Git, week-1

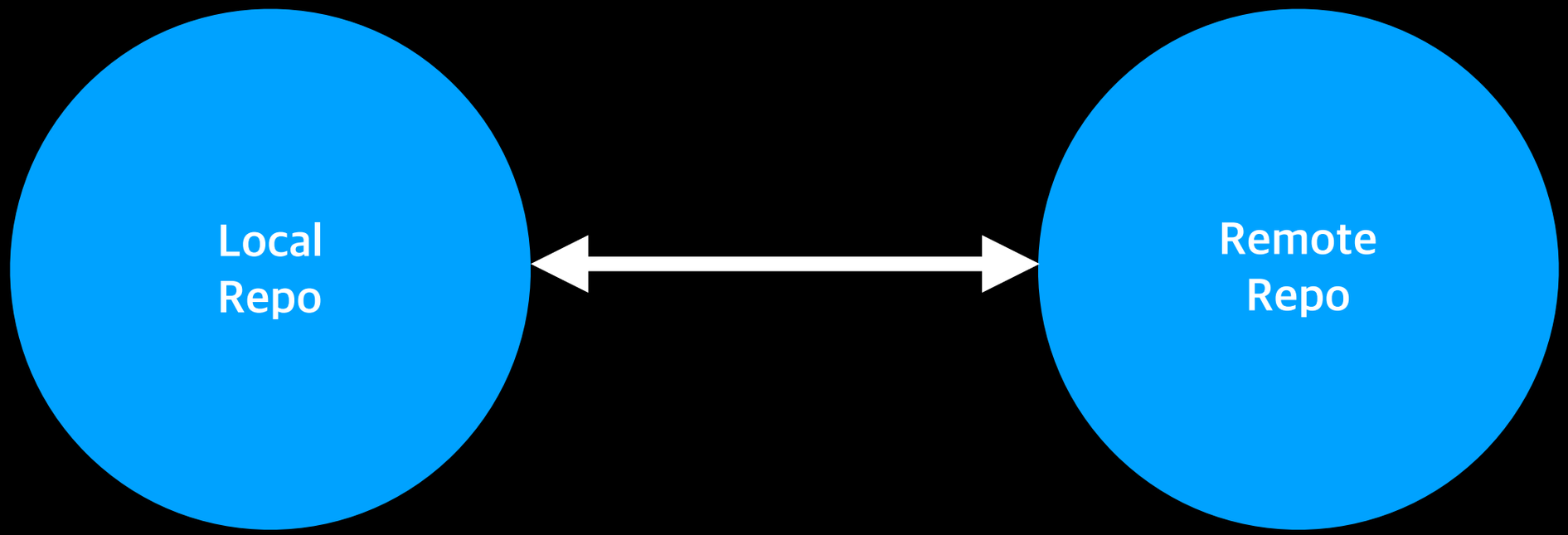
VCS

(Version Control System, 버전 컨트롤 시스템, 형상 관리 시스템)

