
오픈소스 소프트웨어 개발 - 15

(Online) Git & Github - remote

광운대학교 이윤상
2017년 2학기

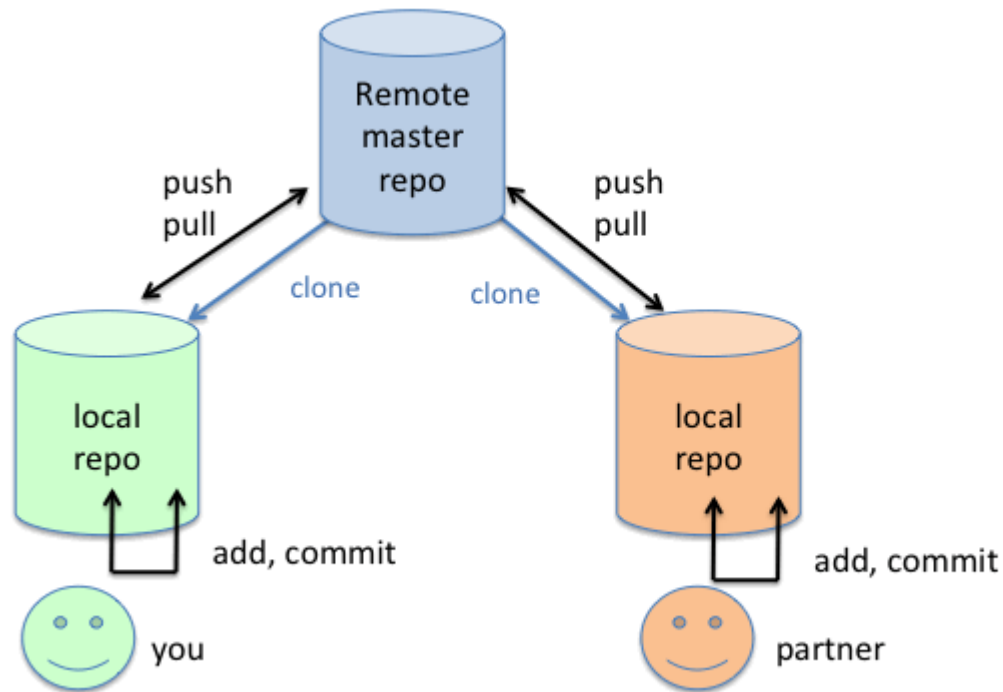
이번 시간에 할 것

- 원격저장소란?
- Github 원격저장소 만들기, 원격저장소 작업 시작하기
- 원격저장소로/에서 변경사항 보내기/가져오기
- git push 및 git pull 시 발생하는 상황들
- 원격저장소 작업 시 주의사항

원격저장소(Remote Repository)란?

- 원격저장소(remote repository)
 - 인터넷이나 네트워크 어딘가에 있는 서버에 있는 Git 저장소
- 지역저장소(local repository)
 - Local machine에 저장되어 있는 Git 저장소 (지금까지 강의에서 만들어서 사용한 저장소)

원격저장소(Remote Repository)란?



