**튜토리얼1: 브랜치를 사용해보자**

**0. 사전 준비**

실습을 위해 먼저 Git 저장소를 만들어야 합니다.

아래 코드를 이용하여 tutorial 이라는 이름으로 새 폴더를 만들고 Git 저장소로 지정해 봅시다.

$ **mkdir tutorial**

$ **cd tutorial**

$ **git init**

Initialized empty Git repository in /Users/eguchi/Desktop/tutorial/.git/

tutorial 폴더에 myfile.txt 이라는 이름으로 파일을 만든 후 커밋합니다.

**원숭이도 이해할 수 있는 Git 명령어**

$ **git add myfile.txt**

$ **git commit -m "first commit"**

[master (root-commit) a73ae49] first commit

1 files changed, 1 insertions(+), 0 deletions(-)

create mode 100644 myfile.txt

여기까지 진행했다면, 아래 그림과 같은 이력이 남게 됩니다.



# 1. 브랜치 만들기

'issue1' 이라는 이름으로 새로운 브랜치를 작성합니다.

브랜치는 branch 란 명령어로 만들 수 있습니다.

$ git branch <branchname>

'issue1' 이라는 이름으로 브랜치를 만들어 봅시다.

$ **git branch issue1**

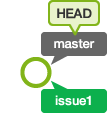
옵션을 지정하지 않고 branch 명령어를 실행하면 브랜치 목록 전체를 확인할 수 있습니다. 앞 부분에 \* 이 붙어있는 것이 현재 선택된 브랜치입니다.

$ **git branch**

issue1

\* master

이 시점까지의 이력을 보면 다음과 같습니다.



# 2. 브랜치 전환하기

앞에서 우리가 새로 만든 'issue1'라는 이름의 브랜치를 사용하여 어떤 작업을 수행하려면, 이 브랜치를 사용 하겠다고 명시적으로 지정해 주어야 합니다. 이 때 사용하는 명령어가 바로 checkout 입니다. 체크아웃(checkout)이란, 내가 사용할 브랜치를 지정하는 것을 의미합니다.

다음과 같이checkout 명령어 뒤에 사용할 브랜치 이름을 입력하면 됩니다.

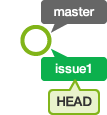
$ git checkout <branch>

아래와 같이 입력하여 'issue1' 브랜치를 체크아웃 해 봅시다.

$ **git checkout issue1**

Switched to branch 'issue1'

이 시점까지의 이력을 보면 아래와 같습니다.



Note

checkout 명령에 -b 옵션을 넣으면 브랜치 작성과 체크아웃을 한꺼번에 실행할 수 있습니다.

$ git checkout -b <branch>

'issue1' 브랜치를 체크아웃한 상태에서 커밋을 수행하면, 'issue1' 브랜치에 그 이력이 기록됩니다. mylife.txt 에 아래 박스에서와 같이 문장을 추가한 후에 커밋해봅시다.

원숭이도 이해할 수 있는 Git 명령어

**add: 변경 사항을 만들어서 인덱스에 등록해보기**

$ **git add myfile.txt**

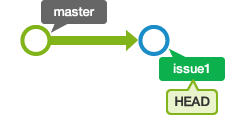
$ **git commit -m "add 설명을 추가"**

[issue1 b2b23c4] add 설명을 추가

1 files changed, 1 insertions(+), 0 deletions(-)

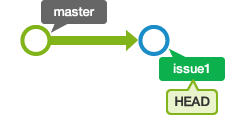
위와 같이 commit 명령어에 -m 옵션을 넣으면 커밋 설명을 포함시킬 수 있습니다.

이 시점까지의 이력을 보면 다음과 같습니다.



# 3. 브랜치 병합하기

이번에는 'issue1' 브랜치의 변경 사항을 'master' 브랜치에 병합해 볼까요?



브랜치 병합은 merge 명령어로 실행합니다. 이 명령어에 병합할 커밋 이름을 넣어 실행하면, 지정한 커밋 내용이 'HEAD'가 가리키고 있는 브랜치에 넣어집니다. 'HEAD'는 현재 사용중인 브랜치에 위치하게 됩니다. 위 그림에서는 'issue1' 커밋에 'HEAD'가 위치하고 있습니다.

