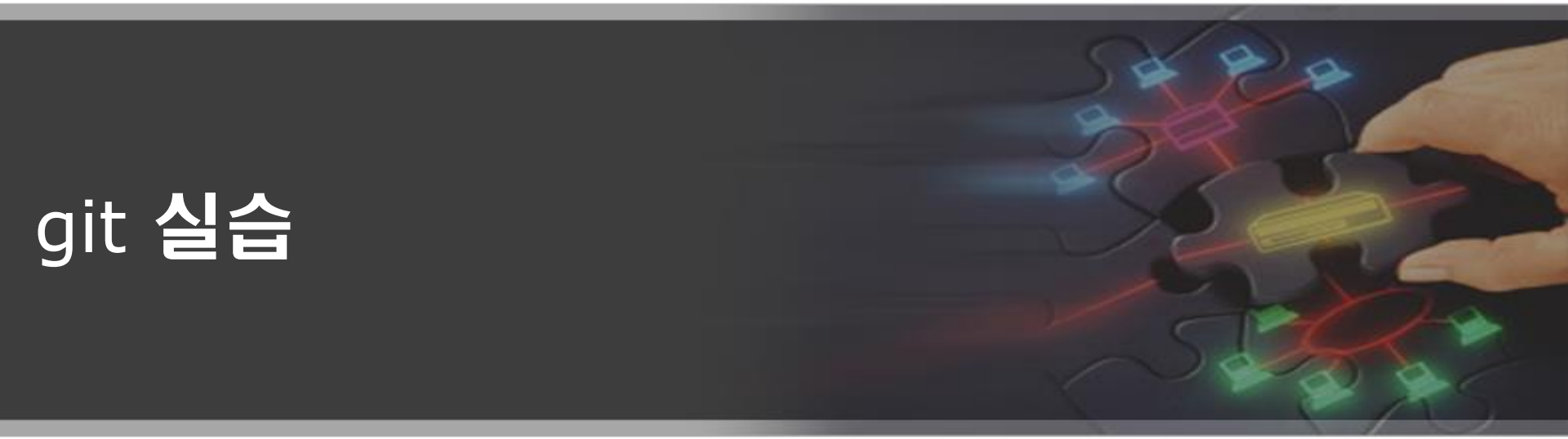
git 실습



About..

컴퓨터소프트웨어공학과
김 원 일

목 차

- TortoiseGit
- **다운로드 및 설치**
- **환경 설정**
- **설치 확인**
- **과목** github **저장소 내려 받기**
-

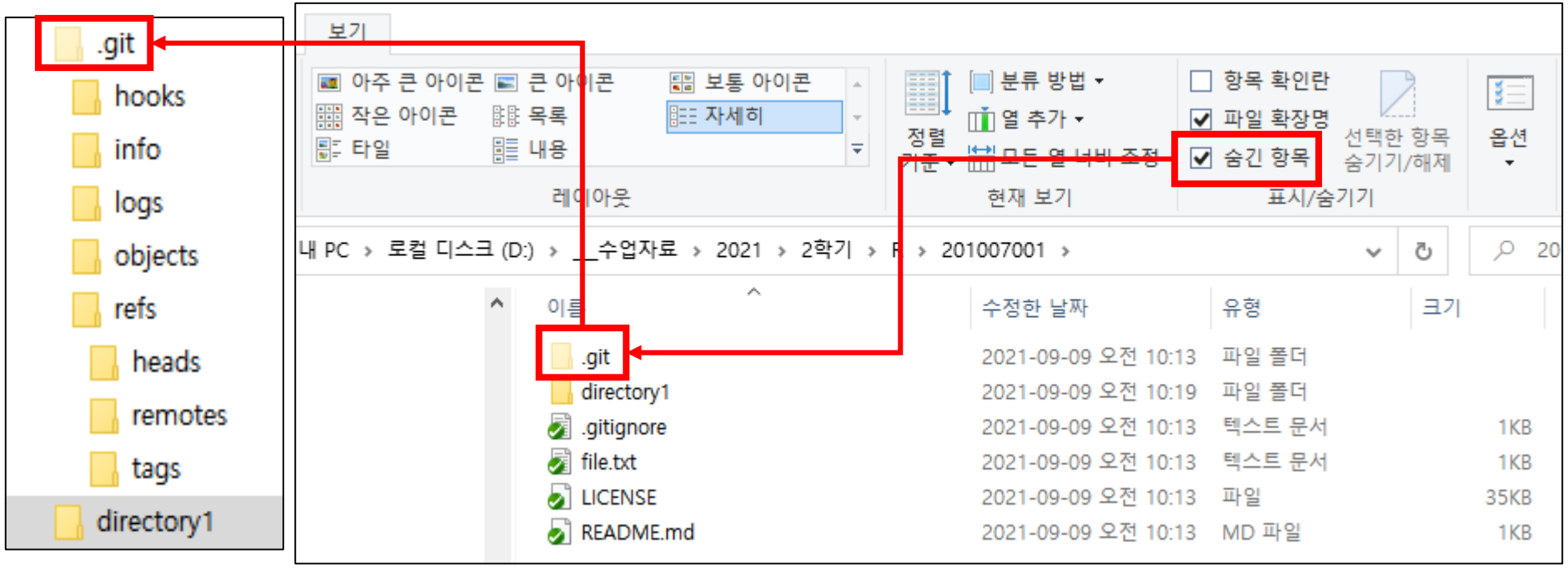
git 개인 실습

- **개인 저장소를 이용한 실습**
  - **텍스트 문서를 이용하여 git을 사용하는 실습**
  - **기본적인 파일 업로드와 브렌치를 이용해보는 실습**
- **실습 환경**
  - **git 프로그램이 PC에 설치되어 있어야 함**
  - **github.com 사이트에 계정이 있어야 함**
  - **아직 준비가 되어 있지 않다면 아래 수업 git 주소에서 자료를 받아 준비**

  - **웹으로 접속 :** https://github.com/ycs-wikim/2021_r1
  - **git으로 다운로드 :** https://github.com/ycs-wikim/2021_r1.git

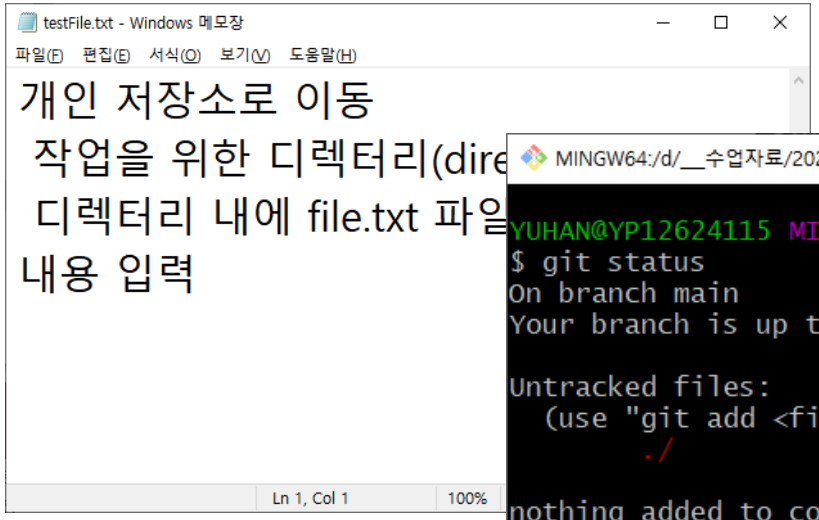
개인 저장소 실습 – 1

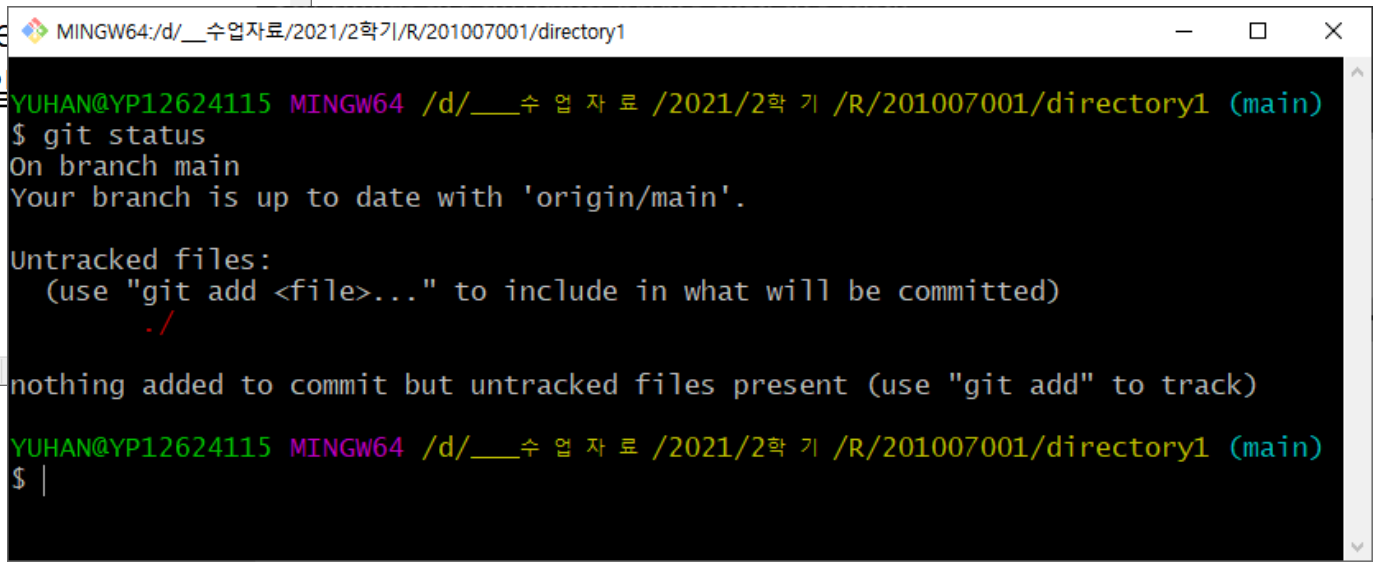
- **개인 저장소로 이동**
  - **작업을 위한 디렉터리(directory1)를 생성**
  - **탐색기 보기 옵션 "숨긴 항목"을 체크하면 저장소의 숨겨진 디렉터리가 보임**
    - **git이 동작하기 위해 필요한 정보들이 포함되어 있음**
    - **숨김 속성의 디렉터리의 임의 수정은 파일 관리에 치명적일 수 있음**
  - **확인하면 다양한 정보를 확인할 수 있음**



개인 저장소 실습 – 2

- **생성된 디렉터리에 파일을 추가**
  - **디렉터리 내에** testFile.txt **파일을 생성하고 내용 입력 후 저장**
    - **파일 내용은 어떤 내용이나 상관 없음**
  - **해당 위치에서** "Git Bash Here" **실행**
  - **현재 저장소의 상태를 출력하여 확인**