(리뷰) 버전 관리 시스템의 효용

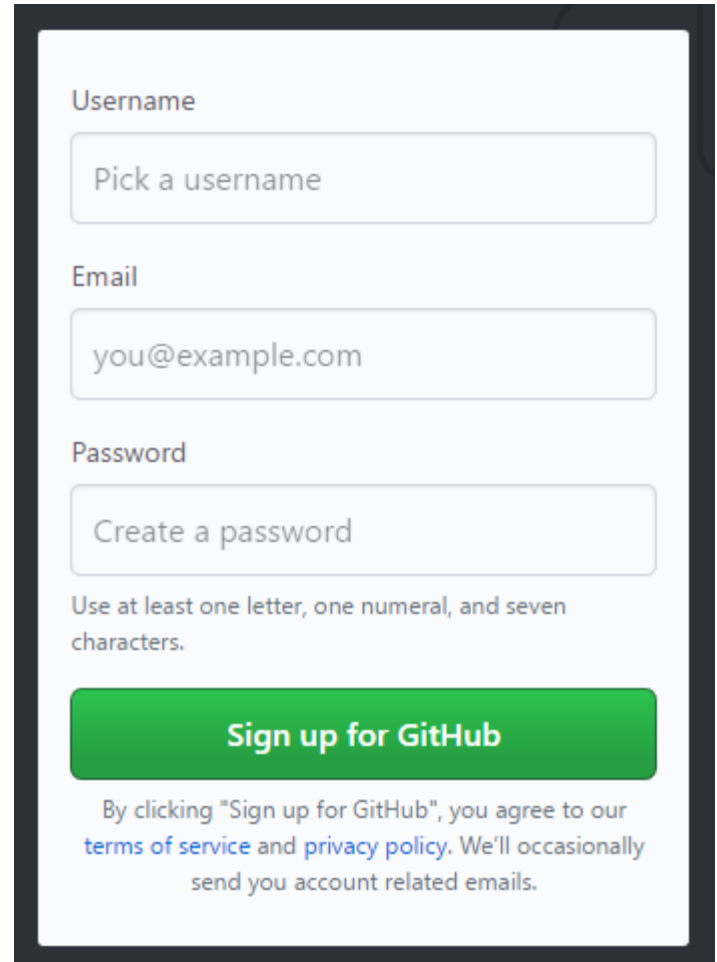
- 변경 내용 추적
 - 과거 특정 시점의 상태로 복원
 - 예기치 못한 사고에 대비한 백업
 - 효과적인 협업
-
- -> Git의 경우, 아래의 두 항목은 원격저장소를
사용해야 얻을 수 있는 효과

원격저장소를 운영하는 방법

- 크게 아래 두 가지로 생각할 수 있다.
- Git 서버를 직접 설치해서 운영
 - 서버로 쓸 수 있는 남는 컴퓨터가 있다면 시도해 볼 수 있다.
- Git 호스팅 서비스를 이용
 - Github, bitbucket 등
- 본 강좌에서는 Github의 원격저장소를 이용하도록 한다.

[실습] Github 시작하기

- <https://github.com/>
- 계정이 없다면 계정 새로 만들기

A screenshot of the GitHub sign-up form. It features three input fields: 'Username' with the placeholder 'Pick a username', 'Email' with the placeholder 'you@example.com', and 'Password' with the placeholder 'Create a password'. Below the password field is a note: 'Use at least one letter, one numeral, and seven characters.' A prominent green button labeled 'Sign up for GitHub' is positioned below the form. At the bottom, a disclaimer states: 'By clicking "Sign up for GitHub", you agree to our terms of service and privacy policy. We'll occasionally send you account related emails.'

Welcome to GitHub

You've taken your first step into a larger world, @yssl-test.



Completed

Set up a personal account



Step 2:

Choose your plan



Step 3:

Tailor your experience

Choose your personal plan



Unlimited public repositories for free.



Unlimited private repositories for \$7/month.

Don't worry, you can cancel or upgrade at any time.

☐ Help me set up an organization next

Organizations are separate from personal accounts and are best suited for businesses who need to manage permissions for many employees.

[Learn more about organizations](#)

☐ Send me updates on GitHub news, offers, and events

Unsubscribe anytime in your email preferences. [Learn more](#)

Continue

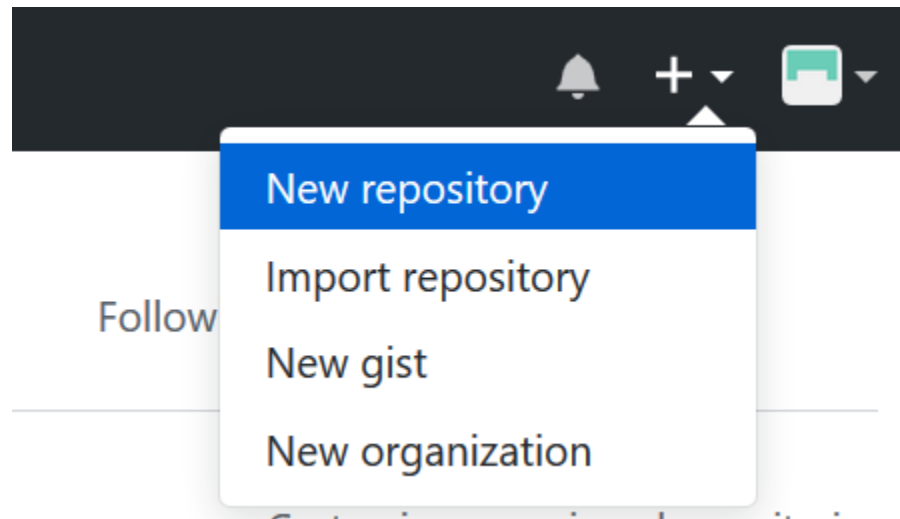
Both plans include:

