

Git

모두의연구소
박은수

Git 설치

- Git Download
 - <https://git-scm.com/downloads>



참고자료

- DataCamp : Introduction to Git for Data Science
 - <https://www.datacamp.com/courses/introduction-to-git-for-data-science>

- <https://www.datacamp.com/courses/introduction-to-git-for-data-science>

이거 취득이 목표입
니다

오늘 다는 못할것 같
고 나머지는 숙제



1-1) What is version control?

- Version Control System
 - 파일 및 디렉토리에 대한 변경사항을 관리하는 도구
 - 대표적인 Version Control System : Git
- Git의 장점
 - 언제든지 이전 버전으로 복구 가능
 - 파일이 충돌할 경우(즉, 누군가 내가 만든 파일을 덮어쓸 경우) 자동으로 알려주고 복구 가능
 - 팀 플레이 가능 - 팀과의 작업 동기화

1-1) What is version control?

- 문제) Git이 하는 역할은 ?

Possible Answers

- ☐ Keep track of changes to files.
- ☐ Notice conflicts between changes made by different people.
- ☒ Synchronize files between different computers.
- ☐ All of the above.

press

1

press

2

press

3

press

4

맞추세요

1-2) Where does Git store information?

- Git의 구성
 - 1) 파일과 디렉토리
 - 직접 편집하고 만듦
 - 2) 프로젝트 히스토리 (history)
 - 버전 관리를 위한 추가적인 정보들
 - 1+2 = repository라고 함
- 프로젝트 히스토리는 어디에?
 - .git 폴더 (‘.’ 은 숨김 파일이죠)

1-2) Where does Git store information?

- Git의 구성
 - 1) 파일과 디렉토리
 - 직접 편집하고 만듦
 - 2) 프로젝트 히스토리 (history)
 - 버전 관리를 위한 추가적인 정보들
 - 1+2 = repository라고 함
- 프로젝트 히스토리는 어디에?
 - Repository 폴더에 숨김 폴더로 존재
 - .git 폴더 ('.' 은 숨김 파일이죠)

dental 폴더 밑의 .git 폴더

```
→ dental git:(master) ls -al
total 0
drwxr-xr-x@ 3 eunsoo  staff   96  1  9 12:23 .
drwxr-xr-x@ 3 eunsoo  staff   96  1  9 12:23 ..
drwxr-xr-x@ 10 eunsoo  staff  320  1  9 12:23 .git
→ dental git:(master) cd .git
→ .git git:(master) ls -al
total 24
drwxr-xr-x@ 10 eunsoo  staff  320  1  9 12:23 .
drwxr-xr-x@ 3 eunsoo  staff   96  1  9 12:23 ..
-rw-r--r--@ 1 eunsoo  staff   23  1  9 12:23 HEAD
drwxr-xr-x@ 2 eunsoo  staff   64  1  9 12:23 branches
-rw-r--r--@ 1 eunsoo  staff  137  1  9 12:23 config
-rw-r--r--@ 1 eunsoo  staff   73  1  9 12:23 description
drwxr-xr-x@ 12 eunsoo  staff  384  1  9 12:23 hooks
drwxr-xr-x@ 3 eunsoo  staff   96  1  9 12:23 info
drwxr-xr-x@ 4 eunsoo  staff  128  1  9 12:23 objects
drwxr-xr-x@ 4 eunsoo  staff  128  1  9 12:23 refs
→ .git git:(master)
```


1-2) Where does Git store information?

- 문제) 당신의 home 디렉토리인 /home/repl 밑에 Git repository인 dental 폴더를 만들었고 이 폴더 밑에는 data란 폴더가 존재한다(즉, repository는 dental이고 그 밑에 하위 디렉토리로 data가 존재함).
- /home/repl/dental/data의 히스토리를 저장하고 있는폴더의 위치는 어디인가?

Possible Answers

☐ /home/repl/.git

press 1

☐ /home/repl/dental/.git

press 2

☐ /home/repl/dental/data/.git

press 3

☒ None of the above.

press 4

1-3) How can I check the state of a repository?

- **git status**
 - Repository의 상태 체크

```
1. eunsoo@bag-eunsuui-MacBook-Pro: ~/Dropbox/05.modu/01_DLC/  
→ dental git:(master) git status  
On branch master  
  
No commits yet  
  
nothing to commit (create/copy files and use "git add" to track)  
→ dental git:(master) █
```

1-3) How can I check the state of a repository?

