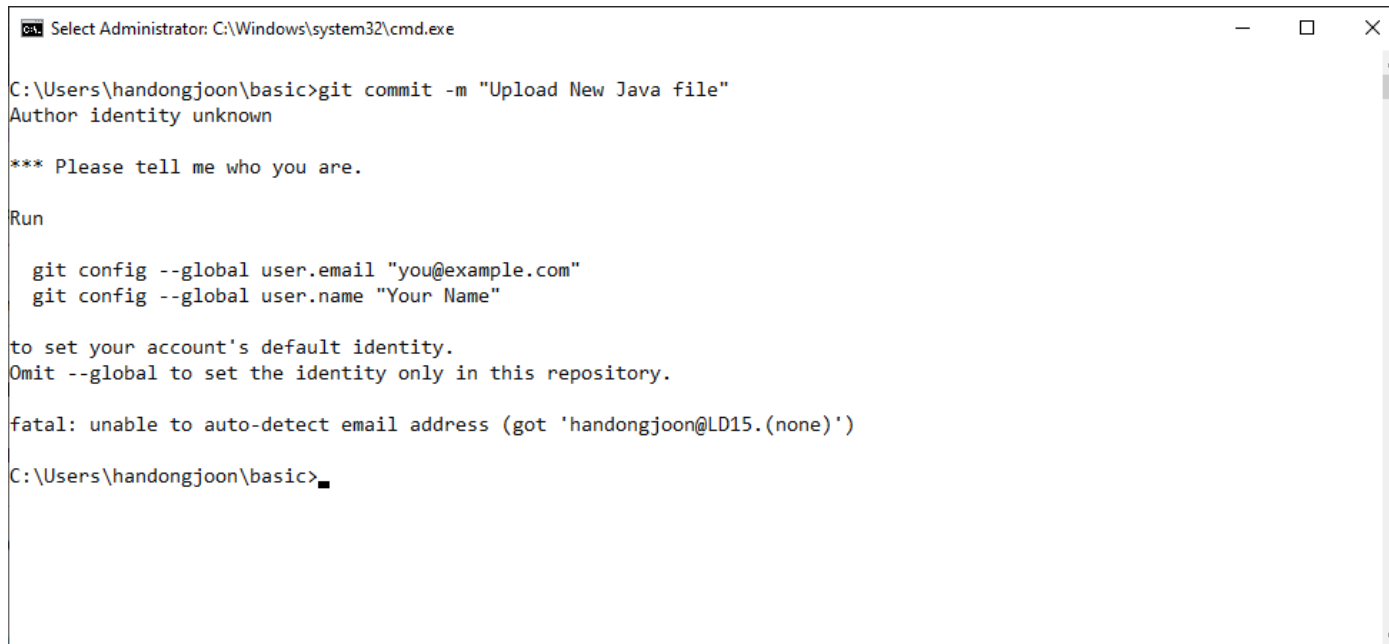

git commit - 1

◆ git commit -m "메시지"

◆ 변경 사항을 커밋해서 등록

- 등록 대상: 로컬 (아직 서버<원격 저장소>에는 공유하지 않음. 나만 가지고 있음)
- <예시 개념> 내 PC에 복제해온 카페에 글을 하나 등록함

◆ 그러나, 첫 Commit인 경우 기본 정보 입력 필요



```
Select Administrator: C:\Windows\system32\cmd.exe

C:\Users\handongjoon\basic>git commit -m "Upload New Java file"
Author identity unknown

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

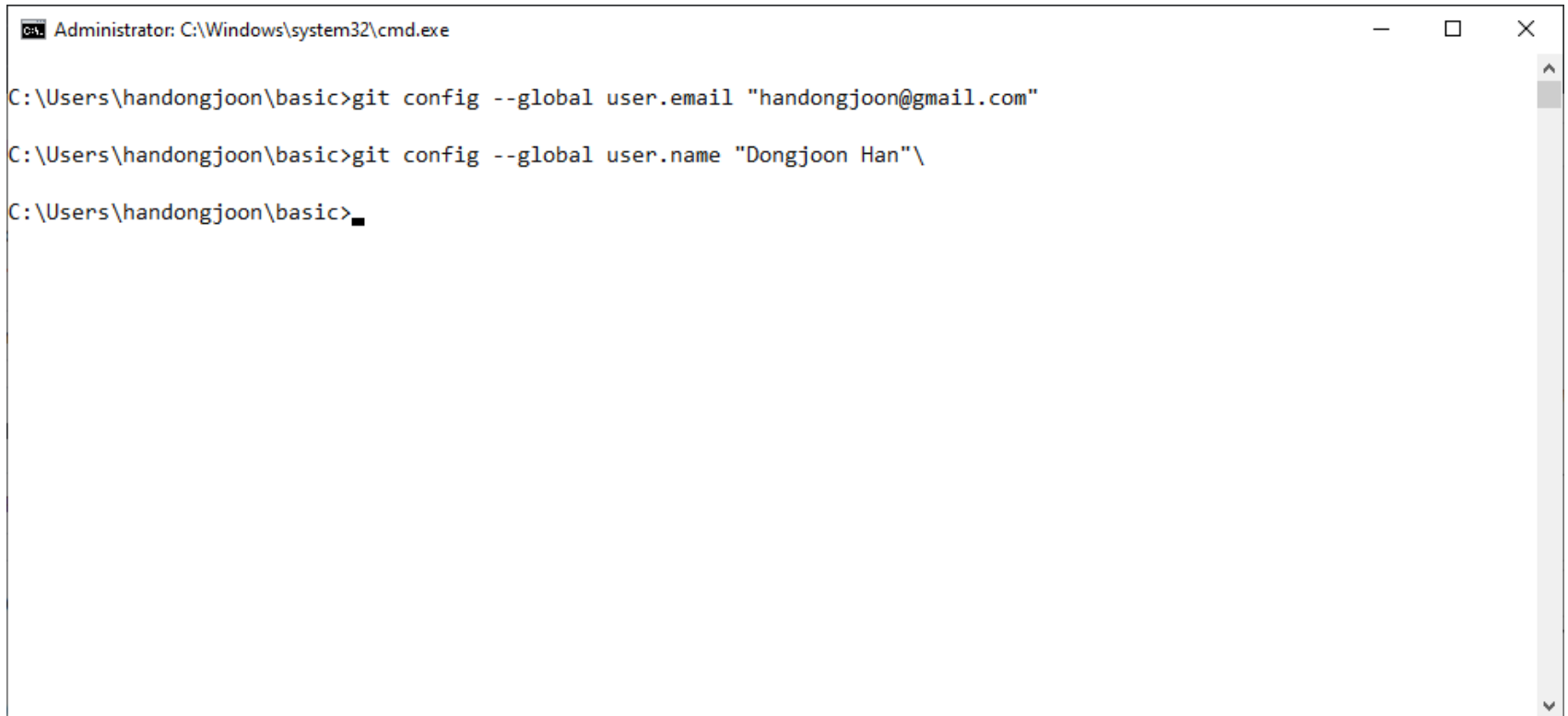
fatal: unable to auto-detect email address (got 'handongjoon@LD15.(none)')

C:\Users\handongjoon\basic>
```