- ✓ Collaborative code review
- ✓ Issue tracking
- ✓ Open source community
- ✓ Unlimited public repositories
- ✓ Join any organization

- Continue 누르면 verification email이 발송되는데, 해당 메일에
서 verify를 눌러주면 된다.

[실습] Github에 원격저장소 만들기

- Github 로그인 - 오른쪽 상단 "+" - New repository 선택



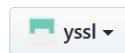
[실습] Github에 원격저장소 만들기

- 저장소의 이름을 입력하고 Create Repository 누름.
- 이후 실습에서는 'TestRepo'라는 이름의 저장소를 만든 것으로 가정

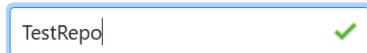
Create a new repository

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

Owner



Repository name



Great repository names are short and memorable. Need inspiration? How about **solid-robot**.

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

이번 강의의 실습에 대해...

- 혼자서 원격저장소를 백업 용도로만 사용한다면, 단순히 이후에 소개될 pull & push를 반복하며 작업하면 됨.
- 여러 명이 협업을 하는 경우, 원격저장소 사용시 여러 문제 상황이 발생할 수 있는데 이 부분을 살펴볼 것임.
- 따라서 이번 강의에서는 여러 명이 하나의 원격저장소를 사용하는 과정을 가정한 실습을 진행할 것임.
- 실제로 여러 명의 Github ID를 이용하여 실습을 진행하지는 않고, 본인의 Github ID 하나만 사용하되 두 군데의 local directory(TestRepo-My, TestRepo-David)에서 작업하는 것으로 두 명의 개발자가 협업을 하는 상황을 흉내내는 실습을 진행할 것임.

원격저장소 작업 시작하는 두 가지 방법

- 1 - 이미 만들어진 지역저장소에 원격저장소를 등록하여 (`git remote add` 이용) 작업하기.
- 2 - 이미 만들어진 원격저장소를 지역저장소로 복제하여 (`git clone` 이용) 작업하기.

[실습] 1 - 이미 만들어진 지역저장소에 원격저장소를 등록

- **git remote add** <원격저장소 이름> <원격저장소 주소> : 현재 지역저장소에 원격저장소를 추가한다.
- **git remote** : 현재 지역저장소에 추가되어 있는 원격저장소 목록을 출력 (자세한 정보는 -v 추가)

(Shell)

```
mkdir -p ~/github-1/TestRepo-My #디렉터리 이름, 위치는 변경가능  
cd ~/github-1/TestRepo-My
```

```
git init  
git remote add origin https://github.com/(id)/TestRepo.git  
  
git remote -v
```

[실습] 원격저장소로 변경사항 보내기

- **git push** <원격저장소 이름> <branch 이름> : 해당 branch의 commit 내역을 해당 원격저장소로 보냄

(Shell)

```
vi file1.txt # Add 'hello'
```

```
git add .
```

```
git commit
```

```
git log --branches --decorate --graph --oneline
```

```
git push origin master # master의 commit내용을 origin의  
master로 보내기
```

```
git log --branches --decorate --graph --oneline
```

```
# Github의 해당 저장소 사이트에서 해당 변경사항이 업데이트 된 것  
을 확인
```

(참고) SSH Key 설정

- 다음과 같이 origin을 SSH url로 바꾸고 SSH key를 설정하면 push 할 때 매번 ID, PW를 입력할 필요 없음.

(Shell)

```
git remote set-url origin git@github.com:(id)/TestRepo.git
```

```
ssh-keygen
```

(출력메시지)

Generating public/private rsa key pair.

Enter file in which to save the key

(/home/schacon/.ssh/id_rsa): # 엔터

Created directory '/home/schacon/.ssh'.

