

Git 및 GitHub 사용법 기초

2016. 11

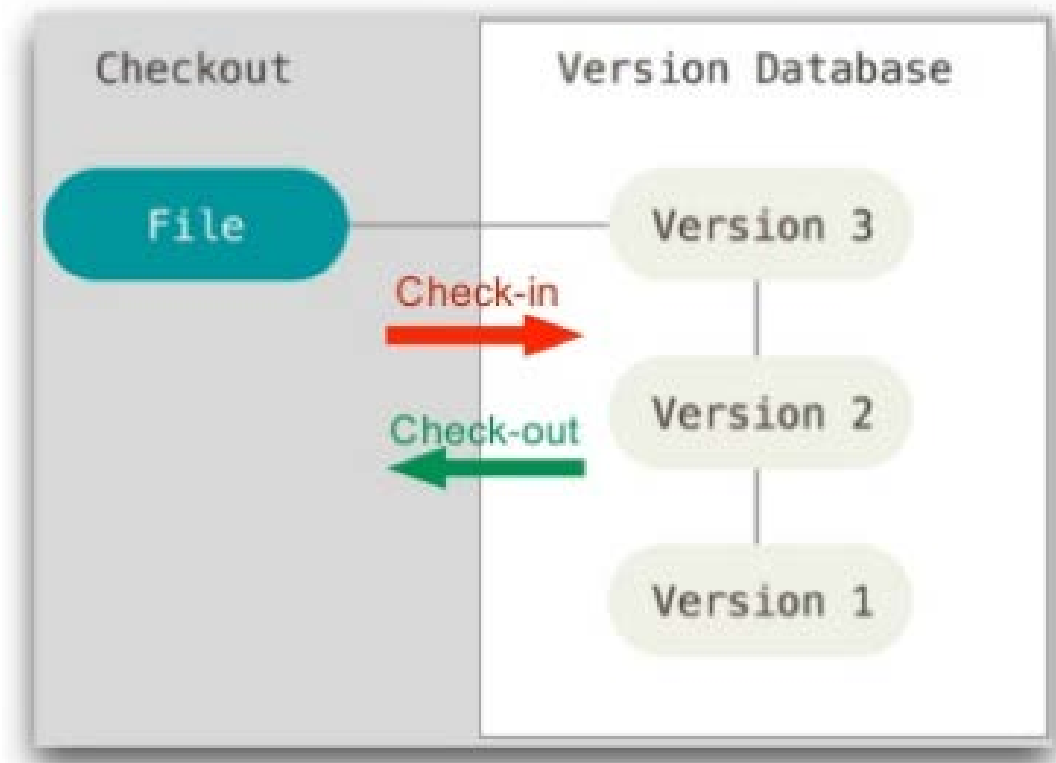
한국산업기술대학교
컴퓨터공학과
공 기 석

버전 관리 시스템이란?

- Version Control System (VCS)
- 소프트웨어 개발 팀이 개발 중에 소스코드의 변경 관리를 돕는 소프트웨어 도구
- 모든 파일 변화를 시간에 따라 기록 한 후 나중에 특정 시점의 버전을 다시 꺼내 올 수 있는 시스템
- 거의 모든 컴퓨터 파일의 버전을 관리할 수 있음

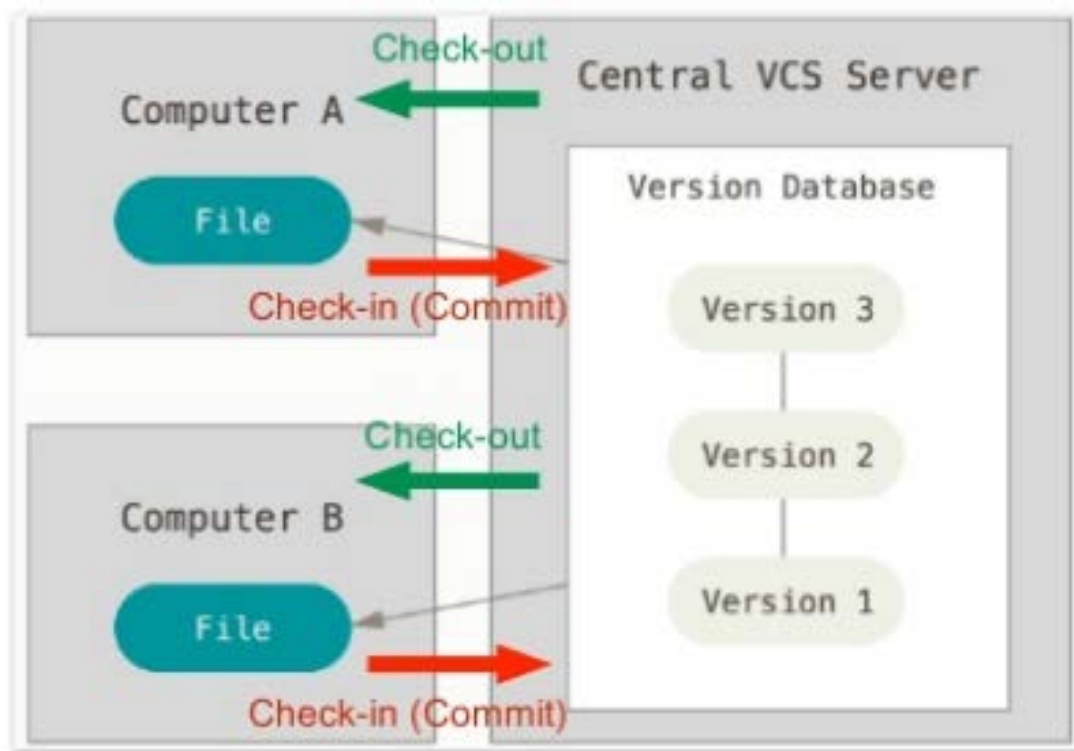
로컬 버전 관리

- 간단한 로컬 데이터베이스를 사용
 - ✓ 예: RCS (Revision Control System)



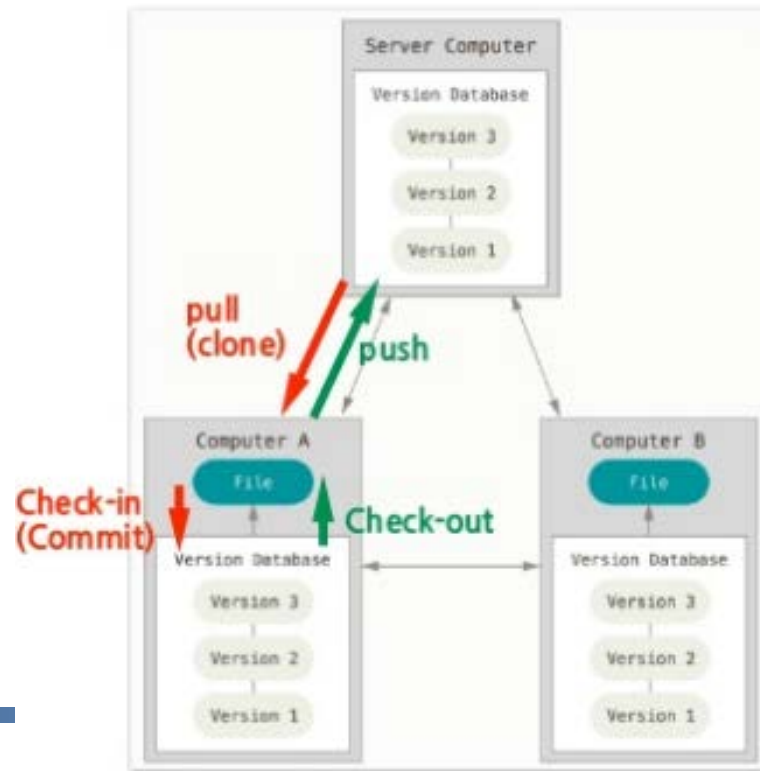
중앙집중식 버전관리 (CVCS)

- 버전 관리를 위해 네트워크 상의 서버를 이용하는 방법
 - ✓ 예: CVS, Subversion, Perforce
 - ✓ 장점: 개발자들의 관리 현황 파악이 용이
 - ✓ 단점: 서버가 다운되면 협업 불가능. 백업 불가능

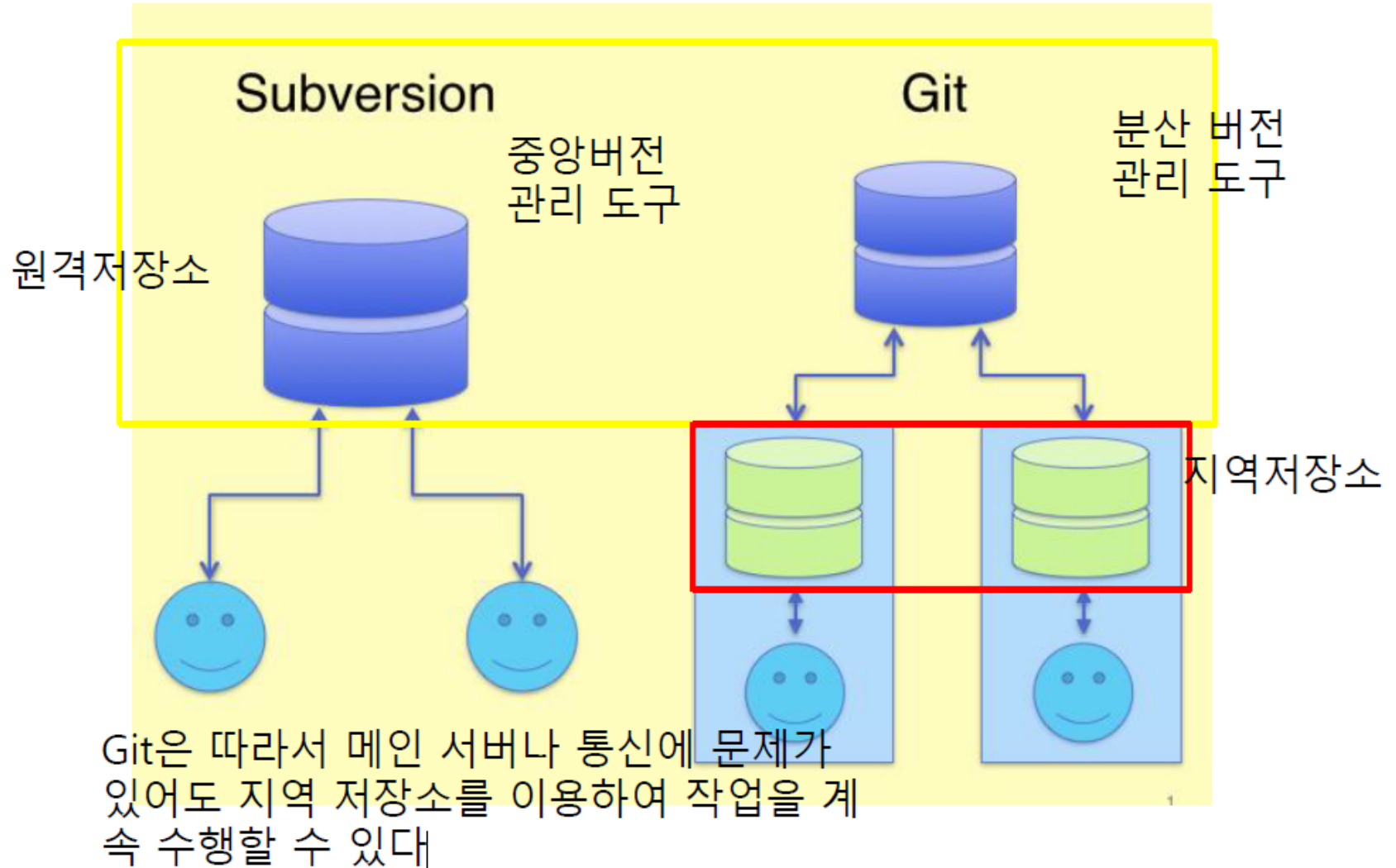


분산 버전관리시스템 (DVCS)

