Git초 remote

작성자 : 이준호
작성일 : 2022. 08. 02
수정일 : 2023. 01. 08
버전: 0.5

TOTORo

✉ totOroprog@gmail.com
https://totOrokr.github.io
https://github.com/TOTORoKR

목차

Local Repo 명령어

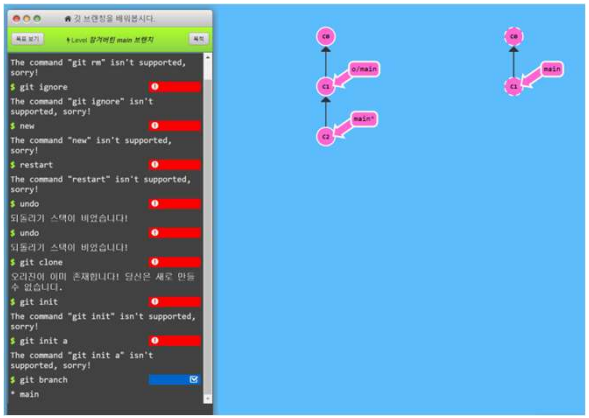
목차

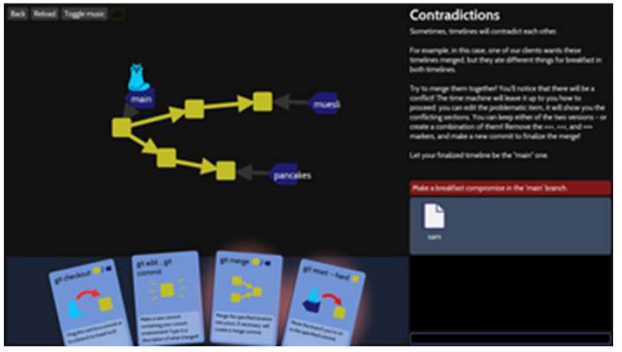
Remote Repo 명령어

목차

기타 명령어

- git tag
- git config
- git log
- git diff
- patch
- git stash
- git reflog
- git clean





Git 연습 사이트

- https://learngitbranching.js.org/?locale=ko: 트레이닝 사이트.
- https://blinry.itch.io/oh-my-git: 게임

목차

Advanced 명령어

Remote Repo 명령어

TOTORo

totOroprog@gmail.com
https://totOrokr.github.io
https://github.com/TOTORoKR

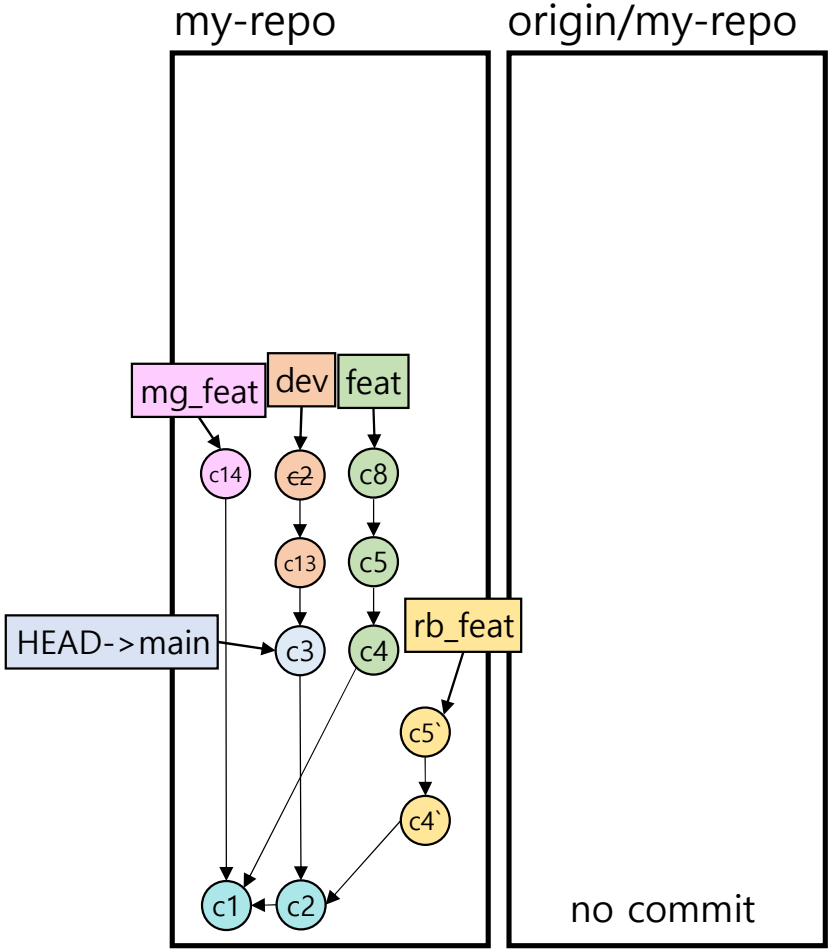
git remote

■Remote repository와 연결

- $ git remote add <리모트> <URL>
  - Remote repo를 <리모트>로 추가
- $ git remote remove <리모트>
  - <리모트>를 연결 해제
- $ git remote rename <SRC> <DST>
  - <SRC>에서 <DST>로 이름 변경
- $ git remote -v
  - Remote repo 정보를 보여줌

■예제

- (main) $ git remote add origin https://github.com/아이디/my-repo.git



git push

■Local commit을 Remote repository에 업로드

- $ git push <리모트> <브랜치>
  - <브랜치>를 <리모트>에 업로드

■Remote branch에 업로드

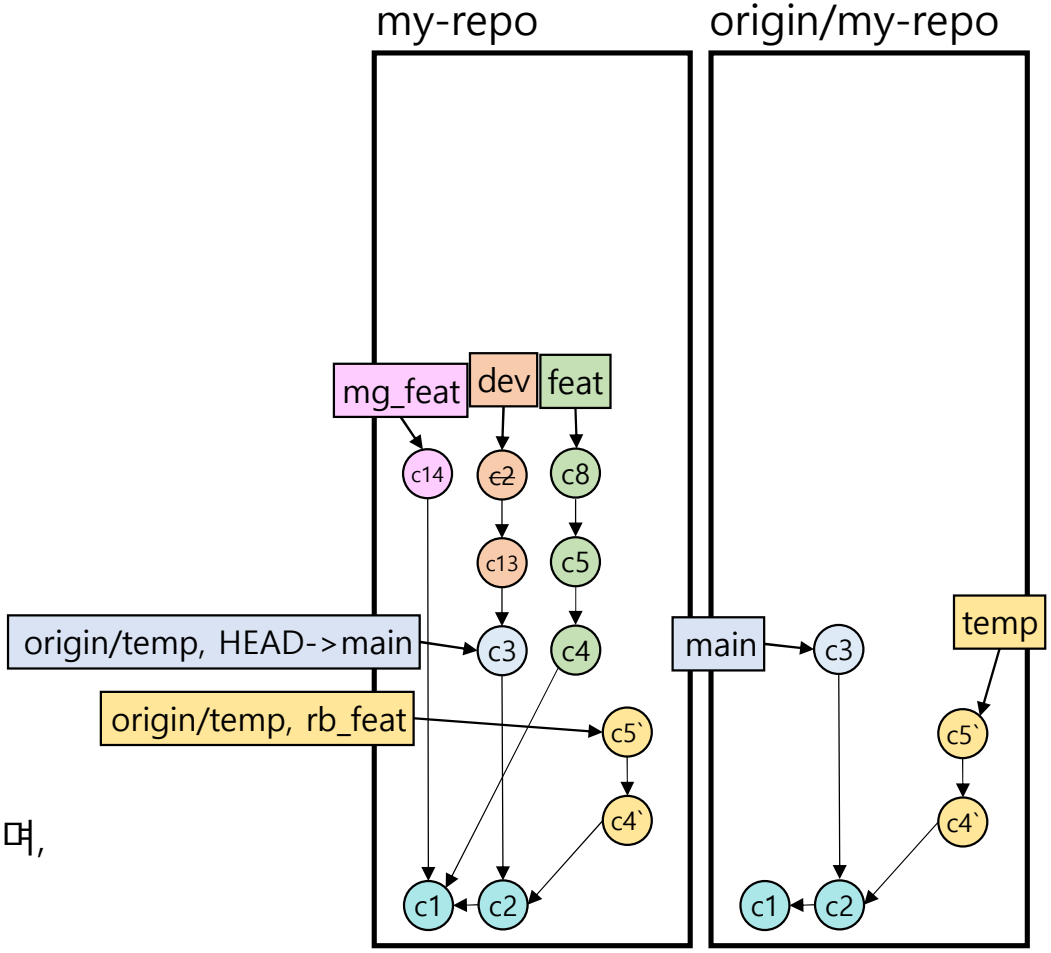
- $ git push <리모트> <로컬브랜치>:<리모트브랜치>
  - <로컬브랜치>를 <리모트>의 <리모트브랜치>에 업로드
  - <리모트브랜치>가 존재하지 않으면 생성

