vim f1.txt

I Insert

esc + : w q 저장하고 종료

```
git config --global user.name "자신의 닉네임"
git config --global user.email "자신의 이메일"

git status

git commit

git log
```

git add f1.txt commit 대기상태
git commit commit 대기상태의 (stage area)파일들을 commit 한다.

git log -p 각각의 소스상에 변경사항 확인

```
commit af72fad995c54e977afb8b2e9fa72ba59cd288ba
Author: large <kdw4458@naver.com>
Date:   Mon Apr 10 17:49:06 2017 +0900

    4

diff --git a/f1.txt b/f1.txt
index 598f999..9462317 100644
--- a/f1.txt            //3버젼의 파일 내용
+++ b/f1.txt           //현재버젼(4)의 파일 내용
@@ -1 +1 @@
-soucrce : 2
+f1.txt : 2
```

```
commit c50cf47bd40905ace75dd6025aa925b1e9b0b6d7
Author: large <kdw4458@naver.com>
Date:   Mon Apr 10 17:45:45 2017 +0900

    3

diff --git a/f2.txt b/f2.txt
new file mode 100644
```

```
index 0000000..598f999
--- /dev/null       //버젼2에서는  없었다
+++ b/f2.txt
@@ -0,0 +1 @@
+soucrce : 2        //f2.txt 내용의 초기내용
```

git log c50cf47bd40905ace75dd6025aa925b1e9b0b6d7

log 뒤의 commit 코드 이전버젼을 나열한다.

```
commit c50cf47bd40905ace75dd6025aa925b1e9b0b6d7
Author: large <kdw4458@naver.com>
Date:   Mon Apr 10 17:45:45 2017 +0900

    3

commit 39bbb461876b0499642cfd2884631c6d58d6a23f
Author: large <kdw4458@naver.com>
Date:   Mon Apr 10 17:44:14 2017 +0900

    2

commit 12610ed957cf0b1edf73aea94d582b8851654816
Author: large <kdw4458@naver.com>
Date:   Mon Apr 10 17:40:24 2017 +0900

    1
```

git diff
c50cf47bd40905ace75dd6025aa925b1e9b0b6d7..12610ed957cf0b1edf73aea94d582b8851654816

각각의 버젼간의 소스코드의 차이점 (여기서는 3버젼과 1버젼간의 차이점)

```
diff --git a/f1.txt b/f1.txt
index 598f999..cf5e520 100644
--- a/f1.txt
+++ b/f1.txt
@@ -1 +1 @@
-soucrce : 2
+soucrce : 1
diff --git a/f2.txt b/f2.txt
deleted file mode 100644
index 598f999..0000000
--- a/f2.txt
+++ /dev/null
```

```
@@ -1 +0,0 @@
-soucrce : 2
```

로그에서 출력되는 버전 간의 차이점을 출력하고 싶을 때

git log -p

버전 간의 차이점을 비교할 때

git diff '버전 id'..'버전 id2'

git add하기 전과 add한 후의 파일 내용을 비교할 때

git diff

아래 명령은 버전 id로 돌아가는 명령입니다.

git reset --hard "버전 id" hard말고 soft등도 있다.

버전 id의 커밋을 취소한 내용을 새로운 버전으로 만드는 명령

git revert "버전 id"

reset 시계를 과거로 돌리는 것

revert 특정사건을 없었던 일로 만드는 것

git commit —help

git commit -a

git commit -am “11”

물론 아직 버전관리를 배우는 중요한 이유인 백업(backup)을 설명드리지 못했습니다. 즉 어떤 사고에 의해서 소스 코드를 잃어버리는 상황에 대한 대비책을 아직 세우지 못했습니다. git은 백업을 위한 방법을 자체적으로 가지고 있습니다. 하지만 이 방법은 다소 어렵기 때문에 천천히 배우면 됩니다. 추천 드리는 방법은 프로젝트 폴더 전체를 dropbox나 google drive에 보관하는 것입니다.

