오픈소스 개발을 위한 GIT 사용법 실습

- SourceTree와 Github를 이용 -

장병진 (jabgbi882@gmail.com)

강의내용 구성

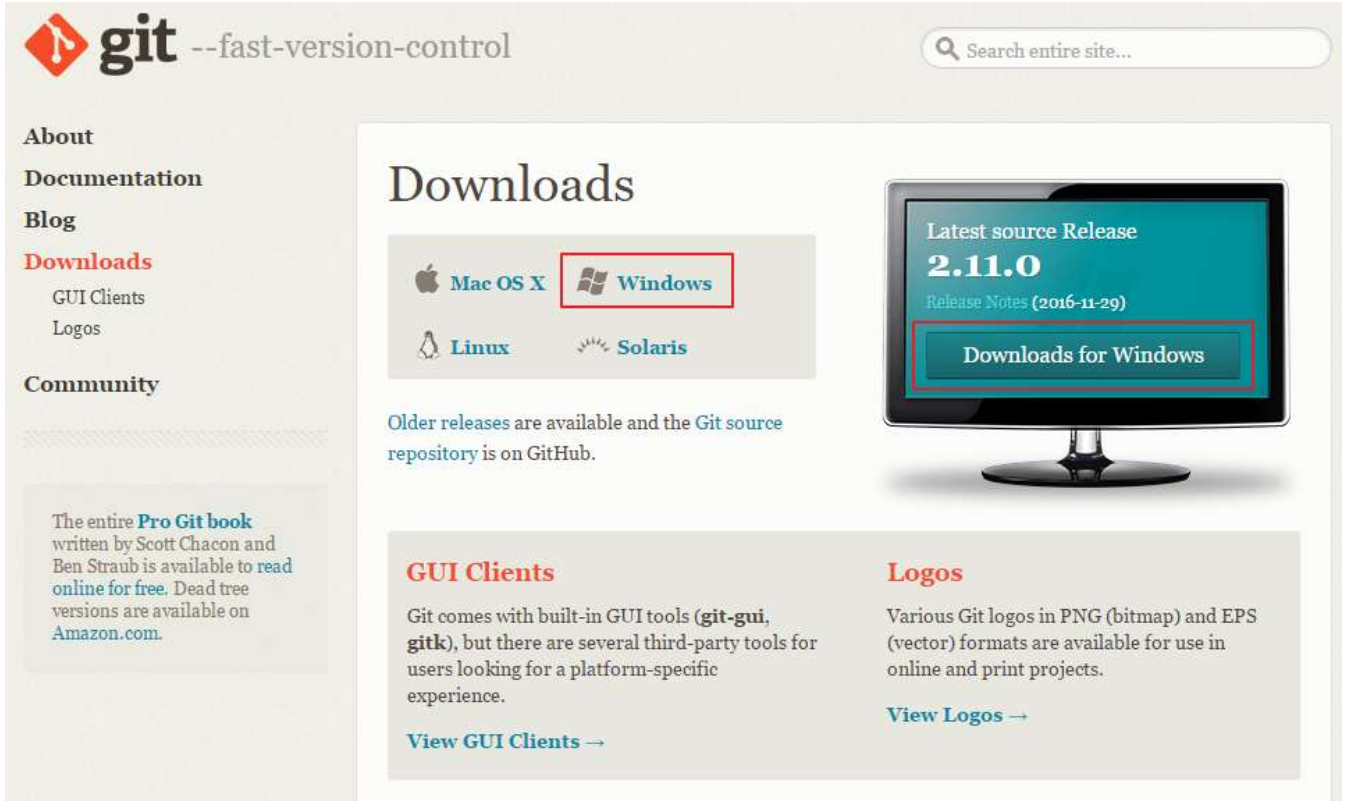
I. 필요 프로그램 설치
II. Github에 저장소 만들기
III. Git 개념잡기
IV. Git에 변경내용 저장
V. 브랜치를 이용한 작업
VI. 변경사항 되돌리기
VII. 오픈소스 참여 위한 저장소 구성

필요 프로그램 설치

Git 설치

Do it!

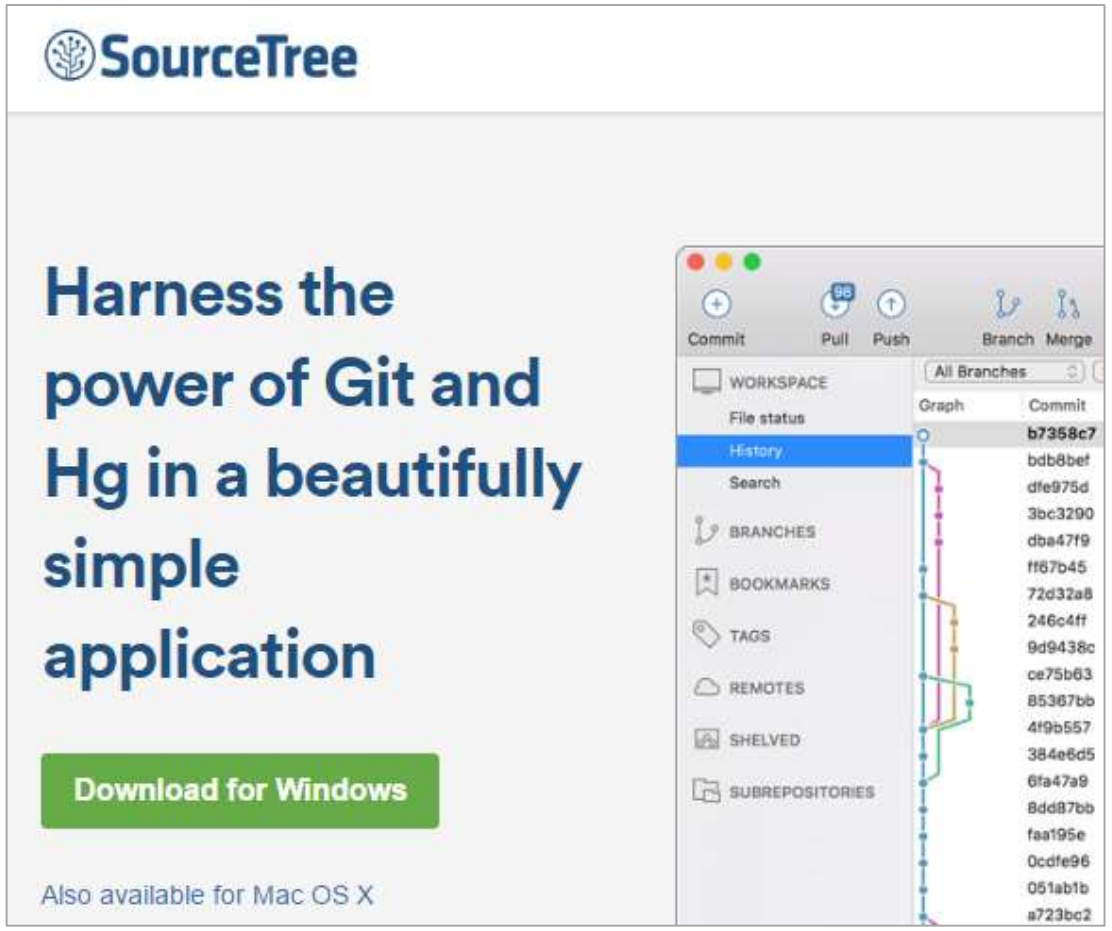
- Git의 저장소 기능과 명령어들을 제공
- 여러가지 git 배포본이 있음
- 윈도우용은 https://git-scm.com/downloads 에서 배포하는 버전 많이 사용
- 설치파일 다운받아 실행하면 쉽게 설치 가능
- 옵션 별도 선택 없이 기본값으로 설치하면 됨



SourceTree 설치

Do it!

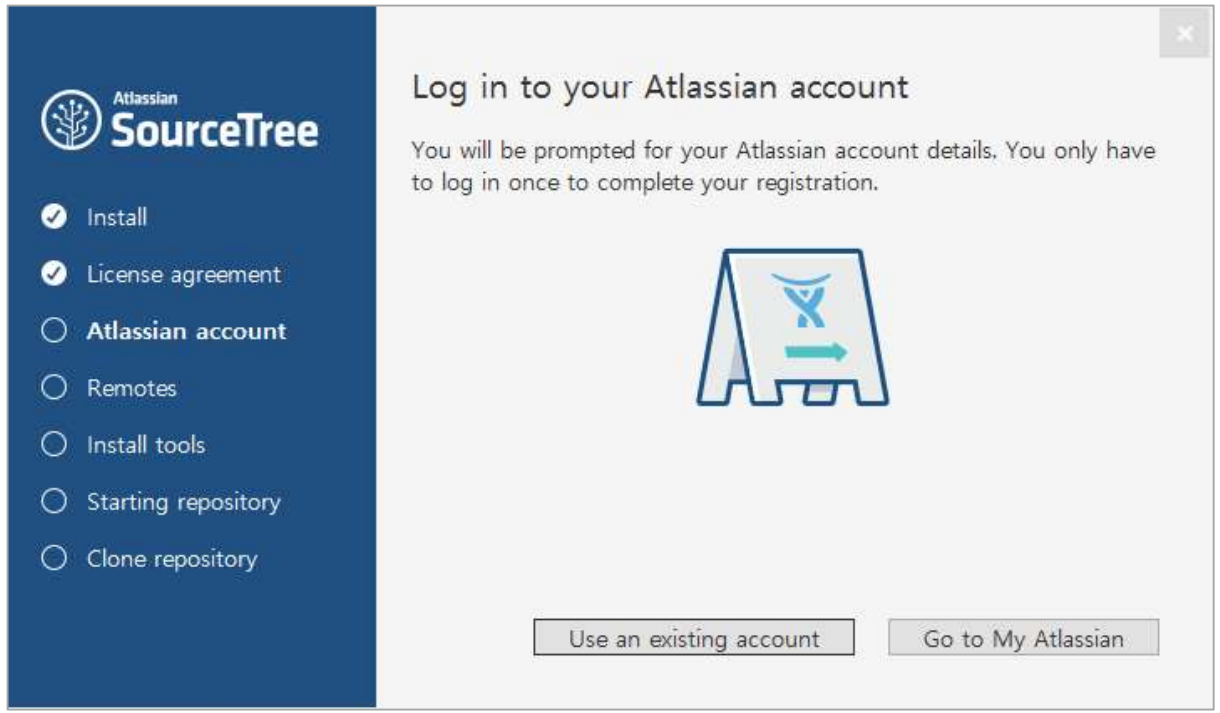
- 명령어 위주인 git을 UI를 통해서 쉽게 사용할 수 있게 해주는 툴
- 추상적인 저장소 내의 History를 시각적으로 보여 줌
- https://www.sourcetreeapp.com/ 에서 다운로드
- 설치파일을 실행해 쉽게 설치

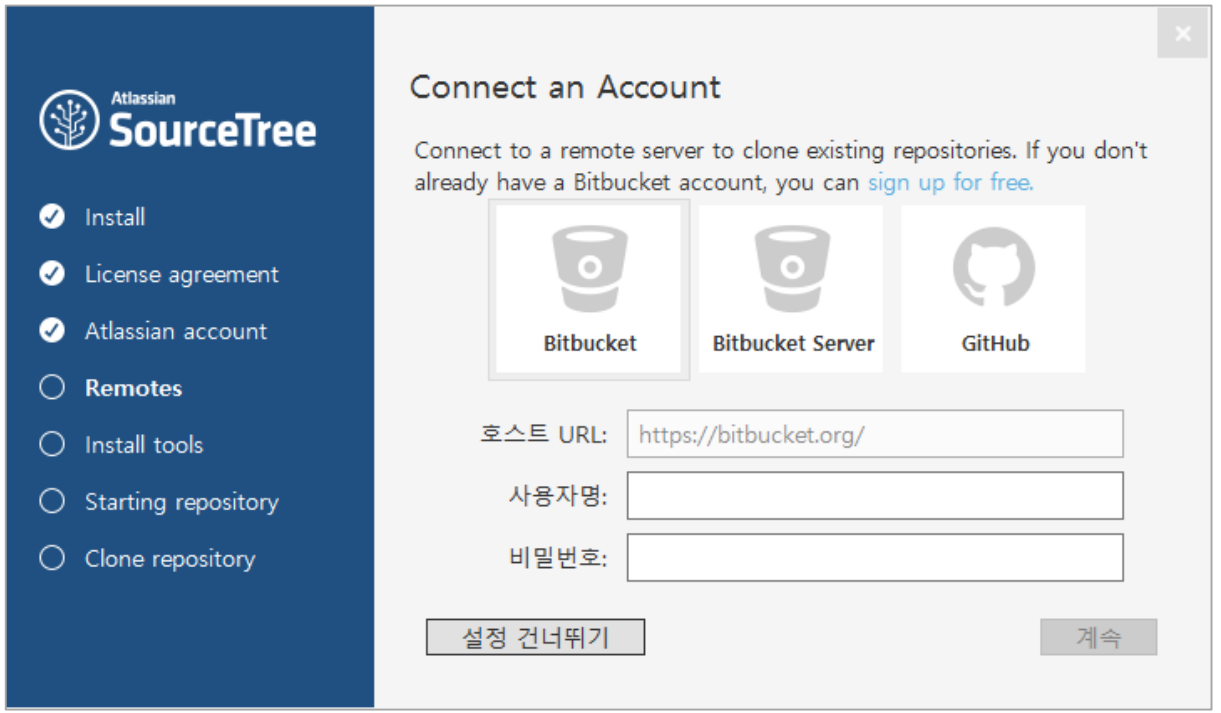


SourceTree 실행

Do it!

- 실행시 여러가지 계정이 필요함
- 무료이지만 Atlassian 계정이 있어야 사용 가능 (한번만 입력)
- Github 연동 위해 Github 계정도 필요

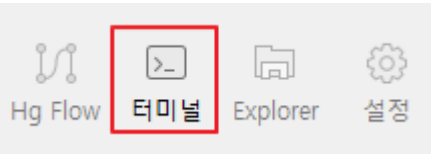




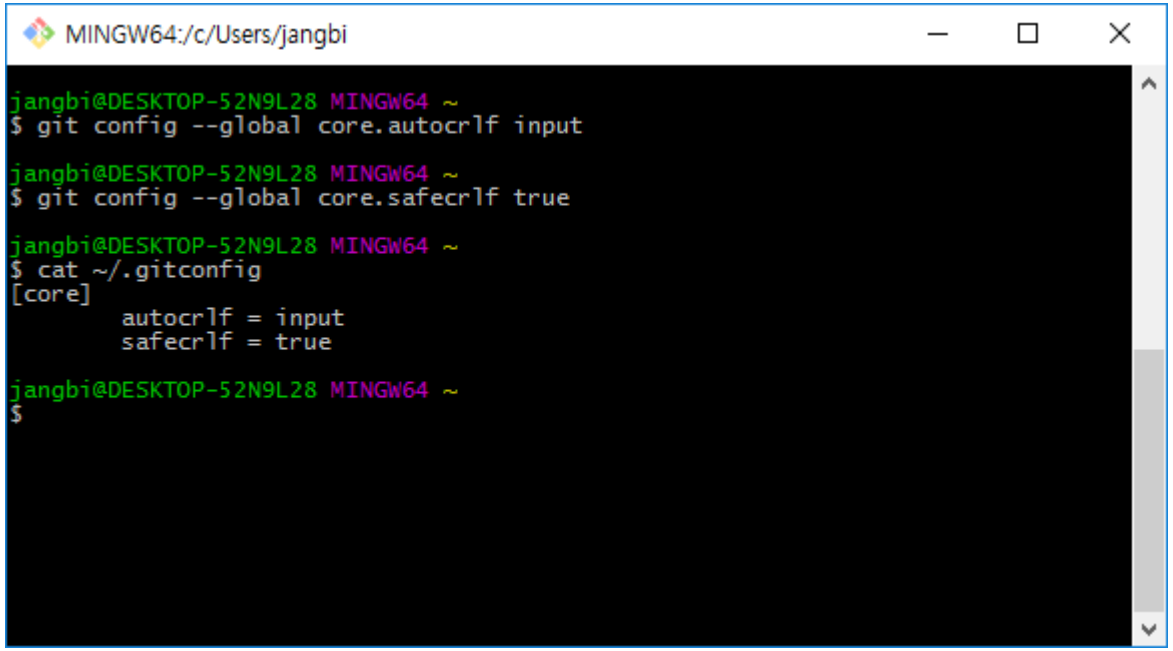
SourceTree의 기본설정 변경

Do it!

- 윈도우가 아닌 Linux, Mac 등과의 협업을 위해 줄바꿈 문자에 대한 설정 변경이 필요 (줄바꿈에 의한 불필요한 변경탐지 방지)
- SourceTree의 터미널 기능에서 직접 명령어를 입력
- 변경된 설정은 ~/.gitconfig 파일에 저장됨




```
git config --global core.autocrlf input
git config --global core.safecrlf true
```



Notepad++ 설치

Do it!

