형상관리도구

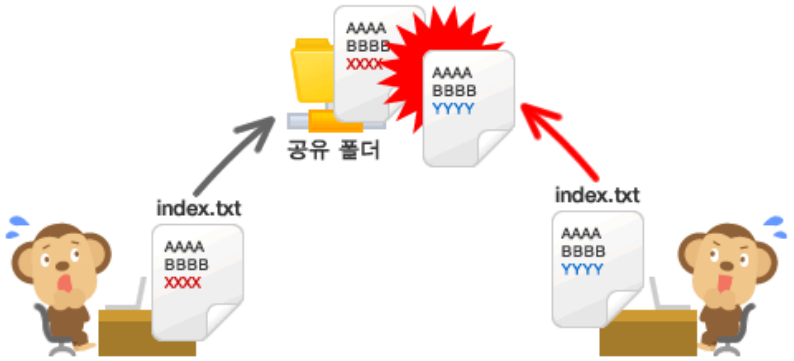
GIT의 정의 및 활용

애플리케이션 배포

이동준

목 차

형상 관리 도구

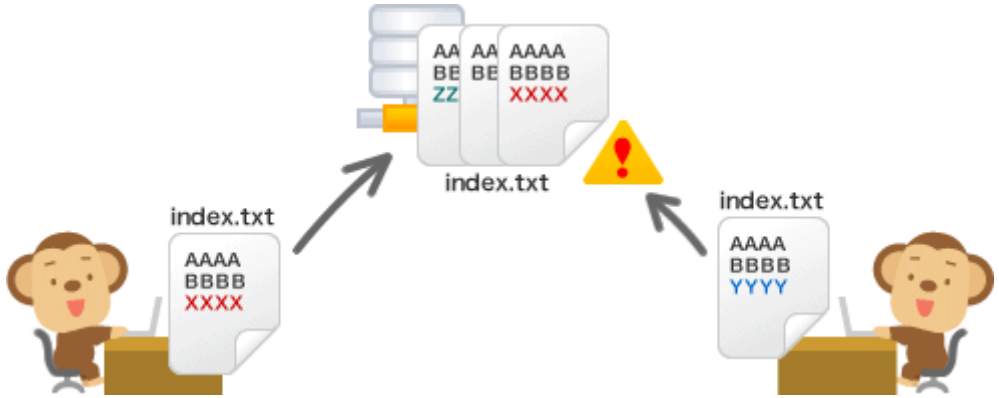


- 형상이란?
  - 파일의 변경 내역 및 현재의 형태에 대한 정보
  - 파일의 형상 정보는 파일 그 자체보다 용량이 훨씬 적음
- 파일의 변경 이력을 관리하는 도구
  - 대표적인 형상 관리 도구 : Git과 SVN
- 형상 관리 도구가 필요한 이유

  1. 각 파일의 히스토리를 편하게 관리하기 위함
  2. 다수의 이용자가 공유파일을 사용시 수정 충돌을 방지하기 위함

GIT이란?

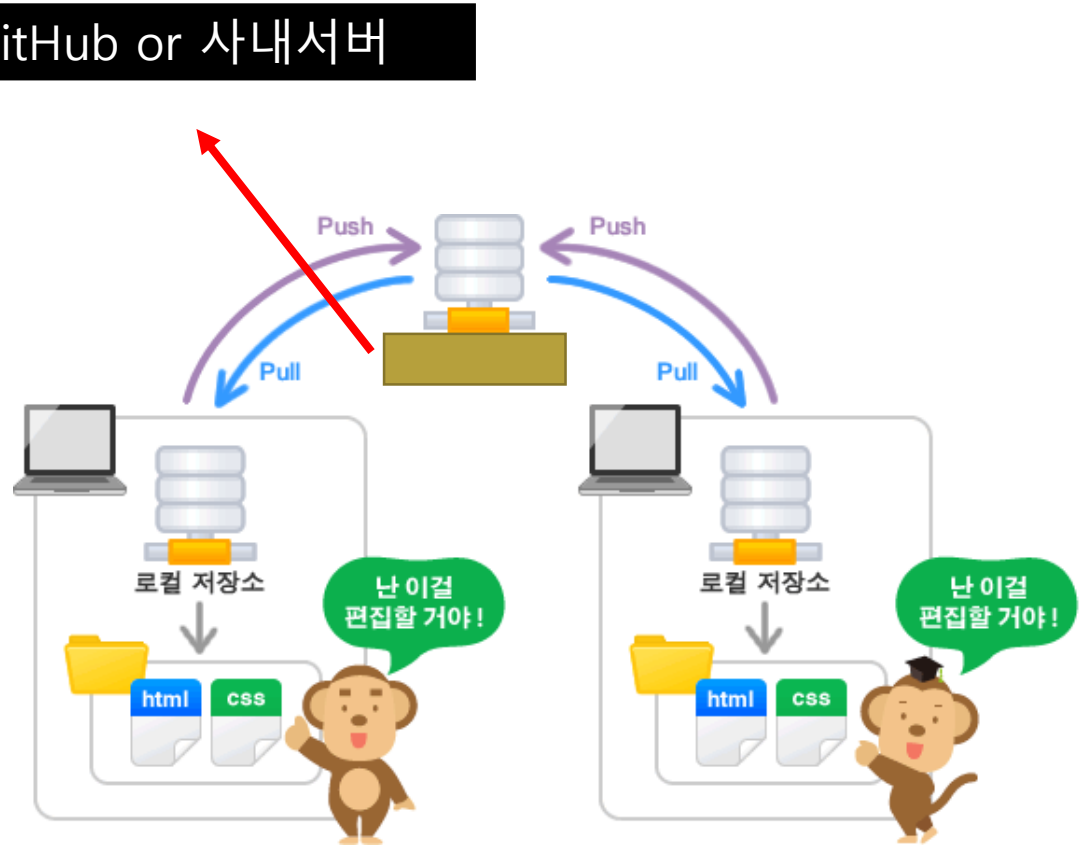
- 형상 관리 도구의 하나로써, 파일의 모든 이력을 관리하는 저장소 또는 서비스
- 리눅스의 개발자 리누즈 토발즈가 리눅스 소스코드의 형상을 관리하기 위하여 만듦
  - 원래 다른 형상 관리 툴을 썼는데, 그게 구려서 만든 게 자기가 2주만에 직접 만듦
- SVN 등의 형상 관리 툴과 라이벌이었으나 최근 독보적 입지를 확보하게 됨
  - Git 자체의 편리함도 있지만 GitHub의 영향이 매우 크다고 볼 수 있음



파일의 변경 이력도 효과적으로 관리되며,
여러 사용자가 동시에 수정할 경우 충돌을 방지함

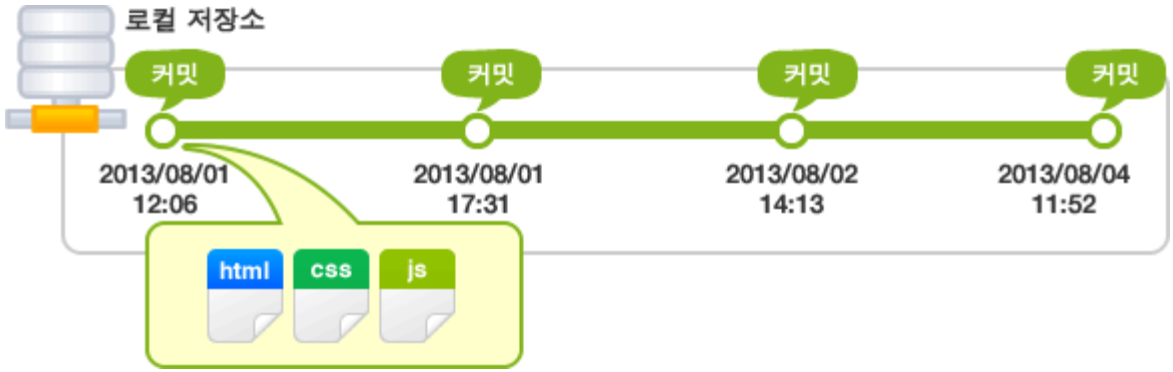
GIT과 GITHUB

- GitHub 사이트에 저장소를 두고, 해당 저장소에 있는 파일들을 가지고 Git 서비스를 이용하게 해주는 서비스
- GitHub 서비스 없이 사내 서버의 공유 폴더에 오직 Git만 설치하여, 해당 파일의 형상을 관리하기도 함
    - 이럴 경우 사내 네트워크에 접속되어 있을 때에만 해당 저장소에 접근이 가능함