- 저장소 전체를 로컬 호스트로 복사한 후 작업을 수행. 협업이 필요할 때 네트워크 서버를 이용
 - ✓ 예: Git, Mercurial, Bazaar, Darcs 등
 - ✓ 장점: 서버에 장애가 발생해도 로컬 컴퓨터에서 작업진행 가능, 다양한 워크플로우 가능
 - ✓ 단점: 바이너리 데이터 저장시 저장공간 문제, 히스토리가 많은 경우 다운로드/업로드에 시간소요



Subversion vs. Git



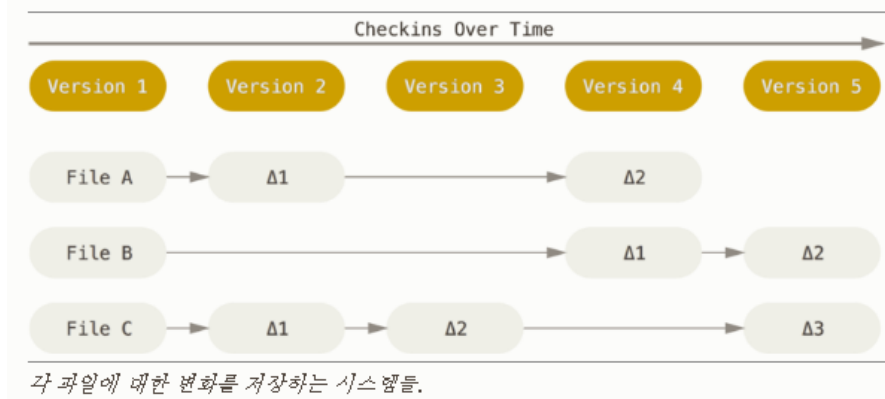
Git의 역사

- 리눅스 커널 프로젝트를 위한 버전 관리 시스템으로 개발
- 2005년 리눅스 토발즈가 발표
- 설계 목표
 - ✓ 빠른 속도
 - ✓ 단순한 구조
 - ✓ 동시 다발적인 브랜치를 통한 비선형적 개발
 - ✓ 완벽한 분산
 - ✓ 리눅스 커널 같은 대형 프로젝트의 효율적인 지원

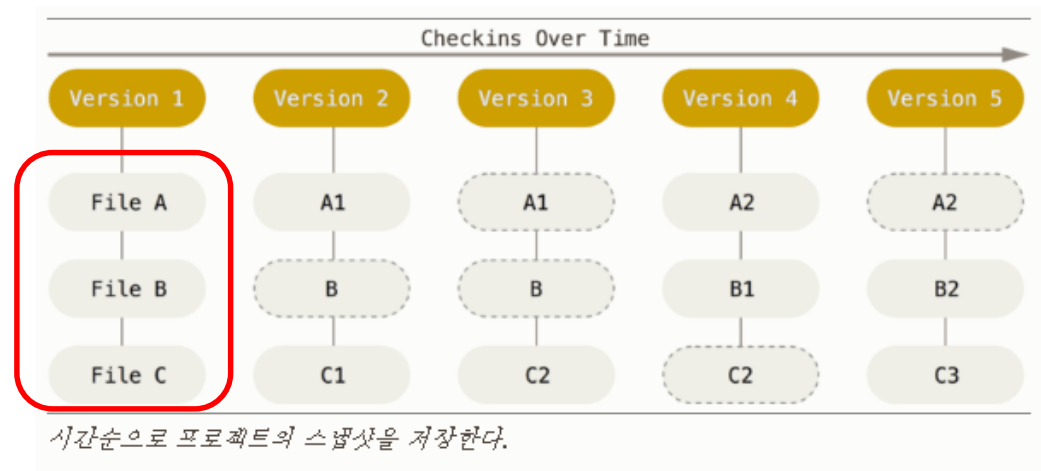


Git의 기본 개념 (1)

- 델타가 아닌 스냅샷으로 저장
 - ✓ 이전 버전에서 바뀐 델타 값을 저장하는 기존 VCS와는 달리 매 커밋 순간의 스냅샷을 저장함



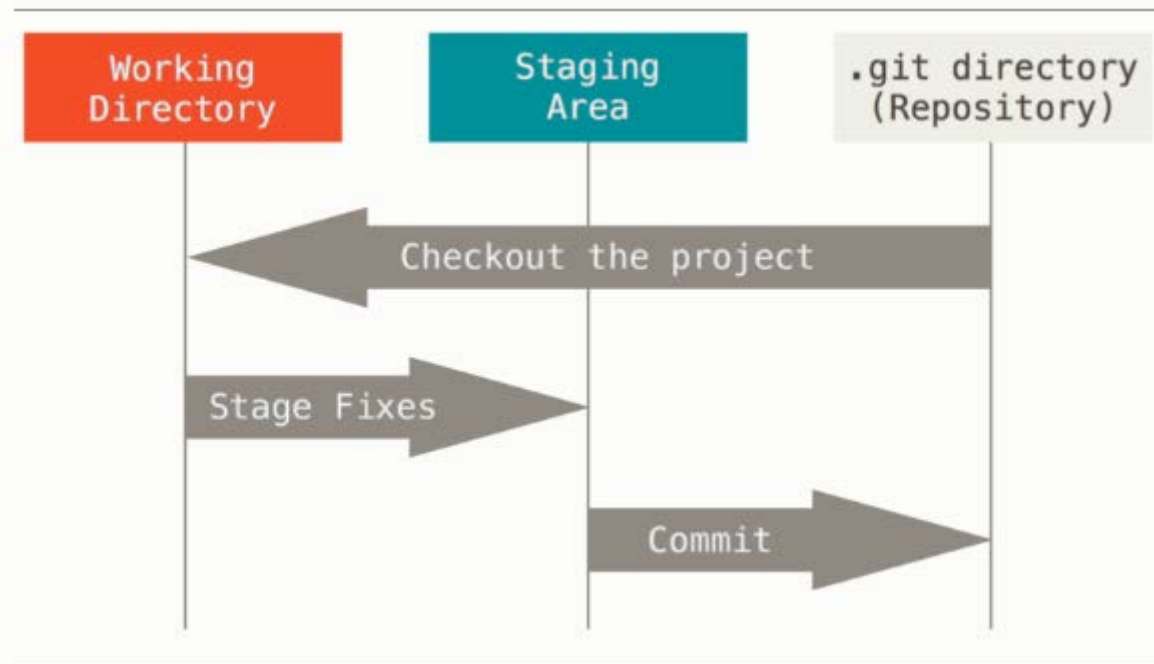
스냅샷



Git의 기본 개념 (2)

■ 파일의 세가지 상태

- ✓ **Modified**: 수정만 하고 **Stage**나 커밋하지 않은 상태. 작업 디렉토리(working directory)에 존재
- ✓ **Staged**: 현재 수정한 파일을 곧 커밋할 것이라고 표시한 상태. Staging area에 존재
- ✓ **Committed**: 파일이 지역저장소(repository)에 안전하게 저장된 상태. git 디렉토리에 존재



Git 의 설치

<http://git-scm.com/downloads>

The 2016 Git User's Survey is now up! 12 September — 20 October 2016.

Please devote a few moments of your time to **fill out the simple questionnaire**. It will help the Git community understand your needs, what you like about Git (and what you don't), and help us improve it in general. The results will be published at the **GitSurvey2016** wiki page.

git --local-branching-on-the-cheap

Search entire site...

About
Documentation
Blog
Downloads
GUI Clients
Logos
Community

Downloads

Mac OS X Windows
Linux Solaris

Older releases are available and the Git source repository is on GitHub.

Latest source Release
2.10.1
Release Notes (2016-10-03)
Downloads for Windows

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to **read online for free**. Dead tree versions are available on Amazon.com.

