제품소프트웨어 패키징
(Git과 GitHub를 활용한 소스코드 관리)

Git Branch 활용

한국기술교육대학교
온라인평생교육원

학습내용

- Git Branch 활용
- Git Branch 실습

학습목표

- Git Branch 명령어를 통해 Branch를 생성하고 삭제할 수 있다.
- Git Branch 의 관리 및 충돌 시 해결하는 방법을 설명할 수 있다.

Git Branch 활용

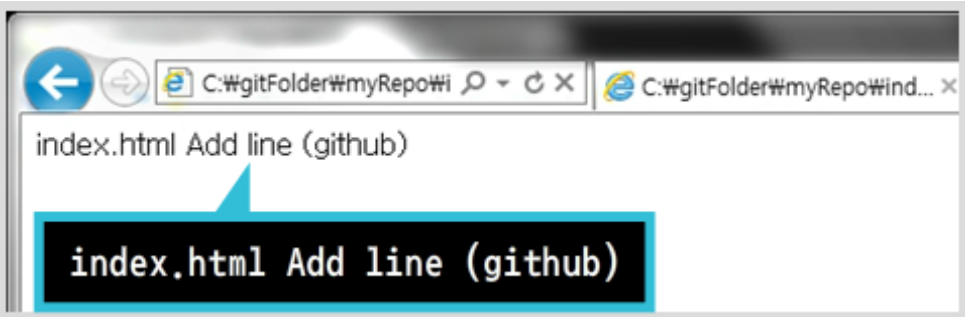
1. Git Branch 기본 동작

1) Branch 생성 및 활용 방법

- Issue 생성myRepo 저장소의 원본 파일 내용 확인
  - git bash창에서 myRepo 저장소의 index.html 파일을 이용

```
user@userpc MINGW64 /c/gitFolder/myRepo (master)
$ cat index.html
<!DOCTYPE html>
<html>
<body>
index.html
Add line (github)
</body>
</html>
```

- index.html 파일을 인터넷브라우저에서 확인



- topic1이름의 branch 생성 및 저장소 확인
  - '$ git branch <Branch 이름>' 명령어로 branch 생성

```
user@userpc MINGW64 /c/gitFolder/myRepo (master)
$ git branch topic1

user@userpc MINGW64 /c/gitFolder/myRepo (master)
$ git branch
* master
  topic1
```

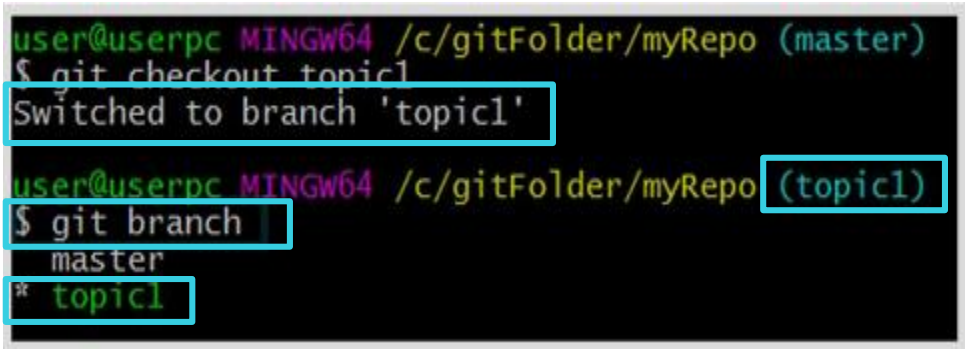
  - '$ git branch : 현재 저장소에서 관리하는 branch의 리스트 출력
  - topic1 : 현재 머무르고 있는 branch