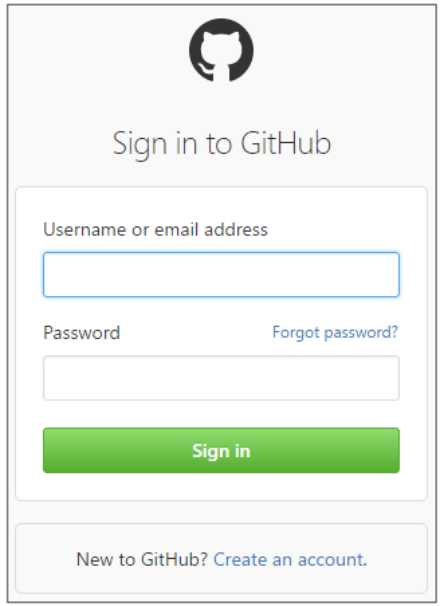
- 무료지만 무척 편리하고 강력한 텍스트 편집기
- 한글 인코딩을 거의 완벽하게 판단함
- 파일의 변경을 자동 판단하는 기능 편리
- https://notepad-plus-plus.org/
- 설치 파일 다운받아 계속 다음으로 넘어가면 쉽게 설치
- 64비트 버전은 안되는 플러그인이 많아 32비트 버전을 사용하는 것이 좋음
- 실습에서 소스변경에 사용

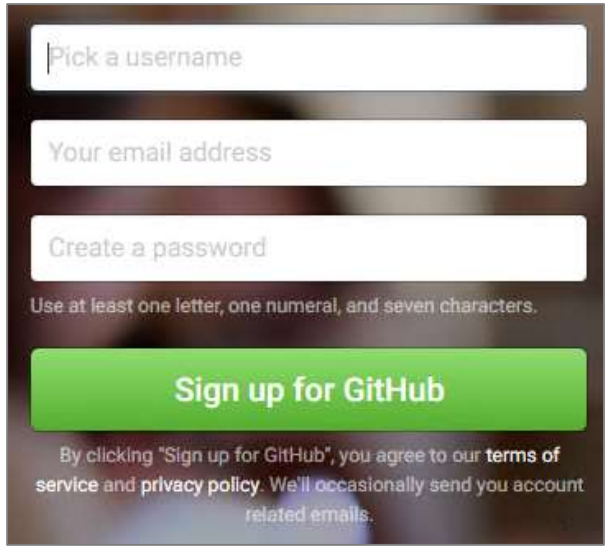
Github에 저장소 만들기

Github 접속과 가입

Do it!

- https://github.com 에 접속한다.
- 계정이 있는 사람은 [Sign in]으로 로그인
- 계정이 없는 사람은 [Sign up]으로 가입
  - 가입시 username과 email이 기존계정과 겹치면 안된다.
  - 비용 플랜은 Free를 선택하면 공개 저장소만 만들 수 있다.
  - 가입 완료 후 꼭! 이메일 인증을 받아야 한다.

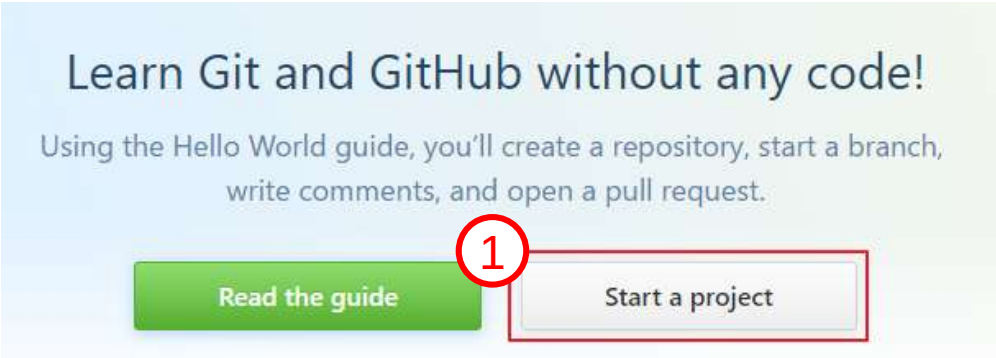


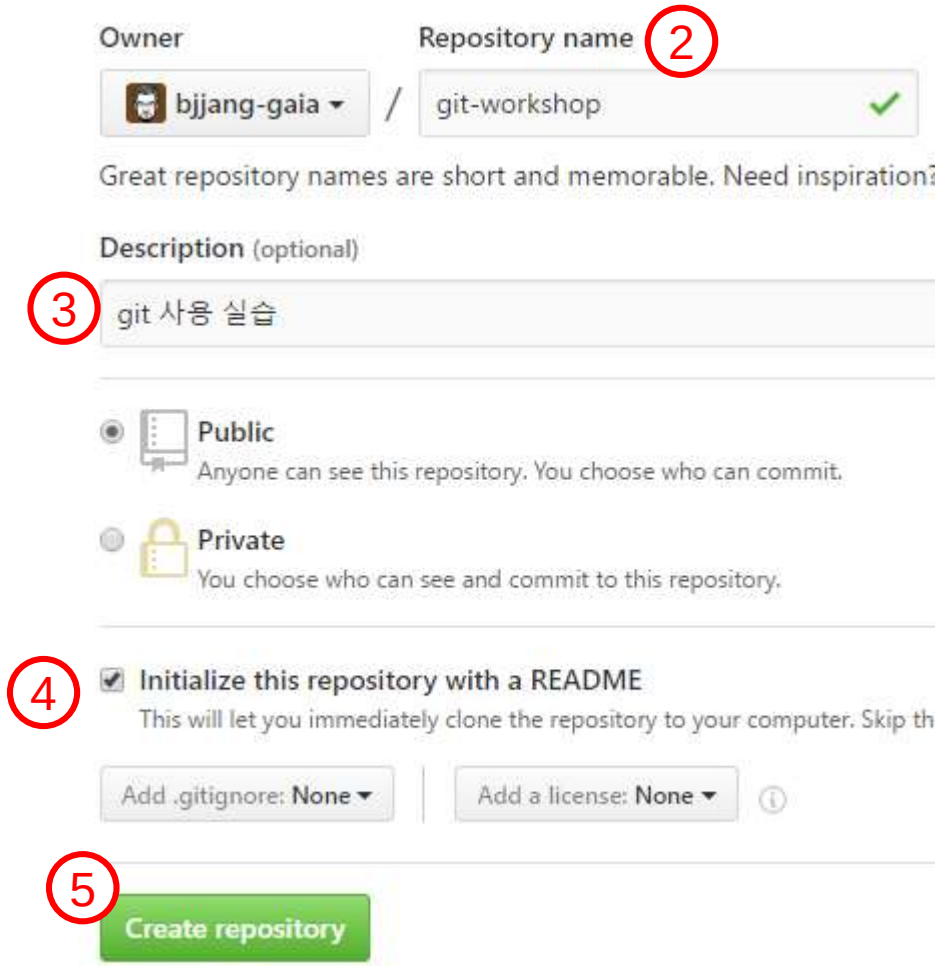


새 github 프로젝트 만들기

Do it!

1. 로그인 한 첫 화면에서 [Start Project]를 누른다.
2. Repository Name에 ‘git-workshop’ 이라 입력한다.
3. Description에 ‘git 사용 실습’ 이라 입력한다.
4. Initialize this repository with a README 옵션에 체크한다.
5. [Create repository] 를 누른다.

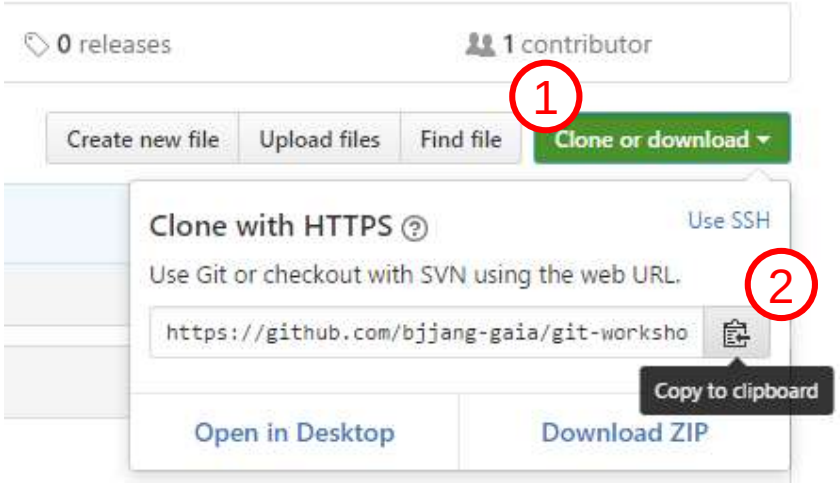


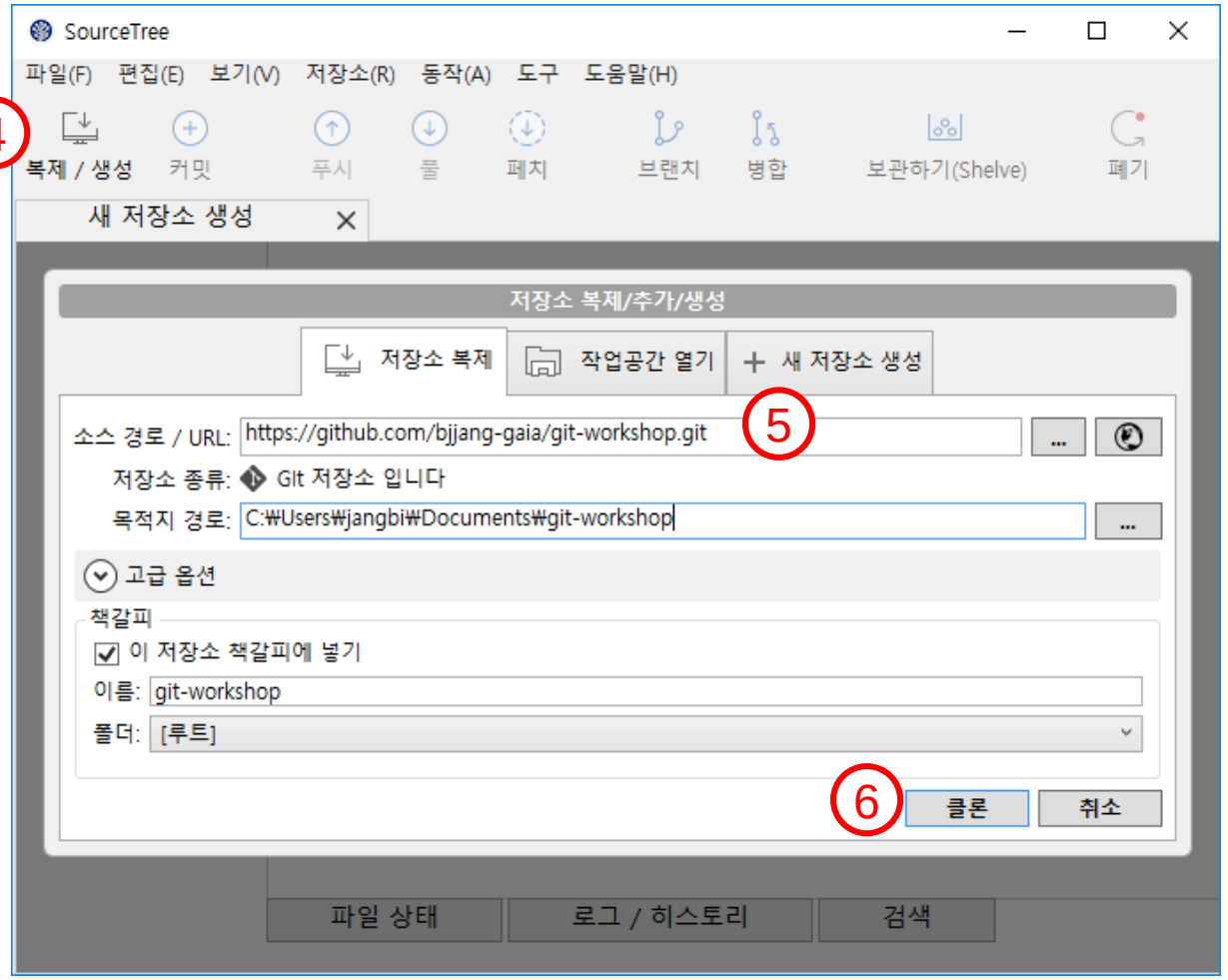


로컬 PC에 받아오기

Do it!

1. 생성된 Github의 저장소 페이지에서 [Clone or download] 버튼을 누른다.
2. 주소를 복사하는 버튼을 누른다.
3. SourceTree를 시작한다.
4. [복제 / 생성] 버튼을 누른다.
5. 소스경로 / URL에 경로의 주소를 붙여 넣는다.
6. [클론] 버튼을 누른다.






```
git clone https://github.com/<github계정>/git-workshop.git
```

원격저장소 복제 확인

Do it!

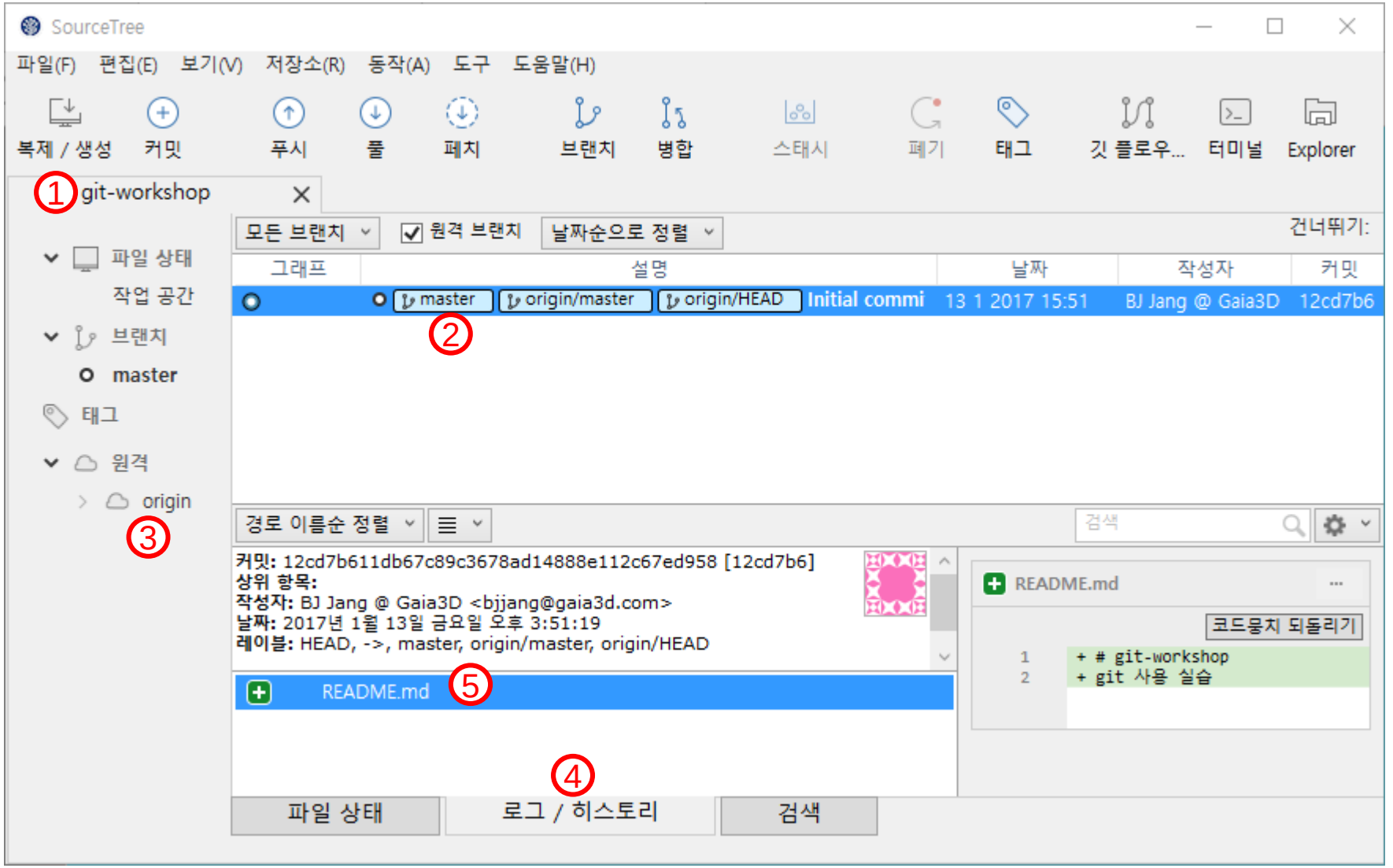
① git-workshop 탭이 새로 생김

② Master 브랜치 확인

③ Origin 원격저장소 확인

④ 로그/히스토리 확인

⑤ README.md 열어보기



Git의 개념 잡기

Git 이란?

