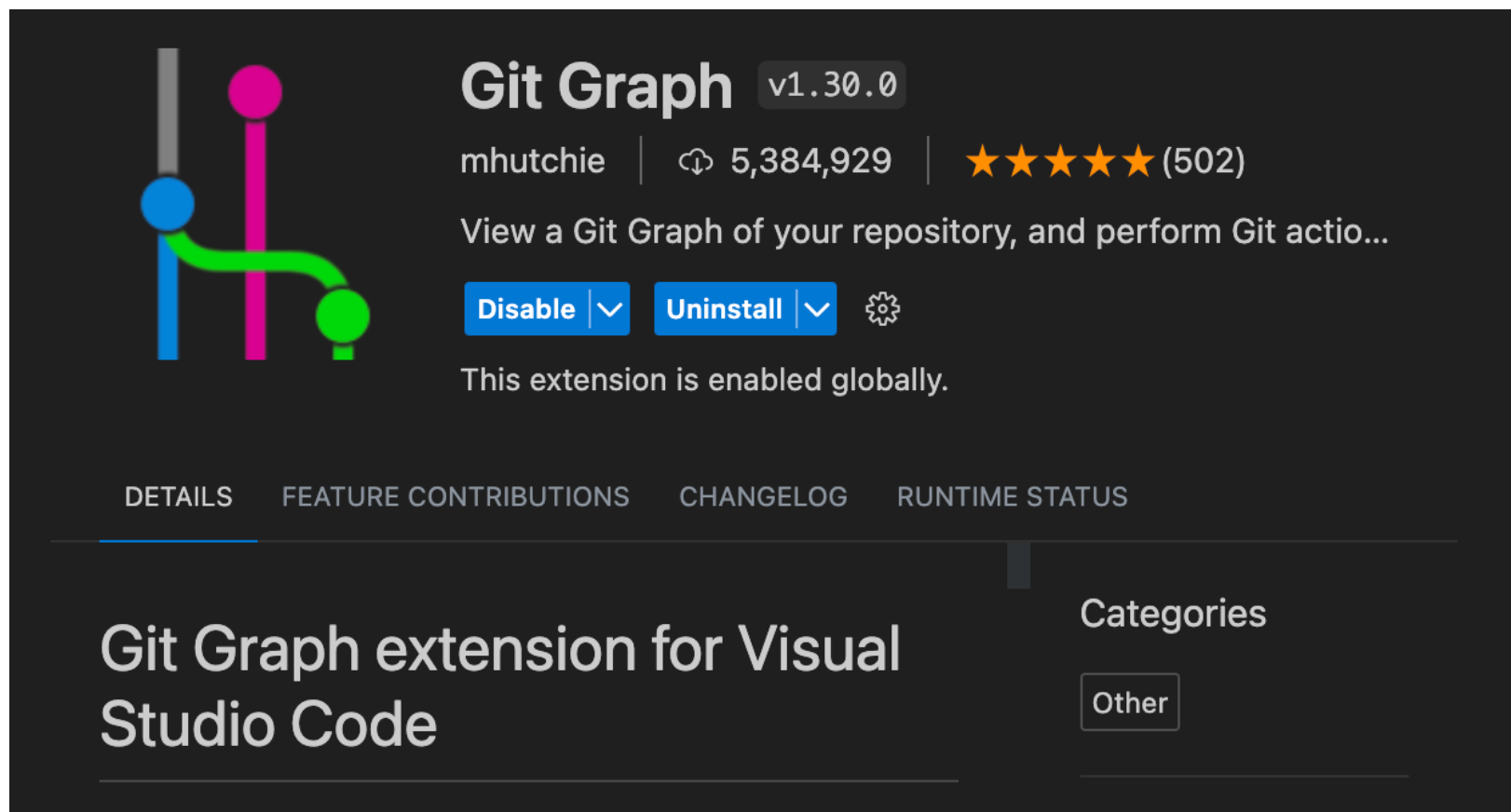


좋은 말씀 전합니다.



