



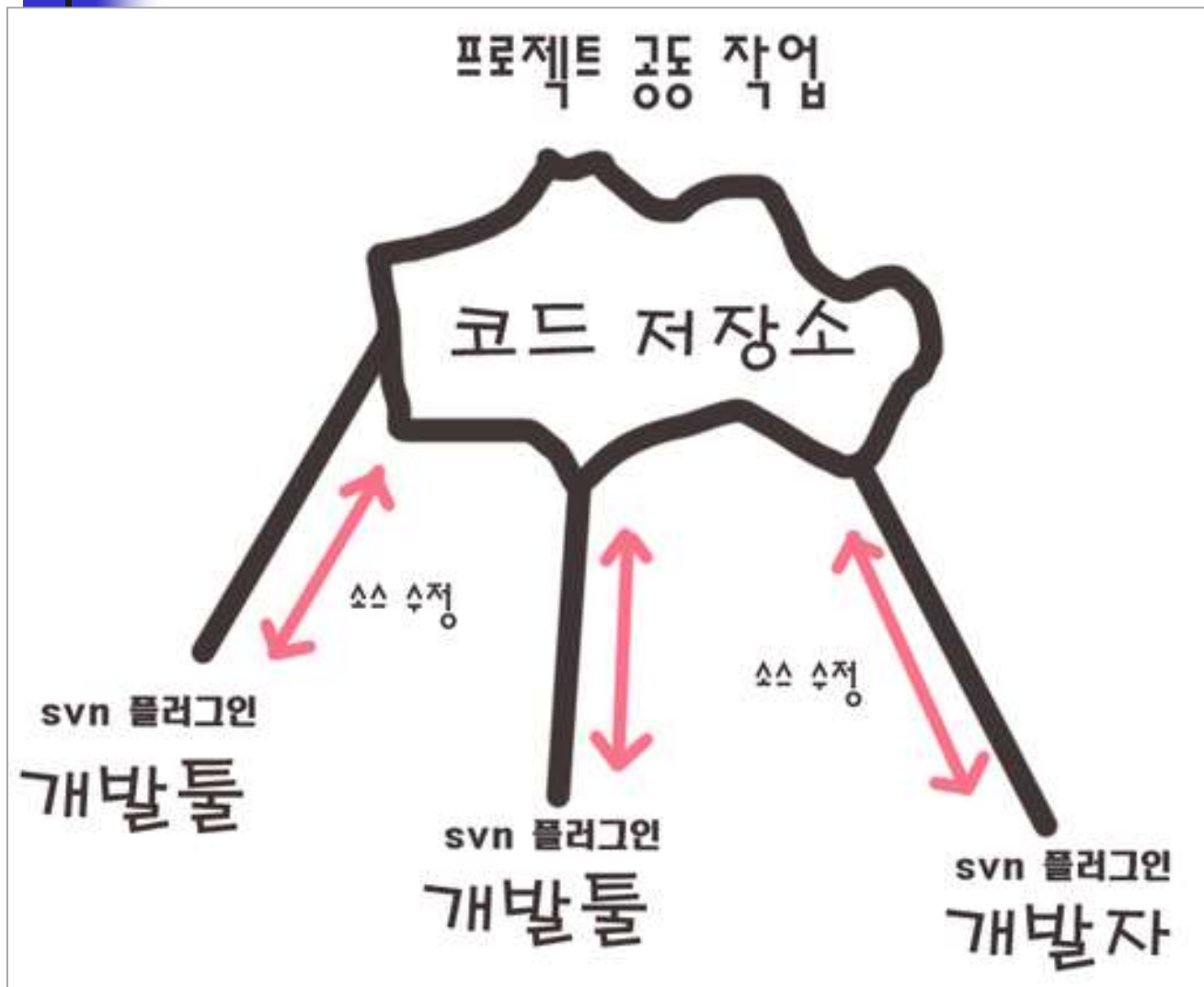
# GIT

---

양 명 속

**[[now4ever7@gmail.com](mailto:now4ever7@gmail.com)]**

# SVN





# SVN

---

- 서브버전(Subversion)
  - 자유 소프트웨어 버전 관리 시스템
  - 소프트웨어의 버전을 관리하고, 팀이 같이 코드를 관리할 수 있게 해주는 등의 기능을 가진 시스템
  - 제한이 있던 CVS를 대체하기 위해 2000년부터 콜랩넷에서 개발되었다.
- 서브 버전은 서버-클라이언트 모델을 따른다.
  - 서버는 작업하는 컴퓨터내에 둘 수도 있고, 전산망에 연결된 별도의 컴퓨터에 두고 사용할 수도 있다.
  - 서브버전 서버와 클라이언트는 http, https, svn, svn+ssh의 규약으로 통신한다.
- 네이버 개발자 센터, SourceForge.net, Tigris.org, Google code에서는 오픈소스 프로젝트를 위해 서브버전 호스팅을 하고 있다.



# 버전 관리 시스템

- 프로젝트엔 여러 사람이 동시에 개발을 진행하게 되기 때문에, 프로젝트 관리를 위한 팀 시스템의 도입이 필수
- **개발 소스관리와 버전 컨트롤을 위해** SVN(Subversion), GIT 등을 사용
- 여러 개발자가 하나의 프로젝트를 진행함에 있어 통합적인 관리를 할 수 있는 툴, 형상관리를 하기 위한 어플리케이션이 필요하다.
  - 예) SVN, GIT
- 각 수정된 소스 파일에 일률적인 버전과 로그를 남기고 각 파일의 빌드 이력을 남겨 하나의 프로젝트를 지원하기 위한 어플리케이션
- 개발자가 어떤 파일을 수정하고 해당 이력을 남기고 서버에 저장하면 다른 사용자는 해당 변경된 내역을 기록과 함께 열람하고 자신이 가지고 있는 파일의 변경사항을 변경, 적용할 수 있도록 도와주는 것이다.
- SVN은 **소스의 변경사항 이력관리가 가능**하고, 타인의 변경 사항에 대해서 현재 작업중인 파일에 대한 비교, 조합(Merge) 가 가능하며, 변경사항에 대해서 다른 개발 방향으로 분기(Branch)가 가능하다.
- SVN은 **소스에 대한 이력관리, 버전 관리하는 프로그램**



# 버전 관리 시스템

---

## ■ 버전관리 시스템

- 파일의 변경 이력을 관리하는 시스템으로 소스코드나 문서 파일의 수정 이력을 보관하고 관리한다.
- 여러 개발자가 동시에 많은 파일을 변경하더라도 덮어 쓰거나 충돌하는 문제를 방지할 수 있어야 한다.
- 다른 버전(Branch)으로 개발된 소스를 현재 소스와 합치고, 변경 이력에 대해서도 추적이 가능하다.
- 버전관리 시스템은 코딩작업 중 예상하지 못한 문제를 방지할 수 있으며, 소스코드의 변경 이력 추적이 가능하고, 동시에 다양한 버전의 개발 작업이 가능하다.
- 버전관리 시스템의 종류로는 CVS(Concurrent Version System), Visual SourceSafe, SVN(Subversion) 그리고 최근 각광받는 GIT(분산 버전관리시스템: Distributed Version Control Systems : DVCS) 이 있다.