그리고 협업이 필요해질 때 원격저장소라는 개념을 익히고 사용하시면 됩니다.

git의 원리

gistory

도메인 복사해서 브라우저에서 OPEN

git파일명이 달라도 내용이 같으면 같은 object로 인식한다.

commit 자체도 객체가 될 수 있다.

commit의 원리

커밋에는

부모를 나타내는 parent

그 커밋이 일어난 시점에 우리의 작업디렉토리에 있는 파일의 이름과 파일의 이름이 담고있는 내용 사이에 정보가 tree라는 곳에 담겨있다.

프로젝트 폴더에대한 상태를 얻어 낼 수 있다.

각각의 버전은 그 버전이 만들어진 시점의 스냅샷을 tree라고하는 정보구조를 통해서 갖고있다.

d1/ 이라는 디렉토리를 만든후 f1.txt 파일을 생성하면 d1이라는 tree아래에 f1.txt(blob)이 생성

object는 commit / tree / blob 이렇게 3가지로 구성되어있다.

status의 원리

자료를 변경후

git status 를 쳐보면

```
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:   f2.txt

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

modified 즉 수정되었다고 알려주는데 이는

index라는 파일의 f2.txt 라는 파일의 내용과 현재의 내용이 다르다는것을 인지

git add f2.txt 를 입력하면