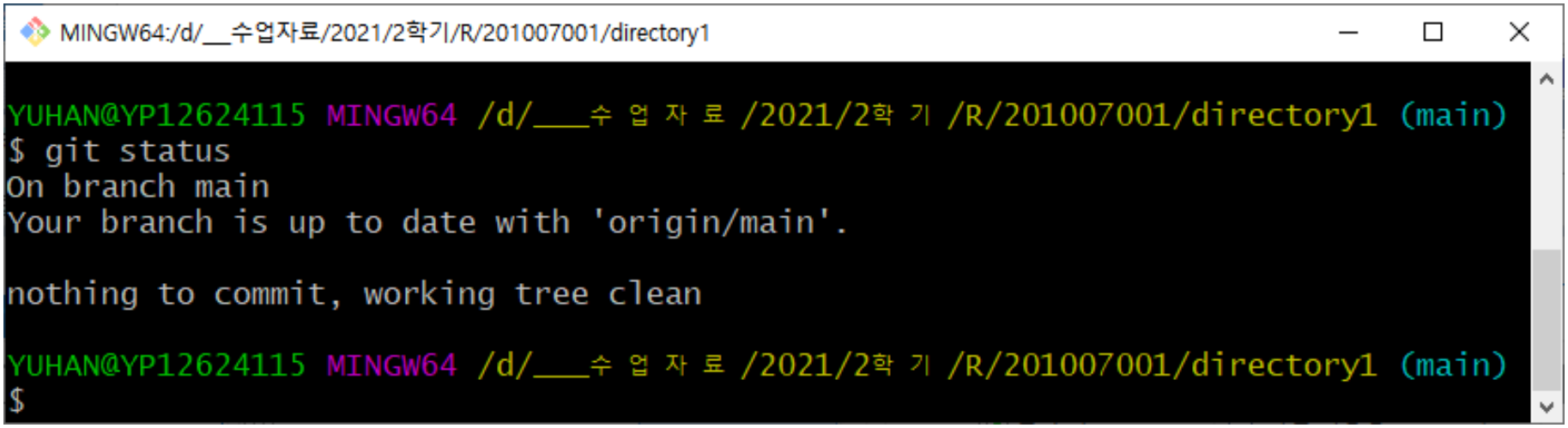




개인 저장소 실습 – 3

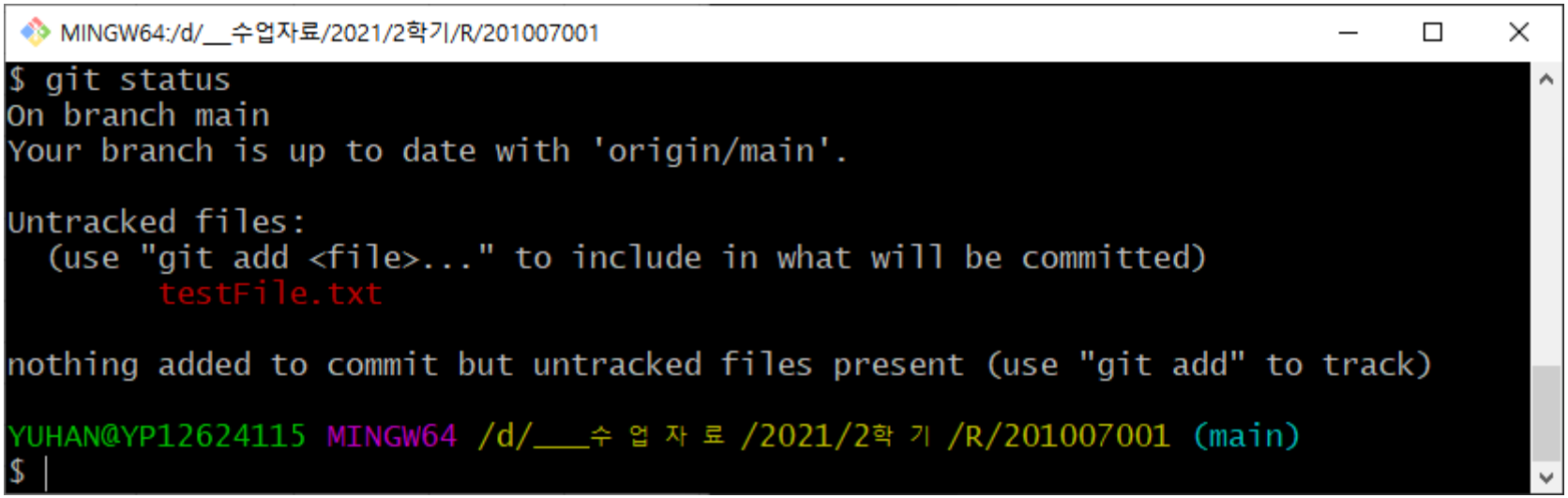
- **파일 상태 정보 확인 방법**
  - git status **명령으로 현재 상태를 확인**
  - **파일 버전 관리 상태를 확인하는 명령**
  - **항상 현재 브렌치에 대한 정보를 출력하고 시작**

  - **모든 파일이** git**에 의해 버전 관리되고 있는 경우**
  - **그림과 같이** commit**할 파일 정보와 작업 디렉터리 상태를 출력**



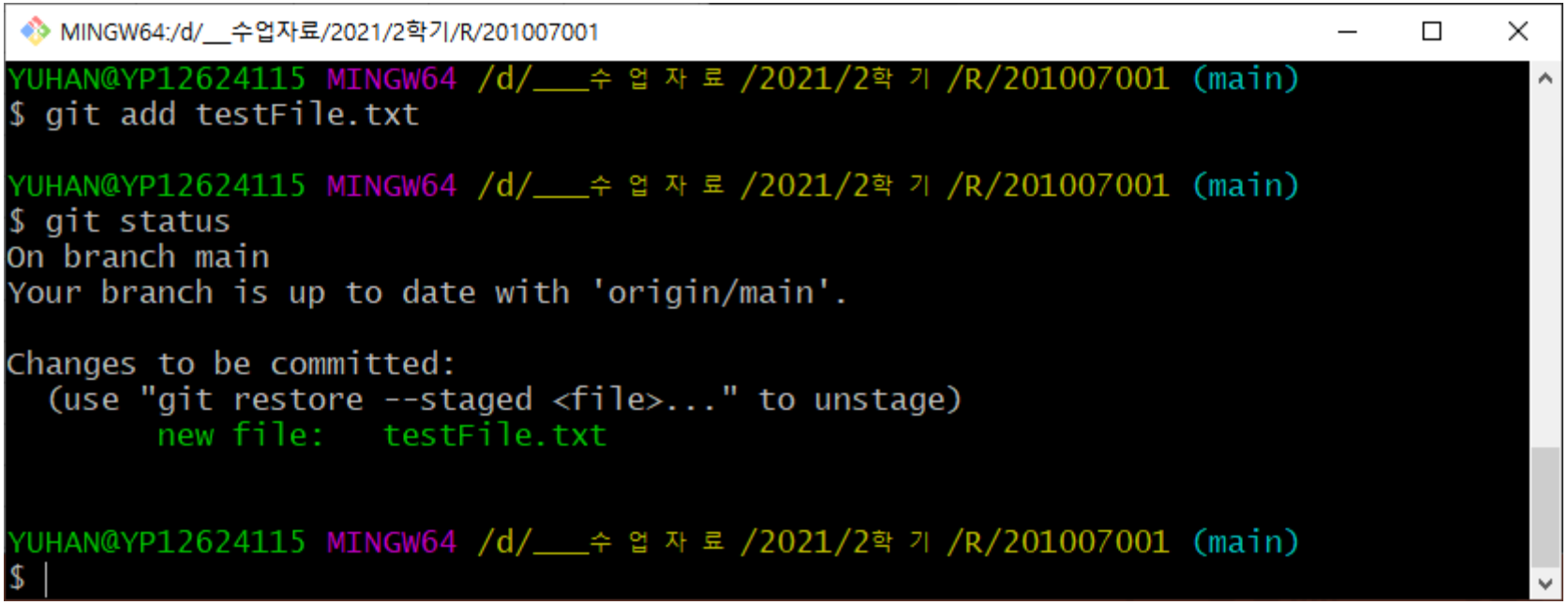
개인 저장소 실습 – 4

- **파일/디렉터리가 추가된 경우**
  - **추가된 파일/디렉터리를 붉은 색으로 출력**
  - **새 파일은 버전 관리가 적용되고 있지 않기 때문에 붉은 색으로 표현**
  - **버전 관리가 적용되지 않은 파일은** untracked **파일이라 함**
  - **추가된 새 파일을 버전 관리를 하고 싶다면** "git add <file>" **명령을 이용**
  - git**에서 확인할 수 있는 모든 파일 경로는 상대 경로로 표현**
  - commit**할 내용은 없는 상태**



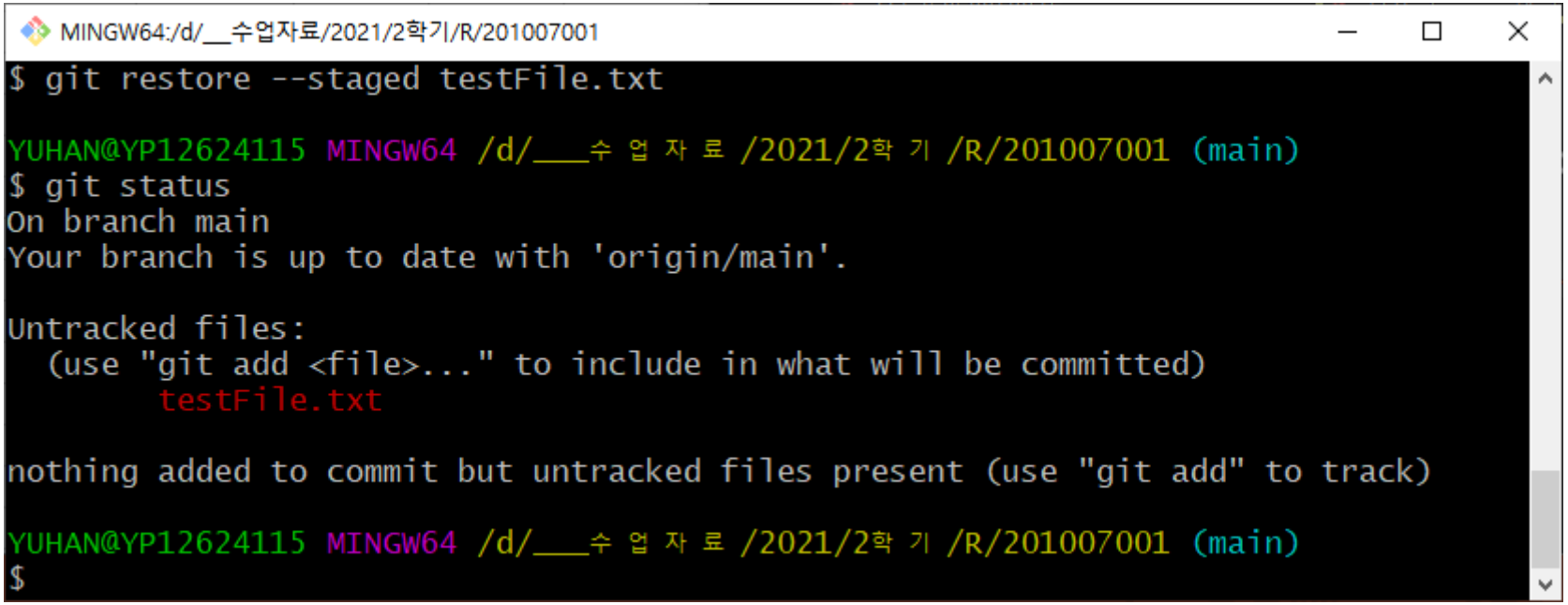
개인 저장소 실습 – 5

- **새 파일/디렉터리 버전 관리 시작**
  - "git add <file> or <directory>" **명령으로 버전 관리를 시작할 수 있음**
  - **추가된 파일을** staging **상태로 변경하는 명령어**
  - staging **상태 취소는** "git restore --staged <file> or <directory>"
  - **새로 추가된 파일이므로** "new file"**로 표시**



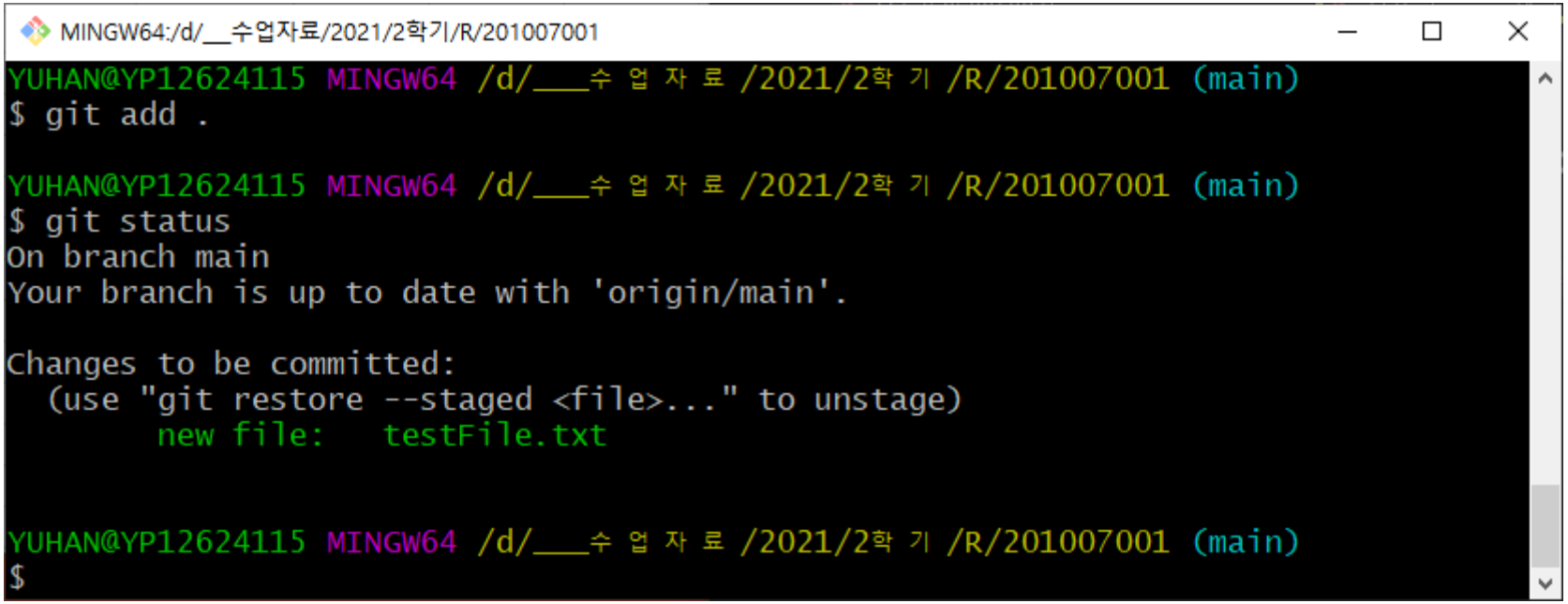
개인 저장소 실습 – 6

- staging **상태 취소**
  - staging **상태를 취소하면**, modified **또는** untracked **상태로 변경**
  - **파일이 새로 추가된 상태였으므로**, untracked **상태로 변경**