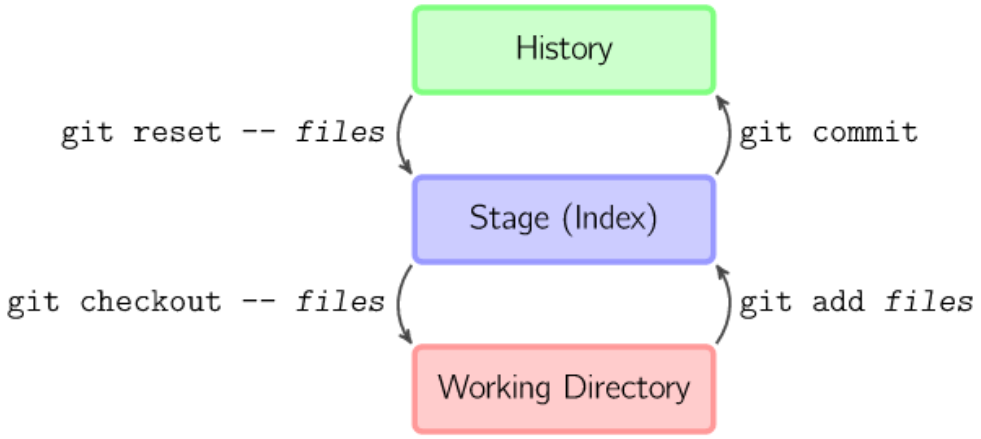
View

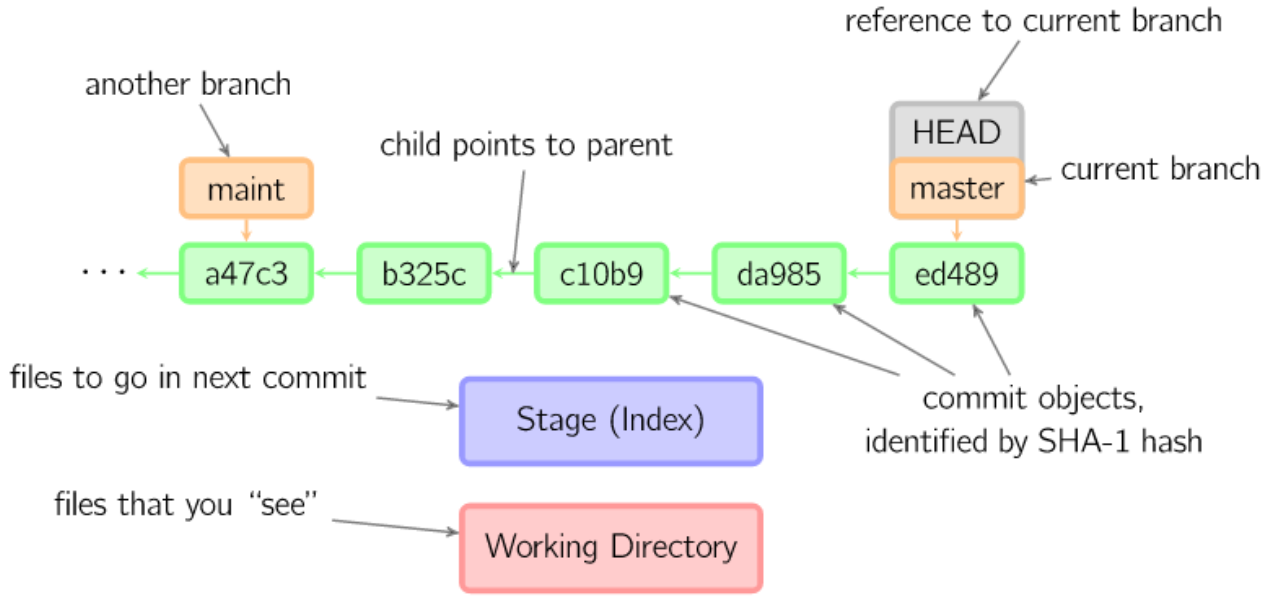
- Git은 분산 버전관리 시스템
  - Distributed Version Control System (DVCS)
  - 여러 사람이 협동작업하는 환경에서 문서변경사항을 관리하는 시스템
  - https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control
  - https://git-scm.com/book/ko/v2/
- 특징
  - 변경사항을 적절히 저장했다가 필요한 시점으로 돌릴 수 있다.
  - 서로 다른 변경사항들을 쉽게 합칠 수 있는 기능을 제공한다.
  - 저장소가 로컬(내 컴퓨터)에 있어 네트워크가 끊어져도 작업 가능하다.
  - 다른 버전관리 시스템보다 빠르다.
  - 원격저장소를 연결해 협동작업이 가능하다.
  - 변경사항을 관리할 대상을 스테이지(Stage)를 이용해 관리 가능하다.
- 참고자료
  - 버전관리를 모르는 분들을 위한 자료: http://www.slideshare.net/ibare/dvcs-git
  - SVN을 사용하던 분들을 위한 자료: http://www.slideshare.net/einsub/svn-git-17386752
  - 시각적으로 설명한 GIT 명령: http://marklodato.github.io/visual-git-guide/index-ko.html
  - 실습위주의 교재: http://www.slideshare.net/flyskykr/github-46014813

Git의 3가지 영역

View

- 작업 폴더(Working Directory)
  - 사용자가 변경하는 실제 파일이 들어가는 폴더
- 스테이지(Stage, Index)
  - 변경사항을 관리할 파일들의 리스트
- 변경이력(Histroy)
  - 커밋(Commit)이라 불리는 변경사항 묶음과 커밋들의 연결관계



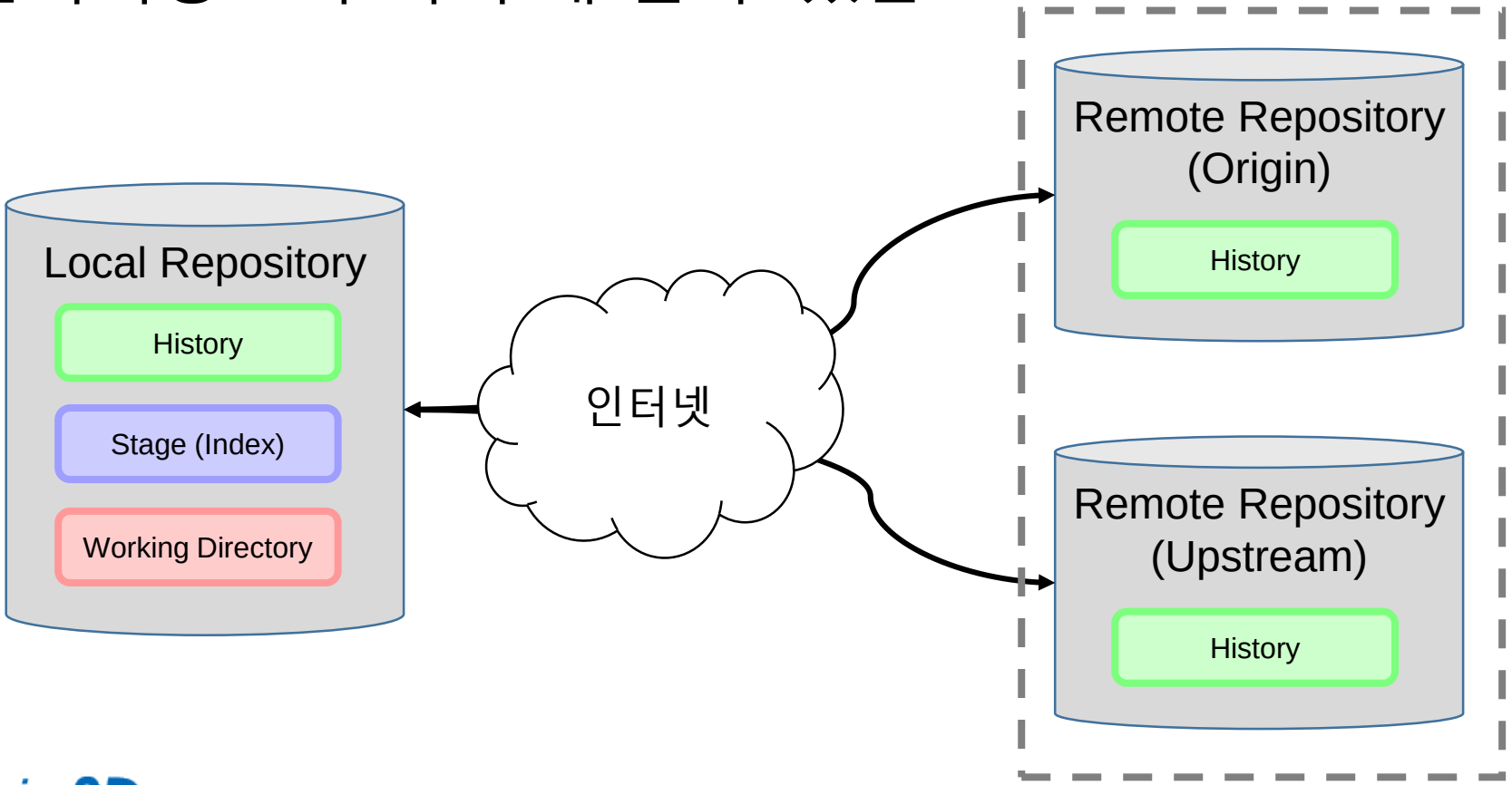


http://marklodato.github.io/visual-git-guide/index-ko.html

로컬저장소와 원격저장소

View

- 협업을 위해서는 원격저장소가 필수적
- 로컬저장소와 원격저장소 간에 이력을 주고받을 수 있음
- 원격저장소가 여러 개 일 수 있음



많이 사용되는 원격 저장소

- GitHub
- BitBucket
- GitLab

Git에 변경내용 저장

README.md 수정과 커밋

Do it!

① Notepad++ 실행

② git-workshop 폴더의 README.md 파일 열기

③ '1. 아침' 추가하고

④ 저장

⑤ SourceTree 실행

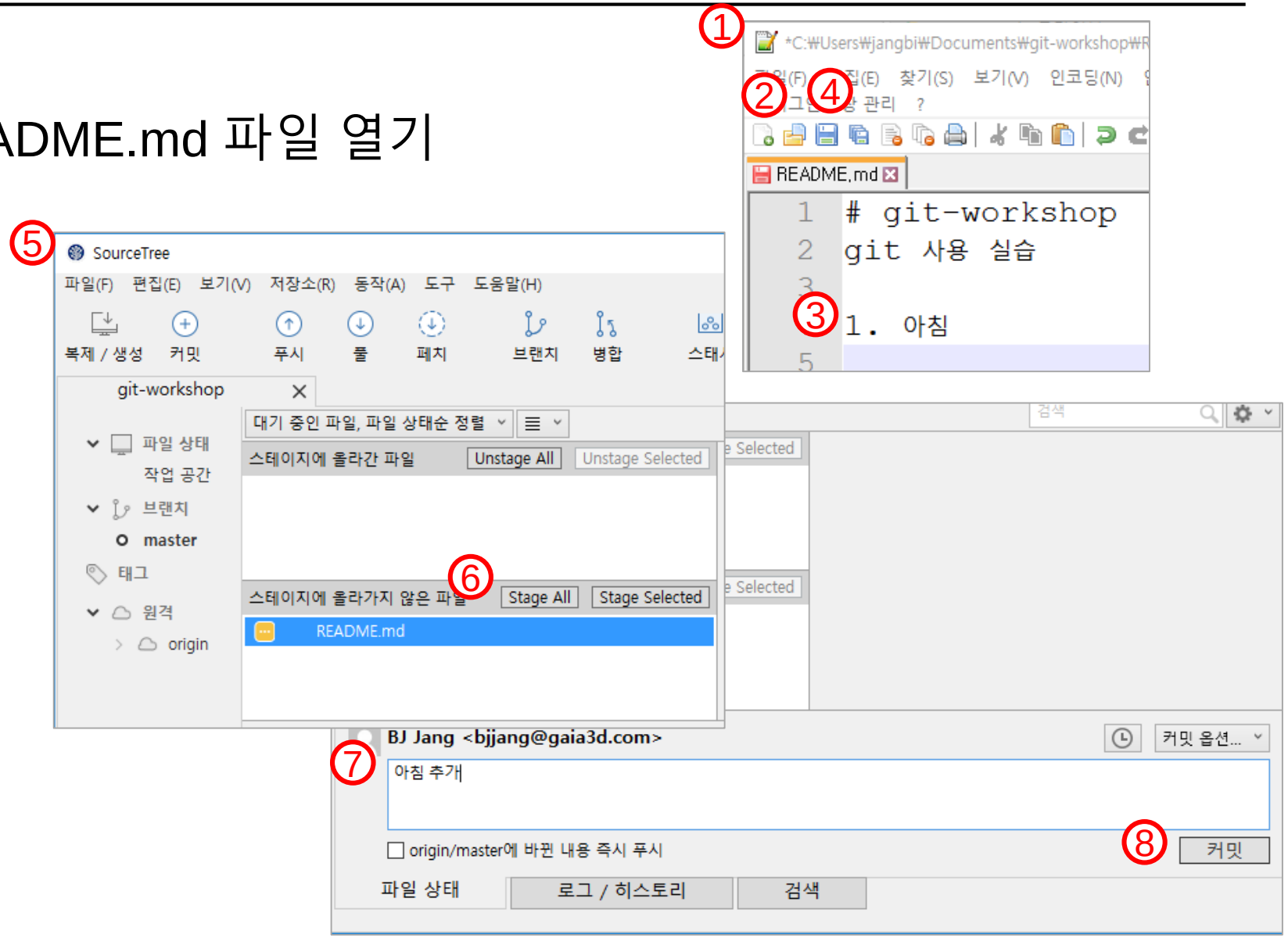
⑥ [Stage All] 눌러 스테이지 올리기

```
git add README.md
```

⑦ '아침 추가 ' 라고 커밋 메시지 입력

⑧ [커밋]

```
git commit -m "아침 추가"
```



점심, 저녁 커밋 생성

Do it!