git commit - 2

◆ 사용자 정보 입력

- `git config --global user.email "you@example.com"`
 - E-mail은 GitHub 등록 메일과 동일하게 설정. GitHub이 이 메일 정보로 사용자 연결
- `git config --global user.name "Your Name"`
 - 이름 작성



```
Administrator: C:\Windows\system32\cmd.exe

C:\Users\handongjoon\basic>git config --global user.email "handongjoon@gmail.com"

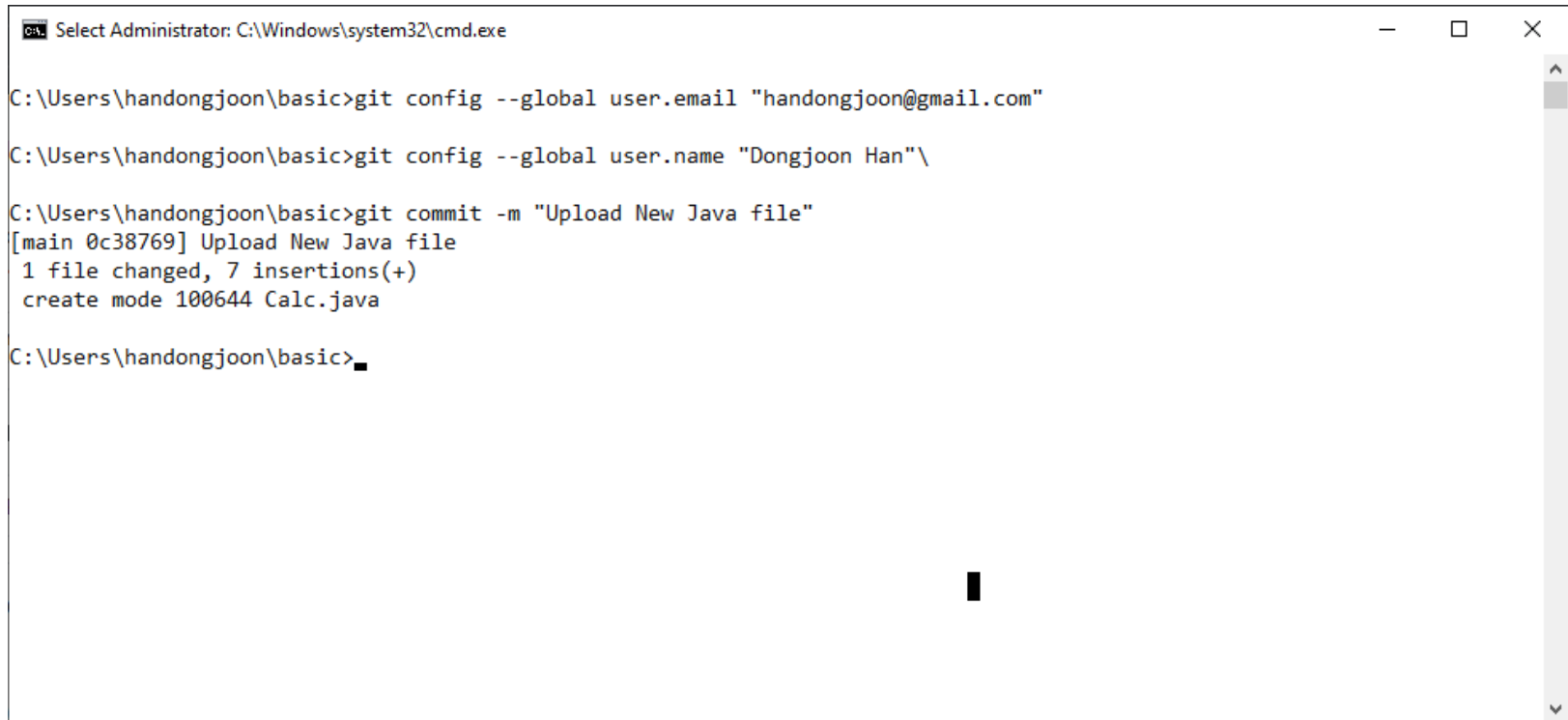
C:\Users\handongjoon\basic>git config --global user.name "Dongjoon Han"

C:\Users\handongjoon\basic>
```

