오픈소스 소프트웨어 개발 – 11

(Online) Git - basic 1

광운대학교 이윤상
2017년 2학기

이번 시간에 할 것

- Git 설치 및 최초 설정
- 저장소 만들기
- 수정하고 저장소에 저장하기
  - 새로운 추적 대상 등록, Commit 하기, Commit message의 의미
- Git 작업 공간의 분류

Git 설치

- Ubuntu

(Shell)

```
sudo apt-get install git
```

- Windows
  - https://git-scm.com/download/win
  - 다운로드 후 설치

- Windows에서의 사용법은 Ubuntu에서와 같기 때문에 본 강의에서는 Ubuntu에서만 실습을 진행할 예정

- 여러 Git GUI client들이 있지만, 기본으로 설치되는 CLI(command line interface) client로 진행할 예정

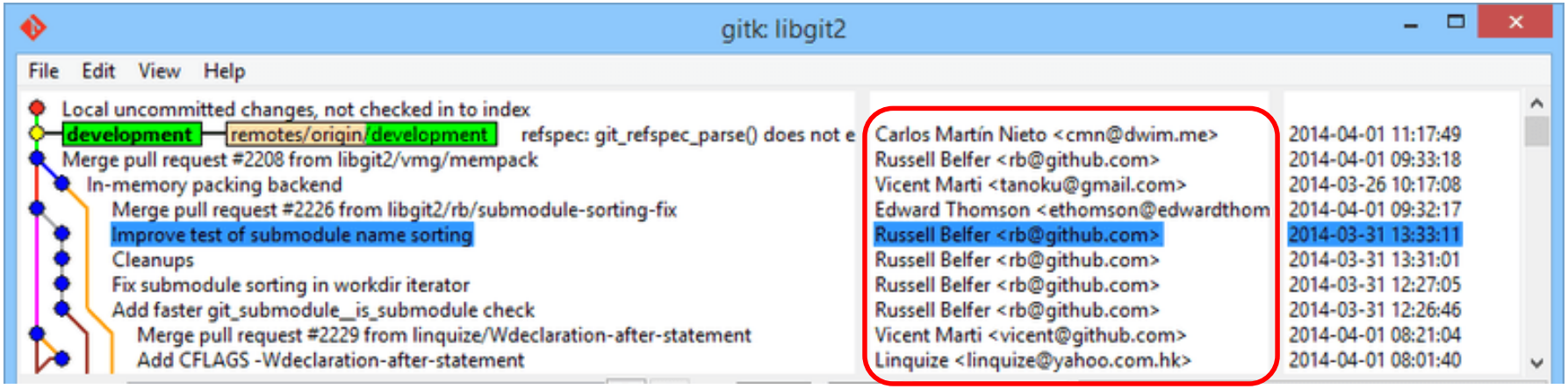
[실습] 최초 설정 - 사용자 이름, 이메일 주소 설정

- 아래 회색 부분을 본인의 정보로 바꾸어 입력

(Shell)

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

- 저장소마다 다르게 설정하고 싶다면, --global을 빼고 실행
- Git은 이 정보를 사용하여 각 commit에 작성자를 기록



[실습] 저장소 만들기

- 저장소(repository)
  - 버전 관리를 할 파일들과 그 변경 이력들을 모두 저장해 두는 곳
  - 보통 작업하는 프로젝트의 가장 상위 디렉터리에 생성

- **git init** : 새로운 Git 저장소를 만든다.

- 적당한 디렉터리를 만들고 이동해서 아래 명령을 실행

(Shell)

```
git init
```

[실습] 저장소 만들기

- .git이라는 디렉터리가 생성되었는지 확인하자

(Shell)

```
ls -al  # 혹은 ll
```

- .git이라는 숨김 디렉터리가 바로 Git 저장소이다. 이 디렉터리에 현재 디렉터리와 그 하위 디렉터리의 파일들의 변경 이력 정보가 저장된다.

- 다시 말하면, 현재 디렉터리를 Git의 관리에서 벗어나게 하려면, 단순히 .git 디렉터리를 삭제하면 된다.