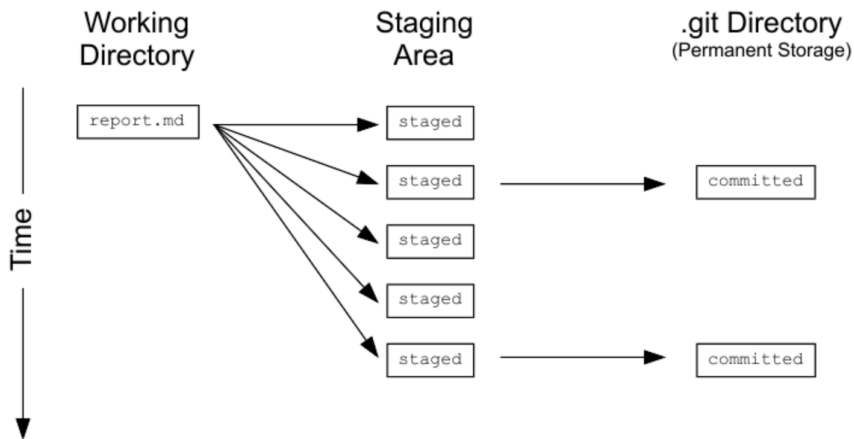
- 문제) **git status** 명령어로 어떤 파일이 변경됐는지 확인하시오

🔍 INSTRUCTIONS 50XP

Possible Answers

<input type="radio"/> data/summer.csv .	press 1
<input type="radio"/> report.txt .	press 2
<input type="radio"/> Neither of the above.	press 3
<input checked="" type="radio"/> Both of the above.	press 4

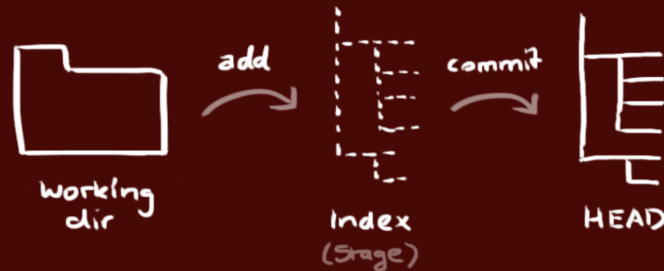
1-4) How can I tell what I have changed?



작업의 흐름

여러분의 로컬 저장소는 git이 관리하는 세 그루의 나무로 구성되어있어요.

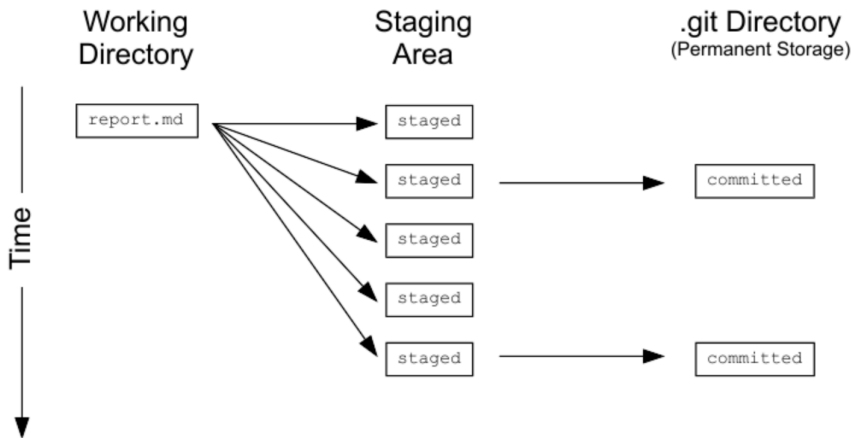
첫번째 나무인 **작업 디렉토리(Working directory)**는 실제 파일들로 이루어져있고, 두번째 나무인 **인덱스(Index)**는 준비 영역(staging area)의 역할을 하며, 마지막 나무인 **HEAD**는 최종 확정본(commit)을 나타내요.



1-4) How can I tell what I have changed?

- **git diff** fileName [directory]
 - 현재의 상태와 staging 된 상태를 비교해 줌

결과는 이런 식임



```
diff --git a/ReadMe.md b/ReadMe.md
index fe265ec..7a7146e 100644
--- a/ReadMe.md
+++ b/ReadMe.md
@@ -1,2 @@
-## Nice to meet you.
\ No newline at end of file
+## Nice to meet you.
+### Hihi
\ No newline at end of file
```

1-4) How can I tell what I have changed?

- **git diff**
 - Repository 의 변경 사항을 보여줌
- **git diff fileName [directory]**
 - File의 변경내용이나 directory의 변경 내용을 보여줌

1-4) How can I tell what I have changed?