$ git merge <commit>

'master' 브랜치에 'issue1'를 넣기 위해서는 우선 'master' 브랜치에 'HEAD'가 위치하게 만들어야 합니다. 이 때에는 checkout 명령어를 이용하여 현재 사용중인 브랜치를 'master'로 전환합니다.

$ **git checkout master**

Switched to branch 'master'

병합하기 전에 myfile.txt 파일을 열어 내용을 확인합니다.

원숭이도 이해할 수 있는 Git 명령어

이번 실습에서의 파일 편집은 'issue1' 브랜치에서 실행 했기 때문에 'master' 브랜치로 브랜치를 전환한 지금, myfile.txt 파일을 확인했을 때 그 내용이 변경되어 있지 않아야 합니다.

그럼 이제 병합을 시작해볼까요?

$ **git merge issue1**

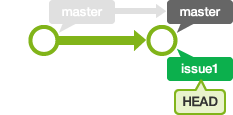
Updating 1257027..b2b23c4

Fast-forward

myfile.txt | 1 +

1 files changed, 1 insertions(+), 0 deletions(-)

이제 'master' 브랜치가 가리키는 커밋이 'issue1'과 같은 위치로 이동했습니다. 이런 방식의 병합을 'fast-forward (빨리감기) 병합'이라고 합니다.



myfile.txt 파일을 열어 내용을 확인해 봅시다.

원숭이도 이해할 수 있는 Git 명령어

**add: 변경 사항을 만들어서 인덱스에 등록해보기**

「add: 변경 사항을 만들어서 인덱스에 등록해보기」 내용이 추가되어 있는 것을 확인합니다.

# 4. 브랜치 삭제하기

'issue1' 브랜치의 내용이 모두 'master'에 통합 되었기 때문에 이제 더 이상 'issue1' 브랜치가 필요없게 되었습니다.

브랜치를 삭제하려면 branch 명령에 -d 옵션을 지정하여 실행하면 됩니다.

$ git branch -d <branchname>

'issue1' 브랜치를 삭제하려면, 다음 명령어를 실행합니다.

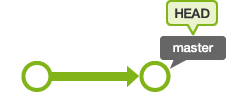
$ **git branch -d issue1**

Deleted branch issue1 (was b2b23c4).

이제 'issue1' 브랜치는 삭제되었습니다. 정말로 브랜치가 잘 삭제 되었는지 branch 명령어를 이용해서 확인합니다. 아래와 같이 'master' 브랜치만 목록에 남아 있게 됩니다.

$ **git branch**

\* master



# 5. 동시에 여러 작업하기

이번에는 두 개의 브랜치를 생성하여 동시에 여러 작업을 처리하는 상황을 만들어 보겠습니다.

'issue2' 와 'issue3' 브랜치를 만든 후, 'issue2' 브랜치로 전환 합니다.

$ **git branch issue2**

$ **git branch issue3**

$ **git checkout issue2**

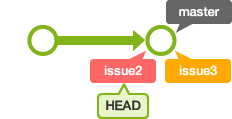
Switched to branch 'issue2'

$ **git branch**

\* issue2

issue3

master



'issue2' 브랜치의 myfile.txt 에 아래와 같이 commit 에 대한 설명을 추가하여 커밋해 보도록 하겠습니다.

원숭이도 이해할 수 있는 Git 명령어

add: 변경 사항을 만들어서 인덱스에 등록해보기

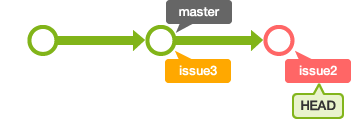
**commit: 인덱스 상태를 기록하기**

$ **git add myfile.txt**

$ **git commit -m "commit의 설명 추가"**

[issue2 8f7aa27] commit의 설명 추가

1 files changed, 2 insertions(+), 0 deletions(-)



이번에는 'issue3' 브랜치로 전환 합니다.

$ **git checkout issue3**

Switched to branch 'issue3'

myfile.txt 파일을 열어 내용을 확인합니다. commit 명령의 설명은 'issue2' 브랜치에서 추가했기 때문에, 'issue3' 브랜치로 전환한 후의 myfile.txt 파일에는 add 명령의 설명 밖에 없습니다.