[실습] 새로운 파일을 추적 대상으로 등록

- **git status** : 현재 작업하고 있는 프로젝트 디렉터리의 상태를 확인.

(Shell)

```
git status
```

- 파일을 하나 만든 후 다시 git status로 상태 확인.

(Shell)

```
vi file1.txt  # 아무 내용이나 입력 후 저장 & 종료
git status   # file1.txt의 상태가 Untracked files로 나옴.
```

[실습] 새로운 파일을 추적 대상으로 등록

- Git은 새로 추가된 파일을 자동으로 관리하지 않음
- **git add** : 새로운 파일을 추적 대상으로 등록 (commit 대기 상태로 만든다)

(Shell)

```
git add file1.txt

git status    # file1.txt의 상태가 Changes to be committed로 변경된 것을 볼 수 있다.
# 즉, 아직 저장소에 반영(commit)되지는 않은 상태이다.
```

[실습] Commit 하기

- **git commit** : commit 대기 상태의 변경 사항 (Changes to be committed로 표시된 변경사항)을 저장소에 반영

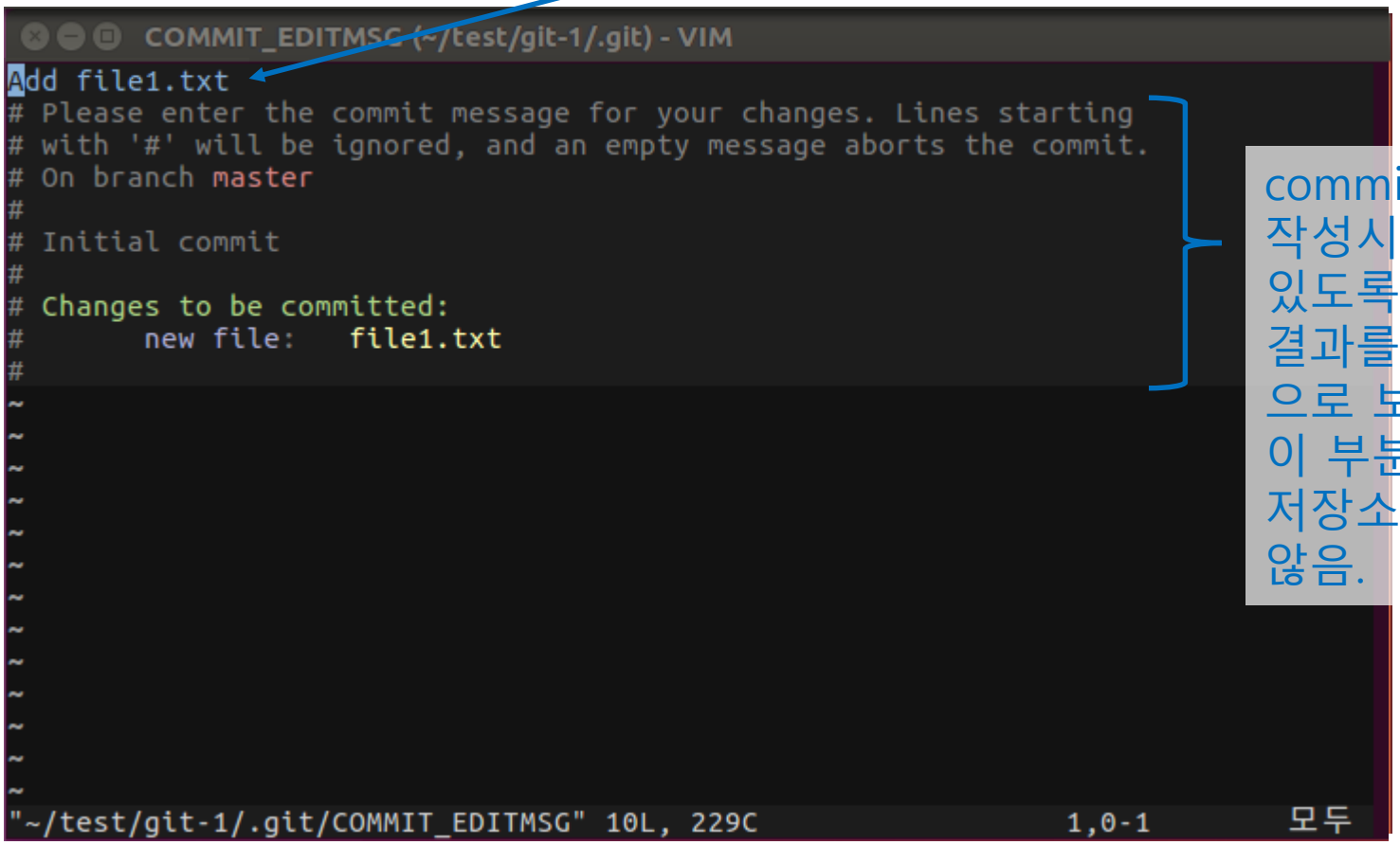
- 새로운 "버전"을 만드는 것이라고 볼 수 있음

(Shell)

```
git commit
```

- Vim이 실행되어 commit message를 입력하도록 요구한다.

이곳에 commit message를 입력하면 된다.



commit message 작성시 참고할 수 있도록 git status의 결과를 아래에 주석으로 보여준다.
이 부분은 실제로 저장소에 저장되진 않음.

- Commit message 작성 후 저장하고 Vim을 종료하면 commit이 완료된다.

- git status로 상태 확인
  - "nothing to commit, working directory clean"

[실습] 파일 수정 후 Commit 하기

(Shell)

```
vi file1.txt     # 먼저 파일을 수정해보자
git status     # file1이 Changes not staged for commit로 분류되고 modified로 표시됨
```

- **git add** : 이미 추적이 되고 있는 파일의 수정 사항을 commit 대기 상태(Changes to be committed)로 변경

(Shell)

```
git add file1.txt
git status     # file1이 Changes to be committed로 분류됨

git commit
git status     # 상태 확인