GUI Clients

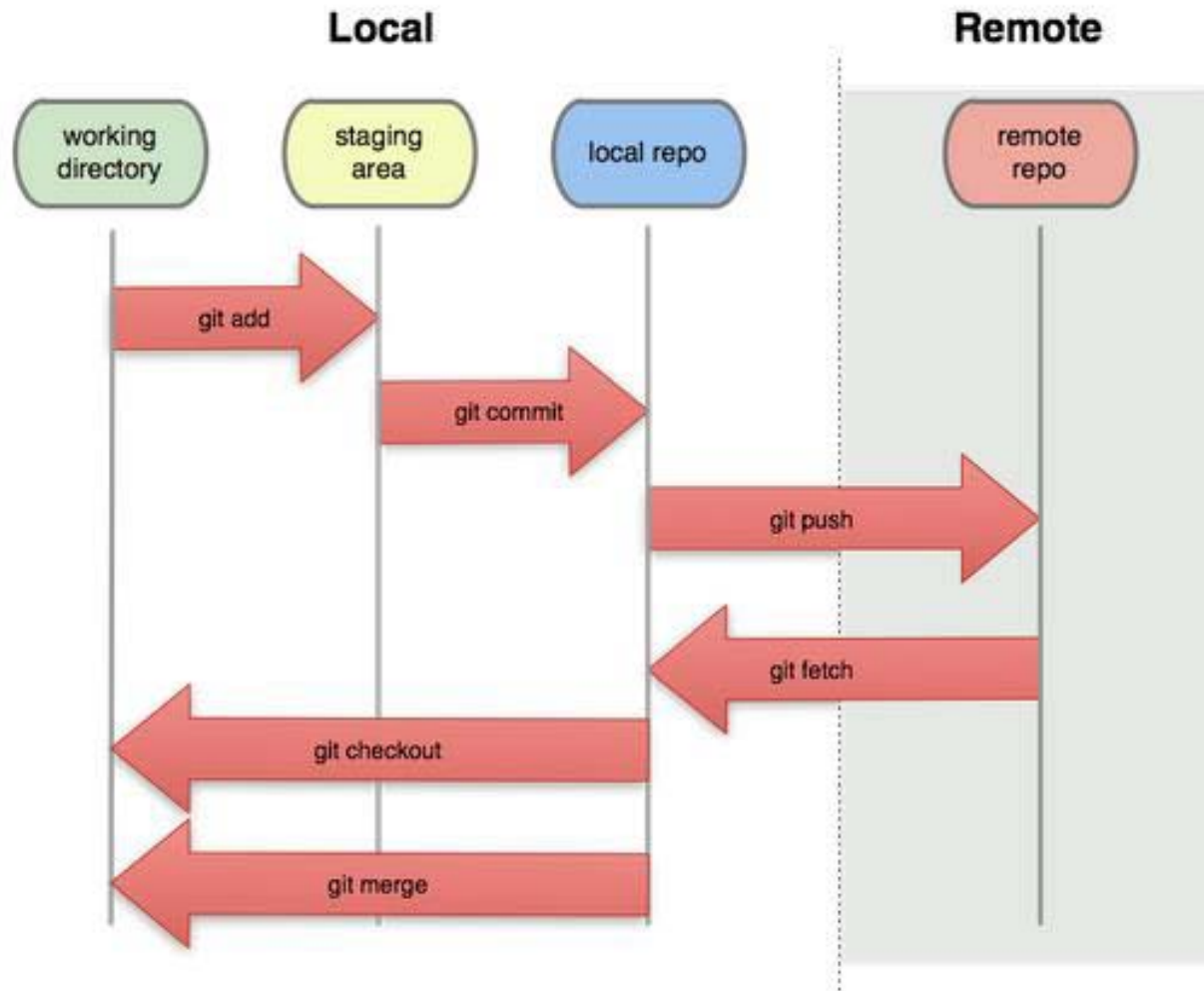
Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.

[View GUI Clients →](#)

OR

```
$ sudo apt-get update  
$ sudo apt-get install git
```

우리가 배울 Git의 주요 명령어



Git 실습

- ① 저장소 생성
- ② 저장소에 Hello World를 출력하는 프로그램 작성 및 추가
- ③ 커밋
- ④ hotfix 브랜치 생성 및 이동
- ⑤ 프로그램 수정
- ⑥ 커밋
- ⑦ master 브랜치에 병합
- ⑧ master 브랜치에 변경점 하나 추가
- ⑨ 커밋
- ⑩ hotfix 브랜치에 변경점 하나 추가
- ⑪ 커밋
- ⑫ master와 hotfix 브랜치 사이에 영향이 없음을 확인
- ⑬ 불필요한 프로젝트 파일을 관리 대상에서 제외하기
- ⑭ 충돌 해결하기
- ⑮ 기록 보기

(1) 저장소 생성

// 사용자와 이메일 등록

```
$ git config --global user.name "Kisok Kong"
```

```
$ git config --global user.email "kskong@kpu.ac.kr"
```

```
$ mkdir test-repo (→ 이 위치가 Working Directory)
```

```
$ cd test-repo
```

```
$ git init
```

// 이 Directory에서 작업

(2) Hello World 프로그램 작성 및 추가 (1)

■ vim helloworld.c 작성

```
computer.kpu.ac.kr - PuTTY
1 #include <stdio.h>
2
3 int main(void) {
4
5     printf("Hello World\n");
6
7     return 0;
8 }
~
~
```

■ 커밋하기 전에 git status 명령을 사용하여 저장소 상태 확인

```
$ git status
On branch master

Initial commit

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

        helloworld.c

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

(2) Hello World 프로그램 작성 및 추가 (2)

- `git add helloworld.c` 명령어를 사용하여 파일 추적 시작
- `git status` 명령으로 상태 확인

```
$ git add helloworld.c
$ git status
On branch master

Initial commit

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

        new file:   helloworld.c

$
```

(3) 커밋

- git commit 명령으로 커밋 실행
 - ✓ 커밋 메시지 작성 (vim 상태)
 - ✓ create “Hello World” program 타이핑

```
Create "Hello World" program
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   new file:   helloworld.c
#
~
~
```

- ✓ [ESC], :wq 로 저장하고 나오면 커밋 완료

```
~
~
"/u1/prof/kskong/test-repo/.git/COMMIT_EDITMSG" 10 행, 260 문자
[master (root-commit) d75d8de] Create "Hello World" program
1 file changed, 8 insertions(+)
create mode 100644 helloworld.c
$
```


(4) hotfix 브랜치 생성 및 이동

- git branch 명령으로 현재 어떤 브랜치가 있는지 확인

```
$ git branch
* master
$
```

- git branch hotfix 명령으로 hotfix 브랜치 생성 후,
git branch 명령으로 확인

```
$ git branch hotfix
$ git branch
hotfix
* master
$
```

- git checkout hotfix 로 hotfix 브랜치로 이동

```
$ git checkout hotfix
Switched to branch 'hotfix'
$ git branch
* hotfix
  master
$
```

참고:

\$ git checkout -b 브랜치이름

→ git branch 와 git checkout을 한번에 실행

(5) 프로그램 수정 (hotfix 브랜치)

■ vim helloworld.c 실행

- ✓ printf("Hello Your World\n"); 추가 (6번 라인)

```
1 #include <stdio.h>
2
3 int main(void) {
4
5     printf("Hello World\n");
6     printf("Hello Your World\n");
7
8     return 0;
9 }
```

- ✓ gcc helloworld.c 로 컴파일해서 동작 여부 확인

(6) git commit -a : 두 번째 커밋

- git status 로 파일 변경여부 확인

```
$ git status
On branch hotfix
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:   helloworld.c

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

- git commit -a 명령을 실행하고 커밋 메시지로 added output "Hello Your World" 추가 후 git status 로 저장소 상태 확인

```
added output "Hello Your World"
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch hotfix
# Changes to be committed:
#       modified:   helloworld.c
#
~
```

```
$ git status
On branch hotfix
nothing to commit, working directory clean
$
```

(7) master 브랜치에 병합(merge)

- master 브랜치로 이동 후 커밋 상태 확인 및 파일 내용 확인

- ✓ git checkout master
- ✓ git status
- ✓ ls
- ✓ cat helloworld.c

```
$ git checkout master
Switched to branch 'master'
$ git status
On branch master
nothing to commit, working directory clean
$ ls
합계 6
1 ./ 3 ../ 1 .git/ 1 helloworld.c
$ cat helloworld.c
#include <stdio.h>

int main(void) {

    printf("Hello World\n");

    return 0;
}
$
```

- hotfix 브랜치 병합

- ✓ git merge hotfix
- ✓ cat helloworld.c

```
$ git merge hotfix
Updating d75d8de..73afa9f
Fast-forward
 helloworld.c | 1 +
 1 file changed, 1 insertion(+)
$
```

```
$ cat helloworld.c
#include <stdio.h>

int main(void) {

    printf("Hello World\n");
    printf("Hello Your World\n");

    return 0;
}
$
```

(8) master 브랜치에 변경점 하나 추가

■ master 브랜치의 helloworld.c 수정

- ✓ `printf("Hello his world\n");` // 추가 (7번 라인)

```
1 #include <stdio.h>
2
3 int main(void) {
4
5     printf("Hello World\n");
6     printf("Hello Your World\n");
7     printf("Hello his World\n");
8
9     return 0;
10 }
```

- ✓ `git status`로 변경사항 확인

```
$ 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:   helloworld.c

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

(9) 세 번째 커밋

■ git commit -a 실행

- ✓ 커밋 메시지: added output "Hello his world"

```
added output "Hello his world"
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# Changes to be committed:
#   modified:   helloworld.c
#
~
```

- ✓ 저장 후 git status 로 확인

```
~
"/u1/prof/kskong/test-repo/.git/COMMIT_EDITMSG" 7 행, 241 문자
[master 70f6c87] added output "Hello his world"
 1 file changed, 1 insertion(+)
$ git status
On branch master
nothing to commit, working directory clean
$
```

(10) hotfix 브랜치에 변경점 하나 추가

- hotfix 브랜치로 이동하여 helloworld.c 에 printf("Hello her world\n") 추가
 - ✓ git checkout hotfix
 - ✓ vim helloworld.c
 - ✓ printf("Hello her world\n"); // 추가 (7번 라인)

```
1 #include <stdio.h>
2
3 int main(void) {
4
5     printf("Hello World\n");
6     printf("Hello Your World\n");
7     printf("Hello her World\n");
8
9     return 0;
10 }
```

(11) 네 번째 커밋

■ git commit -a 실행

- ✓ 커밋 메시지: added output "Hello her world"

```
added output "Hello her world"
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch hotfix
# Changes to be committed:
#   modified:   helloworld.c
#
~
```

- ✓ 저장 후 git status 로 확인

```
~
~
"/u1/prof/kskong/test-repo/.git/COMMIT_EDITMSG" 7 행, 241 문자
[hotfix 6ed7d37] added output "Hello her world"
 1 file changed, 1 insertion(+)
$ git status
On branch hotfix
nothing to commit, working directory clean
$
```


(12) master와 hotfix 브랜치 사이에 영향이 없음을 확인

- cat helloworld.c 를 실행하여 hotfix 브랜치의 helloworld.c 파일을 독립적으로 수정할 수 있음을 확인