■예제

- (main) $ git push origin main
- (main) $ git push origin rb_feat:temp

■비고

- <로컬브랜치>:<리모트브랜치> 형식을 Refspec이라고 부르며, Refs라는 개념 자체가 어려우므로 설명은 생략함



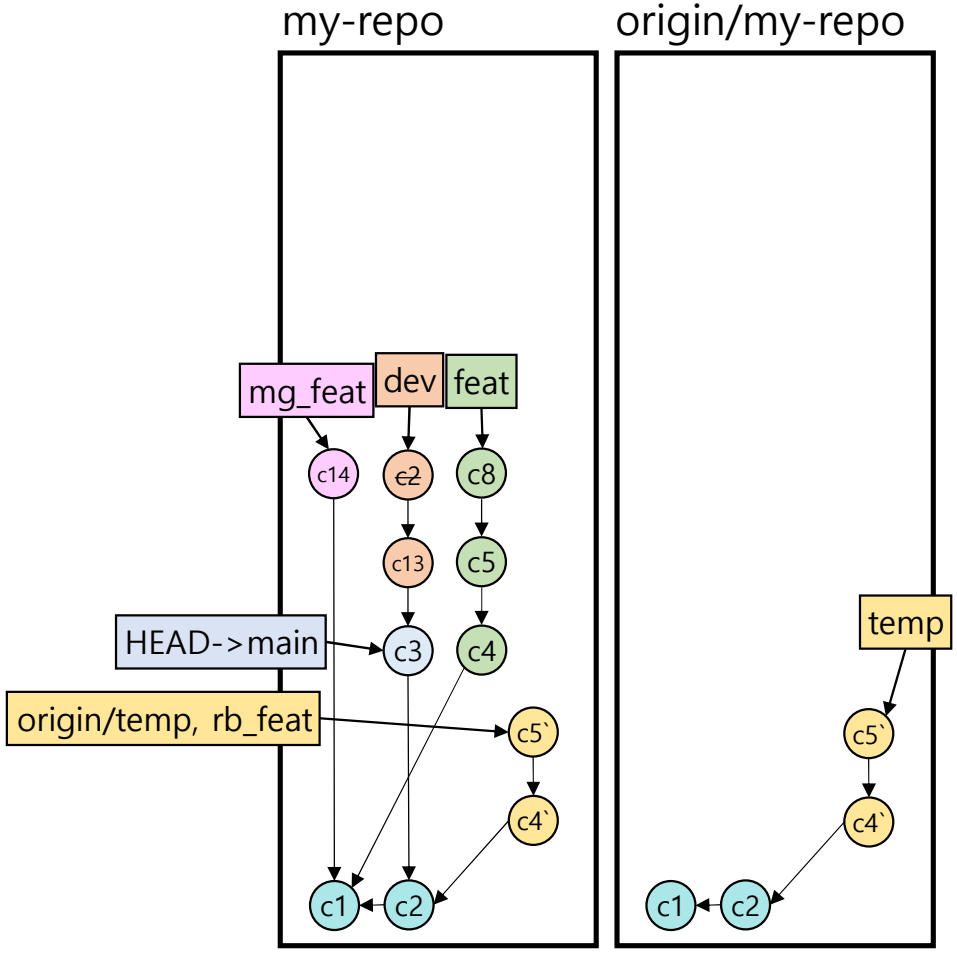
git push - 2

■Remote branch 삭제

- $ git push <리모트> :<리모트브랜치>
  - <로컬브랜치>를 지정해주지 않으면 <리모트브랜치>를 삭제

■예제

- (main) $ git push origin :main



Upstream branch

■Branch 용어

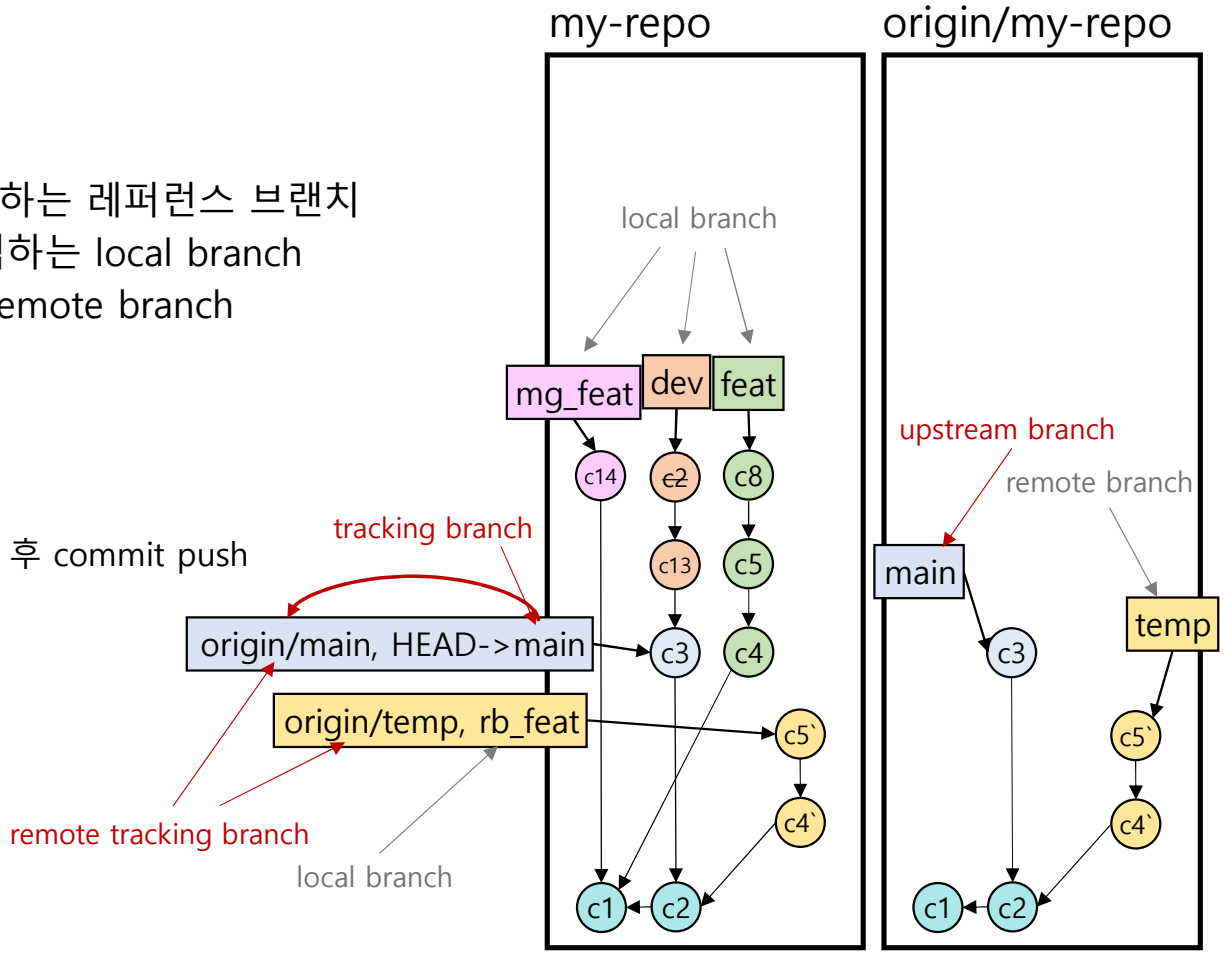
- **local branch:** local repo에 있는 branch
- **remote branch:** remote repo에 있는 branch
- **remote tracking branch:** remote branch를 추적하는 레퍼런스 브랜치
- **tracking branch:** remote tracking branch를 추적하는 local branch
- **upstream branch:** tracking branch가 추적하는 remote branch