# 깃허브

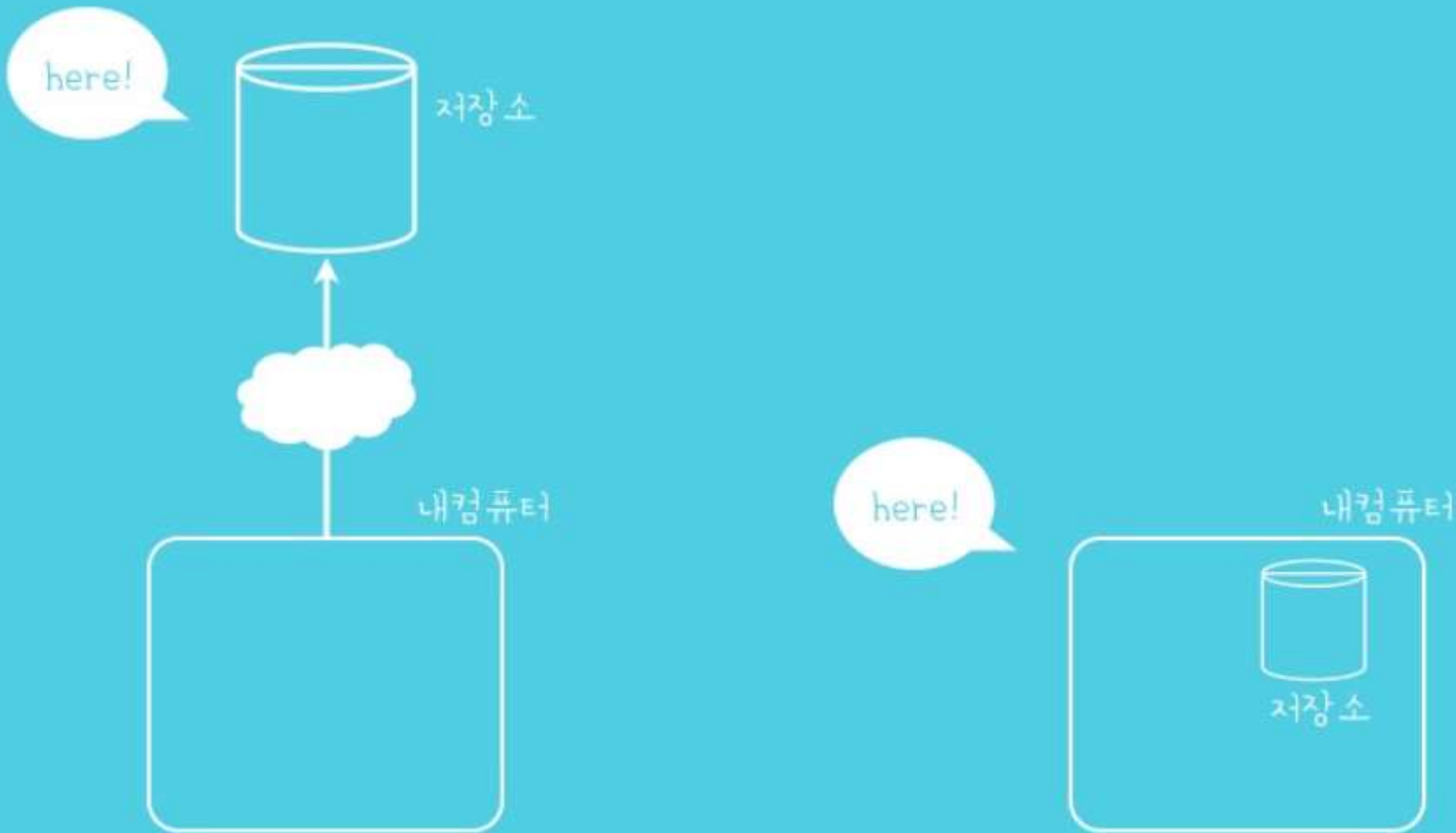
---

## ■ 1. 깃이란

- 깃허브의 심장에서 작동되는 소프트웨어
- 깃은 프로젝트의 어떤 부분도 겹쳐쓰지 않게 **프로젝트의 변경을 관리하는 버전관리 소프트웨어**
- 당신과 동료는 같은 페이지에 각자의 수정사항을 각각 업로드할 수 있고, 깃은 두 개의 복사본을 저장한다. 나중에, 당신들은 그대로 어떤 작업도 잃어버리지 않고 변경사항들을 병합할 수 있다.
- 깃은 이전에 만들어진 모든 변경사항의 “스냅샷”을 저장하기 때문에 **이전 시점의 어떤 버전으로 되돌릴 수도** 있다.
- 깃을 사용할 때 어려운 점은 코드를 타이핑하는 명령어(커맨드 라인)를 사용하여 접근해야 하는 것.

svn은 보통 저장소가  
서버에 있습니다.

git은 저장소가  
내 컴퓨터에 있습니다. 응?

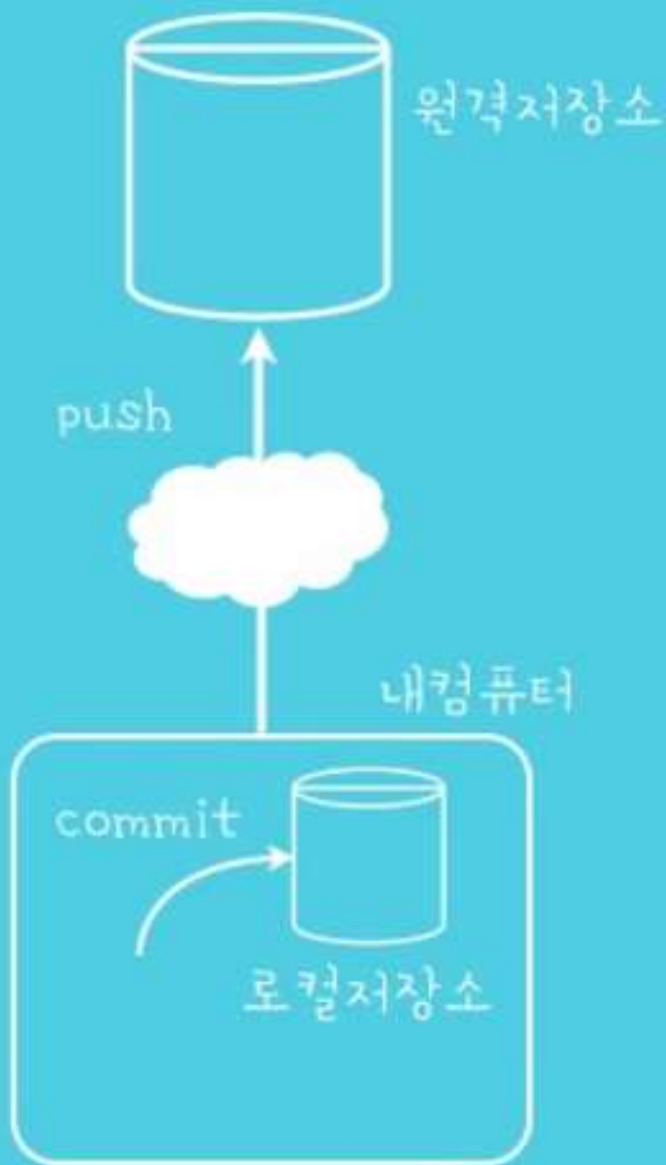


그럼 git은 다른 사람들과 작업을 할 수 없나요?

원격 저장소를 만들면 됩니다.  
remote repository

② 다른 사람과 공유할 때 원격 저장소에 푸시합니다.

① 내 컴퓨터의 저장소에 열심히 작업 내용을 커밋하고





svn



git



이렇게 저장소가 분산되는 구조를 분산버전관리시스템(DVCS)라고 합니다.

그런데 git에는 로컬저장소에 커밋 전, 하나의 단계가 더 있습니다.

바로 **스테이지 영역**입니다.

인덱스(index)라고도 부릅니다.



svn



git



Add index => commit => push

# 스태이지 영역이 어떤 역할을 하는지 알아볼까요?

① 이런 빌드 목표를 가지고 작업을 하고 있습니다.

## 빌드 목표!

로그인 기능 추가

A 버그 수정

B 버그 수정

# 스태이지 영역이 어떤 역할을 하는지 알아보을까요?

② 다음과 같이 파일들이 수정되었습니다.



# 스테이지 영역이 어떤 역할을 하는지 알아보을까요?

- ③ 그런데 로그인 기능 추가가 다음 빌드로 미루어졌습니다.  
svn 이라면 보통 어떻게 할까요?



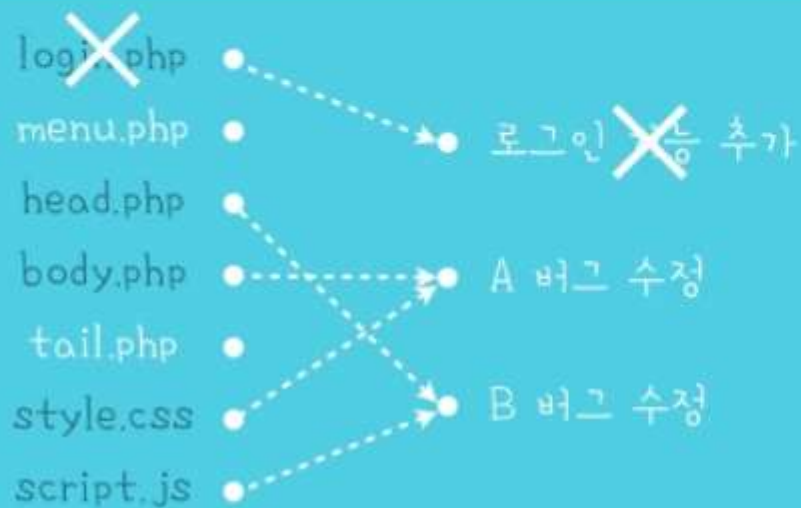
- ① 수정 된 login.php를 어딘가로 백업
- ② svn revert를 이용해 원래 상태로 복원
- ③ 수정 내역 전체를 커밋
- ④ 백업해두었던 login.php를 다시 복구



변경 된 파일들은 무조건 커밋 대상이 되기 때문에 생기는 문제입니다.

# 스테이지 영역이 어떤 역할을 하는지 알아볼까요?

## ④ 그럼 git은 어떻게 하나요?



- ❶ 커밋 할 파일들만 staging area에 추가
- ❷ 로컬저장소로 커밋

귀찮을 수 있습니다.

특별히 파일들을 구분 할 필요가 없을 때에는  
-a 옵션으로 스테이치 추가와 커밋을 동시에 할 수 있습니다.

# git commit -a

커밋을 마쳤으면, 이제 다른 사람들에게 작업물을 공유합니다.

원격저장소에 올리는 명령어는 **push** 입니다.

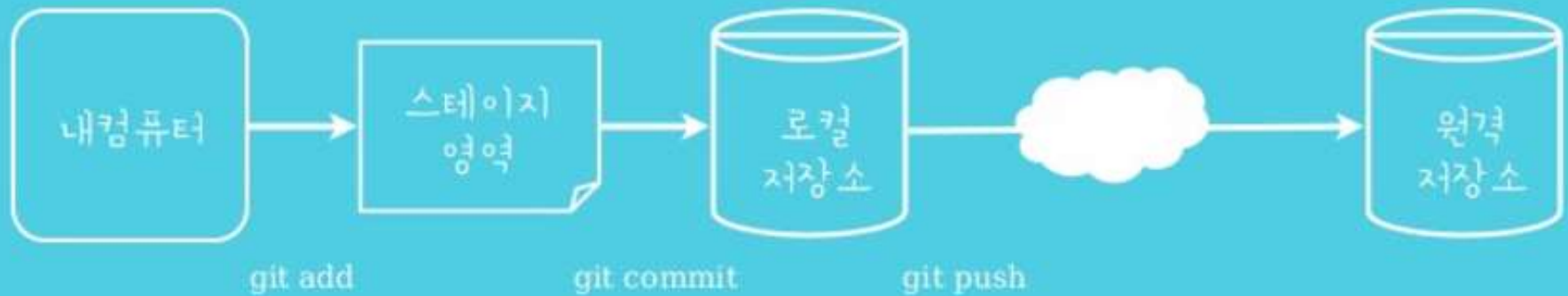
# git push



svn



git



Add index => commit => push

내 작업을 올리는데 오류가 발생했어요.  
다른 사람들이 작업 한 내용을 먼저 받아야 한대요.