핵심개념 : COMMIT, PUSH

- Commit(커밋) : 파일의 변경사항을 기록하며, 로컬저장소에만 우선적으로 적용됨



- Push : 원격저장소에 Commit 내용을 적용

※ Git 말고 GitHub 바로 설치해서 Commit & Push할 수도 있다.

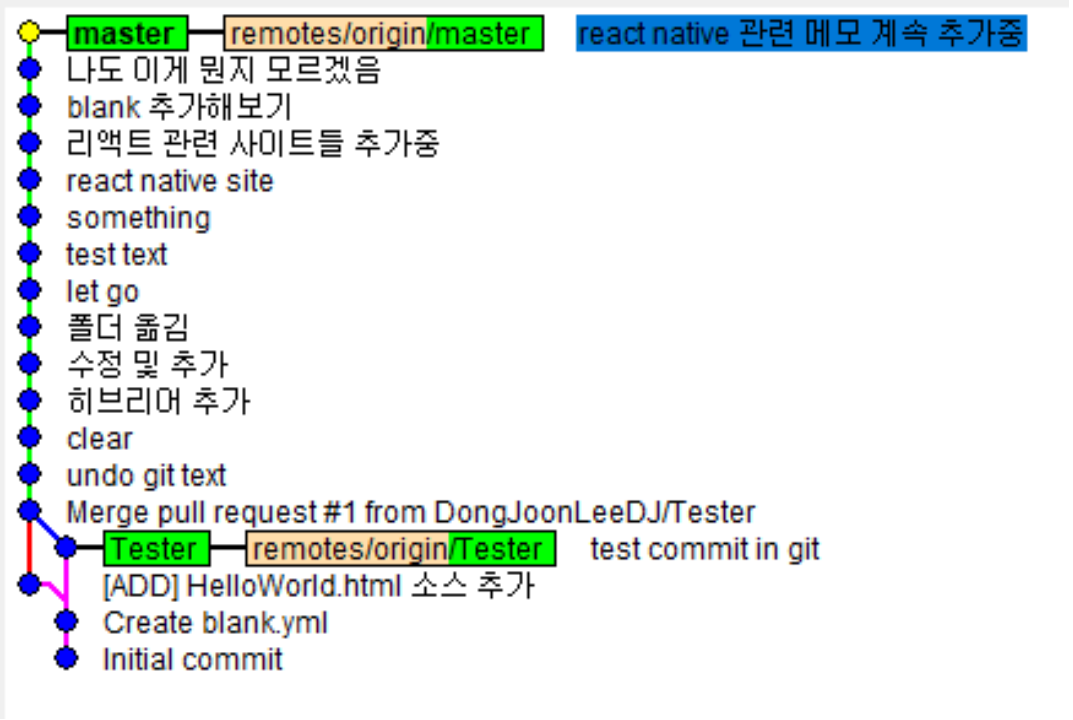
핵심개념 : PULL, BRANCH

- PULL : 원격저장소의 특정 branch의 파일들을 다운로드함

  ※ Clone : 원격저장소의 master branch 파일 및 다른 branch들의 형상정보들을 다운로드함

  ※ 형상 정보 : 원격저장소에서 파일다운로드시 모든 branch에 있는 파일들을 다운로드하지 않고 그 파일들의 형상 정보만을 다운로드한다. Clone을 통하여 다운로드하였을 경우에는 다른 branch의 내용으로 변경이 가능함

- Branch : 일종의 분기점으로써, 파일을 버전별로 유지보수 하다가 분기를 나누고 싶을 때 Branch를 추가할 수 있음
  - 예시) master로 작업하다가 Tester라는 분기를 만든 경우이며, Tester와 master는 서로 독립적임



※ 기본 Branch 명
Git : master
GitHub : main
본래는 동일하였으나 흑인 인권 문제 등의 논란때문에 바뀌었음
그래서 Git을 통하여 GitHub 접근시 이 부분(기본 브랜치명)에 주의를 기울여야 한다.

기본 브랜치 변경

- https://blog.outsider.ne.kr/1503
- 인권 문제때문에 이런 저런 용어들이 제거가 되었는데 그 중에 하나가 master/slave이다.
  - 우선은 Git 버전을 2.28버전이상으로 올린다. 즉, 최신 버전을 설치한다.

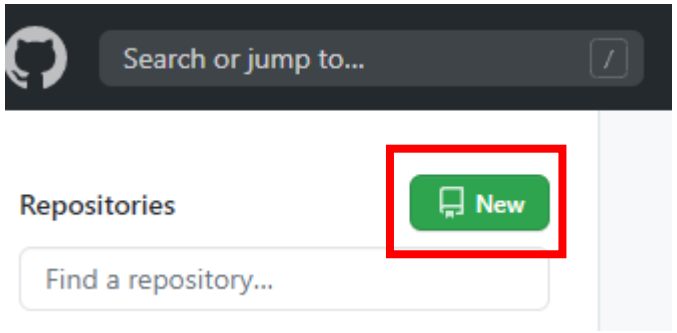
```bash
$ git version
git version 2.28.0
```

`git config --global init.defaultBranch main`으로 `init.defaultBranch`를 설정한다. 전역 설정으로 지정했으므로 이는 `~/.gitconfig`에 아랫부분이 추가된다.(직접 `~/.gitconfig`를 수정해도 된다.)

GITHUB 가입 및 저장소 등록[1/3]

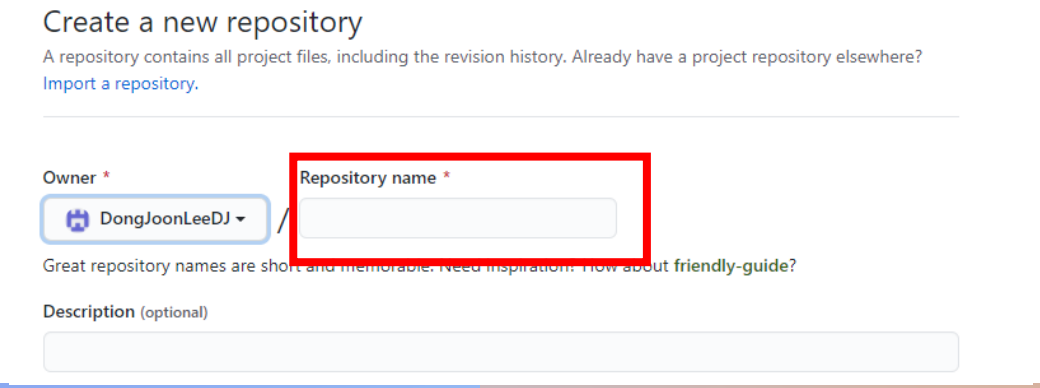
1. GitHub 접속 및 회원가입 (https://github.com/)