Git Branch 활용

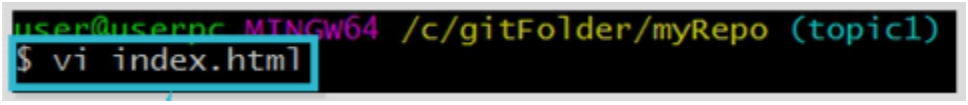
1. Git Branch 기본 동작

1) Branch 생성 및 활용 방법

- 생성된 topic1 branch로 이동 및 branch 확인
  - '$ git checkout <이동할 Branch 이름>' 명령어를 이용



- itopic1 branch에서 index.html 파일 내용 수정



  - git에서 어떻게 파일을 관리하는지가 중요

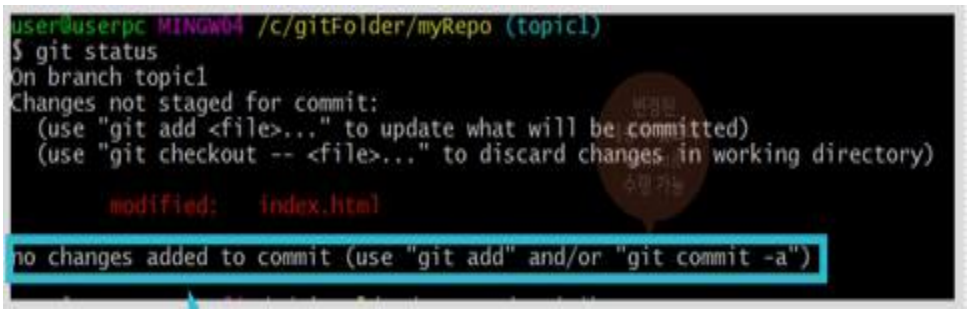
  - 데이터가 제대로 수정되어 있는지 확인

Git Branch 활용

1. Git Branch 기본 동작

1) Branch 생성 및 활용 방법

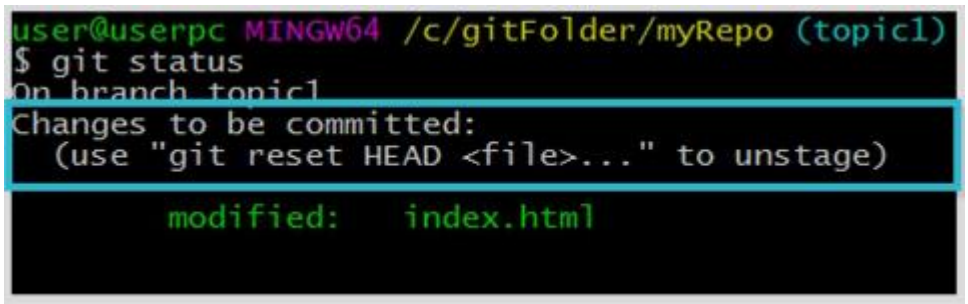
- 파일 수정이 완료되면 저장소 상태 확인



- 변경된 내용이 있으니 commit 수행 가능

- index.html이 수정되었기 때문에 변경사항 적용

- 다시 한 번 저장소 상태 확인



Git Branch 활용

1. Git Branch 기본 동작

1) Branch 생성 및 활용 방법

- commit 명령을 통해 수정된 index.html 파일 등록

```
user@userpc MINGW64 /c/gitFolder/myRepo (topic1)
$ git commit -m "update index.html (branch-topic1)"
[topic1 aa8d9c2] update index.html (branch-topic1)
 1 file changed, 35 insertions(+), 2 deletions(-)
```

  - 업데이트 된 내용에 대해 간단하게 설명 추가

- imaster branch로 이동 및 branch 확인

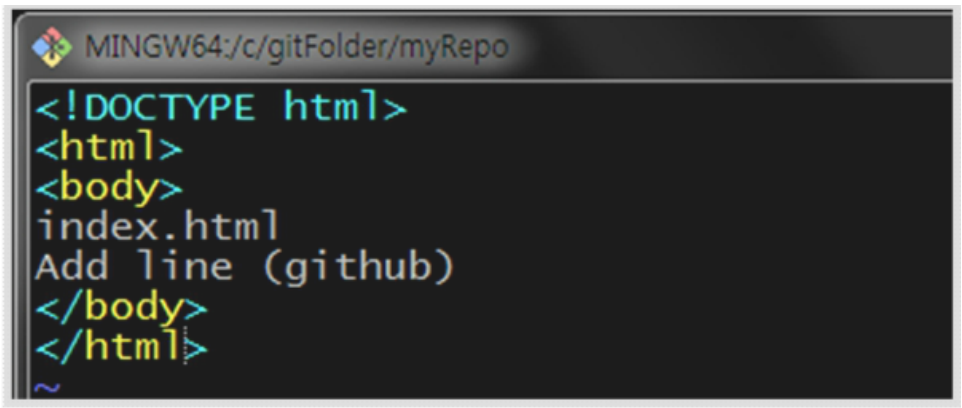
```
user@userpc MINGW64 /c/gitFolder/myRepo (topic1)
$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.

user@userpc MINGW64 /c/gitFolder/myRepo (master)
$ git branch
* master
  topic1
```

1. Git Branch 기본 동작

2) 업데이트한 index.html 파일 병합

- 병합하기 전 index.html 파일 확인



- topic1과 master branch 병합

1. Git Branch 기본 동작

2) 업데이트한 index.html 파일 병합

- 업데이트 된 index.html 파일 확인

```
user@userpc MINGW64 /c/gitFolder/myRepo (master)
$ vi index.html
```