이번에는 pull 명령어의 설명을 추가하여 변경 사항을 커밋해 보도록 하겠습니다.

원숭이도 이해할 수 있는 Git 명령어

add: 변경 사항을 만들어서 인덱스에 등록해보기

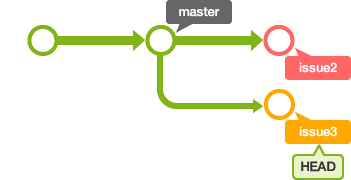
**pull: 원격 저장소의 내용을 가져오기**

$ **git add myfile.txt**

$ **git commit -m "pull 설명을 추가"**

[issue3 e5f91ac] pull 설명을 추가

1 files changed, 2 insertions(+), 0 deletions(-)



지금까지 'issue2' 브랜치에 commit 에 대한 설명을, 'issue3' 브랜치에 pull 에 대한 설명을 포함하여 커밋해 보았습니다. 이처럼 각각의 브랜치에서는 독립적으로 서로 다른 작업을 처리할 수 있습니다.

다음 단계에서는 이 두 브랜치를 병합하여, 내용이 달라진 myfile.txt 파일에 대한 충돌을 해결하는 방법에 대해 알아보도록 하겠습니다.

# 6. 병합할 때 발생하는 충돌 해결하기

이번에는 'issue2' 브랜치에서 변경한 부분과 'issue3' 브랜치에서 변경한 부분을 모두 'master' 브랜치에 통합해 보도록 하겠습니다.

먼저 'master' 브랜치를 체크아웃한 다음 'issue2' 브랜치를 병합합니다.

$ **git checkout master**

Switched to branch 'master'

$ **git merge issue2**

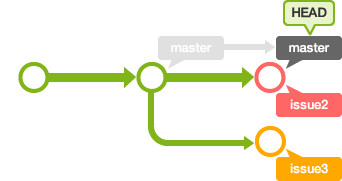
Updating b2b23c4..8f7aa27

Fast-forward

myfile.txt | 2 ++

1 files changed, 2 insertions(+), 0 deletions(-)

이렇게 하면 앞서 설명되었던 'fast-forward(빨리감기) 병합'이 실행됩니다. 아래 그림을 보면, 'master' 브랜치에 'issue2' 브랜치가 병합된 것을 확인할 수 있습니다.



이번에는 'issue3' 브랜치를 병합합니다.

$ **git merge issue3**

Auto-merging myfile.txt

CONFLICT (content): Merge conflict in myfile.txt

Automatic merge failed; fix conflicts and then commit the result.

'CONFLICT(충돌)'이라 나오는 것을 보니 자동 병합에 실패한 것 같습니다. 앞서 각각의 브랜치에서 변경한 내용이 myfile.txt 의 같은 행에 포함되어 있기 때문입니다. 실제로 myfile.txt 의 내용을 확인해 보면 다음과 같이 변경되어 있습니다.

원숭이도 이해할 수 있는 Git 명령어

add: 변경 사항을 만들어서 인덱스에 등록해보기

<<<<<<< HEAD

commit: 인덱스의 상태를 기록하기

=======

pull: 원격 저장소의 내용을 가져오기

>>>>>>> issue3



충돌이 있는 부분에 Git 이 자동으로 위와 같이 충돌 정보를 포함하여 파일 내용을 변경합니다. 이 내용을 통해 어떤 브랜치에서 어떤 부분이 충돌되었는지 확인할 수 있습니다. 충돌이 일어난 부분은 이렇게 일일이 확인해서 수정해 주어야 합니다. 그럼 아래와 같이 파일 내용을 변경해 봅시다.

원숭이도 이해할 수 있는 Git 명령어

add: 변경 사항을 만들어서 인덱스에 등록해보기

**commit: 인덱스의 상태를 기록하기**

**pull: 원격 저장소의 내용을 가져오기**

충돌 부분을 모두 수정했으므로, 다시 커밋합니다.

