

분산 버전 관리 도구: Git

Basic

Hyunchan, Park

<http://oslab.chonbuk.ac.kr>

Division of Computer Science and Engineering

Chonbuk National University

학습 내용

- Git
 - 실습 1
- GitHub
 - 실습 2

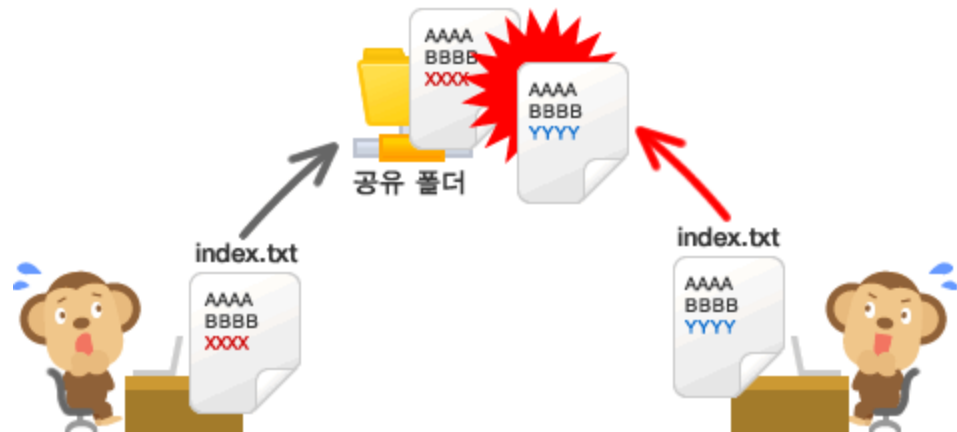


Git



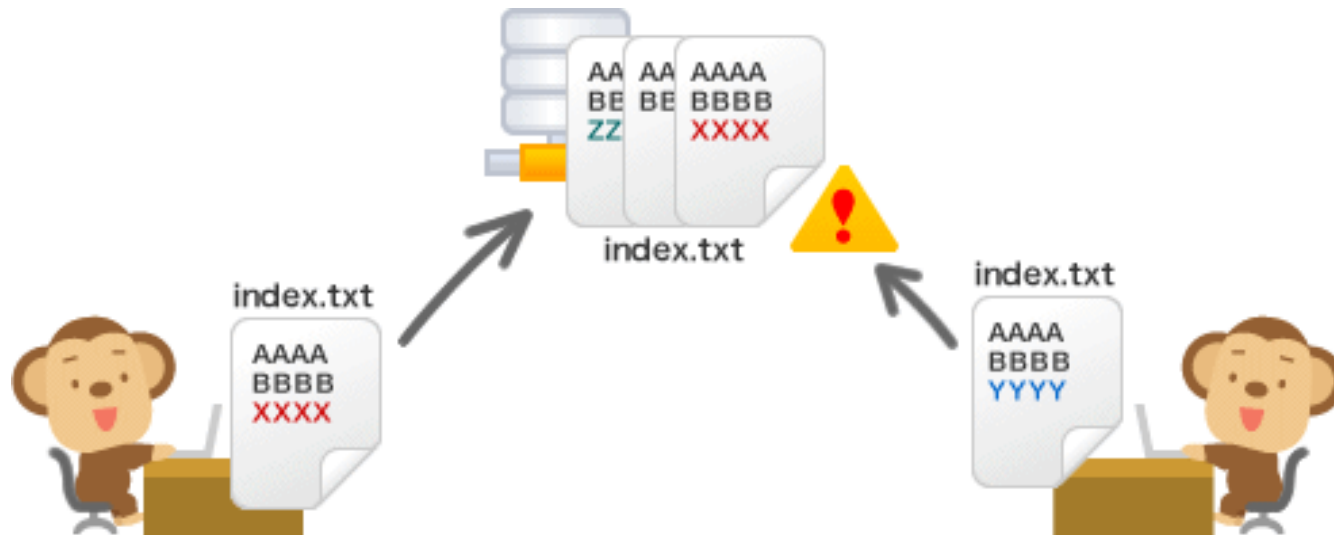
버전 관리

Name
120525_문서_업데이트.txt
120604_문서.txt
120605_문서_수정판.txt
120605_문서_수정판2.txt
120605_문서_최신 복사.txt
120605_문서_최신.txt
120605_문서.txt
1200602_문서.txt
문서_회의용.txt



* 출처: https://backlogtool.com/git-guide/kr/intro/intro1_3.html

버전 관리



버전 관리 시스템 (Version Control System)

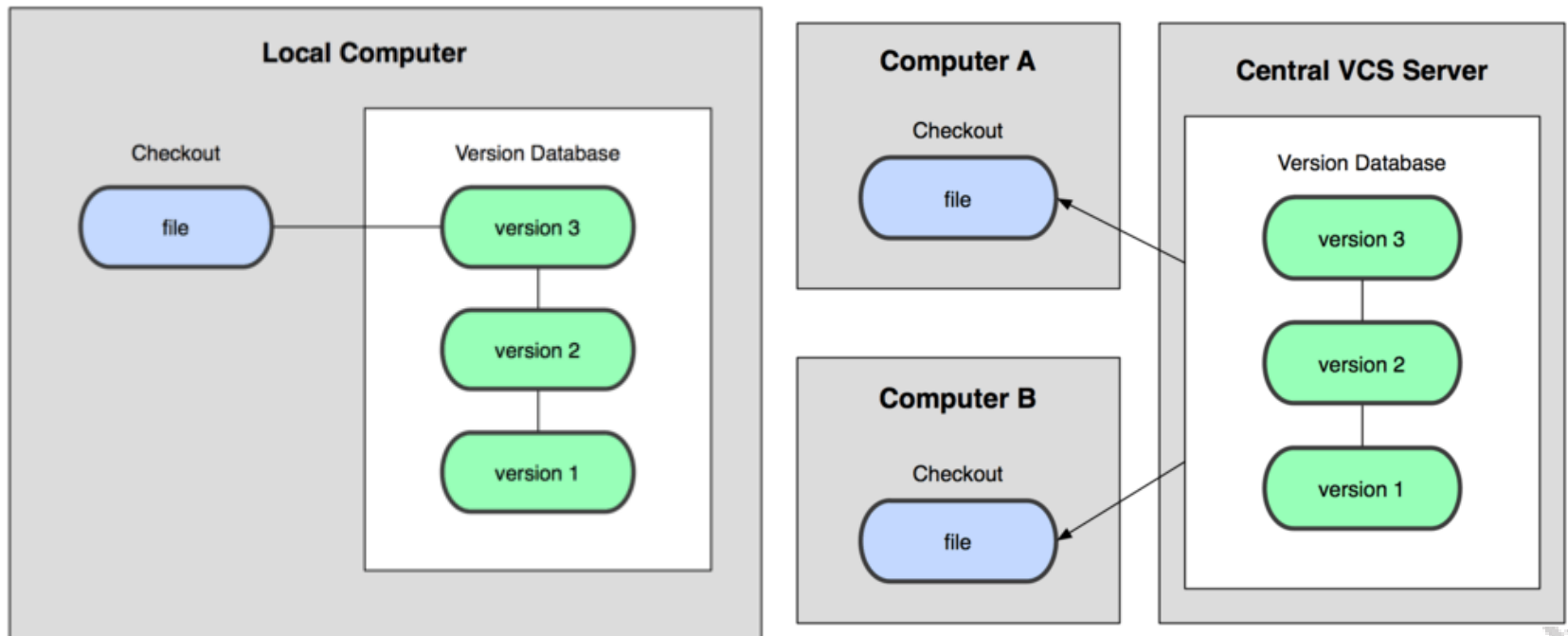
- 동일한 정보에 대한 여러 버전을 관리하는 것
 - 파일의 변화를 시간에 따라 기록하여 과거 특정 시점의 버전을 다시 불러올 수 있는 시스템
- 왜 사용하는가?
 - 백업
 - 잘못되었을 때 복구를 돕기 위하여
 - 버전 관리
 - 프로젝트 진행 중 과거의 어떤 시점으로 돌아갈 수 있게 하기 위하여
 - 소스 코드의 변경 사항을 추적하기 위하여
 - 코드의 특정 부분이 왜 그렇게 쓰여 졌는지 의미를 추적하기 위하여

버전 관리 시스템 (Version Control System)

- 왜 사용하는가? (cont'd)
 - 협업 도구
 - 여러 사람이 같은 프로젝트에 참여할 경우, 각자가 수정한 부분을 팀원 전체가 동기화하는 과정을 자동화하기 위하여
 - 소스 코드에서 누가 수정했는지 추적하기 위하여
 - 대규모 수정 작업을 더욱 안전하게 진행하기 위하여
 - 개발 편의성
 - 가지내기(Branch)로 프로젝트에 영향을 최소화 하면서 새로운 부분을 개발하기 위하여
 - 접붙이기(Merge)로 검증이 끝난 후 새로이 개발된 부분을 본류(trunk)에 합치기 위하여
 - “많은 오픈 소스 프로젝트에서 어떠한 형태로든 버전 관리를 사용하고 있으므로”

로컬 vs 중앙집중형 VCS

- 로컬: 1인 개발 시 사용, 팀 단위 사용 시 부적합
- 중앙집중형: 간단한 구조, single point of failure (SPOF)



버전 관리 시스템 (Version Control System)

- 유사 시스템
 - 소스 코드 관리(Source Code Management, SCM)
 - SW 버전 관리(Software Version Management)
 - SW 형상 관리(Software Configuration Management)
 - 리비전 관리 시스템 (Revision Control System)
- 널리 쓰이는 SW
 - SVN (Subversion)
 - Mercurial
 - Git

버전 관리 시스템 (Version Control System)