- 문제) **git diff**로 repository의 변경 내용을 확인하세요

1-5) What is in a diff?

- **git diff** fileName [directory]

```
diff --git a/report.txt b/report.txt    a : 첫번째 버전, b : 두번째 버전
```

```
index e713b17..4c0742a 100644    두버전의 키 값
```

```
--- a/report.txt
```

```
+++ b/report.txt
```

```
@@ -1,4 +1,4 @@
```

```
-# Seasonal Dental Surgeries 2017-18    없어진 것
```

```
+# Seasonal Dental Surgeries (2017) 2017-18    생긴 것
```

4번째 라인이 없어지고,
4번째 라인에 추가 됐다

```
TODO: write executive summary.
```


1-5) What is in a diff?

- 문제) **git diff** data/northern.txt 명령어를 이용하여 northern.txt의 변경내용을 확인해보세요. 몇 줄이 바뀌어졌나요?

1-6) What's the first step in saving changes?

- Commit 의 대표적 2단계
 - 1) **add** 명령어로 staging 영역으로 파일들 보내기
 - 2) Staging area에 있는 것들을 한번에 **commit**하기
- Stage로 보내는 명령어는 **git add filename**
- 문제1) report.txt 파일을 stage 영역으로 보내세요.
- 문제 2) 그 후 잘 보내졌는데 git 명령어로 확인하세요.

1-7) How can I tell what's going to be committed?

- 파일의 현재상태와 stage 영역에 있는 것을 비교할때
 - `git diff -r HEAD path/to/file`
 - `-r` : 특정 인덱스(or hash)를 비교하겠다는 옵션
 - `HEAD` : 가장 최근 commit의 의미함
 - `path/to/file` : 보고싶은 파일
 - 즉, 보고싶은 파일의 변경 내용을 가장 최근 commit내용과 비교해봐라

1-7) How can I tell what's going to be committed?

- 문제 1) 파일들이 가장최근 commit과 어떻게 다른지 `git diff -r`과 배웠던 키워드를 이용해서 출력해봐라
- 문제 2) `data/northern.csv` 파일은 이미 `git add`로 stage되어 있다. 이 파일이 가장 최근 commit과 어떻게 다른지 출력해라
- 문제 3) 나머지 한 파일도 stage 영역으로 옮겨라

1-8) Interlude: how can I edit a file?

- Nano editor

- nano filename

- filename이 존재하면 파일을 편집
 - filename이 없으면 만들

Ctrl-K: delete a line.

Ctrl-U: un-delete a line.

Ctrl-O: save the file ('O' stands for 'output').

Ctrl-X: exit the editor.

1-8) Interlude: how can I edit a file?

- 문제)
 - nano names.txt 를 입력하여 names.txt를 만들고, names.txt에 아래와 같이 입력하시요

```
Lovelace  
Hopper  
Johnson  
Wilson
```

- 그리고 **Ctrl+o**로 파일저장 후, 파일이름 확인 시 **enter**를 클릭하고 **Ctrl+x**로 빠져나오시오

1-9) How do I commit changes?

- `git commit -m` “commit message”
 - `-m` : 메시지라는 옵션, `commit`에 대한 주석
 - Staging 되어있는 것들을 `commit`(최종본으로 저장) 한다
 - `Commit`되는 것들은 되돌릴 수 있다

1-9) How do I commit changes?

- 문제 1) git add로 이미 report.txt를 stage 영역으로 보냈다. 제대로 되었는지 git 명령어로 확인하시오.
- 문제 2) stage영역에 존재하는 파일들을 "Adding a reference." 메시지와 함께 commit 하세요

1-10) How can I view a repository's history?

- **git log**
 - 최근 commit된 내용부터 그 log를 보여줌

```
commit 0430705487381195993bac9c21512ccfb511056d
Author: Rep Loop <repl@datacamp.com>
Date:   Wed Sep 20 13:42:26 2017 +0000
```