- Local
- 중앙 집중형 (CVS) - SVN
- 분산형 (DVCS) - GIT

VCS

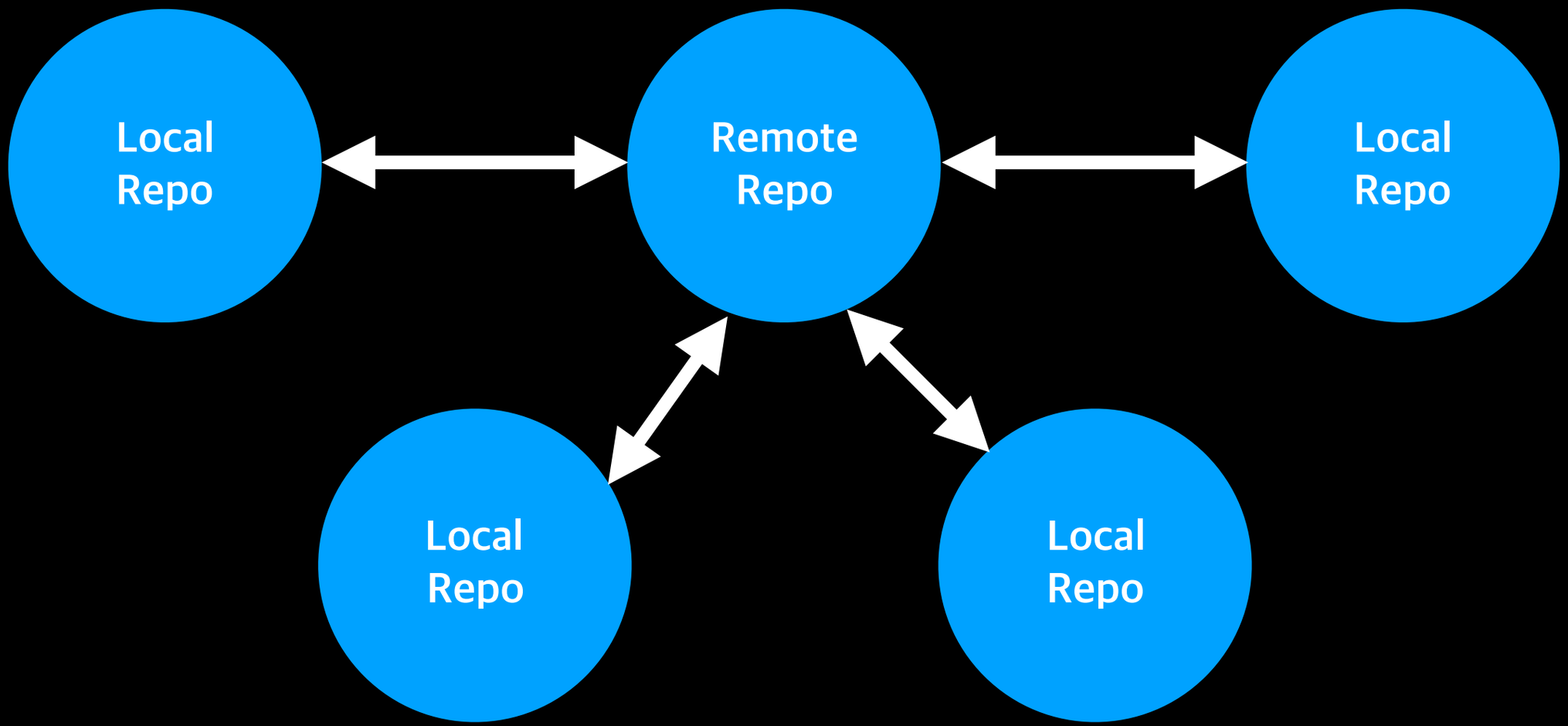
(Version Control System, 버전 컨트롤 시스템, 형상 관리 시스템)