로컬 전용	무료 / 오픈 소스	SCCS (1972), RCS (1982)
	유료 / 상용	PVCS (1985), QVCS (1991)
클라이언트/ 서버	무료 / 오픈 소스	CVS (1986, 1990 in C), CVSNT (1998), QVCS Enterprise (1998), Subversion (2000)
	유료 / 상용	Software Change Manager (1970s), Panvalet (1970s), Endeavor (1980s), DSEE (1984), Synergy (1990), ClearCase (1992), CMVC (1994), Visual SourceSafe (1994), Perforce (1995), StarTeam (1995), Integrity (2001), Surround SCM (2002), AccuRev SCM (2002), SourceAnywhere (2003), SourceGear Vault (2003), Team Foundation Server (2005), Rational Team Concert (2008)
분산	무료 / 오픈 소스	GNU arch (2001), Darcs (2002), DVCVS (2002), ArX (2003), Monotone (2003), SVK (2003), Codeville (2005), Bazaar (2005), Git (2005), Mercurial (2005), Fossil (2007), Veracity (2010)
	유료 / 상용	TeamWare (1990s?), Code Co-op (1997), BitKeeper (1998), Plastic SCM (2006)

* 출처: <http://blog.gaerae.com/2015/03/comparison-of-revision-control-software.html>



Git: the stupid content tracker

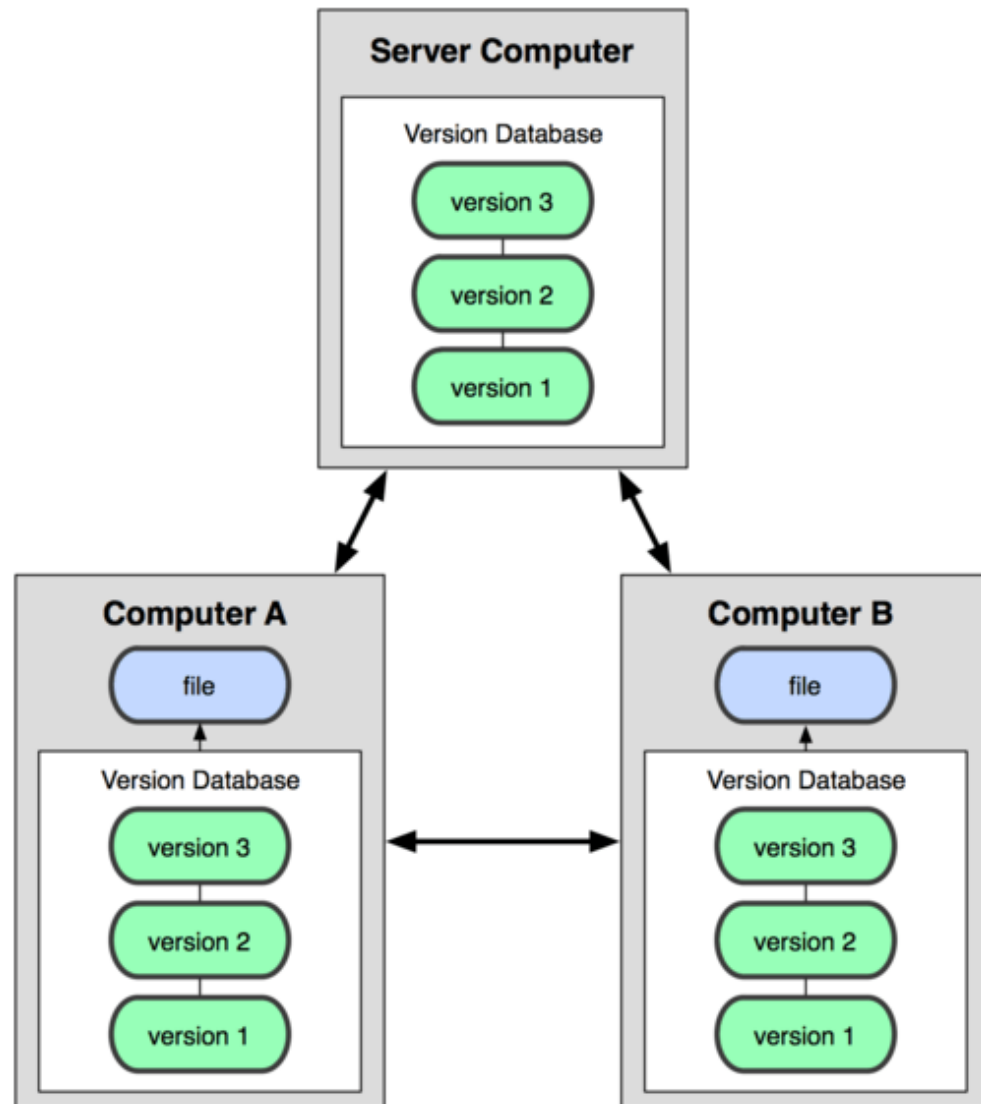
- 분산 버전관리 시스템
 - Distributed Version Control System (DVCS)
 - 여러 사람이 협업하는 환경에서 문서변경사항을 관리하는 시스템
- 이름
 - 영국 속어로 바보
 - Global Information Tracker?
- 특징
 - Free and Open source
 - Easy to learn
 - Tiny footprint
 - Lighting fast performance

git¹

1.(영·속어) 쓸모없는 놈 2.바보 자식

미국 [git]  영국 [git] 

Git: DVCS

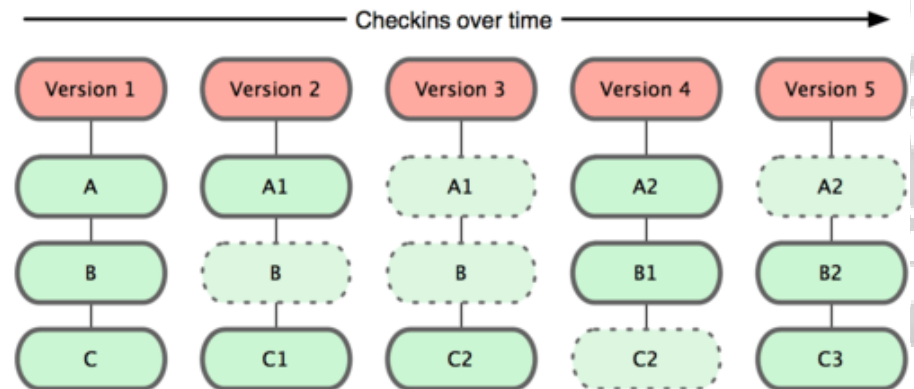
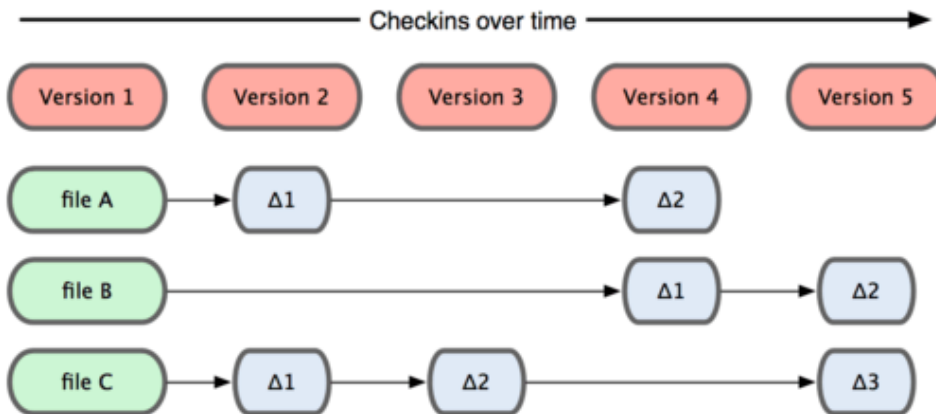


Git: brief history

- 2005년 리눅스 개발 커뮤니티에 의해 개발
- 기존 방식
 - ~2002: 단순 압축 (스냅샷)과 패치를 통해 버전 관리
 - ~2005: BitKeeper 사용. 유료 전환되며 Git 개발
- 설계 목표 (vs. BitKeeper)
 - 빠른 속도
 - 단순한 구조
 - 비선형적인 개발 (수천 개의 동시 다발적인 브랜치)
 - 완벽한 분산
 - 리눅스 커널 같은 대형 프로젝트에도 유용할 것
(속도나 데이터 크기 면)

Git 특징 1

- 단순성: 변화된 부분만을 기록하는 것이 아니라, 전체를 버전 별로 보존
 - 델타 방식 vs. 스냅샷 방식
 - Git은 스냅샷 방식으로, 언제나 데이터를 추가해나가는 방식

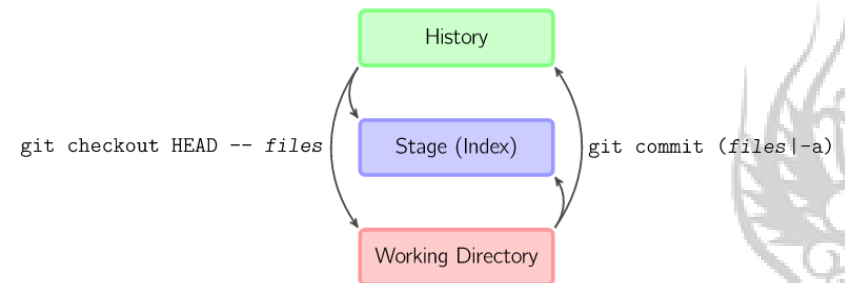
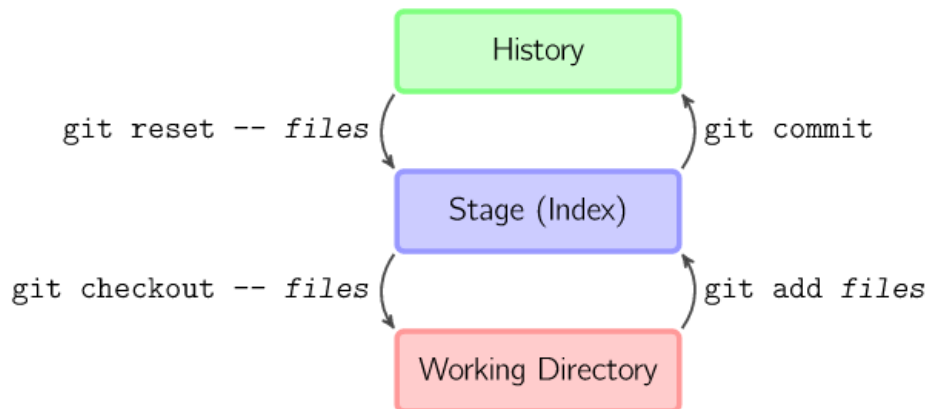