원격저장소로부터 내려받기: **fetch**  
내려받은 데이터를 병합: **merge**

하지만 이 둘을 함께 처리하는 명령은 **pull**



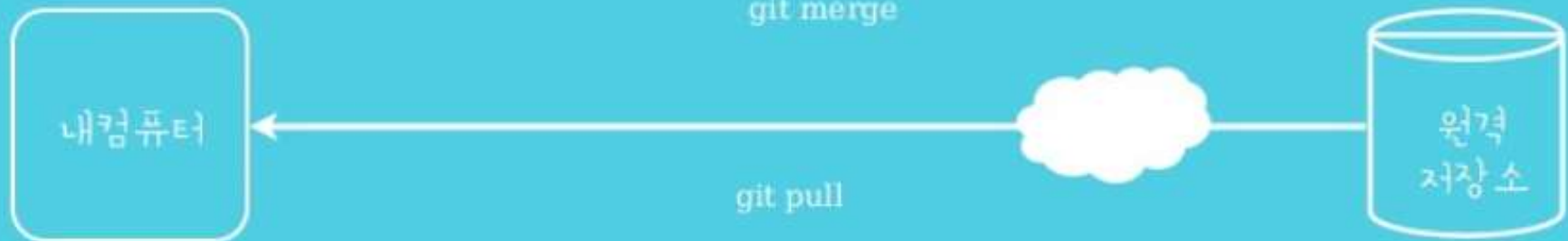
# git pull

pull하기 전에 **commit** 먼저 해야 함

svn



git



## 지금까지 알아본 svn과의 차이점은?

- ① 로컬저장소가 존재한다.
- ② 커밋 이전에 스테이지 영역에 추가하는 과정이 있다.

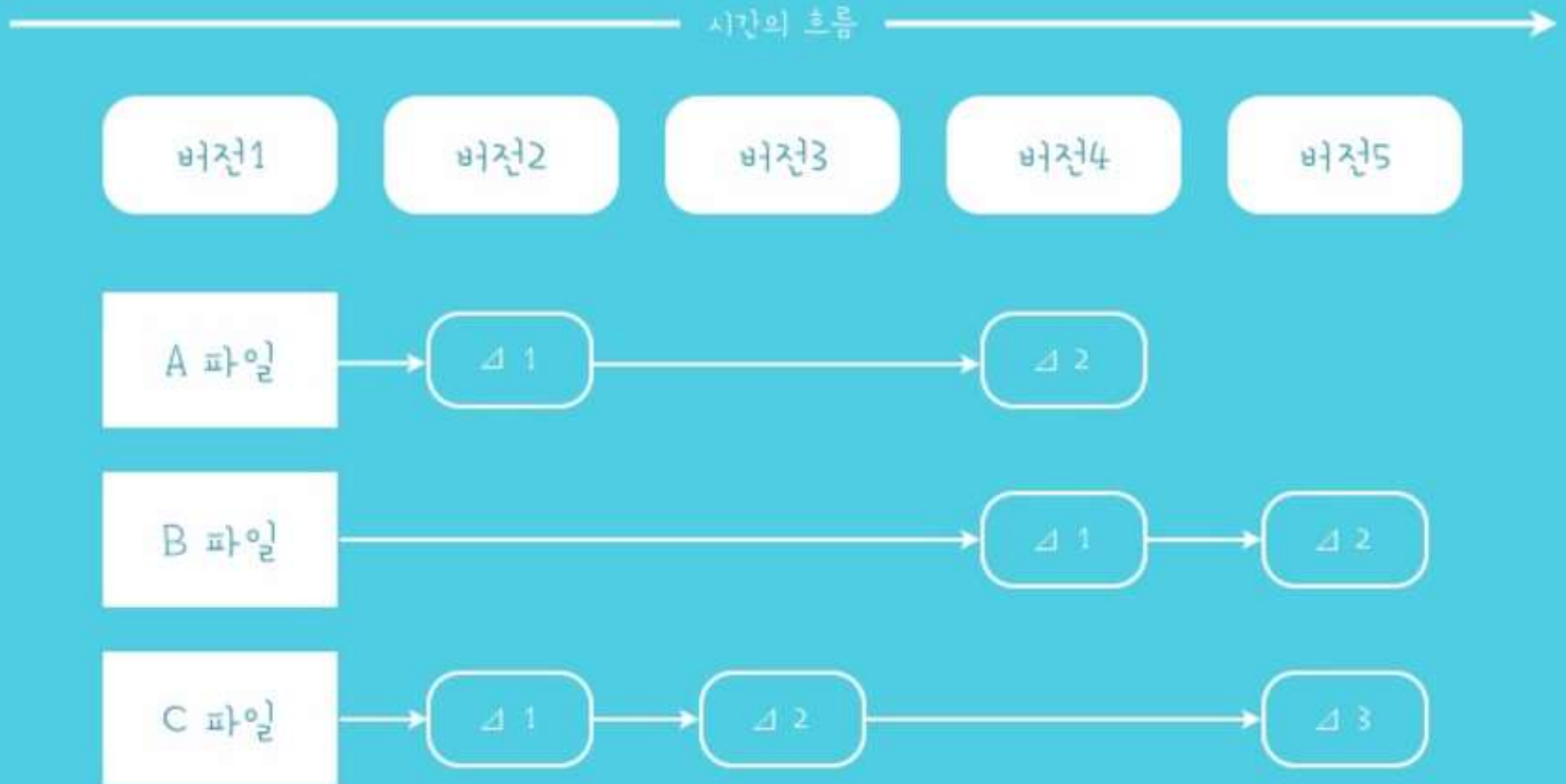
③ 

가장 중요한 차이점이자 git만의 특징이 남았습니다!

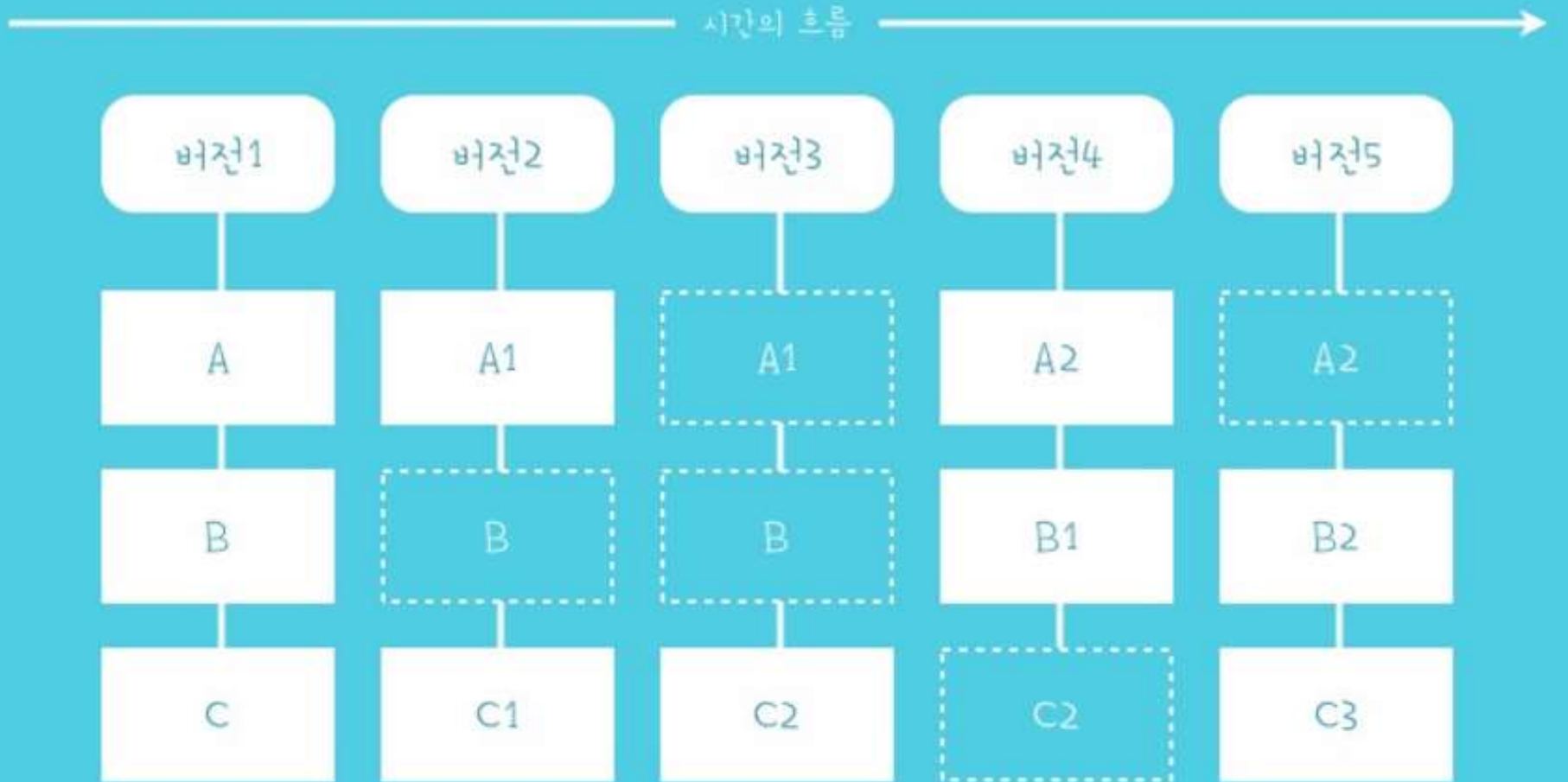
# 스냅샷 snapshot

git은 각각의 버전을 스냅샷으로 저장합니다.

svn은 파일의 변화(차이점)를 저장합니다.