■Remote branch 연결

- $ git push -u <리모트> <브랜치>
  - <브랜치>이름으로 upstream branch 생성 or 설정 후 commit push

■예제

- (main) $ git push -u origin main



Upstream branch - 2

■Local branch와 remote branch 연결 및 tracking branch 생성

- ➢ $ git checkout -b <브랜치> <리모트트래킹브랜치>
  - <리모트트래킹브랜치>에 <브랜치>인 tracking branch **생성** 및 checkout
- ➢ $ git branch -u <리모트트래킹브랜치> <브랜치>
  - <브랜치>를 <리모트트래킹브랜치>의 tracking branch로 설정
- ➢ 여기서 <리모트트래킹브랜치>는 "<리모트>/<리모트브랜치>" 형식으로 표현

■예제

- ➢ (main) $ git branch -u origin/temp mg_feat

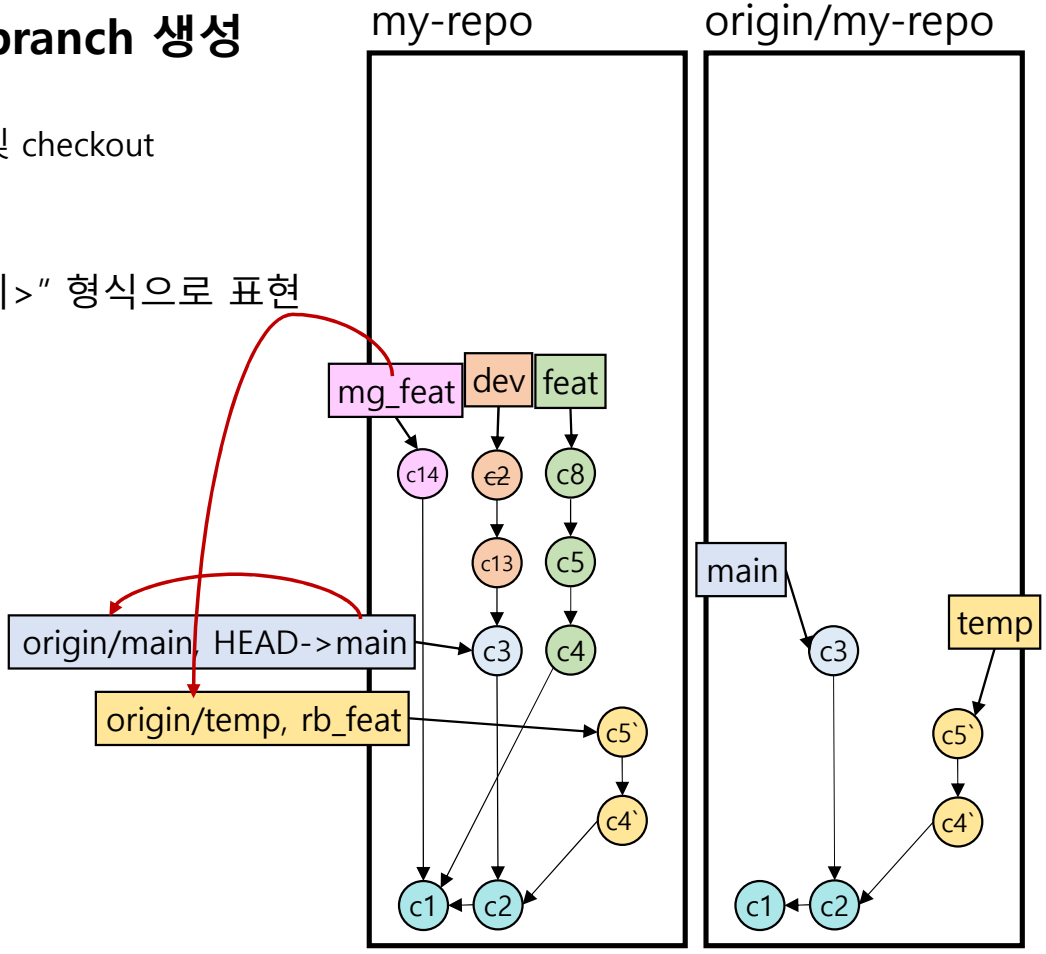
■비고

- ➢ git checkout -b <리모트트래킹브랜치> <브랜치>

  == git branch <브랜치>
  + git branch -u <리모트트래킹브랜치> <브랜치>

- ➢ git push -u <리모트> <브랜치>

  == git push <리모트> <브랜치>
  + git branch -u <리모트트래킹브랜치> <브랜치>



Upstream branch - 3

■Remote tracking branch 확인 방법

- ➢ $ git branch [-r | --remote]
  - remote tracking branch만 확인할 수 있다.
  - local branch를 포함해서 보려면 $ git branch [-a | --all]

■예제

- ➢ (main) $ git branch -a

```
tot0ro@DESKTOP-JUNHO:~/my-repo$  (main) git branch --all
  dev
  feat
* main
  mg_feat
  rb_feat
  remotes/origin/main
  remotes/origin/temp
```