VCS

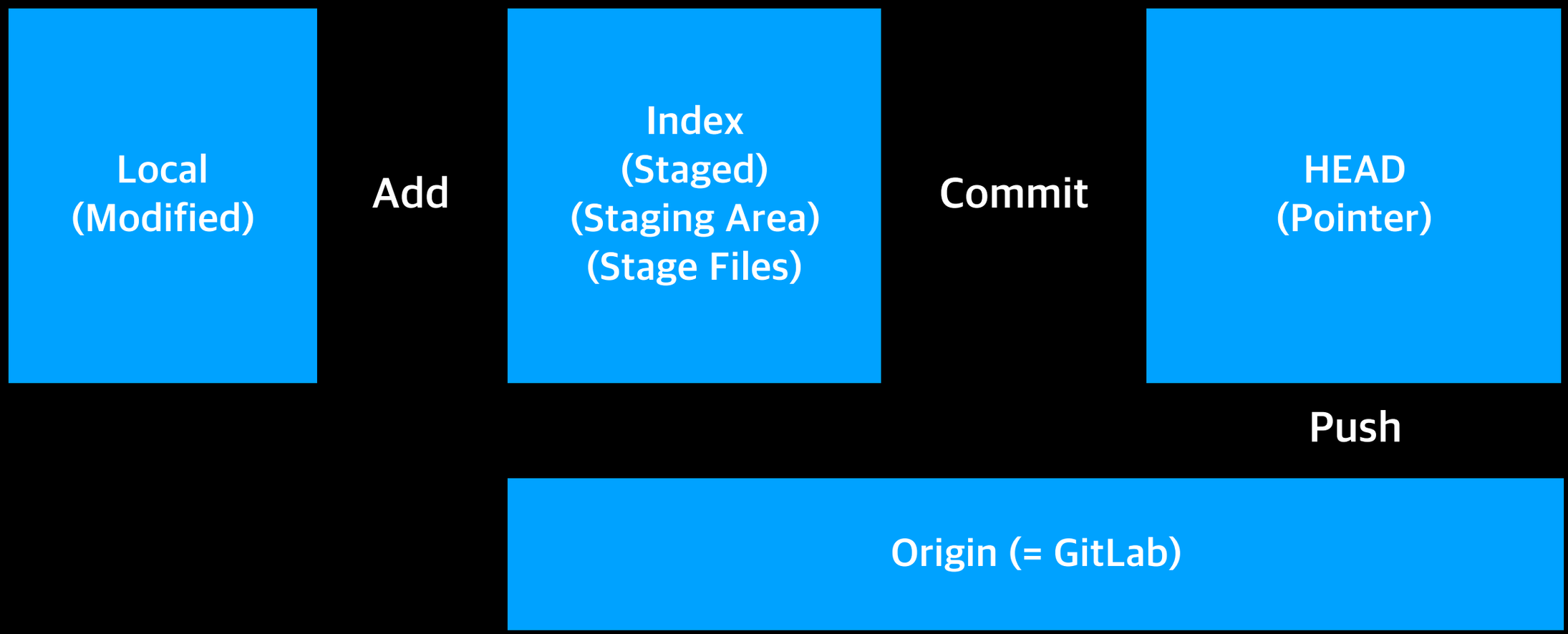
(Version Control System, 버전 컨트롤 시스템, 형상 관리 시스템)

자유로운 Add, Commit, Merge, Branch 관리



VCS

(Version Control System, 버전 컨트롤 시스템, 형상 관리 시스템)



Commit

- **GIT = Product History Manage, 제품 역사 관리 도구**
  - 데이터를 변경 사항으로 기록하지 않고 일련의 스냅샷으로 기록한다.

- **Commit**
  - Modified => Staged 상태에 있는 변경 내용들을 Repository 에 저장하는 것
  - 유저에게 배포하기, 기능 제공 이전의 가장 최소 단위.

- **Commit 의 단위**
  - 될수록, 될수록, 될수록 작은 단위
  - 최소한의 의미를 가지고 있는 단위 (= Atomic 단위)

Branch

- **GIT = Product History Manage, 제품 역사 관리 도구**
  - 데이터를 변경 사항으로 기록하지 않고 일련의 스냅샷으로 기록한다.
- **Branch**
  - 특정 커밋을 가르키는 정보 객체, 포인터
- **Branch 의 단위**
  - 작업 단위
  - 기능 추가? 버그 수정? 제품 배포?

Git* flow

- Git flow, Github flow, Gitlab flow 등, 많은 도구
- 위 많은 flow 는 결국 도구에 지나지 않는다.
- 주객전도가 되지 않도록, 올바르게 사용하기!

‘Checkout’ / ‘Fetch’

- **git checkout**
  - 전체 파일 초기화
- **git checkout -- {filename}**
  - 특정 ‘{filename}’ 파일 초기화
- **git fetch**
  - Remote 상태 업데이트 (git pull = git fetch + git merge)

‘Revert’ / ‘Reset’

- **Reset** (–mixed (default), –soft, –hard)
  - 완전 초기화, 로그 X
  - Branch 의 Commit / HEAD 이동
  - Stage 와 Commit 동기화
  - Working Dir 와 Commit 동기화
- **Revert**
  - 대충 초기화, 로그 O

‘Merge’ / ‘Rebase’