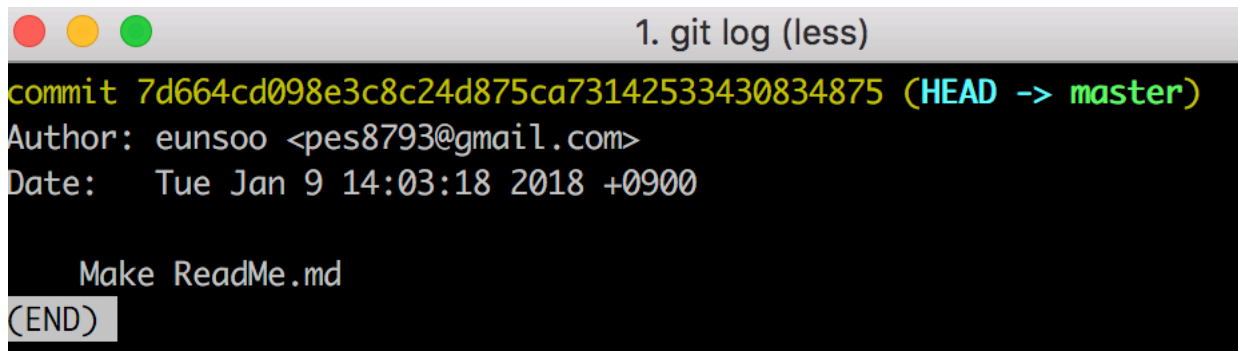
Commit ID

```
Added year to report title.
```

message

1-10) How can I view a repository's history?

- git log
 - Space : 다음 페이지로
 - q : 그만 보기



```
1. git log (less)
commit 7d664cd098e3c8c24d875ca73142533430834875 (HEAD -> master)
Author: eunsoo <pes8793@gmail.com>
Date: Tue Jan 9 14:03:18 2018 +0900

    Make ReadMe.md

(END)
```

1-10) How can I view a repository's history?

- 문제) Repository가 만들어 진 후 최초의 commit 메시지는 무엇인가?

☑ INSTRUCTIONS 50XP

Possible Answers

<input checked="" type="radio"/> "Added summary report file."	press 1
<input type="radio"/> "Adding seasonal CSV data files."	press 2
<input type="radio"/> "Fixed bug and generated results."	press 3
<input type="radio"/> "Adding reminder to cite funding sources."	press 4

1-11) How can I view a specific file's history?

- **git log path**
 - path는 특정 파일이나 폴더가 될 수 있다
 - 즉 특정 파일이나 폴더의 변경이력을 볼 수 있게 된다
- 문제) git log 명령어로 data/southern.csv 파일의 변경내용을 살펴보자. 몇번 변경됐는가?

1-12) How do I write a better log message?

- **git commit**

- 그냥 이렇게 하면 default로 되어있는 text editor가 나타남
- Editor에다가 메세지 남기고 저장하면 됨
 - Nano의 경우 ctrl+o로 저장, 이름 확인 시 enter, 나올 때는 ctrl+x

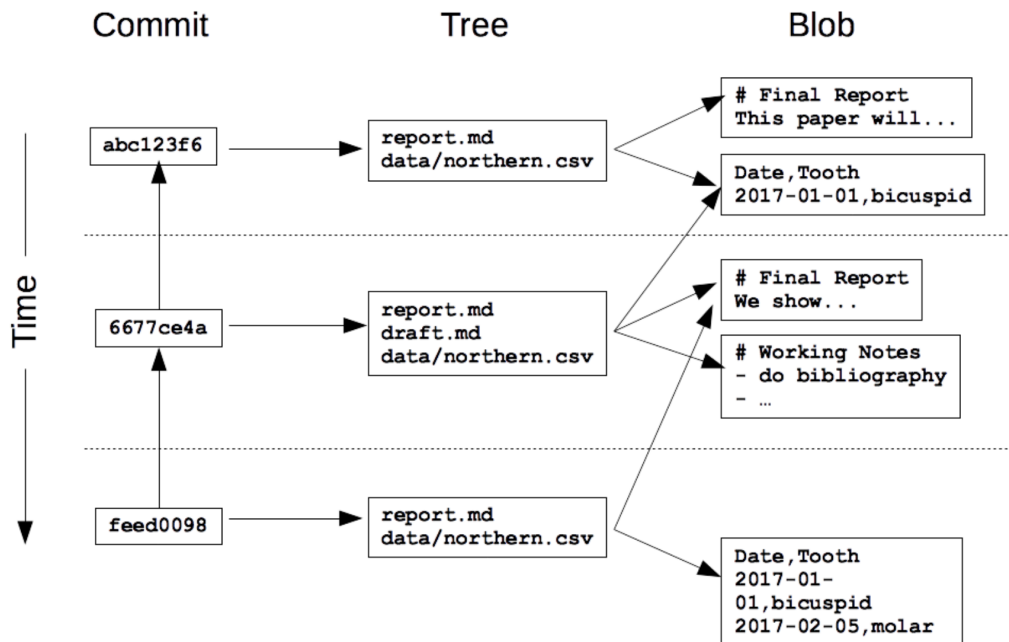
1-12) How do I write a better log message?