Git 특징 2

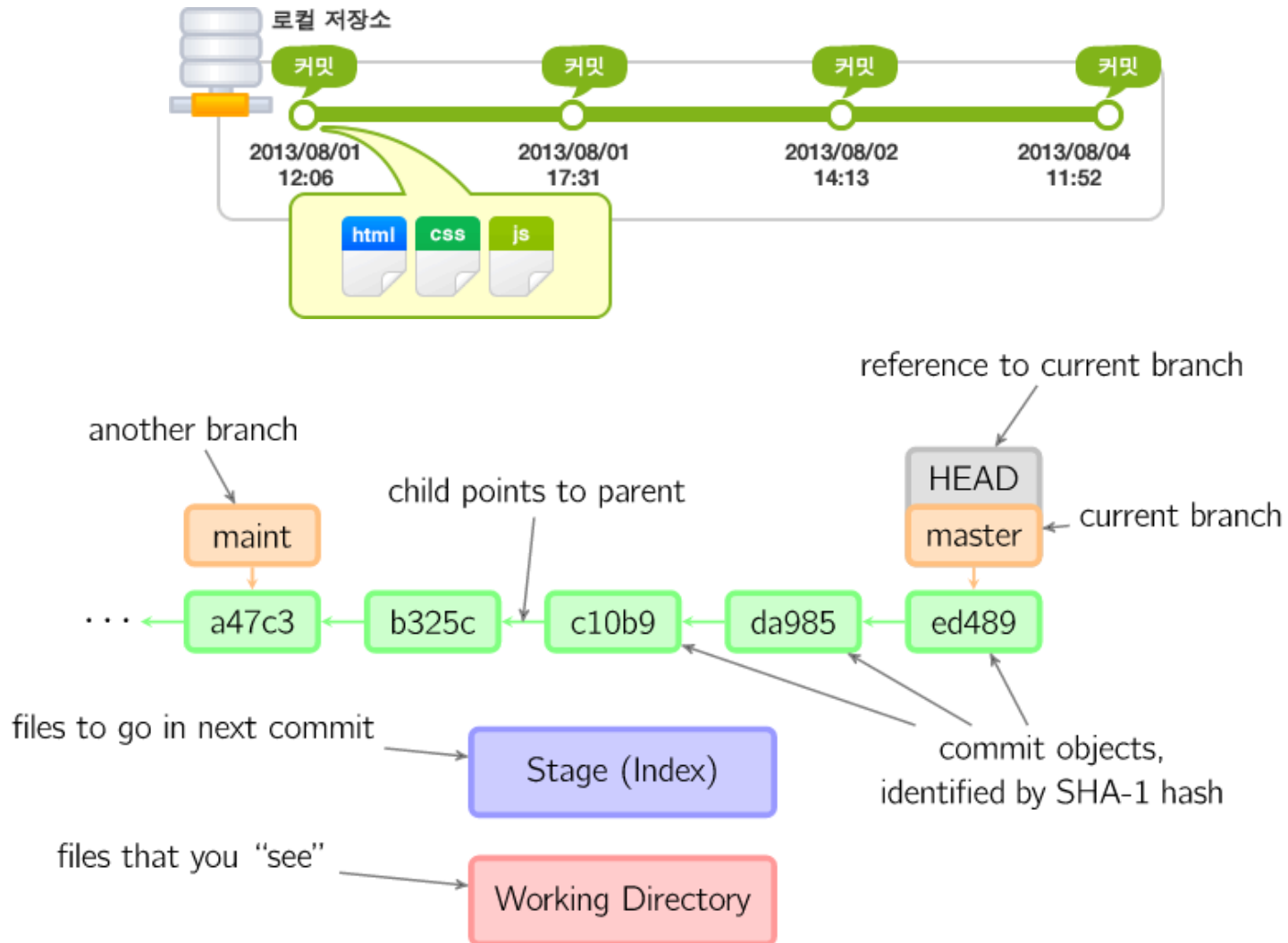
- 빠른 속도: 로컬에서 명령 실행
 - 분산된 형태로 관리되기 때문에 가능함
- 무결성: 변화된 파일에 대한 체크섬 관리
 - 파일, 디렉토리에 대한 SHA-1 Checksum을 이용해 분산 구조에서 무결성을 보장함
 - Git은 모든 데이터를 checksum hash 형태로 관리함

Git의 3가지 영역 (혹은 상태)

- 작업 폴더(Working Directory)
 - 사용자가 변경하는 실제 파일이 들어가는 폴더
- 스테이지(Stage, Index)
 - 변경사항을 관리할 파일들의 리스트
 - 작업 폴더 중에서 커밋할 파일만을 모아둘 수 있음
- 변경이력(History, Git directory, repository)
 - 커밋(Commit)이라 불리는 변경사항 묶음과 커밋들의 연결관계