```

[실습] 파일 여러 번 수정 후 Commit 하기

- 이번에는 file1.txt에 약간의 수정(수정1) 후 git add만 한 상태에서 또 다른 수정(수정2)을 하고 git status를 해보자.

- file1.txt가 Changes to be committed와 Changes not staged for commit 양쪽 모두에 표시된다.

- Changes to be committed – 수정1에 해당하는 수정사항

- Changes not staged for commit – 수정2에 해당하는 수정사항

[실습] 여러 파일 수정 후 Commit 하기

- 이번에는 file2.txt를 새로 추가하고 add, commit 해보자.
- file1.txt과 file2.txt의 내용을 모두 조금씩 수정해보자.

- file1의 변경사항만 add후 commit해보자.
- git status를 보면, file2의 변경 사항은 unstaged 상태(Changes not staged for commit)로 남아있다.

- git add 명령이 따로 있는 이유는?
  - (모든 파일의 모든 작업 내역 중에) 원하는 파일, 원하는 변경만 선택하여 **(stage에 올린 후)** commit을 하기 위해

- stage : commit 대기 상태의 변경 사항들이 있는 곳.
- 바로 이 stage와 git add의 개념이 git의 차별점 중 하나.

(참고) Commit Message는 생각보다 중요하다

- **변경사항에 대한 맥락을 공유할 수 있는 최고의 수단**은 잘 다듬어진 커밋 메시지

- "코드 조각의 앞뒤 맥락을 다시 살펴야 하는 것은 가치 없는 일이다. 이를 완벽하게 피할 수는 없기에, 코드 맥락을 다시 살피는 일을 줄이기 위해서 최대한 노력해야 한다. 커밋 메시지는 정확히 그런 일을 할 수 있고, 이로 인해 커밋 메시지 하나로 어떤 개발자가 좋은 협력자인지 아닌지 알 수 있다"

https://item4.github.io/2016-11-01/How-to-Write-a-Git-Commit-Message/

(참고) 바람직한 Commit Message 작성법

- **1. 제목과 본문을 빈 행으로 분리한다**
- 2. 제목 행을 50자로 제한한다
- **3. 제목 행 첫 글자는 대문자로 쓴다**
- 4. 제목 행 끝에 마침표를 넣지 않는다
- **5. 제목 행에 명령문을 사용한다**
- 6. 본문을 72자 단위로 개행한다
- 7. 어떻게 보다는 무엇과 왜를 설명한다

https://item4.github.io/2016-11-01/How-to-Write-a-Git-Commit-Message/#seven-rules