① 에디터에서 다음줄에 '2. 점심' 입력후 저장

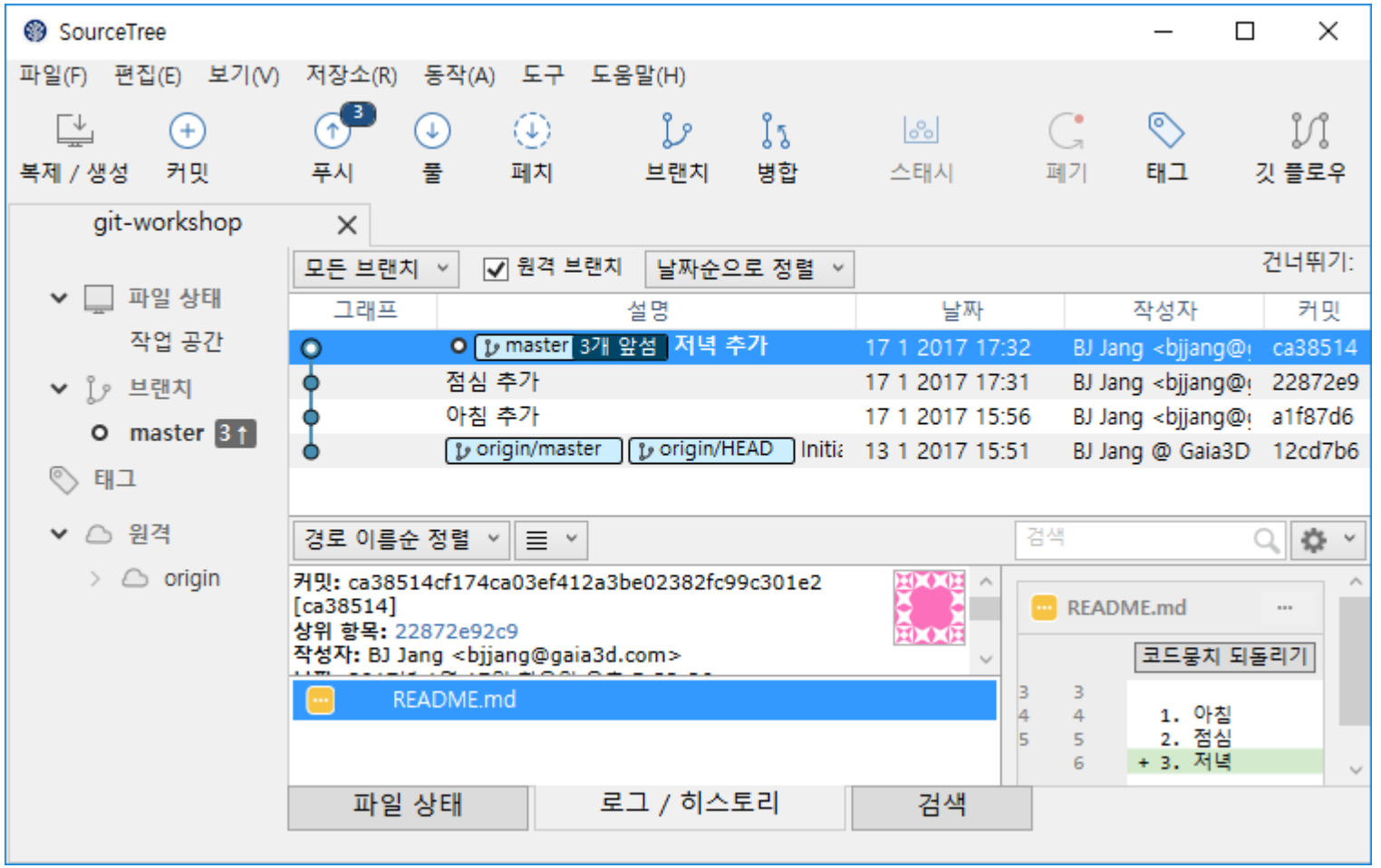
② SourceTree에서 Staging, Commit

③ 에디터에서 다음줄에 '3. 저녁' 입력후 저장

④ SourceTree에서 Staging, Commit

⑤ SourceTree에서 로그/히스토리 확인

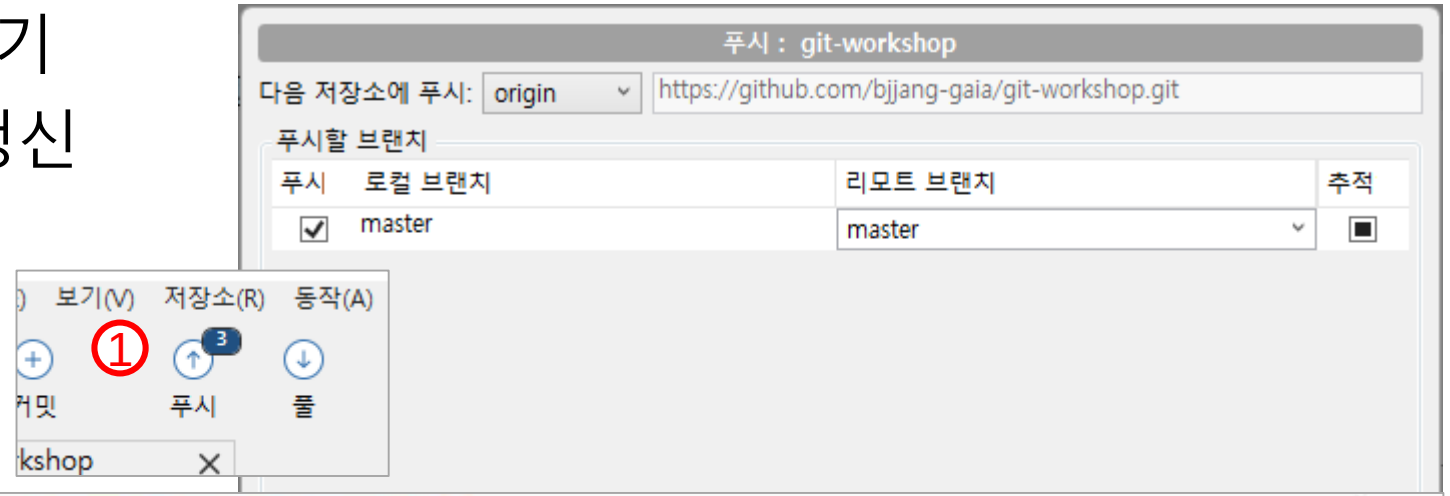
```
git log
```

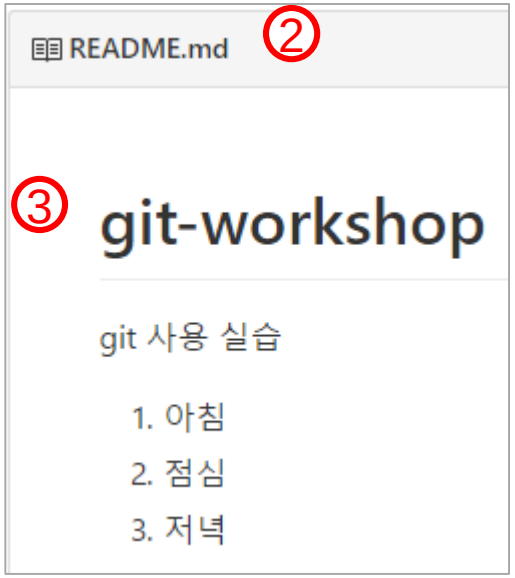


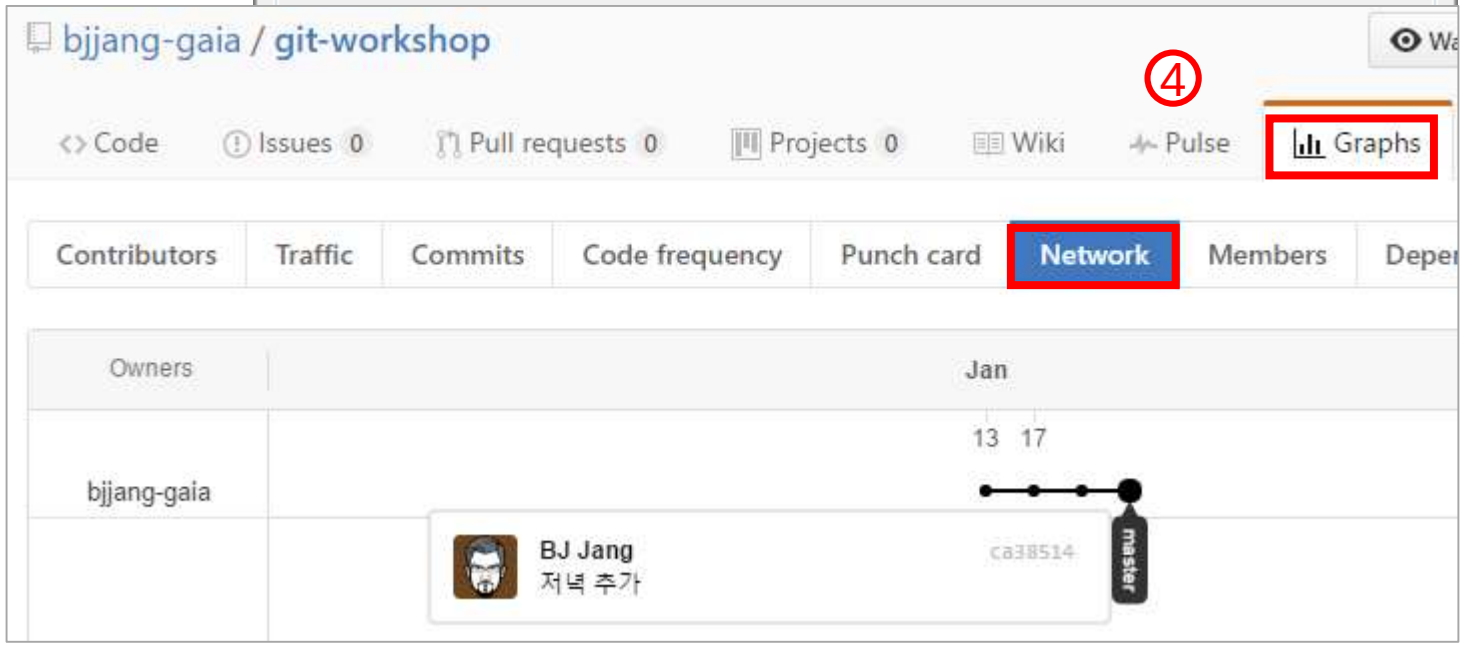
원격저장소에 올리기

Do it!

① [푸시] 버튼으로 원격에 올리기
② Github의 프로젝트 페이지 갱신
③ 메인 페이지 갱신 내용 확인
④ 이력 확인







브랜치를 이용한 작업

새 브랜치 만들기

Do it!

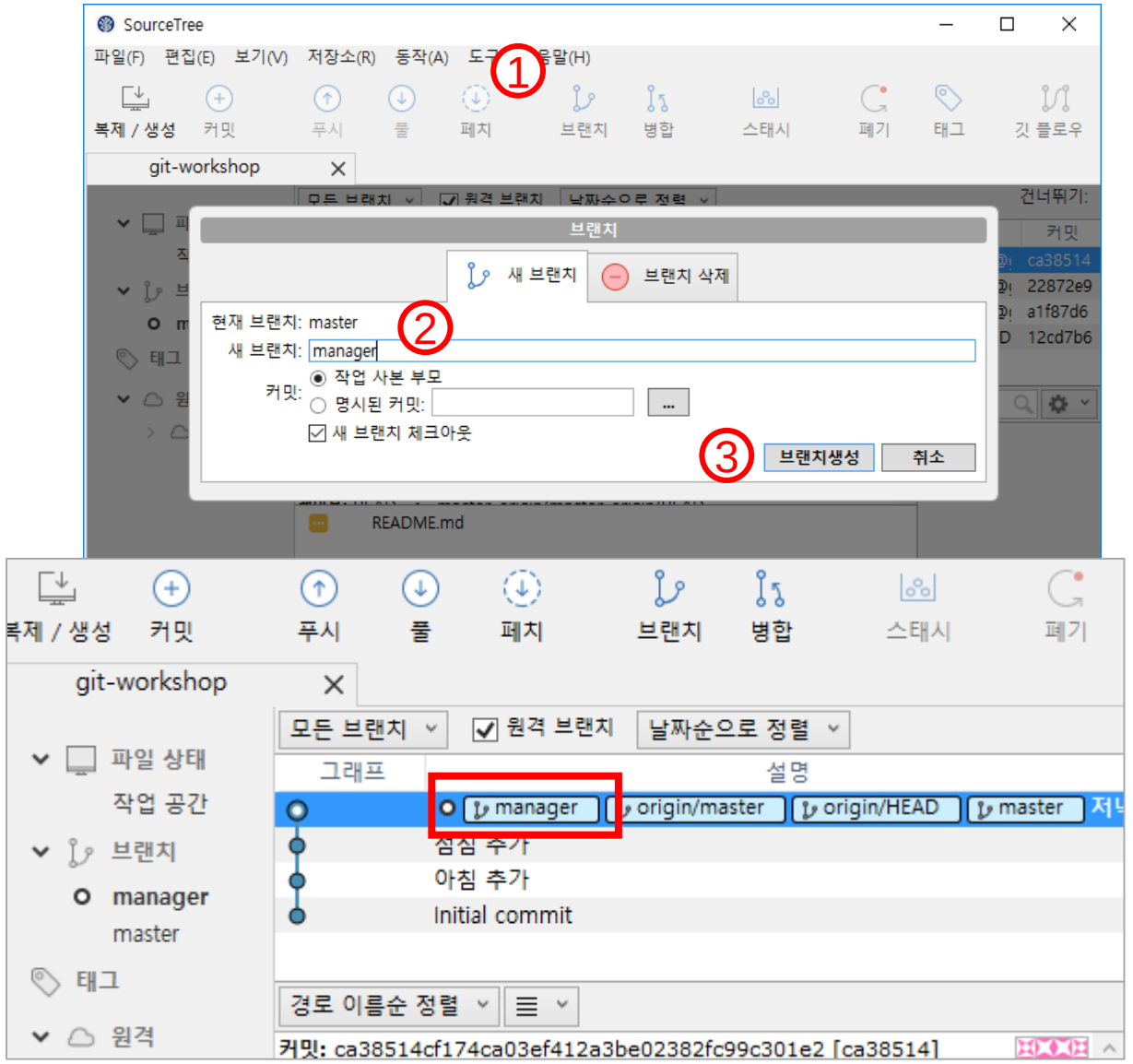
- 브랜치 개념
  - 마음대로 실험해 볼 수 있는 별도의 공간
  - 실험이 잘되면 원 브랜치에 합치고 아니면 버림
  - 공식 소스는 master 브랜치에
  - '깃 플로우'에서는 develop, release, hotfix 등으로 관리할 것을 권장
  - 각 사용자별 브랜치를 만들어 작업하다 해당 브랜치와 통합 하는 것이 좋음
- manager 브랜치 만들기
  - 이 실습에서는 관리자가 manager 브랜치를 만든 상황으로 실습

① SourceTree에서 브랜치 버튼 누르고

② 브랜치 창에서 새 브랜치 이름 입력하고

③ [브랜치생성] 버튼 누르면 완료

```
git branch manager
```

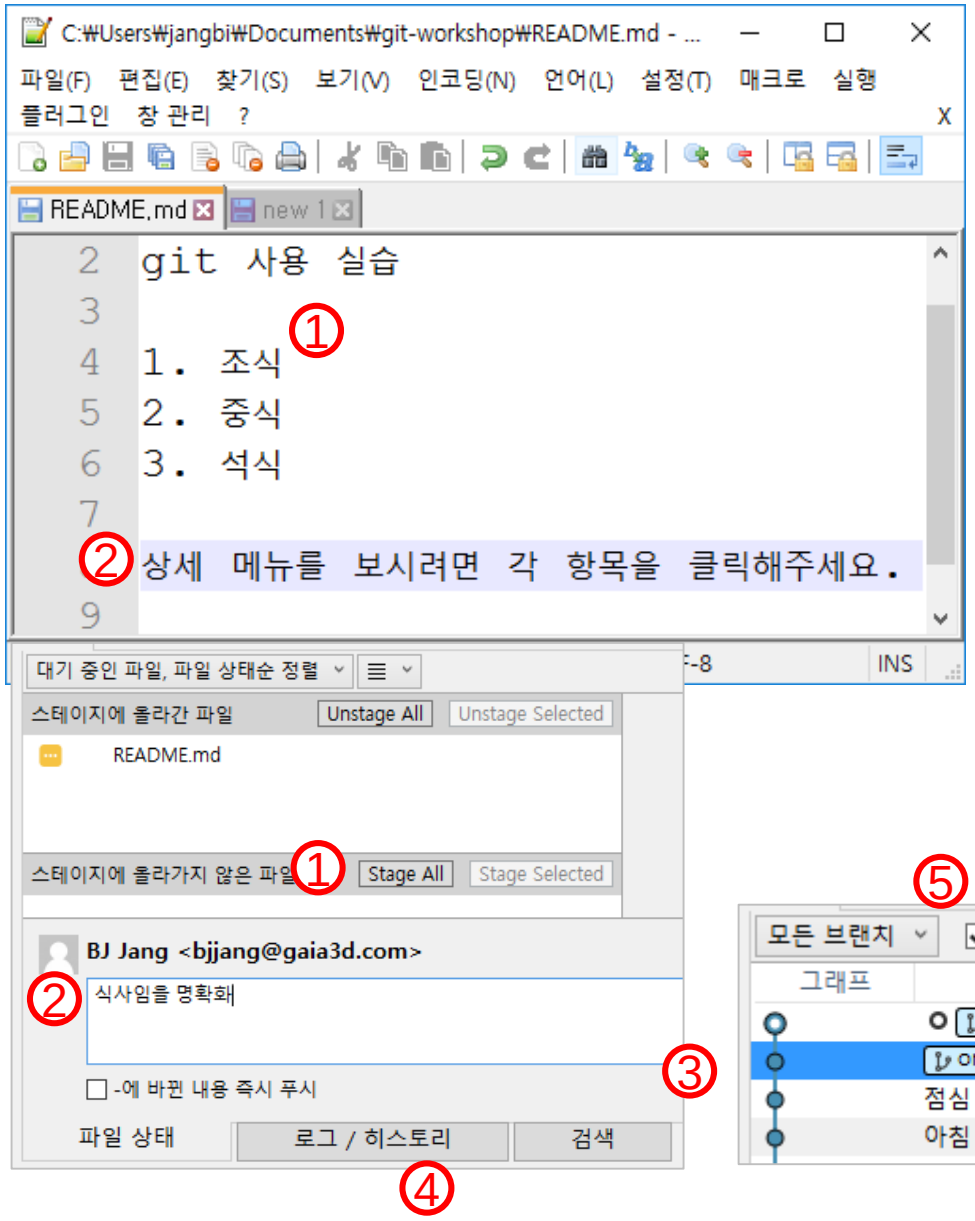


manager 브랜치에서 작업

Do it!