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML</title>
        <meta charset="utf-8" />
        <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
    <header id="header">
        <h1>HTML Tag</h1>
    </header>
    <nav id="top_menu">
        <ul class="menu_list">
                    <li><a href="index.html">Home </a></li>
            <li><a href="tag/header.html">Tag - h</a></li>
            <li><a href="tag/form.html">Tag - form</a></li>
                        <li><a href="tag/table.html">Tag - html</a></li>
        </ul>
    </nav>
    <div id="wrap">
            <div id="content_wrap">
            <div id="content">
                <article>
                    <h1>HTML 이란</h1>
                                        </br>
                                        <p>HTML은 하이퍼텍스트 마크업 언어(HyperText Markup
Language, 문화어: 초본문표식달기언어, 하이퍼본문표식달기언어)라는 의미의 웹 페이지를 위한 기
배적인 마크업 언어다. HTML은 제목, 단락, 목록 등과 같은 본문을 위한 구조적 의미를 나타내는 >
것뿐만 아니라 링크, 인용과 그 밖의 항목으로 구조적 문서를 만들 수 있는 방법을 제공한다. 그리
고 이미지와 객체를 내장하고 대화형 양식을 생성하는 데 사용될 수 있다. HTML은 웹 페이지 콘텐
츠 안의 꺾쇠 괄호에 둘러싸인 "태그"로 되어있는 HTML 요소 형태로 작성한다. HTML은 웹 브라우저
와 같은 HTML 처리 장치의 행동에 영향을 주는 자바스크립트와 본문과 그 밖의 항목의 외관과 배치
를 정의하는 CSS 같은 스크립트를 포함하거나 불러올 수 있다. HTML과 CSS 표준의 공동 책임자인 W
3C는 명확하고 표상적인 마크업을 위하여 CSS의 사용을 권장한다.</p>
                                        </br>
                                        <p>출처 : https://ko.wikipedia.org/wiki/HTML</p>

                </article>

            </div>
        </div>
    </div>
    <footer id="footer">
        <h1>Git Github sample source</h1>
        <address>KOREATECH</address>
    </footer>
</body>
</html>
```

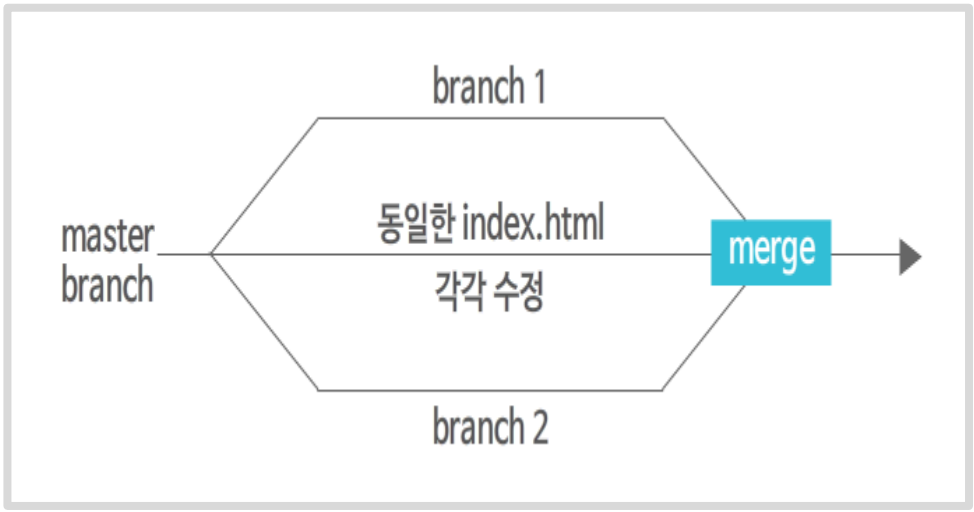
1. Git Branch 기본 동작

2) 업데이트한 index.html 파일 병합

- 사용하지 않는 topic1 branch 삭제 후 branch 확인

- 사용하지 않는 branch들은 삭제

2. Git Branch 응용



1) 두 개의 branch 생성

- git branch 명령어를 통하여 branch 생성

```
user@userpc MINGW64 /c/gitFolder/myRepo (master)
$ git branch
* master

user@userpc MINGW64 /c/gitFolder/myRepo (master)
$ git branch issue1

user@userpc MINGW64 /c/gitFolder/myRepo (master)
$ git branch issue2

user@userpc MINGW64 /c/gitFolder/myRepo (master)
$ git branch
  issue1
  issue2
* master
```

2. Git Branch 응용

2) issue1 branch에서 index.html 파일 수정

- issue1 branch로 이동