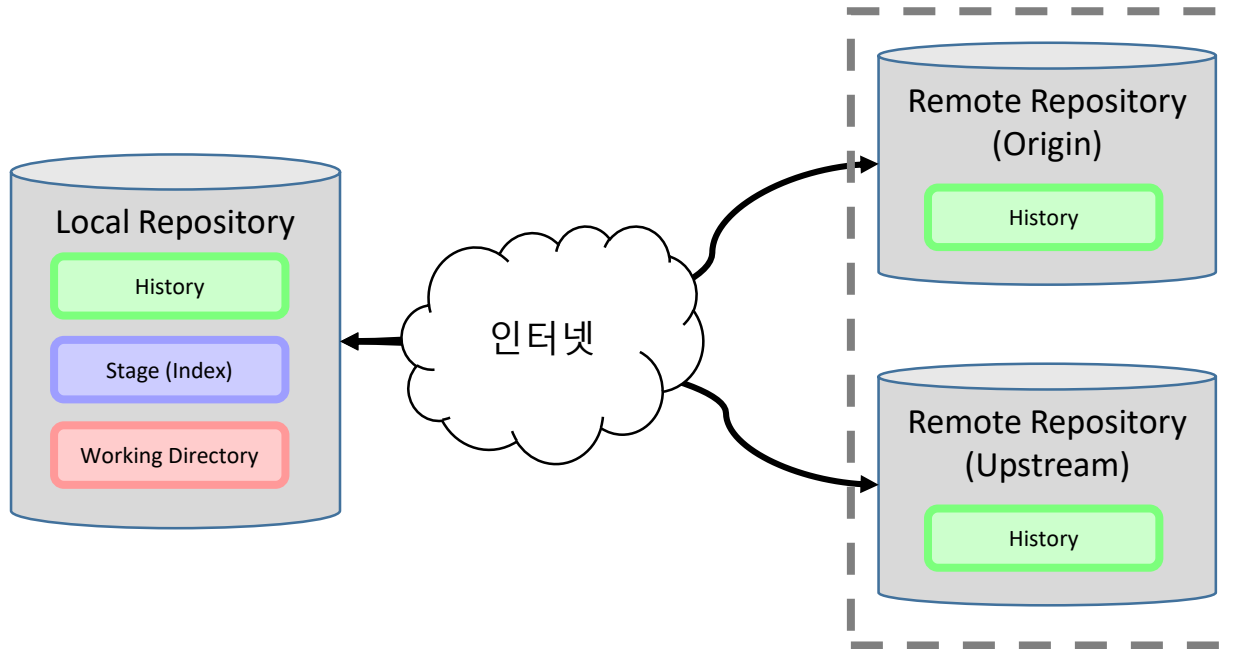


일반적인 git의 동작 상태



로컬저장소와 원격저장소

- 협업을 위해서는 원격저장소가 필수적
- 로컬저장소와 원격저장소 간에 이력을 주고받을 수 있음
- 원격저장소가 여러 개 일 수 있음



많이 사용되는 원격 저장소

- GitHub
- BitBucket
- GitLab

로컬 - 원격 저장소 간의 이동

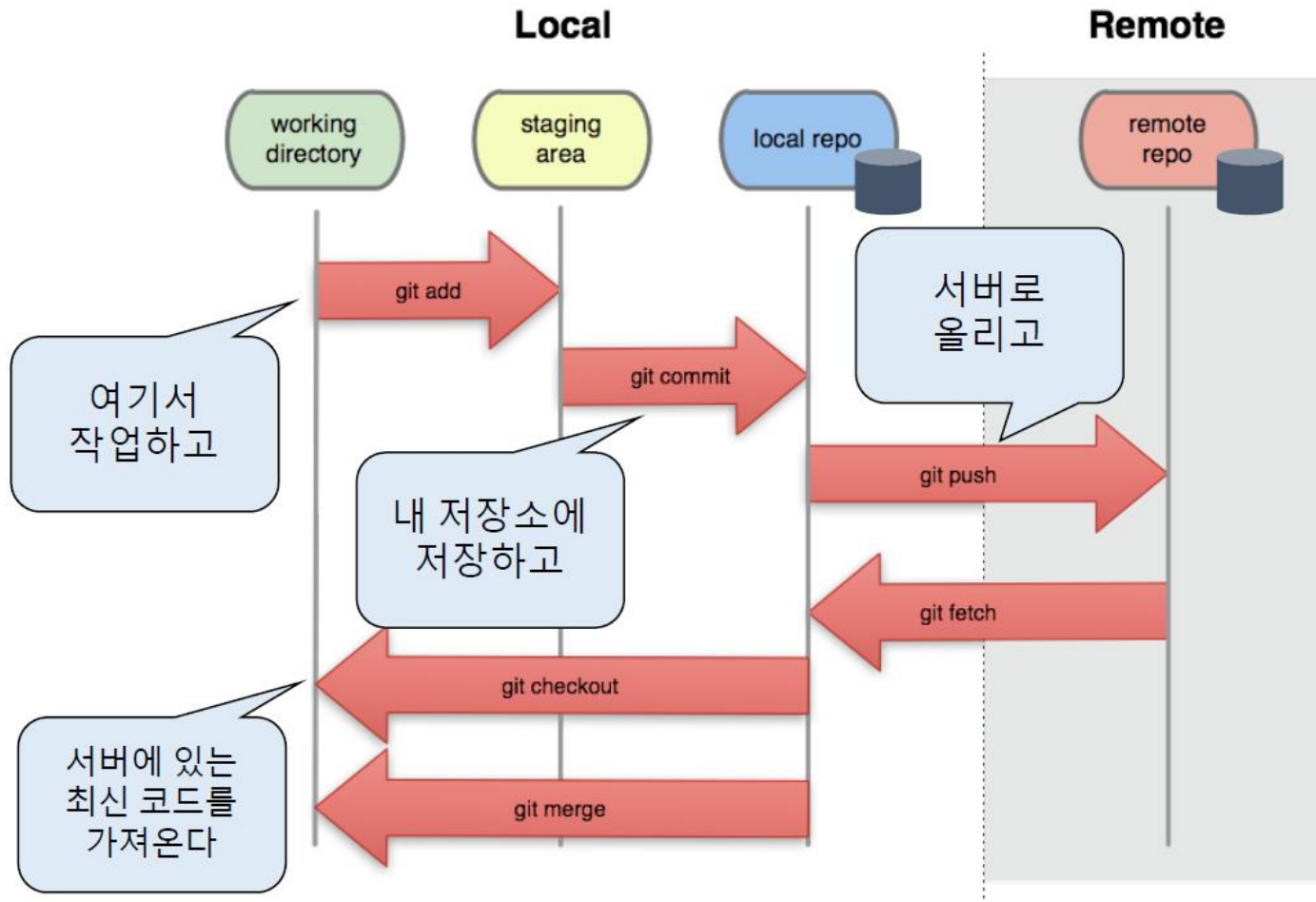


그림 출처 : <http://pismute.github.io/whygitisbetter/#everything-is-local>



실습 #1

Git 설치 및 시작



Git 설치

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



- 설치가 끝나면, 시작 메뉴 > 모든 프로그램 > Git > Git Bash를 실행

```
MINGW64:/c/Users/hcpark
g
hcpark@hcpark-PC MINGW64 ~
$ git --version
git version 2.10.1.windows.1
hcpark@hcpark-PC MINGW64 ~
$
```

Git 저장소 만들기

- \$ mkdir tutorial
- \$ cd tutorial
- \$ git init
 - 결과: Initialized empty Git repository in
c:\Users\yourname\Desktop/tutorial/.git/
- 로컬 저장소 완성
 - ls
 - ls -al

Git 설정

- 사용자명 및 e-mail 설정 (gmail 이용)
 - `$ git config --global user.name "<사용자명>"`
 - `$ git config --global user.email "<메일 주소>"`
- 색상 설정 (자동)
 - `$ git config --global color.ui auto`

Your first COMMIT

- 메모장 등을 이용해 test.c 작성, tutorial 폴더에 저장
 - Notepad test.c
 - vi 사용 가능
- \$ git status

```
hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ ls
test.c

hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ git status
On branch master

Initial commit

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

        test.c

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

hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$
```



Your first COMMIT

- \$ git add test.c

```
hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ git add test.c

hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   test.c

hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$
```

- \$ git commit -m "your comment"

```
hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ git commit -m "my first commit"
[master (root-commit) d7a71a5] my first commit
1 file changed, 7 insertions(+)
create mode 100644 test.c

hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ git status
On branch master
nothing to commit, working tree clean

hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$
```



저장소 이력 확인

- \$ git log

```
hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ git log
commit d7a71a5edfb5e0e2fea75032f6bad8ce374b886f
Author: hcpark <hcpark.class@gmail.com>
Date: Thu Mar 16 16:36:34 2017 +0900

    my first commit

hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$
```



새로운 commit 작성하기

```
hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ notepad test.c

hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   test.c

no changes added to commit (use "git add" and/or "git commit -a")

hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ git commit -m "second commit"
On branch master
Changes not staged for commit:
  modified:   test.c

no changes added to commit

hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ git add test.c

hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ git commit -m "second commit"
[master d9917de] second commit
1 file changed, 1 insertion(+)

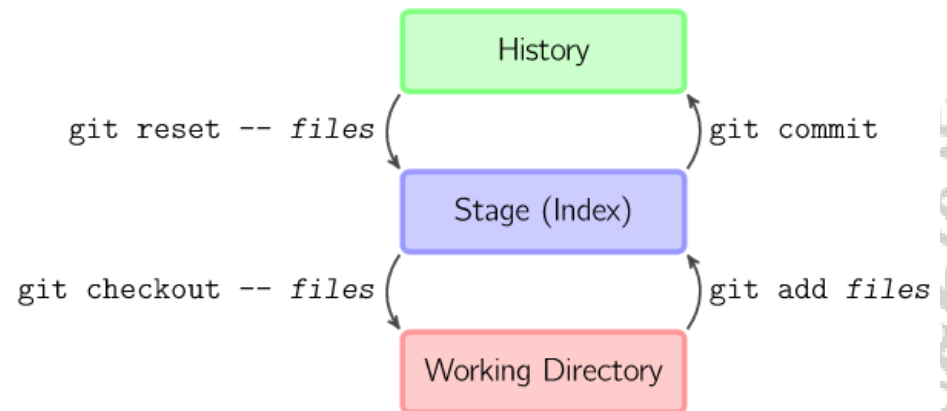
hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ git log
commit d9917de88f98a69a9cd5c557f3da0a8de1cb8b23
Author: hcpark <hcpark.class@gmail.com>
Date:   Thu Mar 16 16:39:40 2017 +0900

    second commit

commit d7a71a5edfb5e0e2fea75032f6bad8ce374b886f
Author: hcpark <hcpark.class@gmail.com>
Date:   Thu Mar 16 16:36:34 2017 +0900

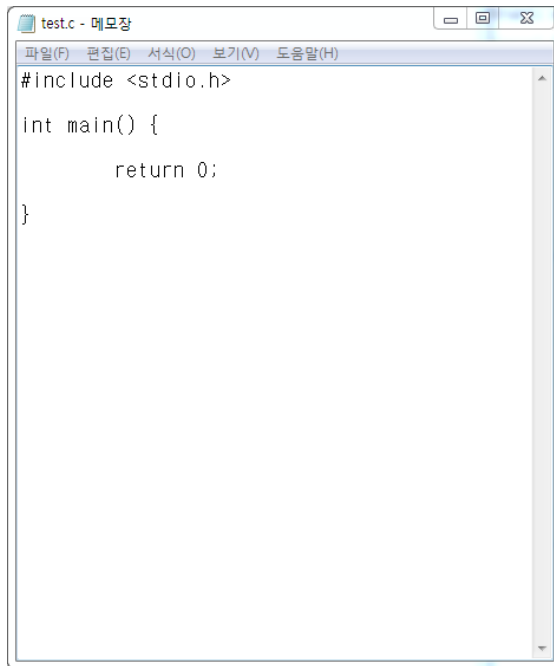
    my first commit

hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$
```



파일 되돌리기 1. working의 변화

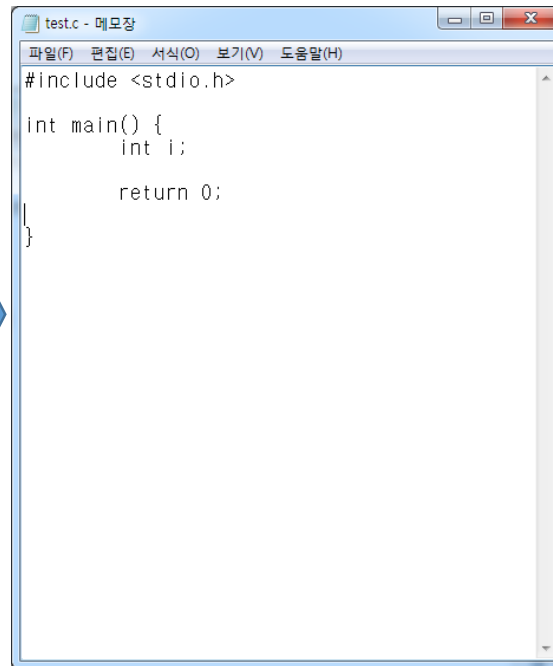
First commit



```
#include <stdio.h>

int main() {
    return 0;
}
```

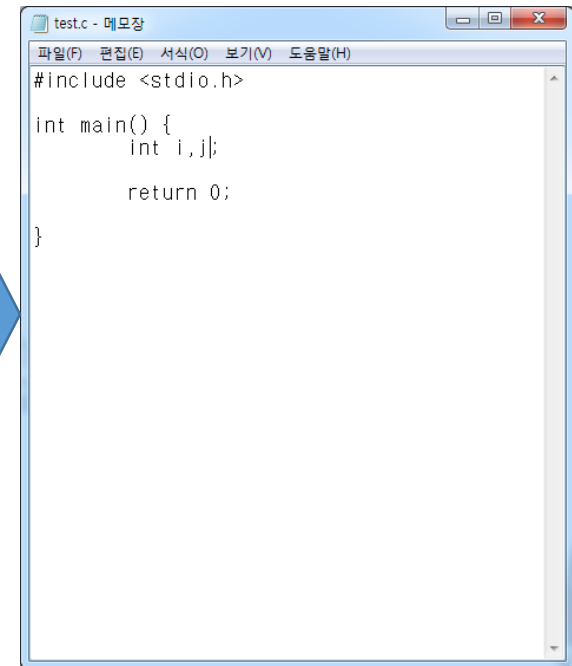
Second commit



```
#include <stdio.h>

int main() {
    int i;
    return 0;
}
```

