Git 명령어 사용법(II)

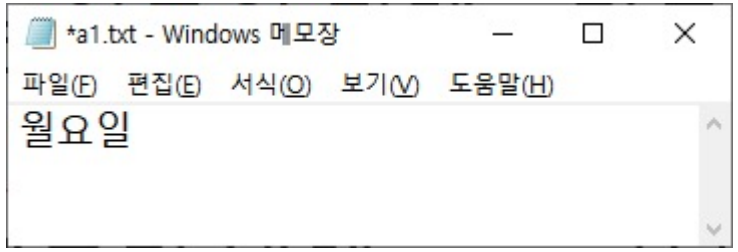
동의과학대학교 컴퓨터정보과

Git 기본 명령어 - 복습

명령어	설명
git init	저장소의 생성(.git 폴더가 만들어짐)
git add <filename> **git add --all**	저장소에 파일 추가(staging area로 올라감)
git commit –m “message”	저장소에 변경 내용 반영
git config --global user.name “gildong"	git 설정(사용자 정보 설정, 설정 확인 등)
git status	저장소 상태 확인
git diff	git add하기 전과 add한 후의 파일 내용을 비교할 때 버전(checksum)들 간의 소스 코드 내용 비교할 때
git log **git log -p** **git log --stat**	로그 상태 보기 각 파일 별 로그 상태 보기

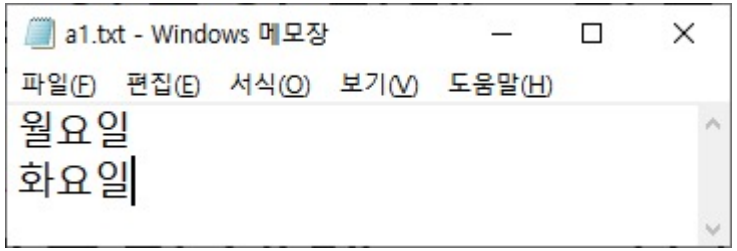
실습 - 복습

1. myfolder 라는 이름의 디렉토리를 바탕화면에 만들기
2. myfolder를 git repository 로 만들기
3. myfolder 디렉토리 내에 a1.txt 작성하기



우분트를 사용하면 vi를 사용해야하나 편리하게 사용하기 위해 메모장을 사용한다.

4. staging area에 a1.txt 올리기
5. repository에 commit 하기
6. 다음과 같이 파일을 변경하고 4, 5 반복하기
   - '토요일' 까지 입력하여 반복할 것



실습 진행 중 git log, git status를 실행하여 상황 파악하기

커밋 로그 확인 : git log -p

- 여러 파일로 이루어진 버전의 경우 –p 옵션 사용
  - 파일 별로 커밋 상태와 소스 내용 확인 가능
- 종료 방법
  - :q
  - :z
  - ctrl + z
- 실습
  - 앞의 실습에서 a2.txt를 작성하여 확인할 것
  - git log <checksum> 확인할 것

```
$ git log -p
commit 807d9421b63bd6b915e176d89c753b995b88dca3 (HEAD -> master)
Author: jinsook <putzmunter7@gmail.com>
Date:   Mon Nov 2 14:08:46 2020 +0900

    third commit - wednesday

diff --git a/a1.txt b/a1.txt
index b46dd8a..ba7ea4b 100644
--- a/a1.txt
+++ b/a1.txt
@@ -1,2 +1,3 @@
 월 요 일
-화 요 일
\ No newline at end of file
+화 요 일
+수 요 일
\ No newline at end of file

commit d0ff6e875c92c426130dedccb4389ef9afc18e4b
Author: jinsook <putzmunter7@gmail.com>
Date:   Mon Nov 2 13:59:51 2020 +0900

    second commit - tuesday

diff --git a/a1.txt b/a1.txt
index e265b5a..b46dd8a 100644
--- a/a1.txt
+++ b/a1.txt
@@ -1 +1,2 @@
-월 요 일
\ No newline at end of file
+월 요 일
+화 요 일
\ No newline at end of file

commit 8f46f84d2709c5cdb6ef782c793150c36b75c973
Author: jinsook <putzmunter7@gmail.com>
```

이전 버전의 파일 내용

현재 버전에서 추가된 내용

파일 내용 비교 : git diff

- git add하기 전과 add한 후의 파일 내용을 비교할 때 사용하는 명령어로 유용하게 사용됨
  - 커밋 전에 자신이 작성한 파일 검토가 가능

```
$ git diff
diff --git a/a1.txt b/a1.txt
index ba7ea4b..b46dd8a 100644
--- a/a1.txt
+++ b/a1.txt
@@ -1,3 +1,2 @@
 월 요 일
-화 요 일
-수 요 일
\ No newline at end of file
+화 요 일
\ No newline at end of file

jinsook@klein MINGW64 /d/1-4. 오 픈 소 스 /1.강 의 노 트 /2020/gitTest (master)
```

- 실습
  - 앞의 실습 문제에서 a1.txt 에 “일요일” 을 추가하여 파일을 변경한 후 git diff 명령 실행하기

파일 내용 비교 : git diff

- git diff <checksum1>..<checksum2>
  - 두 커밋 간의 소스 코드상의 차이점을 출력

```
jinsook@klein MINGW64 /d/1-4. 오픈소스/1.강의노트/2020/gitTest (master)
$ git diff 56cc486520aadac877f357c8b9d81fd632cf4816..3dad887208a168403e710938e5a09f7b77aa2c94
diff --git a/a1.txt b/a1.txt
index e265b5a..a132ff8 100644
--- a/a1.txt
+++ b/a1.txt
@@ -1 +1,4 @@
-월 요 일
\ No newline at end of file
+월 요 일
+화 요 일
+수 요 일
+
```

- 실습
  - git log로 커밋id를 알아내어 두 커밋 간의 소스 내용 차이를 확인하시오.