- README.md 수정
  - Notepad++에서
  ① 조식, 중식, 석식으로 용어 수정
  ② 잔소리 추가
- 커밋 만들기
  - SourceTree에서
  ① [Stage All]
  ② 커밋 메시지 입력하고
  ③ [커밋]
  ④ 로그 / 히스토리 탭 클릭
  ⑤ 생성된 로그를 그래프에서 확인

```
git log
```



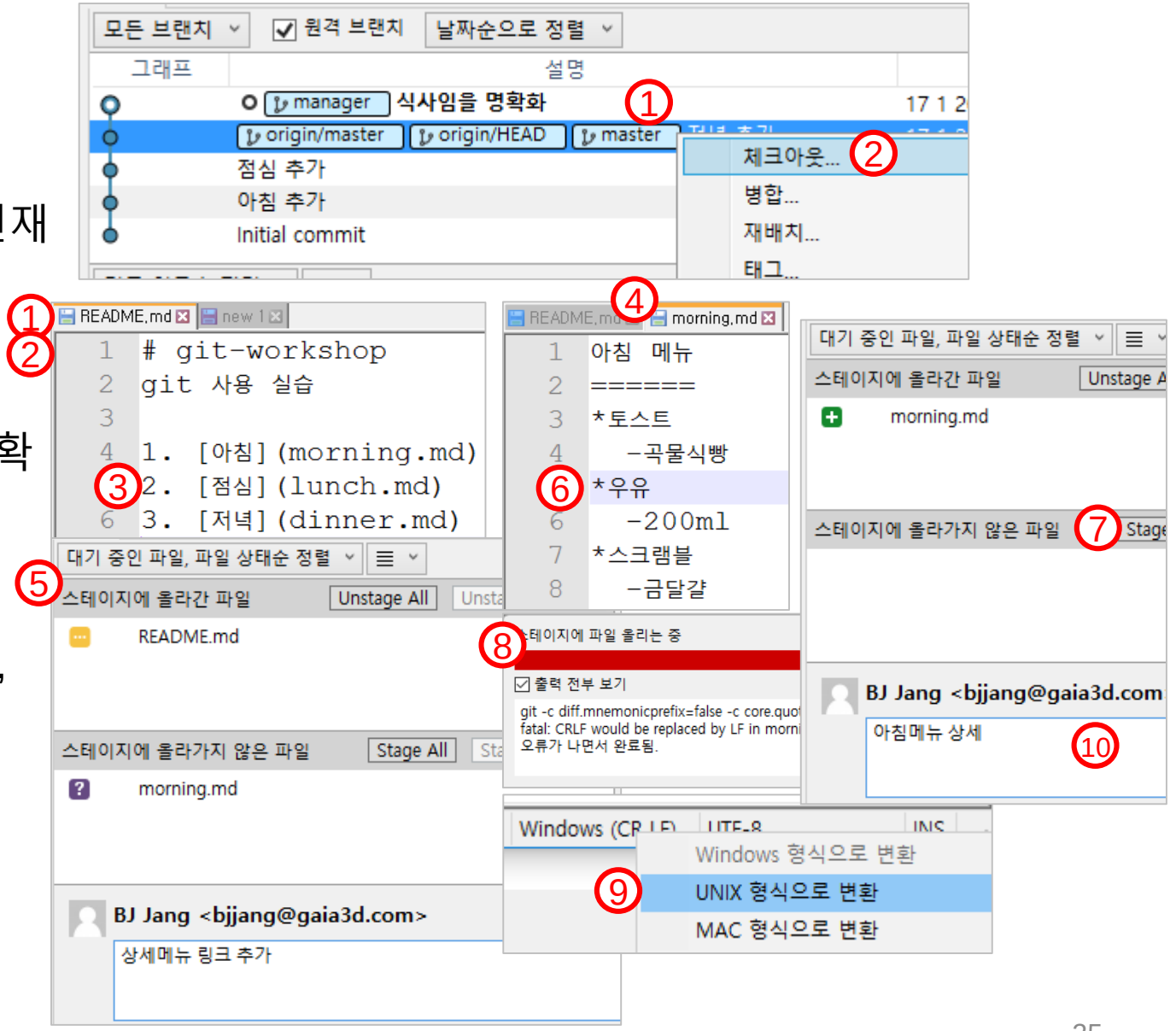
master 브랜치 발전시키기

Do it!

- master 브랜치로 전환
  - SourceTree의 로그/히스토리 탭에서
    - ① Master 브랜치 마크를 오른쪽 클릭하고
    - ② [체크아웃 …] 선택하여 master 브랜치로 현재 브랜치 전환

```
git checkout master
```

- master 브랜치 수정
  - ① Notepad++에서 README.md 보기
  - ② Manager가 수정하기 전 상태로 돌아온 것 확인
  - ③ 꺽쇄[]와 괄호()로 링크 생성
  - ④ Notepad++에서 morning.md 파일 생성
  - ⑤ SourceTree에서 README.md만 스테이지, 커밋
  - ⑥ morning.md 파일에 상세메뉴 기록
  - ⑦ SourceTree에서 스테이지 시도
  - ⑧ 오류가 난다면
  - ⑨ Notepad++ 에서 Unix 형태로 LF 변환
  - ⑩ SourceTree에서 다시 스테이지, 커밋

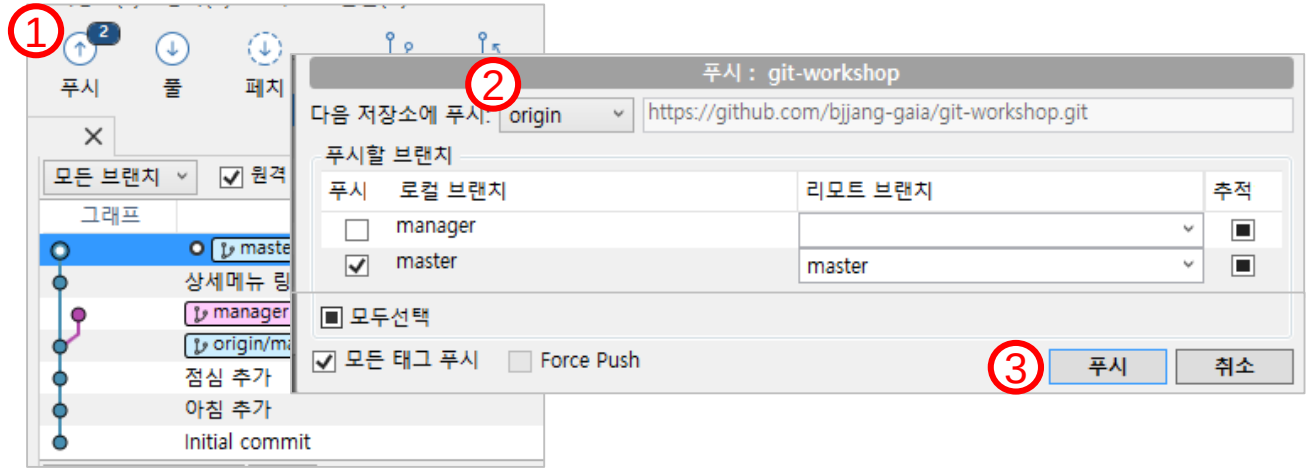


변경사항 Github에도 올리고 직접수정

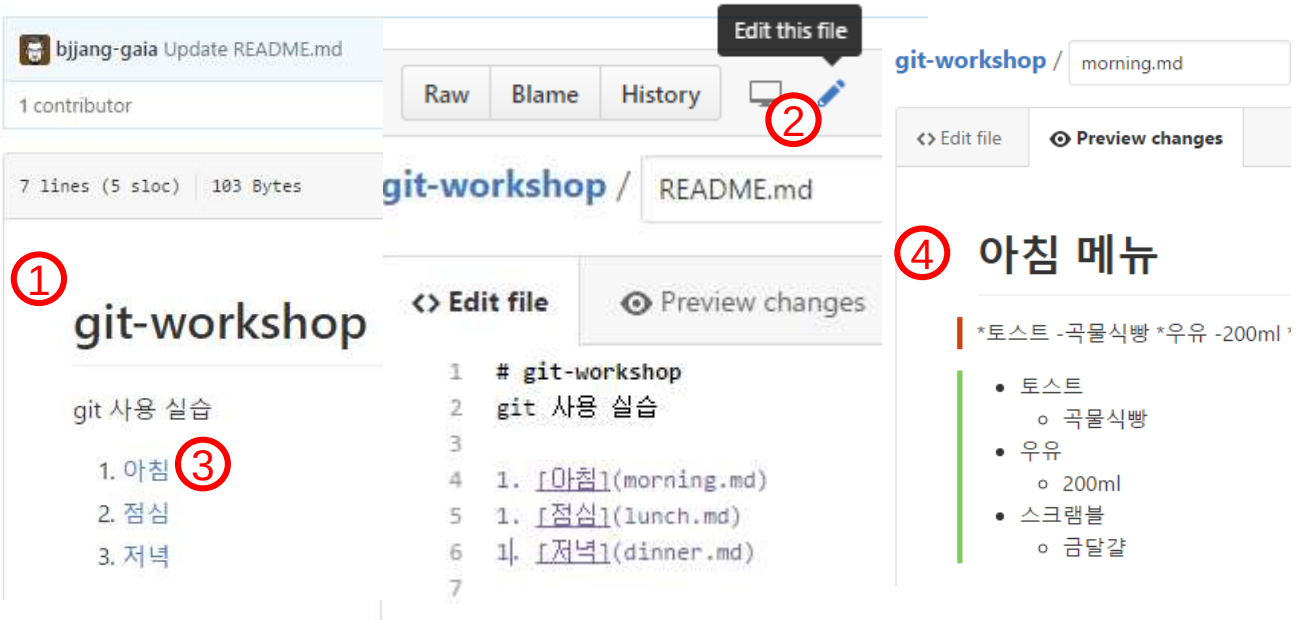
Do it!

- origin 저장소에 푸시
  - ① SourceTree에서 [푸시] 버튼
  - ② 대화상자에서 'origin' 선택된 것 확인
  - ③ [푸시] 하여 원격에 올리기

```
git push origin
```



- Github에서 확인/수정
  - ① md 파일이 원하는 형태로 보이는지 확인
  - ② 아니라면 [수정] 버튼 눌러 웹서서 수정
  - ③ 아침메뉴의 링크로 잘 이동하는지 확인
  - ④ 원하는 대로인지 확인하고 아니라면 수정



Github의 변경사항 로컬에 통합

Do it!

- Github에서 수정사항 받아오기
  - ① SourceTree에서 [페치] 버튼

    ```
    git fetch
    ```

  - ② Origin/master가 앞서짐 확인
  - ③ [풀] 버튼

    ```
    git pull origin master
    ```

  - ④ 깃허브의 변경사항이 로컬에도 반영되었음 확인

[참고] 충돌이 발생할 수도 있습니다.

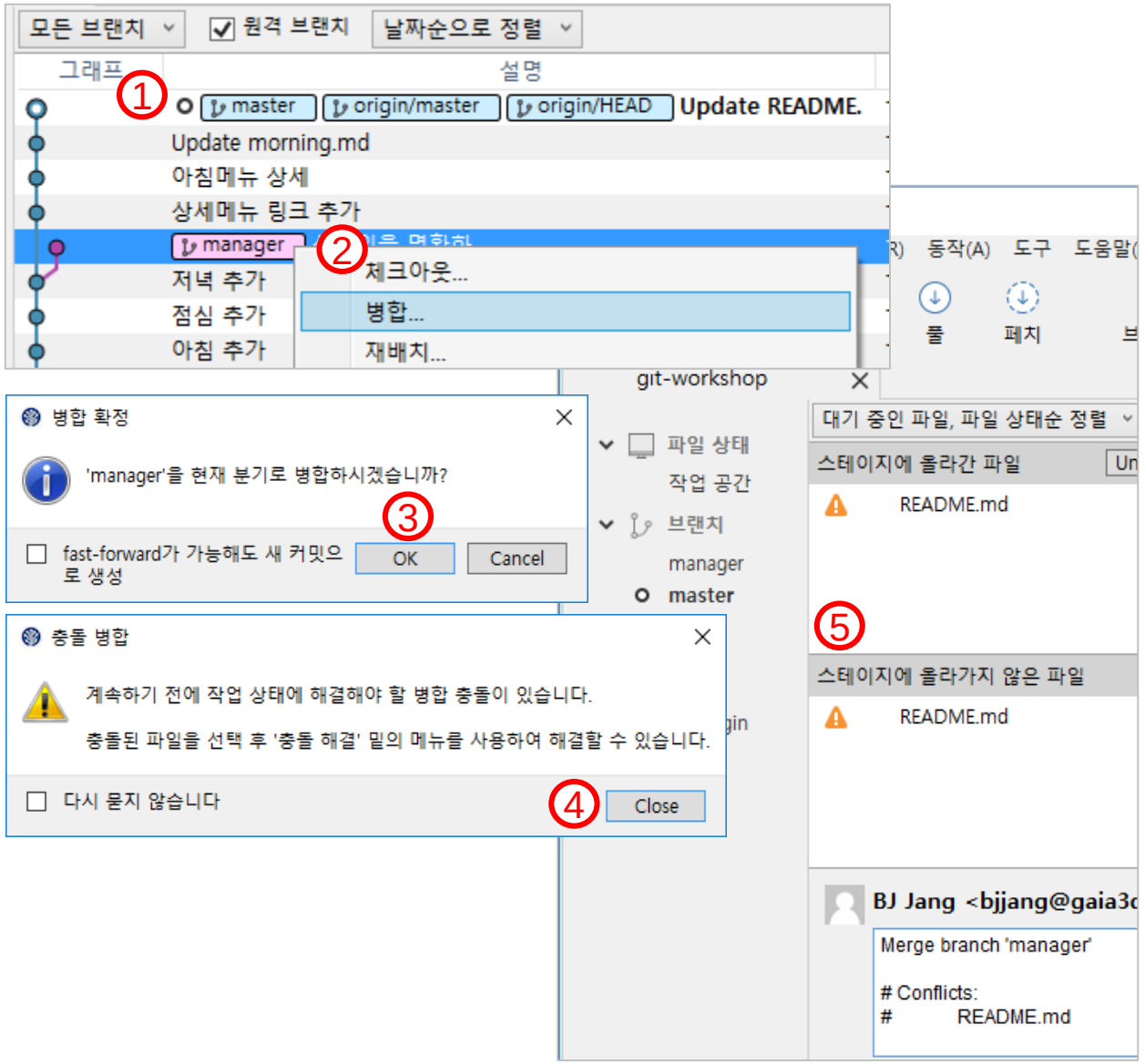
manager 브랜치 병합

Do it!

- master 브랜치에 manager 브랜치 병합
  - ① SourceTree에서 현재 master 브랜치임 확인
  - ② manager 브랜치 오른쪽 클릭하고 [병합]
  - ③ 병합 확정 대화상자에서 [OK]
  - ④ 충돌 병합 대화상자 확인
  - ⑤ 충돌이 된 상황 확인

```
git merge manager
```

원격 저장소의 변경사항 병합과 브랜치 병합은 거의 유사하게 동작함



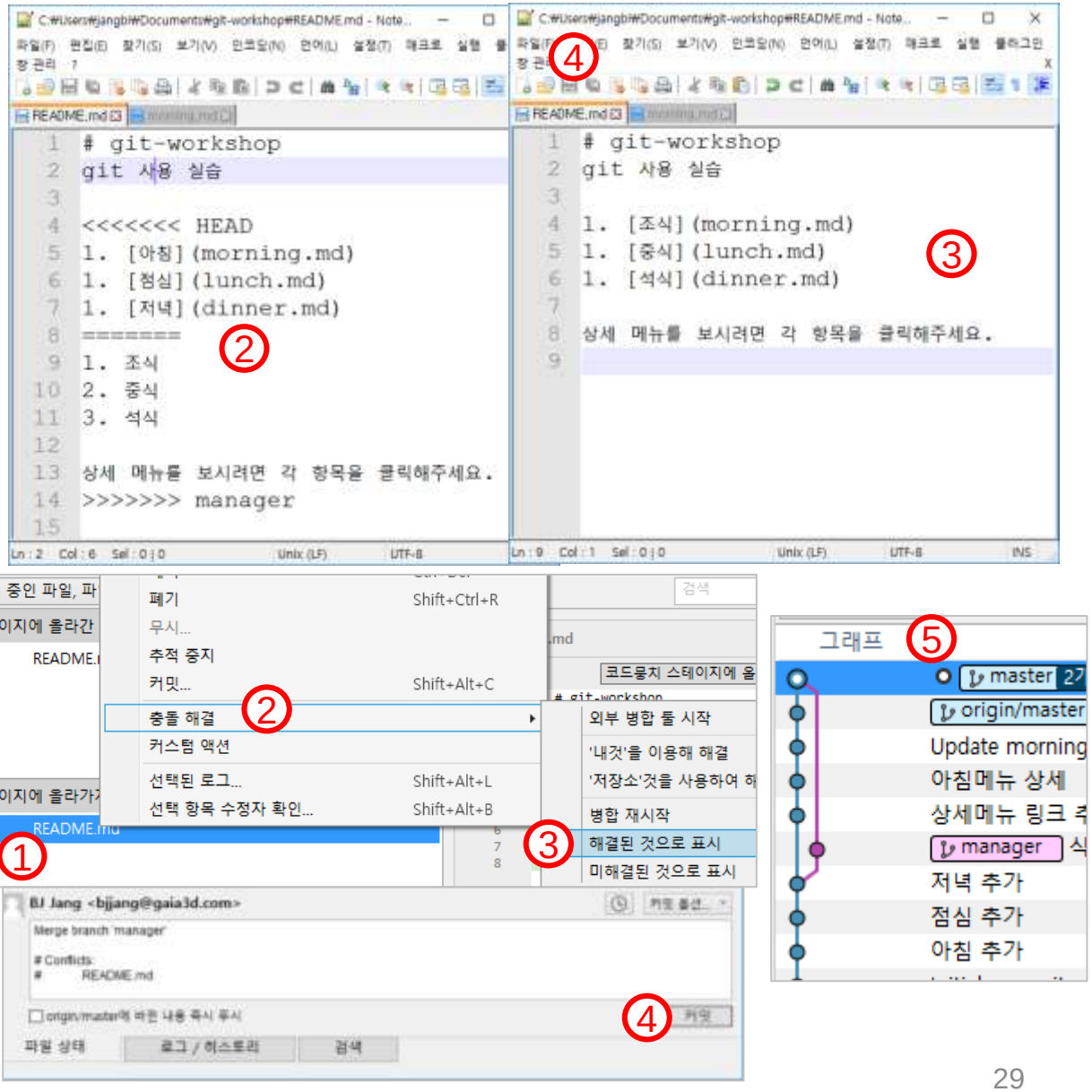
충돌 해결

Do it!