- 문제) report.txt 파일은 이미 stage되어있다. 메시지 옵션 -m 없이 **git commit** 명령어만으로 commit을 해보라.
 - Nano의 경우 ctrl+o로 저장, 이름 확인 시 enter, 나올 때는 ctrl+x

2-1) How do Git store information?

• 데이터 저장을 위한 multi-level 구조를 취함

- 1) Blob : 파일들의 고유한 버전들
- 2) Tree : blob에 해당하는 파일들의 이름과 위치를 기록
- 3) Commit : staging된 파일의 저장 기록들

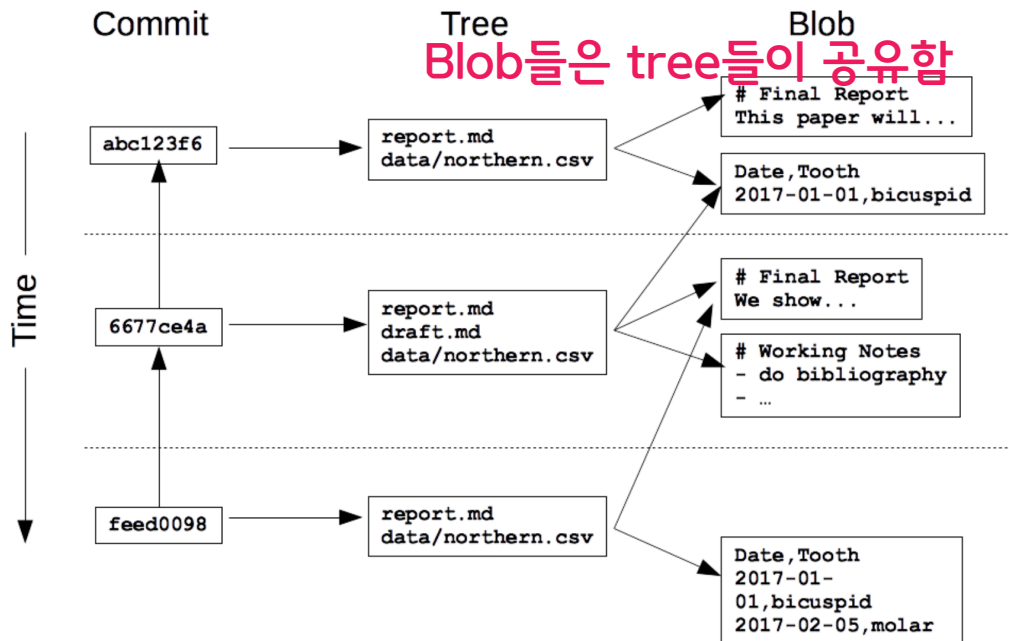


2-1) How do Git store information?

• 데이터 저장을 위한 multi-level 구조를 취함

• 문제)

- 어떤 파일이 마지막 commit에서 변경되었는가?
- 삭제가 아니라 변경



2-2) What is a hash?

- 모든 commit은 40글자의 고유한 id를 갖는데 이를 **hash**라고 함
 - Git은 이 40자중 6~8자 사이면 구분 가능하다고 봄
 - 파일별로 이 **hash**가 같다면 똑같은 파일로 생각함
 - Git은 전체 파일을 비교하지 않고 이 **hash**의 기록을 통해 효율적으로 파일의 변경 내용을 추적함

2-2) What is a hash?

- 문제) `cd` 명령어로 dental폴더에 들어간 후,
`git log`명령어로 log를 살펴 봅니다. 가장 최근
commit의 4자리 `hash`는 무엇인가요?
 - 가장 최근은 가장 위에 먼저 나와요

2-3) How can I view a specific commit?

- Hash 번호로 특정 commit 자세히 보는 법
 - **git show** [hash 앞자리 몇개(4자리 이상)]

```
commit 0430705487381195993bac9c21512ccfb511056d
Author: Rep Loop <repl@datacamp.com>
Date:   Wed Sep 20 13:42:26 2017 +0000
```

```
    Added year to report title.
```

```
diff --git a/report.txt b/report.txt
index e713b17..4c0742a 100644
--- a/report.txt
+++ b/report.txt
@@ -1,4 +1,4 @@
-# Seasonal Dental Surgeries 2017-18
+# Seasonal Dental Surgeries (2017) 2017-18
```

```
TODO: write executive summary.
```

git log의 결과와 같음

git diff의 결과와 같음

2-3) How can I view a specific commit?