Git Graph v1.30.0

mhutchie | 5,384,929 | ★★★★★ (502)

View a Git Graph of your repository, and perform Git actio...

[Disable](#) | [Uninstall](#) | ⚙️

This extension is enabled globally.

[DETAILS](#) | [FEATURE CONTRIBUTIONS](#) | [CHANGELOG](#) | [RUNTIME STATUS](#)

Git Graph extension for Visual Studio Code

Categories

[Other](#)

어머! 저건 붙여야해

- TAG
 - 기념할 만한 무엇인가에 붙일 수 있는
깃의 배려



깃 허브에서 붙여 볼까요?

1. 커밋 히스토리를 봅니다.
2. 메인 커밋 바로 뒤 시점의 저장소를 browse 해 봅니다.
3. Create a new release
4. Choose a tag에서 Find or create a new tag 입력 후 create new tag
5. Publish release
→그러면 뭔가 좀 과한 일이 일어납니다.
6. 이 녀석을 pull해서, 태그를 선택하고 수정 하고 난 후에 커밋을 하면...

깃 허브에서 붙여 볼까요?

7. 뭔가 부모 없는 자식이 생겨 버렸습니다. 이 녀석을 어떻게 해보죠.

8. vs에서... 임시 브랜치를 선택하면 위에서 브랜치를 만들라고 합니다.

9. 하나 만들었어요.

10. 이 시점에서 git graph를 볼까요?

결론. tag는 브랜치와 다릅니다. 우리는 사실 하나의 선형 커밋을 하고 있었던 겁니다.

조금 더 가 봅시다.

- 11. 선형 커밋이긴 한데, 메인이 가장 최신 브랜치는 아닙니다.
- 12. 이때 병합을 하면 되지요.

요약하자면, 태그에서 커밋하면 태그를 만든 브랜치의 어떤 선상에 갈 곳 잃은 브랜치 상태가 됩니다.

(참고로, 이 상황은 tag를 checkout한 후에 하는 커밋에 해당됩니다.)

이 상태에서 브랜치를 만들고, 마스터에서 병합을 하면 적당한 답이 나옵니다.

이번엔 vs code에서 붙여 보지요.

Source Control에서 ... 에서 Tags > Create Tags (v1.1.0)
과거 커밋에 붙이고 싶을 땐 extention을 이용합니다.

이제 이렇게 태그를 붙였으면, 리모트에서도 공유를 하고
싶겠지요.

그런데, sync 알림도 없고, 그렇다고 깃헙에 가서 봐도 태그가
없습니다. 태그는 push가 되지 않는 녀석이었던 겁니다.

이번에도 extension을 이용 합니다.

Console에서 붙이려면

- `git tag -a tagname -m "message"` (anotated tag)
- `git tag tagname` (lightweight tag)
- 과거 버전에 붙이려면
- `git tag tagname commitid`
- 일단, (push하기 전인) 가장 최신 버전에 v1.1.1 태그를 붙여 봅니다. (lw)
- 그리고 1.2.0 1.3.0, ... 마구 붙입니다. 하나의 커밋에 여러 개의 태그가 붙을 수 있다는 사실을 알게 됩니다.
- 마지막으로, 로컬에서 tag를 push 하려면, `git push 저장소이름 tagname` 또는 `git push 저장소이름 --tags` (모두)

먹다 남아 버렸는데, 이걸 어떻게...

- stash
 - 일단 저장하고 다른 브랜치로 넘어 갑니다.
 - 무엇을?
 - Modified이면서 tracked인 대상과
 - Staging 된 대상을



설마

- 우리 문화인들은 github 웹에서 stash 어떻게 하냐고 묻지 맙시다.
- untracked 파일은 어떻게 되냐고도 묻지 맙시다.
 - 사실은 -u 옵션을 붙이면 가능해요
- 그리고 stash는 브랜치랑 무관한 것도 아시지요?