이전 버전으로 되돌리기

명령어	설명
git reset --<모드> <checksum> **git reset --hard <checksum>**	• 버전 삭제 • local repository에서만 사용할 것(협업 시 사용 불가능) • 없어진 commit 들은 삭제된 것이 아니고 남아 있으면 복구도 가능 • 옵션에 따라 HEAD, 스테이지작업, 작업디렉토리 내용이 달라짐
git revert	• 버전을 되돌리기(실제 되돌리는 것이 아니라 새로운 커밋을 하는 것)

이전 버전으로 되돌리기는 협업시에 문제의 소지가 많기 때문에
조심하여 다루어야 한다!!!

버전 삭제 : git reset --mode <checksum>

- 이전 작업 결과를 저장한 상태로 되돌리기(공유되기 전까지만 사용)

mode	의미
--hard	복구된 이력 이후의 내용을 모두 삭제 후 초기화(주의)
--soft	복구된 이력 이후 내용 모두 유지
--mixed	복구된 이력 이후 내용 모두 유지, 인덱스 초기화로 변경 내용 다시 추가 해야 함

```
jinsook@klein MINGW64 /d/1-4. 오픈소스/1.강의노트/2020/gitTest (master)
$ git reset d0ff6e875c92c426130dedccb4389ef9afc18e4b --hard
HEAD is now at d0ff6e8 second commit - tuesday

jinsook@klein MINGW64 /d/1-4. 오픈소스/1.강의노트/2020/gitTest (master)
$ git log
commit d0ff6e875c92c426130dedccb4389ef9afc18e4b (HEAD -> master)
Author: jinsook <putzmunter7@gmail.com>
Date:   Mon Nov 2 13:59:51 2020 +0900

    second commit - tuesday

commit 8f46f84d2709c5cdb6ef782c793150c36b75c973
Author: jinsook <putzmunter7@gmail.com>
Date:   Mon Nov 2 13:58:41 2020 +0900

    first commit - monday

jinsook@klein MINGW64 /d/1-4. 오픈소스/1.강의노트/2020/gitTest (master)
$ cat a1.txt
월요일
```

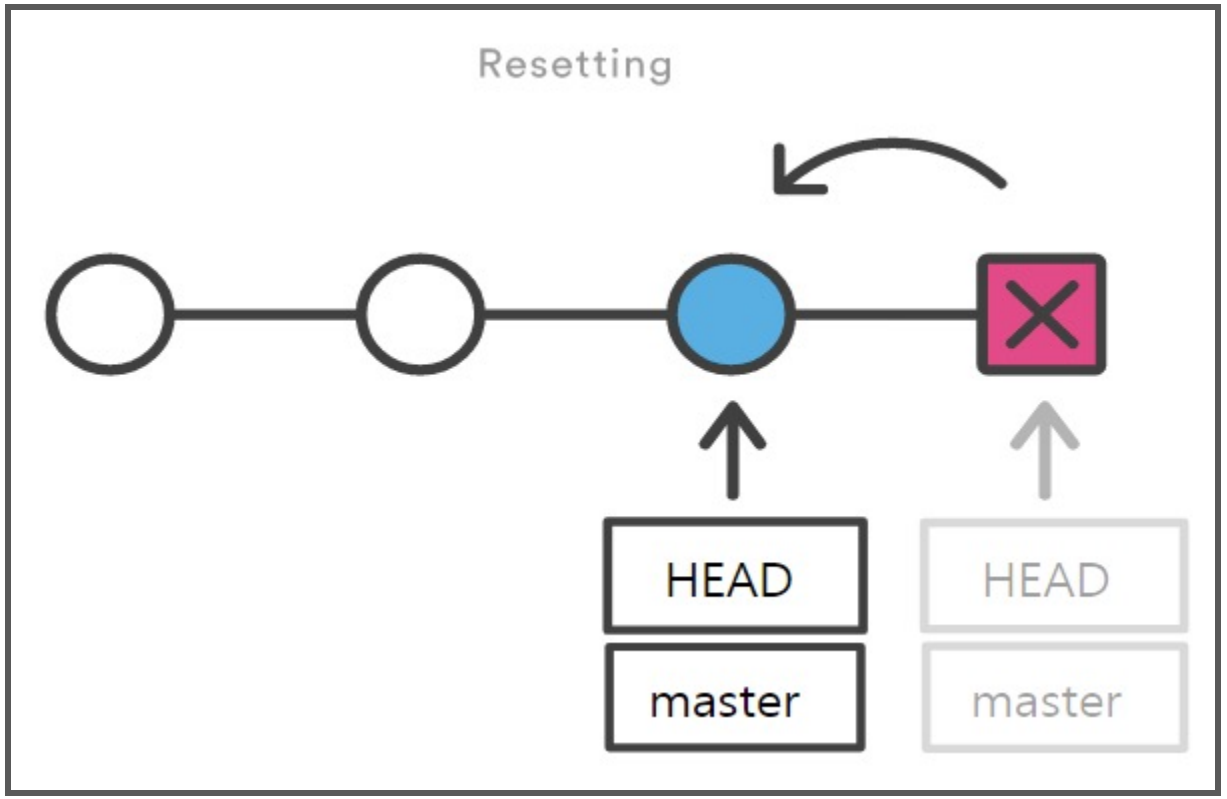
- ORIG_HEAD : git reset 실행 시 삭제된 커밋 내역 보관 파일
- git reset으로 삭제한 커밋 복구 방법
  - git reset --hard ORIG_HEAD