Current status



```
#include <stdio.h>

int main() {
    int i, j;
    return 0;
}
```

파일 되돌리기 1. working의 변화

```
hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   test.c

no changes added to commit (use "git add" and/or "git commit -a")

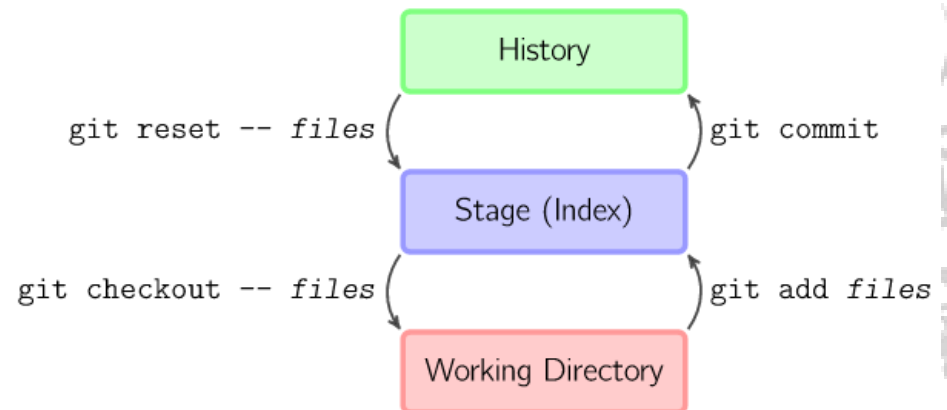
hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ git checkout -- test.c

hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ git status
On branch master
nothing to commit, working tree clean

hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ cat test.c
#include <stdio.h>

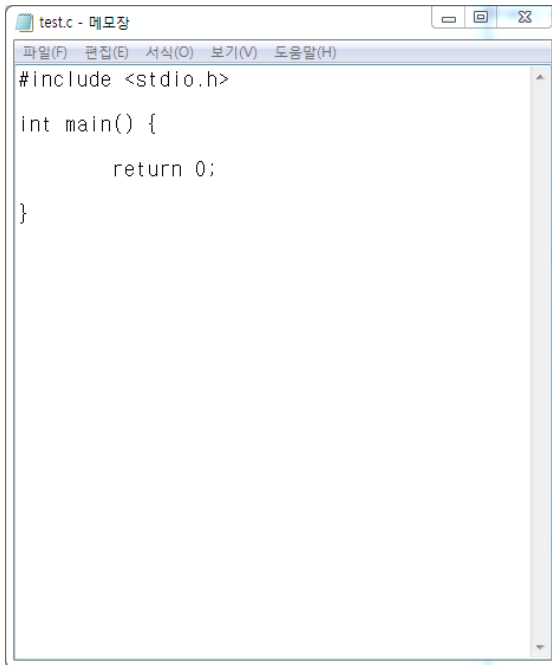
int main() {
    int i;

    return 0;
}
hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$
```



파일 되돌리기 2. stage의 변화

First commit

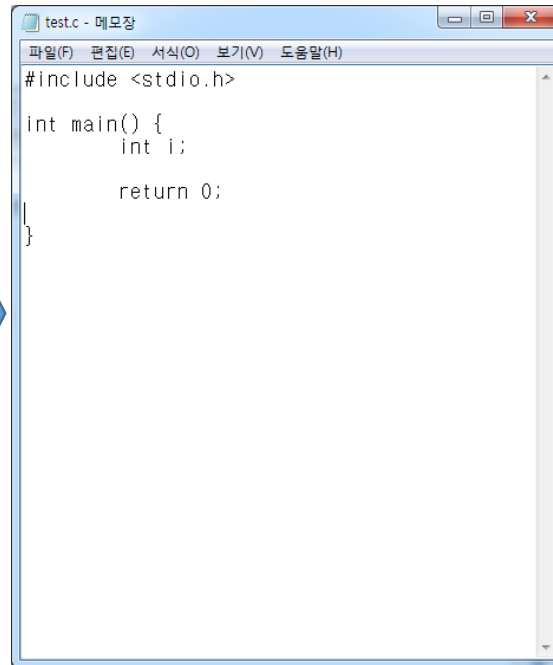


A screenshot of a code editor window titled 'test.c - 메모장'. The menu bar includes '파일(F)', '편집(E)', '서식(O)', '보기(V)', and '도움말(H)'. The code content is as follows:

```
#include <stdio.h>

int main() {
    return 0;
}
```

Second commit

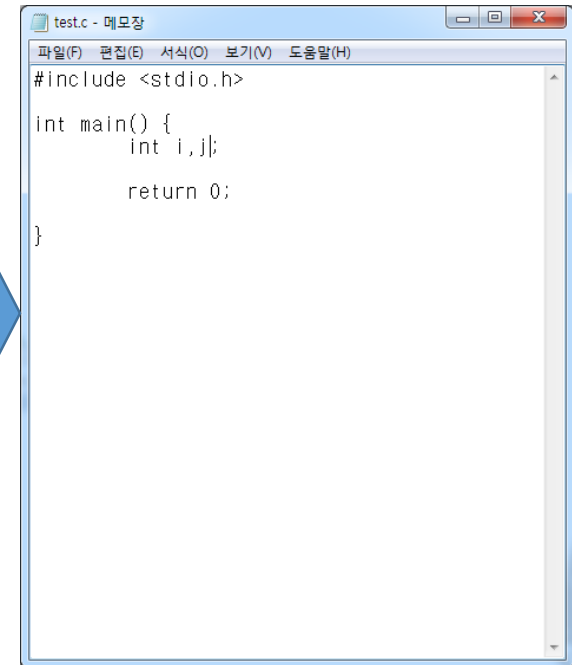


A screenshot of a code editor window titled 'test.c - 메모장'. The menu bar includes '파일(F)', '편집(E)', '서식(O)', '보기(V)', and '도움말(H)'. The code content is as follows:

```
#include <stdio.h>

int main() {
    int i;
    return 0;
}
```

Current status



A screenshot of a code editor window titled 'test.c - 메모장'. The menu bar includes '파일(F)', '편집(E)', '서식(O)', '보기(V)', and '도움말(H)'. The code content is as follows:

```
#include <stdio.h>

int main() {
    int i, j;
    return 0;
}
```



파일 되돌리기 2. stage의 변화

```
hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ notepad test.c

hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ git add .

hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ status
bash: status: command not found

hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   test.c

hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ git reset -- test.c
Unstaged changes after reset:
M       test.c

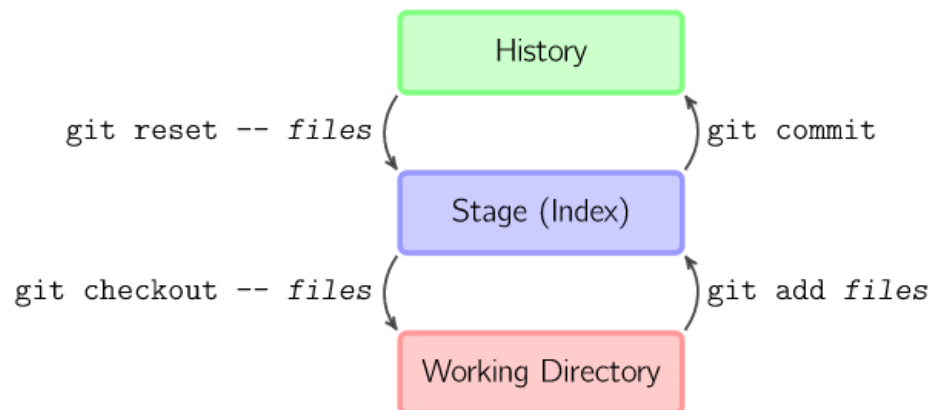
hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   test.c

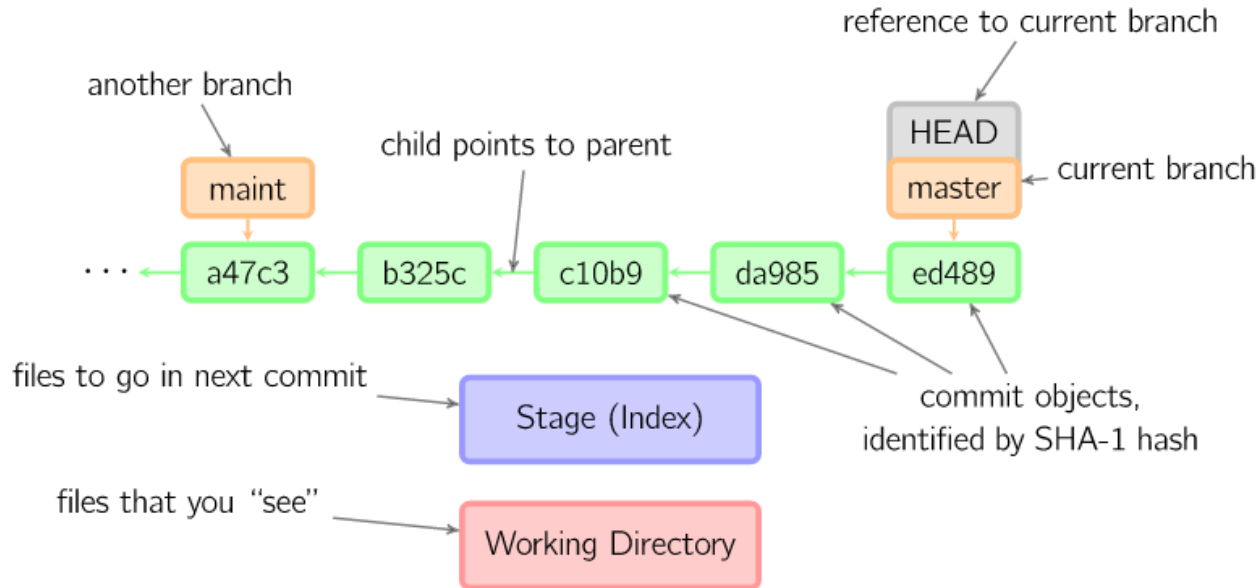
no changes added to commit (use "git add" and/or "git commit" to
hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ cat test.c
#include <stdio.h>

int main() {
    int i,j;

    return 0;
}
hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$
```