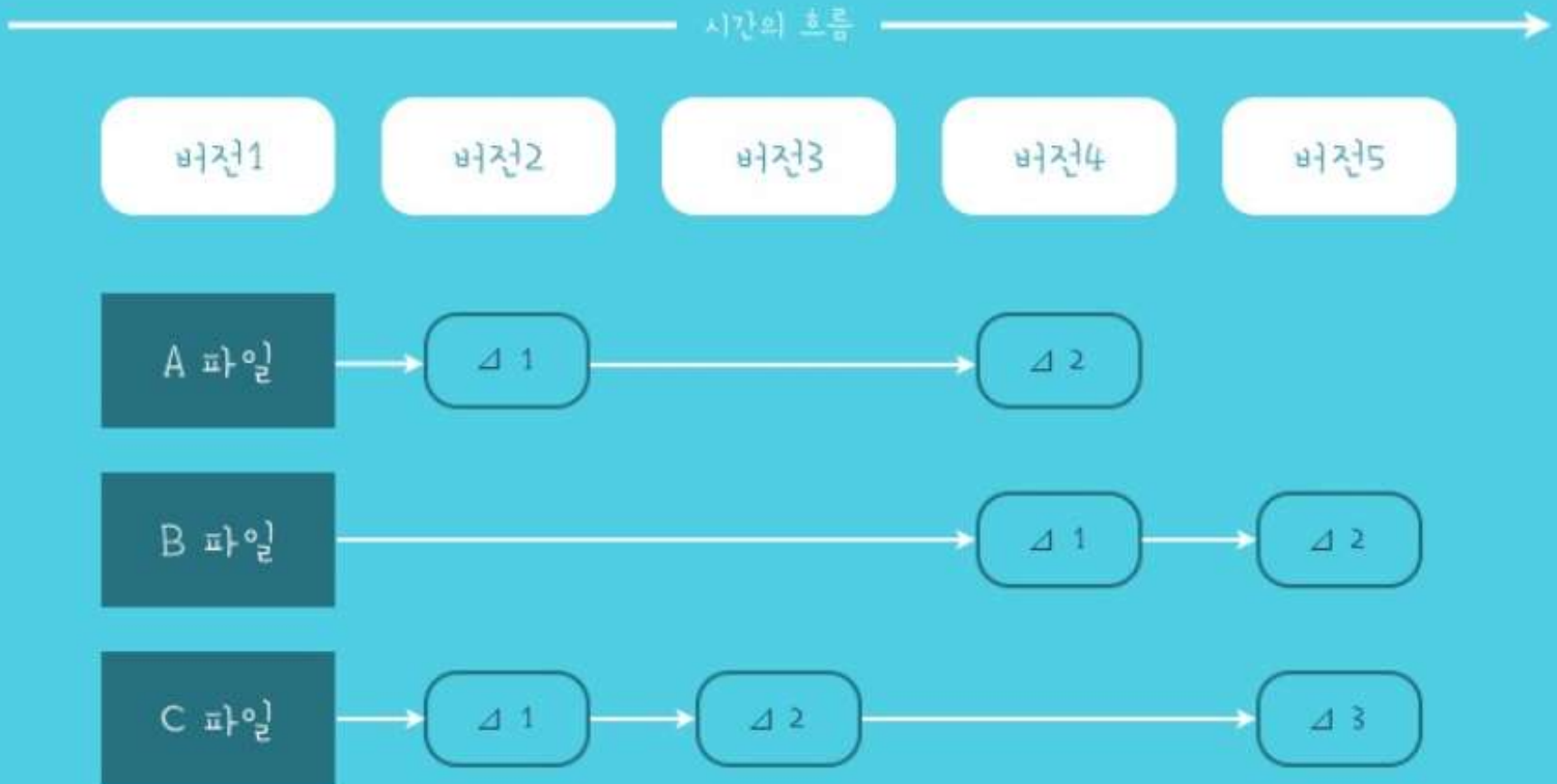


git은 그 순간의 스냅샷으로 저장합니다.



svn에서 버전5의 파일들을 가져오겠습니다.

기초가 되는 파일과 함께 모든 변경 내역을 서버로부터 내려받습니다.