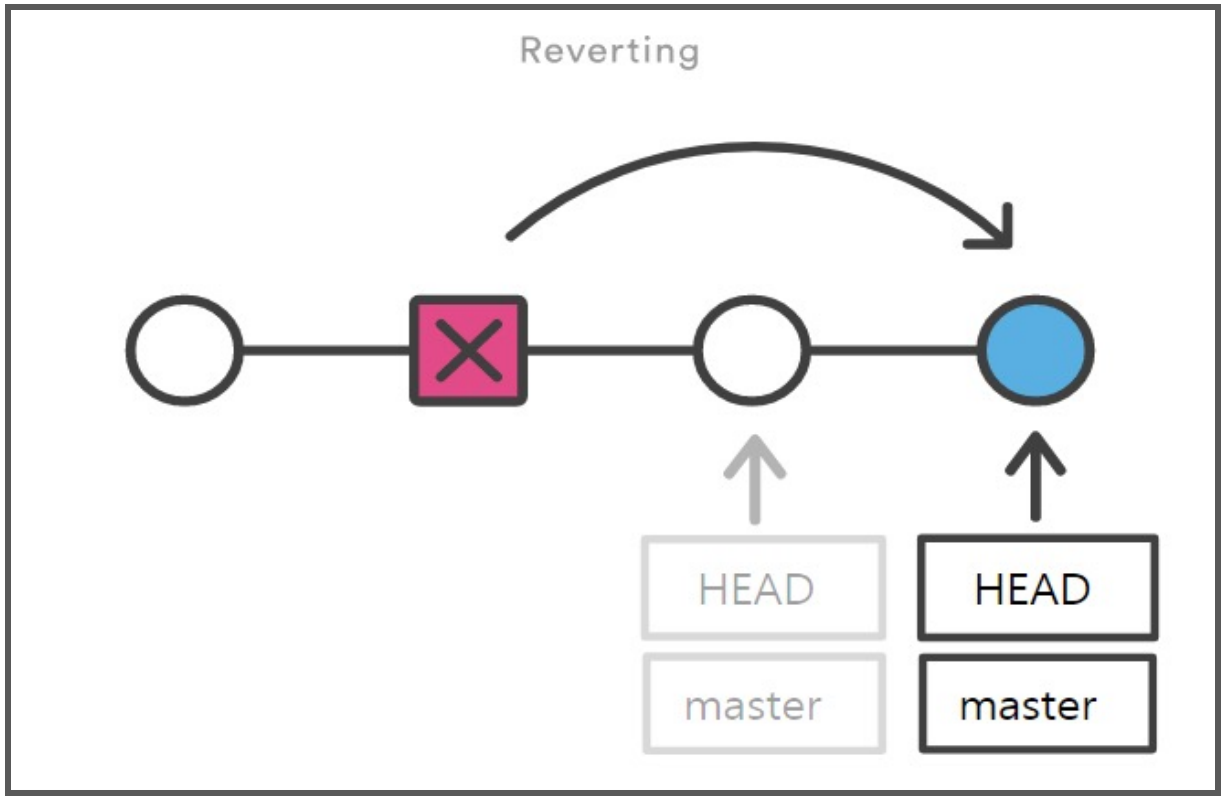
협업 시에는 절대 사용할 수 없음!!!!

버전 되돌리기 : git revert <checksum>

- 공개된 커밋의 변경 내역을 되돌리기
- 실제 되돌리는 것이 아니고 되돌리는 것과 같은 효과를 내는 것
- 대상 커밋의 변경 내역을 거꾸로 적용하는 새 커밋을 만드는 것
- git reset 보다 안전한 방법(협업 시 사용)
- 커밋의 **역순**으로 한 단계씩 변경 내역을 되돌려야 충돌을 피할 수 있음
- 한 단계씩 돌아가는 것은 HEAD를 사용
  - git revert HEAD

```
jinsook@klein MINGW64 /d/1-4. 오픈소스/1.강의노트/2020/gitTest (master)
$ git revert d0ff6e875c92c426130dedccb4389ef9afc18e4b
[master 56a3ea5] Revert "second commit - tuesday"
 1 file changed, 1 insertion(+), 2 deletions(-)

jinsook@klein MINGW64 /d/1-4. 오픈소스/1.강의노트/2020/gitTest (master)
$ git log
commit 56a3ea576d4ab387c87bd738d43e7aff202b29fe (HEAD -> master)
Author: jinsook <putzmunter7@gmail.com>
Date:   Tue Nov 3 00:39:14 2020 +0900

    Revert "second commit - tuesday"

    This reverts commit d0ff6e875c92c426130dedccb4389ef9afc18e4b.

commit 2f2dabb0f492b138180c4f2934c599fa367f6520
Author: jinsook <putzmunter7@gmail.com>
Date:   Tue Nov 3 00:38:38 2020 +0900

    fourth commit

commit 807d9421b63bd6b915e176d89c753b995b88dca3
Author: jinsook <putzmunter7@gmail.com>
Date:   Mon Nov 2 14:08:46 2020 +0900

    third commit - wednesday

commit d0ff6e875c92c426130dedccb4389ef9afc18e4b
Author: jinsook <putzmunter7@gmail.com>
Date:   Mon Nov 2 13:59:51 2020 +0900

    second commit - tuesday

commit 8f46f84d2709c5cdb6ef782c793150c36b75c973
Author: jinsook <putzmunter7@gmail.com>
Date:   Mon Nov 2 13:58:41 2020 +0900
```





