(JIHYEONNY)의 git 정복하기

<git 이름설정>

1. **git config --global user.name ‘각자이름 영어로’**
2. **git config --global user.email ‘각자이메일’**
3. **git config --global --list (정보설정확인)**



<git 로그인정보 초기화>

1. **git credential reject**
2. **protocol=https**
3. **hosy=lab.ssafy.com**

<원격 저장소 추가>

1. **git remote add github ‘주소’**
2. **git remote –v**

<branch의 구조>

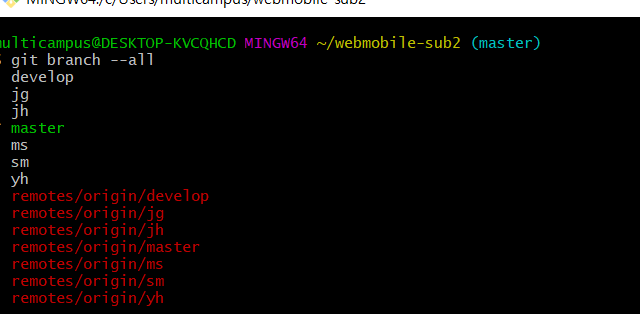
**master>develop>ms, jg, sm, yh, jh**

**이유: master는 git에 올라가는 메인, 원본이 훼손되면 복구가 힘드니까 항상 develop이라는 예비 원본을 두고 그 안에서 또 업부별로 브랜치를 생성해서 작업 후 develop에 merge 하기 위해서이다. branch를 생성할 때, 사람별로 파는 것도 좋지만 같은 파일을 건들경우 오류나 충돌이 많이난다. 그래서 업부별로 생성하는 것이 좋다.**