- Merge
- Rebase

.gitignore

```
576B  1  1 22:54 .git
111B 12 13 13:04 .gitignore
```

```
vendor/
node_modules/
npm-debug.log

manage/extraModule/*.txt

assets/file/cont_db.php
#assets/img/of_partner/
```

프로젝트에 필요 없는 백업, 개발 로그, 보안, 기타 파일들을 Git 에서 제외 시킬 수 있는 설정

https://github.com/github/gitignore

Stash

- 1. “특정 프로젝트에서 한 부분을 담당하고 있습니다. 그리고 여기에서 뭔가 작업하던 일이 있고 다른 요청이 들어와서 잠시 브랜치를 변경해야 할 일이 생겼습니다. 아직 완료하지 않은 일을 커밋하는 것은 좀 껄끄럽다. 이런 상황에서는 커밋하지 않고 나중에 다시 돌아와서 작업을 다시 하고 싶을 것 같습니다. 으악!”
- 2. “개발 브랜치에서 개발해야 되는데 마스터 브랜치에 열심히 개발해 버렸네요, 마스터 브랜치에 작업한걸 개발 브랜치로 넘겨야 됩니다. 으악!”
- 3. “프로젝트를 열심히 마치고 배포가 되었습니다. 추가 기능을 열심히 개발을 다시 시작합니다. 그런데 이게 왠 걸? 개발한 코드에 문제가 있어 장애가 발생했습니다. 땀이 줄줄 가슴은 철렁, 빨리 수정을 해야 합니다. 그런데 지금까지 개발한 코드를 버리긴 아깝네요. 얼른 워킹 디렉토리를 HEAD로 돌려야 하는데.. 으악!”

“Modified 이면서 Tracked 상태인 파일과 Staging Area에 있는 파일”

- git stash
- git stash list / drop / clear
- git stash pop / apply (git stash apply —index)

Git Hook

```
→  hooks git:(hotfix-tutorial_bug) pwd
/Users/teddy/Desktop/TEDDY/PROJECT/TROST/TROST/.git/hooks
→  hooks git:(hotfix-tutorial_bug) ll
total 80
-rwxr-xr-x  1 teddy  staff   478B 12  9 15:50 applypatch-msg.sample
-rwxr-xr-x  1 teddy  staff   896B 12  9 15:50 commit-msg.sample
-rwxr-xr-x  1 teddy  staff   189B 12  9 15:50 post-update.sample
-rwxr-xr-x  1 teddy  staff   424B 12  9 15:50 pre-applypatch.sample
-rwxr-xr-x  1 teddy  staff   1.6K 12  9 15:50 pre-commit.sample
-rwxr-xr-x  1 teddy  staff   1.3K 12  9 15:50 pre-push.sample
-rwxr-xr-x  1 teddy  staff   4.8K 12  9 15:50 pre-rebase.sample
-rwxr-xr-x  1 teddy  staff   1.2K 12  9 15:50 prepare-commit-msg.sample
-rwxr-xr-x  1 teddy  staff   3.5K 12  9 15:50 update.sample
```

특정 이벤트 전/후에 미리 만들어둔 스크립트 실행을 도와주는 Hook (Bash-Shell)

e.g.)
Git-Hooks (http://githooks.com/)
Git-Lint (https://github.com/sk-/git-lint)
Git-Swift-Lint (https://github.com/realm/SwiftLint/blob/master/README_KR.md)
Kotlin-Lint (https://github.com/shyiko/ktlint)

배포 자동화 ?

별거 없답니다..

(a) 작업 내용 업데이트 (Master Push)

(b) 스타일 가이드 (Style Guide Lint) 체크

(c) 테스트 코드 테스트 진행 (Unit, UI …)

(f) Apk 파일 추출

…

…

…

(d) QA 작업 진행 > 신규 이슈 찾았!