출처 : https://gcapes.github.io/git-course/07-undoing/

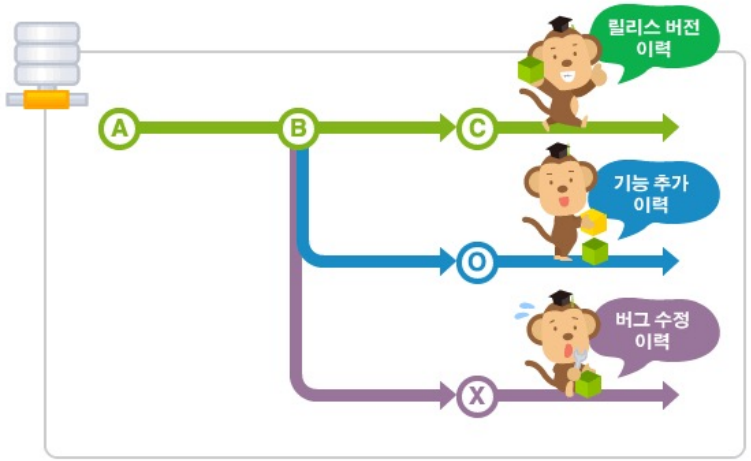
Git branch

버전 관리 용어

- 저장소(**Repository**)
  - 프로젝트의 현재 버전과 변경 이력 등의 정보를 저장하는 저장소
- 가져오기(**Import**)
  - 버전 관리되고 있지 않은 로컬 디렉토리의 파일을 처음으로 저장소(repository)에 복사함
- 체크 아웃(**Check out**)
  - 저장소(repository)에서 파일을 가져옴
- 체크 인(**Check In**, **commit**)
  - 체크 아웃(check out)한 파일의 수정이 끝난 경우 저장소(respository)에 새로운 버전으로 갱신하는 작업
  - 이때 이전에 갱신된 것이 있는 경우에 충돌(conflict)을 알려주며, diff 도구를 이용해 수정하고 commit하는 과정을 거침