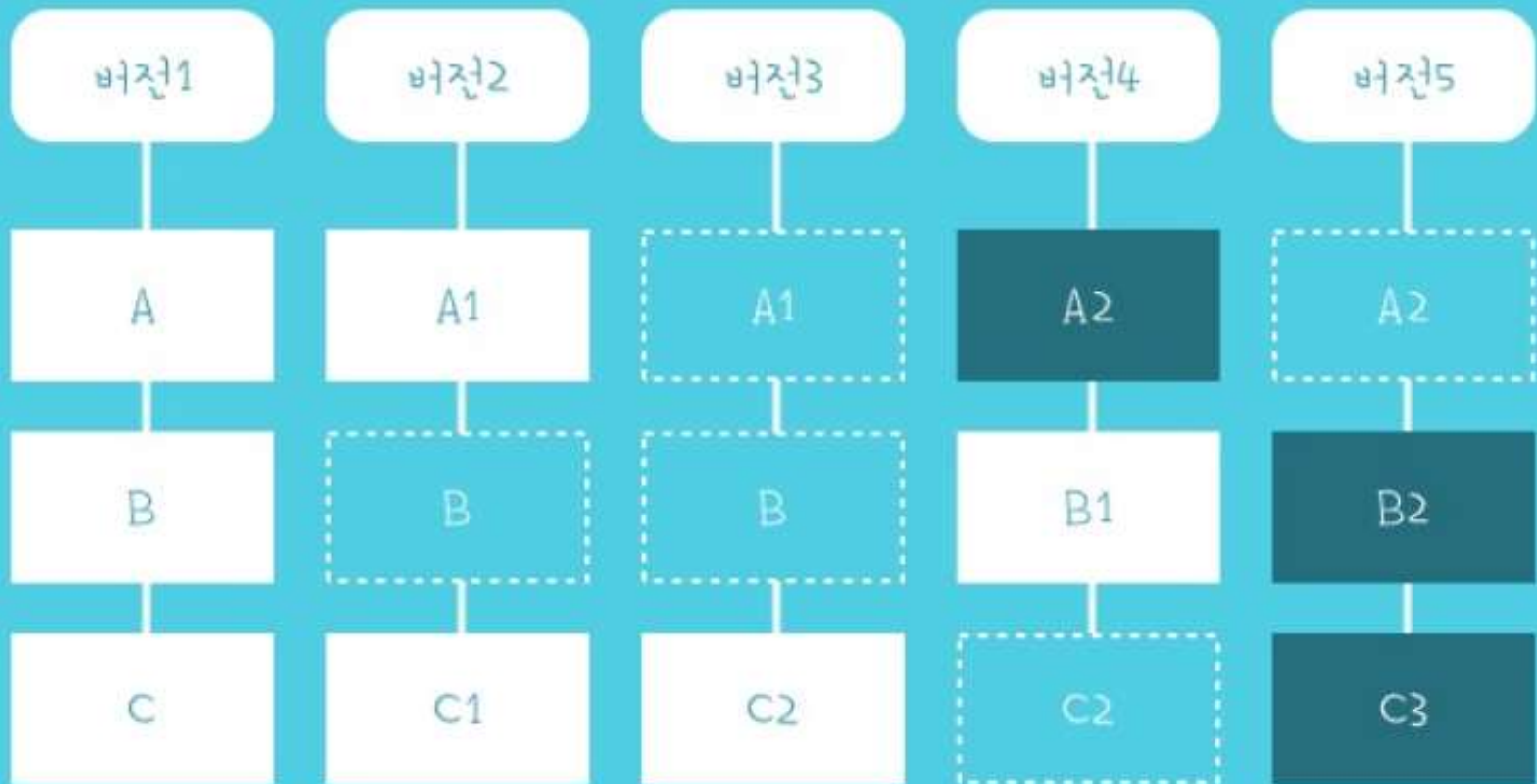




# git에서 버전5의 파일들을 가져오겠습니다.

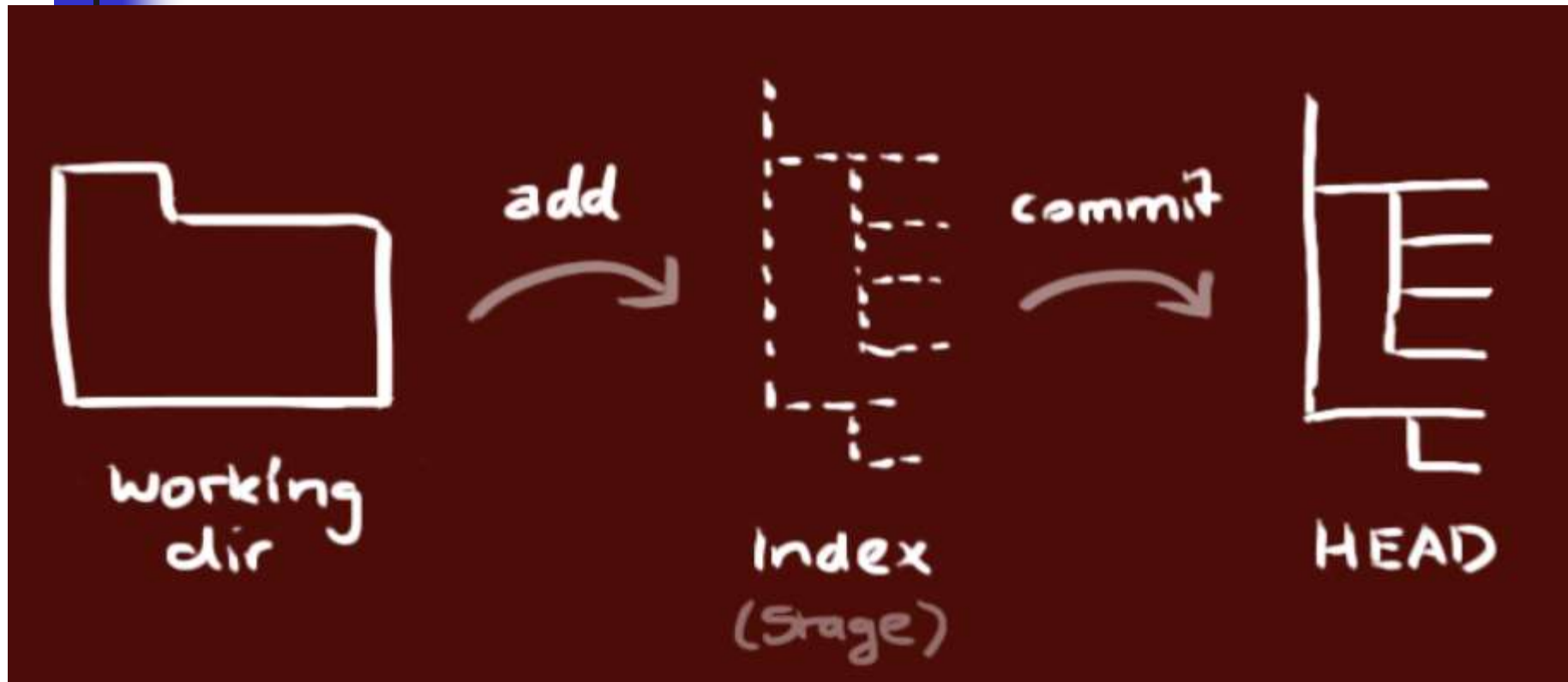
가장 가까운 스냅샷들만으로 특정 버전을 빠르게 만들어낼 수 있습니다.  
게다가 네트워크를 거치지 않습니다.

시간의 흐름





# 작업의 흐름



- 로컬 저장소는 git이 관리하는 세 그루의 나무로 구성
  - 첫번째 나무인 **작업 디렉토리(Working directory)**는 실제 파일들로 이루어져있고,
  - 두번째 나무인 **인덱스(Index)**는 준비 영역(staging area)의 역할을 하며,
  - 마지막 나무인 **HEAD**는 최종 확정본(commit)을 나타냄



# GIT

---

- 추가와 확정(commit)

- 변경된 파일은 아래 명령어로 (인덱스에) 추가할 수 있다.

git **add** <파일 이름>

git add \*

이것이 바로 git의 기본 작업 흐름에서 첫 단계에 해당됨.

- 하지만 실제로 변경 내용을 확정하려면 아래 명령을 내려야 함

git **commit** -m "이번 확정본에 대한 설명"

이제 변경된 파일이 **HEAD**에 반영됐다.

하지만, 원격 저장소에는 아직 반영이 안 됐다.

- 변경 내용 발행(push)하기

- 현재의 변경 내용은 아직 로컬 저장소의 HEAD 안에 머물고 있다.

이제 이 변경 내용을 원격 서버로 올려보자.

git **push** origin master



# GIT

---

- 갱신과 병합(merge)

- 로컬 저장소를 원격 저장소에 맞춰 갱신하려면  
아래 명령을 실행

git pull

이렇게 하면 원격 저장소의 변경 내용이 로컬 작업 디렉토리에 *받아지고(fetch)*, *병합(merge)*된다.

pull하기 전에 **commit**먼저 해야 함



# GIT

---

- 로컬 변경 내용 되돌리기

- 실수로 무언가 잘못된 경우,  
아래 명령으로 로컬의 변경 내용을 되돌릴 수 있다.

`git checkout -- <파일 이름>`

위 명령은 로컬의 변경 내용을 변경 전 상태(HEAD)로 되돌려줌.

다만, 이미 인덱스에 추가된 변경 내용과  
새로 생성한 파일은 그대로 남는다.

- 만약, 로컬에 있는 모든 변경 내용과 확정본을 포기하려면,  
아래 명령으로 원격 저장소의 최신 이력을 가져오고,  
로컬 master 가지가 저 이력을 가리키도록 할 수 있다.

`git fetch origin`

`git reset --hard origin/master`



# 이클립스와 GIT 연동

---

[Pull requests](#) [Issues](#) [Gist](#)

github 가입하기

<https://github.com/>

## Create a new repository

A repository contains all the files for your project, including the revision history.

Owner



now4ever ▾

/

Repository name



Great repository names are short and memorable. Need inspiration? How about **expert-carnival**.

Description (optional)



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.



Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾

Add a license: **None** ▾



Create repository

# 이클립스와 GIT 연동

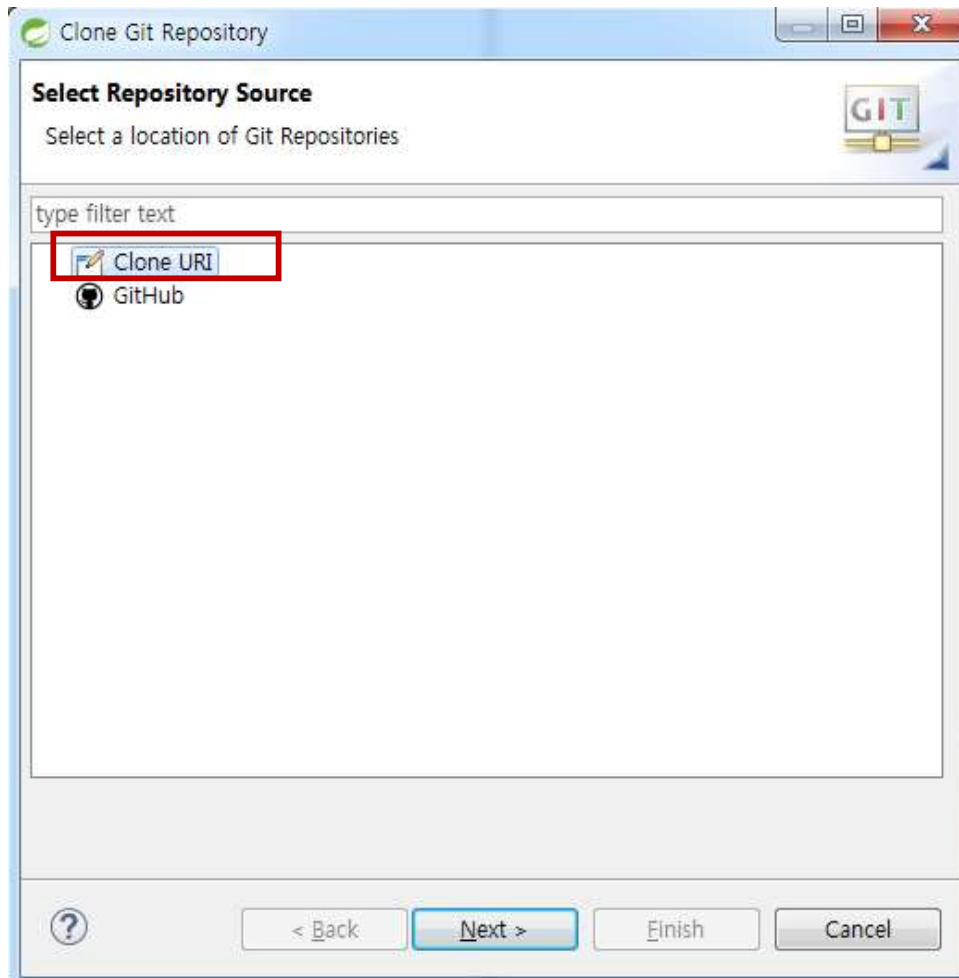
- 1. github 가입하기
  - <https://github.com/>
- 2. github에 repository 등록하기
  - https 경로로 된 URL 주소로 이클립스와 연동하여 사용할 수 있다.
- 3. STS 의 경우 Git 이 설치되어있기 때문에 별도로 설치할 필요는 없으며 없을 경우 Egit을 마켓에서 설치한다.
- 4. perspective 에서 Git 을 선택
- 5. git 에 들어가면 3개의 항목이 있다. 이중에 Clone a Git repository 를 선택

Select one of the following to add a repository to this view:

-  [Add an existing local Git repository](#)
-  [Clone a Git repository](#)
-  [Create a new local Git repository](#)

# 이클립스와 GIT 연동

- 6. 2개의 메뉴 중 Clone URI 를 선택





# 이클립스와 GIT 연동

- 7. 개인의 깃허브 URI 와 아이디 패스워드를 입력
  - port는 입력하지 않아도 됨

Clone Git Repository

**Source Git Repository**  
Enter the location of the source repository.

Location

URI:  Local File...