git commit - 3

◆ 다시 한 번 커밋 시도

- 키보드 ↑ 버튼 누르면 이전 실행했던 명령어가 표시됨



```

C:\Users\handongjoon\basic>git config --global user.email "handongjoon@gmail.com"

C:\Users\handongjoon\basic>git config --global user.name "Dongjoon Han"

C:\Users\handongjoon\basic>git commit -m "Upload New Java file"
[main 0c38769] Upload New Java file
 1 file changed, 7 insertions(+)
 create mode 100644 Calc.java

C:\Users\handongjoon\basic>

```

◆ GitHub에서 확인하면, Calc.java가 추가되어 있지 않음

git push

◆ 내 로컬 저장소의 commit 내역을 원격 저장소(github)에 등록




- <예시 개념> 내 PC의 카페에 등록한 글을 네이버 카페에도 올리기

```
Administrator: C:\Windows\system32\cmd.exe

C:\Users\handongjoon\basic>git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 315 bytes | 315.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/DongJoonHan/basic.git
   0ca0cd4..0c38769  main -> main

C:\Users\handongjoon\basic>
```

- GitHub에서 확인해보기

	DongJoonHan Upload New Java file	0c38769 2 minutes ago	🕒 2 commits
	Calc.java	Upload New Java file	2 minutes ago
	README.md	Initial commit	32 minutes ago

실습

- ◆ Calc.java 변경하고 커밋 3회
- ◆ git push
- ◆ import 폴더 생성하고 Add.java 생성하기
- ◆ git add
- ◆ 커밋
- ◆ Add.java 변경하고 커밋 1회
- ◆ Calc.java와 Add.java 변경하고 커밋 1회
- ◆ git push

얼마나 자주 커밋하고 있는가?

- ◆ 정답은 없으나, 공통적인 의견은..

로직이 바뀌었다고 개발자가 판단하면 Commit

- ◆ 너무나 상대적인 이야기
- ◆ Change Set에 대한 고민 -> git add를 통해 commit 대상을 선정하는 것

결국, 변경을 공유하는 커뮤니케이션

커밋 메시지는 어떻게 작성해야 하는가?

◆ 현재까지는...

- 다른 사람이 이해 가능하도록 작성
- 커밋 메시지는 SW 개발의 매우 중요한 커뮤니케이션 요소

◆ Git이 익숙해지면

- 한 줄의 제목
- 한 줄의 공백
- 1~3 줄의 설명

◆ Redmine/Jira 등과 연동하게 되면 (이후 알아볼 내용)

- 이슈# 한 줄의 제목
- 한 줄의 공백
- 1~3 줄의 설명

Git의 Branch

◆ 브랜치(Branch)

- 원본의 복사본
- 대부분의 VCS(Version Control System)에서 지원
- 용도
 - 특정 기능을 원본에 영향을 주지 않고 개발/테스트 하기 위해
 - 베이스라인을 설정하기 위해

◆ Git를 보다 잘 사용하고 싶다면, 브랜치를 어떻게 사용하는가에 좌우

Branch 사용

◆ git branch

- branch 현황 보기

◆ git branch <<branch이름>>

- <<branch이름>> 으로 branch 만들기

◆ git switch <<branch이름>>

- <<branch이름>> 으로 branch 옮겨가기

◆ git merge <<합쳐질_branch이름>>

- branch를 합치는 것
- 합칠 branch에서 합쳐질 branch를 지정
- 예) (mast에서) git merge hotfix

[예제] Branch 사용

- ◆ git branch hotfix
- ◆ git switch hotfix
- ◆ (hotfix 에서) 새로운 파일을 만들고 커밋 2회
- ◆ git switch master
- ◆ git merge hotfix

04 Eclipse에서 git 사용하기

◆ Maven을 사용한다는 가정

[저장소 Fork] JUnit4 저장소 Fork

◆ Fork: 다른 사람의 저장소를 수정하기 위해, 내 저장소로 복사해 오는 기능

- 향후 Pull Request를 통해 원래 저장소에 병합
- 오픈소스에서 Contribute에 많이 사용하는 방식

The screenshot shows the GitHub repository page for `junit-team / junit4`. The repository has 647 watches, 6,581 stars, and 2,513 forks. It is a Java-based testing framework with 2,195 commits, 5 branches, 20 releases, and 138 contributors. The license is EPL-1.0. The page shows the 'master' branch and a list of recent commits, including one by stefanbirkner about building with Oracle JDK 7 on Travis.