```
user@userpc MINGW64 /c/gitFolder/myRepo (master)
$ git checkout issue1
Switched to branch 'issue1'
```

- vi 에디터로 index.html 파일 수정

```
user@userpc MINGW64 /c/gitFolder/myRepo (issue1)
$ vi index.html
```

- index.html 파일에 링크 항목 추가 하고 저장

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML</title>
        <meta charset="utf-8" />
        <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
    <header id="header">
        <h1>HTML Tag</h1>
    </header>
    <nav id="top_menu">
        <ul class="menu_list">
                        <li><a href="index.html">Home </a></li>
            <li><a href="tag/header.html">Tag - h</a></li>
            <li><a href="tag/form.html">Tag - form</a></li>
                        <li><a href="tag/issue1.html">issue1</a></li>
        </ul>
    </nav>
    <div id="wrap">
            <d
                                                    </br>
```

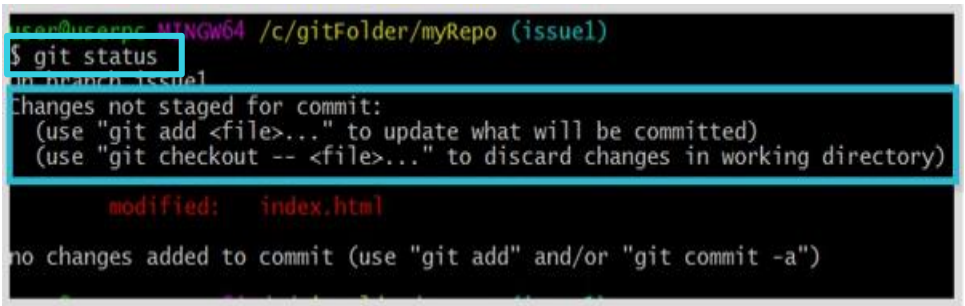
issue1

- 링크생성 라인

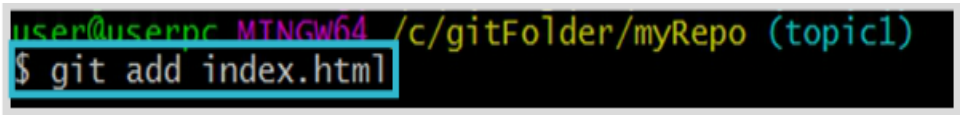
2. Git Branch 응용

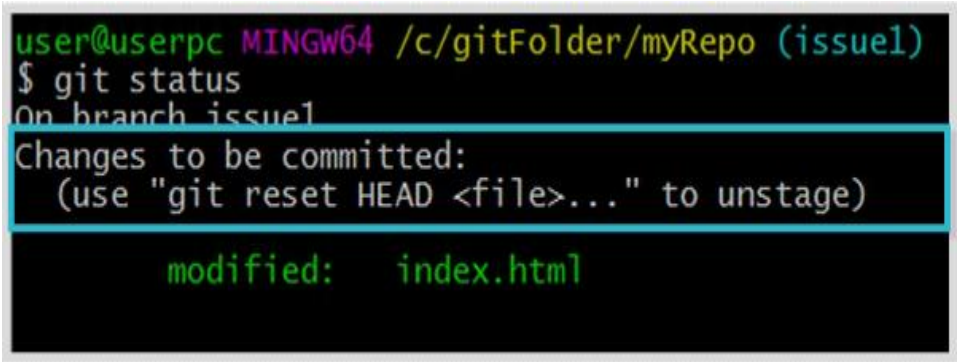
2) issue1 branch에서 index.html 파일 수정

- 로컬 저장소에서 업데이트 된 상태 확인

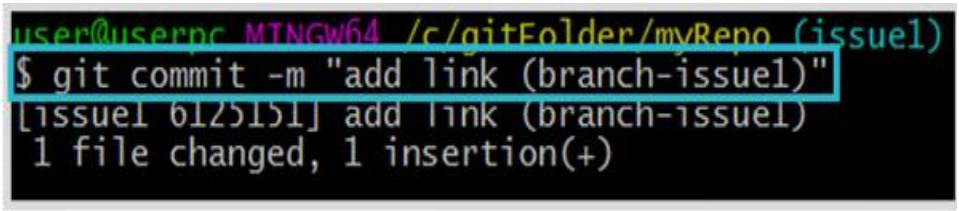


- index.html 파일을 추가 후 저장소 확인





- index.html를 로컬 저장소에 등록



- issue1 branch에서 작업한 내용

2. Git Branch 응용

3) issue2 branch에서 index.html 파일 수정

- issue2 branch 이동 후 branch 확인

```
user@userpc MINGW64 /c/gitFolder/myRepo (issue1)
$ git checkout issue2
Switched to branch 'issue2'
```

```
user@userpc MINGW64 /c/gitFolder/myRepo (issue2)
$ git branch
  issue1
* issue2
  master
```