   E-mail로 가입하고,
   E-mail로 로그인하며,
   E-mail 꼭 메모해두기



2. Repositories 옆 New 클릭

3. 저장소 이름 지정

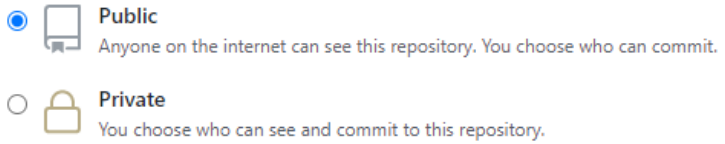


GITHUB 가입 및 저장소 등록[2/3]

4. Public, Private 지정

   Public : 다른 사람들이 내 저장소에 접근 가능

   Private : 자신 및 허용된 사람들만 저장소에 접근 가능



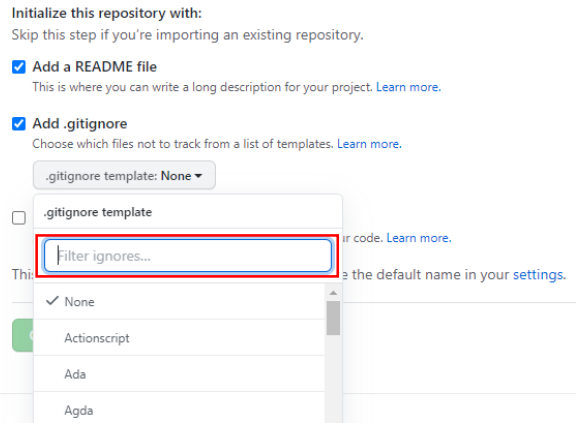
5. 이 부분은 전부 체크하지 않아도 상관은 없음

   1. README file : 저장소에 readme 파일이 생성됨
   2. .gitignore 파일 추가 여부

      => 예를 들어서 java를 입력하면 java를 빌드할 때

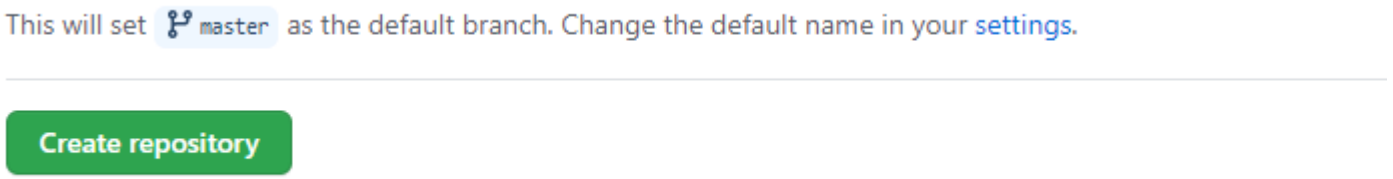
      발생되는 불필요한 파일들에 대해서는 commit&push가

      되지 않고 .java 소스와 같은 유효한 파일들만 commit&push됨



GITHUB 가입 및 저장소 등록[3/3]

Create repository를 클릭하여 저장소 생성



※ GitHub를 사용하지 않는 경우에 저장소 추가 방법은 서버 담당자나 선임 개발자에게 물어볼 것
필자의 전직장에서는, Git의 저장소를 추가,삭제하는 웹 페이지가 구축되어 있었음

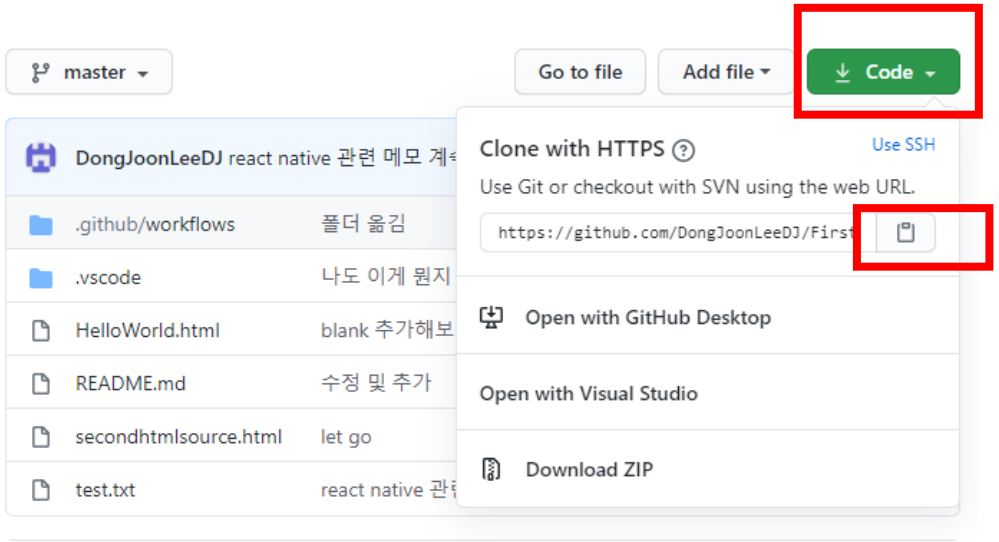
GIT 설치

- https://coding-factory.tistory.com/245
- 특별한 거 없고 그냥 next 연타하고, 15~16번 부분은 Git Bash 열어서 수행하기
  - GitHub 회원 가입시 사용한 e-mail 주소를 사용할 것

※ 사내 서버에 접속하기 위함이라면 아무 메일이 써도 상관없지만 본 실습에서는 GitHub를 활용할 것이므로 GitHub용 e-mail을 활용할 것

GIT 활용(CONSOLE)[1/7]

1. 원격지 주소 가져오기
    1. 사내 네트워크
        - 담당자에게 물어보기 등
        - 예시) http://192.168.0.168/Bonobo.Git.Server2/LabelPrinter_India.git
    2. GitHub



클릭하여 원격지 주소 복사

GIT 활용(CONSOLE)[2/7]

2. 원격저장소에 있는 파일들을 다운로드할 빈(Empty) 폴더 선택

마우스 우 클릭 후 Git Bash 선택하기

3-1. Clone을 이용하여 master branch의 내용과 나머지 branch들의 형상을 전부 다운로드

`Git clone [원격지 주소]`

예시) `git clone https://github.com/DongJoonLeeDJ/FirstTestGitHub.git`

```
ddeng@DESKTOP-KHOLR4R MINGW64 /e/KB/GitCloneConsole
$ git clone https://github.com/DongJoonLeeDJ/FirstTestGitHub.git
Cloning into 'FirstTestGitHub'...
```

※ 모든 branc를 보고 싶을 경우, 해당 폴더 안에서 Git Bash를 다시 실행 후, `git branch -a` 명령어 입력

```
$ git branch -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/Tester
  remotes/origin/master
```

※ git checkout [branch명]을 통하여 branch 변경 가능

```
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
```

GIT 활용(CONSOLE)[3/7]

3-2. 특정 branch의 파일들만 다운로드하고 싶을 경우 절차

1. git init 입력

```
ddeng@DESKTOP-KHOLR4R MINGW64 /e/KB/GitNew20200831
$ git init
Initialized empty Git repository in E:/KB/GitNew20200831/.git/
```

※ 1번만 할 경우에는 순수 local 저장소가 됨. 즉 그 어떤 원격 저장소에도 연결되지 않음

2. 원격저장소의 위치를 origin으로 치환함

git remote add [변수명] [원격지주소]

```
ddeng@DESKTOP-KHOLR4R MINGW64 /e/KB/GitNew20200831 (master)
$ git remote add origin https://github.com/DongJoonLeeDJ/FirstTestGitHub.git
```