먹다 남은 것을 킵 해 둡시다. git stash

- 어느 브랜치건 파일 하나 추가한 후
- git stash
- 브랜치를 바꾸고난 후에
- git stash apply
- 다시 git stash
- 다시 파일 생성, git add .
- 다시 git stash
- stash 목록을 보려면? git stash list
 - 어라 apply를 했는데도, 여전히 목록이 남아 있네.

먹다 남은 것을 킵 해 둡시다. git stash

- 어느 브랜치건 파일 하나 추가한 후
- git stash
- 브랜치를 바꾸고난 후에
- git stash apply
- 다시 git stash
- 다시 파일 생성, git add .
- 다시 git stash,
- stash 목록을 보려면? git stash list
 - 어라 apply를 했는데도, 여전히 목록이 남아 있네.

먹다 남은 것을 킵 해 둡시다. git stash

- 어느 브랜치건 파일 하나 추가한 후
- git stash
- 브랜치를 바꾸고난 후에
- git stash apply
- 다시 git stash
- 다시 파일 생성, git add .
- 다시 git stash,
- stash 목록을 보려면? git stash list
 - 어라 apply를 했는데도, 여전히 목록이 남아 있네.

먹다 남은 것을 킵 해 둡시다. git stash

- 그래서 목록을 지워줘야 합니다.
 - git stash drop
 - git stash pop (적용하면서 삭제)
- 그리고 선택적으로 stash를 하고 싶을 때는
 - git stash -patch
- shash가 있을 때
 - git stash branch branchname → 새 브랜치를 만들고 거기에 stash apply를 하고, stash 목록을 삭제
- vs code에서는 source control에서 추적 창에서 선택적으로 stash 가능

종종 보는 에러인데...

- 나는 그저 push를 하려고 pull을 했을 뿐인데...

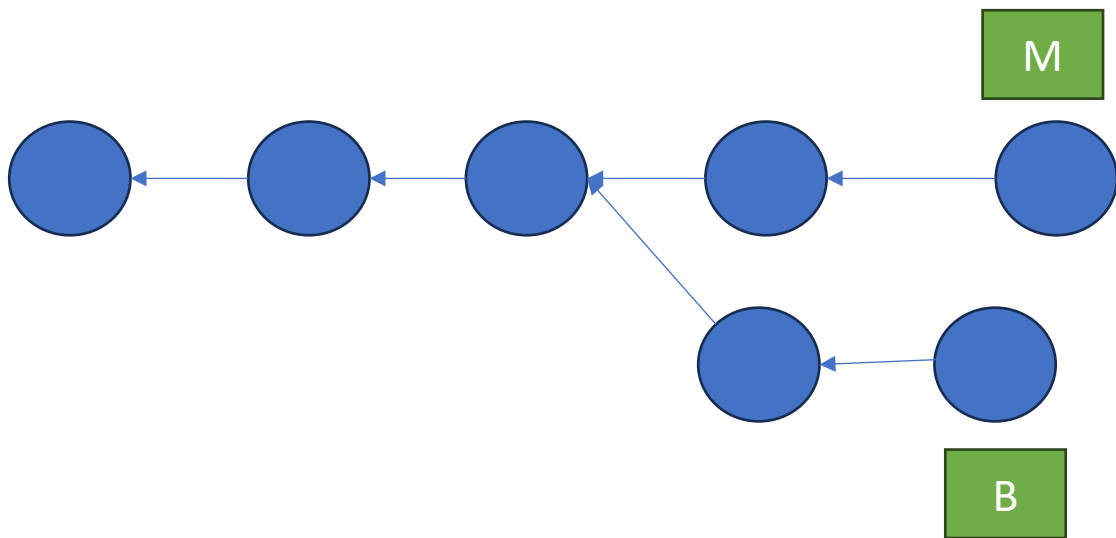
```
git config pull.rebase false # merge
git config pull.rebase true  # rebase
git config pull.ff only      # fast-forward only
```

- 그럼 rebase가 뭔지 알아야 겠습니다.