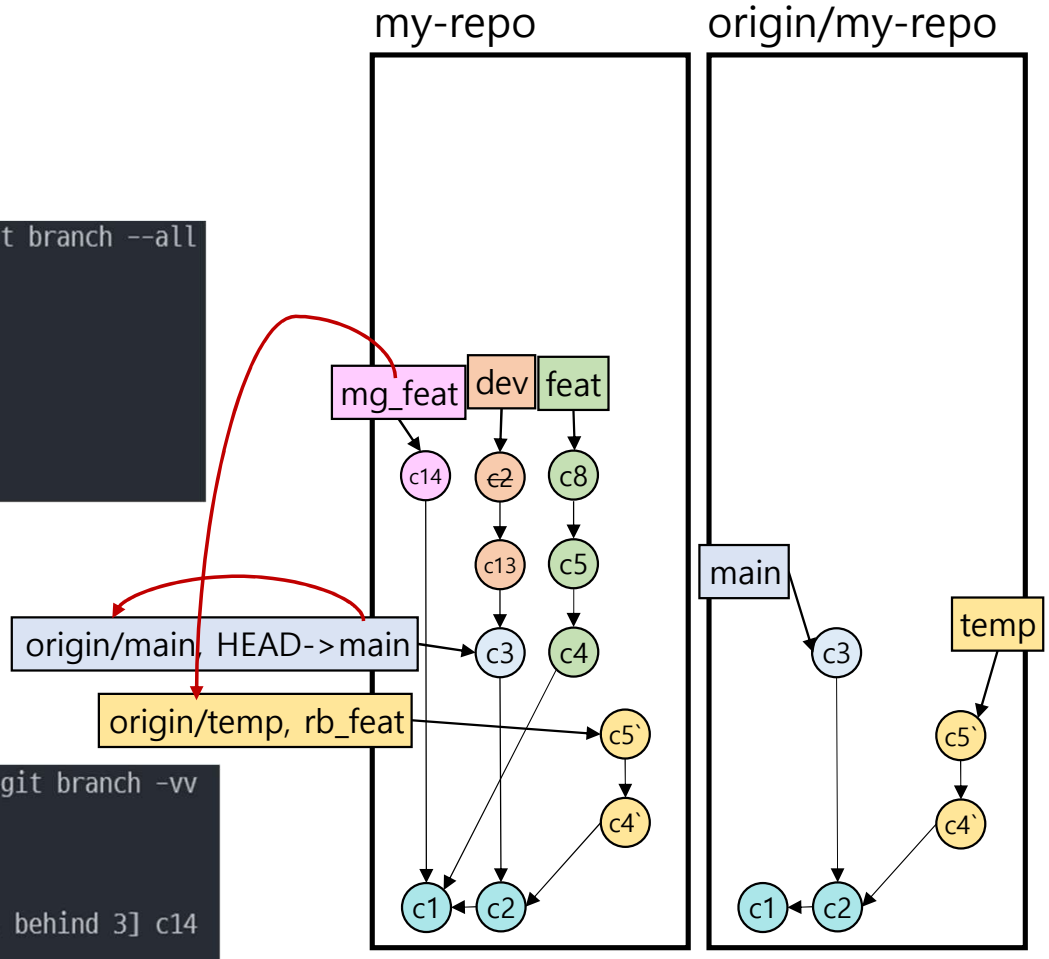
■Tracking branch 확인 방법

- ➢ $ git branch -vv

■예제

- ➢ (main) $ git branch -vv

```
tot0ro@DESKTOP-JUNHO:~/my-repo$  (main) git branch -vv
  dev      cd308da Revert "c2"
  feat     ea5b272 c8
* main     227aa52 [origin/main] c3
  mg_feat 571b465 [origin/temp: ahead 1, behind 3] c14
  rb_feat 4958d62 c5
```



git clone

■원격 repository를 로컬 repository에 복제

- ➢ $ git clone <URL>
  - 외부 Repository 다운로드

■특정 디렉토리 명으로 변경

- ➢ $ git clone <URL> <PATH>

■특정 Branch만 다운로드

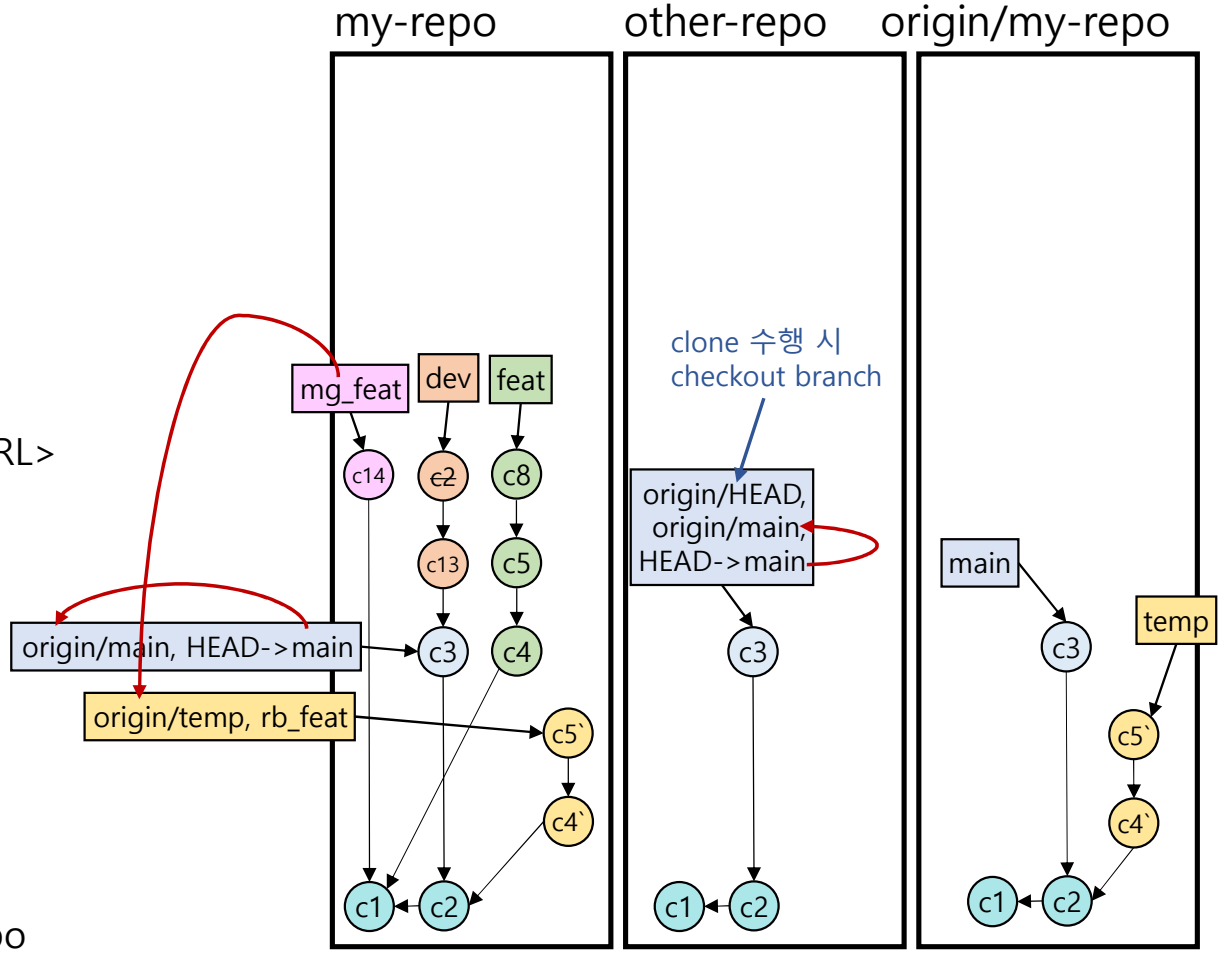
- ➢ $ git clone -b branch_name --single-branch <URL>
  - -b: main이 아닌, 다른 branch를 받고 싶을 때 사용
  - --single-branch: 오직 한 branch만 다운