- 에디터를 이용한 충돌 해결
  - ① Notepad++에서 README.md를 열어
  - ② =====을 기준으로 로컬(<<<<)과 원격 (>>>>)의 변경사항이 보임 확인
  - ③ 이를 잘 판단하여 올바르게 정리
  - ④ 저장
- 충돌 해결로 표시
  - ① SourceTree에서 느낌표가 있는 READMW.md를 오른쪽 클릭하고
  - ② [충돌해결]
  - ③ [해결된 것으로 표시] 선택
  - ④ 커밋 메시지를 확인하고 [커밋]
  - ⑤ 히스토리에서 병합된 것을 확인

```
git commit -a -m "conflict solved"
```

[참고] Eclipse에서는 다음 링크처럼
http://blog.naver.com/PostView.nhn?blogId=lge920904&logNo=220274506449



변경사항 되돌리기

상황별 방금 한 동작 취소

View

- 마지막 commit 상태로 되돌리기

  `git checkout --force`

- 특정 파일을 최종 commit 상태로

  `git checkout README.md`

- (원격에 올리지 않은) 마지막 commit을 취소

  `git reset HEAD~` (commit 3개 취소하려면 `git reset HEAD~3`)

- 방금 한 commit의 메시지 바꾸기

  `git commit -amend`

- 실패한 merge를 취소

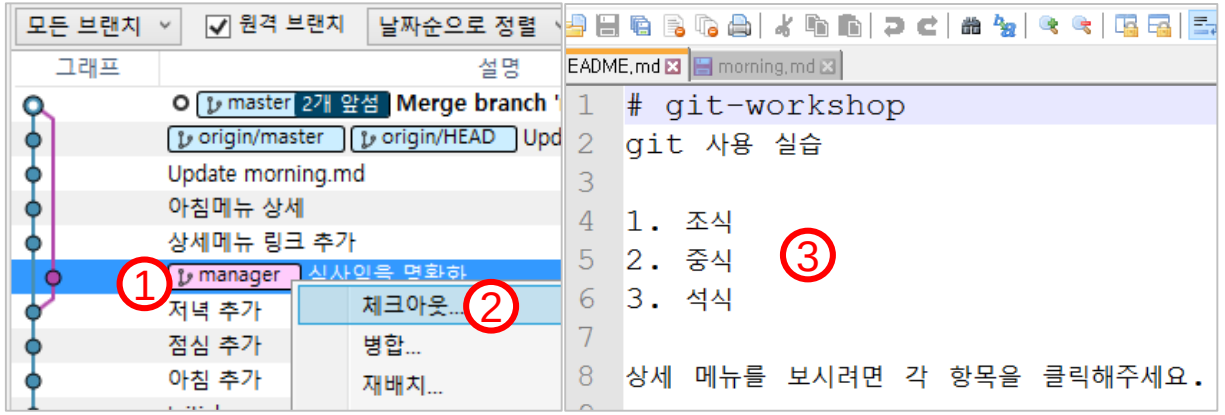
  `git merge --reset`

History 유지하며 특정 commit으로 돌아가기

Do it!

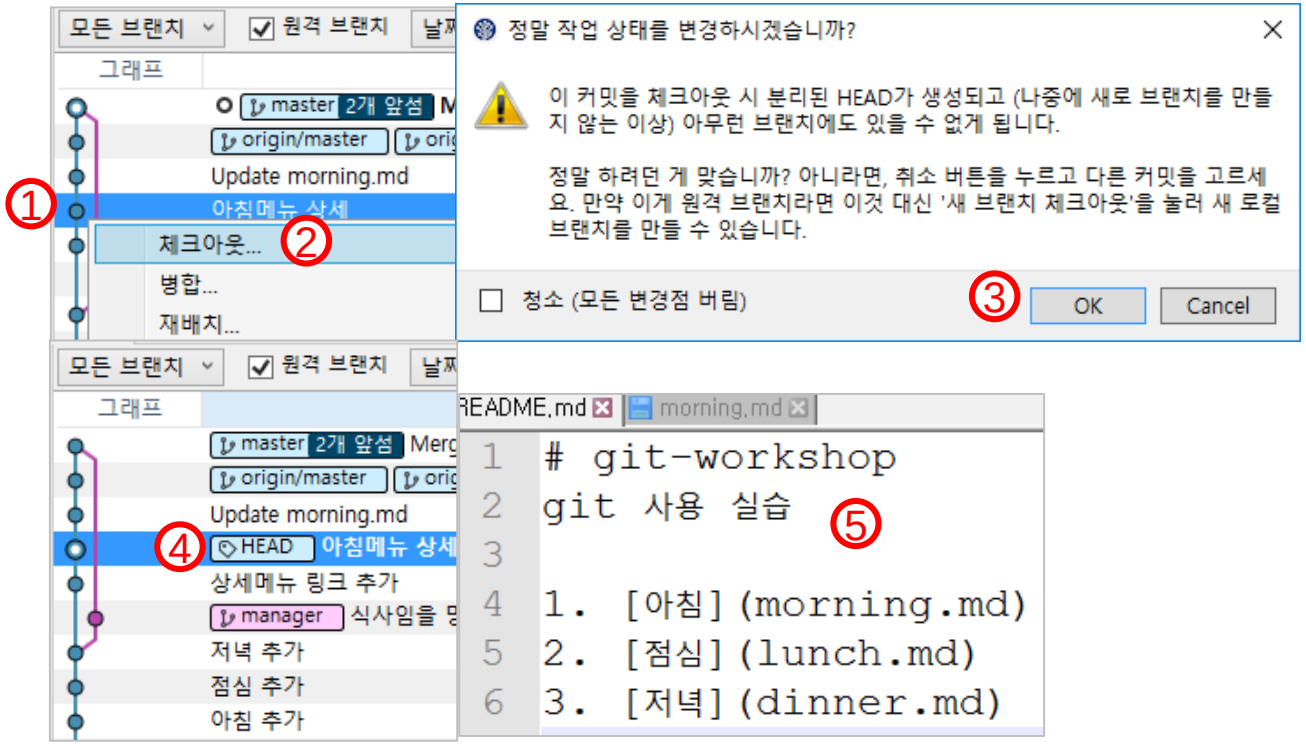
- 다른 브랜치로 가기
  - ① SourceTree에서 manager 브랜치 오른쪽 클릭
  - ② [체크아웃] 메뉴 선택
  - ③ Notepad++에서 변경된 것 확인

```
git checkout manager
```



- 특정 commit 상태로 전체를 되돌리기
  - ① SourceTree에서 '아침메뉴 상세' 커밋을 오른쪽 클릭
  - ② [체크아웃] 메뉴 선택
  - ③ 경고 확인하고 [OK]
  - ④ 선택한 커밋이 활성화되고 분리된 [HEAD]가 생김 확인
  - ⑤ Notepad++에서 변경된 것 확인

```
git checkout 51aaecf
(51aaecf는 특정 commit의 ID)
```



특정 커밋에서 새 브랜치 생성/삭제

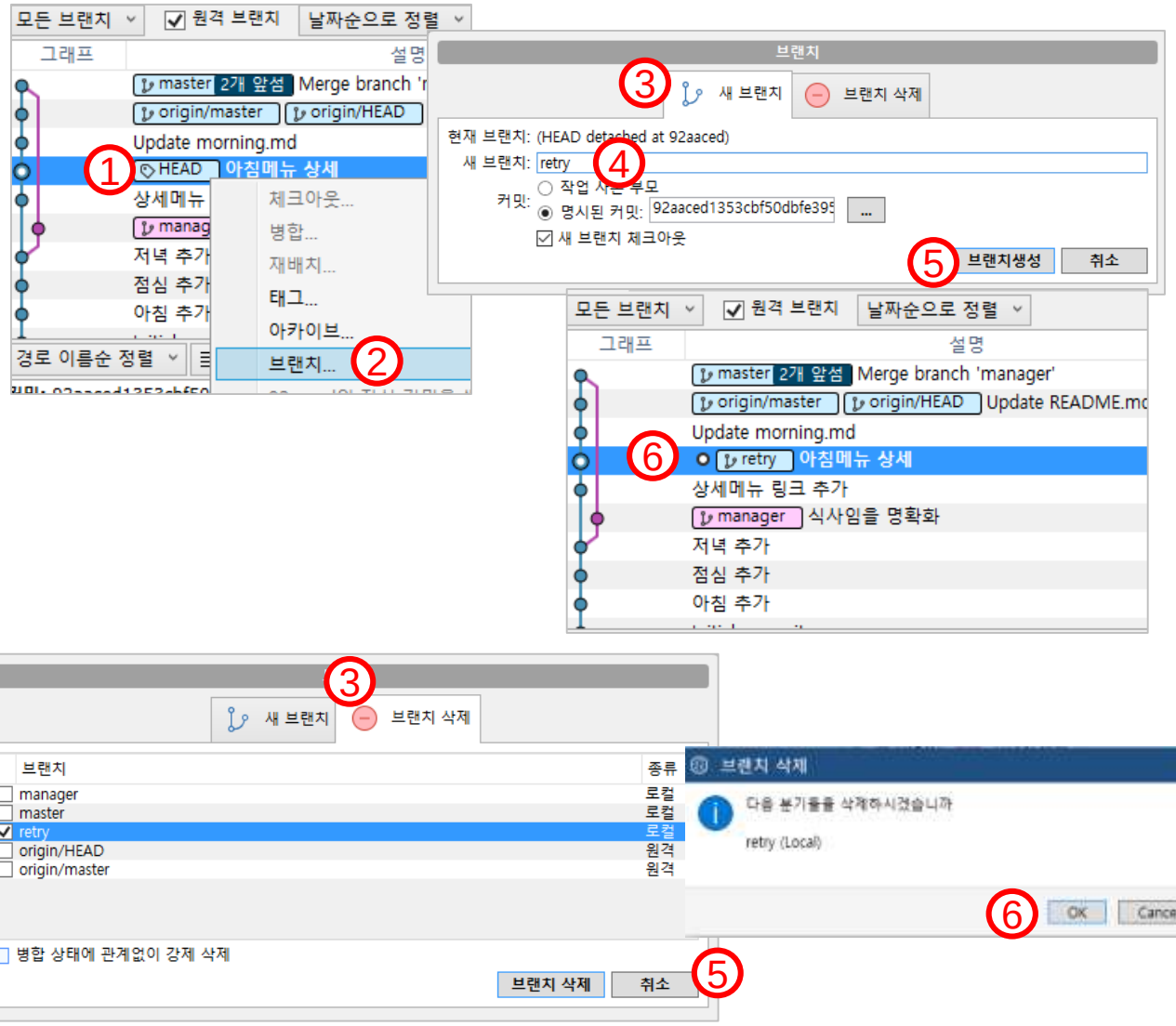
Do it!

- 분리된 HEAD에서 브랜치 생성
  - ① SourceTree에서 방금 만든 분리된 HEAD를 오른쪽 클릭
  - ② [브랜치] 메뉴 선택
  - ③ 브랜치 창에서 [새 브랜치] 탭 선택
  - ④ 새 브랜치 이름으로 'retry' 입력
  - ⑤ [브랜치생성]
  - ⑥ retry 브랜치가 생성 되었음 확인

```
git branch retry
```

- retry 브랜치 삭제
  - ① retry 브랜치 오른쪽 클릭
  - ② [브랜치] 메뉴 선택
  - ③ 브랜치 창에서 [브랜치 삭제] 탭 선택
  - ④ retry 브랜치 체크
  - ⑤ [브랜치 삭제]
  - ⑥ 경고 확인하고 [OK]

```
git branch -d retry
```



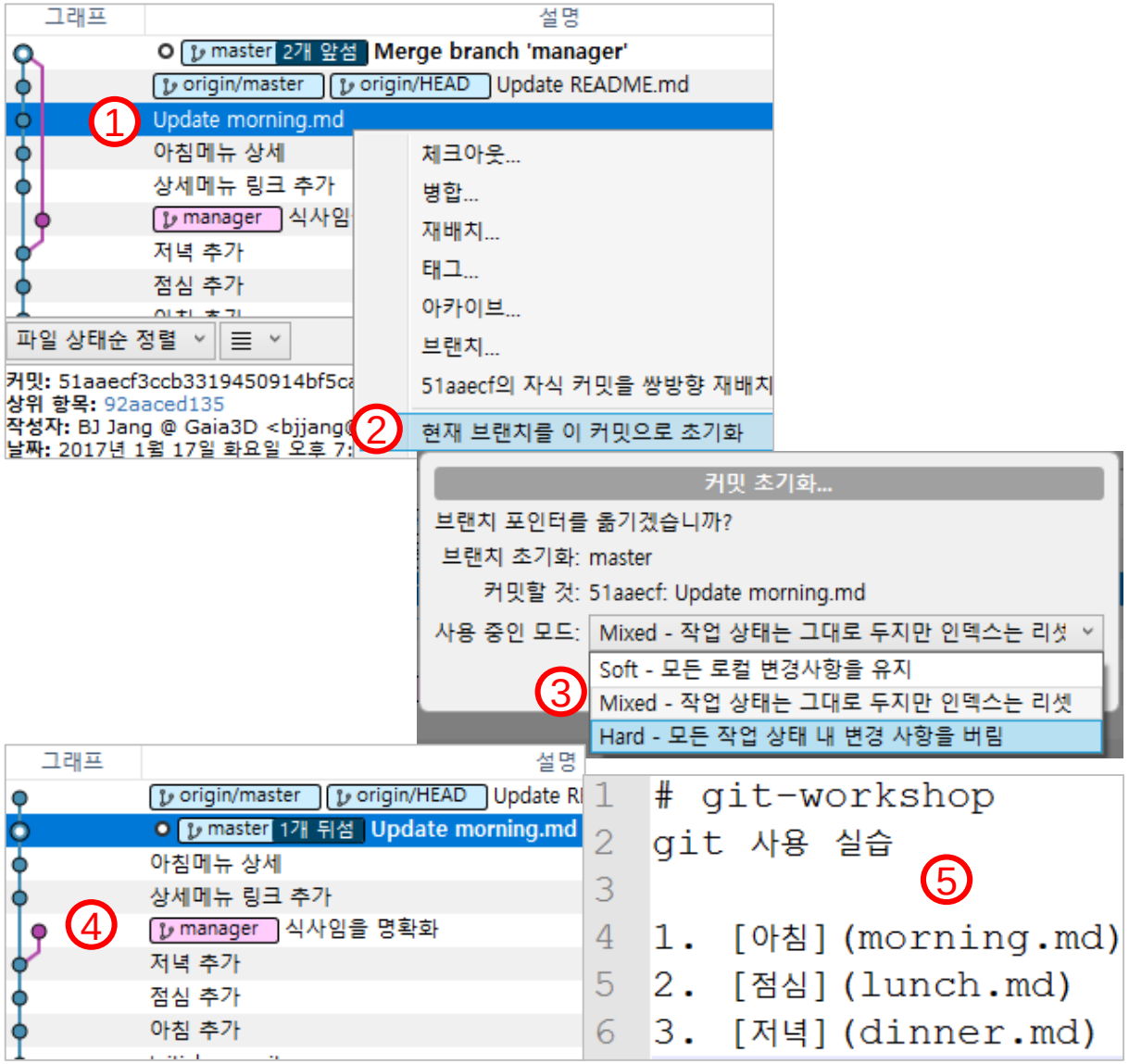
History와 Working Directory까지 모두 되돌리기

Do it!