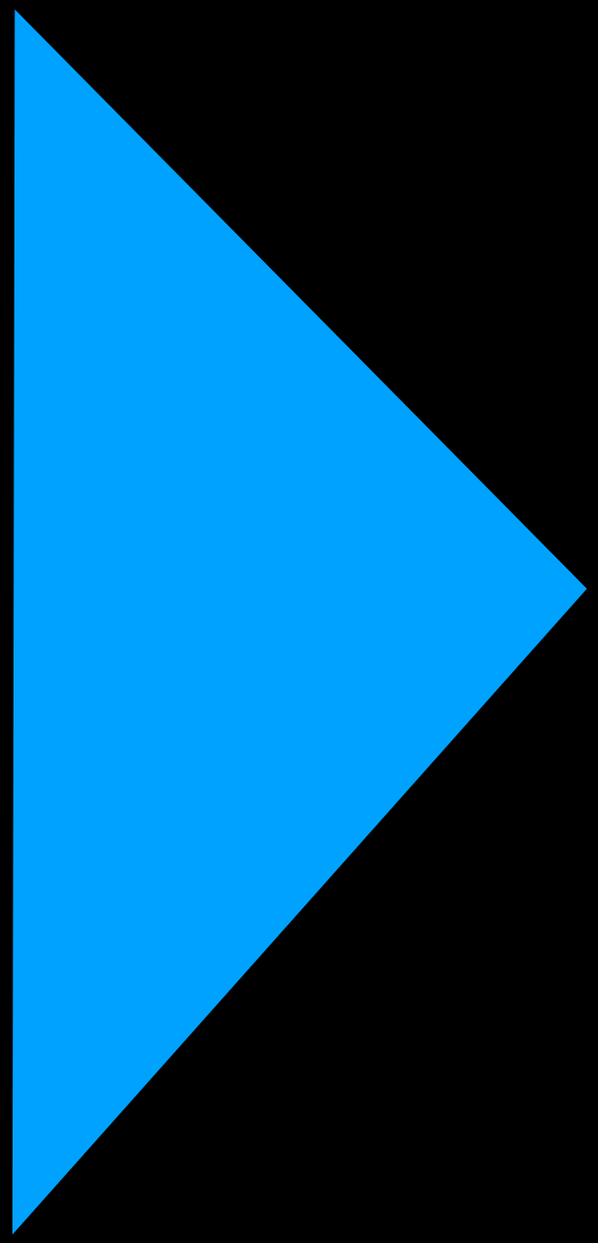
(e) 이슈 해결 내용 업데이트 (Hotfix Push) > 자동 체크

…

(b) > (c) > (f) 재실행

…

No Problem > Apk 업데이트 !
(가장 Simple, Normal 버전)



Master Push 후 자동화

License 관리

I want it simple and permissive.

The **MIT License** is a permissive license that is short and to the point. It lets people do anything they want with your code as long as they provide attribution back to you and don't hold you liable.

Babel, **.NET Core**, and **Rails** use the MIT License.

I'm concerned about patents.

The **Apache License 2.0** is a permissive license similar to the MIT License, but also provides an express grant of patent rights from contributors to users.

Elasticsearch, **Kubernetes**, and **Swift** use the Apache License 2.0.

I care about sharing improvements.

The **GNU GPLv3** is a copyleft license that requires anyone who distributes your code or a derivative work to make the source available under the same terms, and also provides an express grant of patent rights from contributors to users.

Ansible, **Bash**, and **GIMP** use the GNU GPLv3.

Usage

- “A 마무리 하고, B 하다가 망쳤다! ㅅㅂ!”
- “A 마무리 하고, B 도 개발 마무리 했고, C 개발 하고 있는데 A 까지만 업데이트를 하재! ㅅㅂ!”
- “gitIgnore 다시 관리하고 싶은데 이미 푸시를 해버렸어! ㅅㅂ”
- “A 마무리 하고 푸시 하려는데 풀 받아가라네! ㅅㅂ!”
- “몽땅 모든 것이 Conflict 나버렸어! ㅅㅂ!”

Next Week

- Merge, Rebase, Reset, Revert 실습!
- Usage 분석!
- Conflict 와 친해지기!
- GUI 에서는 넘나 쉬웠던 작업을 터미널에서!
- 팀 내 Commit, Branch 전략 결정 > 코드 리뷰 준비!
- 4주 정도 꼼꼼하게 다 파내보면 끝나겠죠?

HumartCompany

김태욱