■모든 LOG를 다 다운받고 싶지 않을 때

- ➢ git clone --depth=<NUM> <URL>
  - History가 너무 클 때 사용

■예제

- ➢ (main) $ git clone --single-branch ₩
  https://github.com/아이디/my-repo.git other-repo



git fetch

■Remote branch를 다운로드

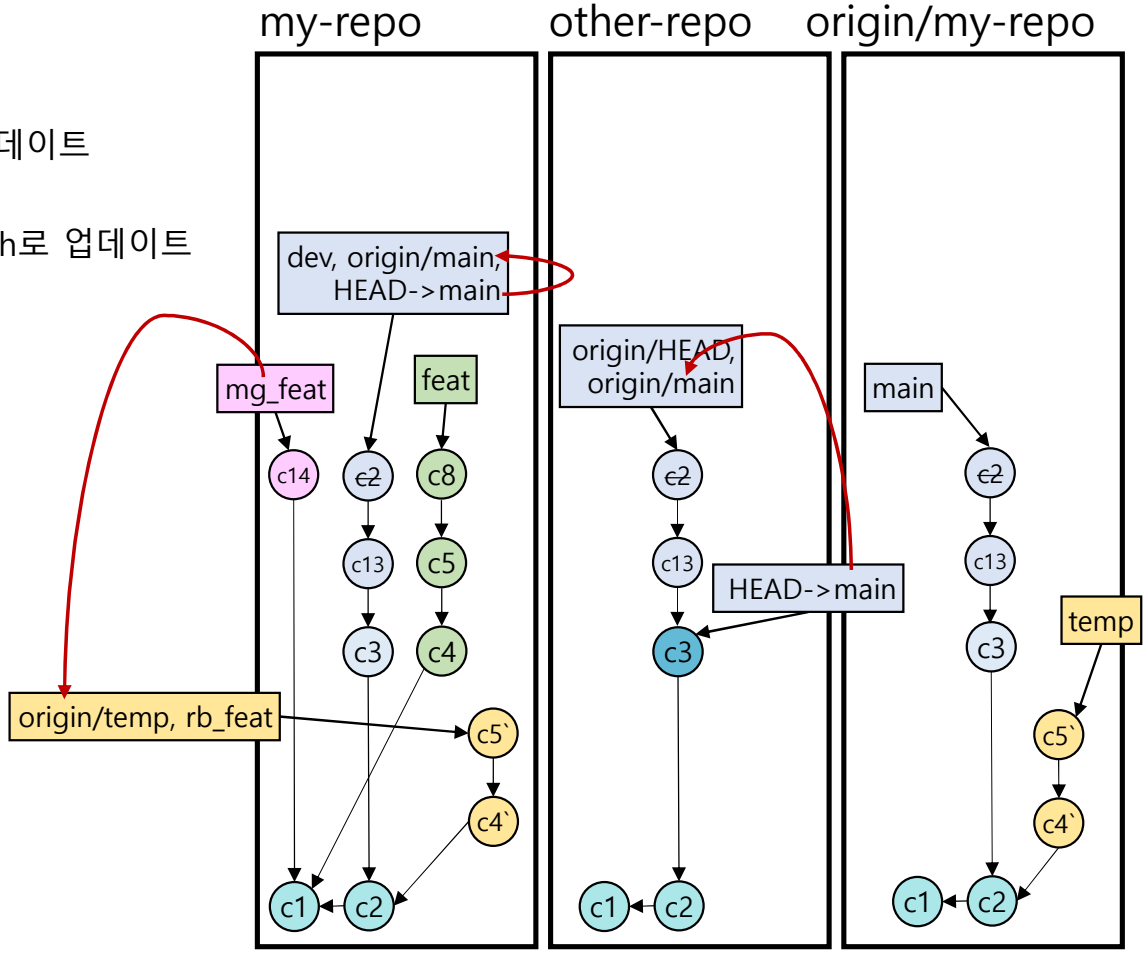
- ➢ $ git fetch <리모트> <브랜치>
  - remote tracking <브랜치>를 upstream <브랜치>로 업데이트
- ➢ $ git fetch
  - 모든 remote tracking branch에 대해서 upstream branch로 업데이트

■주의할 점

- ➢ remote tracking branch가 없는 경우 fetch하지 않음
  - 설령 fetch 되었더라도 추적할 수 없음

■예제

- ➢ my-repo(main) $ git merge dev
- ➢ my-repo(main) $ git push
- ➢ other-repo(main) $ git fetch origin