```
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

	modified:   f2.txt
```

라고 뜨게 된다. 이는 f2.txt 파일이 commit 대기상태임을 알려준다.

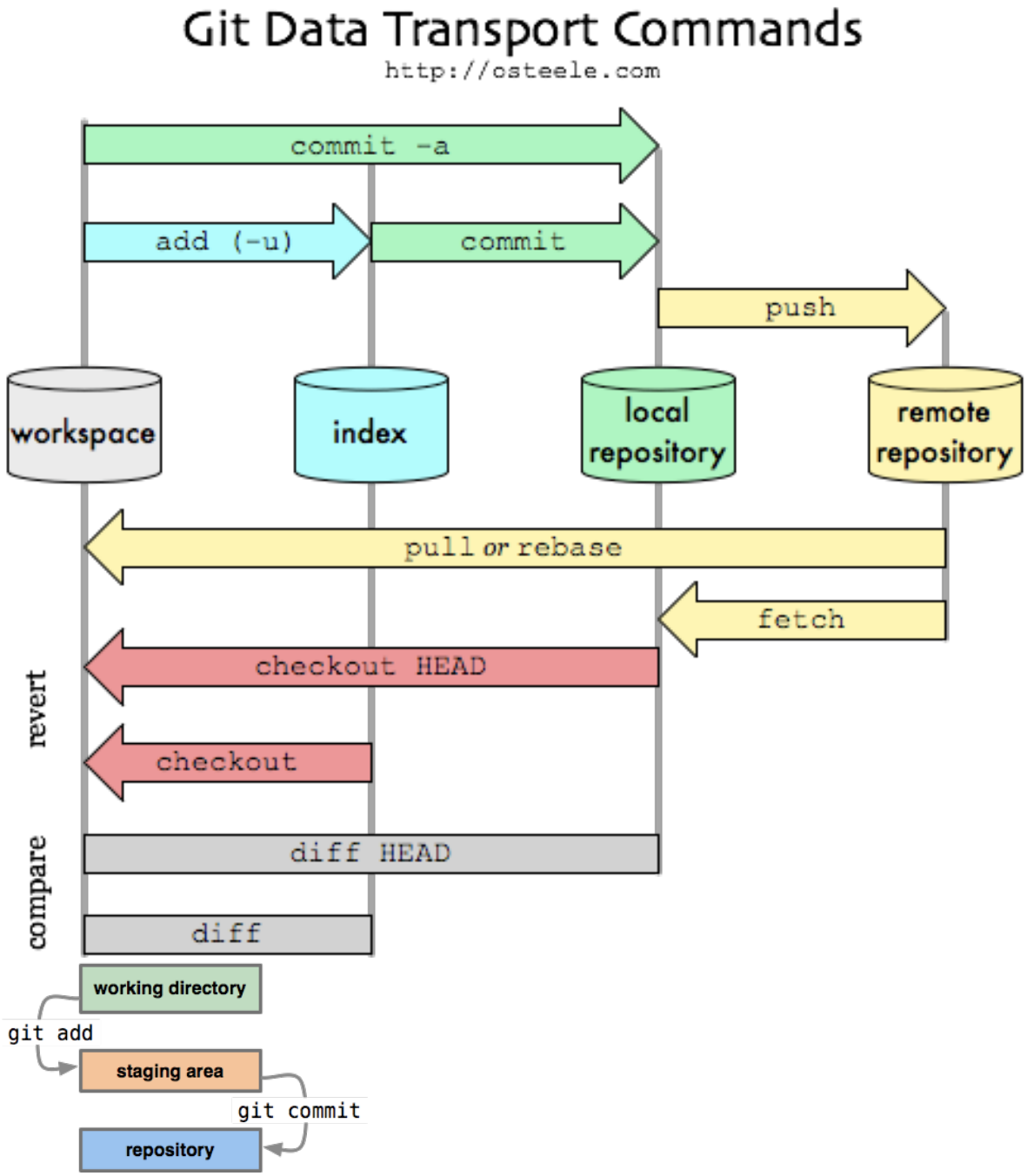
다시말해 git은 최신 index의 f2.txt와

마지막(최신)commit의 f2.txt의 내용이 다르다면

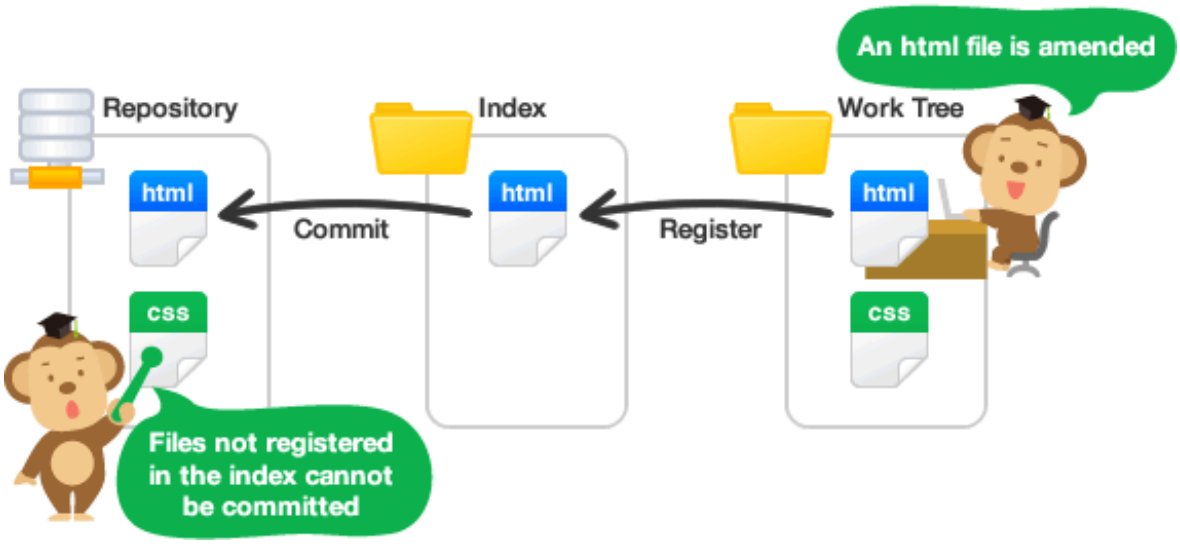
index에 add 되어서 현재 f2.txt는 commit대기상태라는것을 알려준다.

이상태에서 commit을 해주면

저장소와 index와 워킹카피(프로젝트폴더) 이 3가지가 정확하게 일치하기 때문에 git status를 입력하면 더이상 commit할것이 없다고 알려주는 것이다.



index 는 사용자 입장에서 staging area(commit 대기상태)라고도 부른다.



working directory - index, staging area, cache - repository