**또한 , npm install 하는 것 잊지말기! (누군가가 무엇을 설치했을 경우)**

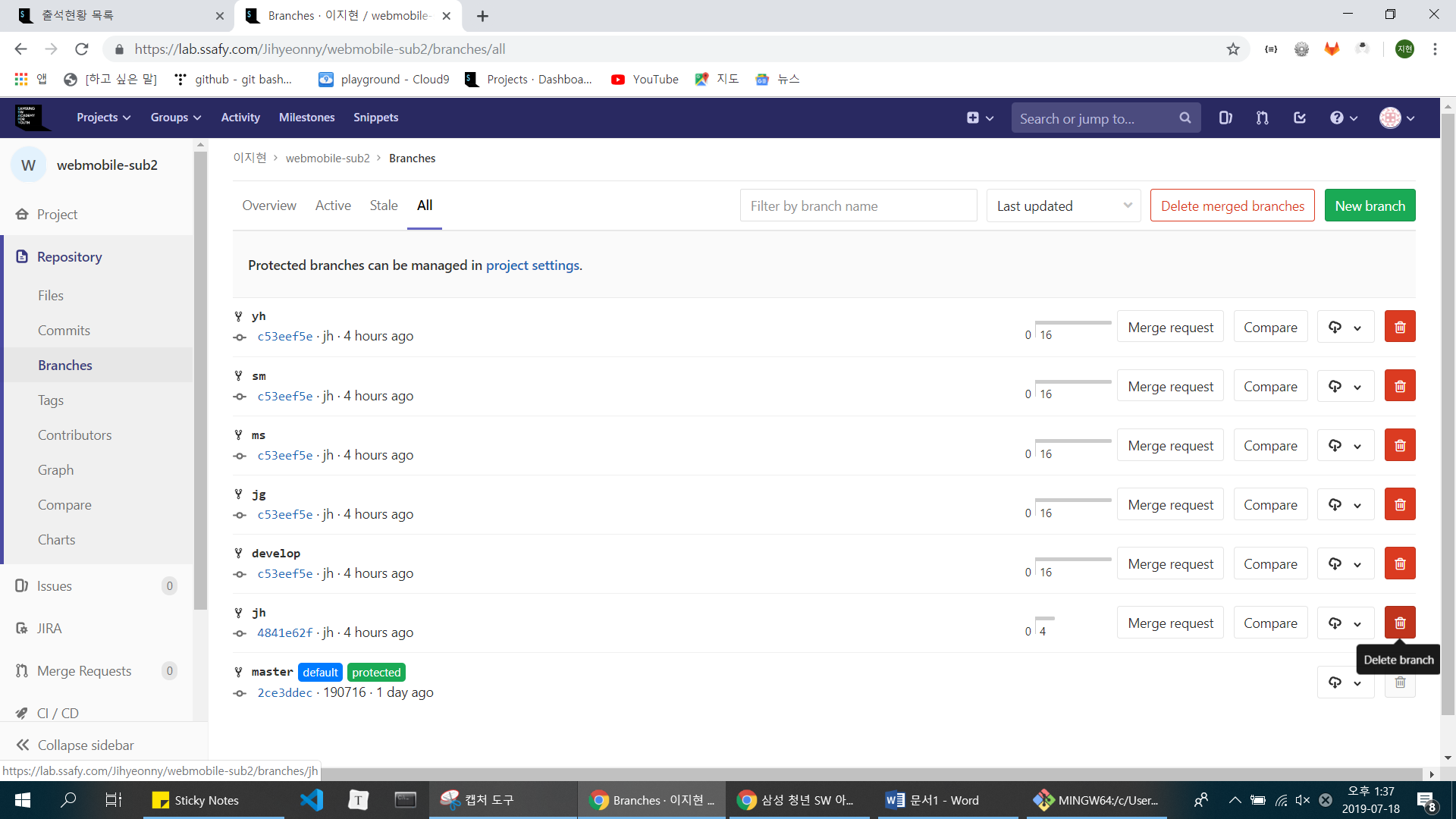
**<git branch 생성하기>**

1. **git branch develop**
2. **git checkout develop**
3. **git branch ms**
4. **git checkout develop**
5. **git branch jg**
6. **git checkout develop**



**<branch 삭제하는 법>**

1. lab.ssafy >Repository > Branches 에서 지울 브랜치 삭제



1. gitbash에서 삭제

만약, jh 를 삭제하고싶으면 ( 그 상위 브랜치에서 삭제)

**git checkout develop**

**git branch –d jh (강제 삭제 시, git branch –D jh)**

**<develop에 하위 브랜치 merge 하기1>**

1. **cd [폴더명]**

이때, 현재 브랜치 = master

1. **git checkout [develop]**

이때, 현재 브랜치 = develop

1. **git checkout [develop 하위 브랜치(각자브랜치)]**

**git pull origin [develop 하위 브랜치(각자브랜치)**]

모든 develop 하위 브랜치(각자브랜치)를 왔다갔다 거리면서 내려받는다.

이때, develop 하위 브랜치(각자브랜치)위치인지 확인!!!!!

ex) jh을 내려받고 yh을 내려받을땐?

**git checkout jh**

**git pull origin jh**

**git checkout yh**

**git pull origin yh**

이런식으로 각자브랜치에 들어가서 pull

1. **git checkout develop**
2. **git merge [develop 하위 브랜치(각자브랜치)]**

하나하나 다 해준다.

하면서 오류 메세지 주의

이때, CONFLICT(content) 메시지가 뜨면 충돌이 일어나서 자동병합실패

ex) CONFLICT (content): Merge conflict in src/store.js

src/store.js 파일 들어가서 충돌된 부분수정 후,

(develop|MERGING) 이라고 되어있을거야!

이때, 바로 파일을 열어서 충돌확인하고 (충돌이 안일어났을수도있음)

git add .

git commit –m “merging| **develop 하위 브랜치(각자브랜치)”**

**git push origin develop**

**이렇게 하면 lab.ssafy.com > repo> commit 에서 만약 다음 작업에서 오류나도 쉽게 볼 수 있음**

1. git checkout [**develop 하위 브랜치(각자브랜치)]**

git merge develop

두가지 경우가 있다. 둘중 하나의 경우에 따라 병합하면된다.

첫번째 경우, CONFLICT(content) 메시지가 뜨면 충돌이 일어나서 자동병합실패

ex) CONFLICT (content): Merge conflict in src/store.js

src/store.js 파일 들어가서 충돌된 부분수정 후,

(jh|MERGING) 이라고 되어있을거야!

이때, 바로 파일을 열어서 충돌확인하고 (충돌이 안일어났을수도있음)

**git add .**

**git commit –m “[develop 하위 브랜치(각자브랜치)]”**

**git push origin [develop 하위 브랜치(각자브랜치)]**

두번째 경우, 에러가 뜨지 않고 +++이렇게 파일 마다 다 수정된 정도가 표시된 파일들이 나열되어있으면

**git add .**

**git commit –m “**[**develop 하위 브랜치(각자브랜치)]”**

**git push origin** [**develop 하위 브랜치(각자브랜치)]**

**<develop에 하위 브랜치 merge 하기2>**

1. **cd [폴더명]**

이때, 현재 브랜치 = master

1. **git checkout [develop]**

이때, 현재 브랜치 = develop

1. **git pull origin [develop 하위 브랜치(각자브랜치)]**

두가지 경우가 있다. 둘중 하나의 경우에 따라 병합하면된다.

첫번째 경우, CONFLICT(content) 메시지가 뜨면 충돌이 일어나서 자동병합실패

ex) CONFLICT (content): Merge conflict in src/store.js

src/store.js 파일 들어가서 충돌된 부분수정 후,

(develop|MERGING) 이라고 되어있을거야!