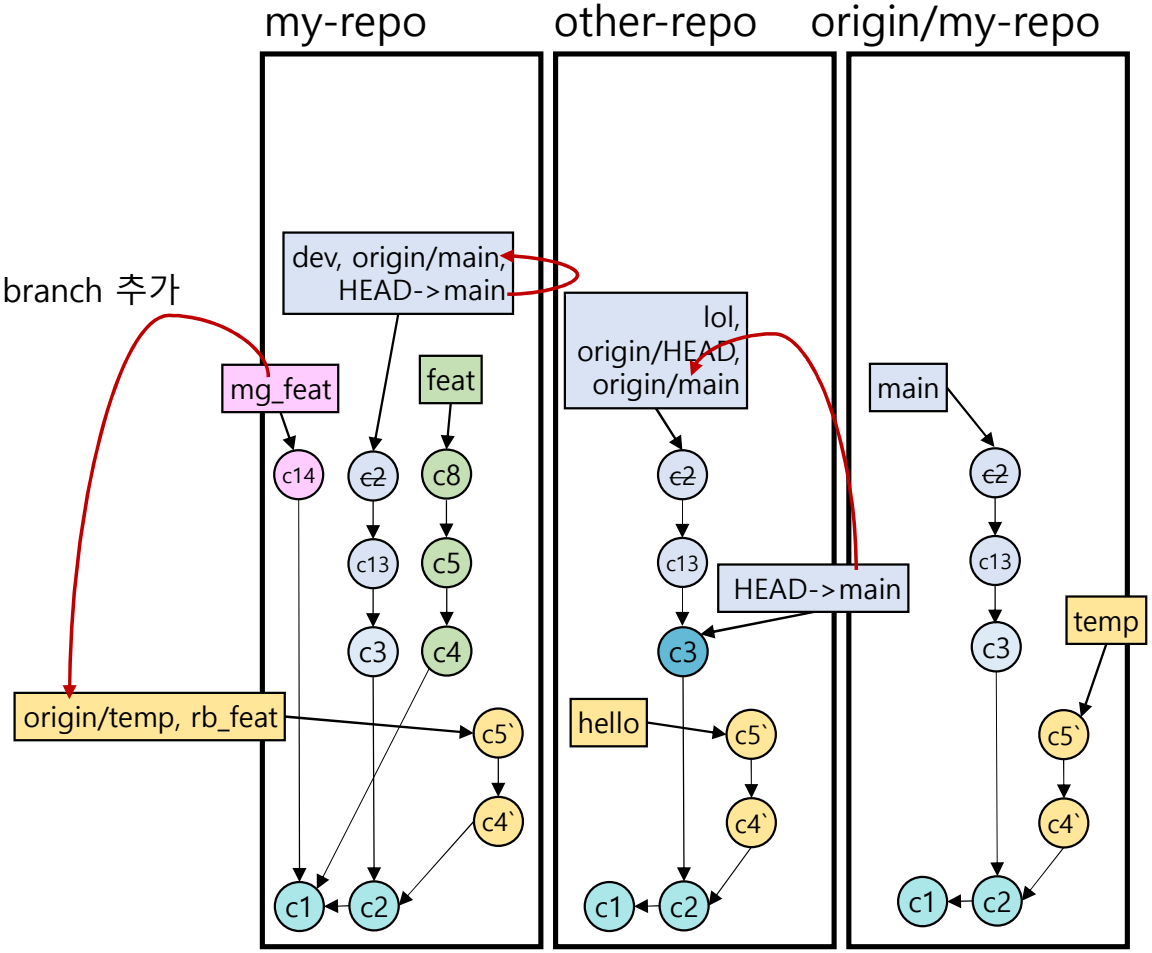
git fetch - 2

■Remote tracking branch 없이 commit을 다운 받고 싶다면

- $ git fetch <리모트> <리모트브랜치>:<로컬브랜치>
  - <리모트브랜치>를 <로컬브랜치>로 다운로드
  - <로컬브랜치>가 없는 경우, 해당 branch 생성
  - <리모트브랜치>를 지정하지 않을 경우 origin/HEAD에 branch 추가

■예제

- other-repo(main) $ git fetch origin temp:hello
- other-repo(main) $ git fetch origin :lol



git pull

■pull = fetch + merge

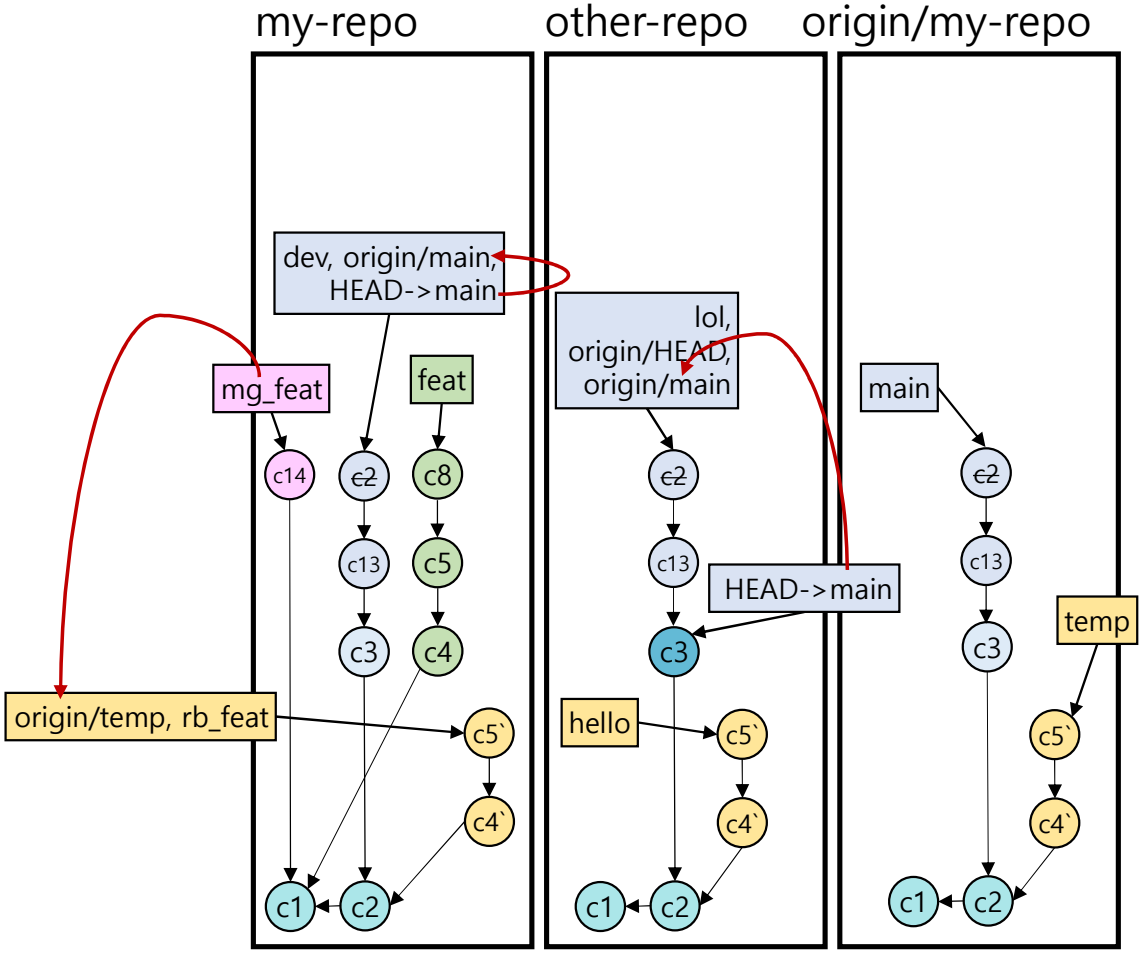
- ➢ $ git pull
  - git fetch + git merge와 동등하다.
  - 인자는 fetch 인자와 동일

■pull --rebase = fetch + rebase

- ➢ $ git pull --rebase
  - git fetch + git rebase와 동등하다.
  - 인자는 fetch 인자와 동일

■예제

- ➢ 생략



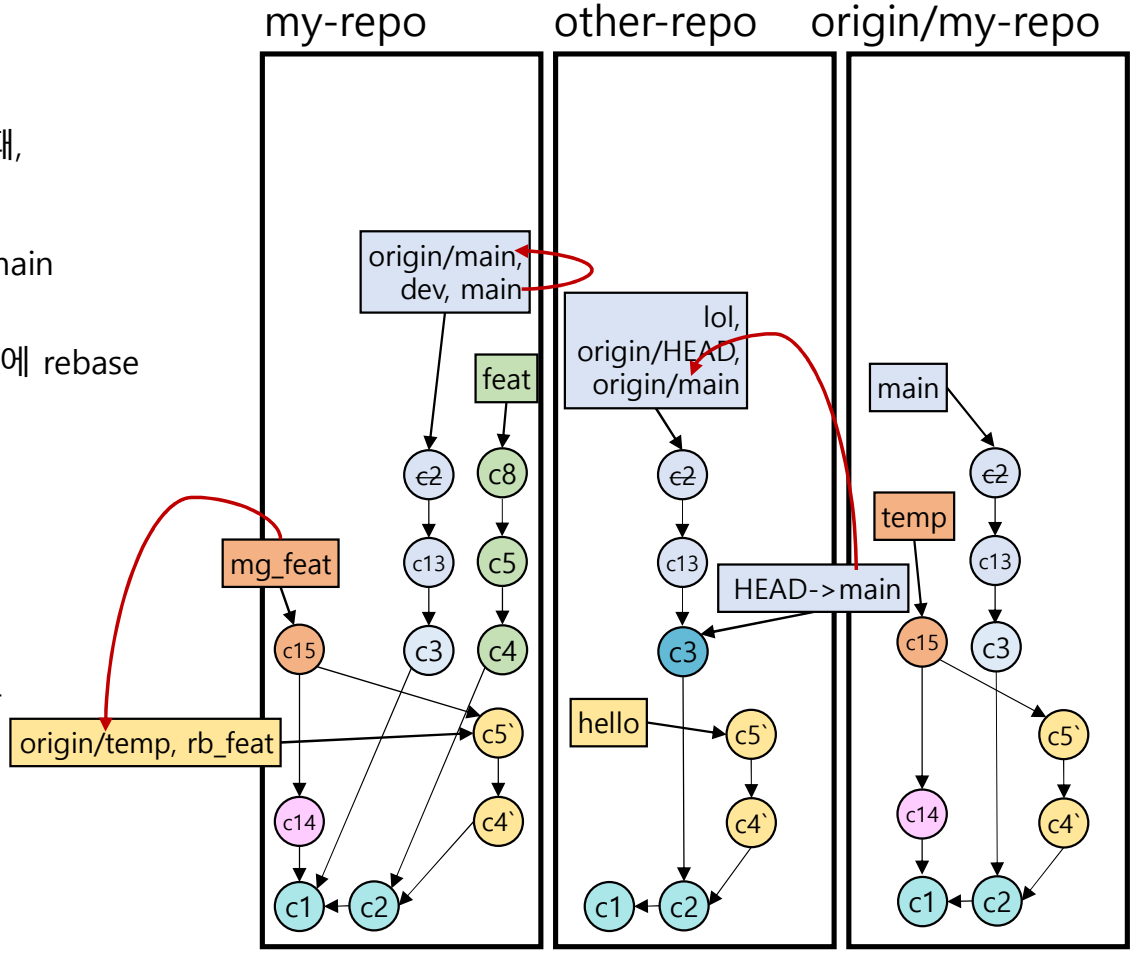
git pull - 2 (merge, rebase)