Host:

Repository path:

Connection

Protocol:

Port:

Authentication

User:

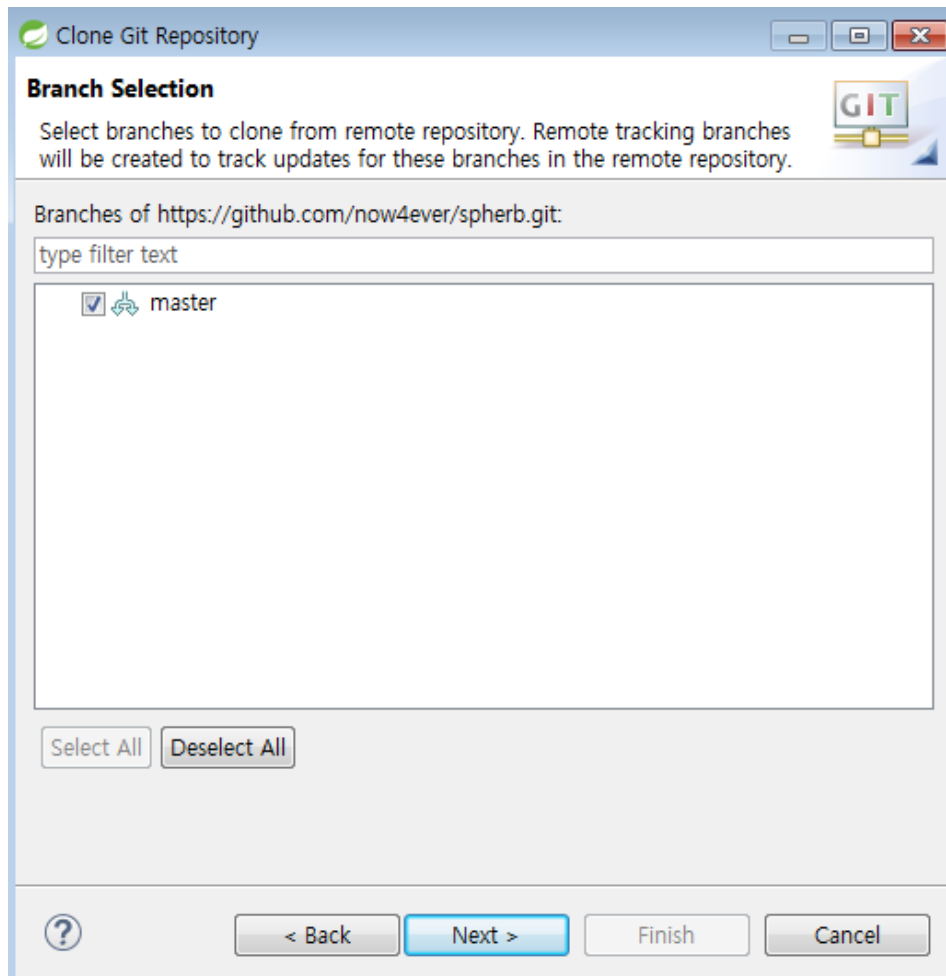
Password:

☒ Store in Secure Store

? < Back Next > Finish Cancel

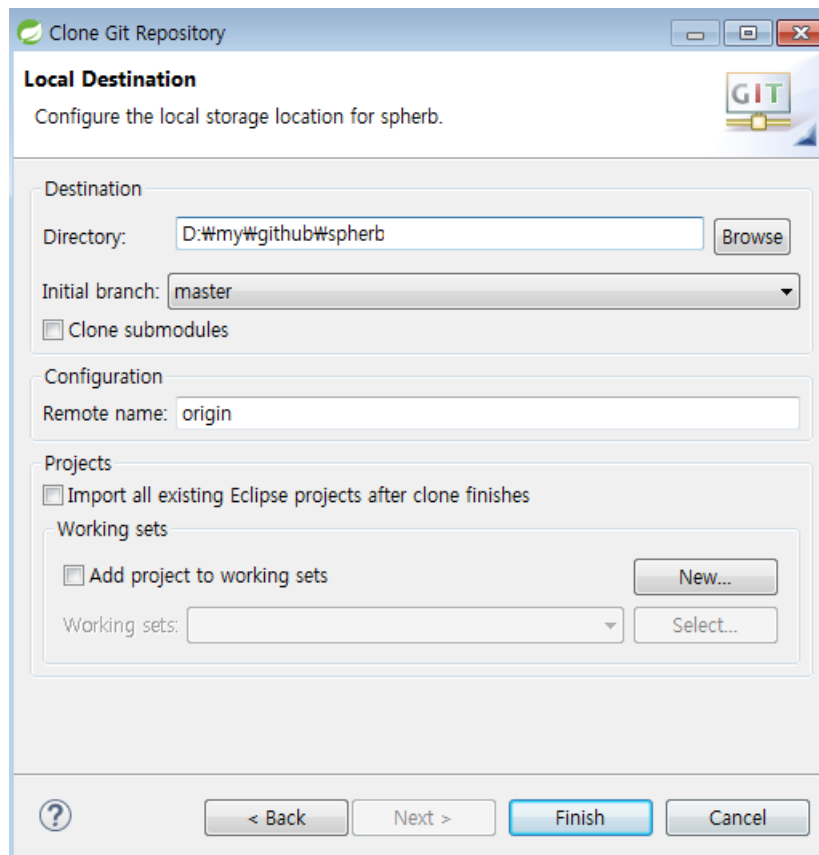
# 이클립스와 GIT 연동

## ■ 8. 기본 Branch인 master를 선택



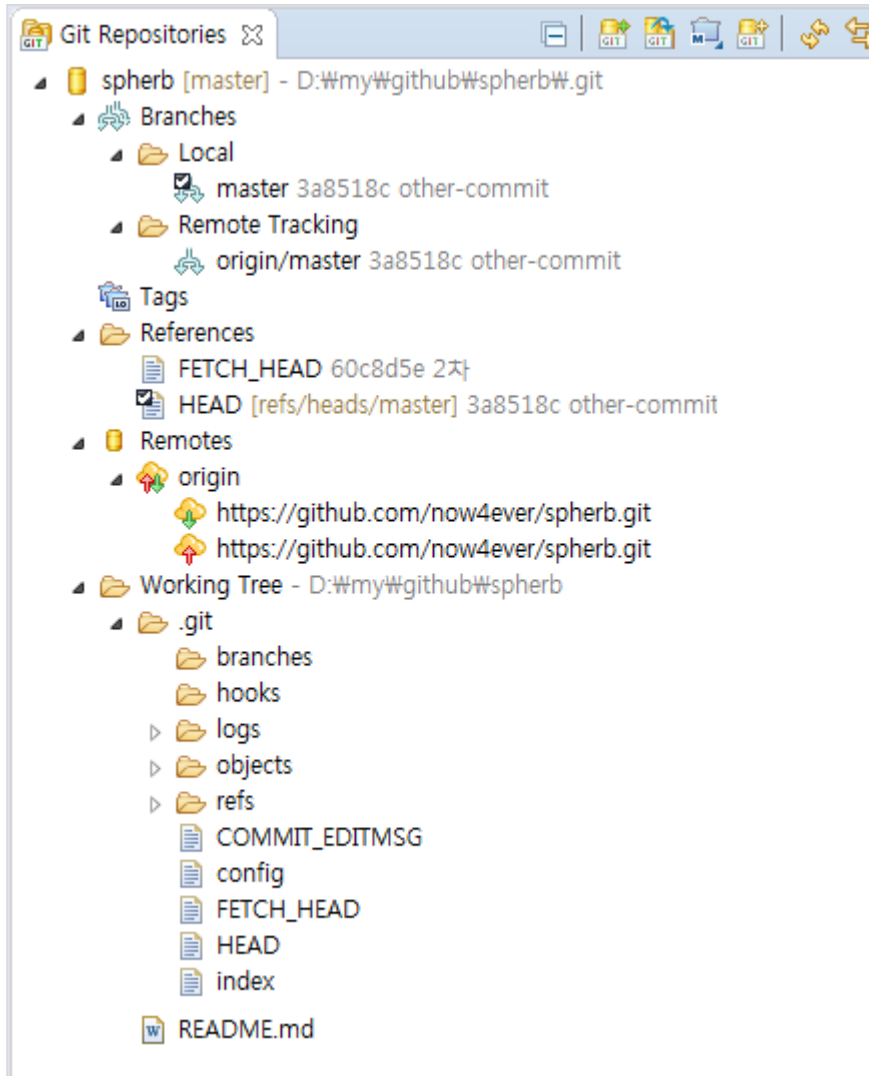
# 이클립스와 GIT 연동

- 9. 깃이 연동될 디렉토리를 선택해준다
  - 보통 SVN과는 다르게 Local의 파일에 commit을 한 뒤에 git 에 push하여 다시 한번 update를 하게 됨
  - local 의 저장소 지정라고 보면 됨