```
$ cat helloworld.c
#include <stdio.h>

int main(void) {

    printf("Hello World\n");
    printf("Hello Your World\n");
    printf("Hello her World\n");

    return 0;
}
$
```

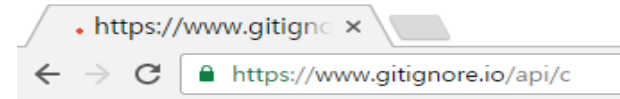
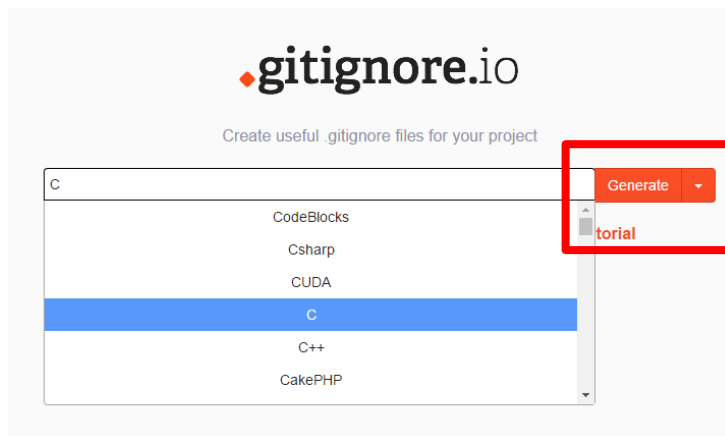
(13) 불필요한 파일을 관리 대상에서 제외하기 (1)

- .gitignore 파일을 생성
 - ✓ touch .gitignore

```
$ touch .gitignore
$ ls
합계 6
1 ./ 3 ../ 1 .git/ 0 .gitignore 1 helloworld.c
$ ls -al
합계 6
1 drwxr-xr-x 3 kskong prof 512 2016-10-20 15:11 ./
3 drwx-----x 55 kskong prof 2560 2016-10-20 14:49 ../
1 drwxr-xr-x 8 kskong prof 512 2016-10-20 14:50 .git/
0 -rw-r--r-- 1 kskong prof 0 2016-10-20 15:11 .gitignore
1 -rw-r--r-- 1 kskong prof 139 2016-10-20 14:26 helloworld.c
$
```

(13) 불필요한 파일을 관리 대상에서 제외하기 (2)

- <https://www.gitignore.io> 사이트에서 IDE와 프로그래밍 언어를 지정하여 .gitignore 파일 생성후 현재 디렉토리에 있는 .gitignore 파일에 복사



```
# Created by https://www.gitignore.io/api/c
```

```
### C ###  
# Prerequisites  
*.d
```

```
# Object files  
*.o  
*.ko  
*.obj  
*.elf
```

```
# Linker output  
*.ilk  
*.map  
*.exp
```

```
# Precompiled Headers  
*.gch  
*.pch
```

```
# Libraries  
*.lib  
*.a  
*.la  
*.lo
```

```
# Shared objects (inc. Windows DLLs)  
*.dll  
*.so  
*.so.*  
*.dylib
```

```
# Executables  
*.exe  
*.out  
*.app  
*.i*86  
*.x86_64  
*.hex
```

(13) 불필요한 파일을 관리 대상에서 제외하기 (3)

- 작업 완료 후 커밋함
 - ✓ `git add .gitignore`
 - ✓ `git commit -m "added '.gitignore' file"`

```
$ git add .gitignore
$ git commit -m "added '.gitignore' file"
[hotfix 6de7669] added '.gitignore' file
1 file changed, 54 insertions(+)
create mode 100644 .gitignore
$
```

(14) 충돌 해결하기 (1)

- master 브랜치로 체크아웃한 후 병합을 시도
 - ✓ git checkout master
 - ✓ git merge hotfix
 - ✓ cat helloworld.c

```
$ git checkout master
Switched to branch 'master'
$ git merge hotfix
Auto-merging helloworld.c
CONFLICT (content): Merge conflict in helloworld.c
Automatic merge failed; fix conflicts and then commit the result.
$ cat helloworld.c
#include <stdio.h>

int main(void) {

    printf("Hello World\n");
    printf("Hello Your World\n");

<<<<<<< HEAD
    printf("Hello his World\n");
=====
    printf("Hello her World\n");
>>>>>>> hotfix

    return 0;
}
$
```

(14) 충돌 해결하기 (2)

- vim 편집기로 helloworld.c 편집 (충돌 해결) 후 커밋

- ✓ vim helloworld.c

```
1 #include <stdio.h>
2
3 int main(void) {
4
5     printf("Hello World\n");
6     printf("Hello Your World\n");
7     printf("Hello his World\n");
8     printf("Hello her World\n");
9
10    return 0;
11 }
```

- ✓ git commit -a -m "conflict resolved"

(15) 기록 보기 – git log (1)

■ git log 명령의 옵션

옵션	설명
git log -p	각 커밋에 적용된 실제 변경 내역을 보여 줌
git log --word-diff	diff 명령의 실행 결과를 단어 단위로 보여 줌
git log --stat	각 커밋에서 수정된 파일의 통계 정보를 보여 줌
git log --name-only	커밋 정보 중에서 수정된 파일의 목록만 보여 줌
git log --relative-date	상대적인 시간을 비교형식으로 보여 줌
git log --graph	브랜치 분기와 병합 내역을 아스키 그래프로 보여 줌

(15) 기록 보기 – git log (2)

■ git log --graph 명령

```
$ git log --graph
* commit a83988a6c79fa20fee591bed38ac21fc3704981f
Merge: 2104e30 30e406e
Author: Kisok Kong <kskong@kpu.ac.kr>
Date: Mon Oct 31 00:45:58 2016 +0900

    conflict resolved

* commit 30e406eea9702b4a839af921de638682f9ed7b6d
Author: Kisok Kong <kskong@kpu.ac.kr>
Date: Mon Oct 31 00:44:58 2016 +0900

    added '.gitignore' file

* commit 1f70c9553e244727f5d6edfdd04074774a753494
Author: Kisok Kong <kskong@kpu.ac.kr>
Date: Mon Oct 31 00:43:18 2016 +0900

    added output "Hello her world"

* commit 2104e3012d5df04af7a26059827cc822adf3b241
Author: Kisok Kong <kskong@kpu.ac.kr>
Date: Mon Oct 31 00:42:09 2016 +0900

    added output "Hello his world"

* commit 75fa7dba184e87fe3b1e1bb3130a30b9e552d6c0
Author: Kisok Kong <kskong@kpu.ac.kr>
Date: Mon Oct 31 00:37:59 2016 +0900

    added output "Hello Your World"

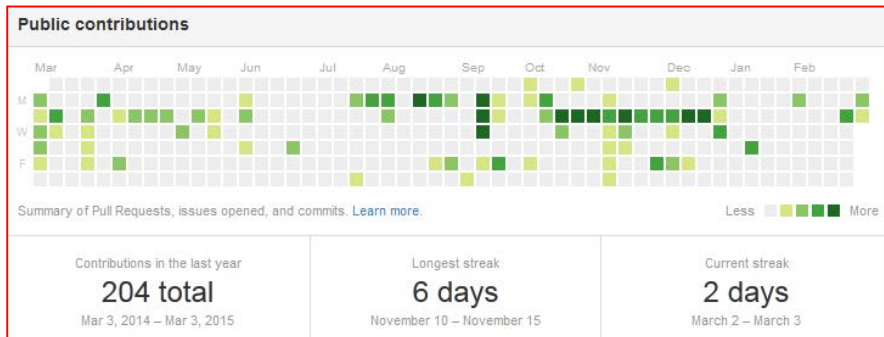
* commit 9731fdc0cd24acb9be7d7a7c94646559c2d92801
Author: Kisok Kong <kskong@kpu.ac.kr>
Date: Mon Oct 31 00:34:39 2016 +0900