■"merge"와 "rebase"에 인자가 없으면?

- ➢ $ git merge
  - git merge <브랜치>의 경우, <브랜치>가 생략되었을 때, 현재 branch가 tracking branch일 경우, 대응하는 remote tracking branch와 merge
  - 예) (main) $ git merge == (main) $ git merge origin/main
- ➢ $ git rebase
  - 마찬가지로 <브랜치> 생략 시, remote tracking branch에 rebase

■예제

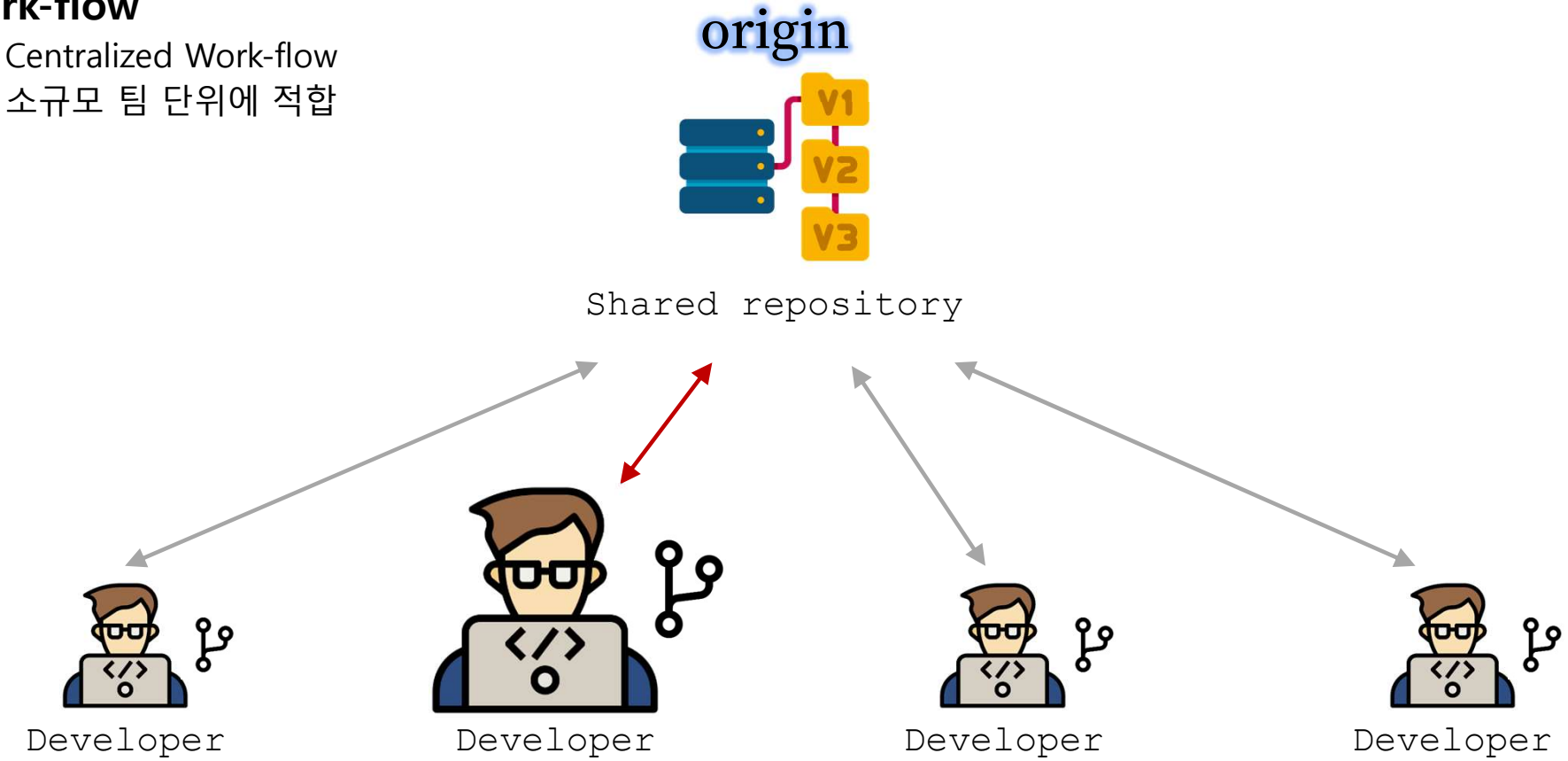
- ➢ my-repo(main) $ git co mg_feat
- ➢ my-repo(mg_feat) $ git merge
- ➢ my-repo(mg_feat) $ git push origin HEAD:test
  - tracking branch와 upstream branch의 이름이 다르므로 인자를 생략할 수 없음
  - HEAD 대신 mg_feat를 적어도 됨