Repository: `junit-team / junit4`

Watch 647 Star 6,581 Fork 2,513

Code Issues 106 Pull requests 13 Projects 0 Wiki Insights

A programmer-oriented testing framework for Java. <http://junit.org/junit4/>


2,195 commits 5 branches 20 releases 138 contributors EPL-1.0


Branch: master New pull request Create new file Upload files Find file Clone or download

stefanbirkner Don't build with Oracle JDK 7 on Travis anymore ... Latest commit a832c5a on 16 Oct

File	Commit Message	Time
<code>.mvn/wrapper</code>	Build with Maven 3.1.1 (using Maven Wrapper)	2 months ago
<code>.settings</code>	Revert on request of kcooney:	4 years ago
<code>doc</code>	Build with Maven 3.1.1 (using Maven Wrapper)	2 months ago
<code>lib</code>	Add Hamcrest source JAR for easy reference	6 years ago
<code>src</code>	Fix dead link to the ant task in FAQ (#1478)	4 months ago
<code>.classpath</code>	Add resource source folders to Eclipse project	3 years ago
<code>.gitattributes</code>	Use Unix-style line endings. (#1405)	11 months ago
<code>.gitignore</code>	Use Unix-style line endings. (#1405)	11 months ago

[저장소 Fork] 내 저장소에 Fork 확인

 This repository Search Pull requests Issues Marketplace Explore

 handongjoon / junit4
forked from junit-team/junit4

Unwatch 1 Star 0 Fork 2,513

[Code](#) Pull requests 0 Projects 0 Wiki Insights Settings


A programmer-oriented testing framework for Java. <http://junit.org/junit4/> [Edit](#)




[Add topics](#)

2,196 commits 5 branches 20 releases 138 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

This branch is 1 commit ahead of junit-team:master. [Pull request](#) [Compare](#)

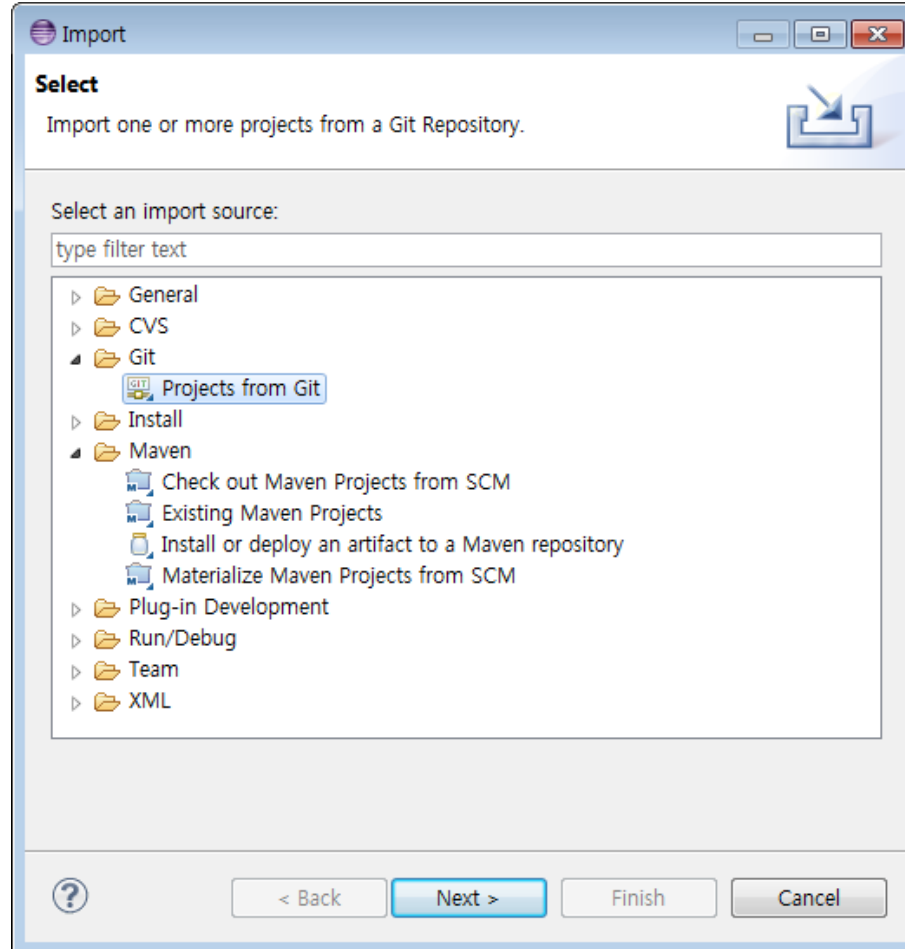
 lexia q123 Latest commit db13b1a an hour ago

 .mvn/wrapper	Build with Maven 3.1.1 (using Maven Wrapper)	2 months ago
 .settings	Revert on request of kcooney:	4 years ago
 doc	Build with Maven 3.1.1 (using Maven Wrapper)	2 months ago