    create "Hello World" program

$
```

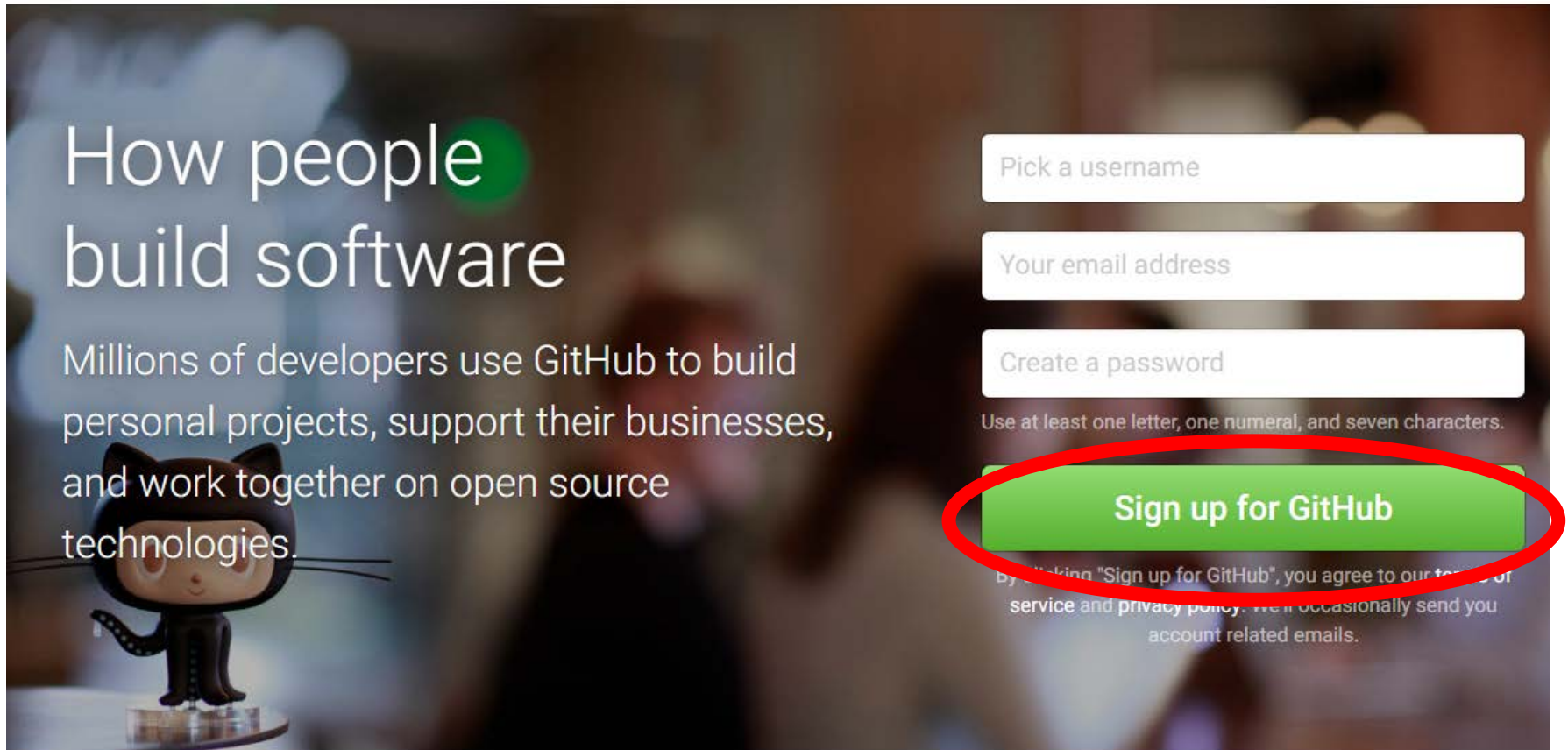

GitHub?

- Git을 기반으로 한 소프트웨어 프로젝트 관리 사이트
- GitHub와 Git의 관계?
 - ✓ Git이라는 Source Control 방법을 GitHub이 사용할 뿐
- GitHub을 쓰는 것은
 - ✓ Source Control
 - ✓ Issue Tracking/Control
 - ✓ 협업도구 (fork/pull request)
 - ✓ Statistics 등을 쓰는 것



GitHub에 가입

■ GitHub 계정 생성 (sign up)



How people build software

Millions of developers use GitHub to build personal projects, support their businesses, and work together on open source technologies.

Pick a username

Your email address

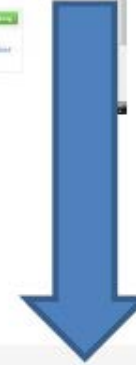
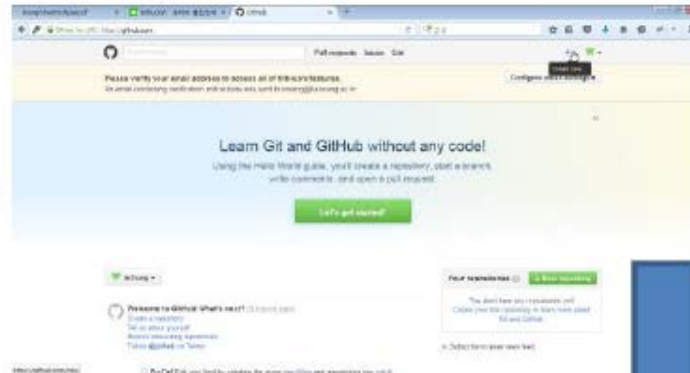
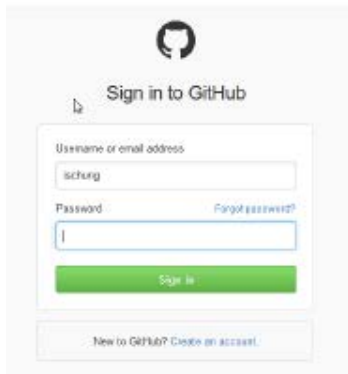
Create a password

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

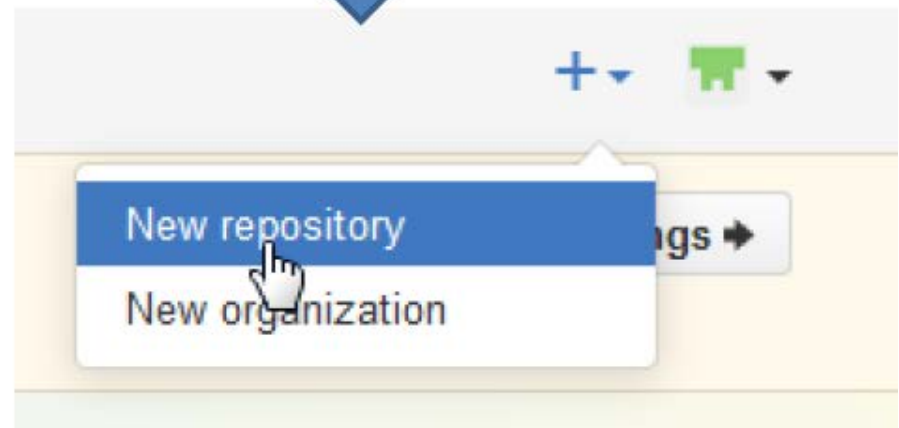
Sign up for GitHub

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

GitHub에서 원격저장소 생성 (1)



가입한 계정으로 로그인(sign in)후
새 원격저장소(New repository)
만들기



GitHub에서 원격저장소 생성 (2)

Owner: kskong / Repository name: test-repo ✓

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

Description (optional): Test Repository

☒ 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 ⓘ

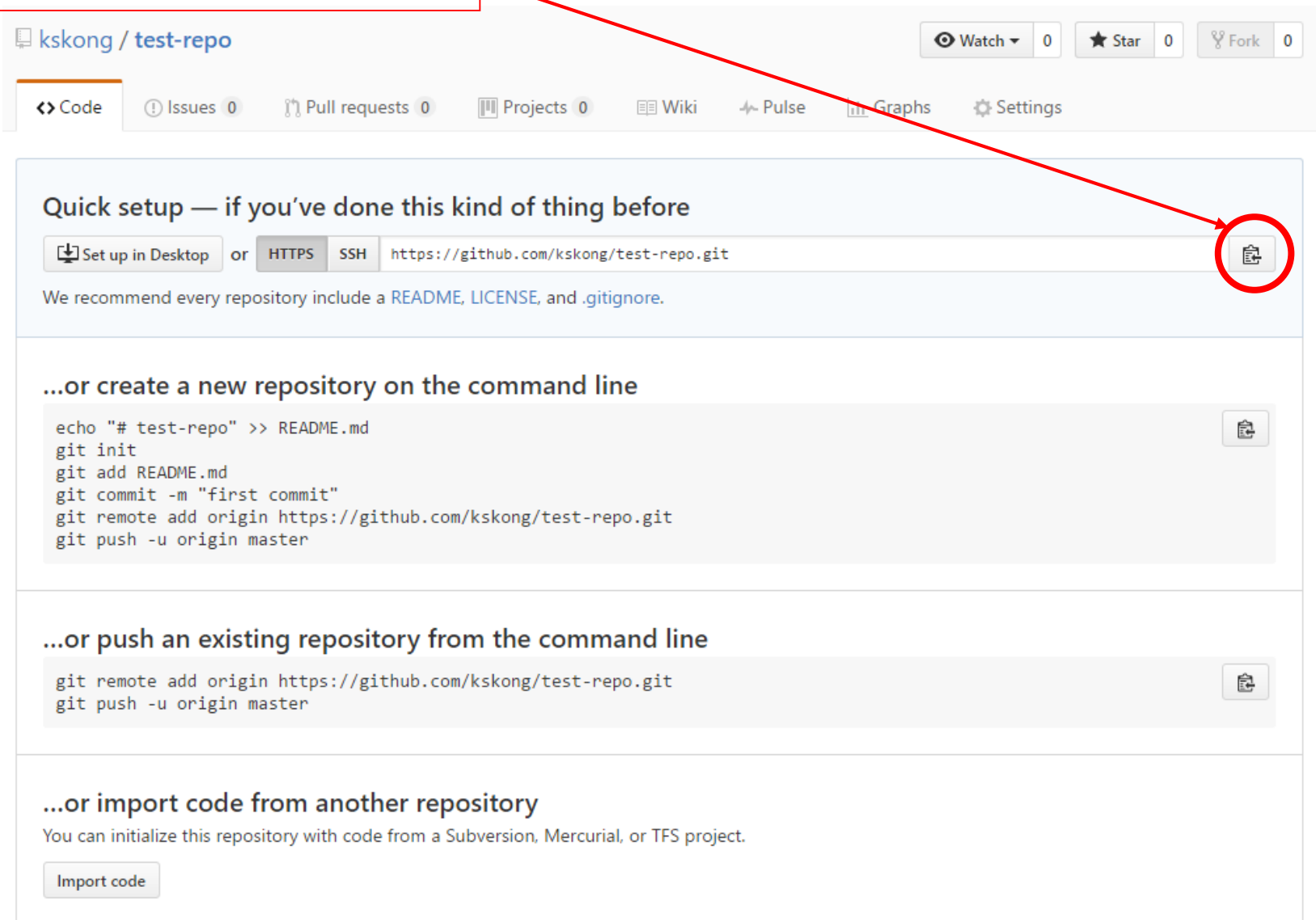
Create repository

저장소 이름과 설명

남들도 볼 수 있게, **private**은 유료

GitHub에서 원격저장소 생성 (3)

원격저장소의 **URL** 주소 복사



kskong / test-repo

Watch 0 Star 0 Fork 0

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

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

Set up in Desktop or **HTTPS** SSH `https://github.com/kskong/test-repo.git`

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

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

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

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

```
git remote add origin https://github.com/kskong/test-repo.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

원격저장소 연결 및 연결 확인

- git remote add <저장소 별칭> <원격저장소 URL>

```
$ git remote add origin https://github.com/kskong/test-repo.git
```

- git remote -v

```
$ git remote -v
origin https://github.com/kskong/test-repo.git (fetch)
origin https://github.com/kskong/test-repo.git (push)
$
```

원격저장소에 작업내용 올리기

- git push <원격저장소 별칭> <로컬 브랜치>
- git push origin --all (로컬의 모든 브랜치를 푸시함)

```
$ git push origin --all
```

```
Username for 'https://github.com': kskong
Password for 'https://kskong@github.com':
Counting objects: 18, done.
Compressing objects: 100% (14/14), done.
Writing objects: 100% (18/18), 1.84 KiB | 0 bytes/s, done.
Total 18 (delta 4), reused 0 (delta 0)
remote: Resolving deltas: 100% (4/4), done.
To https://github.com/kskong/test-repo.git
 * [new branch]      hotfix -> hotfix
 * [new branch]      master -> master
$
```

GitHub에서 확인

The screenshot shows the GitHub interface for a repository named 'test-repo' by user 'kskong'. The repository has 6 commits, 2 branches, 0 releases, and 1 contributor. The 'master' branch is selected. The file list includes '.gitignore' and 'helloworld.c'. A 'Add a README' button is at the bottom.