Topology

■Work-flow

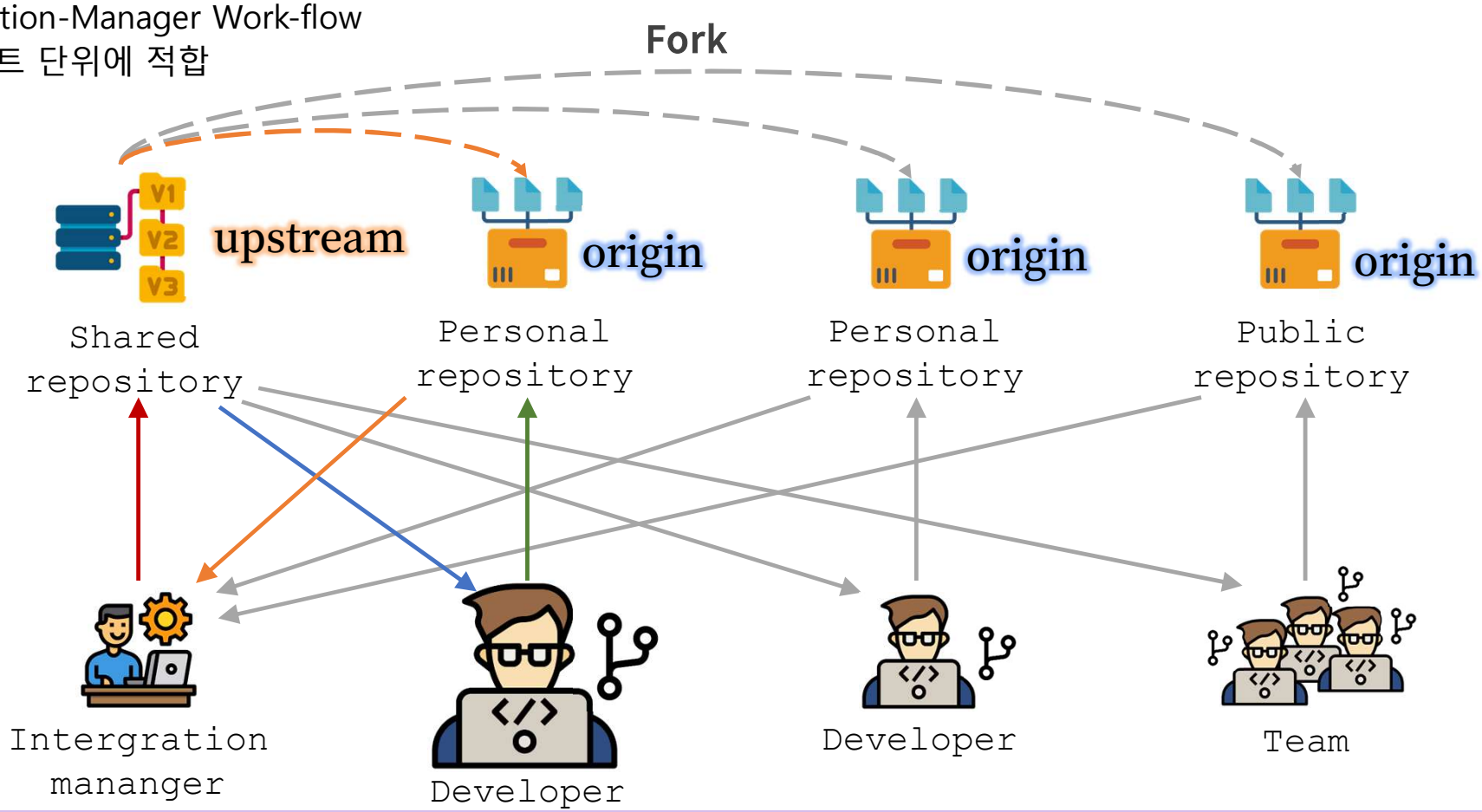
- ➢ Centralized Work-flow
- ➢ 소규모 팀 단위에 적합



Topology

■Work-flow

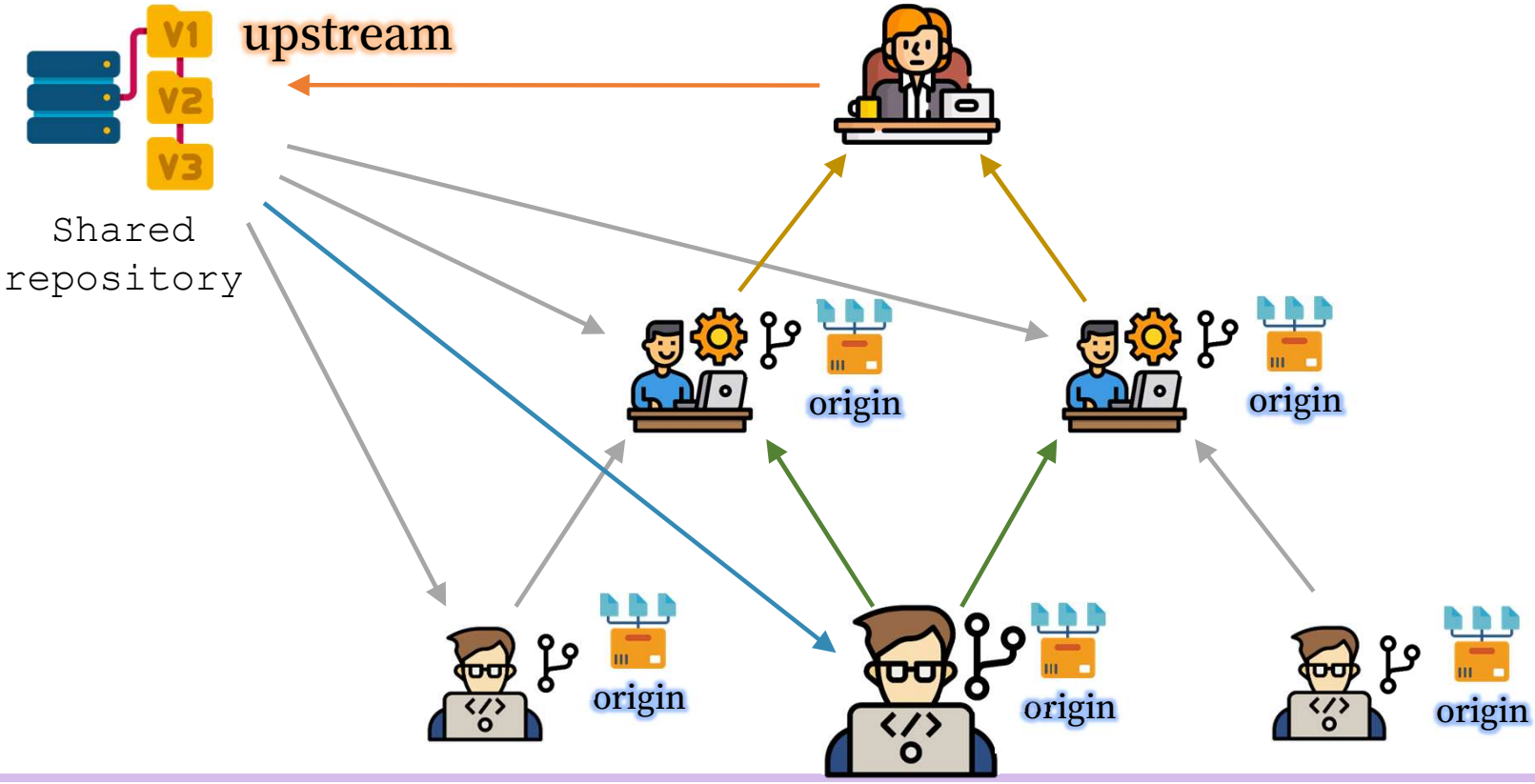
- Integration-Manager Work-flow
- 프로젝트 단위에 적합



Topology

■Work-flow

- Dictator and Lieutenants Work-flow
- 대규모 프로젝트 및 오픈소스에서 적합



Pull Request

■WHAT-IS?

- 기본적으로 branch를 병합하는 것

■즉시 병합하지 않음

- 병합 전에 몇 가지 프로세스를 추가

■커밋 리뷰

- 리뷰어들이나 메인테이너가 해당 커밋에 대한 이슈와 코드를 리뷰하고 소통함
- https://github.com/tensorflow/tensorflow/pulls

■이슈 트래킹

- 버그 발견, 신기능 요청 등 생성한 이슈 ID를 commit에 연결
- https://github.com/tensorflow/tensorflow/issues

■코딩 스타일 확인 및 테스트

- 가독성이 저해되지 않게 규칙에 일관성 있게 개발되었는지, 코드가 정상적으로 동작하는 지 확인
- https://github.com/tensorflow/tensorflow/actions

Pull Request