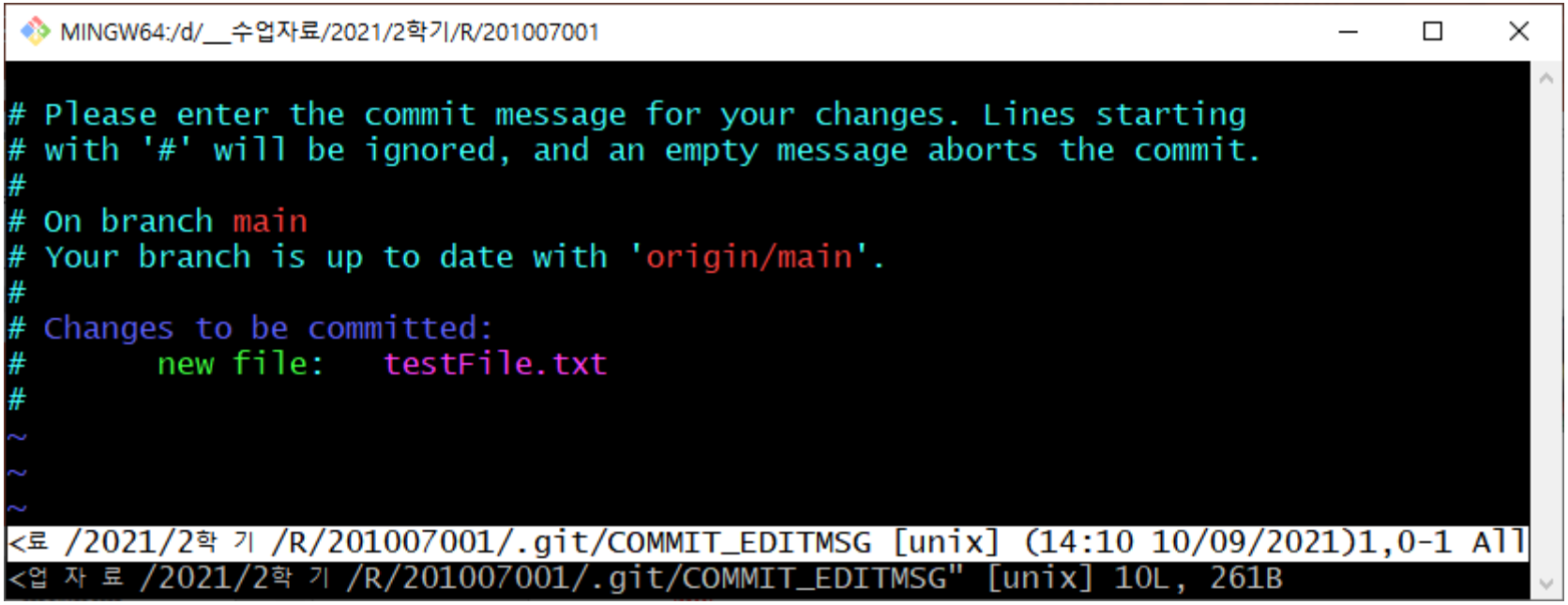
개인 저장소 실습 – 7

- **경로를 이용한 git 명령어 수행**
  - Git Bash는 **리눅스 기반 터미널**
    - **기본적인 리눅스의 경로(상대, 절대) 지정 방법을 사용할 수 있음**
  - "git add" **뒤에 리눅스 경로를 이용하여 한꺼번에 추가 가능**
  - **다수의 파일이 수정 또는 추가된 경우 해당 디렉터리에서** "git add ." **실행**
  - **실습 상황은 파일이 하나지만, 다수 파일을 추가해도 모두** staging **됨**



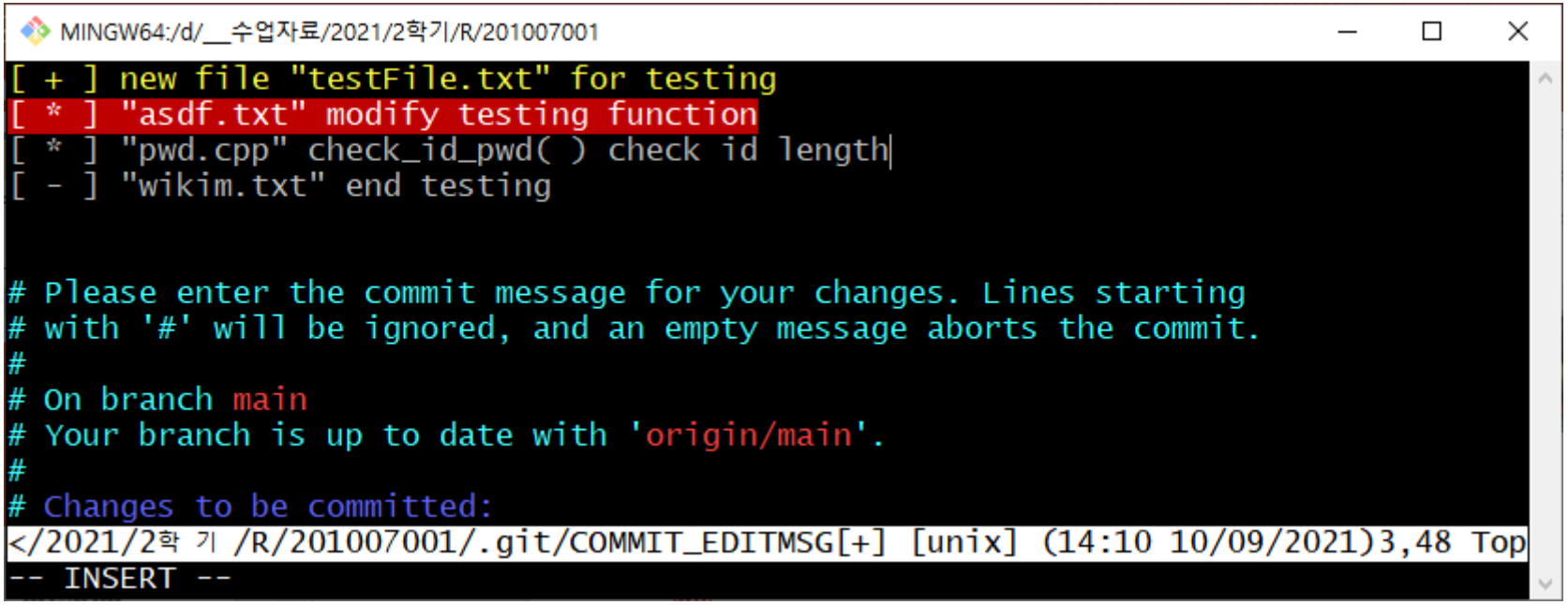
개인 저장소 실습 – 8

- **파일 버전 관리의 시작**
  - "git commit" **명령은 파일의 버전 관리가 시작됨을 의미**
  - commit **명령만 수행하면 기본 편집기가 실행됨. 기본은** vim
  - **신규 파일 또는 이전 파일 버전이 업데이트에 대한 정보를 기록**
  - **능수능란하게** vim**으로 정보를 입력하고, 저장 후 종료하면 기록됨**



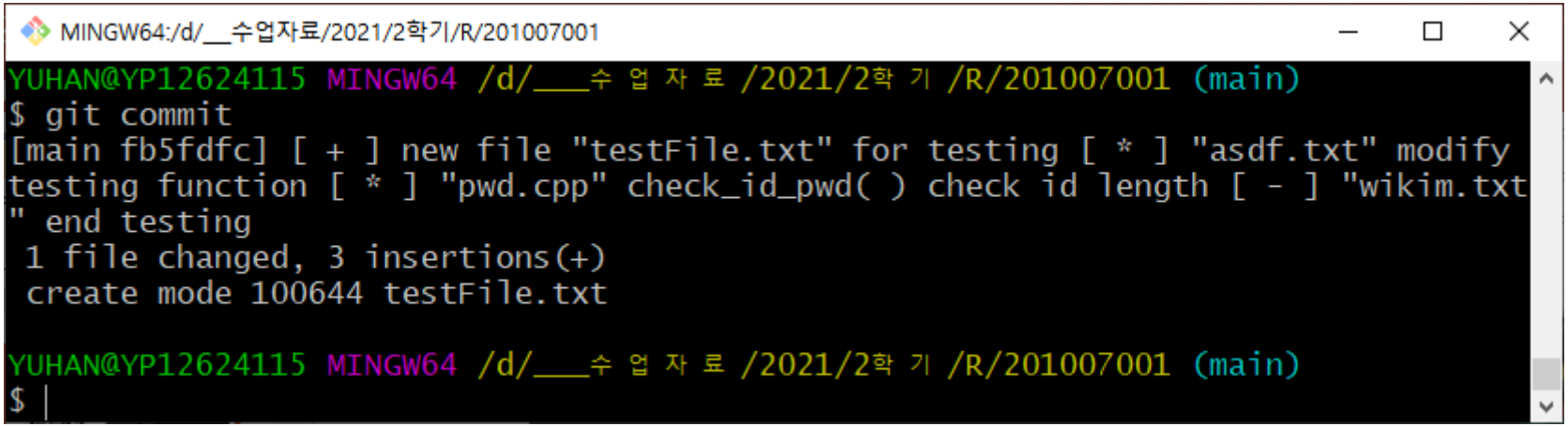
개인 저장소 실습 – 9

- commit **메시지**
  - commit **된 상태의 파일들에 대해 기술**
  - **라인의 첫 글자가 #이면 주석으로 처리되고, 기록되지 않음**
  - **파일 변경 사항을 다음과 같이 처리하면 보기 좋음**
    - [+] : **새롭게 추가된 파일이나 기능에 대해 기록**
    - [*] : **변경된 파일이나 기능에 대해 기록**
    - [-] : **삭제된 파일이나 기능에 대해 기록**



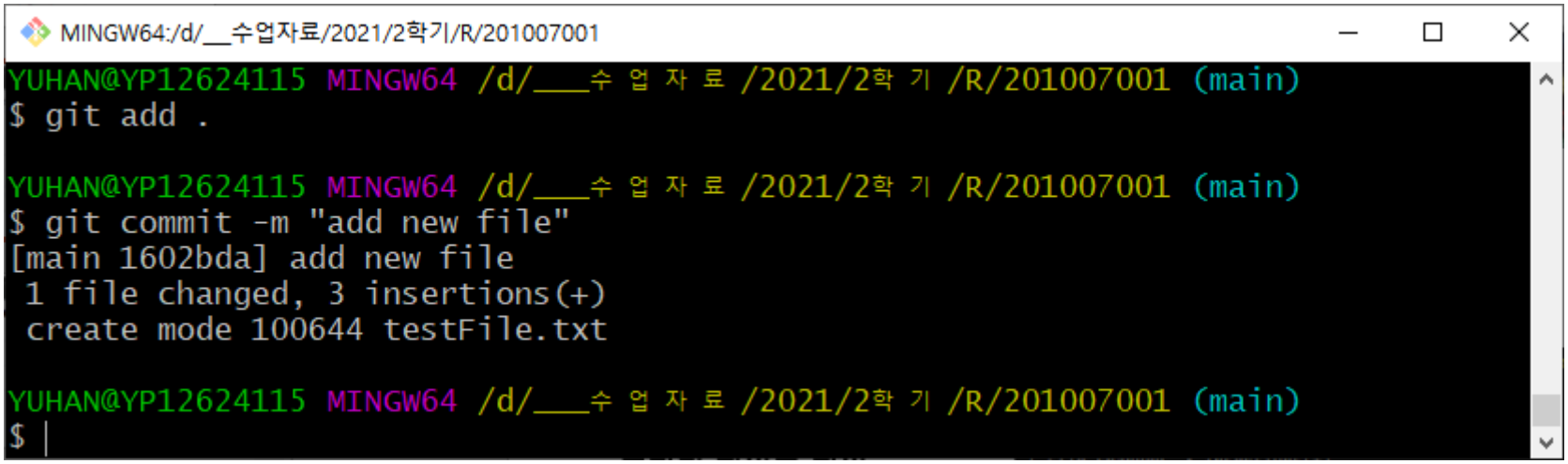
개인 저장소 실습 – 10

- **정상** commit
  - **브렌치와 해시 값을 가지 표시**
  - **입력한** commit **내용도 같이 표시**
  - **추가/수정/삭제된 파일의 개수와 라인 수에 대한 정보 출력**
  - **추가/수정/삭제된 파일과 관련된 모드 값 출력**