※ origin 대신 다른 이름을 써도 되고, 이 단계는 생략해도 관계없음

※ 본 단계를 생략한 경우 origin 대신 원격지 주소를 입력해야 하며, 참고로 필자는 이 단계를 생략했었음

3. git pull [변수명][branch명]

```
git pull origin master
```

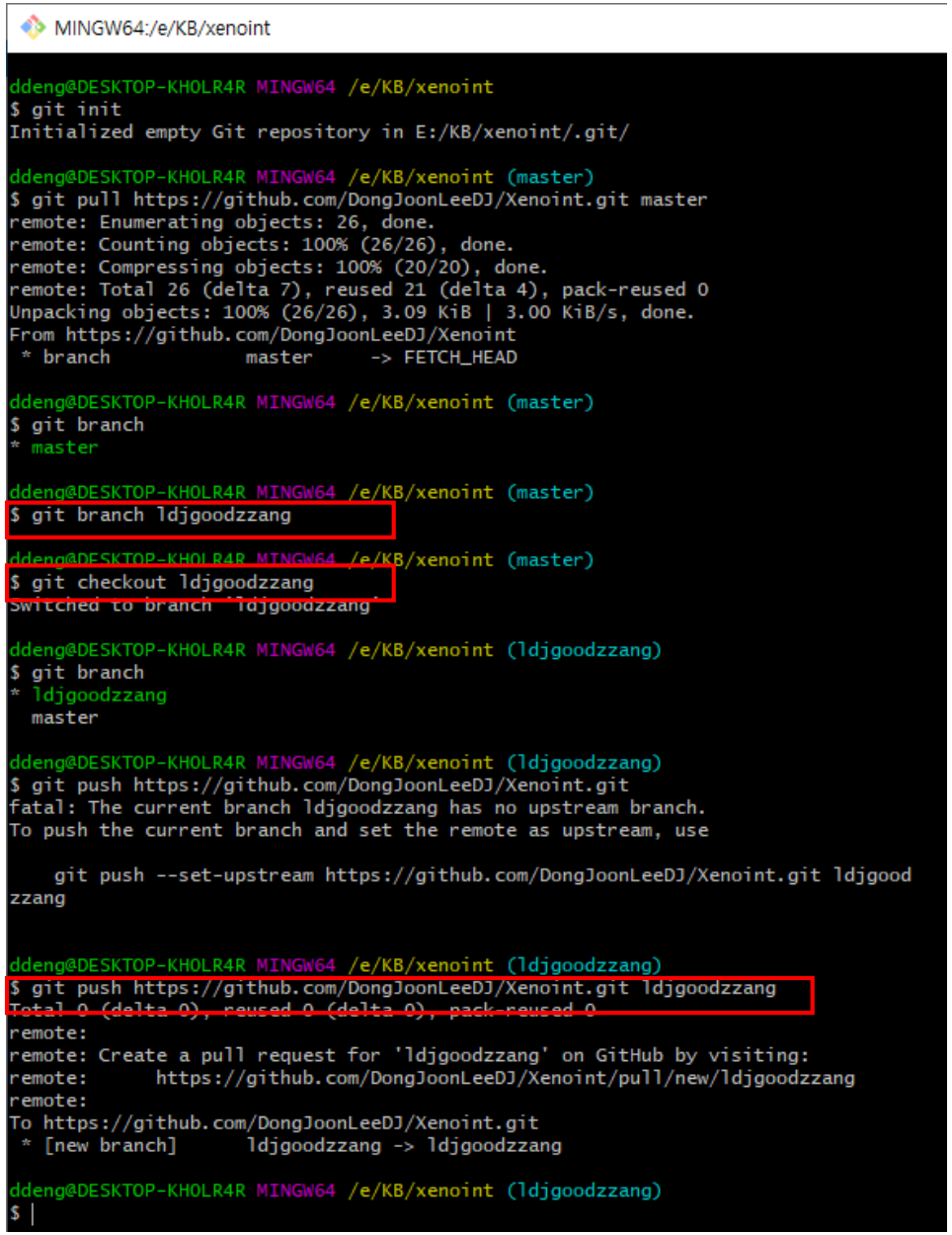
GIT 활용(CONSOLE)[4/7]

4. 새로운 분기점(branch) 생성 및 원격지 적용

git branch [추가하고 싶은 branch명]

git checkout [추가한 branch명]

git push [원격지주소] [추가한 branch명]



GIT 활용(CONSOLE)[5/7]

5. 파일 추가, 삭제, 수정하기

1-1. 특정 파일 하나에 대해서만 작업할 경우

git add [파일명]

※ 파일을 삭제한 경우 git status 명령어를 통하여 삭제한 파일명을 지정할 것

```
ddeng@DESKTOP-KHOLR4R MINGW64 /e/KB/xenoint (ldjgoodzzang)
$ git status
On branch ldjgoodzzang
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    second file.txt
        modified:   "\354\235\264\353\217\231\354\244\200 \354\204\240\354\203\2
35\353\213\230 \354\232\260\354\243\274 \354\265\234\352\260\225.txt"

no changes added to commit (use "git add" and/or "git commit -a")

ddeng@DESKTOP-KHOLR4R MINGW64 /e/KB/xenoint (ldjgoodzzang)
$ git add "second file.txt"
```

1-2. 폴더에 있는 모든 내용들에 대해서 작업할 경우

git add .

```
ddeng@DESKTOP-KHOLR4R MINGW64 /e/KB/xenoint (master)
$ git add .
```

GIT 활용(CONSOLE)[6/7]

5. 파일 추가, 삭제, 수정하기

2. commit

git commit –m “메시지”

```
ddeng@DESKTOP-KHOLR4R MINGW64 /e/KB/xenoint (ldjgoodzzang)
$ git commit -m "test commit and push"
```

3. push

git push [원격지주소 혹은 origin 과 같은 변수명] [branch명]

아래 두 가지 예시 모두 같은 저장소를 가리키나, origin을

해당 저장소로 등록하지 않았다면 origin 대신 주소지를 직접 써줘야 함

```
ddeng@DESKTOP-KHOLR4R MINGW64 /e/KB/xenoint (ldjgoodzzang)
$ git push origin ldjgoodzzang
```

```
ddeng@DESKTOP-KHOLR4R MINGW64 /e/KB/xenoint (ldjgoodzzang)
$ git push https://github.com/DongJoonLeeDJ/Xenoint.git ldjgoodzzang
```

GIT 활용(CONSOLE)[7/7]

6. 기타 참고 사항

git log : 그 동안 파일 수정,삭제,추가 내역

git status : 현재 저장소의 상태(변경사항이 있는지, branch명은 뭔지)

★ 항상 pull로 최신 내용을 다운받은 다음에 수정을 해야 충돌이 방지됨

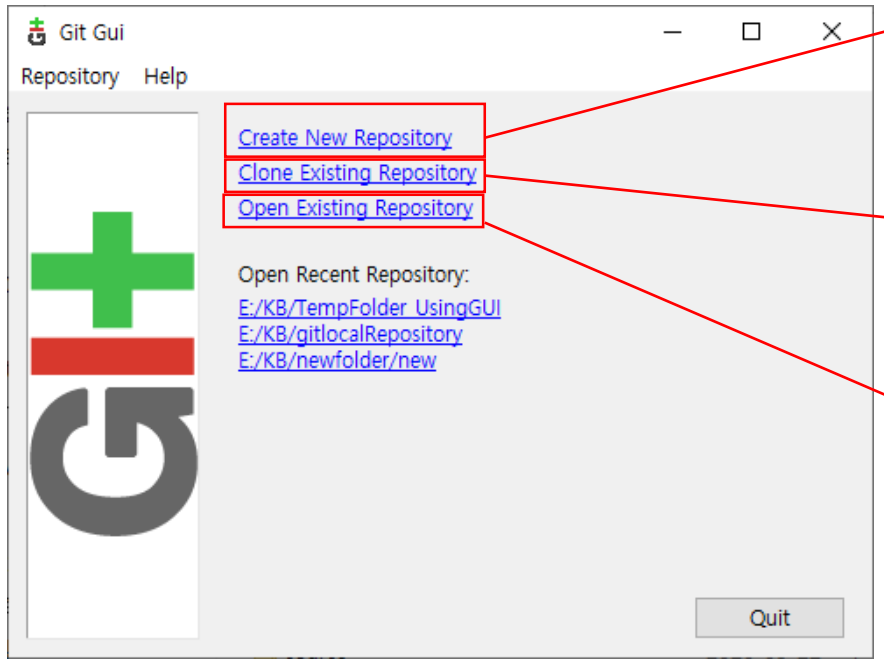
*충돌 : A.txt 파일을 옛날 버전을 pull한 뒤, 수정하려고 하였으나 이미 최신 버전에