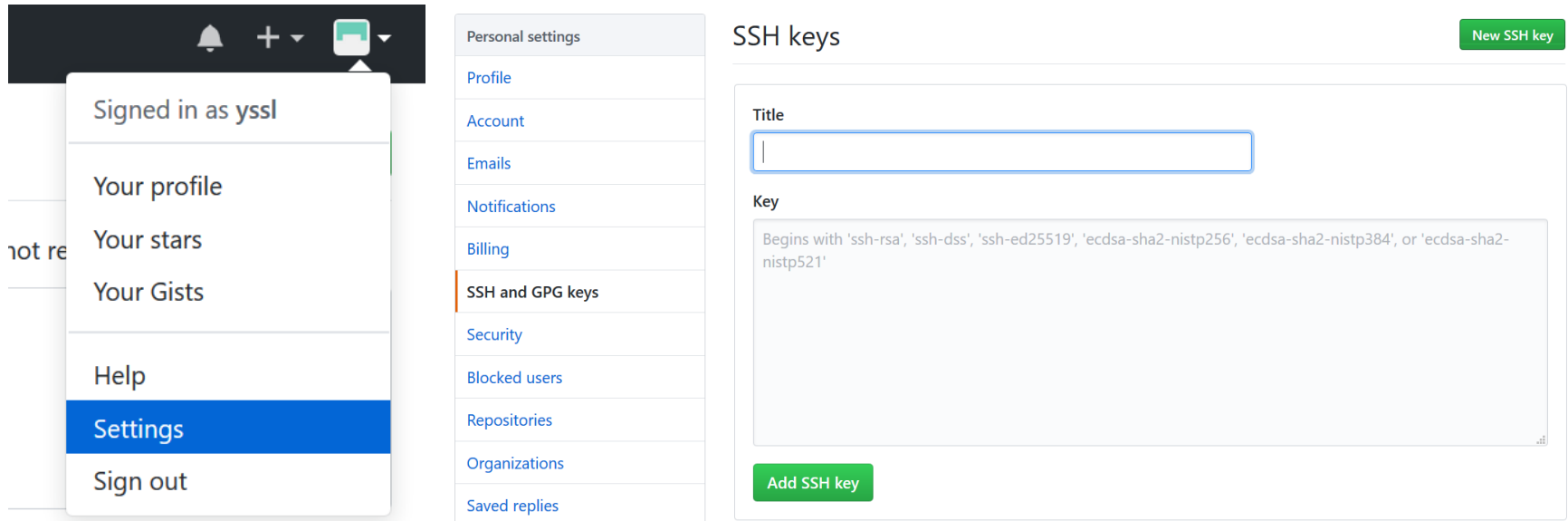
Enter passphrase (empty for no passphrase): # 암호문 입력

Enter same passphrase again: # 암호문 입력

```
cat ~/.ssh/id_rsa.pub
```

출력되는 내용을 복사해서...

(참고) SSH Key 설정



- 적당히 Title 적고 Key 부분에 복사한 내용 붙여넣고 Add SSH key
- git push를 해보면 Github ID, PW를 요구하지 않는 것을 확인할 수 있다.
- 이후 온라인 강의에서는 ssh url을 이용할 것이나, 각자 실습할 때는 본인의 선호도에 따라 https url을 이용해도 무방

(참고) --set-upstream, push.default 설정

(Shell)

```
git push # 그냥 git push만 하면 에러 발생
```

```
# git push만 입력해도 master의 commit내용을 origin의 master로 보내도록 설정
```

```
git push --set-upstream origin master
```

```
# git push에서 branch를 지정하지 않을 때의 기본 동작을 설정
```

```
# simple : 현재 branch만 push
```

```
# matching : 원격저장소와 지역저장소에 동일한 이름을 가지는 branch를 모두 push
```

```
# simple로 지정하도록 하자.
```

```
git config --global push.default simple
```

[실습] 2 - 이미 만들어진 원격저장소를 지역저장소로 복제

- **git clone** <원격저장소 주소> : 원격저장소를 local machine의 지역저장소로 복제해옴
 - 해당 원격저장소는 'origin'이라는 이름으로 자동 추가됨