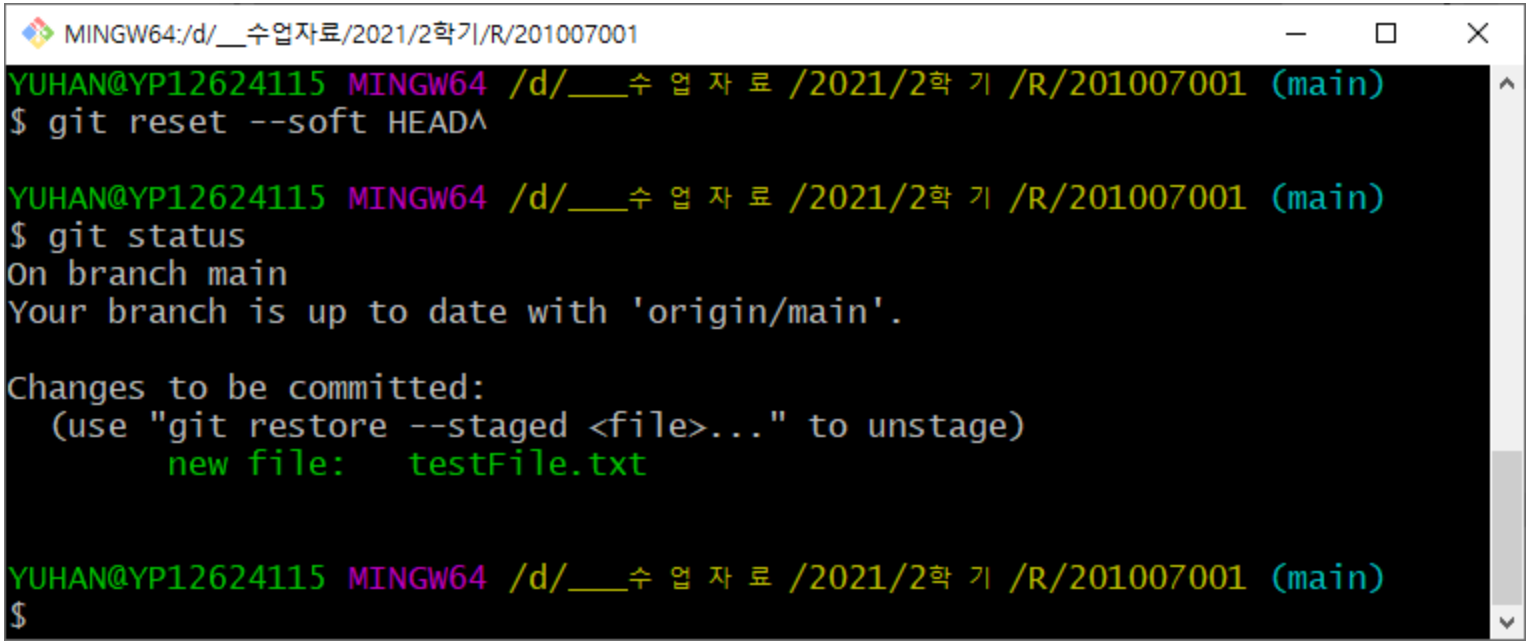
개인 저장소 실습 – 11

- **빠른 commit과 메시지**
  - **기본 편집기를 실행하지 않고** commit**과 메시지를 추가**
  - git commit –m “messages”**로 빠르게** commit **가능**
  - **콘솔에서 입력하므로, 많은 메시지를 기록하기 어려움**
  - “git commit”**과 동일한 결과 확인 가능**



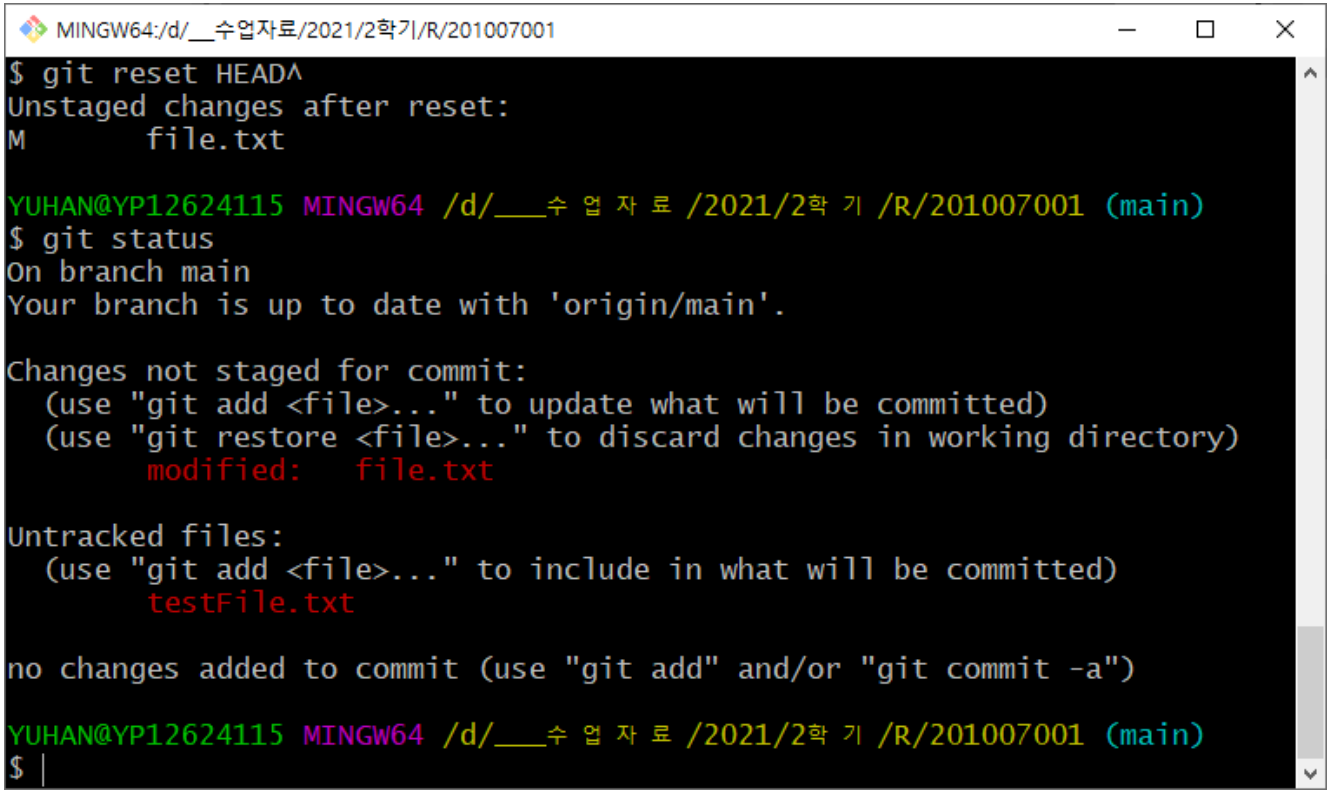
개인 저장소 실습 – 12

- commit **취소하기** – 1
  - "git reset --soft HEAD^"
    → commit**을 취소하고, 파일은** staged **상태로 유지**
  - **가장 일반적으로** commit**을 취소하는 방법**



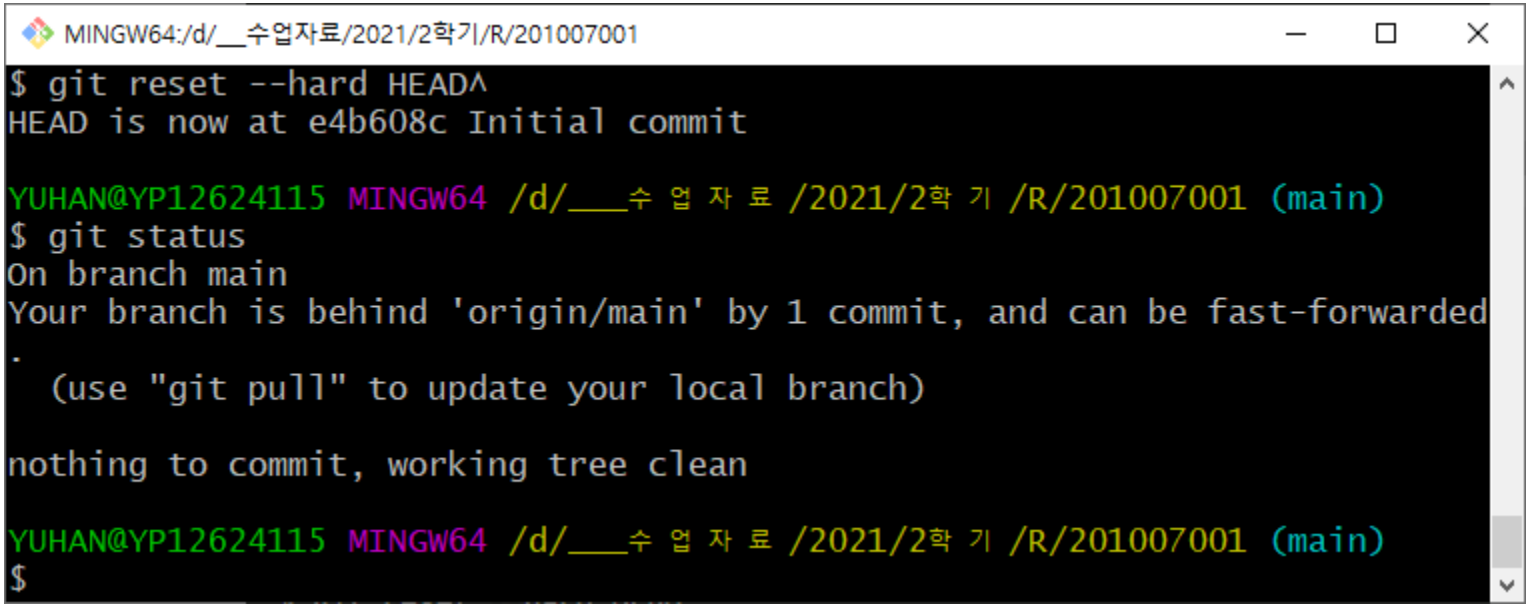
개인 저장소 실습 – 13

- commit **취소하기** – 2
  - "git reset --mixed HEAD^" or "git reset HEAD^"
    → commit**을 취소하고, 새 파일은** untracked **상태로 유지**
    → **수정된 기존 파일은** modified **상태로 유지**
  - **현재** commit **상태만 해제하는 취소 방법**



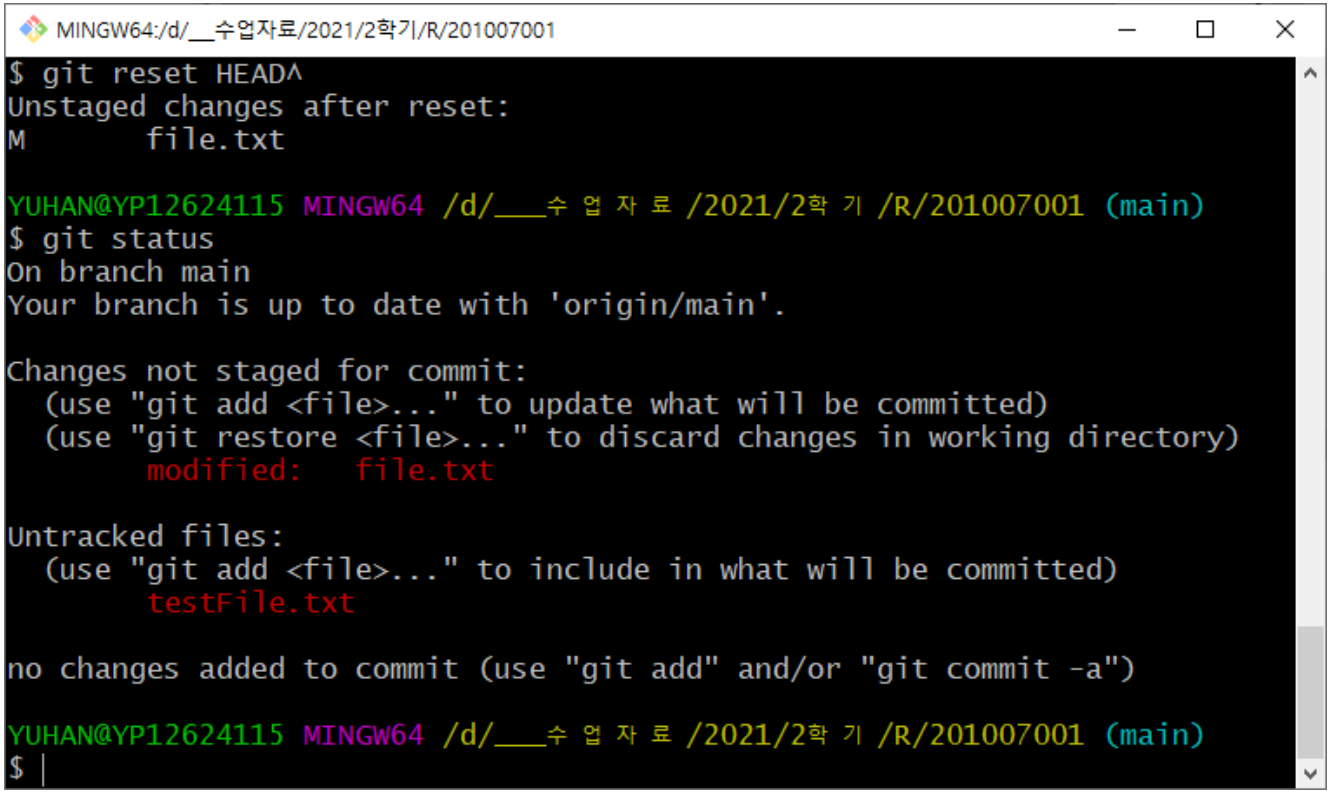
개인 저장소 실습 – 13

- commit **취소하기** – 3
  - "git reset --hard HEAD^"
    → commit**을 취소하고, 모든 파일을 최종** commit **상태로 복원**
  - **수정하거나 작성한 모든 파일이 제거(!!)되므로, 주의해서 사용**
  - **이전 상태로 완전 복원되므로, 작업한 파일 모두가 완전 삭제**



개인 저장소 실습 – 14

- "git reset" **옵션**
  - "-- soft" : **현재 파일 상태를 그대로 유지하면서 취소**
  - "-- mixed" : add **전 상태로 모든 파일을 되돌리면서 취소**
  - "-- hard" : **모든 정보를 취소하므로 주의해서 사용**