A.txt에 대한 수정 내용이 있음. 이럴 경우에는 A.txt를 최신버전으로 pull한 뒤

수정하고 싶은 내용을 수정해야 충돌이 발생하지 않음

GIT 활용(GUI)[1/9]

- 원격지 주소를 가져오고, 빈 폴더를 선택하는 것 까지는 Git Bash와 동일
- 마우스 우클릭 후 Git Gui 선택

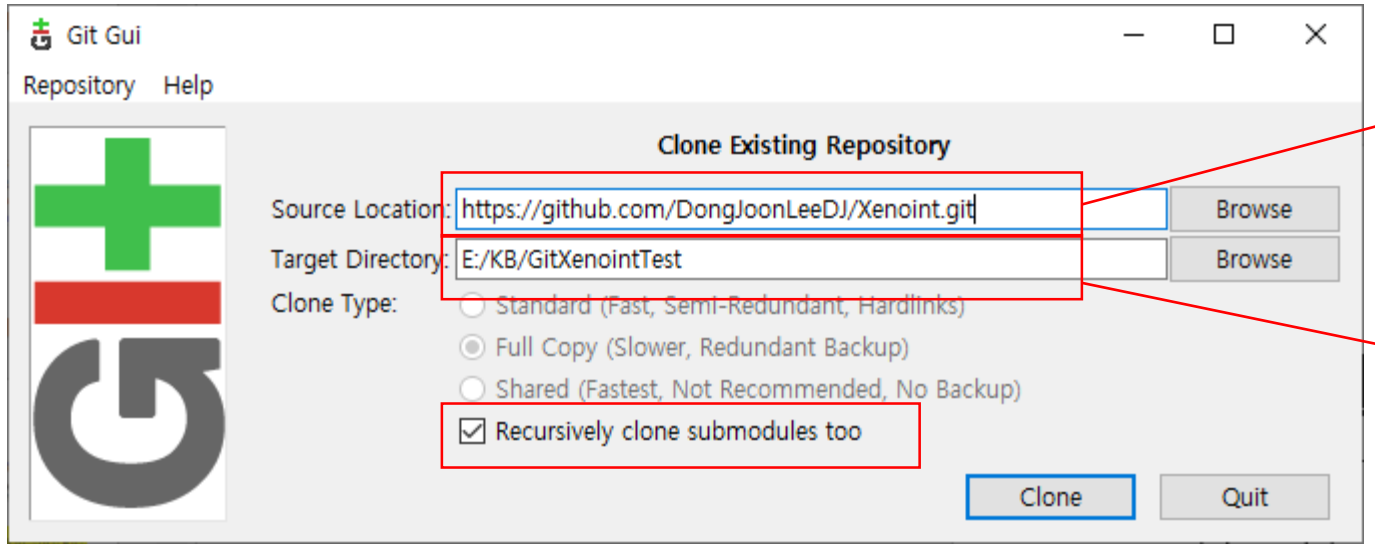


로컬에서만 사용할 수 있는 저장소 생성
Bash에서 git init한 것이랑 동일함

지정된 저장소를 Clone하는 기능
(Git GUI에서는 특정 branch만 pull 하는 기능이 없음)

만들어진 저장소를 open하는 것인데,
해당 폴더에 가서 마우스 우 클릭 후 Git Gui 선택하면 됨

GIT 활용(GUI)[2/9]