파일 되돌리기 3. HEAD 위치 변경



- git reflog
 - HEAD의 변경 이력을 보는 명령
- git checkout
 - HEAD를 옮기는 명령

파일 되돌리기 3. HEAD 위치 변경

```
hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ git reflog
d7a71a5 HEAD@{0}: checkout: moving from master to master
d7a71a5 HEAD@{1}: reset: moving to HEAD~
d9917de HEAD@{2}: reset: moving to HEAD
d9917de HEAD@{3}: commit: second commit
d7a71a5 HEAD@{4}: commit (initial): my first commit

hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ git checkout d9917de
Note: checking out 'd9917de'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

    git checkout -b <new-branch-name>

HEAD is now at d9917de... second commit

hcpark@hcpark-PC MINGW64 ~/tutorial ((d9917de...))
$ git log
commit d9917de88f98a69a9cd5c557f3da0a8de1cb8b23
Author: hcpark <hcpark.class@gmail.com>
Date: Thu Mar 16 16:39:40 2017 +0900

    second commit

commit d7a71a5edfb5e0e2fea75032f6bad8ce374b886f
Author: hcpark <hcpark.class@gmail.com>
Date: Thu Mar 16 16:36:34 2017 +0900

    my first commit

hcpark@hcpark-PC MINGW64 ~/tutorial ((d9917de...))
$ cat test.c
#include <stdio.h>

int main() {
    int i;

    return 0;
}
hcpark@hcpark-PC MINGW64 ~/tutorial ((d9917de...))
$
```

```
hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ git commit -m "second commit"
[master d9917de] second commit
1 file changed, 1 insertion(+)

hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$ git log
commit d9917de88f98a69a9cd5c557f3da0a8de1cb8b23
Author: hcpark <hcpark.class@gmail.com>
Date: Thu Mar 16 16:39:40 2017 +0900

    second commit

commit d7a71a5edfb5e0e2fea75032f6bad8ce374b886f
Author: hcpark <hcpark.class@gmail.com>
Date: Thu Mar 16 16:36:34 2017 +0900

    my first commit

hcpark@hcpark-PC MINGW64 ~/tutorial (master)
$
```



파일 되돌리기 3. HEAD 위치 변경

- 또다른 방법
 - Git reset 사용
 - Branch 학습 이후에 설명



Git tag

```
phcph@NOTE9 MINGW64 ~/tutorial (master)
$ git log
commit 798c47abffd84e764492e2f83aa8876feadf681f
Author: hcpark <hcpark.class@gmail.com>
Date: Thu Mar 16 20:46:31 2017 +0900
```

3rd commit

```
commit ee23f1a1f351bea6b19caba6ff656461fcbe0758
Author: hcpark <hcpark.class@gmail.com>
Date: Thu Mar 16 20:46:04 2017 +0900
```

2nd commit

```
commit b645b93df906e3e37bf7853790e9ae5796e933ca
Author: hcpark <hcpark.class@gmail.com>
Date: Thu Mar 16 20:45:28 2017 +0900
```

1st commit

```
phcph@NOTE9 MINGW64 ~/tutorial (master)
$ git tag v0.1
```

```
phcph@NOTE9 MINGW64 ~/tutorial (master)
$ git tag
v0.1
```



Git tag

```
phcph@NOTE9 MINGW64 ~/tutorial ((v0.2))
$ git tag
v0.1
v0.2
gi
phcph@NOTE9 MINGW64 ~/tutorial ((v0.2))
$ git checkout v0.1
Previous HEAD position was 668e013... 5th commit
HEAD is now at 798c47a... 3rd commit

phcph@NOTE9 MINGW64 ~/tutorial ((v0.1))
$ git log
commit 798c47abffd84e764492e2f83aa8876feadf681f
Author: hcpark <hcpark.class@gmail.com>
Date: Thu Mar 16 20:46:31 2017 +0900

    3rd commit

commit ee23f1a1f351bea6b19caba6ff656461fcbe0758
Author: hcpark <hcpark.class@gmail.com>
Date: Thu Mar 16 20:46:04 2017 +0900

    2nd commit

commit b645b93df906e3e37bf7853790e9ae5796e933ca
Author: hcpark <hcpark.class@gmail.com>
Date: Thu Mar 16 20:45:28 2017 +0900

    1st commit

phcph@NOTE9 MINGW64 ~/tutorial ((v0.1))
$
```



실습 과제 (git 개인 실습 #1)

- local repository 생성 후, 7개 이상의 commit을 만든다
 - File은 “test.c” 1개, 내용은 “1”, “2”, “3” 식으로 단순 증가
 - git log 캡처: log.jpg
- 중간에 tag를 3개 이상 만든다
 - v0.1~v0.3
 - git tag 캡처: tag.jpg
- HEAD를 다양한 방식으로 움직여본다
 - git checkout <commit id> 2가지
 - git checkout <tag> 2가지
 - 각각 cat test.c 로 내용 확인 (HEAD 위치에 따라 적절히 변경되어야 함)
 - 수행 화면 캡처: head.jpg
- 제출 기한:
 - 10/13 (일) 23:59
 - 지각 감점: 5%p / day (3주 내 제출해야 함)

GitHub



실습 과제 (git 개인 실습 #2)

- 앞서의 local repo 를 GitHub에 업로드
 - GitHub 프로젝트 페이지 전체 캡처: github.jpg
- 새로운 local repo 를 만들고, clone
 - Local repo의 status, log를 캡처: local_new.jpg
- 충돌관리 실습
 - 충돌 1
 - 새로운 local repo 에서 기존 파일 수정 후 push
 - 기존 local repo 에서 기존 파일 수정 후 push
 - 결과 화면 캡처: result1.jpg
 - 충돌 2
 - 기존 local repo 에서 새로운 파일 생성 후 push
 - 결과 화면 캡처: result2.jpg
 - 해결한 후 화면 및 GitHub commit history캡처: result3.jpg history.jpg

GitHub는?

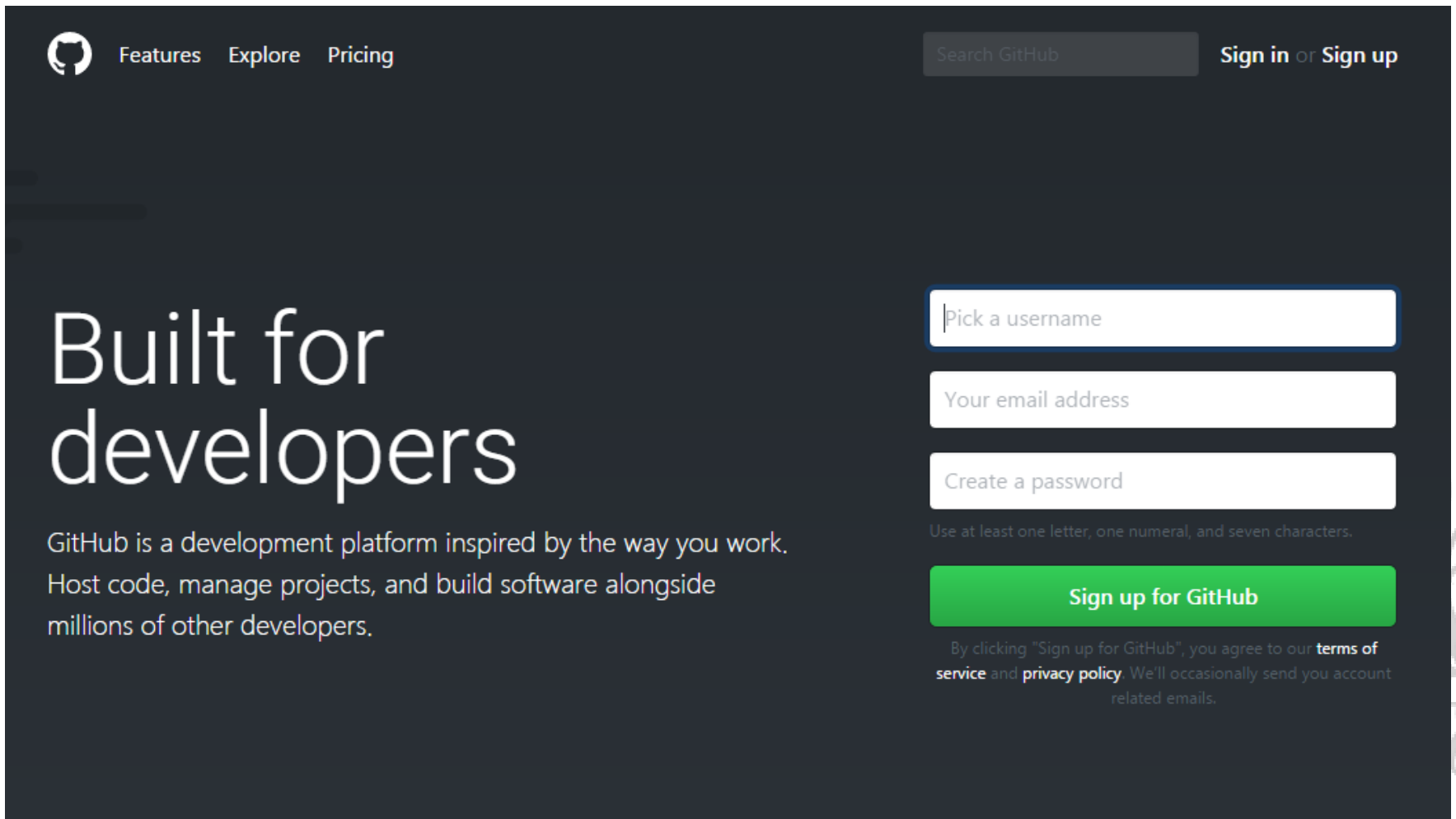
- Git의 원격 저장소를 제공
- 프로젝트 관리 도구 제공
 - 위키, 이슈관리, 머지 요청 관리, 팀원 관리 등
- 비용
 - 오픈소스 프로젝트는 무료