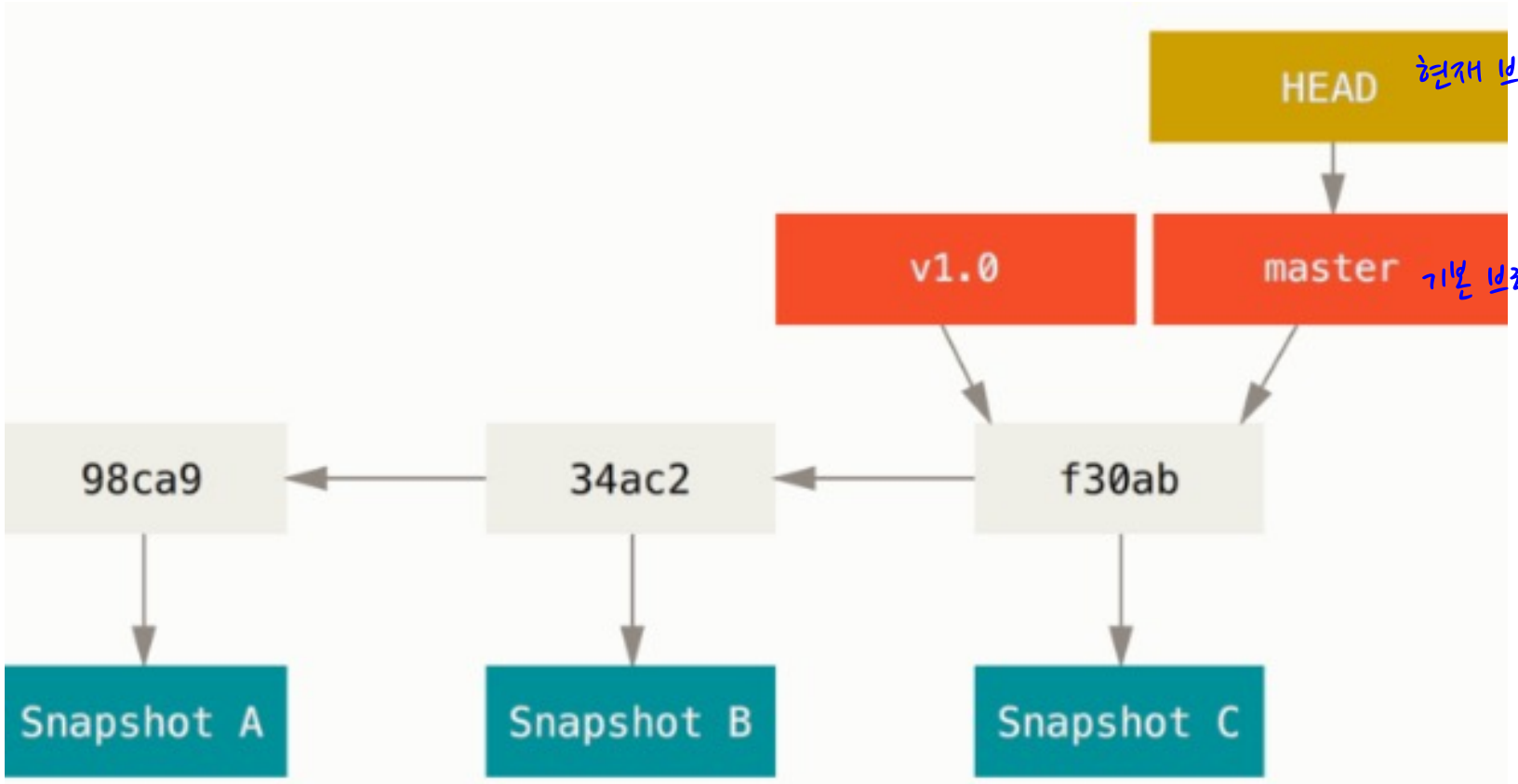
branch

- 기존 프로젝트와 독립적으로 개발을 진행할 수 있는 단위
- 현재 프로젝트가 진행되는 상황에서 **추가적인 기능**을 만들 때를 가정
  - 추가 기능은 현재 프로젝트의 메인 기능에 영향을 주면 안 됨
  - 메인 작업과 추가 작업은 개별적으로 진행됨
- git의 강력한 기능이며 다른 것과 구별되는 특징이 **branch**
- 브랜치의 작업 내용은 merge 과정을 통해 메인 작업에 통합 가능



참고사이트
https://backlog.com/git-tutorial/kr/stepup/stepup1_1.html

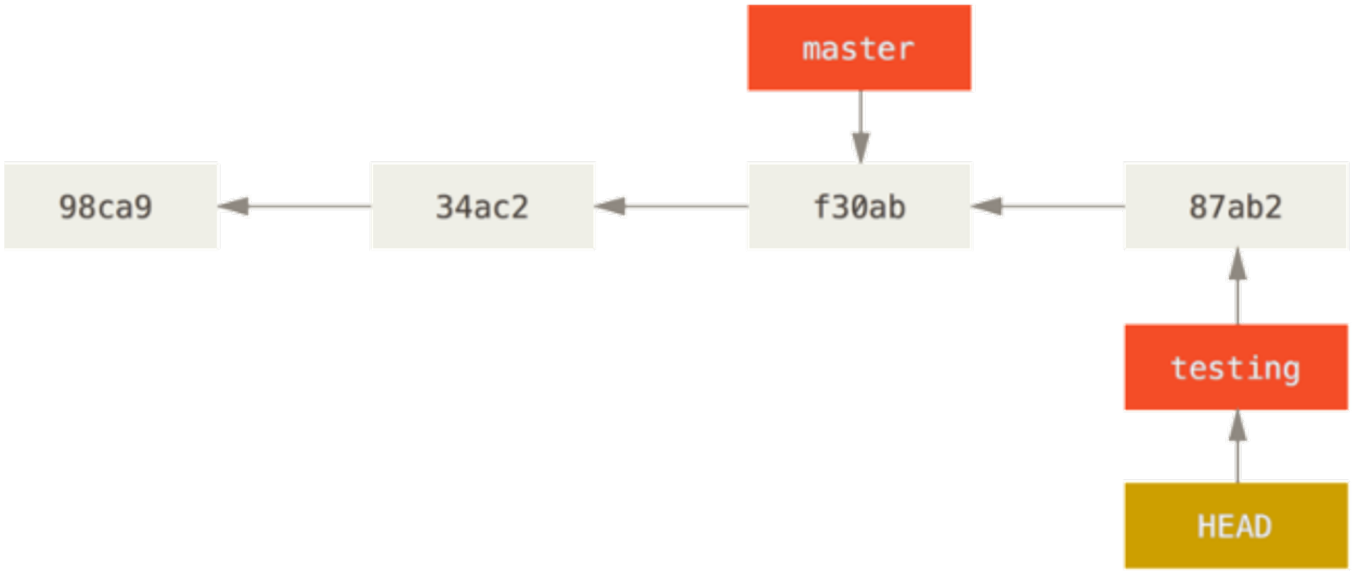
- 기본 브랜치와 커밋 히스토리



- 브랜치 새로 만들기
  - **$ git branch testing**
  - 새로 만든 브랜치도 지금 작업 중인 마지막 커밋을 가리킨다.