[Eclipse] 내 로컬로 Clone

◆ SVN과 동일하게, Import 기능 사용

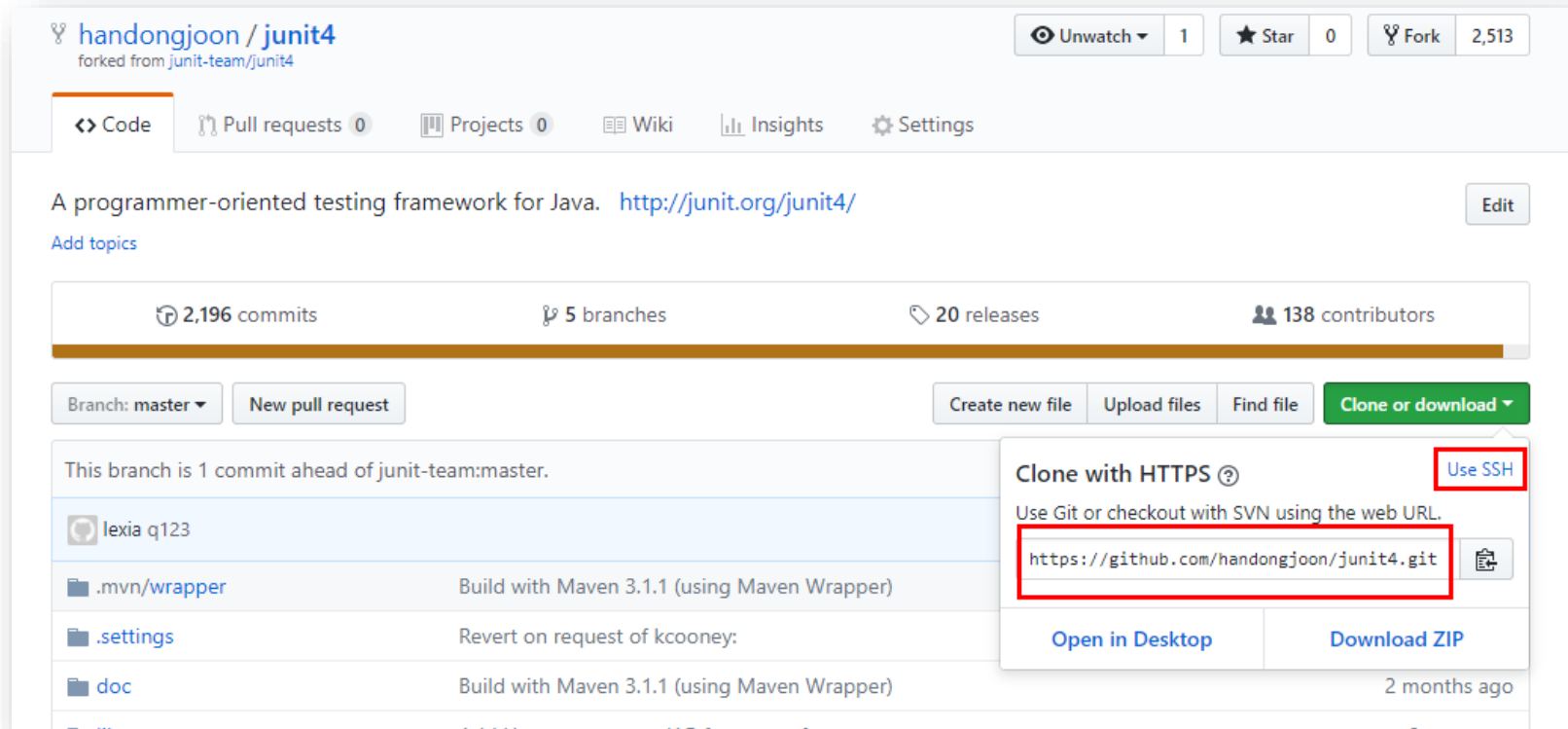
- Maven 프로젝트 이므로, Maven 하위의 체크아웃 메뉴 이용



[Eclipse] 내 로컬로 Clone

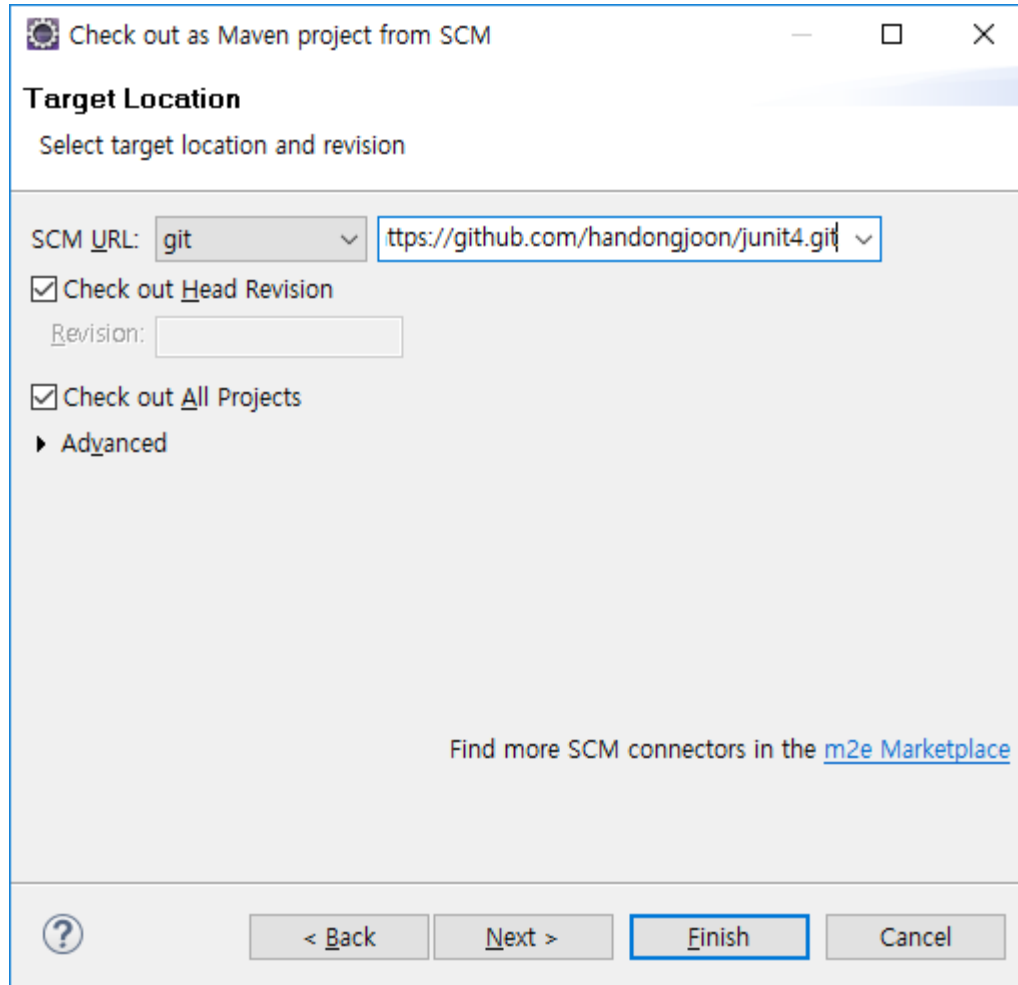
◆ Fork 한 저장소의 Clone을 위한 주소 받아오기

- ssh 설정이 되어있지 않다면, Github 권한으로 사용하기 위해 'Use HTTPS' 클릭



[Eclipse] 내 로컬로 Clone

◆ Eclipse에 Clone 대상 저장소 위치 지정



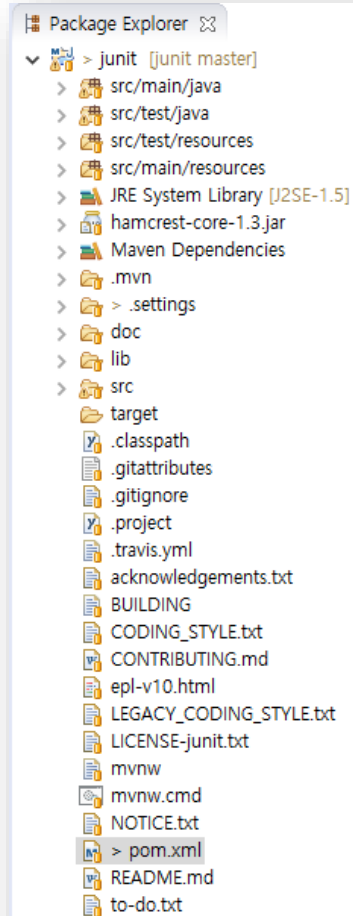
- 여기서 SCM URL에 git이 안보인다면!!!
- 우측 하단 m2e Marketplace에서 Maven 플러그인 설치가 필요

[Eclipse] 소스코드 수정

◆ 에러가 나는 pom.xml의 플러그인 주석처리

- 저장 후, pom.xml 파일과 상위 폴더에 '>' 표시

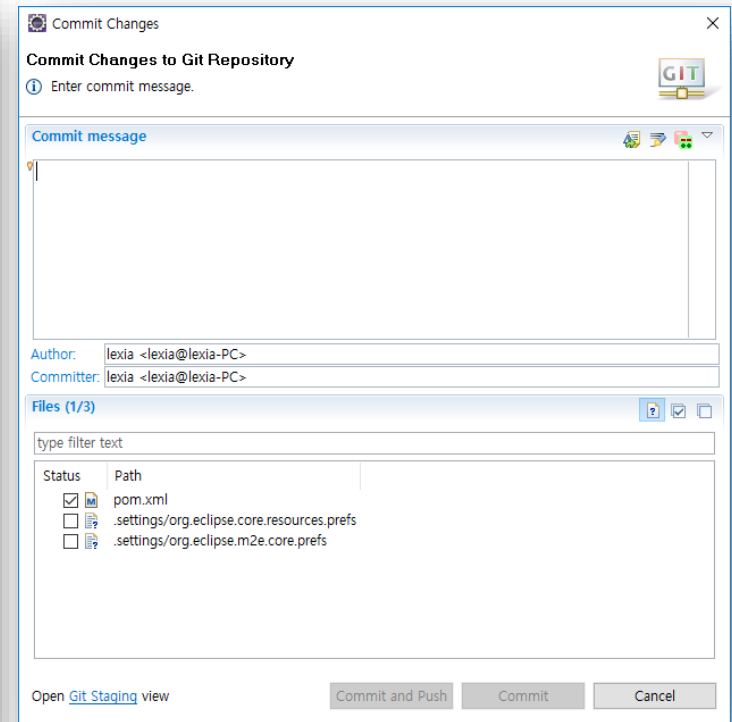
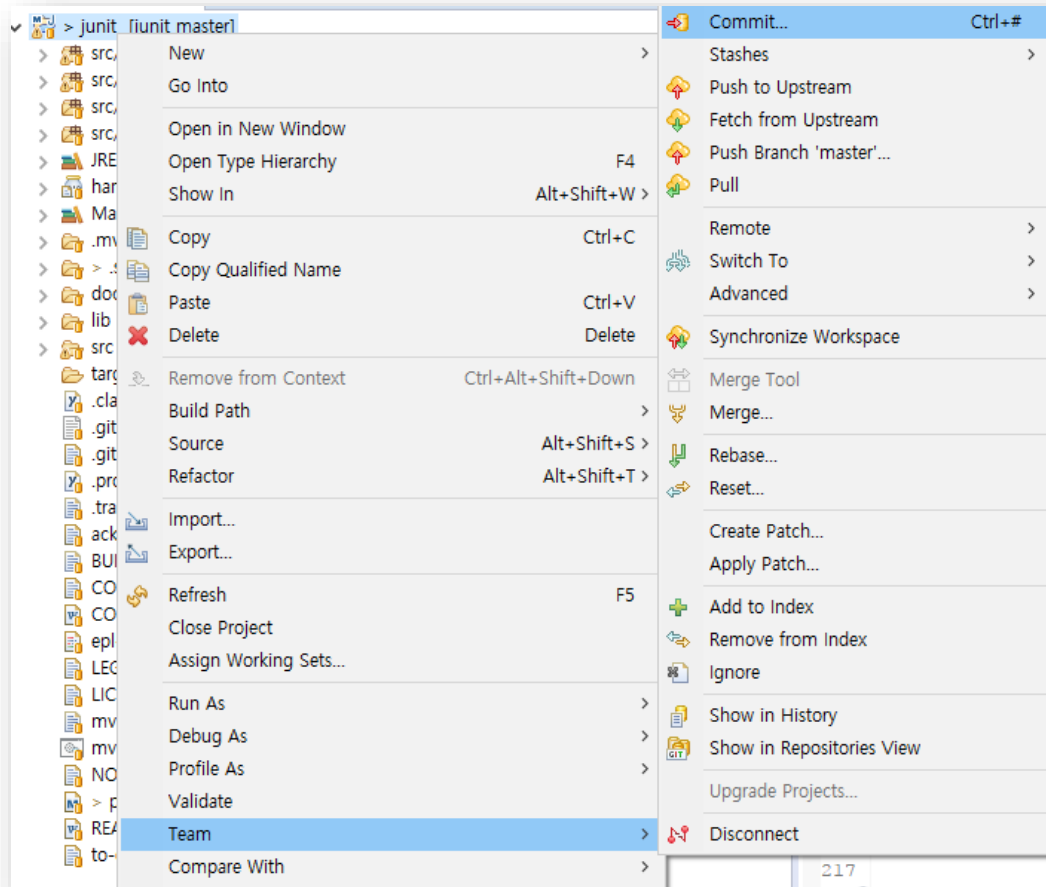
```
193         </plugin>
194         <!--
195         <plugin>
196
197             <groupId>com.google.code.maven-replacer-plugin<
198             <artifactId>replacer</artifactId>
199             <version>1.5.3</version>
200             <executions>
201
202             </executions>
203             <configuration>
204                 <ignoreMissingFile>false</ignoreMissingFile>
205                 <file>${project.build.sourceDirectory}/juni
206                 <outputFile>${project.build.sourceDirectory
207                 <regex>false</regex>
208                 <token>@version@</token>
209                 <value>${project.version}</value>
210             </configuration>
211         </plugin>-->
212         <plugin><!-- Using jdk 1.5.0_22, package-info.java
213         <!--
214             java compiler plugin forked in extra process
```