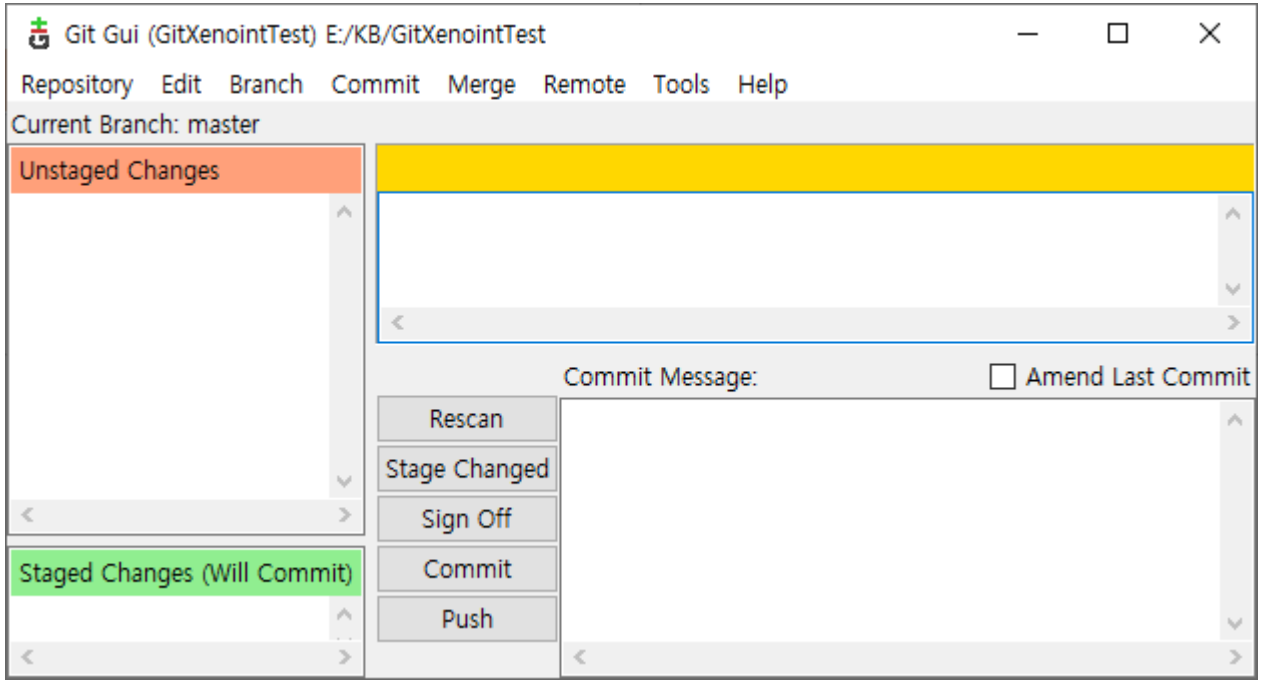
원격지 주소 입력

Clone한 자료를 저장할 폴더

- Target 폴더는 미리 만들지 말아야 하며 Clone 클릭하면 자동으로 생성됨

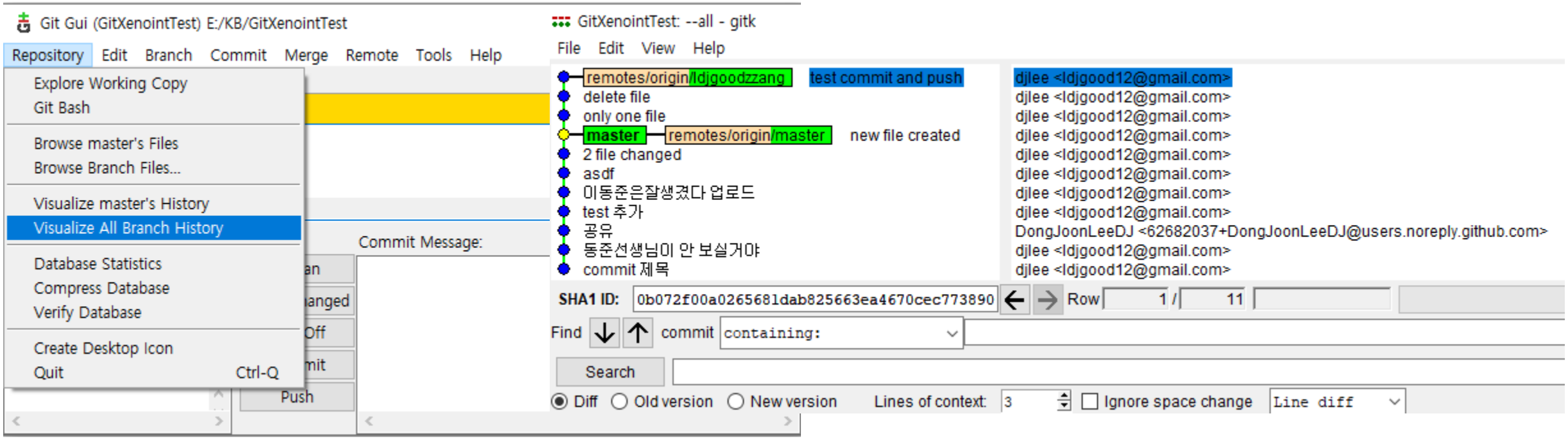
GIT 활용(GUI)[3/9]

- Clone이 완료되면 아래와 같은 창이 나타나며, 이 후에 해당 폴더 들어가서 마우스 우클릭하여 Git Gui를 누르면 아래 창으로 들어가짐



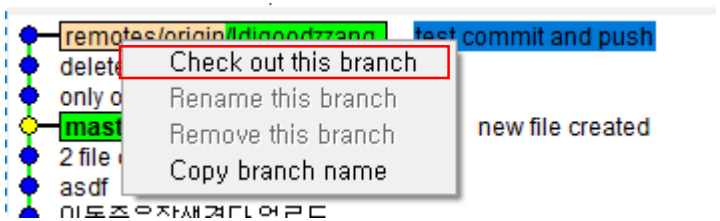
GIT 활용(GUI)[4/9]

- Branch 및 파일 변경 내역 확인
  - Repository->Visualize All Branch History 클릭



GIT 활용(GUI)[5/9]

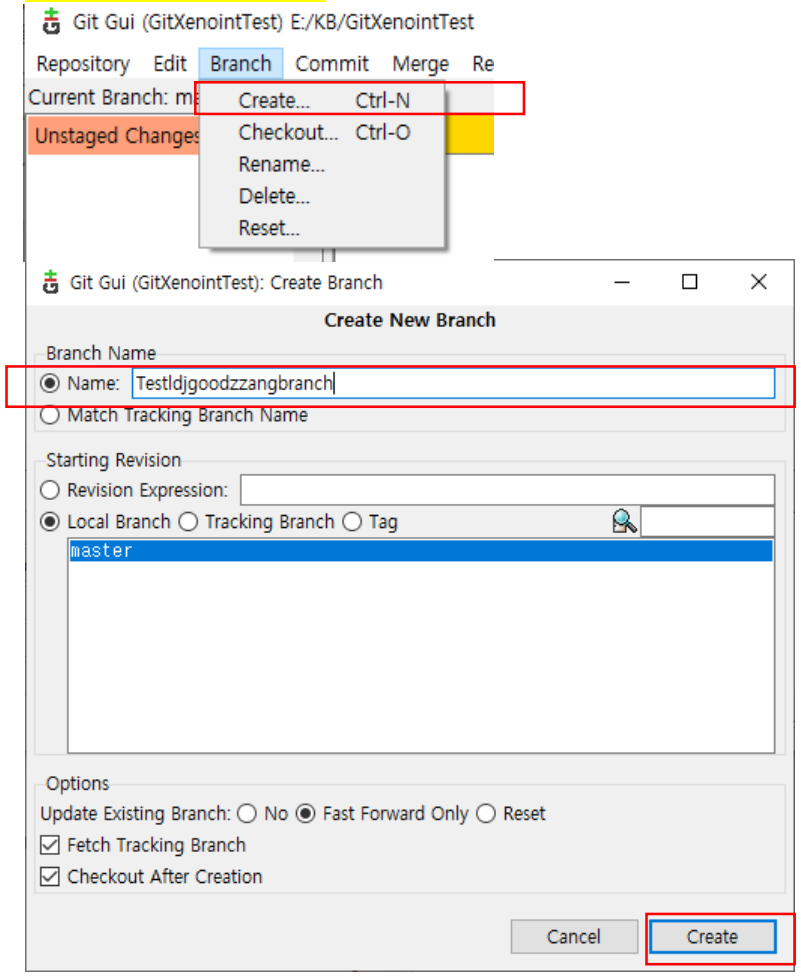
- Branch 변경
  - 해당 브랜치 부분 마우스 우 클릭 후 Check out this branch 선택



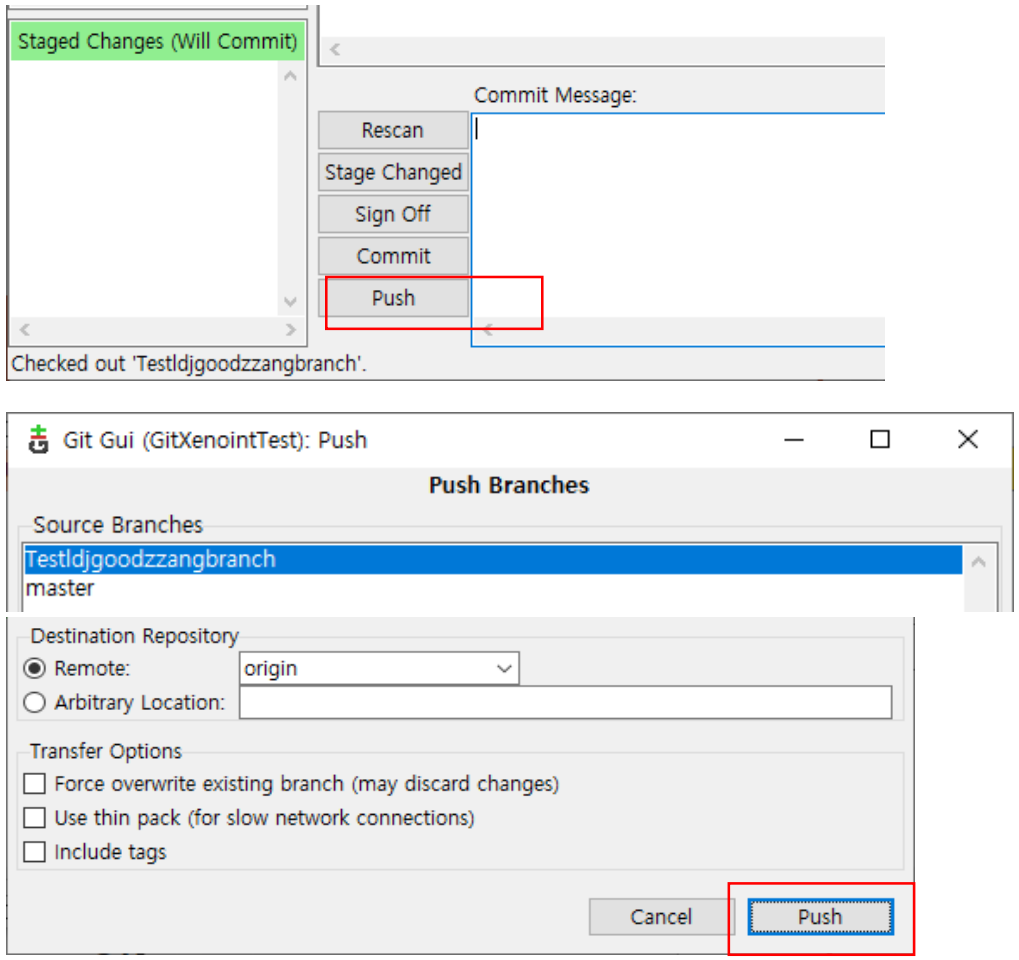
GIT 활용(GUI)[6/9]