(Shell)

```
cd ~/github-1 #디렉터리 이름 및 위치는 자유롭게
```

```
git clone git@github.com:(id)/TestRepo.git # ssh 주소를  
clone하면 앞에서 설정한 ssh key를 통해 push시 자동로그인
```

```
mv TestRepo TestRepo-David  
cd TestRepo-David  
git remote -v
```

```
# TestRepo-My와 TestRepo-David의 working directory와 local  
repository가 모두 같은지 ls와 git log로 확인해보자.
```

[실습] 원격저장소에서 변경사항 가져오기

- **git pull** : 원격저장소의 변경사항을 지역저장소로 가져오고 working directory에 반영

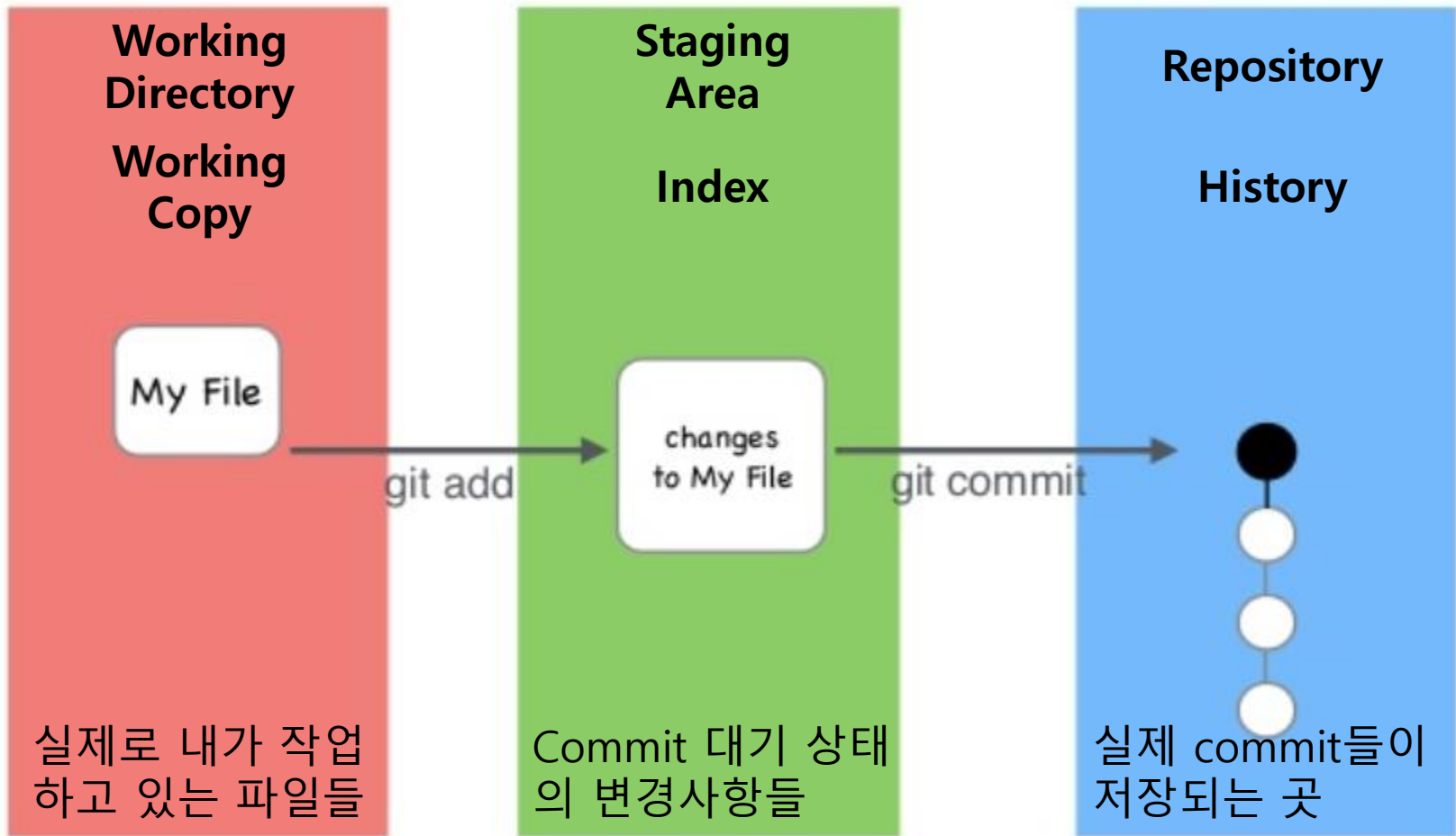
(Shell: TestRepo-David)

```
vi file2.txt # Add 'dog'  
git add .  
git commit  
git push
```