- 문제) `git log`로 log를 살핀 후, 가장 최근 `commit`의 `hash번호`를 이용해서 `git show`로 자세히 살펴보세요. 몇개의 파일이 변경되었나요?

2-4) What is Git's equivalent of a relative path?

- Hash는 commit에 대한 절대경로로 볼 수 있다
- 가장 최근 commit에 해당하는 Hash는 HEAD로 쓸수 있고, 이며 이로부터의 뒤로 하나의 commit은 HEAD~1, 뒤로 두번째는 HEAD~2 처럼 사용 가능하다

2-4) What is Git's equivalent of a relative path?

- 문제) 가장 최근 바로 전 (HEAD~1)의 commit을 확인해 보자. 이 메시지를 봤을때 수정된 파일은 무엇인가?

2-5) How can I see who changed what in a file?

- 파일의 변경 사항을 자세히 체크해보기
 - `git annotate fileName`

```
1. git annotate ReadMe.md (less)
7d664cd0      (    eunsoo      2018-01-09 14:03:18 +0900      1)## Nice to meet you.
7d664cd0      (    eunsoo      2018-01-09 14:03:18 +0900      2)
00000000      (Not Committed Yet      2018-01-09 15:18:29 +0900      3)# Nono
00000000      (Not Committed Yet      2018-01-09 15:18:29 +0900      4)
00000000      (Not Committed Yet      2018-01-09 15:18:29 +0900      5)
00000000      (Not Committed Yet      2018-01-09 15:18:29 +0900      6)
00000000      (Not Committed Yet      2018-01-09 15:18:29 +0900      7)- This one
00000000      (Not Committed Yet      2018-01-09 15:18:29 +0900      8)- that one
00000000      (Not Committed Yet      2018-01-09 15:18:29 +0900      9)- I like you
```

2-5) How can I see who changed what in a file?

- 파일의 변경 사항을 자세히 체크해보기
 - `git annotate fileName`

1. git annotate ReadMe.md (less)			
7d664cd0	(eunsoo	2018-01-09 14:03:18 +0900	1)## Nice to meet you.
7d664cd0	(eunsoo	2018-01-09 14:03:18 +0900	2)
00000000	(Not Committed Yet	2018-01-09 15:18:29 +0900	3)# Nono
00000000	(Not Committed Yet	2018-01-09 15:18:29 +0900	4)
00000000	(Not Committed Yet	2018-01-09 15:18:29 +0900	5)
00000000	(Not Committed Yet	2018-01-09 15:18:29 +0900	6)
00000000	(Not Committed Yet	2018-01-09 15:18:29 +0900	7)- This one
00000000	(Not Committed Yet	2018-01-09 15:18:29 +0900	8)- that one
00000000	(Not Committed Yet	2018-01-09 15:18:29 +0900	9)- I like you
Hash (commit id)	누가 commit 했 는가?	만들어진 날짜	줄번호와 바뀐 글 들

2-5) How can I see who changed what in a file?

- 문제) `report.txt` 파일을 `git annotate` 함수로 살펴 보세요. 여기엔 몇번의 변경, 즉 몇개의 다른 종류의 `hash`가 존재하나요?

2-6) How can I see what changed between two commits?

- 두개의 commit간의 차이를 살펴보기
 - `git diff ID1(hash)..ID2(hash)`
- 예시)
 - `git diff abc123..def456`
 - `git diff HEAD~1..HEAD~3` (현재에서 1나 전과 3개 전을 비교)

2-7) How do I add new files?

• 파일이 추가 됐을 경우

```

→ dental git:(master) ✕ touch sencond.md
→ dental git:(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:   ReadMe.md

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

       sencond.md

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

```

Second.md 파일 추가
상태보기
Git 이 관리하고 있지 않다고 알려줌

따라서 새로 추가한 파일은 **git add** 명령어를 통해 Staging 해줘야 한다.

2-7) How do I add new files?