- Branch 추가 및 원격지 적용

Branch 추가

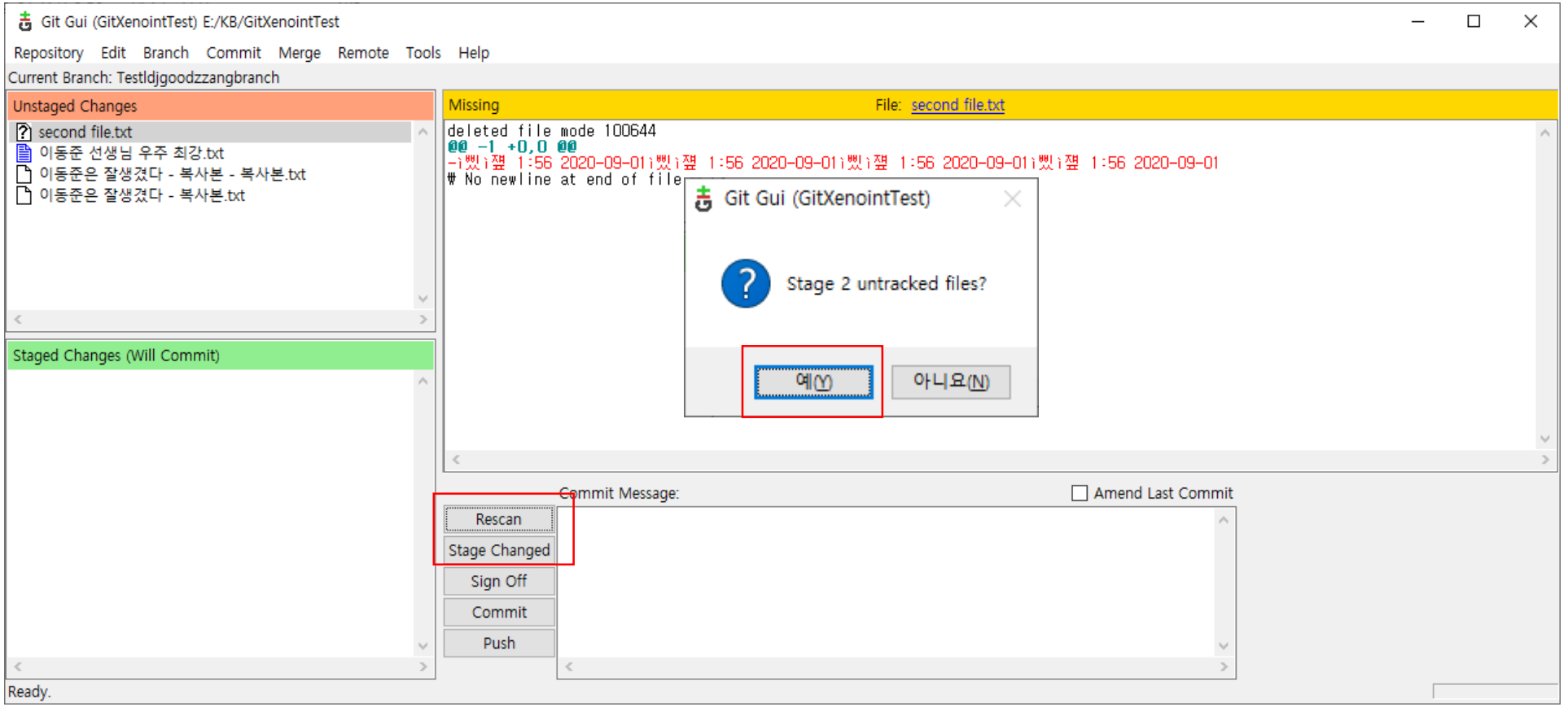


추가된 branch를 원격지에 적용



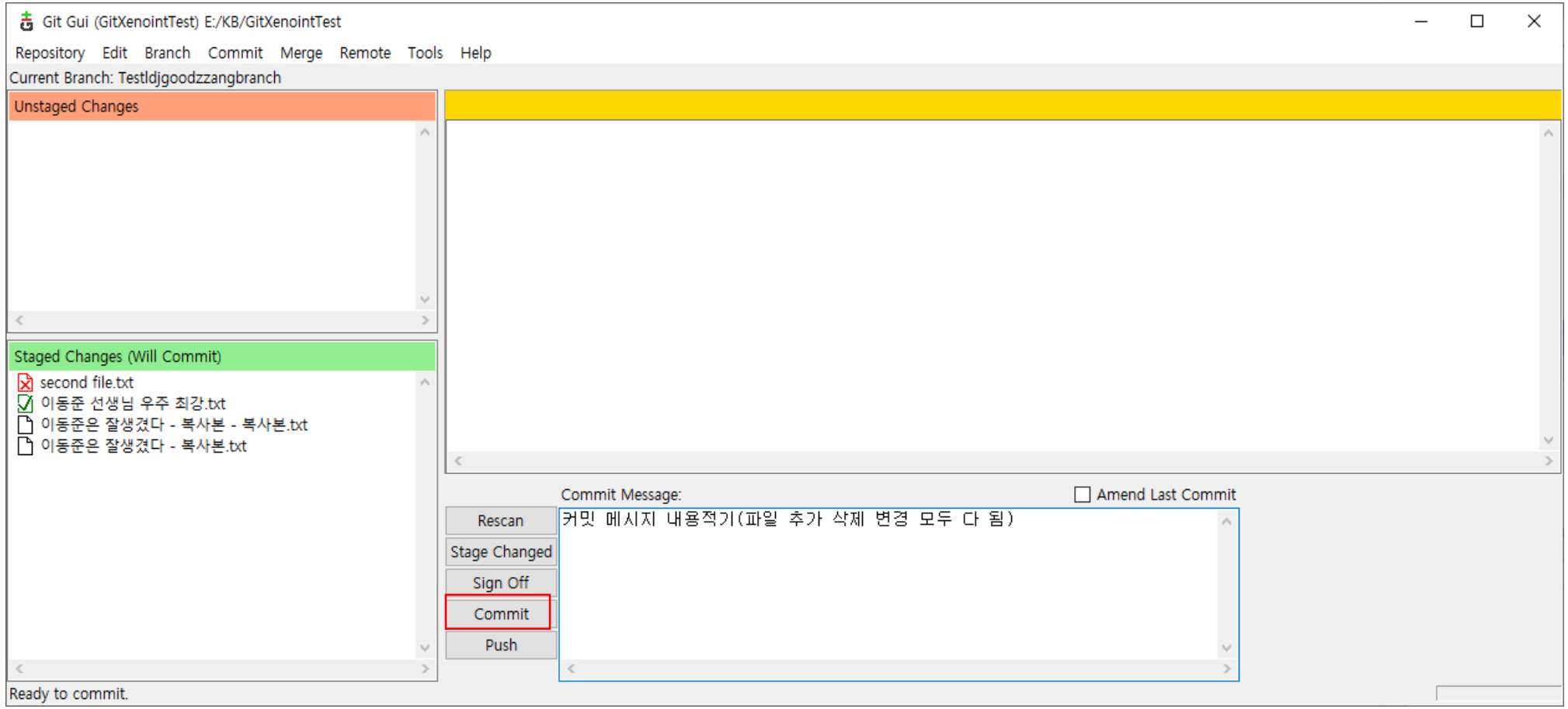
GIT 활용(GUI)[7/9]