리베이스

- 시나리오

- master에서 브랜치를 만들고, 마스터에서 수정을 하고 커밋을 했음
- 그 후에 브랜치에서 수정 커밋 (모든게 충돌이 없도록)



b1	b_2	11 Jul 2023...	crapas	05e07f27
b_1		11 Jul 2023...	crapas	9323c303
master	m_2	11 Jul 2023...	crapas	3b63bb7b
m_1		11 Jul 2023...	crapas	f10c8ec9
5th		11 Jul 2023...	crapas	1fcdcf6d
6th		11 Jul 2023...	crapas	56995955

리베이스

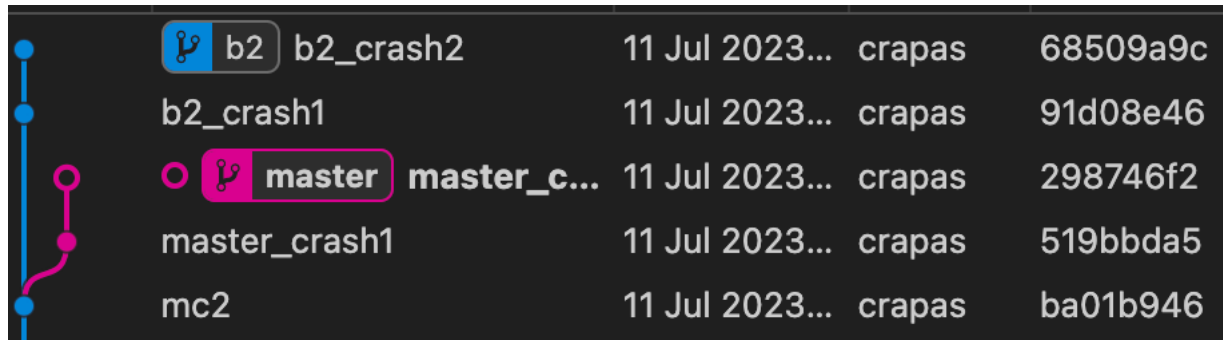
- 패치!패치!
 - (in master) git rebase b1




















































































































































































































































































































































































































































병합의 경우



충돌이 날 때 경우



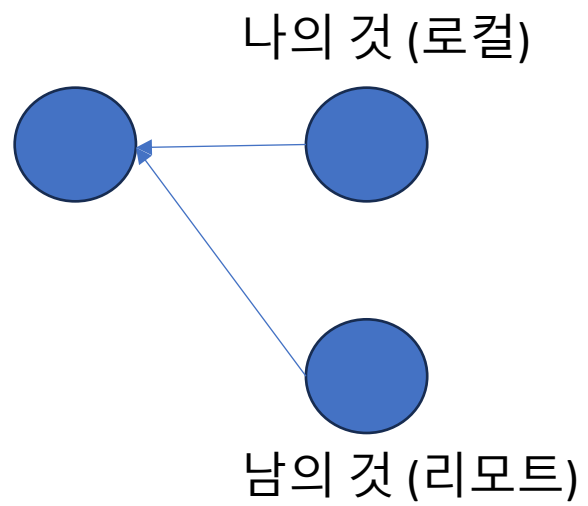
- master의 각각의 commit에 대해서 브랜치의 최종 커밋의 '패치'를 적용



○	master	master_crash2	11 Jul 2023...	crapas	e0c8190d
		master_crash1	11 Jul 2023...	crapas	cf1f0a2e
	b2	b2_crash2	11 Jul 2023...	crapas	68509a9c
		b2_crash1	11 Jul 2023...	crapas	91d08e46
		mc2	11 Jul 2023...	crapas	ba01b946

revisit

- git pull : git fetch + git merge
- 언제 발생하는가?
 - fast-forward merge가 안되는 경우
 - 이 경우에 merge냐 rebase냐의 전략을 옵션으로 설정해줘야 함
- 어떻게 해결할까?
 - merge로 선택
 - rebase로 선택
 - ff-only를 유지하고 안되면 fetch 후 확인하고 필요에 따라 병합
 - 안전하고
 - 리뷰를 거침



기타

- remote에서 만들어진 branch 정보는 git fetch를 한 다음에 알 수 있습니다.