- 특정 commit으로 모두 되돌리기
  - ① SourceTree에서 'Update morning.md' 커밋 오른쪽 클릭
  - ② [현재 브랜치를 이 커밋으로 초기화] 메뉴 선택
  - ③ '커밋 초기화' 창에서 Hard 모드 선택하고 [확인]
  - ④ master 브랜치가 선택한 커밋이 현재상태로 변경되었음 확인 (merge 되었던 것도 사라짐)
  - ⑤ Notepad++에서 이전 상태로 파일이 돌아가 있음을 확인

```
git reset 51aaecf --hard
```

과거의 상황을 확인해보려면 체크아웃(checkout)이, 완전히 버리고 되돌리려면 초기화(reset)가 적합



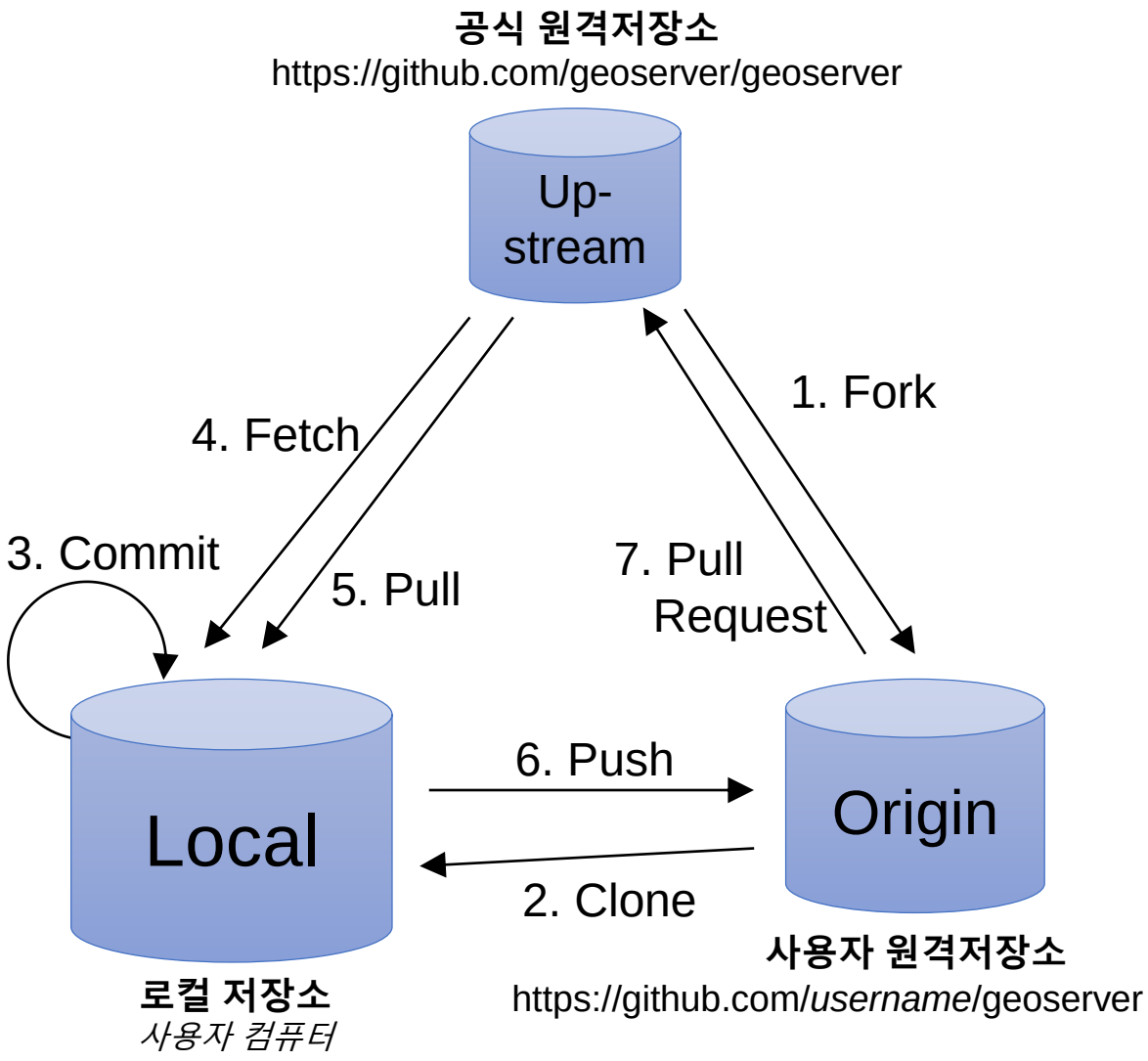
오픈소스 참여 위한 저장소 구성

권장 원격저장소 구성 및 저장소 간 동작

View

- 3가지 저장소
  - **Upstream**: 각 오픈소스의 **'공식 원격저장소'**
  - **Origin**: 공식 저장소를 **'사용자 원격저장소'**에 복사해 지속적으로 내 변경을 올리는 곳
  - **Local**: 내 컴퓨터에 있는 **'로컬 저장소'**
- 저장소 간 동작
  1. **Fork**: 공식 원격저장소를 사용자 계정에 복사
  2. **Clone**: 사용자 원격저장소를 내 컴퓨터에 복사
  3. **Commit:** 사용자가 개발한 내용을 로컬 저장소에
  4. **Fetch**: 공식 원격저장소의 변경이력을 받아옴
  5. **Pull**: 공식 원격저장소의 변경을 로컬 저장소의 브랜치에 반영
  6. **Push**: 로컬 저장소의 변경이나 공식 원격저장소에서 받아온 변경을 사용자 원격저장소에 반영
  7. **Pull Request**: 사용자 원격저장소에 올린 내용을 공식 원격저장소에 반영해줄 것을 요청

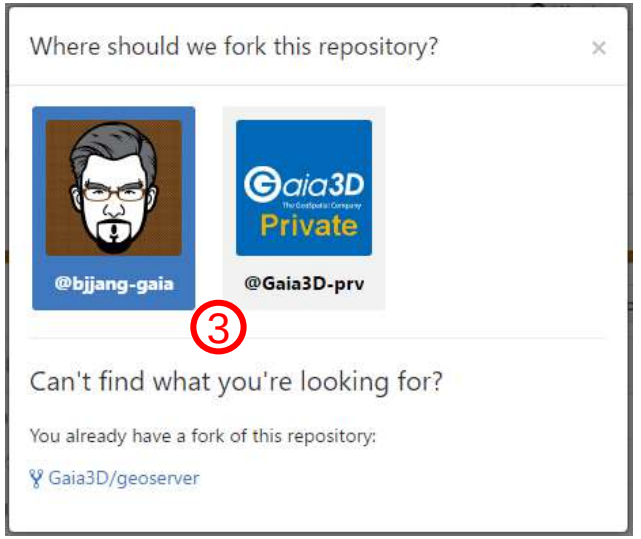
일반적으로 각 프로젝트의 Committer가 아닌 경우 변경사항을 공식 원격저장소에 직접 반영할 권한이 없어 Pull Request를 이용해야만 함



공식 원격저장소 Fork

Do it!

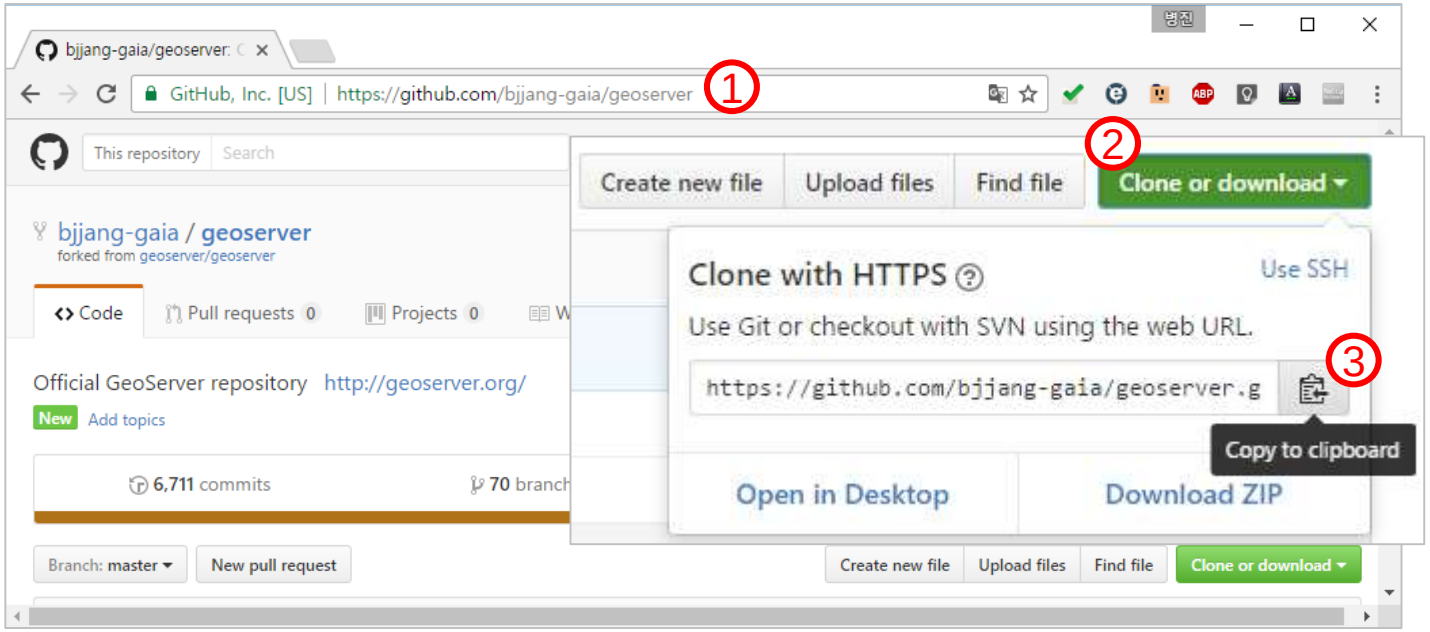
- Fork란?
  - Github에서 제공하는 기능
  - 공식 원격저장소를 내 계정에 복사해 오는 기능
  - 사용자 원격저장소는 임의로 수정 가능
  - 원 소스 수정 혹은 확장개발시 필요
- Fork 과정
  ① 웹브라우저에서 github의 각 프로젝트공식 저장소로 이동 (https://github.com/geoserver/geoserver)
  ② [Fork] 버튼 클릭
  ③ 사용자가 속해있는 그룹이 있다면 개인 개정과 그룹개정 중 선택
  ④ Fork가 완료되면 사용자 계정의 저장소 페이지로 이동됨

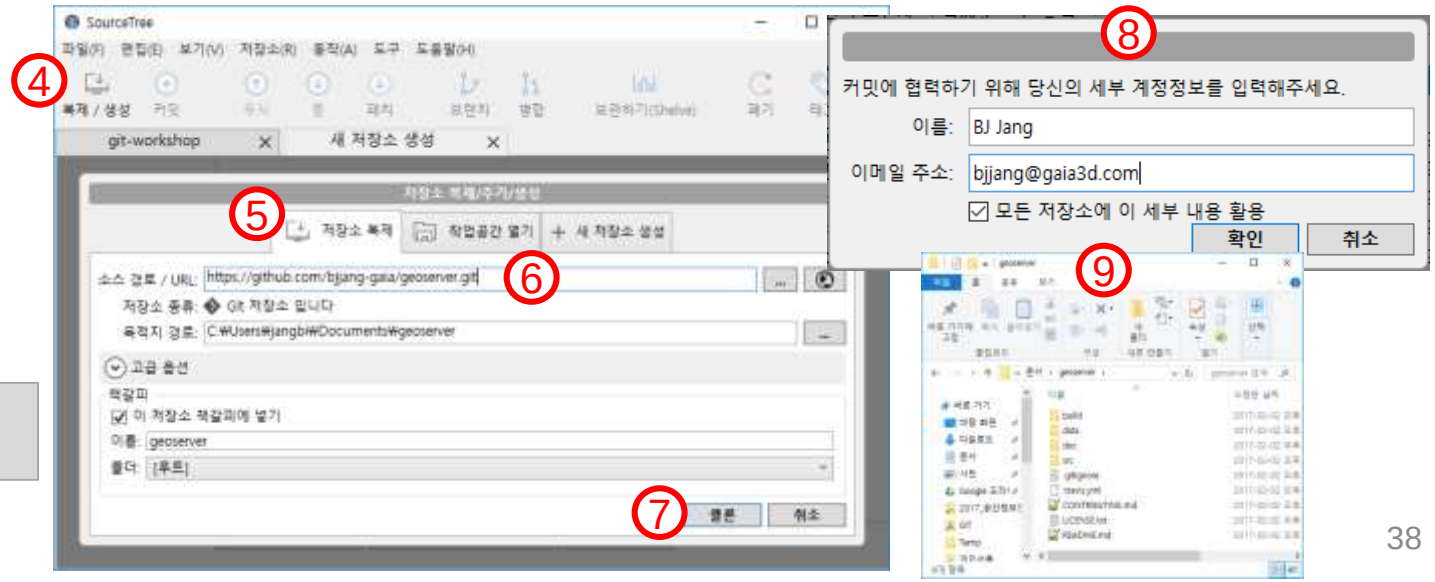


사용자 원격저장소에서 clone

Do it!

- 사용자 원격저장소를 사용하는 이유
  - 공식 원격저장소에서 바로 Clone 하면 권한이 없어 내가 변경한 내용을 올릴 수 없다.
  - 사용자 원격저장소는 브랜치처럼 활용 가능
- 사용자 원격저장소 Clone 과정
  - ① 웹브라우저에서 사용자 원격저장소 URL로 이동
  - ② [Clone or Download] 버튼 누르고
  - ③ 저장소 접근 URL을 [Copy to clipboard] 눌러 복사
  - ④ SourceTree에서 [복제/생성] 버튼
  - ⑤ '저장소 복제' 탭 선택
  - ⑥ 소스경로에 복사해둔 URL 붙여넣기
  - ⑦ [클론]
  - ⑧ (필요시) 사용자정보 입력 후 [확인]
  - ⑨ 클론이 진행되어 PC의 사용자 폴더 아래의 Documents\geoserver 폴더에 소스가 들어와 있음 확인





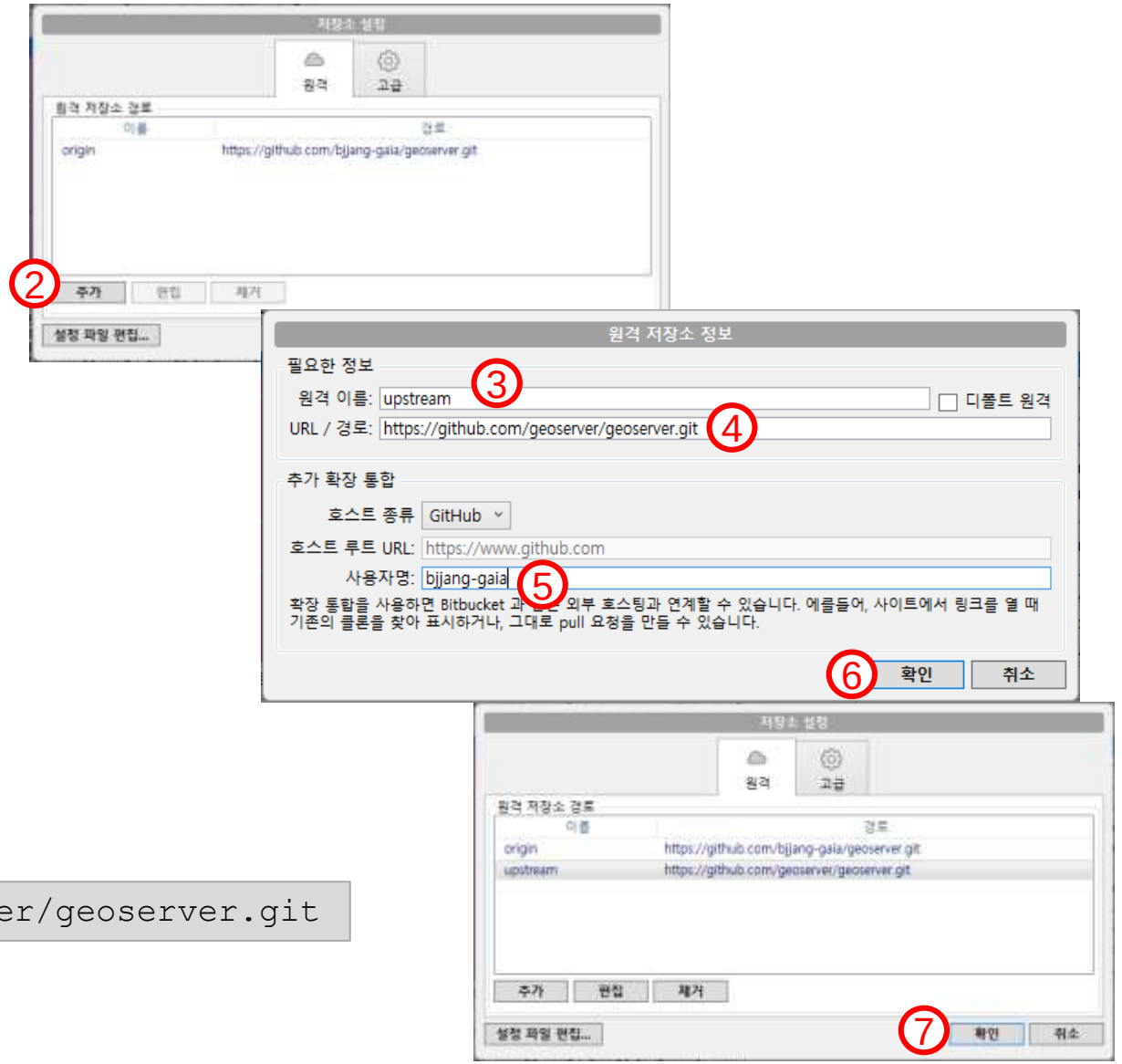

```
git clone git://github.com/사용자 계정/geoserver.git
cd geoserver
```

upstream 원격 저장소 추가

Do it!