Settings 메뉴의 **Collaborator** 설정에서
프로젝트 참여자,
즉 이 **repo.**에 **commit** 할 수 있는
사람 **ID**를 추가 가능

Contributor
이 프로젝트 참여자수

master branch 표시

이 **Repository**에서 유지하고 있는
소스 **Branch** 수

kskong / test-repo

Watch 0 Star 0 Fork 0

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

Test Repository — Edit

6 commits 2 branches 0 releases 1 contributor

Branch master New pull request

kskong conflict resolved Latest commit a83988a 5 days ago

.gitignore added '.gitignore' file 5 days ago

helloworld.c conflict resolved 5 days ago

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

원격저장소 파일 수정 (1)

■ test-repo에서 수정할 파일 선택 (helloworld.c)

The screenshot shows the GitHub interface for a repository named 'test-repo' by user 'kskong'. 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 main heading is 'Test Repository — Edit'. Below this, a summary bar shows '6 commits', '2 branches', '0 releases', and '1 contributor'. A row of buttons includes 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and a green 'Clone or download' button. The commit history is displayed below, showing two commits: one for '.gitignore' and another for 'helloworld.c'. The 'helloworld.c' commit is highlighted with a red box. At the bottom, there is a prompt to 'Add a README'.

kskong / test-repo

Watch 0 Star 0 Fork 0

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

Test Repository — Edit

6 commits 2 branches 0 releases 1 contributor

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

kskong conflict resolved Latest commit a83988a 5 days ago

.gitignore	added '.gitignore' file	5 days ago
helloworld.c	conflict resolved	5 days ago

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

원격저장소 파일 수정 (2)

■ helloworld.c 편집 메뉴 선택

kskong / test-repo

Watch 0 Star 0 Fork 0

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

Branch: master test-repo / helloworld.c Find file Copy path

kskong conflict resolved a83988a 5 days ago

1 contributor

12 lines (8 sloc) | 169 Bytes Raw Blame History Edit this file

```
1 #include <stdio.h>
2
3 int main(void) {
4
5     printf("Hello World\n");
6     printf("Hello Your World\n");
7     printf("Hello his World\n");
8     printf("Hello her World\n");
9
10    return 0;
11 }
```

원격저장소 파일 수정 (3)

- helloworld.c에 2라인 추가

test-repo / helloworld.c  or cancel

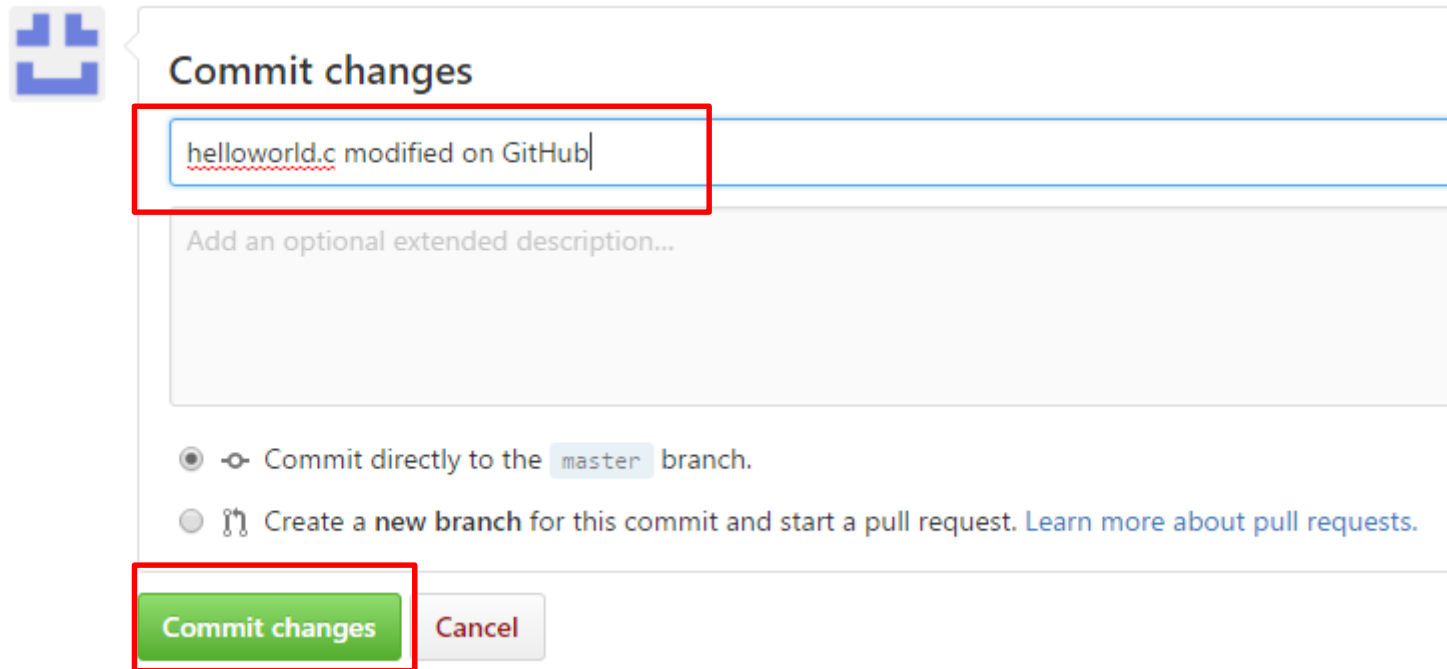
 Edit file


 Preview changes

```
1 // For command line git tutorial
2 // GitHub modification|
3 #include <stdio.h>
4
5 int main(void) {
6
7     printf("Hello World\n");
8     printf("Hello Your World\n");
9     printf("Hello his World\n");
10    printf("Hello her World\n");
11
12    return 0;
13 }
14
```

원격저장소 파일 수정 (4)

- 커밋 메시지 입력 후 “Commit changes”버튼 클릭



 **Commit changes**

helloworld.c modified on GitHub

Add an optional extended description...

☒ Commit directly to the `master` branch.

☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes Cancel

로컬 저장소에서 파일 수정

- 로컬 저장소에서 helloworld.c에 2라인 추가

```
1 // For command line git tutorial
2 // Local repository modification
3 #include <stdio.h>
4
5 int main(void) {
6
7     printf("Hello World\n");
8     printf("Hello Your World\n");
9     printf("Hello his World\n");
10    printf("Hello her World\n");
11
12    return 0;
13 }
```

- 커밋 후 확인

✓ git commit -a -m "helloworld.c modified on Local repository"

```
$ vi helloworld.c
$ git commit -a -m "helloworld.c modified on Local repository"
[master 5d3ebb0] helloworld.c modified on Local repository
 1 file changed, 2 insertions(+)
$ git status
On branch master
nothing to commit, working directory clean
$
```

로컬 저장소에서의 변경사항 푸시

- git push origin master ➔ 푸시 거절됨!!

```
$ git push origin master
Username for 'https://github.com': kskong
Password for 'https://kskong@github.com':
To https://github.com/kskong/test-repo.git
! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/kskong/test-repo.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.
$ █
```

페치 및 병합 (1)

- `git fetch` 명령을 실행해 원격 저장소의 커밋 정보를 로컬 저장소로 가져옴
 - ✓ `git fetch`: 원격 저장소에서 변경한 내용을 확인하고 싶지만, 로컬 저장소에 반영시키고 싶지 않은 경우 사용

```
$ git fetch
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/kskong/test-repo
   a83988a..56cdad4  master    -> origin/master
$ git status
On branch master
nothing to commit, working directory clean
```

- `git branch -va` 명령을 실행하여 로컬 및 원격저장소의 브랜치 정보 확인 → 통합할 브랜치명 확인

```
$ git branch -va
hotfix          30e406e added '.gitignore' file
* master        5d3ebb0 helloworld.c modified on Local repository
remotes/origin/hotfix 30e406e added '.gitignore' file
remotes/origin/master 56cdad4 helloworld.c modified on GitHub
$
```

페치 및 병합 (2)

- git merge origin/master 명령 실행 → 충돌 발생!!

```
$ git merge oriqin/master
Auto-merging helloworld.c
CONFLICT (content): Merge conflict in helloworld.c
Automatic merge failed; fix conflicts and then commit the result.
$
```

- git diff 명령을 사용하여 변경사항 확인

```
$ git diff
diff --cc helloworld.c
index fac6cc7,b211562..0000000
--- a/helloworld.c
+++ b/helloworld.c
@@@ -1,5 -1,5 +1,9 @@@
 // For command line git tutorial
++<<<<<<< HEAD
+// Local repository modification
++=====
+ // GitHub modification
++>>>>>>> origin/master
 #include <stdio.h>

int main(void) {
$
```

-. 삭제된 부분

+: 추가된 부분

페치 및 병합 (3)

- vim helloworld.c 로 변경사항 수정

```
1 // For command line git tutorial
2 // First: GitHub modification
3 // Secon: Local repository modification
4 #include <stdio.h>
5
6 int main(void) {
7
8     printf("Hello World\n");
9     printf("Hello Your World\n");
10    printf("Hello his World\n");
11    printf("Hello her World\n");
12
13    return 0;
14 }
```


- 커밋 후 푸시

```
$ git commit -a -m "conflict resolved GitHub"
[master b2fd68e] conflict resolved GitHub
$ git status
On branch master
nothing to commit, working directory clean
$ git push origin master
Username for 'https://github.com': kskong
Password for 'https://kskong@github.com':
Counting objects: 6, done.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 718 bytes | 0 bytes/s, done.
Total 6 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 1 local objects.
To https://github.com/kskong/test-repo.git
   56cdad4..b2fd68e  master -> master
$
```


페치 및 병합 (4)