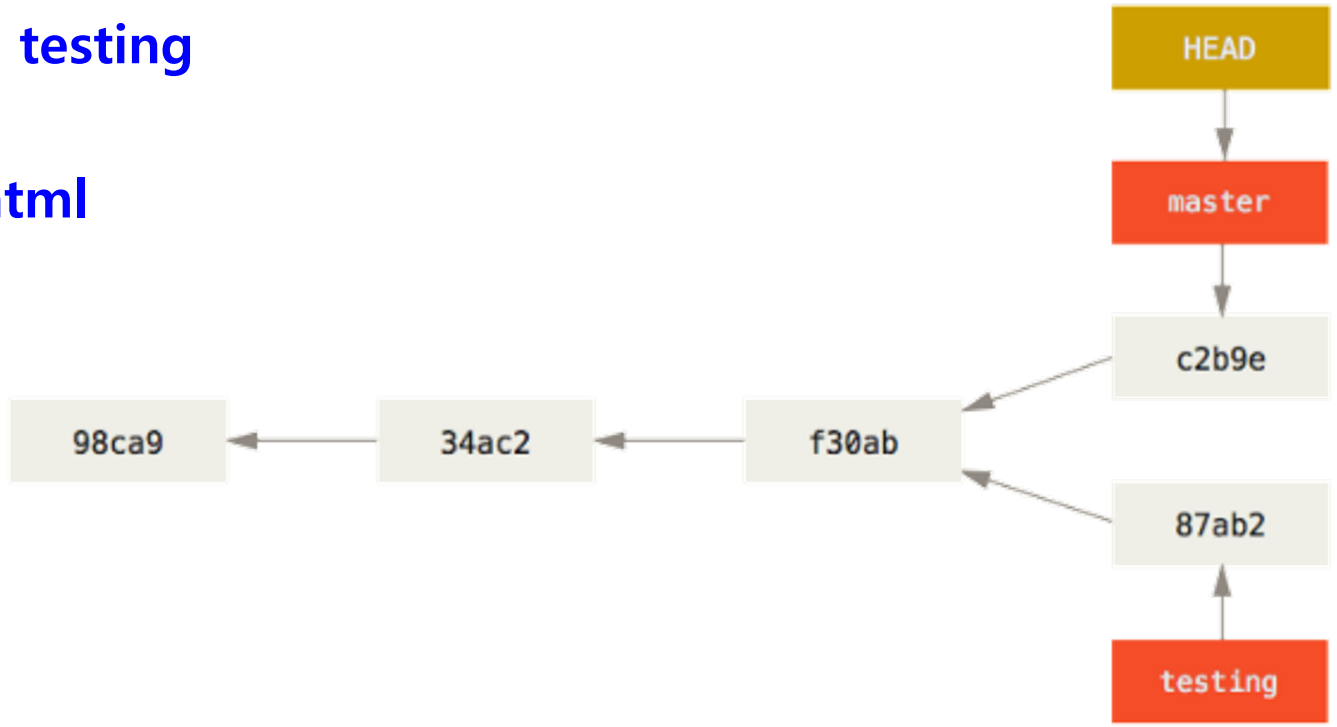
- 브랜치 이동하기
  - **$ git checkout testing**
  - 브랜치를 이동시키면 포인터 HEAD는 testing을 가리킨다.

- 브랜치에 따른 파일상태 확인하기(testing branch에서 작업)
  - **$ vi hello.html** *//파일에 문장을 추가한다.*
  - **$ git add hello.html**
  - **$ git commit –m “make a change”**
  - 새로 커밋해서 testing브랜치는 앞으로 이동
  - master 브랜치는 여전히 이전 커밋을 가리킴



- master 브랜치로 이동하여 같은 파일을 변경
  - **$ git checkout master**
  - **$ vi hello.html** *//파일 변경을 한다.* 한번도 add되지 않은 파일은 git add명령을 해주어야 함
  - **$ git commit –a –m "made other change"** *//add 와 commit 을 한꺼번에 실행*

- master 브랜치의 hello.html파일과 testing 브랜치의 hello.html파일 내용을 확인
  - **$ git branch** *//현재 브랜치 확인(master)*
  - **$ cat hello.html**
  - **$ git branch testing**
  - **$ git branch**
  - **$ cat hello.html**