이때, 바로 파일을 열어서 충돌확인하고 (충돌이 안일어났을수도있음)

**git add .**

**git commit –m “merging|[develop 하위 브랜치(각자브랜치)]”**

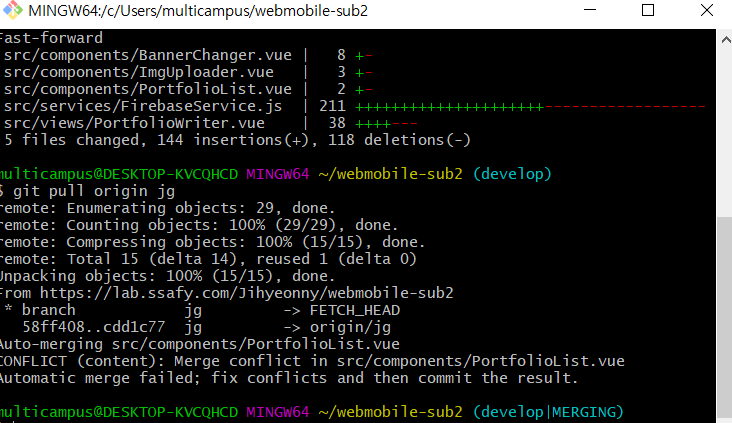
**git push origin develop (merge하면서 오류나 전 파일 확인을 위해 push 해두고 lab.ssafy.com에서 commit확인하기위해)**

두번째 경우, 에러가 뜨지 않고 +++이렇게 파일 마다 다 수정된 정도가 표시된 파일들이 나열되어있으면

**git add .**

**git commit –m “merging|[develop 하위 브랜치(각자브랜치)]”**

**git push origin develop (merge하면서 오류나 전 파일 확인을 위해 push 해두고 lab.ssafy.com에서 commit확인하기위해)**



1. **git merge [develop하위브랜치(각자브랜치)]**

아마, Already up to date 라고 뜰거야. 이미 develop에서 하위 브랜치들을 pull받았기때문충돌난 하위브랜치말고는 자동병합되었을거야.

1. **git checkout [develop 하위브랜치(각자브랜치)]**

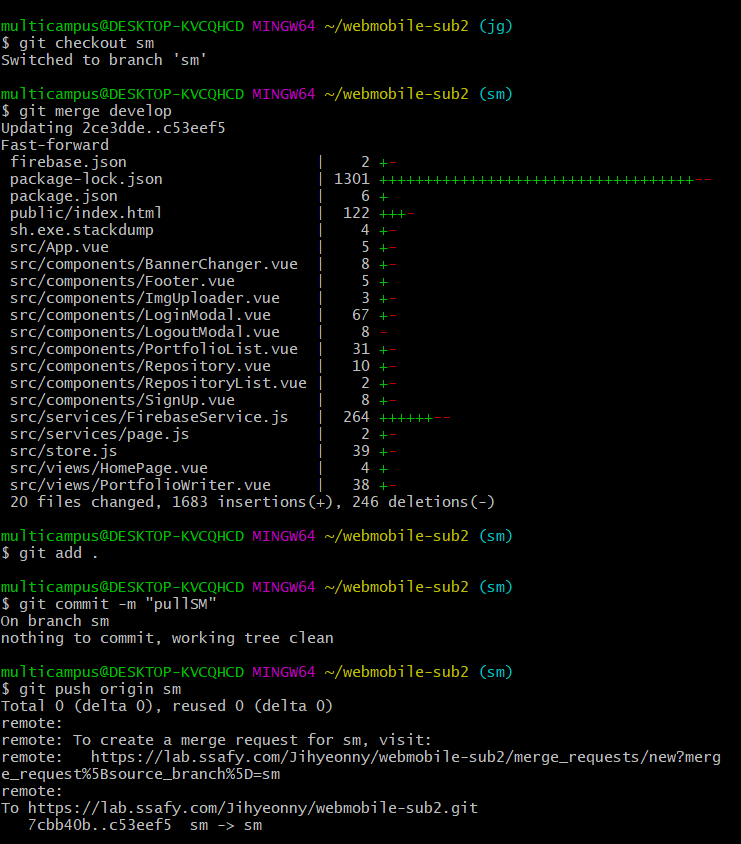
**git pull origin [develop하위브랜치(각자브랜치)]**

**git merge develop**

**git add .**

**git commit –m “메시지”**

**git push origin [develop 하위브랜치(각자브랜치)]**



1. 이제 조원들이 해야 할일 (무조건 이 작업 후에 작업시작)

**git checkout [develop 하위브랜치]**

**git pull origin [develop 하위브랜치]**

**이때, pull 안된다?**

* **git add .**
* **git commit –m “메시지”**

**까지만 진행하고 다시**

* **git pull origin [develop 하위브랜치]**