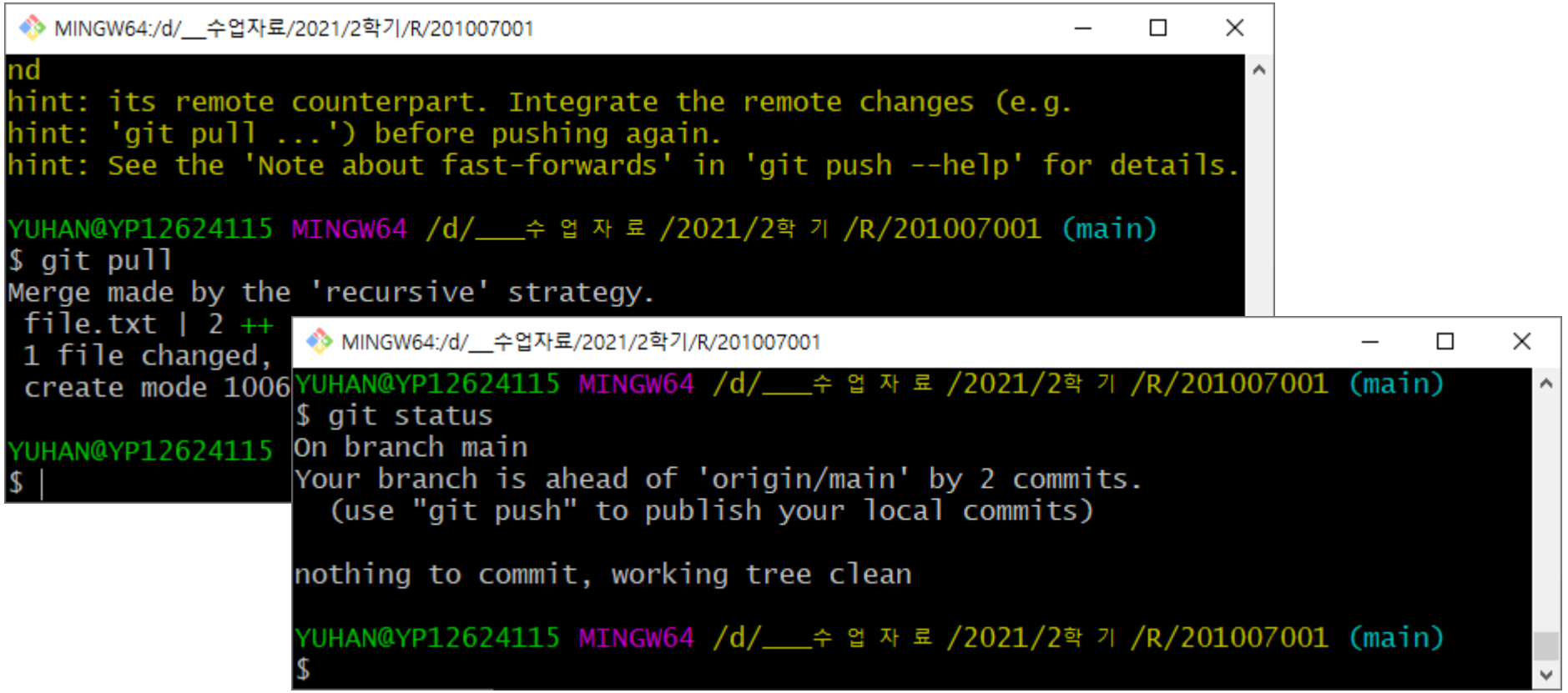


개인 저장소 실습 – 13

- **서버 업로드 시의 문제점 – 1**
  - "git push"**로 서버에 로컬 저장소를 업로드**
  - **내 브렌치가 서버** commit**보다 뒤에 있다면 문제가 발생**
  - **동일한 소스를 수정하지 않았다면** "git pull"**로 해결 가능**
  - **로컬과 서버가 다르다면 자동으로 병합에 대한** commit**이 발생**

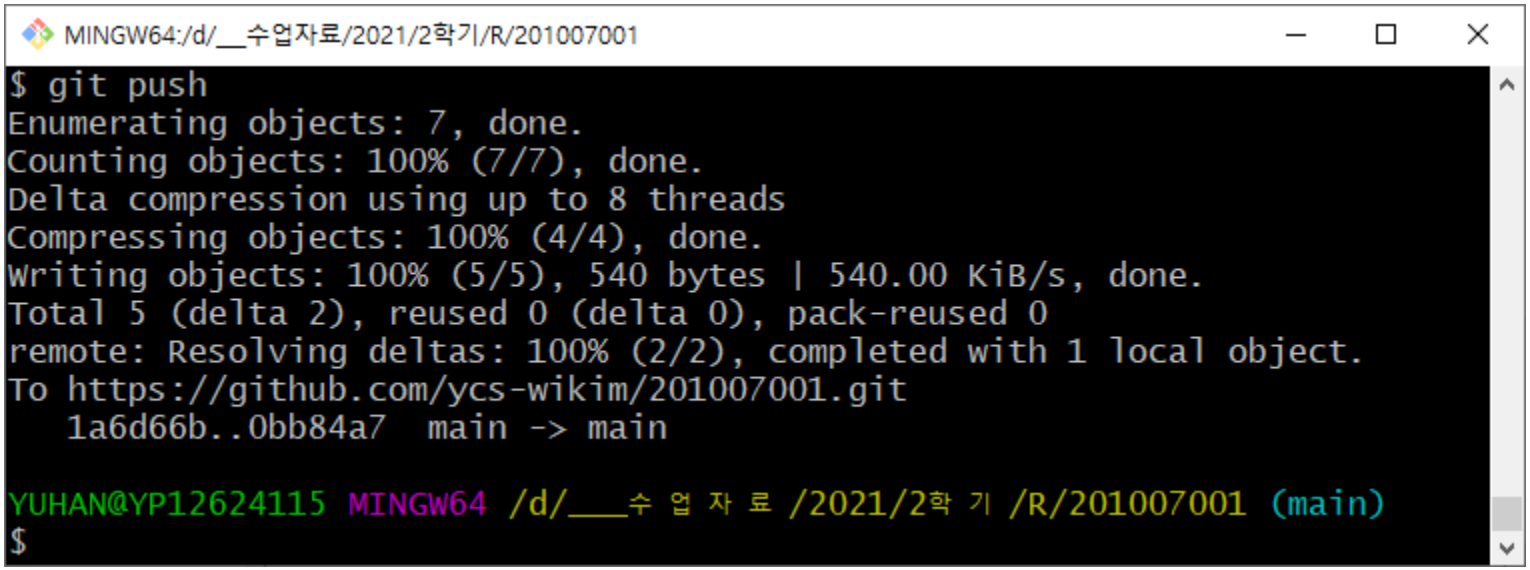
개인 저장소 실습 – 14

- **서버 업로드 시의 문제점 – 2**
  - **문제 없이 병합되면 병합 관련 정보가 출력**
  - **현재 상태 정보가 정상적으로 출력되는 것을 확인**
  - **"git pull"은 서버로부터 저장소 정보를 업데이트하는 명령어**



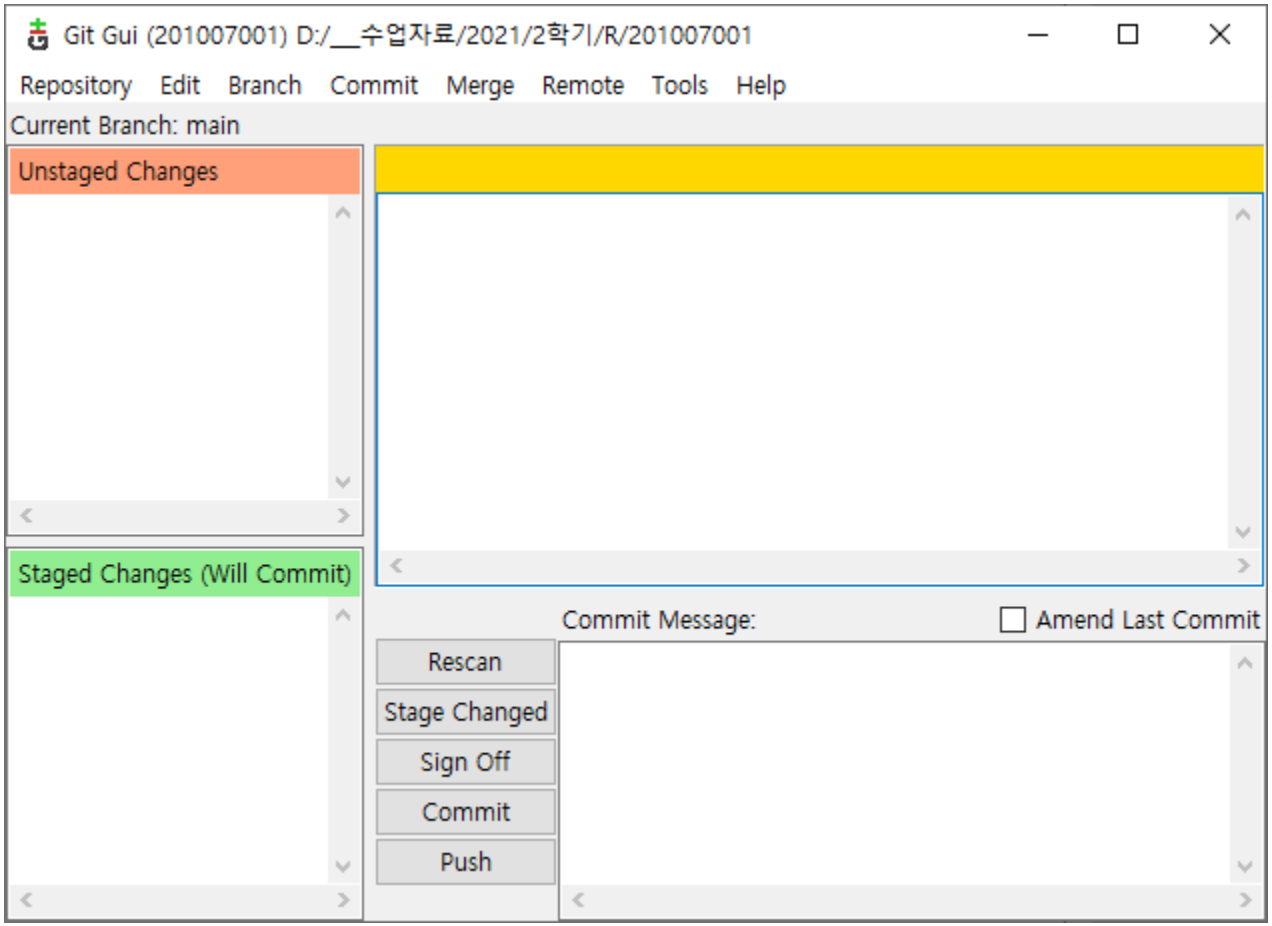
개인 저장소 실습 – 15

- **서버 업로드 시의 문제점** – 2
  - "git push" **명령어로 서버에 로컬 저장소를 그대로 업로드**
  - **저장소 업로드 되면** github.com **에서 즉시 확인 가능**