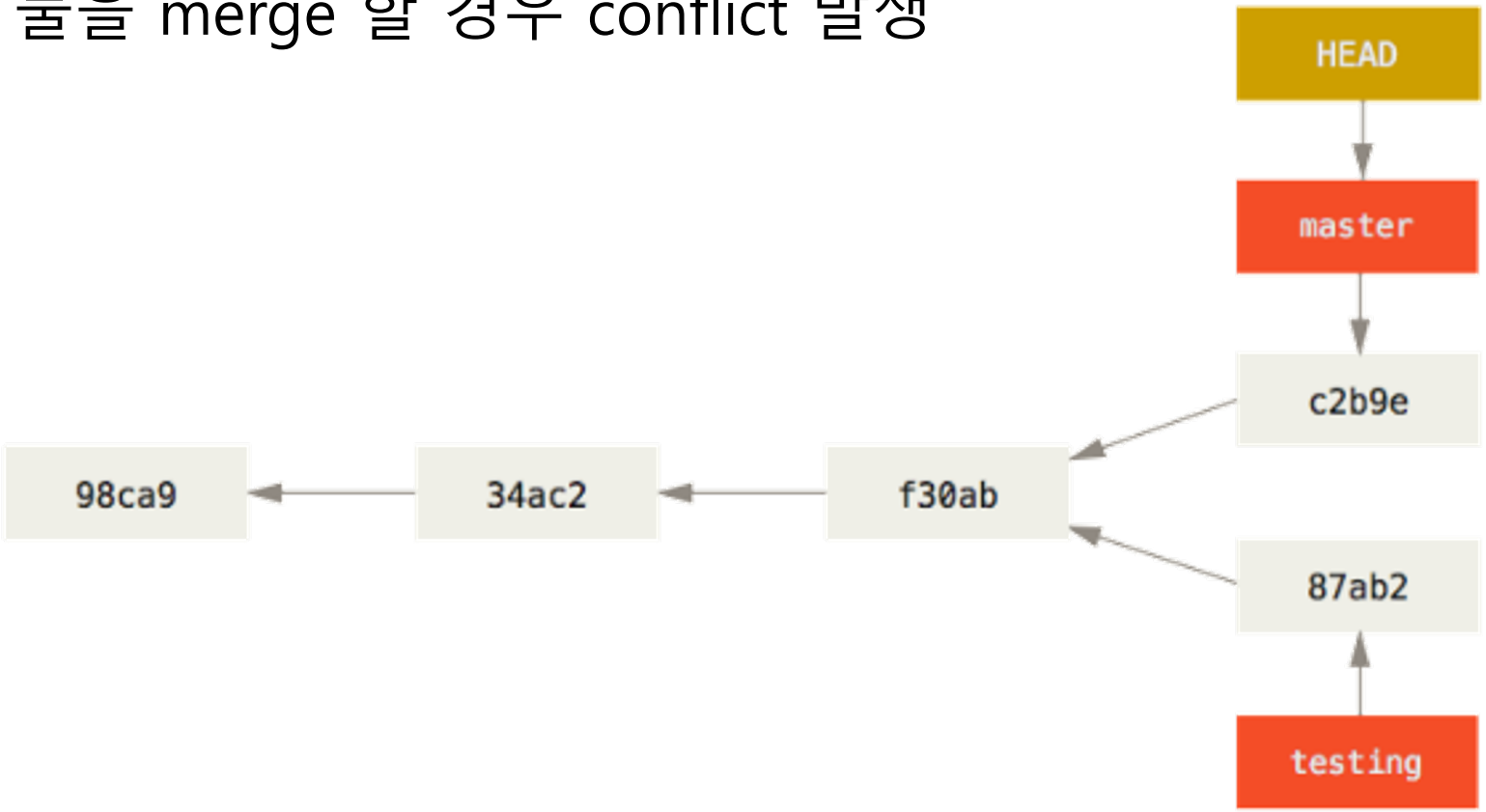


branch 관련 명령

명령어	설명
git branch	현재 사용하고 있는 브랜치(*)와 생성된 브랜치 목록을 화면에 출력
git branch -v	브랜치마다 마지막 커밋 메시지 출력
git branch <branch명>	<branch명>으로 branch 생성
git checkout <branch명>	<branch명>으로 branch 변경
git checkout -b <branch명>	해당 브랜치를 생성하면서 checkout까지(브랜치변경) 하는 명령
git branch –d <branch명> **git branch –D <branch명>**	브랜치 삭제(현재 브랜치가 아닌 것만 삭제 가능)
git diff	충돌 시 충돌 내용 화면 출력
git merge --abort	

conflict

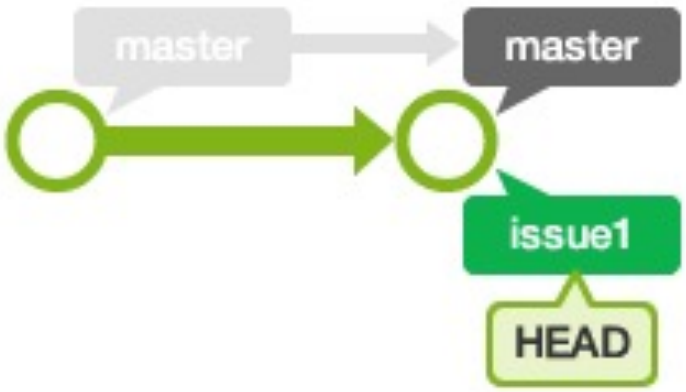
- 서로 다른 branch끼리 병합했을 때 충돌이 일어나는 현상
- 서로 다른 branch에서 같은 부분을 수정했을 때 conflict 발생
  - 아래 그림에서 master와 testing에서 각각 같은 파일을 수정
  - 이 둘을 merge 할 경우 conflict 발생



merge

- 여러 개의 브랜치를 하나로 모을 수 있음
- 변경 내용의 이력이 모두 그대로 남아 있음
  - **$ git merge testing**
  - master브랜치가 가리키는 커밋이 testing과 같은 위치로 이동

    → fast-forward(빨리 감기 병합)




```
$ git status
On branch testing
nothing to commit, working tree clean
$ ls -la
total 0
drwxrwxr-x 1 gildong gildong 4096 Nov 21 18:36 .
drwxr-xr-x 1 gildong gildong 4096 Nov 21 18:36 ..
drwxrwxr-x 1 gildong gildong 4096 Nov 21 18:37 .git
-rw-rw-r-- 1 gildong gildong  196 Nov 21 18:22 hello.html
-rw-rw-r-- 1 gildong gildong   15 Nov 21 18:00 test.rb
-rw-rw-r-- 1 gildong gildong   17 Nov 21 18:36 test2.txt
$ git checkout master
Switched to branch 'master'
$ ls -la
total 0
drwxrwxr-x 1 gildong gildong 4096 Nov 21 18:38 .
drwxr-xr-x 1 gildong gildong 4096 Nov 21 18:36 ..
drwxrwxr-x 1 gildong gildong 4096 Nov 21 18:38 .git
-rw-rw-r-- 1 gildong gildong  196 Nov 21 18:22 hello.html
$ git branch
* master
  testing
$ git merge testing
Updating bc1845f..cb4964b
Fast-forward
 test.rb   | 1 +
 test2.txt | 1 +
 2 files changed, 2 insertions(+)
 create mode 100644 test.rb
 create mode 100644 test2.txt
$ ls
hello.html  test.rb  test2.txt
```