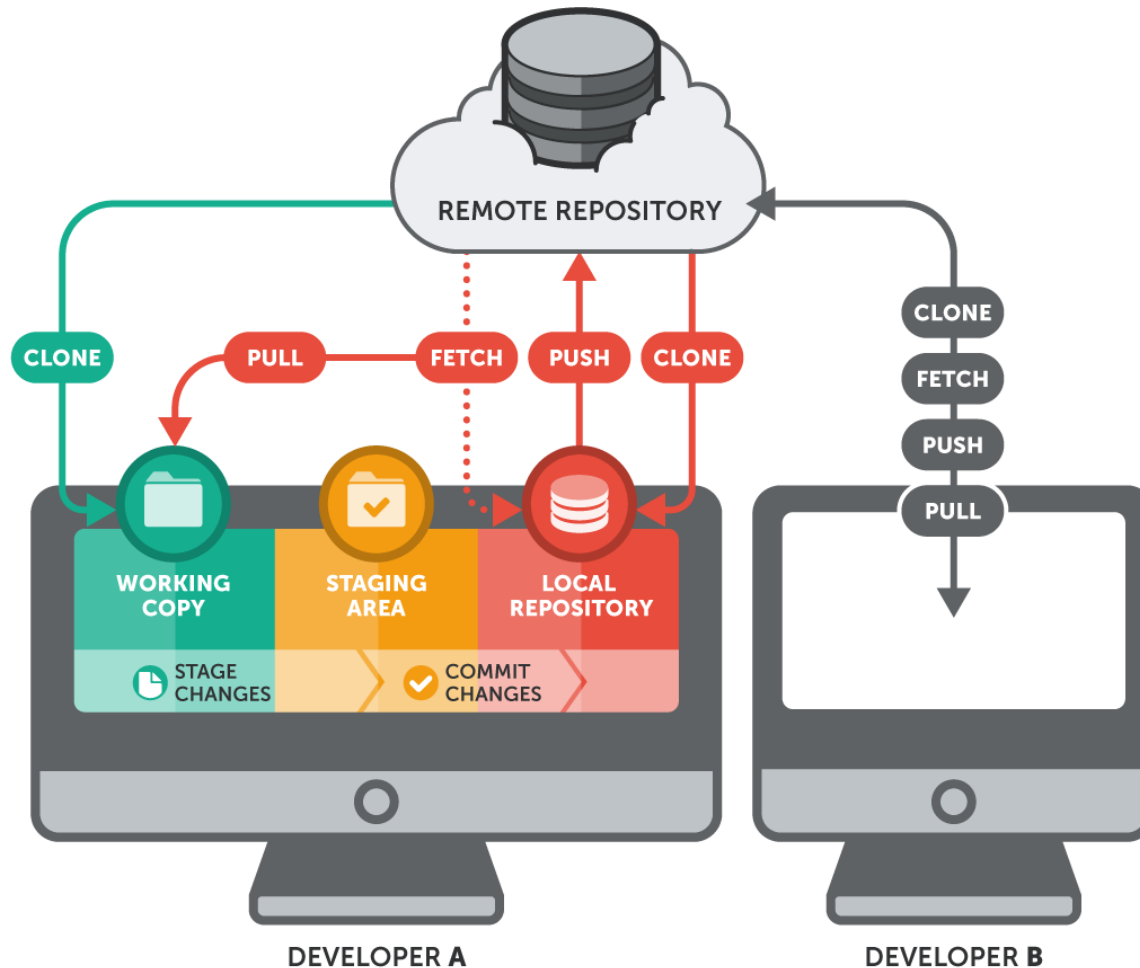
(Shell: TestRepo-My)

```
git log --branches --decorate --graph --oneline  
git pull  
git log --branches --decorate --graph --oneline
```