■ 원격저장소에서 병합 여부 확인

Branch: master ▾ test-repo / helloworld.c Find file Copy path

 kskong conflict resolved GitHub b2fd68e 20 minutes ago

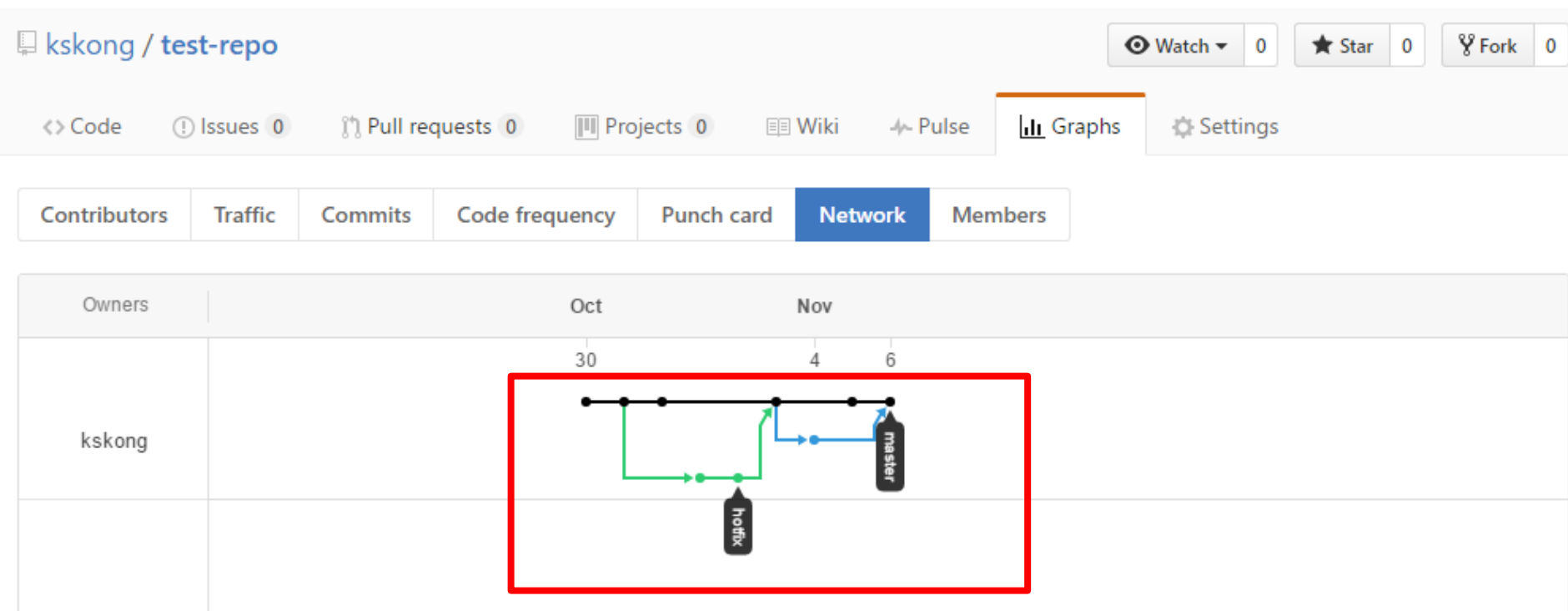
1 contributor

15 lines (11 sloc) | 272 Bytes Raw Blame History   

```
1 // For command line git tutorial
2 // First: GitHub modification
3 // Secon: Local repository modification
4 #include <stdio.h>
5
6 int main(void) {
7
8     printf("Hello World\n");
9     printf("Hello Your World\n");
10    printf("Hello his World\n");
11    printf("Hello her World\n");
12
13    return 0;
14 }
```

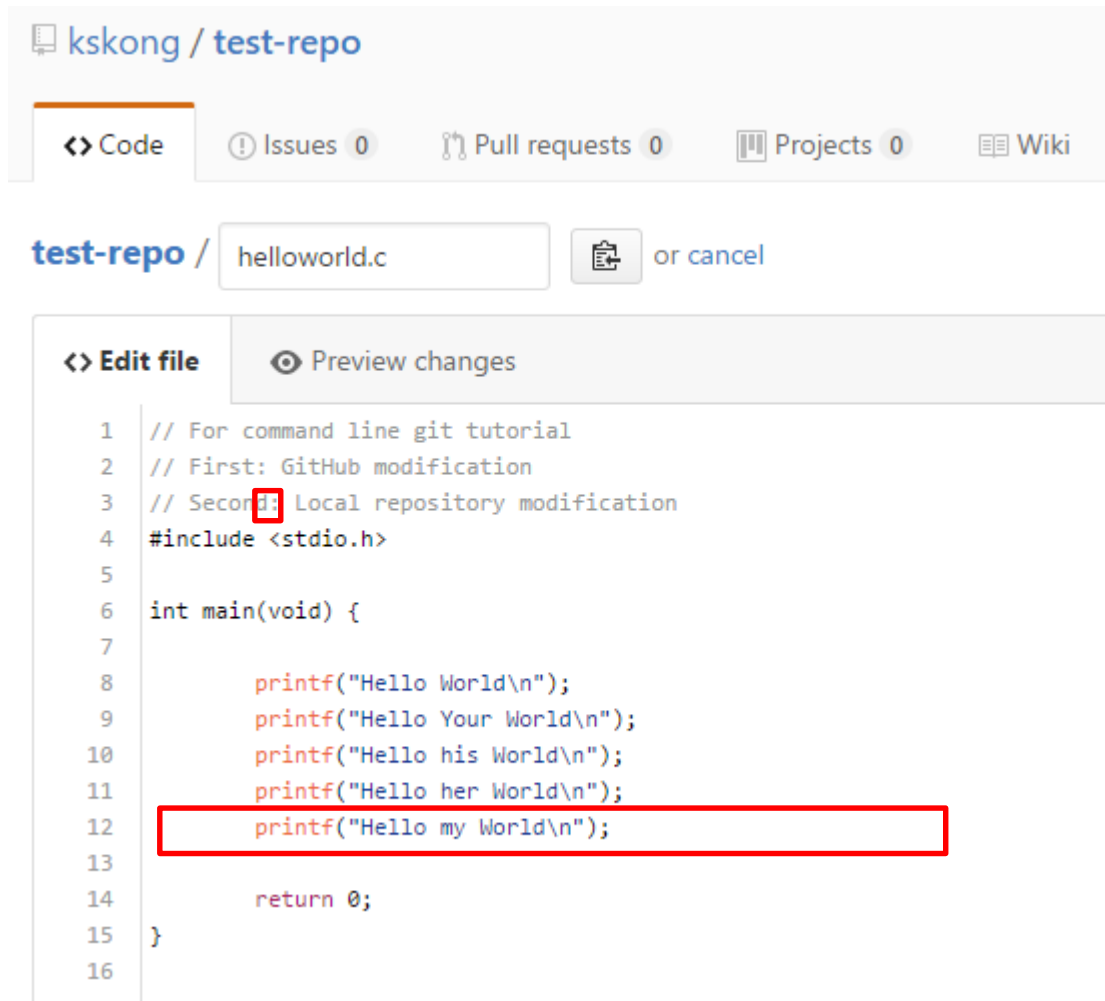
페치 및 병합 (5)

- network 항목의 변경내역 그래프를 통한 확인
 - ✓ 원격저장소 우측 상단의 [Graph] → [Network]를 선택해서 로컬의 master 브랜치의 변경내역이 GitHub master에 반영된 것을 확인



git pull 명령 (1)

- 원격저장소에서 helloworld.c 수정



kskong / test-repo

<> Code ! Issues 0 🔗 Pull requests 0 📁 Projects 0 📖 Wiki

test-repo / helloworld.c 📄 or cancel

<> Edit file 👁 Preview changes

```
1 // For command line git tutorial
2 // First: GitHub modification
3 // Second: Local repository modification
4 #include <stdio.h>
5
6 int main(void) {
7
8     printf("Hello World\n");
9     printf("Hello Your World\n");
10    printf("Hello his World\n");
11    printf("Hello her World\n");
12    printf("Hello my World\n");
13
14    return 0;
15 }
16
```

git pull 명령 (2)

- 지역저장소에서 git pull 명령 실행 후 확인

```
$ git pull origin master
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/kskong/test-repo
 * branch                master      -> FETCH_HEAD
    b2fd68e..bfe01b8      master      -> origin/master
Updating b2fd68e..bfe01b8
Fast-forward
 helloworld.c | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
$ cat helloworld.c
// For command line git tutorial
// First: GitHub modification
// Second: Local repository modification
#include <stdio.h>

int main(void) {

    printf("Hello World\n");
    printf("Hello Your World\n");
    printf("Hello his World\n");
    printf("Hello her World\n");
    printf("Hello my World\n");

    return 0;
}
```

GitHub를 사용한 협업 시 작업 흐름

1. 프로젝트 팀장이 빈 원격저장소를 만들고 로컬 저장소에 있는 소스 파일을 원격저장소로 **push** 함
2. 프로젝트 팀장 (원격저장소 관리자)이 원격저장소에 있는 **[Settings]** 메뉴를 클릭하여 협업자(**Collaborator**, 프로젝트 팀원의 아이디)를 추가함
3. 협업자(팀원)은 **GitHub** 아이디 생성시 등록한 이메일 계정으로 원격저장소의 링크가 전송됨
4. 팀원은 프로젝트의 원격저장소를 로컬저장소로 복제 (**clone**)하여 소스 코드 수정, 추가 작업 수행후 커밋 (별도의 브랜치를 만들어서 작업한 후 병합해도 됨)
5. 원격저장소로 **push**
6. 충돌 발생시 원격저장소 **fetch**, 병합 후 다시 **push**

원격저장소의 복제(clone)

■ 작업 예시

- ✓ mkdir github_tutorial
- ✓ cd github_tutorial
- ✓ git clone <https://github.com/kskong/test-repo.git>
- ✓ cd test-repo
- ✓ git status
- ✓ ...