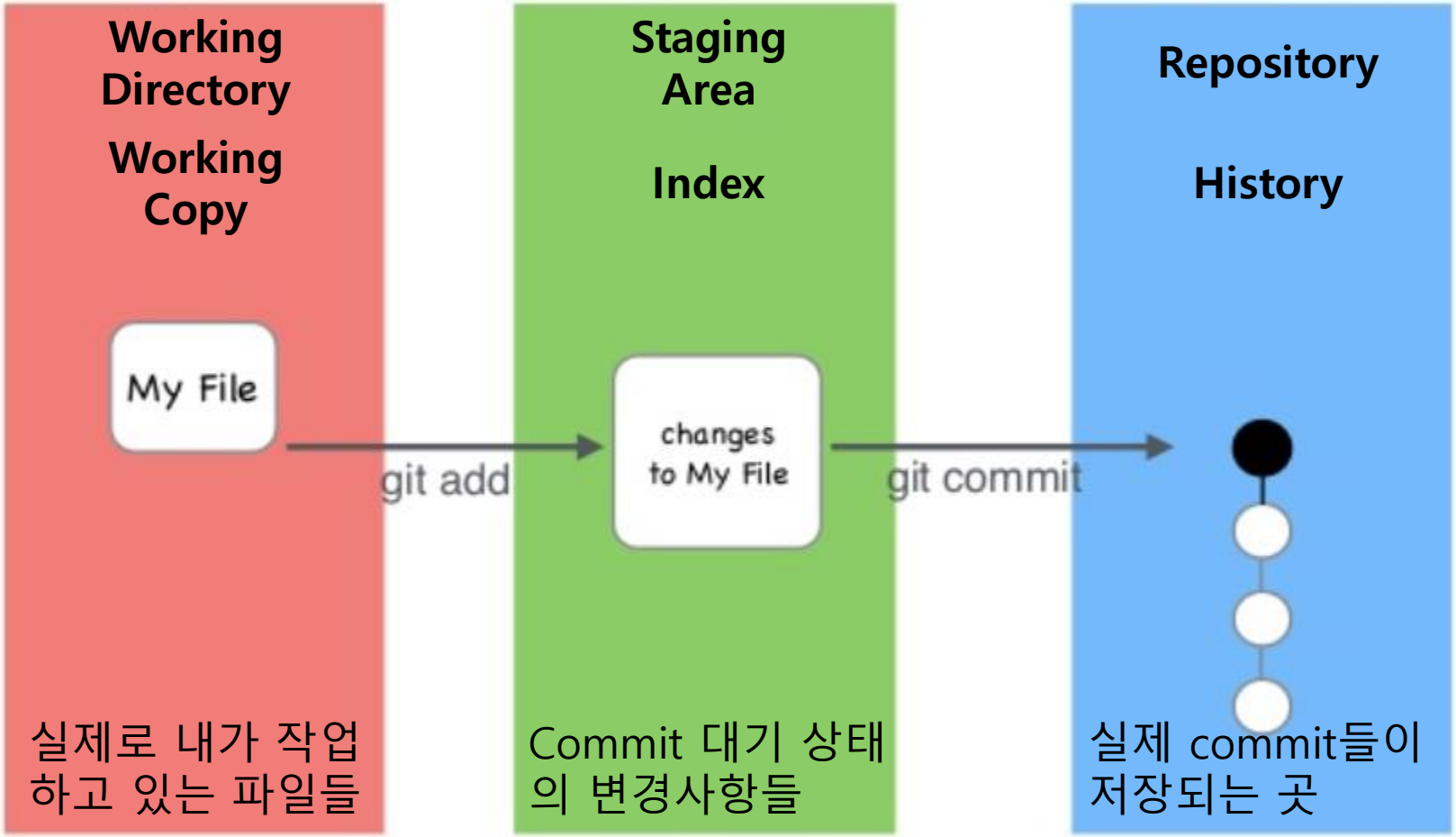
- 본 강의의 실습에서는 편의상 commit message의 제목 한 줄만을 작성할 것임.