- 두 브랜치를 병합
  - **$ git checkout master**
  - **$ git merge testing** *//conflict 발생*

```
$ git merge testing
Auto-merging hello.html
CONFLICT (content): Merge conflict in hello.html
Automatic merge failed; fix conflicts and then commit the result.
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both modified:   hello.html

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

- conflict 해결 – 파일을 수정한다.
  - **$ vi hello.html**
  - **$ git add hello.html**
  - **$ git commit –m “final hello”**

```
$ git commit -m "merged"
[master 1e42ac2] merged
```

  - **$ git log** *//작업 확인*

```
<html>
        <head>
                <title>연습입니다.</title>
        </head>
        <body>
                <h2>Hello, World!!</h2>
                <hr>
                참 반갑습니다.<br>
                오늘도 좋은 하루 되기를 바랍니다.
<<<<<< HEAD
                master에서 변경하였습니다.
=======
                오늘은 금요일입니다.
>>>>>>> testing
        </body>
</html>
```

충돌난 부분을 수작업으로 정리한다.

- **$ git log** *//작업 확인*
  - 가장 상위의 내용이 가장 최신

```
$ git log
commit 1e42ac2b9d1d1c43edee06facf9a8705b8c3f671 (HEAD -> master)
Merge: 436151d 6a82ee8
Author: Hong, gildong <gildong@gmail.com>
Date:   Fri Nov 22 01:02:23 2019 +0900

    merged

commit 436151d2ab8f1699e6b06ae5dbdb51a54cdf54fb
Author: Hong, gildong <gildong@gmail.com>
Date:   Fri Nov 22 00:34:25 2019 +0900

    master

commit 6a82ee80a7440226a069d2106b2e2251cf2379f4 (testing)
Author: Hong, gildong <gildong@gmail.com>
Date:   Fri Nov 22 00:31:35 2019 +0900

    testing 변경

commit cb4964b0ca237ebe9512eaea944ebdd81531e16b
Author: Hong, gildong <gildong@gmail.com>
Date:   Thu Nov 21 18:37:28 2019 +0900

    test2 commit first

commit bc1845f40d9258af80d034638fd1d6e3d6ec98a7
Author: Hong, gildong <gildong@gmail.com>
Date:   Thu Nov 21 00:06:03 2019 +0900

    third commit

commit a077d4f13ee2692ddeb510ae0c03bfd3752a5316
Author: Hong, gildong <gildong@gmail.com>
Date:   Thu Nov 21 00:03:01 2019 +0900
```

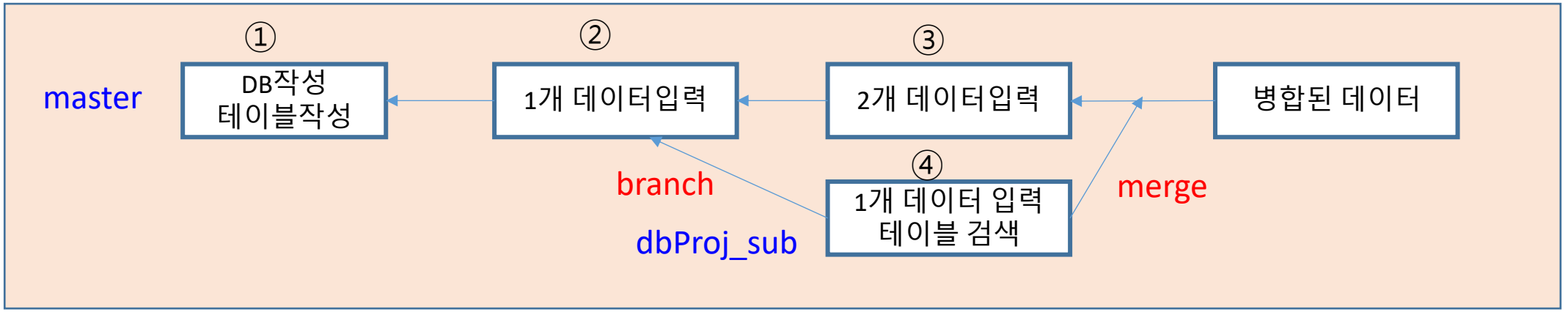
실습 : branch

한 회사의 고객정보 데이터베이스 구축을 위한 프로젝트를 실행 중에 있다.
git으로 모든 파일을 관리하고 있다. 프로젝트 진행 중 다음과 같이 local repository 에 발생하는 conflict를 해결 하시오.

- 워킹 디렉토리 : dbProject
- 위의 실습을 위해 필요한 명령어들을 나열하여 보시오

```
git init  / git add  / git commit  / git status  / git log  / git
branch  / git checkout / git config -global user.name
git diff / git merge --abort
```

실습 : branch - 과제



① 파일명 : company.sql

```
create database company;

use company;

create table dept(
        id varchar(10),
        name varchar(50),
        chief varchar(10)
);
```

②

```
insert into dept values('001', '기획부', '홍길동');
```

③

```
insert into dept values('002', '총무부', '이경성');

insert into dept values('003', '인사부', '한이성');
```

④

```
insert into dept values('004', '경리부', '김성경');


select * from dept;
```