- 왜 원격 저장소가 2개?
  - 내가 개발을 진행하는 동안에도 커뮤니티에서 계속 개발이 진행됨
  - 공식소스의 변경사항은 공식 원격저장소인 upstream에서 받아와야 함
  - 내가 개발한 내용은 사용자 원격저장소인 origin에 올림
- 공식 원격저장소 추가
  - 사용자 원격저장소인 origin은 github에서 clone하면 기본적으로 추가되어 있음

① SourceTree에서 [저장소]-[원격 저장소 추가] 메뉴
② '저장소 설정' 창에서 [추가] 버튼
③ '원격 저장소 정보' 창에서 원격 이름에 'upstream'
④ URL경로에 공식 원격저장소 경로 입력
⑤ 사용자명에 github 계정(혹은 이메일) 입력
⑥ [확인]
⑦ '저장소 설정' 창에서 추가된 것 확인하고 [확인]

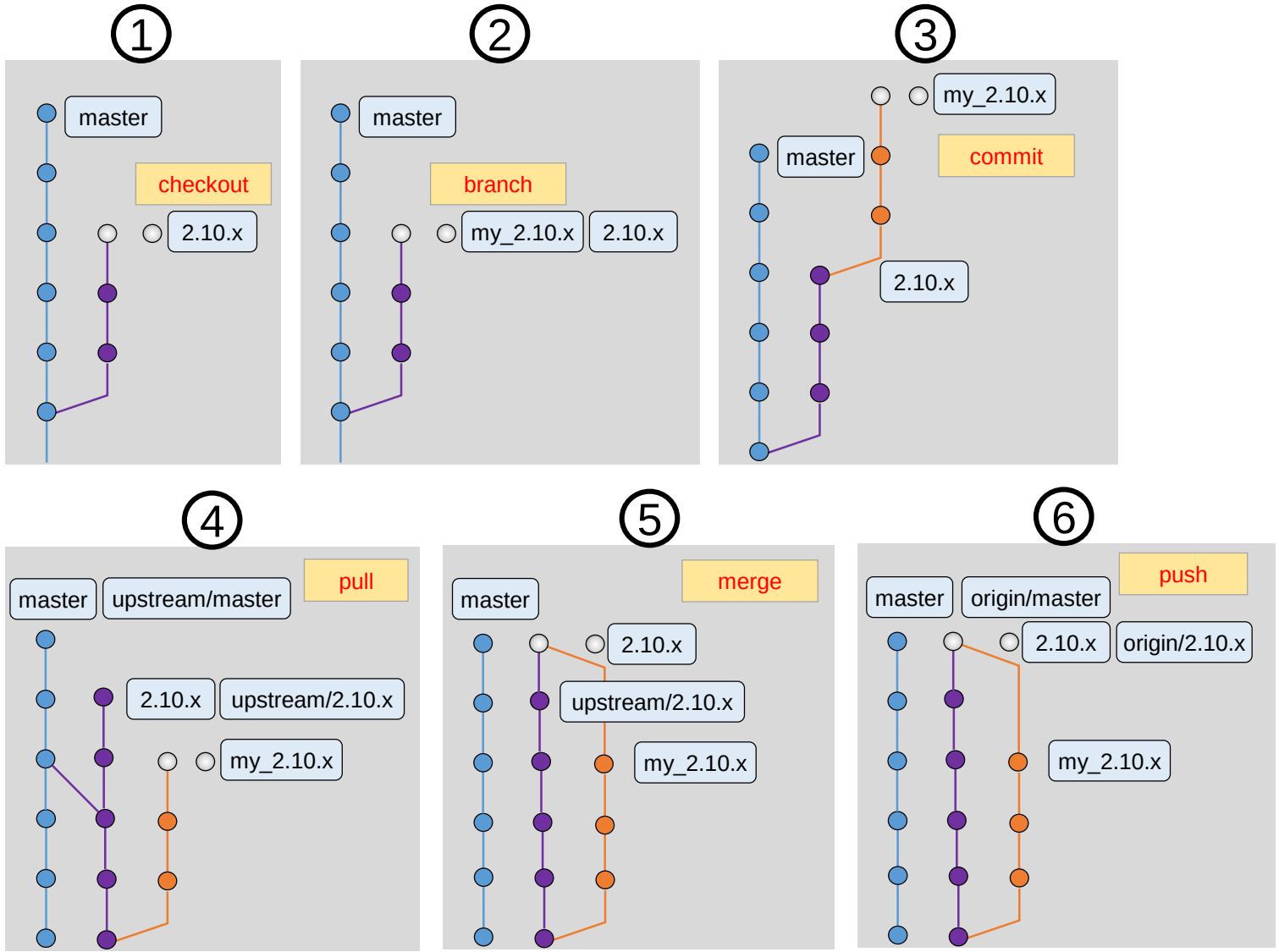



```
git remote add upstream git://github.com/geoserver/geoserver.git
```

원 소스 기반으로 개발하기

View

- 어떤 때 원 소스를 기반으로 개발?
  - 보통 플러그인이나 확장모듈을 개발
  - 버그수정이나 개선시에는 원 소스 변경
- 원 소스 수정 과정
  - ① 원 소스의 브랜치 중 내 개발의 기반이 되는 브랜치로 checkout
  - ② 기존 브랜치에서 나만의 브랜치 생성
  - ③ 나만의 브랜치에 사용자가 원하는 방향으로 변경 개발하여 지속적으로 commit
  - ④ 개발 완료시 기존 브랜치의 변경사항을 Upstream에서 받아 합치고
  - ⑤ 나만의 브랜치를 기존 브랜치에 merge
  - ⑥ 변경사항을 사용자 원격인 origin에 push
  - ⑦ 변경사항이 공식 원격저장소에 반영되도록 Pull Request 생성

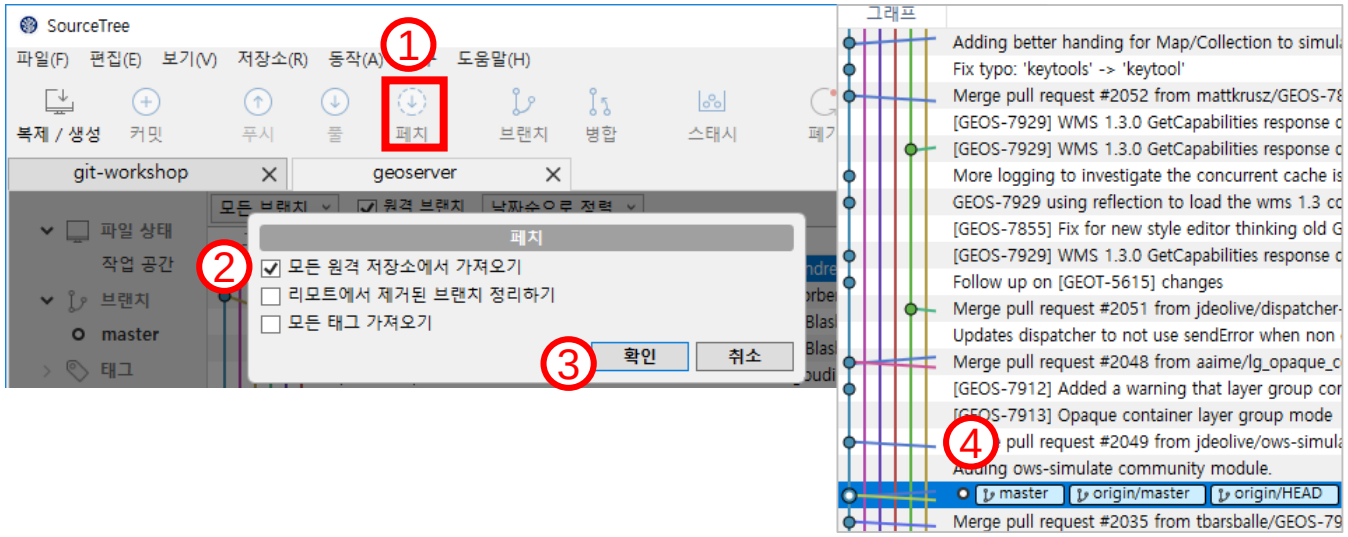


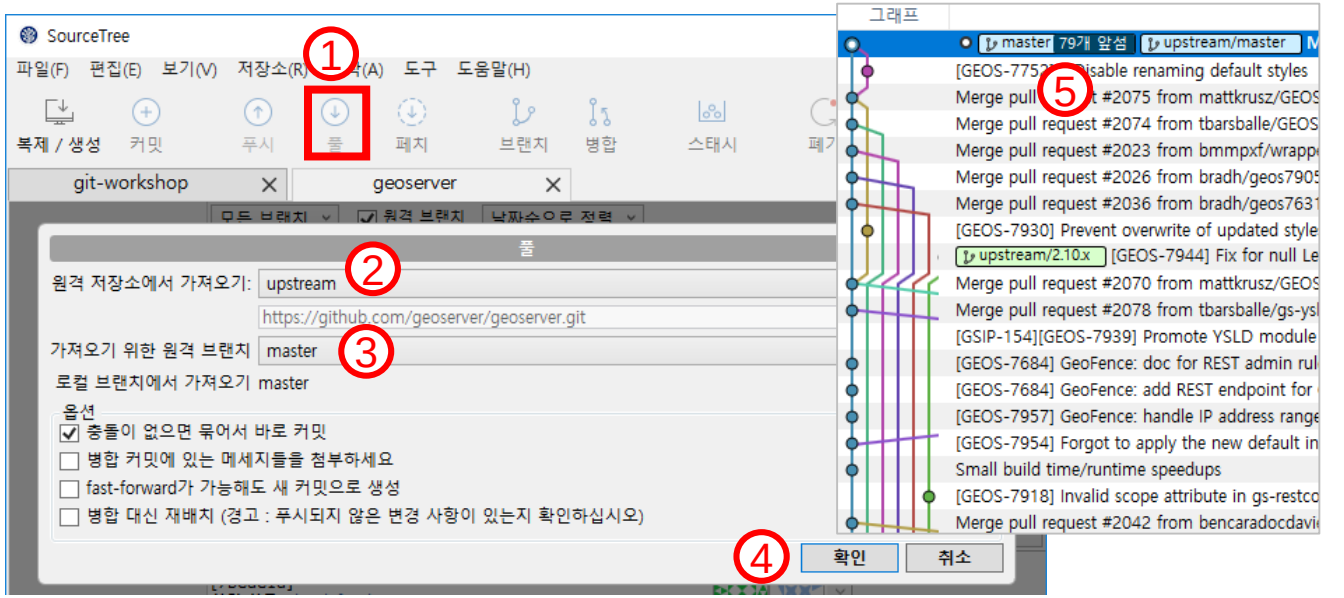
공식 원격저장소의 변경이력 받기, 반영하기

Do it!

- upstream에서 변경이력 받아오기
  - ① SourceTree에서 [패치] 버튼
  - ② '패치' 창에서 '모든 원격 저장소에서 가져오기' 체크
  - ③ [확인]
  - ④ 그래프에서 'master' 위로 이력이 늘어남 확인

  [참고] 패치는 변경이력을 받아오지만 현재 로컬의 소스를 변경하지는 않는다.
- upstream에서 변경이력 받아와 합치기
  - ① SourceTree에서 [풀] 버튼
  - ② '풀' 창에서 '원격 저장소에서 가져오기'에 'upstream' 선택
  - ③ '가져오기 위한 원격 브랜치'에 master 선택
  - ④ [확인]
  - ⑤ 그래프에서 'master'와 'upstream/master'가 같아짐 확인
- 주의사항
  - fetch나 pull을 수행하기 전에 내 변경사항을 commit 해야 한다.
  - pull을 하는 과정에서 충돌이 발생할 수 있다.
  - 일반적으로 fetch로 이력을 가져와 확인하고 pull을 하는 것이 좋다.

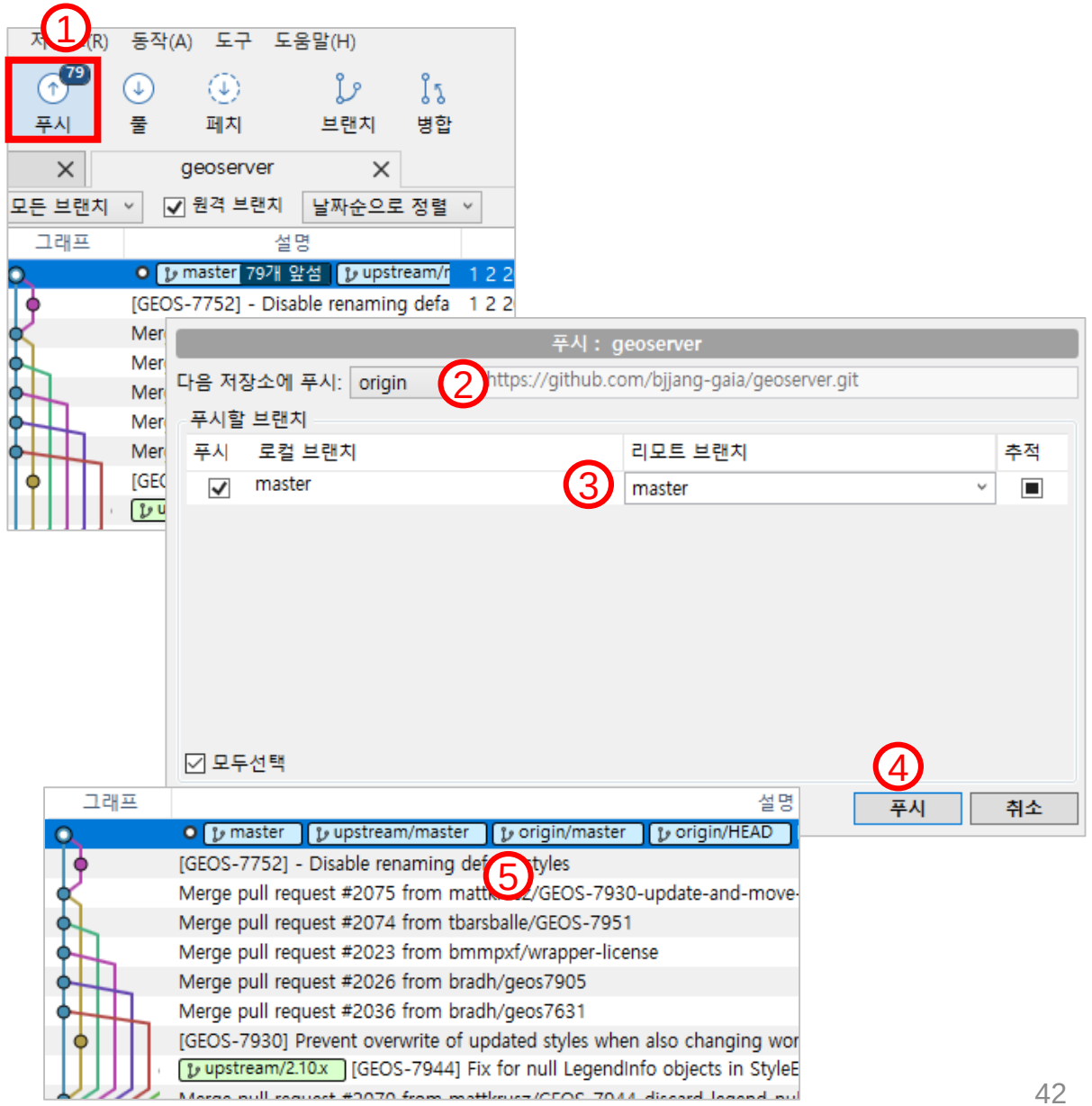




사용자 원격저장소에 변경사항 올리기

Do it!

- 사용자 원격에 올리기가 필요한 경우
  - 사용자가 소스를 수정하고 commit을 한 경우
  - upstream에서 받아온 이력을 merge 한 경우
  - origin에 upstream의 변경을 반영
- 사용자 원격저장소에 올리기
  - ① SourceTree에서 [푸시] 버튼
  - ② ‘푸시’ 창에서 ‘다음 저장소에 부시 ‘ 항목에 ‘origin’ 선택
  - ③ 로컬 브랜치 중 원격에 올릴 소스 브랜치와 원격의 대상 브랜치 선택
  - ④ [푸시] 버튼
  - ⑤ 그래프에서 ‘origin/master’가 ‘master’와 동일해 졌음을 확인



내 변경사항을 공식 저장소에 반영요청

Do it!

- Pull Request 가 필요한 경우
  - 오픈소스의 원 소스를 고친 경우
  - 내가 변경한 소스가 다른 사람에게도 의미가 있는 경우 (개선, 버그수정 등)
- 반영요청 과정
  ① 내가 변경한 내용을 담은 브랜치(my_2.10.x)를 원 브랜치(2.10.x)에 merge
  ② SourceTree에서 [저장소]-[Pull 요청 생성] 메뉴
  ③ (필요한 경우) Github 로그인 정보 입력
  ④ 'Pull 요청 생성' 창에서 반영요청 할 로컬 브랜치와 원격 브랜치 선택
  ⑤ [웹에서 pull 요청 생성] 버튼
  ⑥ 웹브라우저에서 생성될 구체적인 Pull Request 확인 (차이, 충돌여부)
  ⑦ Pull Request를 최종적으로 등록