[Eclipse] 변경내용 Commit

◆ Team - Commit 실행

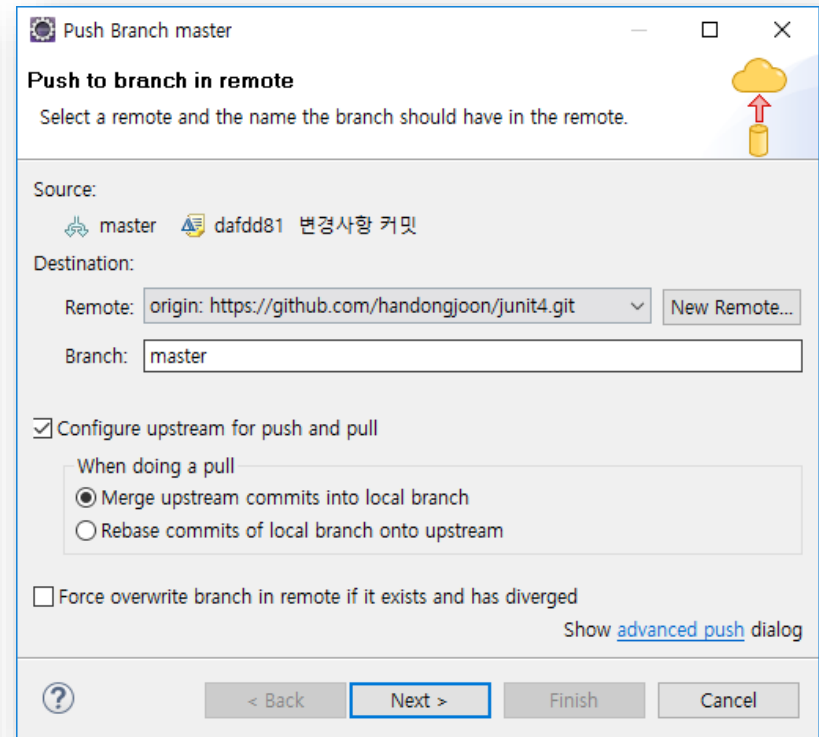
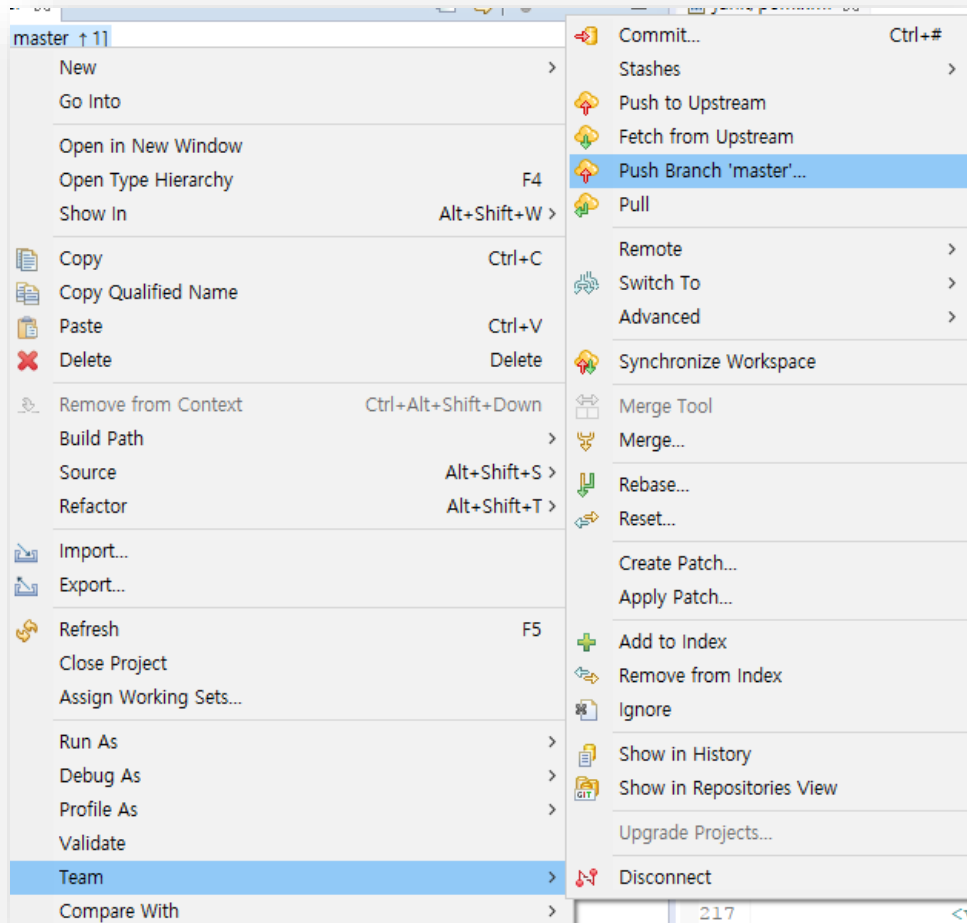
- SVN과 유사하게 커밋 메시지 작성 부분 표시
- 커밋 대상 선정 가능