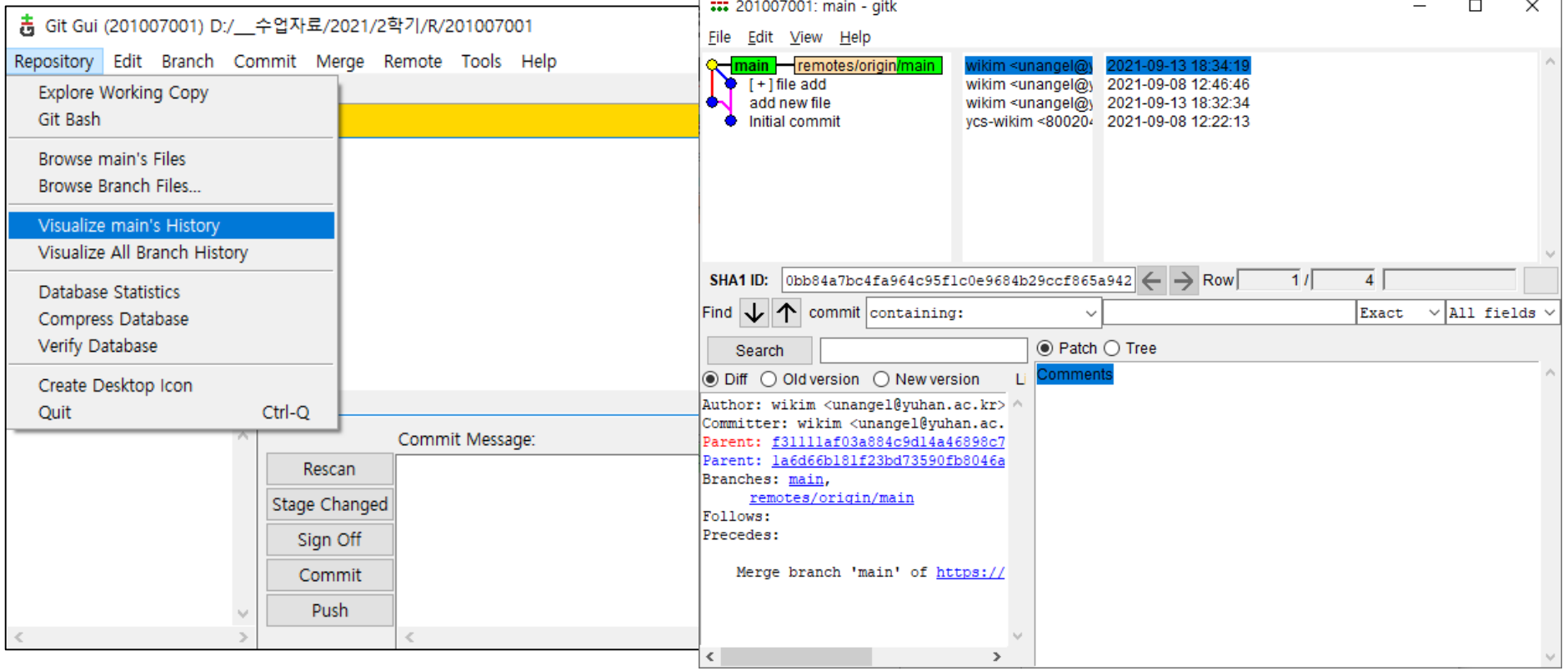
개인 저장소 실습 – 16

- **업로드 된 내용 확인**
  - **탐색기에서** "Git GUI Here"**로** GUI **프로그램 실행**



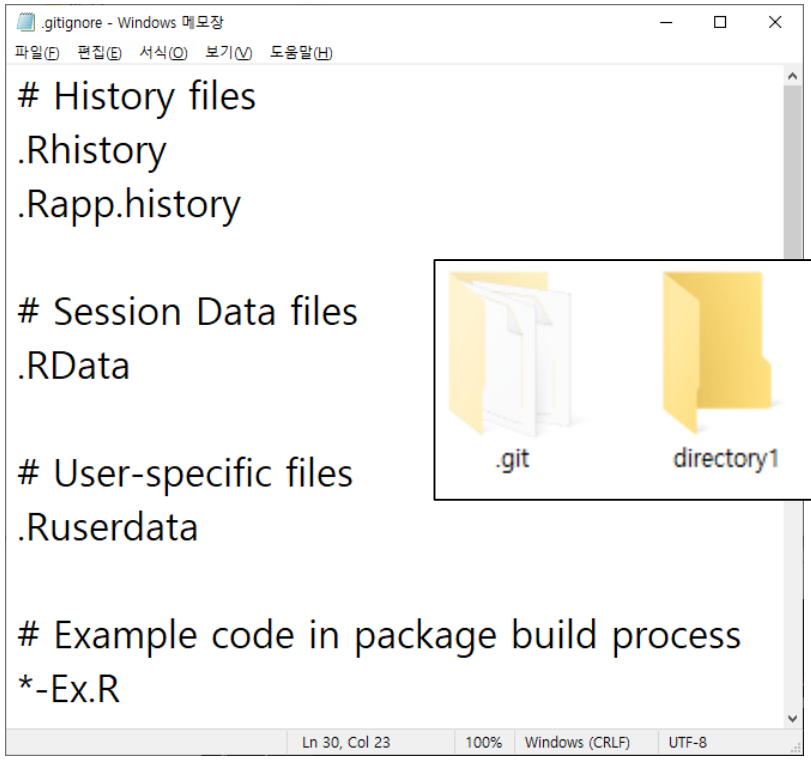
개인 저장소 실습 – 17

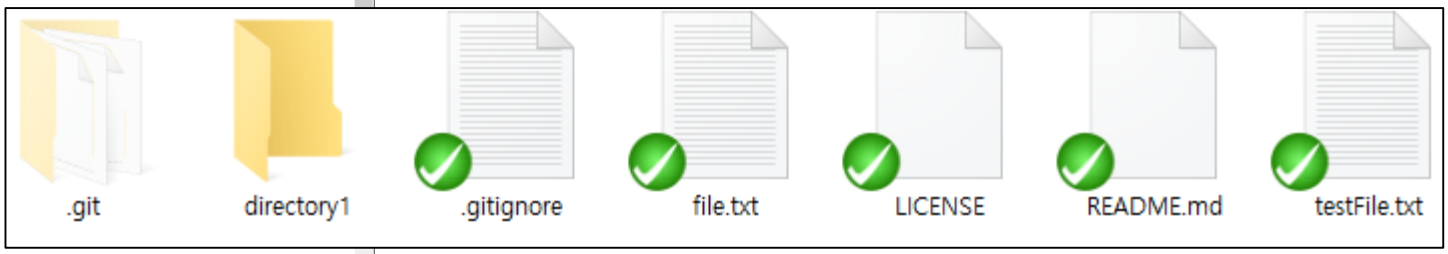
- **업로드 된 내용 확인**
  - **탐색기에서** "Git GUI Here"**로** GUI **프로그램 실행**
  - "Repository" → "Visualize main's History"**로** git flow **확인**
  - **커밋 내용과 추가/수정/삭제된 내용을 확인할 수 있음**



.gitignore 파일 – 1

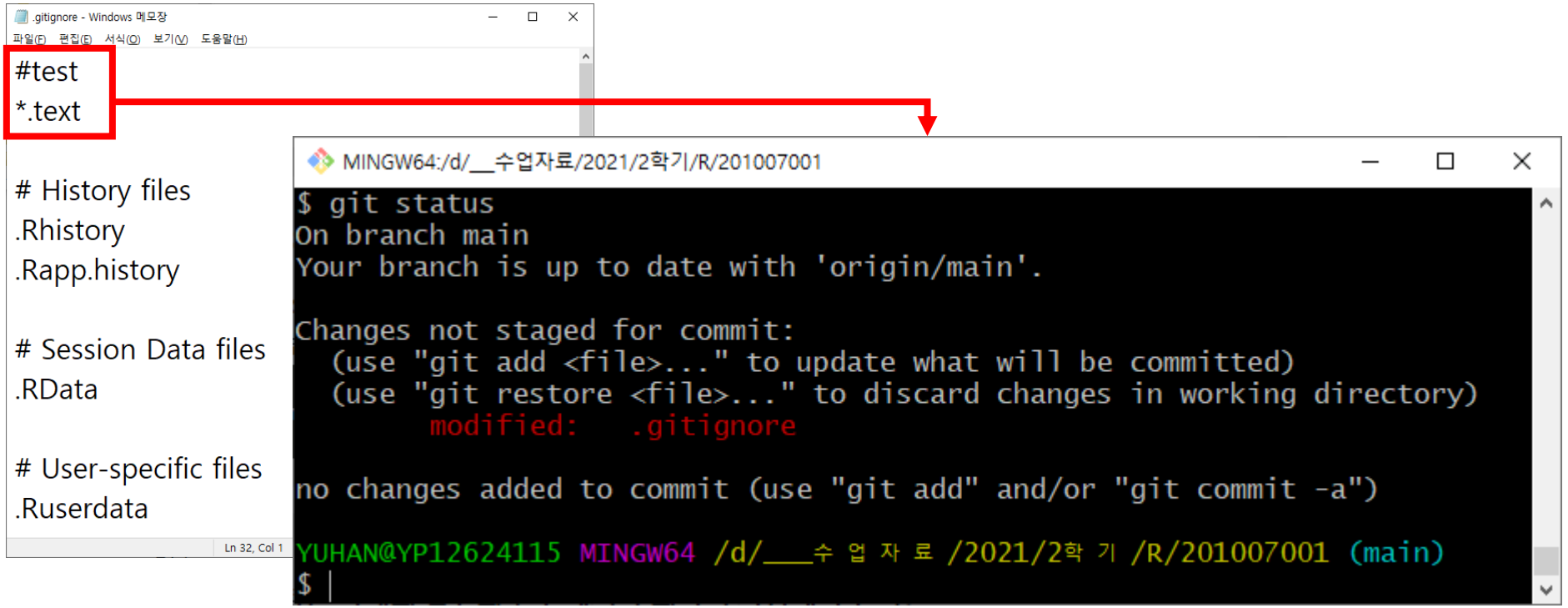
- **파일 버전 관리 예외 정보 기록**
  - **“.”으로 시작하는 이유는 리눅스를 기반으로 하기 때문**
  - **리눅스에서는 파일명이 “.”으로 시작하면 숨김 파일**
  - **굳이 사용자에게 노출할 필요가 없으므로 숨김 처리**
  - **윈도우에서도 숨김 속성을 가지도록 구성**





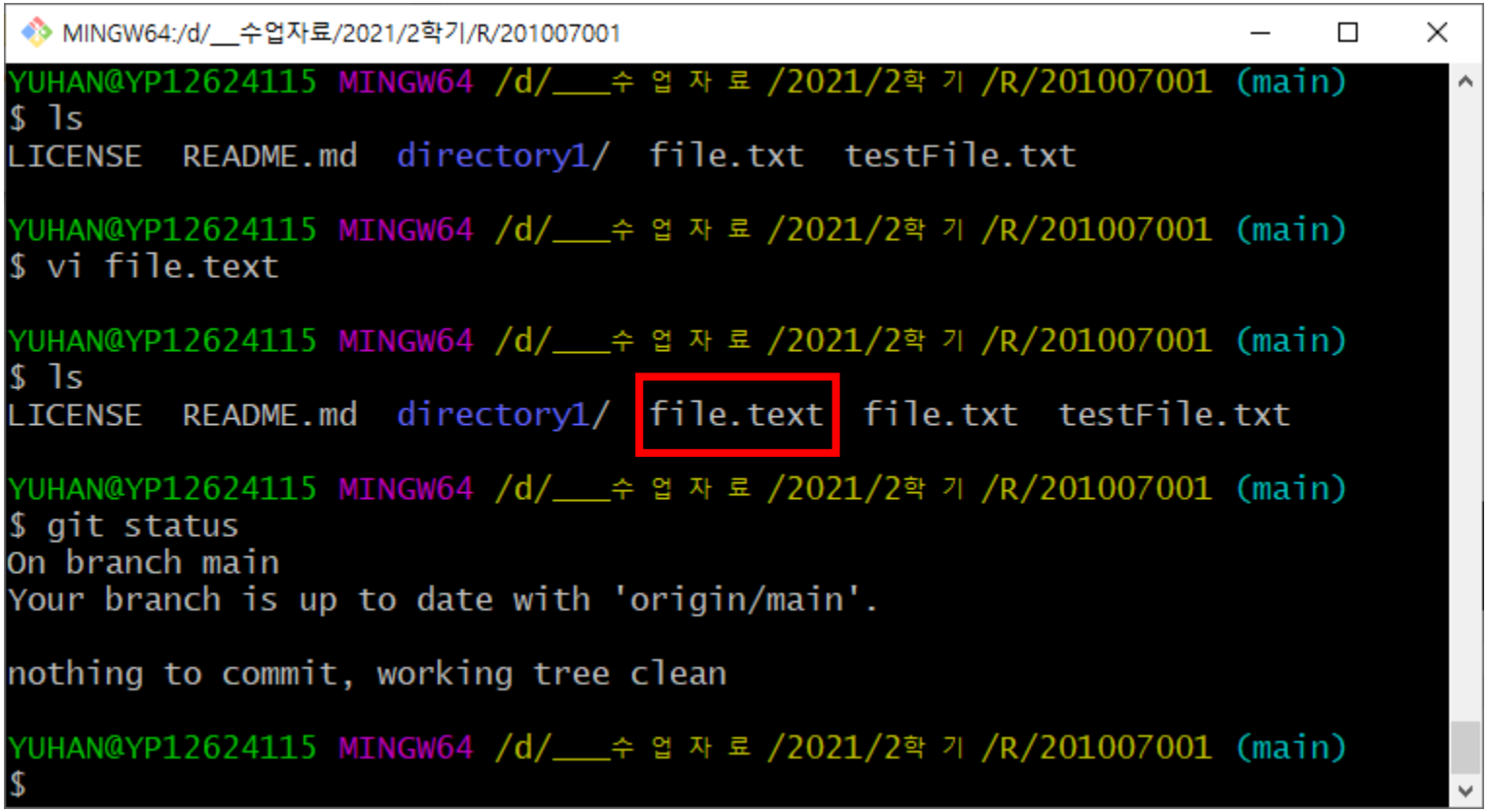
.gitignore 파일 – 2

- **파일과 디렉터리 모두 지정 가능**
  - #으로 시작되면 주석으로 아무 처리하지 않음
  - 리눅스의 상대, 절대 경로를 이용하여 지정도 가능
  - 예외로 지정되면 어떠한 추가/수정/삭제가 발생해도 관리하지 않음
  - .gitignore도 파일이기 때문에 수정하면 반드시 commit해야 함