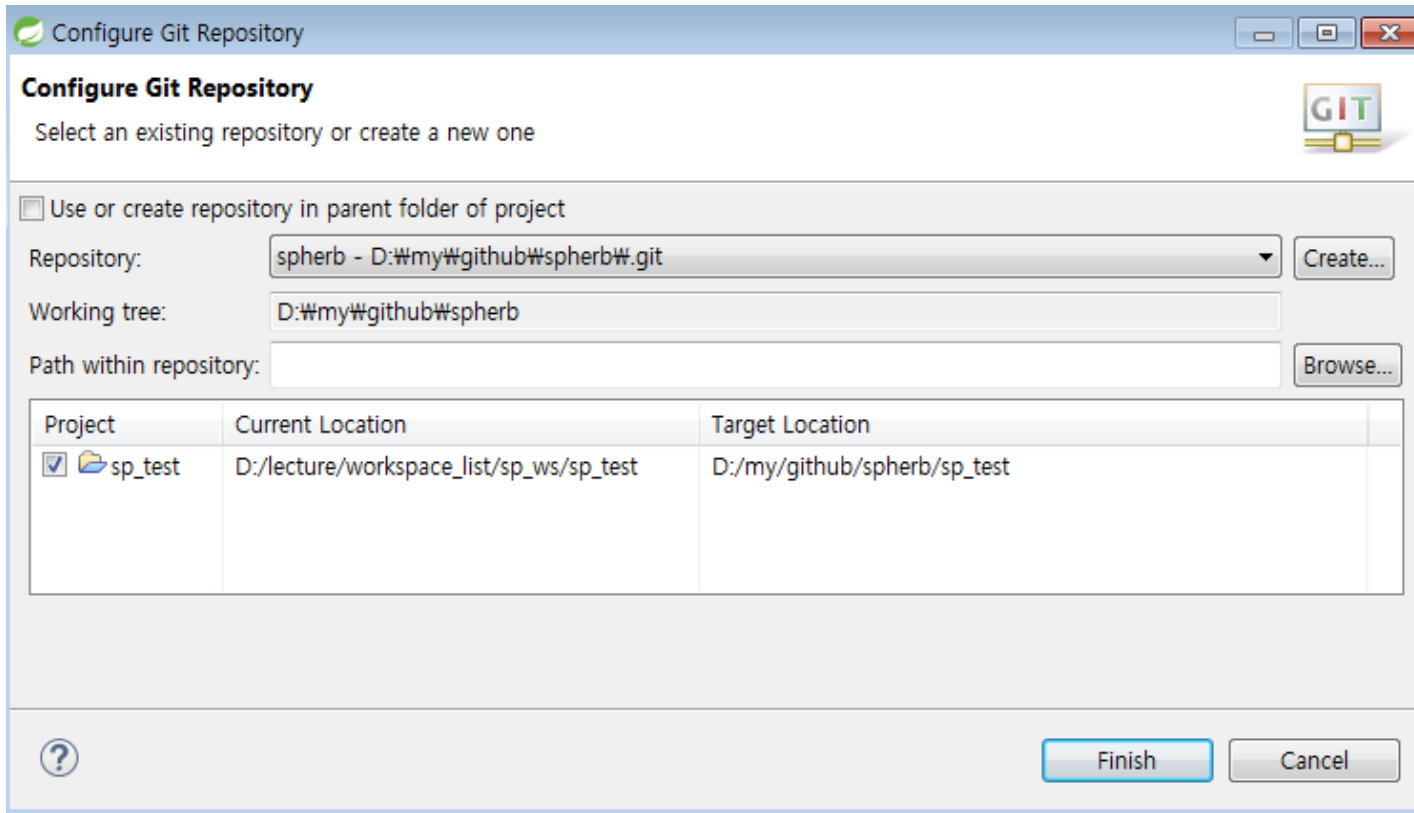
# 이클립스와 GIT 연동

- 10. 깃이 정상적으로 연결이 되면 아래처럼 세팅이 됨



# 이클립스와 GIT 연동

- 11. 개발된 소스 또는 개발 시작하려는 소스를 업로드한다
  - 업로드 하려는 프로젝트에 오른쪽 클릭을 하여 Team -> Share Project 를 선택
- 12. Local 저장소를 선택하고 Finish를 누르면 Local의 저장소와 함께 동기화가 됨
  - 실제로 깃에 올라간 것은 아니며 깃에 올리기 전 Local 의 저장소에 Commit을 한 뒤에 Update가 가능함



## 이클립스와 GIT 연동

- 13. 프로젝트에서 마우스 오른쪽 => team - Add to Index를 선택하면 프로젝트 파일에 있던 물음표가 사라짐
  - 로컬 저장소에 커밋하기 전에 커밋할 파일들을 스테이지 영역(index)에 추가하는 것
- 14. 그런 후 Commit 을 해준다.
  - Commit 을 실행하면 commit 할 대상과 함께 메시지를 남길 수 있다.
  - Commit 을 누르면 Local 저장소와 싱크만 맞추며 Commit and Push를 선택하면 Github에 소스가 업로드 된다.

Git Staging

Filter files

> **spherb [master]**

Unstaged Changes (0)

Staged Changes (1)

ReBoardController.java - springherb/src/main/java/com/herb/app/reboard/controller

Commit Message

1차 커밋

Author:

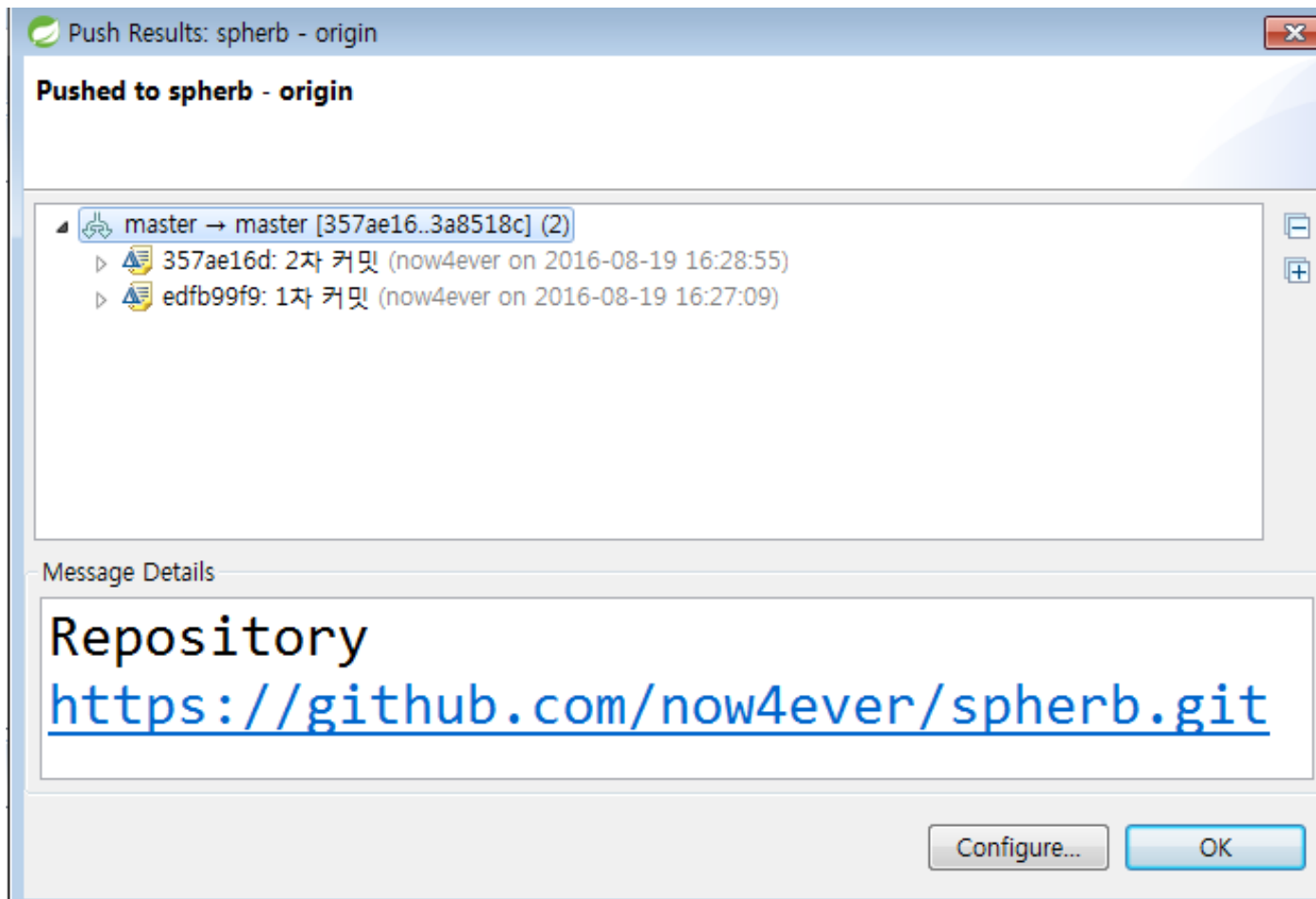
Committer:

Commit and Push...

Commit

# 이클립스와 GIT 연동

- 15. github에 push 할 때 정보를 확인 한 후 OK 버튼을 누르면 실제 github에 올라가게 된다.





# 이클립스와 GIT 연동

- 16. 깃허브 사이트에서 올라간 소스를 확인 할 수 있다.

The screenshot shows the GitHub interface for the repository 'now4ever / spherb'. The browser address bar displays 'https://github.com/now4ever/spherb'. The repository name 'now4ever / spherb' is prominently displayed. Below the name, there are buttons for 'Watch', 'Star', and 'Fork', each with a count of 0. A navigation bar includes links for 'Code', 'Issues' (0), 'Pull requests' (0), 'Wiki', 'Pulse', 'Graphs', and 'Settings'. A message states 'No description or website provided. — Edit'. A summary bar shows '7 commits', '1 branch', '0 releases', and '1 contributor'. Below this, there are buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and a green 'Clone or download' button. The commit history is listed below, showing 'now4ever 2차 커밋' as the latest commit (357ae16, 7 minutes ago), followed by 'springherb' (2차 커밋, 7 minutes ago) and 'README.md' (Initial commit, a day ago).

← → ↻ [GitHub, Inc. \[US\]](#) <https://github.com/now4ever/spherb>

This repository Search Pull requests Issues Gist + ▾

now4ever / **spherb** Watch ▾ 0 Star 0 Fork 0

<> Code Issues 0 Pull requests 0 Wiki Pulse Graphs Settings

No description or website provided. — Edit

7 commits 1 branch 0 releases 1 contributor

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

now4ever 2차 커밋 Latest commit 357ae16 7 minutes ago

springherb 2차 커밋 7 minutes ago

README.md Initial commit a day ago



# 이클립스와 GIT 연동

---



- git 레포지토리에서 프로젝트 다운받기
  - package Explore에서 Import -> Git -> Projects from Git을 선택하고 next 클릭
  - Clone URI를 선택하고 다시 next를 누르고 아래와 같이 채워 주고, next

## Select

Import one or more projects from a Git Repository.