[Eclipse] 원격 저장소로 Push

◆ Commit 내용 원격 저장소로 Push 하기

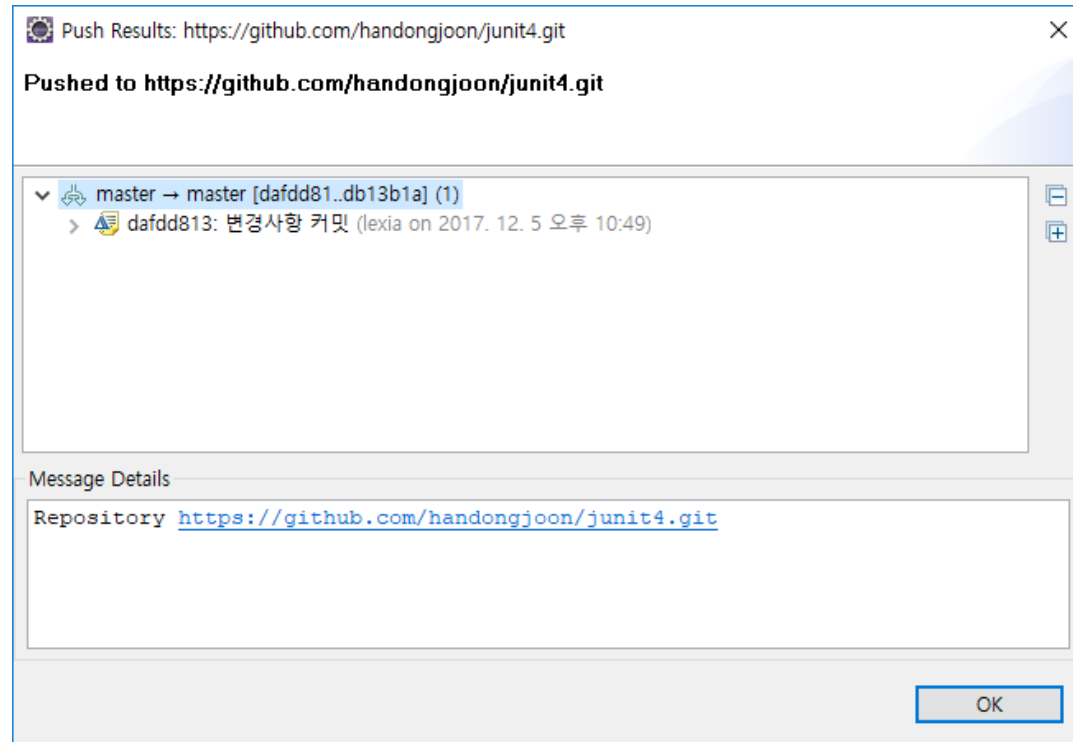
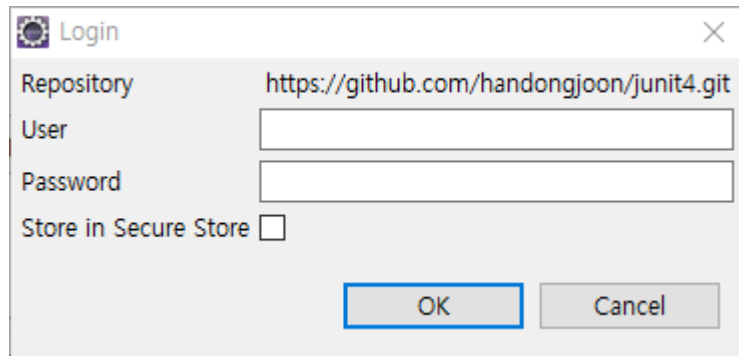
- 현재 작업 중인 master 브랜치 Push



[Eclipse] 원격 저장소로 Push

◆ Github 계정으로 권한 확인

- Github의 내 저장소에서 변경내용 적용 확인



감사합니다.