Developer		Team		Business	
\$7		\$9		\$21	\$21*
per month		per user / month		per user / month	per user / month
Includes: Personal account Unlimited public repositories Unlimited private repositories Unlimited collaborators		Includes: Organization account Unlimited public repositories Unlimited private repositories Team and user permissions		Hosted on GitHub.com Organization account SAML single sign-on Access provisioning 24/5 support with 8-hour response time 99.95% Uptime SLA Team sync (Coming 2017)	GitHub Enterprise Multiple organizations SAML, LDAP, and CAS Access provisioning 24/7 support for urgent issues Advanced auditing Host on your servers, AWS, Azure, or GCP
Free for students as part of the Student Developer Pack .		Starting at \$25 / month which includes your first 5 users.			



GitHub 가입

- <https://github.com/>

A screenshot of the GitHub sign-up page. The page has a dark blue background. At the top left is the GitHub logo (Octocat) followed by links for 'Features', 'Explore', and 'Pricing'. At the top right is a search bar labeled 'Search GitHub' and links for 'Sign in' or 'Sign up'. The main heading 'Built for developers' is in large white text. Below it, a paragraph describes GitHub as a development platform. On the right side, there are three input fields: 'Pick a username', 'Your email address', and 'Create a password'. Below the password field is a note: 'Use at least one letter, one numeral, and seven characters.' A green button labeled 'Sign up for GitHub' is below the inputs. At the bottom of the sign-up section, there is a disclaimer about terms of service and privacy policy.

Features Explore Pricing

Search GitHub Sign in or Sign up

Built for developers

GitHub is a development platform inspired by the way you work. Host code, manage projects, and build software alongside millions of other developers.

Pick a username

Your email address

Create a password

Use at least one letter, one numeral, and seven characters.

Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our **terms of service** and **privacy policy**. We'll occasionally send you account related emails.



GitHub 학생용: free private repository

GitHub Education[Stories](#)[Events](#)[Student pack](#)[Classroom](#)[Community](#)[Contact us](#)[Request a discount](#)

Student Developer Pack

The best developer tools, free for students

Are you a student?

The GitHub Student Developer Pack is **only available to students aged 13 or older**. Before you receive access to the offers we need to verify that you are a student.

Teachers, researchers, faculty, staff, and other educational users can get free and discounted access to GitHub, but are not eligible for the pack. If you're not a student, you can still request a regular GitHub for education discount.

Yes, I'm a student

[No, I'm not a student but would still like a discount](#)



Repository 생성

Create a new repository

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

Owner



hyunchan-park ▾

/

Repository name

Great repository names are short and memorable. Need inspiration? How about **congenial-giggle**.

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



전북대학교 컴퓨터공학부

Division of Computer Science and Engineering
Chonbuk National University

Repository 생성 후

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



Set up in Desktop

or

HTTPS

SSH

`https://github.com/hyunchan-park/swproject.git`



We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# swproject" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/hyunchan-park/swproject.git
git push -u origin master
```



...or push an existing repository from the command line

```
git remote add origin https://github.com/hyunchan-park/swproject.git
git push -u origin master
```



...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code



로컬 - 원격 저장소 간의 이동

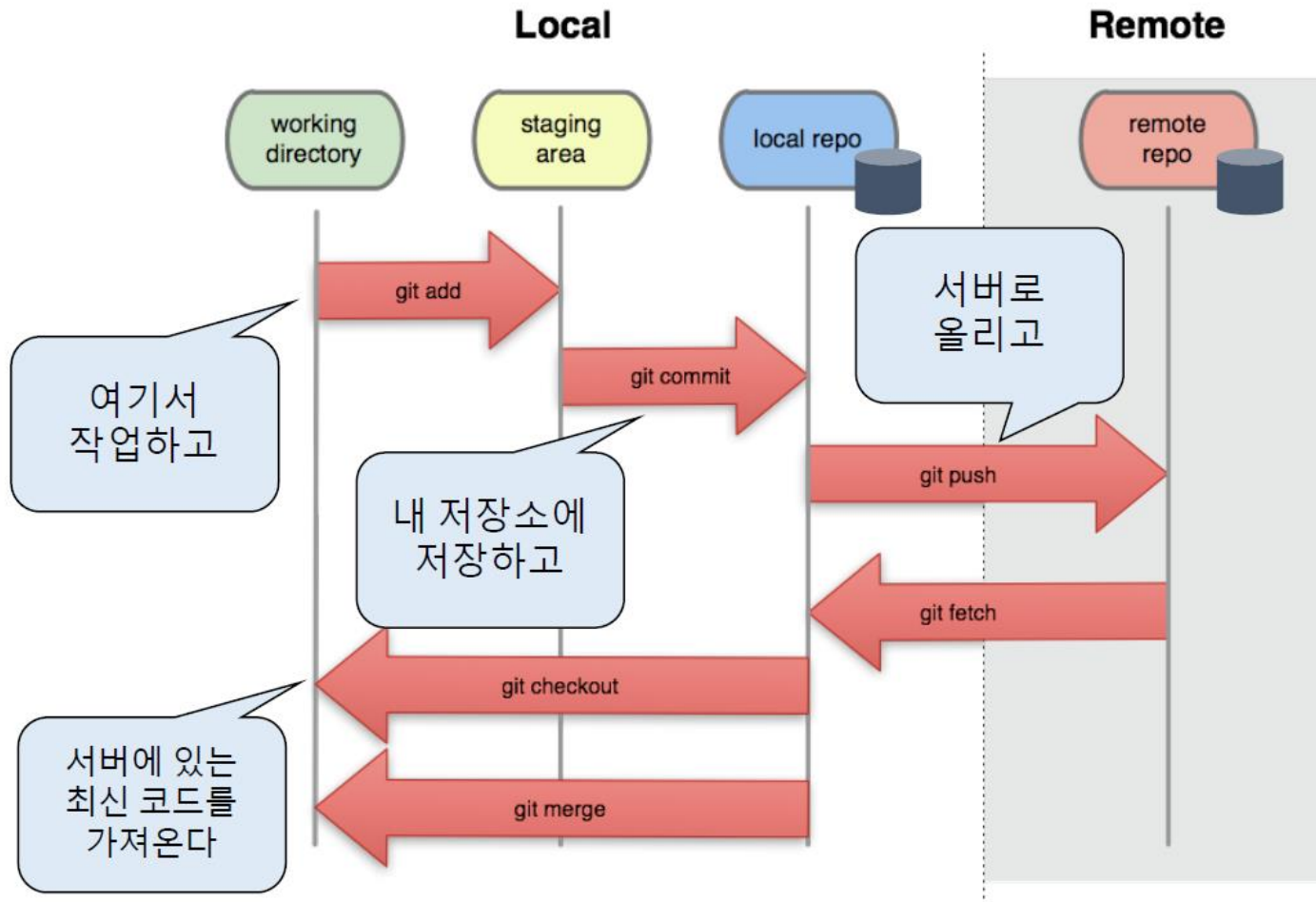


그림 출처 : <http://pismute.github.io/whygitisbetter/#everything-is-local>



Local repository 를 GitHub remote로 push

* GitHub 로그인 창이 뜰 수 있음

```
phcph@NOTE9 MINGW64 ~/tutorial ((v0.1))
$ git remote add origin https://github.com/hyunchan-park/swproject.git


phcph@NOTE9 MINGW64 ~/tutorial ((v0.1))
$ git remote add origin https://github.com/hyunchan-park/swproject.git
fatal: remote origin already exists.




phcph@NOTE9 MINGW64 ~/tutorial ((v0.1))
$ git push -u origin master
Counting objects: 15, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (15/15), 1022 bytes | 0 bytes/s, done.
Total 15 (delta 0), reused 0 (delta 0)
To https://github.com/hyunchan-park/swproject.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.