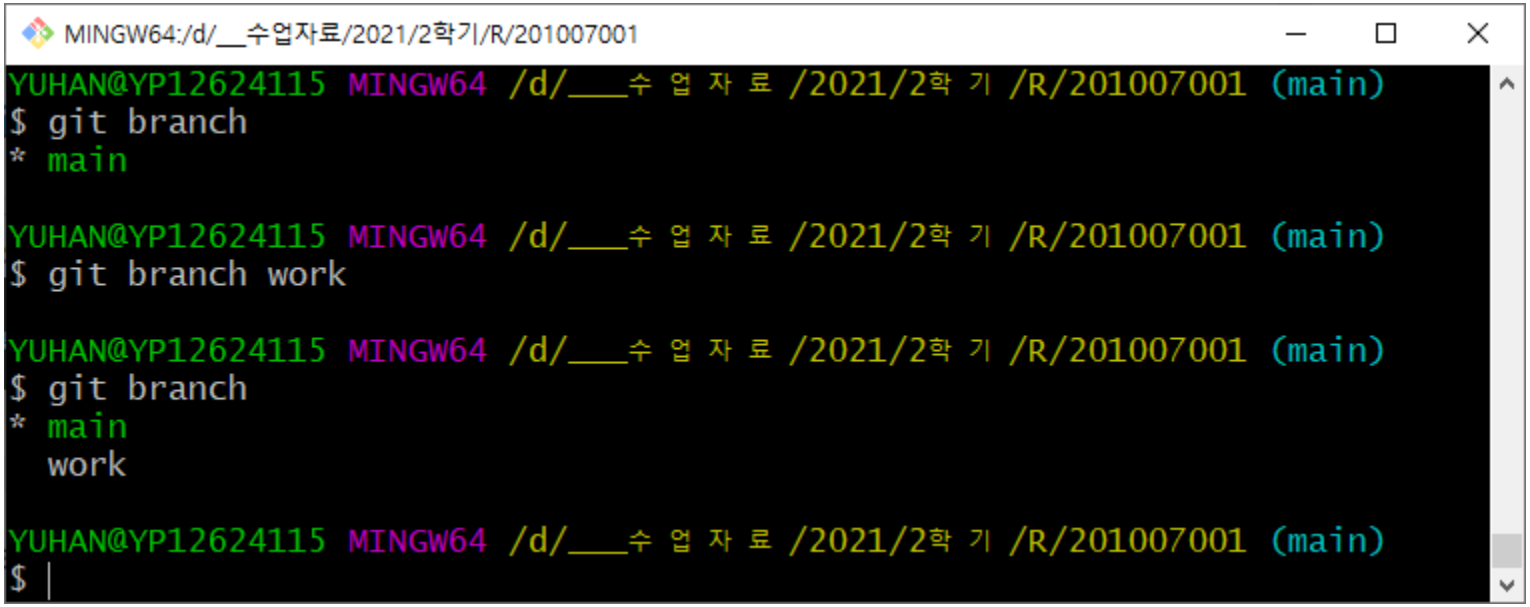
.gitignore 파일 – 3

- **파일과 디렉터리 모두 지정 가능**
  - **확장자가 ".text"라면 파일에 대한 추적을 수행하지 않음**
  - **새롭게 파일을 추가하더라도 추적 대상에 포함되지 않음**
  - **디버깅 또는 임시 파일들을 추가하지 않도록 구성할 때 사용**



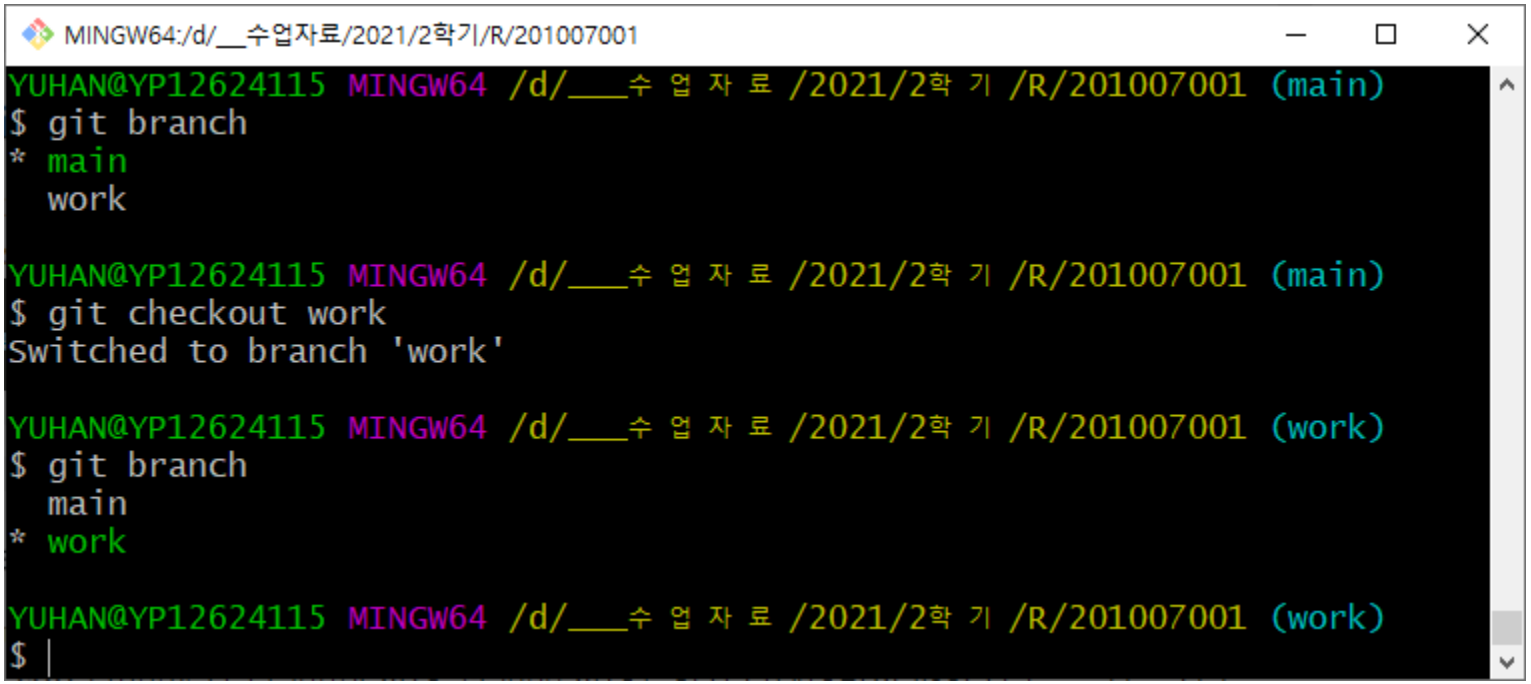
git branch 실습 – 1

- **브렌치 생성과 확인 명령어**
  - **현재 브렌치 정보 확인** : git branch
  - **현재 작업중인 브렌치는 앞에 "*"이 출력됨**
  - **새로운 논리 파일 관리를 위해 브렌치를 생성** : git branch work
  - **브렌치만 생성한 경우는 여전히 현재 브렌치가 작업 브렌치로 설정**



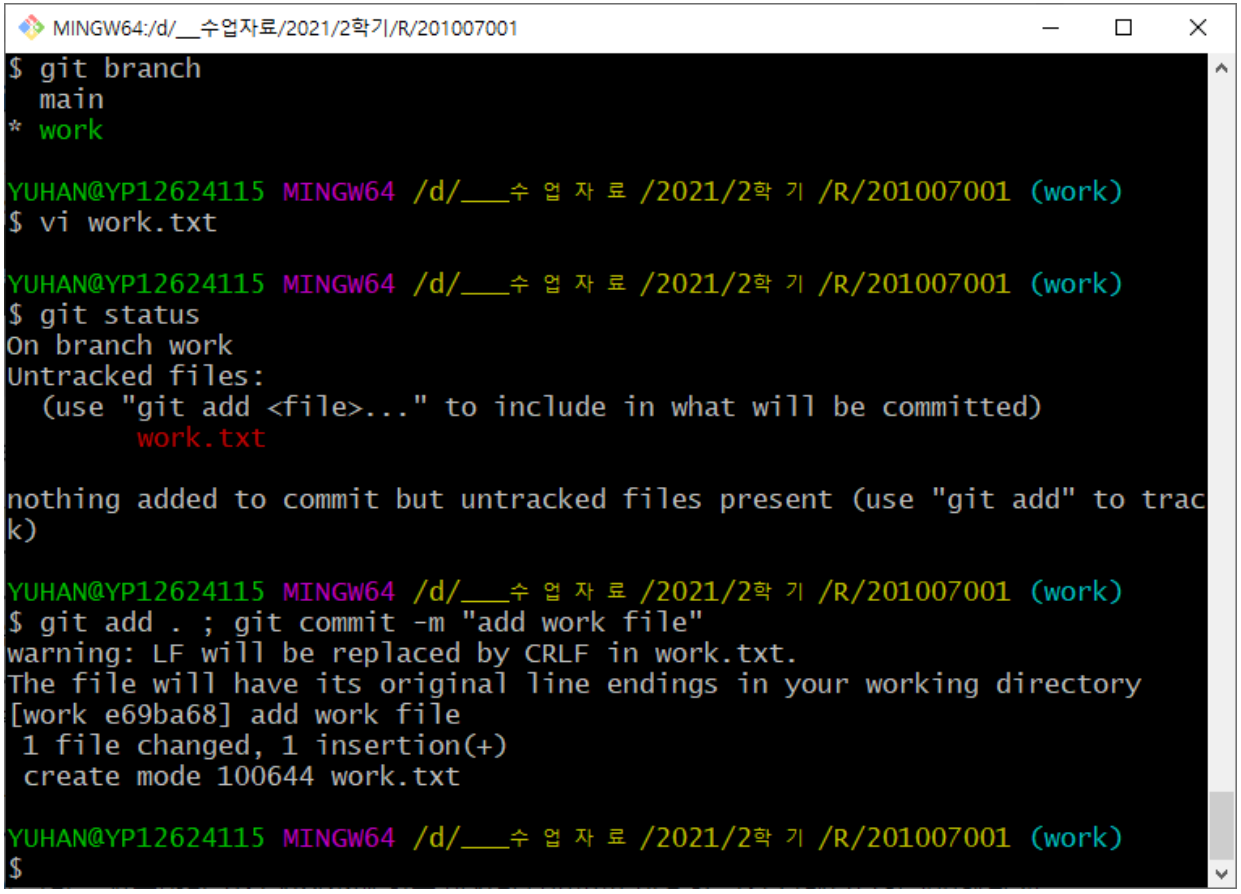
git branch 실습 – 2

- **브렌치 이동**
  - **현재 브렌치 정보 확인** : git branch
  - **다른 브렌치로 이동** : git checkout "**브렌치명**"
  - **이동한 후에 브렌치 정보를 확인하면 변경된 정보 확인 가능**



git branch 실습 – 3

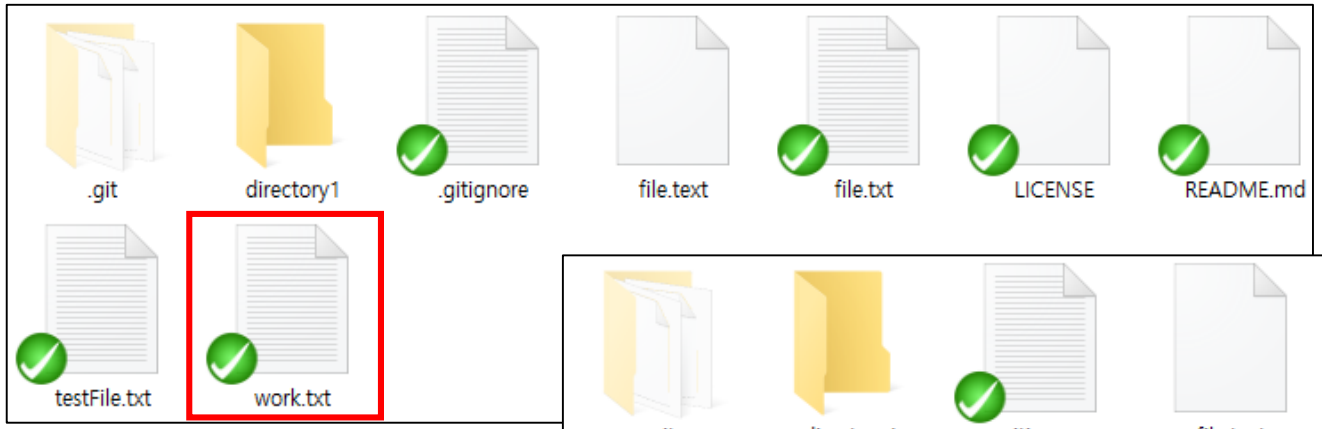
- **파일 수정 및** commit
  - **변경된 브렌치에서 새로운 파일을 추가 :** work.txt
  - **추가된 파일을** staging**하고**, commit **수행**