- vi 에디터로 index.html 파일 수정

```
user@userpc MINGW64 /c/gitFolder/myRepo (issue2)
$ vi index.html
```

- index.html 파일에 링크 항목 추가 하고 저장

```
MINGW64:/c/gitFolder/myRepo
<!DOCTYPE html>
<html>
<head>
    <title>HTML</title>
        <meta charset="utf-8" />
        <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
    <header id="header">
    <nav
        <li><a href = " tag/table.html " >Tag - html</a></li>
            <li><a href="tag/header.html">Tag - h</a></li>
            <li><a href="tag/table.html">Tag - html</a></li>
    </ul>
```

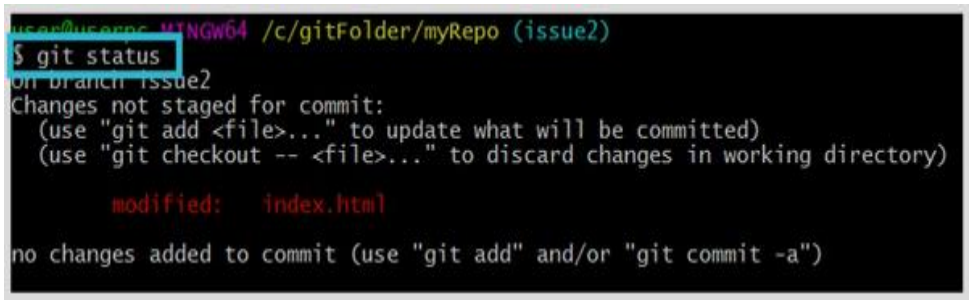
- Issue 1에서 수정된 내용은 issue2 branch에 적용되지 않음

```
<li><a href="tag/table.html">Tag - html</a></li>
<li><a href="tag/issue2.html">issue2</a></li>
```