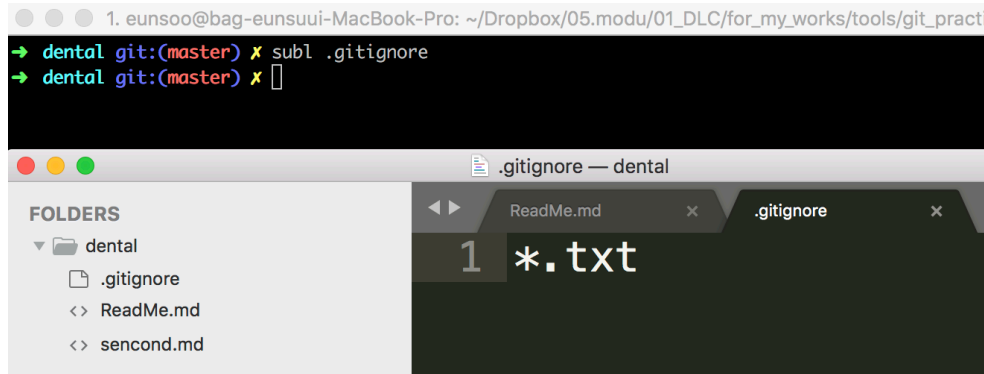
- 문제 1) **git status** 명령어로 git이 추적하지 않고 있는 파일을 확인 하시오.
- 문제 2) 이 파일을 **git add** 를 통해 stage 영역으로 보내시오.
- 문제 3) **git commit** 으로 stage에 있는 파일을 저장하고 commit 메시지(-m)는 “Starting to track data sources.” 로 하시오.

3-1) How do I tell Git to ignore certain files?

- 소스를 컴파일하거나 실행하면 버전관리에서 필요로 하지 않는 임시적인 데이터 혹은 실행시만 필요한 바이너리 데이터들이 생성될 경우가 있습니다.
- 예시)
 - build 폴더 : 빌드시 생성되는 중간파일들
 - *.pyc : 파이썬 실행 시 나타나는 파일

3-1) How do I tell Git to ignore certain files?

- **.gitignore**파일은 버전관리에 필요없는 **파일**들이나 **폴더**를 명시해서 이들에 대한 버전관리를 하지 않게 할 수 있습니다



The screenshot shows a terminal window at the top with the following commands and output:

```
1. eunsoo@bag-eunsui-MacBook-Pro: ~/Dropbox/05.modu/01_DLC/for_my_works/tools/git_practi
→ dental git:(master) ✕ subl .gitignore
→ dental git:(master) ✕ []
```

Below the terminal is a code editor window titled ".gitignore — dental". The left sidebar shows a file tree for the "dental" folder containing ".gitignore", "ReadMe.md", and "sencond.md". The main editor area shows the content of ".gitignore" with the text:

```
1 *.txt
```

예시) .gitignore를 만들고 확장자가 txt일 경우는 무시하게 함

3-1) How do I tell Git to ignore certain files?

```
→ dental git:(master) ✗ touch hi.py
→ dental git:(master) ✗ touch hi.txt
→ dental git:(master) ✗ git status
```

hi.py와 hi.txt의 빈 파일 생성
상태확인

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: ReadMe.md

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

.gitignore
hi.py
seconcd.md

hi.py만 Untracked files 에 나타남

no changes added to commit (use "git add" and/or "git commit -a")
→ dental git:(master) ✗ ls

ReadMe.md hi.py hi.txt seconcd.md

hi.txt는 존재함

3-1) How do I tell Git to ignore certain files?

문제) .gitignore에 다음과 같이 기록하였다.
(폴더명과 파일명 혹은 *과 같이 기호도 적용된다.)

.gitignore

```
pdf  
*.pyc  
backup
```

보기 중 ignore가 적용되지 않은 파일은 무엇일까?

Possible Answers

- ☐ report.pdf
- ☐ bin/analyze.pyc
- ☐ backup/northern.csv
- ☒ None of the above.

press 1

press 2

press 3

press 4

3-2) How can I remove unwanted files?

- **git clean -n** 은 repository 내에서 tracking 되지 않은 파일들을 보여준다.

```
→ dental git:(master) ✕ git clean -n
Would remove .gitignore
Would remove hi.py
Would remove sencond.md
→ dental git:(master) ✕
```

```
→ dental git:(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:   ReadMe.md

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

    .gitignore
    hi.py
    sencond.md
```

3-2) How can I remove unwanted files?

- 전체를 stage로 보냈더니 git clean -n 해도 나타나지 않음

```
→ dental git:(master) ✕ git add .  
→ dental git:(master) ✕ git clean -n  
→ dental git:(master) ✕
```

- 다시 untrack 파일들을 만들고 해보면 나타남

```
→ dental git:(master) ✕ touch hello.py  
→ dental git:(master) ✕ git clean -n  
Would remove hello.py
```

3-2) How can I remove unwanted files?

- Untrack 파일들의 삭제는 **git clean -f**

```
→ dental git:(master) ✕ git clean -n
Would remove hello.py
→ dental git:(master) ✕ ls
ReadMe.md  hello.py  hi.py      hi.txt      sencond.md
→ dental git:(master) ✕ git clean -f
Removing hello.py
→ dental git:(master) ✕ ls
ReadMe.md  hi.py      hi.txt      sencond.md
→ dental git:(master) ✕
```

3-2) How can I remove unwanted files?