Git 작업 공간의 분류



Git Clone, Pull, Push



[실습] git push중 오류 발생

(Shell: TestRepo-David)

```
vi file1.txt # Add 'hi'
git commit -a -m "Add hi"
git log --branches --decorate --graph --oneline
git push
git log --branches --decorate --graph --oneline
```

(Shell: TestRepo-My)

```
vi file2.txt # Add 'cat'
git commit -a -m "Add cat"
git log --branches --decorate --graph --oneline
git push
(출력메시지)
To git@github.com:yssl/TestRepo.git
 ! [rejected]          master -> master (fetch first)
error: failed to push some refs to
'git@github.com:yssl/TestRepo.git'
...
```

[실습] git push중 오류 발생

- 내가 pull한 이후로 아무도 push하지 않았을 경우에만 바로 push가 가능
- 그렇지 않은 경우, push하기 전에 pull을 해서 그 사이 발생한 변경사항을 merge한 후 push가 가능
- **push 하기 전에는 항상 pull을 먼저 하는 습관!**

(Shell: TestRepo-My)

```
git pull
git log --branches --decorate --graph --oneline

git push
git log --branches --decorate --graph --oneline
```

git pull의 의미

- git pull = git fetch + git merge
- 변경사항을 가져와서(fetch) 병합(merge) 함.
- 따라서 branch merge 할 때와 마찬가지로 **fast-forward merge** 혹은 **3-way merge**를 하게 되고, 3-way merge 중 **conflict**이 발생할 수 있다.
- conflict이 발생하면 branch merge 할 때와 마찬가지로 방법으로 resolve하면 된다.

[실습] git pull 시 fast-forward merge

(Shell: TestRepo-David)

```
git pull
vi file1.txt # Add 'bye'
git commit -a -m "Add bye"
git push
```

(Shell: TestRepo-My)

```
git log --branches --decorate --graph --oneline
git pull
(출력메시지)
...
Fast-forward
 file1.txt | 1 +
 1 file changed, 1 insertion(+)
git log --branches --decorate --graph --oneline
```

[실습] git pull 시 3-way merge

(Shell: TestRepo-David)

```
git pull
vi file1.txt # Add new lines
git commit -a -m "Add david's new lines"
git push
```

(Shell: TestRepo-My)

```
vi file2.txt # Add new lines
git commit -a -m "Add my new lines"
git log --branches --decorate --graph --oneline
git pull
(출력메시지)
...
Merge made by the 'recursive' strategy.
 file2.txt | 1 +
 1 file changed, 1 insertion(+)
git log --branches --decorate --graph --oneline
```

[실습] 3-way merge 중 conflict

(Shell: TestRepo-David)

```
git pull
vi file1.txt # hello -> Hello
git commit -a -m "Change hello to Hello"
git push
```

(Shell: TestRepo-My)

```
vi file1.txt # hello -> HELLO
git commit -a -m "Change hello to HELLO"
git log --branches --decorate --graph --oneline
git pull
(출력메시지)
```

...

CONFLICT (content): Merge conflict in file1.txt
Automatic merge failed; fix conflicts and then commit the result.

- 13-(Online)Git-branch2 강의에서와 마찬가지로 file.txt에서 >>>>, <<<<, ===== 등을 지우고 제대로 고쳐서 저장한 후 add & commit 하면 된다.

원격저장소 작업 시 기억해야 할 사항

- 이미 push한 commit은 수정 / 삭제하지 말 것
 - commit --amend으로 commit message 수정도 불가능
 - 굳이 하려면 방법은 있지만, 이미 해당 commit을 pull한 개발자들의 저장소의 history가 꼬이는 문제가 발생하므로 절대 금물
- 새로운 작업을 시작하기 전에 항상 git pull을 먼저 할 것
- 작업이 끝나면 꼭 git push를 할 것

원격저장소 이름 변경, 삭제

- `git remote rename` : 원격저장소 이름 변경 자세한 내용은 `git help` 참조.
- `git remote remove` : 원격저장소 삭제. 자세한 내용은 `git help` 참조.