Select an import wizard:

type filter text

- ▶ General
- ▶ EJB
- ▲ Git
  -  Projects from Git
  -  Repositories from GitHub
- ▶ Install
- ▶ Java EE
- ▶ Maven
- ▶ Oomph
- ▶ Plug-in Development
- ▶ Remote Systems
- ▶ Run/Debug
- ▶ Tasks
- ▶ Team
- ▶ Web



< Back

Next >

Finish

Cancel




## Import Projects from Git

### Select Repository Source

Select a location of Git Repositories



type filter text

-  Existing local repository
-  Clone URI
-  GitHub

Import Projects from Git

Source Git Repository

Enter the location of the source repository.

Location

URI:

Host:

Repository path:

Connection

Protocol:

Port:

Authentication

User:

Password:

☒ Store in Secure Store

?

< Back

Next >

Finish

Import Projects from Git

Branch Selection

Select branches to clone from remote repository. Remote tracking branches will be created to track updates for these branches in the remote repository.

Branches of https://github.com/now4ever/spherb.git:

☒ master

Select All

Deselect All

?

< Back

Next >

Finish

Cancel

Import Projects from Git

**Local Destination**

Configure the local storage location for spherb.

Destination

Directory:

Initial branch:

☐ Clone submodules

Configuration

Remote name:

Local Destination 화면에서는 다운 받을 프로젝트의 로컬 레포지토리의 디렉토리를 설정해준다.

Cloning from <https://github.com/now4ever/spherb.git>

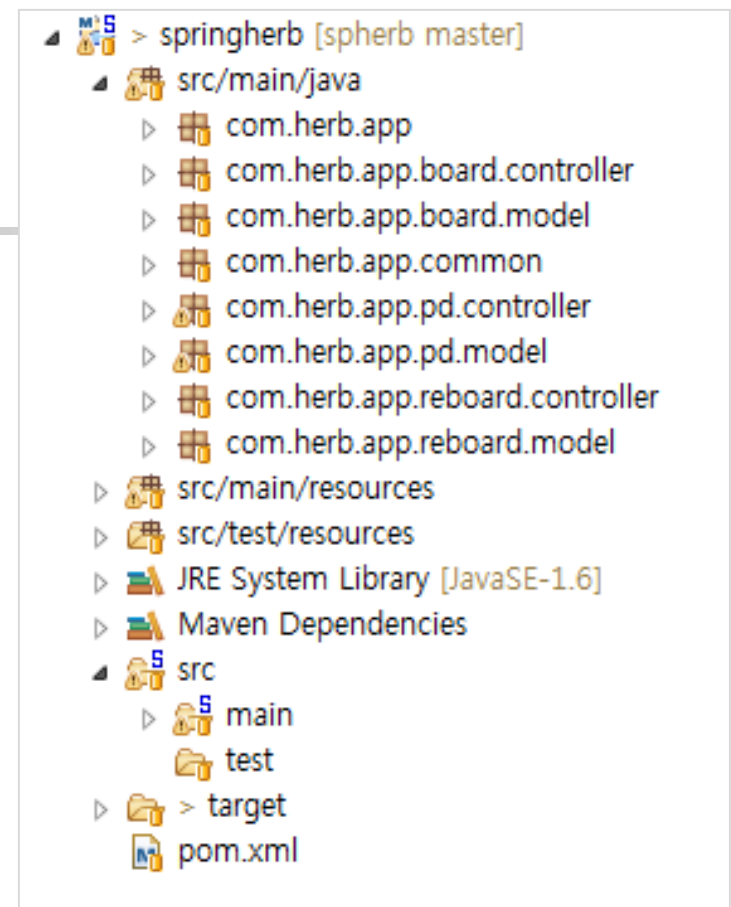
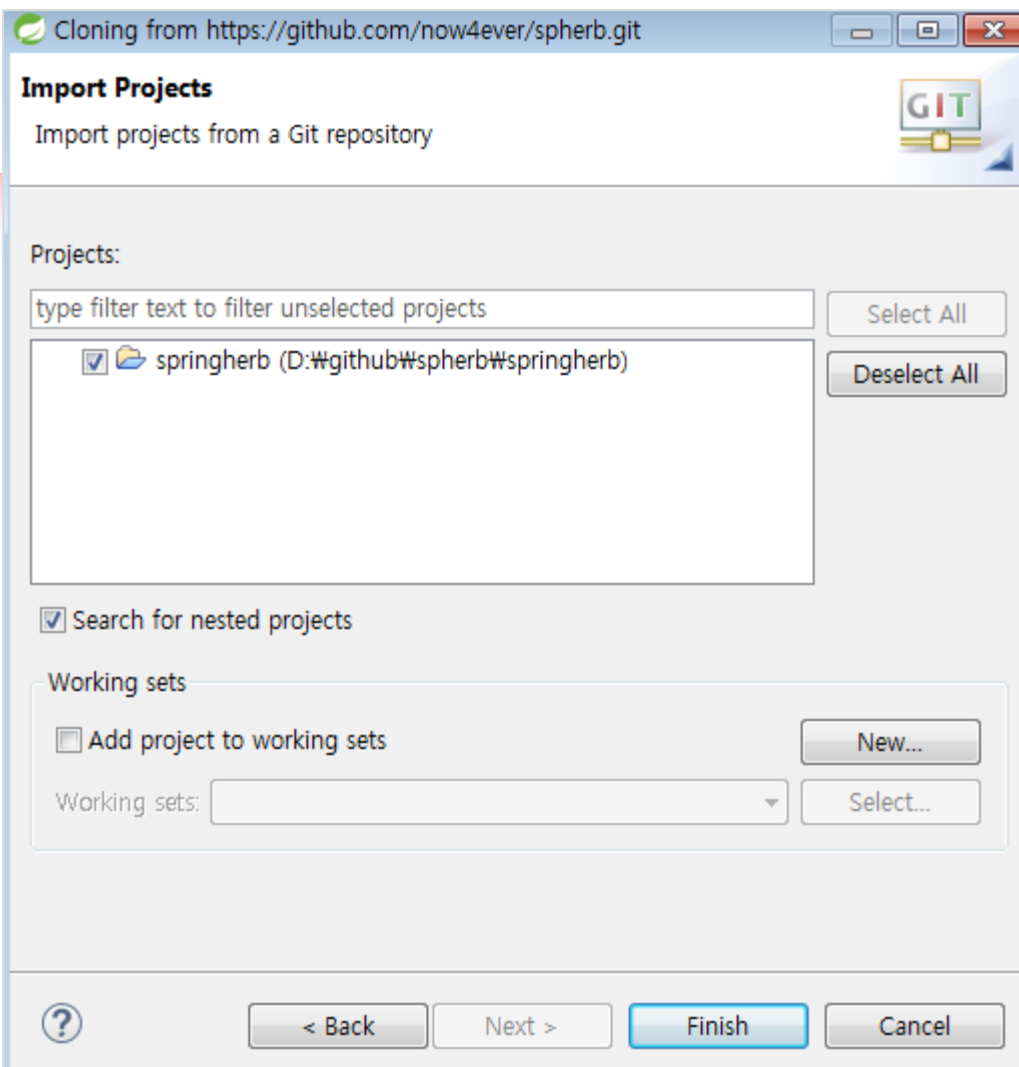
**Select a wizard to use for importing projects**

Depending on the wizard, you may select a directory to determine the wizard's scope

Wizard for project import

- ☒ Import existing Eclipse projects
- ☐ Import using the New Project wizard
- ☐ Import as general project

Working Tree -



프로젝트가 임포트 됨.

# 이클립스와 GIT 연동

- 프로젝트를 수정하기 전에 먼저 git 레포지토리에서 최신 프로젝트를 pull 한다
- 프로젝트에서 오른쪽 마우스 - team - pull
  - Pull 하기전에 commit 먼저 한다.
  - => conflict된 부분을 표시해줌

```
28 public void showInfo() {  
29 <<<<<<< HEAD  
30     System.out.println("Person클래스의 showInfo()메서드!!");  
31     System.out.println("test");  
32 =====  
33     System.out.println("Person클래스의 showInfo()메서드...!!~~");  
34 >>>>>>> branch 'master' of https://github.com/now4ever/mytest1.git  
35 }  
36  
37 }
```

- 충돌난 부분을 제외하고는 정상적으로 pull됨
- 충돌난 부분을 수정한 후 commit and push 한다



# 팀 프로젝트시 GIT 사용

---

- [1] 팀프로젝트를 할 레포지토리 선택
- [2] Settings 메뉴 - Collaborators 탭
  - 추가할 팀원 아이디나 이메일 검색
  - Add Collaborator
- [3] 초대 받은 사람이 메일 확인하기
- [4] Copy invite link 의 주소
  - (<https://github.com/now4ever/gittest/invitations>)로 이동 해서 수락
- [5] 메일을 확인하고, 수락한 경우 해당 레포지토리에 대한 권한을 받을 수 있다



Options

Collaborators

Branches

Webhooks

Integrations &amp; services

Deploy keys

## Collaborators

Push access to the repository



Awaiting now4evern's response

Copy invite link ▾

Cancel invite

Search by username, full name or email address

You'll only be able to find a GitHub user by the name you enter instead.

## Copy invite link

now4evern's shareable invitation link.

<https://github.com/now4ever/gittest/invitations>

Enter their username

Add collaborator



@now4ever has invited you to collaborate on the  
**now4ever/gittest** repository

You can [accept](#) or [decline](#) this invitation. You can also head over to  
<https://github.com/now4ever/gittest> to check out the repository or visit [@now4ever](#) to  
learn a bit more about them.

[View invitation](#)

**Note:** This invitation was intended for [now4ever@nate.com](#). If you were not expecting this  
invitation, you can ignore the invitation, you can ignore the invitation, you can ignore the invitation,  
block them or report abuse.

[leeunkyu](#) / [gitlecture](#) kosta



[leeunkyu](#) invited you to collaborate

[Accept invitation](#)

[Decline](#)

Is this user sending spam or malicious content? You can [block @leeunkyu](#).