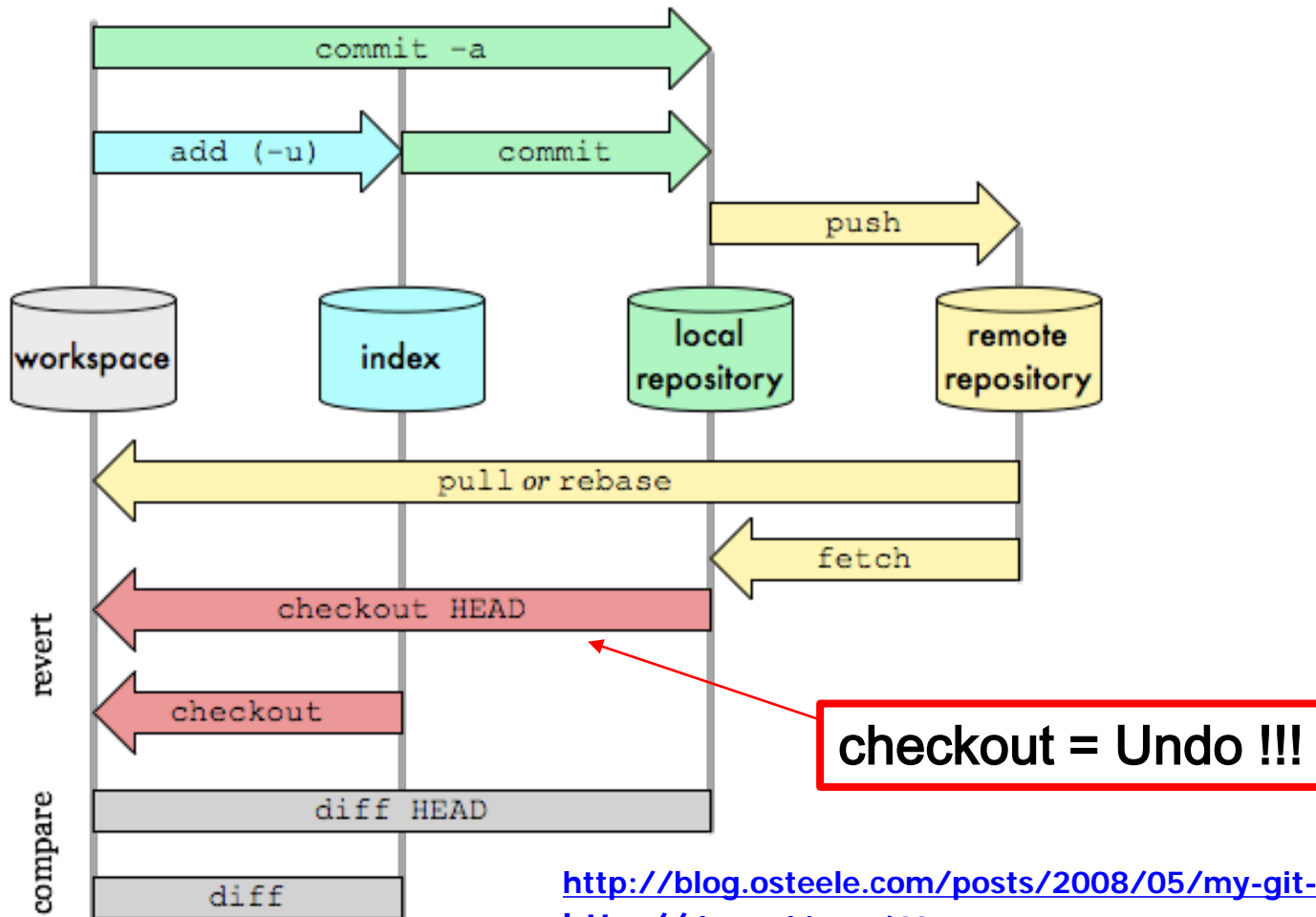
유용한 git 명령어

- **git tag** : 커밋을 참조하기 쉽도록 알기 쉬운 이름을 붙임
 - ✓ 예: git tag v1.0
- **git commit --amend** : 마지막 커밋 수정하기
 - ✓ 기존 커밋 메시지의 수정
- **git revert** : 공개된 커밋의 변경된 내역을 되돌리기
 - ✓ git revert HEAD (마지막 커밋 내용을 되돌리고 커밋하기)
 - ✓ git revert <커밋 ID 또는 태그 이름> (커밋 아이디는 SHA-1 체크섬값 앞 네자리 숫자. 특정 커밋 시점으로 되돌리고 커밋하기)
 - ✓ git revert는 이전 커밋을 남겨 둠
- **git reset** : 어떤 커밋을 지우고 특정 버전으로 되돌림
 - ✓ git reset --soft HEAD~~~ (이전 세개의 커밋을 되돌림. ^ 혹은 ~은 커밋 하나를 의미. 커밋만 지우고 파일 내용은 그대로 둠)
 - ✓ git reset --hard HEAD~~~ (이전 세개의 커밋을 지우고 파일 변경 내역도 삭제)
- **git checkout HEAD -- <filename>** : 특정파일을 최종 커밋 시점으로 되돌리기
 - ✓ git checkout HEAD -- helloworld.c
 - ✓ git checkout <커밋 아이디> -- helloworld.c

git 작업 흐름(workflow)

Git Data Transport Commands

<http://osteele.com>



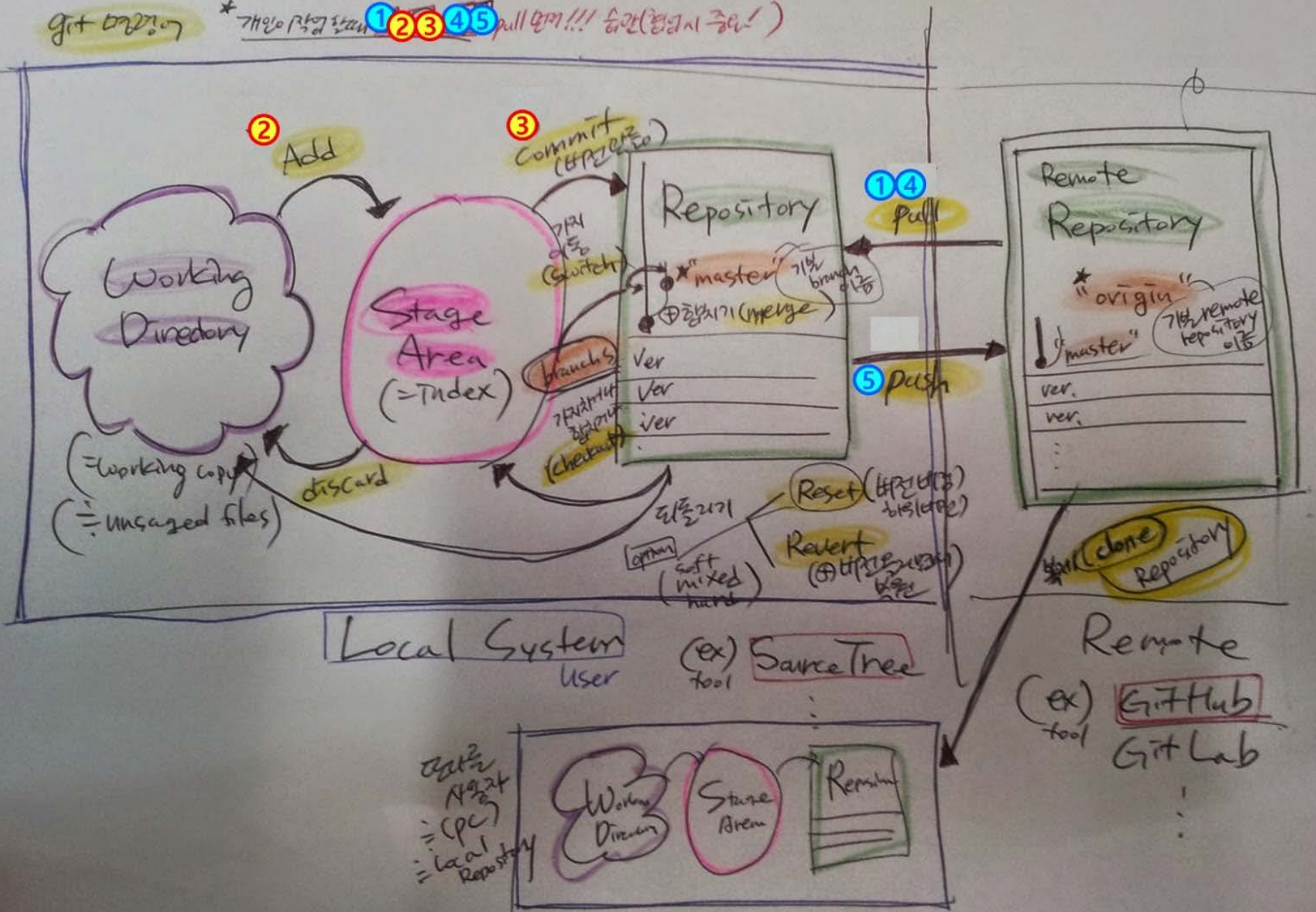
<http://blog.osteele.com/posts/2008/05/my-git-workflow/>
<http://dev.azki.org/40>

Git (By Funny Rella)


<http://funnyrella.blogspot.kr/2014/04/97-git.html>

git 명령어

* 개인이 작업할때 1 2 3 4 5 pull 하기!!! 습관(절대시 중요!)



생활 코딩의 git 강좌 (<https://opentutorials.org/course/1492>)



Git : 버전관리란 무엇일까요?

생활코딩 구독 32,424

조회수 24,523회

+ 추가 공유 더보기

게시일: 2014. 9. 22.

생활코딩은 일반인에게 프로그래밍을 알려주는 활동입니다. 더 많은 정보를 원하시면 <http://opentutorials.org/course/1>를 방문해주세요.

더보기

GIT

생활코딩 • 1/동영상 29개

- 1. Git : 버전관리란 무엇일까요? 생활코딩
- 2. Git : Git과 SourceTree 설치 - 윈도우 생활코딩
- 3. Git : Git과 SourceTree 설치 - OSX 생활코딩
- 4. Git - 수업 예제 설명 생활코딩

Git & SourceTree 설치 Windows 설치 <https://opentutorials.org/module/1213> Git 4:01

6. 버전관리와 Git 생활코딩 opentutorials.org/course/596 46:19

Git : 버전만들기 (commit) 생활코딩 조회수 16,609회

<https://youtu.be/XUEuYq64HKI?list=PLuHgQVnccGMCB06JE7zFIAOJtdcZBVrap>

Git/GitHub 고수가 되려면?

- git rebase 명령 익히기
- GitHub 협업도구 사용하기
 - ✓ Pull Request
 - GitHub 포크(fork)기능을 이용한 협업 시 보내는 병합요청
 - ✓ Issue Tracker
 - 버그 보고, 기능개선 건의, 기타 프로젝트 관련 이슈 등록 공간
 - ✓ Wiki
 - 특정 주제나 단어 등에 대한 정보
 - Markdown 문법을 사용하여 작성
 - ✓ Code Review
 - 커밋이나 개별 코드에 대해 댓글 추가 가능

GitHub의 적극적인 사용이 중요:

1. 샘플 코드를 **GitHub**에서 **fork / clone** 할 것 !!
2. 과제 소스를 **GitHub**의 자기 **repository**에 **push** 할 것!!
3. 모든 프로젝트는 반드시 **repository, issue tracking** 할 것!!

참고 사이트

- “git – 간편 안내서”, <http://rogerdudler.github.io/git-guide/index.ko.html> (초보자용 기초 정보 제공)
- “누구나 쉽게 이해할 수 있는 Git 입문“, <http://backlogtool.com/git-guide/kr/> (실습을 통한 학습)
- “GIT – 생활코딩“, <https://opentutorials.org/course/1492> (동영상 학습 사이트) (youtube 사이트 주소: https://youtu.be/N_rpDCZxRCY?list=PLuHgQVnccGMCB06JE7zFIAOJtdcZBVrap)
- “Git – Book“, <https://git-scm.com/book/ko/v2> (Git 개발자들이 쓴 Git/GitHub 설명서인 *ProGit* 2판의 웹 버전, 중급이상의 실력자를 위한 책)
- 윤웅식, 만들면서 배우는 Git GitHub 입문, 한빛미디어, 2015. (교재 스타일 학습서)
- “Git 작업 흐름과 명령어“, <http://www.insightbook.co.kr/wp-content/uploads/2013/04/git-치트시트프린트.pdf> (명령어 요약집)

Q & A