phcph@NOTE9 MINGW64 ~/tutorial ((v0.1))
$
```




Push 완료 후 GitHub 확인






 hyunchan-park / swproject


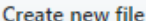
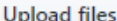


 Unwatch ▾ 1  Star 0  Fork 0


 Code  Issues 0  Pull requests 0  Projects 0  Wiki  Security  Insights  Settings




No description, website, or topics provided. 



[Manage topics](#)

 6 commits  1 branch  0 releases  1 contributor  Apache-2.0

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

 hyunchan-park 5th commit Latest commit 3c4b482 1 minute ago

 LICENSE	Initial commit	2 minutes ago
 README.md	Initial commit	2 minutes ago
 test.c	5th commit	1 minute ago

 README.md 

swproject



GitHub commit history 확인













hyunchan-park / swproject

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Security Insights Settings

Branch: master

Commits on Sep 27, 2019

5th commit hyunchan-park committed 20 seconds ago	Verified	 3c4b482	
4th commit hyunchan-park committed 32 seconds ago	Verified	 c52fd9f	
3rd commit hyunchan-park committed 44 seconds ago	Verified	 2b7e77d	
2nd commit hyunchan-park committed 1 minute ago	Verified	 b31206d	
1st commit hyunchan-park committed 1 minute ago	Verified	 bff547e	
Initial commit hyunchan-park committed 2 minutes ago	Verified	 77b374c	

다른 Local repository 생성 및 GitHub clone

The screenshot shows the GitHub interface for a repository named 'swproject' by user 'hyunchan-park'. At the top, there are buttons for 'Watch', 'Star', and 'Fork', each with a count of 0. Below this is a navigation bar with links for 'Code', 'Issues', 'Pull requests', 'Projects', 'Wiki', 'Pulse', 'Graphs', and 'Settings'. The repository description is 'For 2017 SW Project Class (NOT A PUBLIC PROJECT)' with an 'Edit' button. Below the description, statistics show '5 commits', '1 branch', '0 releases', and '0 contributors'. A row of buttons includes 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and a green 'Clone or download' button. A dropdown menu is open from the 'Clone or download' button, showing 'Clone with HTTPS' (selected), 'Use SSH', and the URL 'https://github.com/hyunchan-park/swproject'. Below the URL is a 'Copy to clipboard' button. At the bottom of the dropdown are 'Open in Desktop' and 'Download ZIP' buttons. The repository content area shows a commit by 'hcpark' with a file 'test.c' and a message to add a README.

hyunchan-park / swproject

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

For 2017 SW Project Class (NOT A PUBLIC PROJECT) Edit

Add topics

5 commits 1 branch 0 releases 0 contributors

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

Clone with HTTPS Use SSH

Use Git or checkout with SVN using the web URL.

https://github.com/hyunchan-park/swproject Copy to clipboard

Open in Desktop Download ZIP

hcpark 5th commit

test.c 5th commit

Help people interested in this repository understand your project by adding a README.

다른 Local repository 생성 및 GitHub clone

- git clone https://github.com/hyunchan-park/swproject.git <folder>

```
phcph@NOTE9 MINGW64 ~  
$ mkdir test  
  
phcph@NOTE9 MINGW64 ~  
$ cd test  
  
phcph@NOTE9 MINGW64 ~/test  
$ git clone git://github.com/hyunchan-park/swproject.git .  
Cloning into '.'...  
remote: Counting objects: 15, done.  
remote: Compressing objects: 100% (5/5), done.  
remote: Total 15 (delta 0), reused 15 (delta 0), pack-reused 0  
Receiving objects: 100% (15/15), done.  
  
phcph@NOTE9 MINGW64 ~/test (master)  
$
```

* GitHub 로그인 창이 뜰 수 있음

- git config --global http.sslVerify false
- git remote set-url origin https://github.com/hyunchan-park/swproject.git

GitHub 충돌 관리

- 새로운 local repo TEST 에서 파일 수정 후 commit, push 수행
 - \$ git push origin master
 - Local repository의 commit 내역을 remote로 전송
- 기존 local repo Tutorial 에서 파일 수정 후 commit, push 수행
 - 결과는?

GitHub 충돌 관리

local repo TEST

```
phcph@NOTE9 MINGW64 ~/test (master)
$ vi test.c

phcph@NOTE9 MINGW64 ~/test (master)
$ git add .

phcph@NOTE9 MINGW64 ~/test (master)
$ git commit -m "6th commit"
[master 030bee4] 6th commit
1 file changed, 1 insertion(+), 1 deletion(-)

phcph@NOTE9 MINGW64 ~/test (master)
$ git log
commit 030bee4f22445f87379d9a50438ff2d6afba947c
Author: hcpark <hcpark.class@gmail.com>
Date: Thu Mar 16 21:54:58 2017 +0900
```

6th commit

```
commit 668e0135fad439bc1b876131de3704ac8cd934f2
Author: hcpark <hcpark.class@gmail.com>
Date: Thu Mar 16 20:49:56 2017 +0900
```

5th commit

```
commit 5db18f0156c68db7e4e181569b4822c8d1d134d3
Author: hcpark <hcpark.class@gmail.com>
Date: Thu Mar 16 20:49:35 2017 +0900
```

4th commit

```
commit 798c47abffd84e764492e2f83aa8876feadf681f
Author: hcpark <hcpark.class@gmail.com>
Date: Thu Mar 16 20:46:31 2017 +0900
```

3rd commit

```
commit ee23f1a1f351bea6b19caba6ff656461fcbe0758
Author: hcpark <hcpark.class@gmail.com>
Date: Thu Mar 16 20:46:04 2017 +0900
```

2nd commit

```
commit b645b93df906e3e37bf7853790e9ae5796e933ca
Author: hcpark <hcpark.class@gmail.com>
```

hyunchan-park / swproject

Watch 0 Star 0 Fork 0

Code Issues Pull requests Projects Wiki Pulse Graphs Settings

Branch: master

Commits on Mar 16, 2017

5th commit	668e013
hpcark committed 35 minutes ago	
4th commit	5db18f0
hpcark committed 35 minutes ago	
3rd commit	798c47a
hpcark committed 38 minutes ago	
2nd commit	ee23f1a
hpcark committed 39 minutes ago	
1st commit	b645b93
hpcark committed 39 minutes ago	

local repo TUTORIAL

```
phcph@NOTE9 MINGW64 ~/tutorial ((v0.2))
$ git log
commit 668e0135fad439bc1b876131de3704ac8cd934f2
Author: hcpark <hcpark.class@gmail.com>
Date: Thu Mar 16 20:49:56 2017 +0900
```

5th commit

```
commit 5db18f0156c68db7e4e181569b4822c8d1d134d3
Author: hcpark <hcpark.class@gmail.com>
Date: Thu Mar 16 20:49:35 2017 +0900
```

4th commit

```
commit 798c47abffd84e764492e2f83aa8876feadf681f
Author: hcpark <hcpark.class@gmail.com>
Date: Thu Mar 16 20:46:31 2017 +0900
```

3rd commit

```
commit ee23f1a1f351bea6b19caba6ff656461fcbe0758
Author: hcpark <hcpark.class@gmail.com>
Date: Thu Mar 16 20:46:04 2017 +0900
```

2nd commit

```
commit b645b93df906e3e37bf7853790e9ae5796e933ca
Author: hcpark <hcpark.class@gmail.com>
Date: Thu Mar 16 20:45:28 2017 +0900
```

1st commit

GitHub 충돌 관리

```
phcph@NOTE9 MINGW64 ~/test (master)
$ git push origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 235 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/hyunchan-park/swproject.git
668e013..030bee4 master -> master
```

```
phcph@NOTE9 MINGW64 ~/test (master)
$ |
```

```
phcph@NOTE9 MINGW64 ~/tutorial ((4adbac1...))
$ git push origin master
To https://github.com/hyunchan-park/swproject.git
! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/hyunchan-park/swproject.g
it'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

```
phcph@NOTE9 MINGW64 ~/tutorial ((4adbac1...))
$ |
```



GitHub 충돌 관리: 새로운 파일이 추가되면?

```
phcph@NOTE9 MINGW64 ~/tutorial (master)
$ touch another.c

phcph@NOTE9 MINGW64 ~/tutorial (master)
$ git add .

phcph@NOTE9 MINGW64 ~/tutorial (master)
$ git commit -m "file added"
[master 90d69cb] file added
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 another.c

phcph@NOTE9 MINGW64 ~/tutorial (master)
$ git push origin master
To https://github.com/hyunchan-park/swproject.git
! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/hyunchan-park/swproject.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

phcph@NOTE9 MINGW64 ~/tutorial (master)
$
```



Git 충돌 관리: pull before push

- Remote repository를 중앙 서버처럼 이용하고 있으므로
 - git pull : 타 사용자의 작업 내용을 local 에 반영하여,
 - 최종 상태로 업데이트 후에 내 변경 내용을 push

```
phcph@NOTE9 MINGW64 ~/tutorial (master)
$ git pull
Merge made by the 'recursive' strategy.
 test.c | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

phcph@NOTE9 MINGW64 ~/tutorial (master)
$ cat test.c
6

phcph@NOTE9 MINGW64 ~/tutorial (master)
$ |

























phcph@NOTE9 MINGW64 ~/tutorial (master)
$ git push origin master
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 566 bytes | 0 bytes/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To https://github.com/hyunchan-park/swproject.git
 030bee4..375f67a master -> master

phcph@NOTE9 MINGW64 ~/tutorial (master)
$ |
```



Git 충돌 관리: 최종 결과 GitHub에서 확인

Commits on Mar 16, 2017

	Merge branch 'master' of https://github.com/hyunchan-park/swproject hcpark committed 2 minutes ago	 375f67a 
	file added hcpark committed 4 minutes ago	 90d69cb 
	6th commit hcpark committed an hour ago	 030bee4 
	5th commit hcpark committed 2 hours ago	 668e013 
	4th commit hcpark committed 2 hours ago	 5db18f0 
	3rd commit hcpark committed 2 hours ago	 798c47a 
	2nd commit hcpark committed 2 hours ago	 ee23f1a 
	1st commit hcpark committed 2 hours ago	 b645b93 



실습 과제 (git 개인 실습 #2)

- GitHub Desktop 설치하고, 두 개의 local repo 를 이용, 충돌 시연
 - Slide #52 의 화면처럼 각각의 repo 에서 git log 수행 후, 캡처
 - git log 캡처: log.jpg
 - Slide #53 처럼, Test 에서 먼저 push 한 후, tutorial 에서 push 오류 확인
 - crash.jpg
 - Slide #54,55 처럼, tutorial 에서 새로운 파일 추가 후, pull 하고, 다시 test.c 내용 확인 한 후, push 수행
 - pull.jpg
- 제출 기한:
 - 10/20 (일) 23:59
 - 지각 감점: 5%p / day (3주 내 제출해야 함)