파일 추가, 수정, 커밋 연습

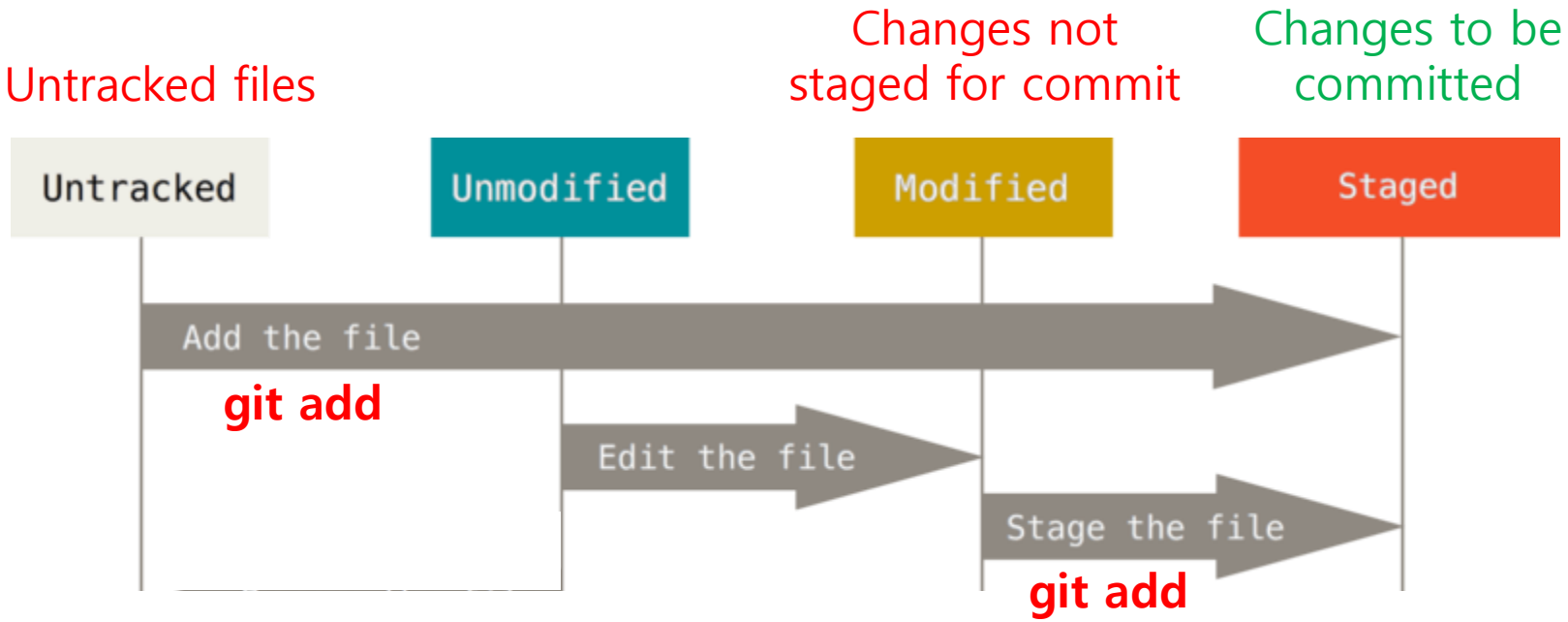
- file1.txt, file2.txt의 내용을 자유롭게 수정하고 add와 commit 해보고, 새로운 파일도 추가하는 것을 여러 번 연습해보자.

- git 명령어를 한 번 실행할 때마다 git status로 파일들의 상태가 어떻게 변했는지 꼭 확인하면서 진행할 것.

Git 작업 공간의 분류


Working Directory
Working Copy
My File
Staging Area
Index
changes to My File
Repository
History
git add
git commit
실제로 내가 작업하고 있는 파일들
Commit 대기 상태의 변경사항들
실제 commit들이 저장되는 곳

파일의 상태 변화


Untracked files
Changes not staged for commit
Changes to be committed
Untracked
Unmodified
Modified
Staged
Add the file
git add
Edit the file
Stage the file
git add