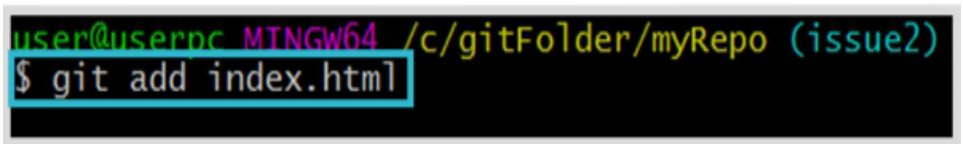
2. Git Branch 응용

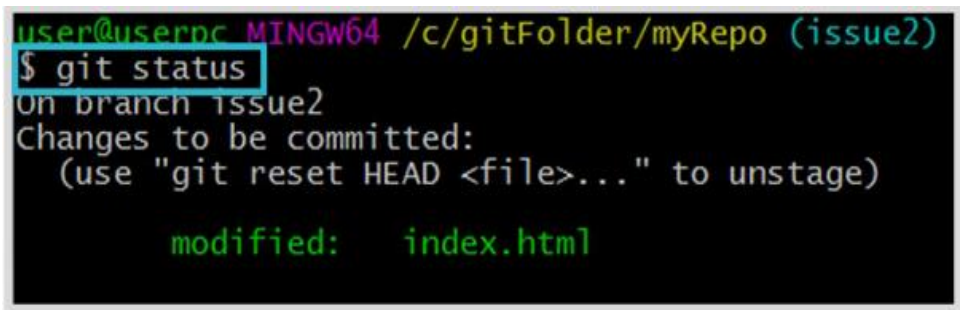
3) issue2 branch에서 index.html 파일 수정

- 로컬 저장소에서 업데이트 된 상태 확인



- index.html 파일을 추가 후 저장소 확인





- 저장소의 상태 변화를 확인하기 위해 사용

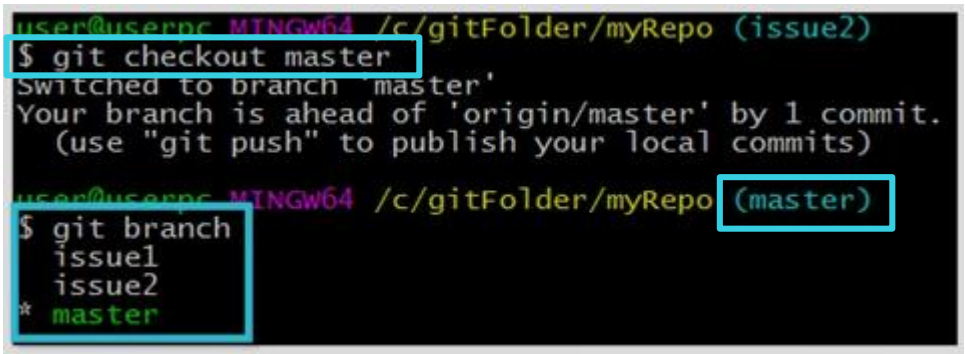
- index.html를 로컬 저장소에 등록

- issue2 branch에서 commit을 진행

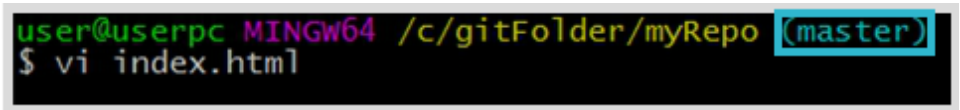
2. Git Branch 응용

4) 수정부분 master branch에서 병합

- master branch로 이동 후 branch 확인



- master branch 에서 index.html 파일 확인



2. Git Branch 응용

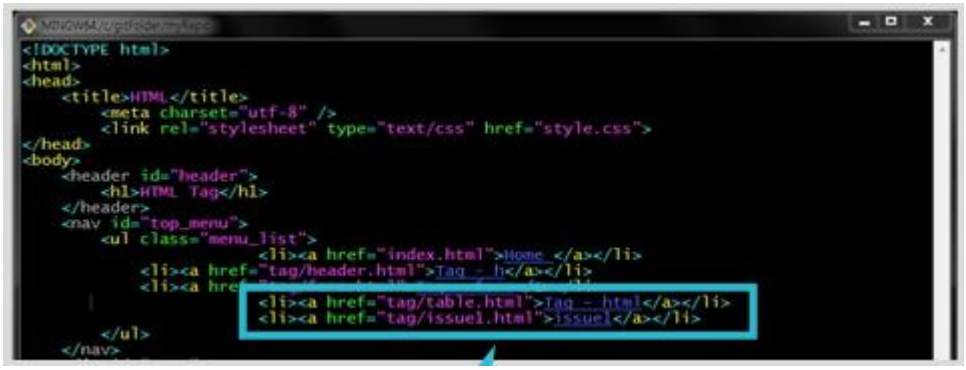
4) 수정부분 master branch에서 병합

- master branch에서 issue1 branch 병합

```
user@userpc MINGW64 /c/gitFolder/myRepo (master)
$ git merge issue1
Updating aa8d9c2..6125151
Fast-forward
 index.html | 1 +
 1 file changed, 1 insertion(+)
```

- index.html 파일이 업데이트 됨

- 병합된 index.html 파일 확인



- git log를 이용하여 commit history 조회

```
user@userpc MINGW64 /c/gitFolder/myRepo (master)
$ git log
commit 612515178b524ea3203aeb112dc9fa143c2e05c1
Author: GirrrMaster <girrr.master@gmail.com>
Date:   Wed Dec 16 16:35:07 2015 +0900

    add link (branch-issue1)

commit aa8d9c2d9288c2bc24f2b5f2ffbf818a1a714797
Author: GirrrMaster <girrr.master@gmail.com>
Date:   Wed Dec 16 15:52:24 2015 +0900

    update index.html (branch-topic1)

commit 313c2769b565173fc14d572e97f048c0ef6615b1
Author: GirrrMaster <girrr.master@gmail.com>
Date:   Fri Nov 27 12:12:44 2015 +0900

    Update index.html (github)

    Add line (github)
```