- 문제 1) **ls 명령어**로 폴더에 존재하는 파일 및 폴더 확인해 보세요
- 문제 2) **untrack 파일을 삭제하는 명령어**로 삭제하세요
- 문제 3) 다시 **ls 명령어**로 삭제 결과를 확인하세요

3-2) How can I remove unwanted files?

- Untrack 파일들의 삭제는 **git clean -f**

```
→ dental git:(master) ✕ git clean -n
Would remove hello.py
→ dental git:(master) ✕ ls
ReadMe.md  hello.py  hi.py      hi.txt      sencond.md
→ dental git:(master) ✕ git clean -f
Removing hello.py
→ dental git:(master) ✕ ls
ReadMe.md  hi.py      hi.txt      sencond.md
→ dental git:(master) ✕
```

3-3) How can I see how Git is configured?

- Git의 환경설정 확인하는 명령어
 - **git config --list**

```
→ dental git:(master) ✕ git config --list
credential.helper=osxkeychain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
user.name=eunsoo
user.email=pes8793@gmail.com
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
core.ignorecase=true
core.precomposeunicode=true
```

추가할 수 있는 옵션들

- **--system** : settings for every user on this computer.
- **--global** : settings for every one of your projects.
- **--local** : settings for one specific project.

3-3) How can I see how Git is configured?

- Git의 환경설정 확인하는 명령어
 - git config --list --system
 - git config --list --global
 - git config --list --local
- 우선순위

```
→ dental git:(master) * git config --list --system
fatal: unable to read config file '/etc/gitconfig': No such file or directory
→ dental git:(master) * git config --list --global
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
user.name=eunsoo
user.email=pes8793@gmail.com
→ dental git:(master) * git config --list --local
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
core.ignorecase=true
core.precomposeunicode=true
```

3-3) How can I see how Git is configured?

- 문제) 얼마나 많은 수의 **local** configuration 변수가 존재하나요?

3-4) How can I change my Git configuration?

- Git configuration은 대부분 그냥 default로 사용하지만 무조건 바꿔야 할 2개는 user.name과 user.email 이다
- 바꾸는 방법의 예시
 - `git config --global user.name eunsoo`
`user.email "es.park@modulabs.co.kr"`

3-4) How can I change my Git configuration?

- 예시) user.name을 hihi로 바꾸기

```
→ dental git:(master) x git config --global user.name hihi
→ dental git:(master) x git config --list --global
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
user.name=hihi
user.email=pes8793@gmail.com
```

3-4) How can I change my Git configuration?

- 문제) 모든 프로젝트에 대한 Git의 user의 이메일 주소를 rep.loop@datacamp.com 으로 변경하세요

3-5) How can I commit changes selectively?

- 변경한 파일을 한번에 모두 stage 영역으로 보낼 필요는 없습니다
- 예를 들면 hi.py 를 만든 후, bye.py 파일을 만들었는데 이를 모두 stage에 보낸 후 한번에 commit하면 메시지를 넣기 애매해 집니다

3-5) How can I commit changes selectively?

1개의 not staged 파일과 3개의 untrack 파일

```
→ dental git:(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:   ReadMe.md

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

       .gitignore
       hi.py
       sencond.md
```

3-5) How can I commit changes selectively?

선택적 staging

```
→ dental git:(master) ✗ git add hi.py
→ dental git:(master) ✗ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   hi.py

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:   ReadMe.md

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

    .gitignore
    sencond.md
```

3-5) How can I commit changes selectively?

문제 1) data/northern.csv 파일을
stage영역으로 옮기고

문제 2) “Adding data from northern
region” 메시지로 commit 하세요

나머지는 숙제

- Datacamp를 완료하시오 Slack #lec_tools 채널에 certification 주소 혹은 다운 받은 것을 올려주세요
– 완료한 코스 보기

COURSES IN PROGRESS (2)

COMPLETED (7)

Explore Course Library

Intro to Python for Data Science (edX 4.2)

Lab exercises on 2D Numpy Arrays.

Share

Replay Course

Intro to Python for Data Science (edX 4.3)

Lab exercises on Numpy Basic Statistics.

Share

Replay Course

Intro to Python for Data Science (edX 5.1)

Lab exercises on Basic Plots with matplotlib.

Share

Replay Course

감사 합니다



박 은 수 Research Director

E-mail : es.park@modulabs.co.kr