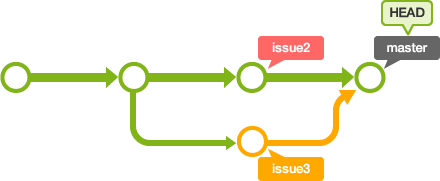
$ **git add myfile.txt**

$ **git commit -m "issue3 브랜치 병합"**

# On branch master

nothing to commit (working directory clean)

이 시점까지의 이력을 보면 다음과 같습니다. 이번 병합은 충돌 부분을 수정했기 때문에 그 변화를 기록하는 병합 커밋이 새로 생성 되었습니다. 그리고 'master' 브랜치의 시작('HEAD')이 그 위치로 이동해 있는 것을 확인할 수 있습니다. 아래와 같은 방식을 'non fast-forward 병합'이라고 합니다.

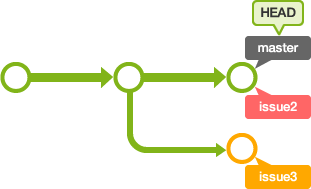


# 7. rebase 로 병합하기

앞선 튜토리얼에서 우리는 두 개의 브랜치를 'master' 브랜치로 모두 병합 시켰습니다. 그로 인해 두 개의 줄기로 브랜치가 분기 되었다가 다시 하나로 합쳐지는 것을 확인 했었습니다.

'issue3' 브랜치를 병합 할 때에 rebase 를 먼저 실행한 후 병합을 시도한다면 그 이력을 하나의 줄기로 만들 수도 있습니다. 이번에는 이와 같은 경우를 만들어 보도록 하겠습니다. 이를 위해 일단 이전의 튜토리얼에서 마지막으로 진행했던 병합 명령을 취소합니다.

$ **git reset --hard HEAD~**



이제 'issue3' 브랜치로 전환하여 'master' 브랜치에 rebase 를 실행합니다.

$ **git checkout issue3**

Switched to branch 'issue3'

$ **git rebase master**

First, rewinding head to replay your work on top of it...

Applying: pull 설명을 추가

Using index info to reconstruct a base tree...

<stdin>:13: new blank line at EOF.

+

warning: 1 line adds whitespace errors.

Falling back to patching base and 3-way merge...

Auto-merging myfile.txt

CONFLICT (content): Merge conflict in myfile.txt

Failed to merge in the changes.

Patch failed at 0001 pull 설명을 추가

When you have resolved this problem run "git rebase --continue".

If you would prefer to skip this patch, instead run "git rebase --skip".

To check out the original branch and stop rebasing run "git rebase --abort".

merge 때와 마찬가지로 myfile.txt 파일 내용에 충돌이 있기 때문에 그 부분을 이전의 튜토리얼에서와 동일하게 처리합니다. 충돌난 파일 내용을 적절히 변경해 주세요.

원숭이도 이해할 수 있는 Git 명령어

add: 변경 사항을 만들어서 인덱스에 등록해보기

<<<<<<< HEAD

commit: 인덱스의 상태를 기록하기

=======

pull: 원격 저장소의 내용을 가져오기

>>>>>>> issue3

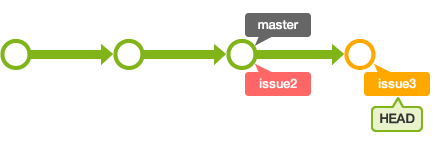
rebase 의 경우 충돌 부분을 수정 한 후에는 commit 이 아니라 rebase 명령에 --continue 옵션을 지정하여 실행해야 합니다.

$ **git add myfile.txt**

$ **git rebase --continue**

Applying: pull 설명을 추가

만약 rebase 자체를 취소하려면 --abort 옵션을 지정하면 됩니다.



이전 튜토리얼에서 merge 명령어를 사용했을 때와 같이, rebase 만 실행한 경우에는 위의 그림처럼 'issue3' 브랜치가 두 브랜치의 앞 쪽으로 위치가 옮겨졌을 뿐 'master' 브랜치는 아직 'issue3'의 변경 사항이 적용되지 못한 상태로 뒤에 남겨져 있습니다. 이제 'master' 브랜치로 전환 하여 'issue3' 브랜치의 변경 사항을 모두 병합할 차례입니다.

$ **git checkout master**

Switched to branch 'master'

$ **git merge issue3**

Updating 8f7aa27..96a0ff0

Fast-forward

myfile.txt | 1 +

1 files changed, 1 insertions(+), 0 deletions(-)

myfile.txt 의 최종적인 내용은merge 했을 경우와 동일하지만, 그 이력은 아래 그림과 같이 달라집니다.