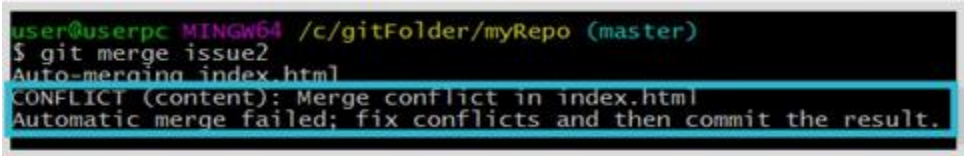
2. Git Branch 응용

4) 수정부분 master branch에서 병합

- master branch에서 issue2 branch 병합

- merge 명령어 사용

- index.html 파일 확인



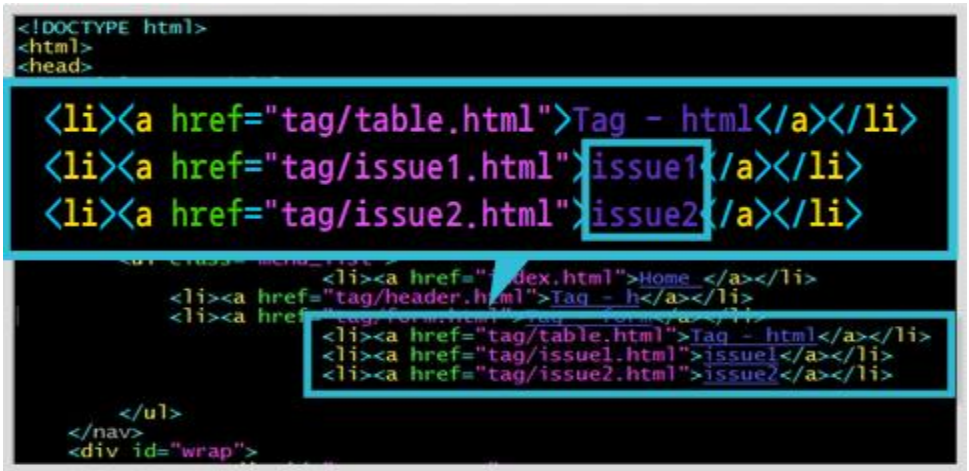
- 자동병합 실패

2. Git Branch 응용

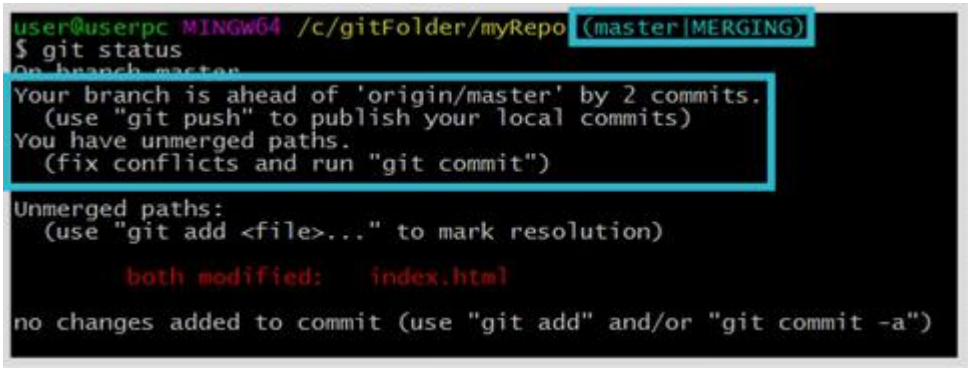
4) 수정부분 master branch에서 병합

- 병합 시 충돌이 일어나면 직접 소스코드에서 수정

직접 코드를 살펴보며 수정



- git status로 저장소 상태 확인



- master branch 보다 앞서 2개의 commit이 있었다

2. Git Branch 응용

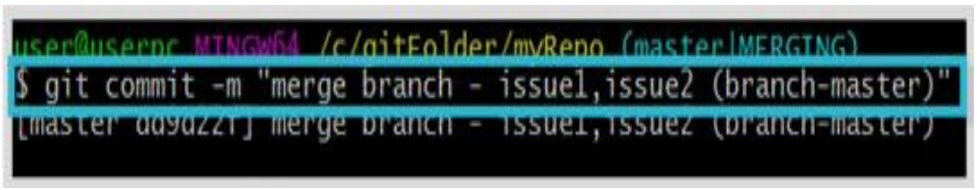
4) 수정부분 master branch에서 병합

- 모든 병합이 마친 후 git영역에 파일 추가

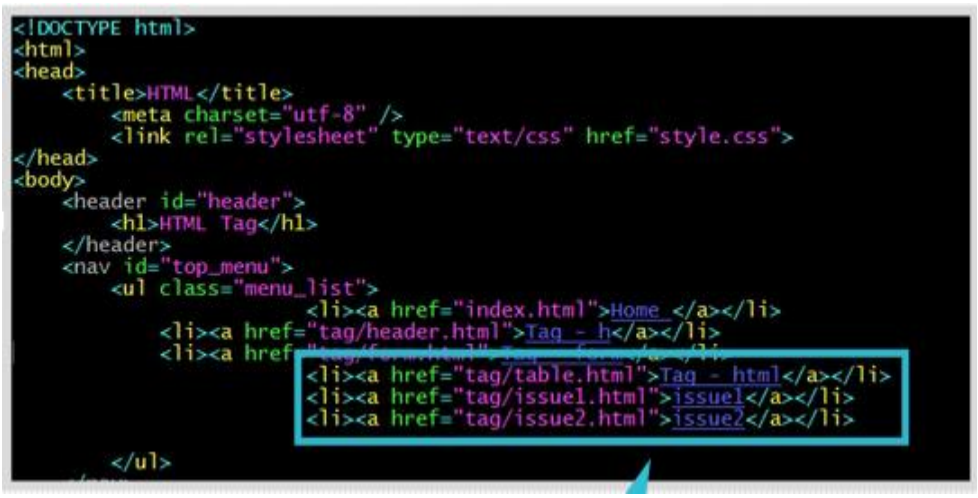
2. Git Branch 응용

5) 로컬 저장소 등록 및 확인

- 로컬 저장소에 수정된 파일을 등록 및 확인



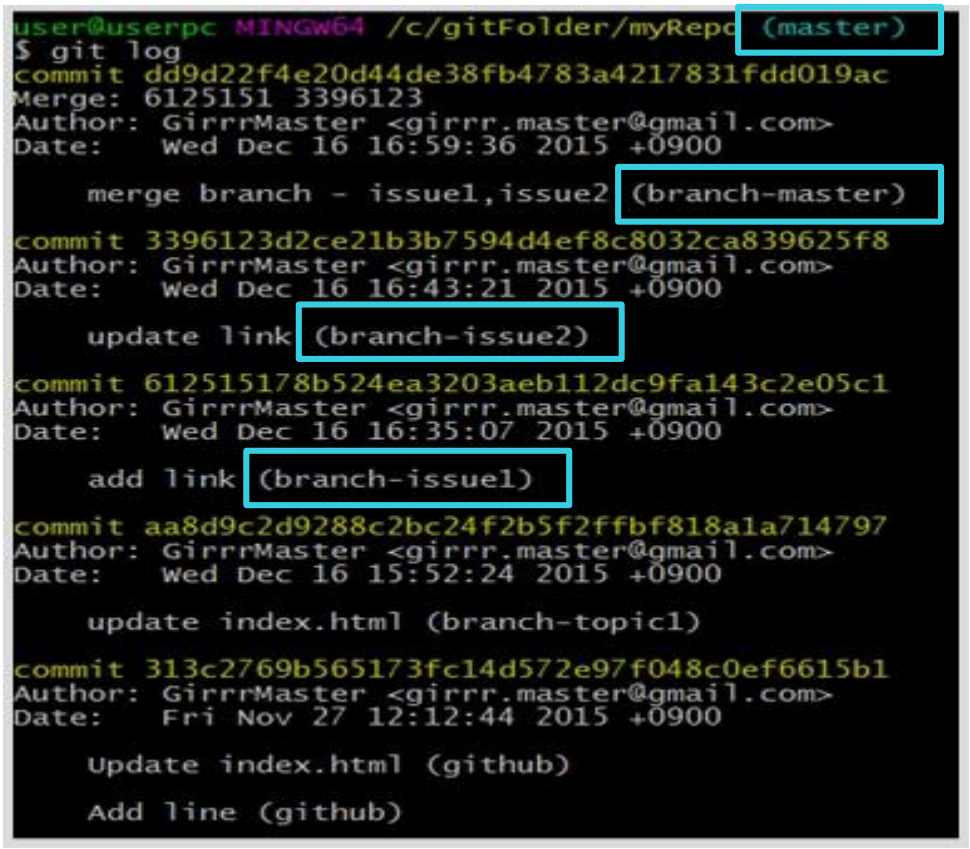
- 병합이 완료된 index.html 파일 확인



2. Git Branch 응용

5) 로컬 저장소 등록 및 확인

- git log 를 이용하여 commit history 조회



2. Git Branch 응용

5) 로컬 저장소 등록 및 확인

- index.html 파일을 인터넷브라우져에서 확인

핵심정리

Git Branch 활용

1. Git branch 기본 동작

- Branch 생성
- Branch 이동
- Branch 병합
- Branch 삭제
- Brach 의 충돌

2. Git Branch 응용

- 두 개의 branch를 생성하여 각각의 branch는 issue1, issue2 라고 명칭
- git branch 명령어를 통하여 branch를 생성
- 생성된 branch 확인

핵심정리

Git Branch 실습

1. Branch 생성

- [Create Branch] → Branch명 입력

2. 생성한 Branch에서 작업하기

- [Switch/Checkout] → Branch 이동

3. Branch의 데이터 비교

- 작업 내용 수정 → Git Commit에 메시지 입력 →수정된 내용 확인

4. Branch Merge 하기

- Master Branch로 이동 → Log 확인 → Merge에서 통합할 Branch 선택 → Merge 확인