- 파일 변경 사항 Commit & Push(1/3)
  - F5키나 Rescan을 눌러서 새로고침하고, Stage Changed 누르기
  - 아래와 같은 메시지 나오면 예 누르기



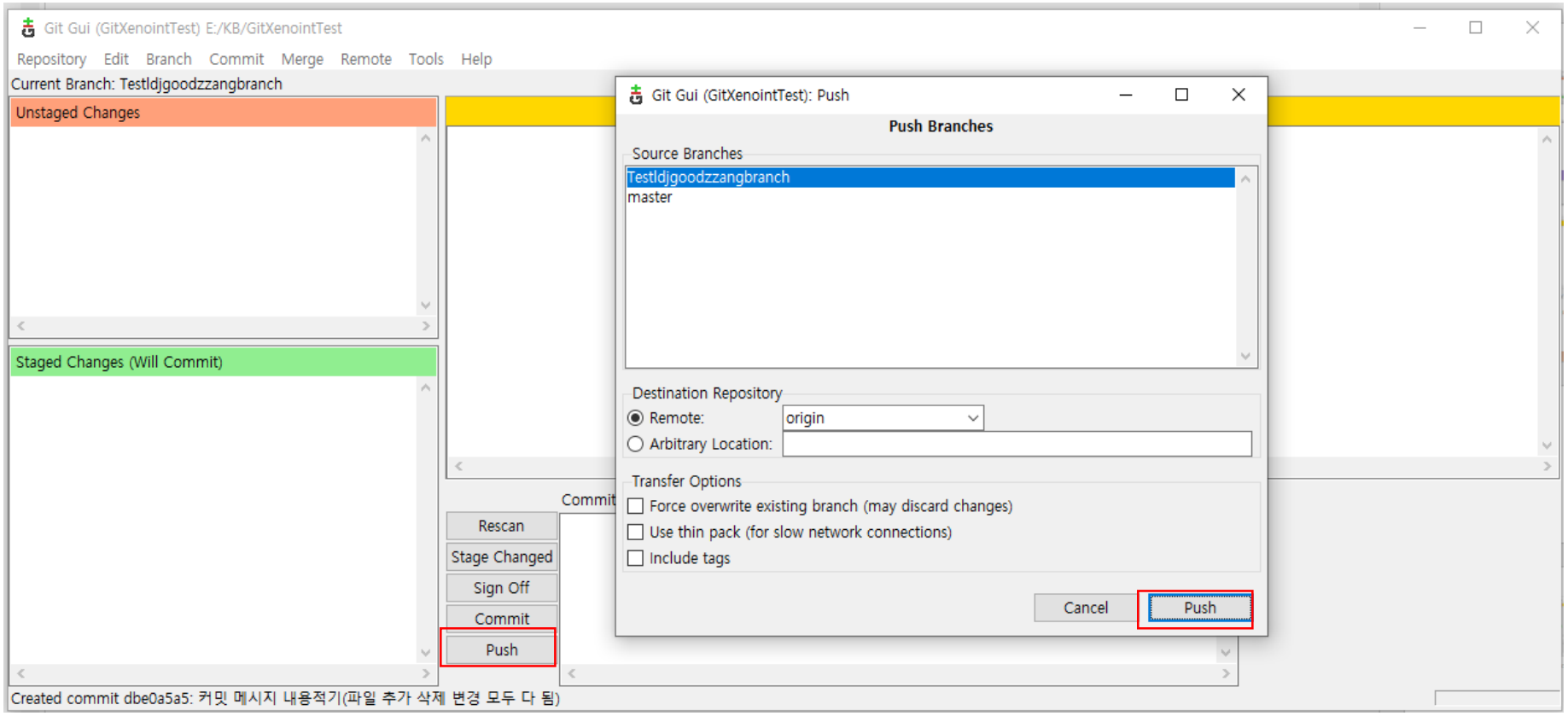
GIT 활용(GUI)[8/9]

- 파일 변경 사항 Commit & Push(2/3)
  - Commit 할 메시지 적고 Commit 버튼 클릭하기



GIT 활용(GUI)[9/9]

- 파일 변경 사항 Commit & Push(3/3)
  - Push 버튼 클릭하고 업로드할 저장소 선택하기
    - 만약 origi이 등록되지 않은 상태라면 Arbitrary Location에 원격지 주소 적기



필자의 주된 GIT 활용 패턴

- 소스코드 다운로드&업로드
  - Git bash의 pull로 해당 저장소의 지정된 branc의 소스코드를 다운로드함
    - Clone은 부득이한 경우가 아니면 하지 않음

      ※ 용량문제 및 지나치게 많은 git 형상 보관으로 인한 과부하
  - 소스코드 수정 후 Git Gui에서 Commit 및 Push를 진행함
  - 즉, 다운로드는 Bash로 하였고, 업로드는 GUI로 진행함

※ 만약 Git Hub를 사용하는 회사라면, 다운로드는 GitHub Desktop을 사용하는 것이 편해 보이며, 업로드는 Git GUI나 GitHub Desktop 둘 중 하나를 활용하면 될 것 같음

- 소스코드 활용
  - 회사 노트북에 pull로 소스다운 받고, 출장지에서 GUI로 작업 함
  - 팀장님께 코드리뷰받고, 회사 서버에 있는 소스를 pull로 다운받고, 비욘드컴페어로 비교한 뒤, 업로드해야 하는 부분만 업로드하여 commit & push함

주의 사항

Push할 때 충돌에 의하여 잘 되지 않는 경우가 많다.

강제적으로 Push하는 방법은 GUI와 Bash에 모두 있다. (인터넷 검색하면 나옴)

하지만 되도록이면 이는 사용하지 않았으면 좋겠다.

강제로 Push하면 그 파일이 갖고 있던 형상들을 강제로 덮어쓰기를 해버린다.

필자는 한 저장소에서 옛날에 pull로 받은 파일을 실수로 강제 push를 하여서 최근에 작업한 내용까지 모두 날릴 번한 적이 있었다. 다행히 최신 버전의 백업본이 있어서 그 백업본을 다시 강제 push하여서 아무에게도 들키지 않았다.

GITHUB DESKTOP 설치 및 활용

- https://post.naver.com/viewer/postView.nhn?volumeNo=24623677&memberNo=42458017
- 위 사이트말고도 구글링하면 많은 자료들이 있음
- 굉장히 직관적으로 되어 있어서 GitHub 로그인에 문제 없고 Git만 잘 깔려있다면 무리없이 쓸 수 있음
- 많이 쓰는 메뉴
  - Clone : File -> Clone Repository
  - Commit : 파일에 변경점이 생기면 Changes 탭의 좌측 하단에 Commit 메시지 입력할 수 있음
  - Pull or Push : Repository -> Pull, Repository -> Push

자료 출처 모음

- 원숭이 그림들
  - 원숭이도 Git을 배울 수 있다는 취지에서 그려진 그림들인데... 난 원숭이만도 못한건가 T_T
  - https://backlog.com/git-tutorial/kr/intro/intro1_1.html
- GitHub 활용 및 Git 기본 용어 설명
  - https://tagilog.tistory.com/377
- 그 외 Git 명령어는 'Git 명령어 정리.txt' 파일 참고
  - 실무중 정리한 내용들