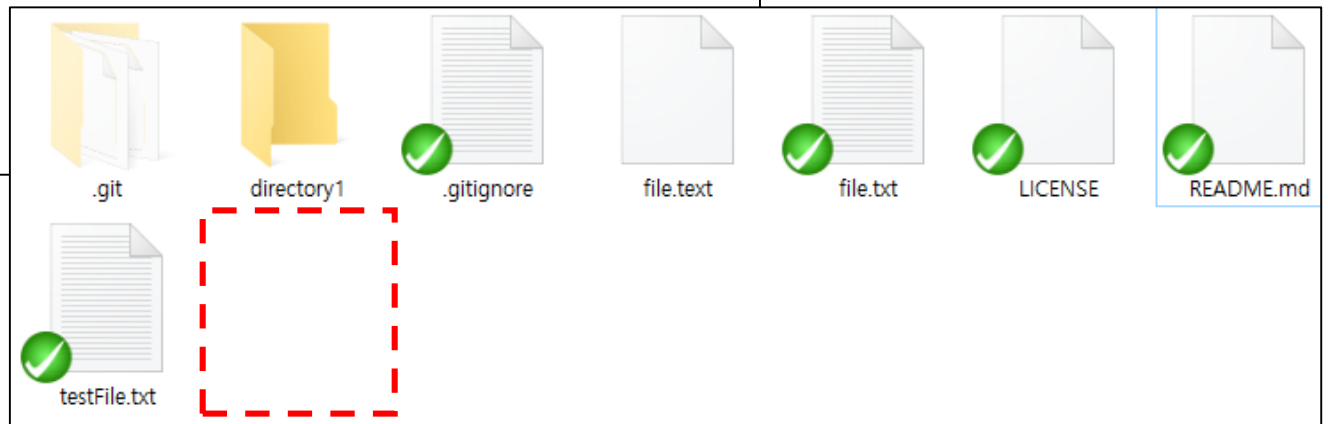


git branch 실습 – 4

- **브렌치를 이동해 가면서 파일 확인**

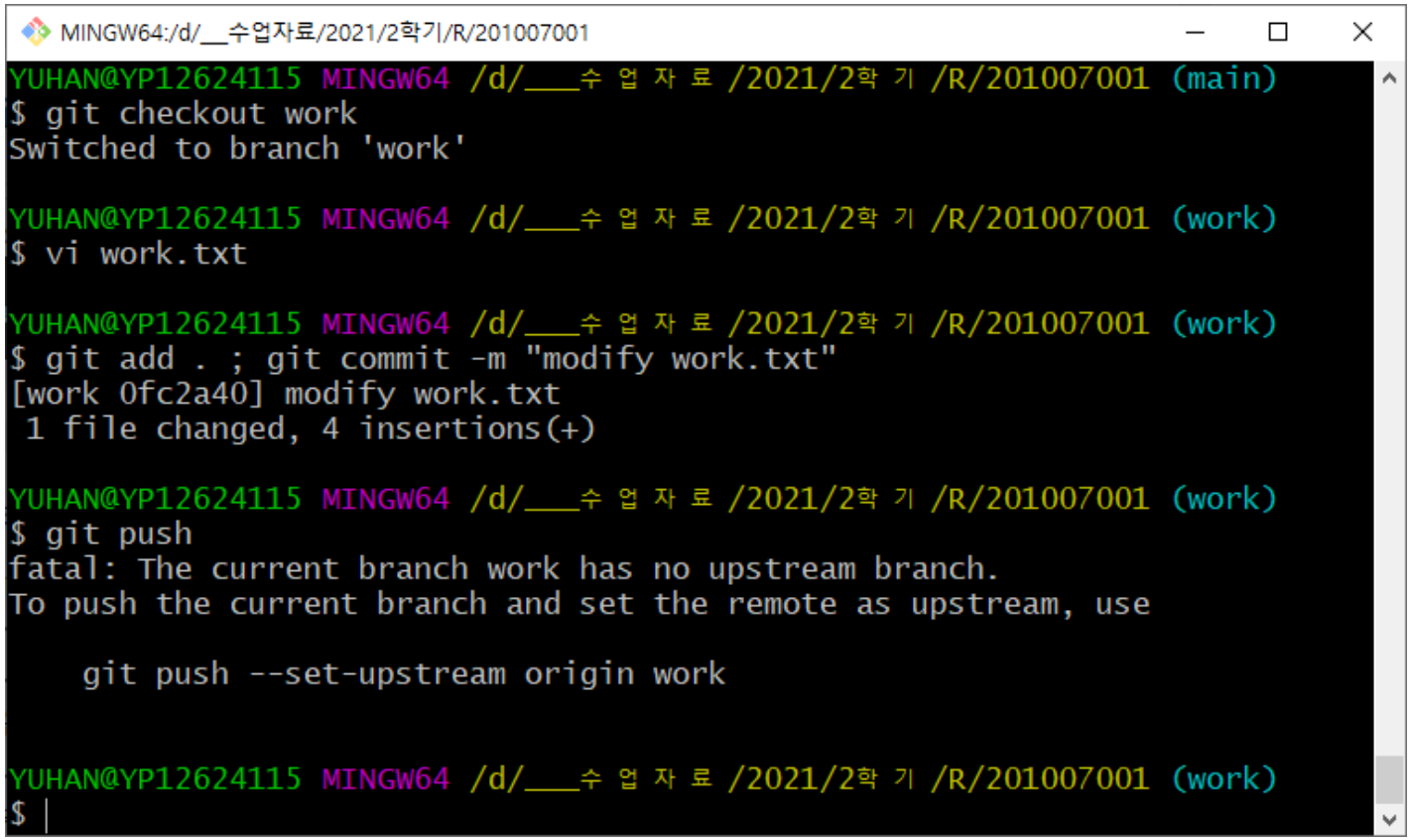
```
MINGW64:/d/__수업자료/2021/2학기/R/201007001
YUHAN@YP12624115 MINGW64 /d/___수 업 자 료 /2021/2학 기 /R/201007001 (work)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

YUHAN@YP12624115 MINGW64 /d/___수 업 자 료 /2021/2학 기 /R/201007001 (main)
$
```





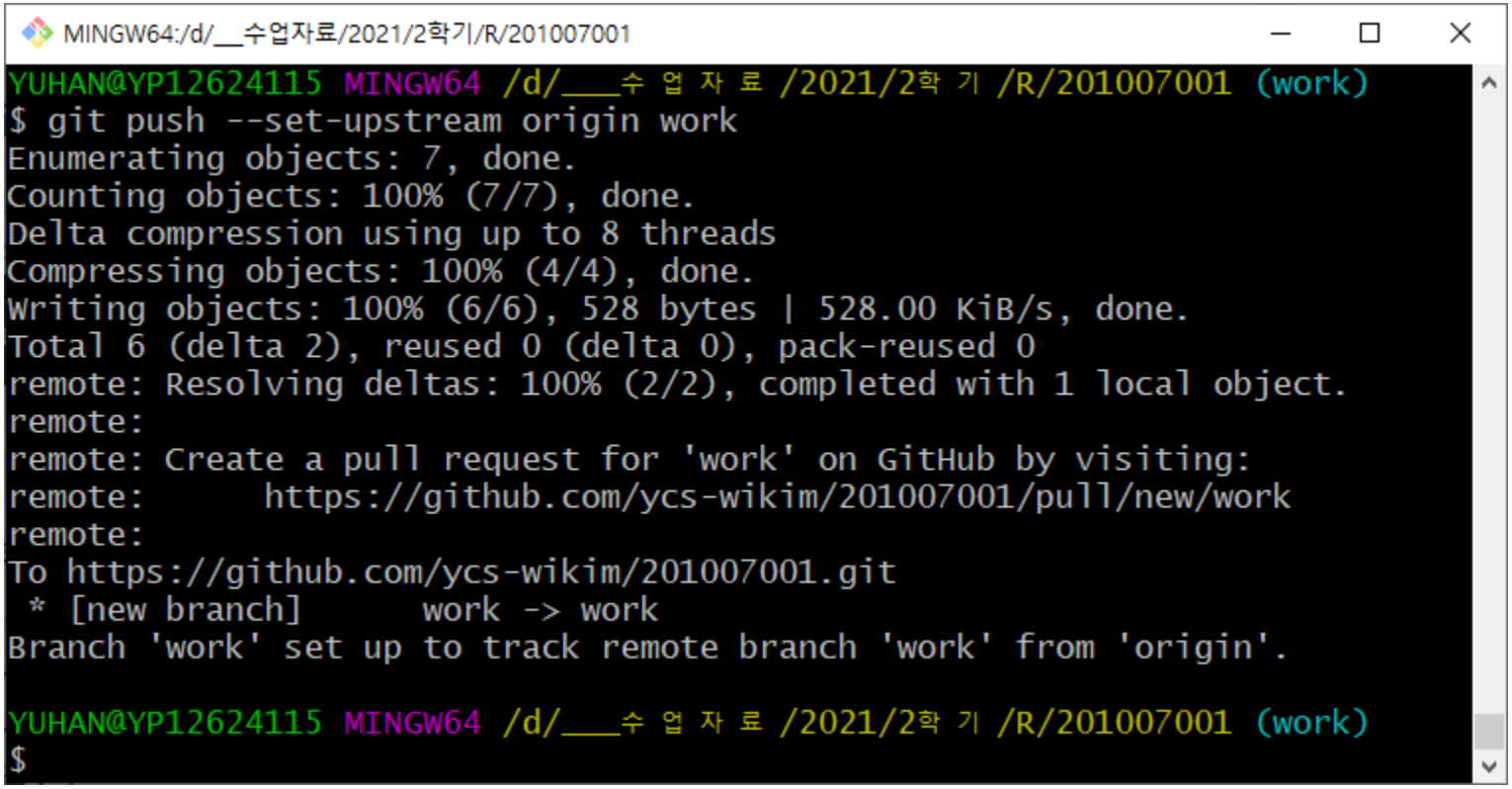
git branch 실습 – 5

- work 브렌치를 서버로 업로드
  - main 브렌치에서 work 브렌치로 이동 후 work.txt 파일을 수정
  - 수정된 내용을 commit, "git push"로 서버 업로드
  - 서버에 브렌치 정보가 없으므로, 서버 브렌치를 생성하면서 업로드



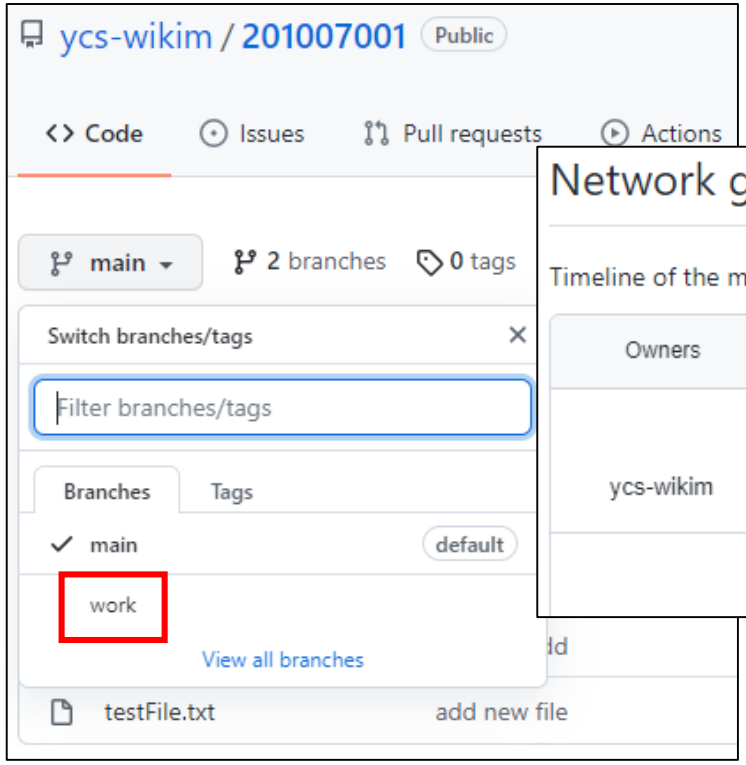
git branch 실습 – 6

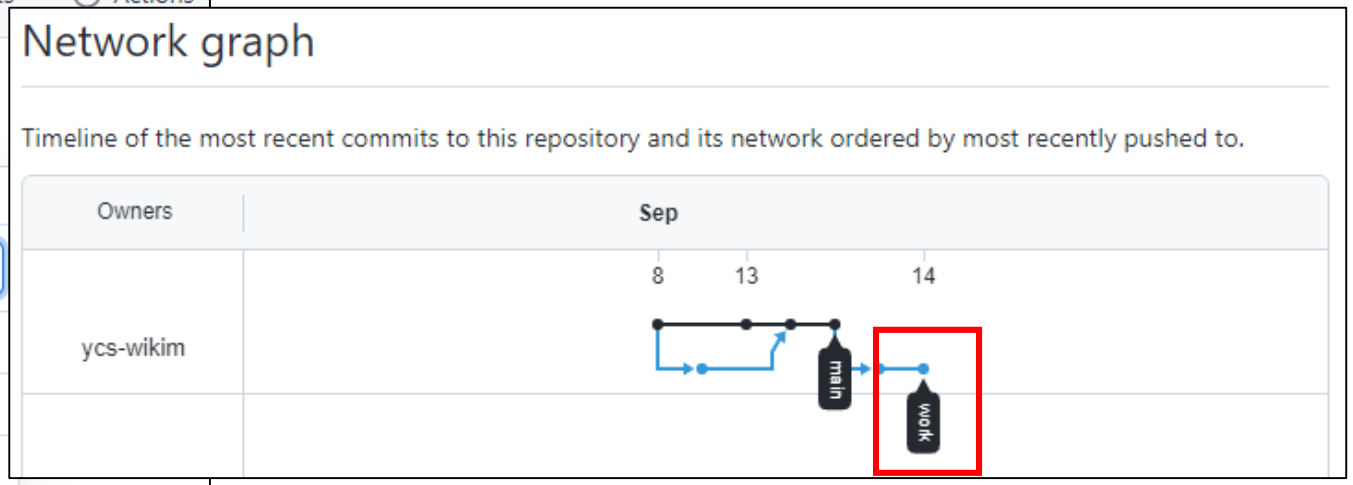
- **새로운 브렌치 생성과 업로드**
  - 서버에 새로운 브렌치가 생성되어 논리적인 파일 관리 정보가 업로드
  - 다른 사용자들도 "git pull"을 통해 해당 브렌치를 받아볼 수 있음
  - 중간 과정이나 정보를 공유하기 위한 용도로 사용 가능



git branch 실습 – 7

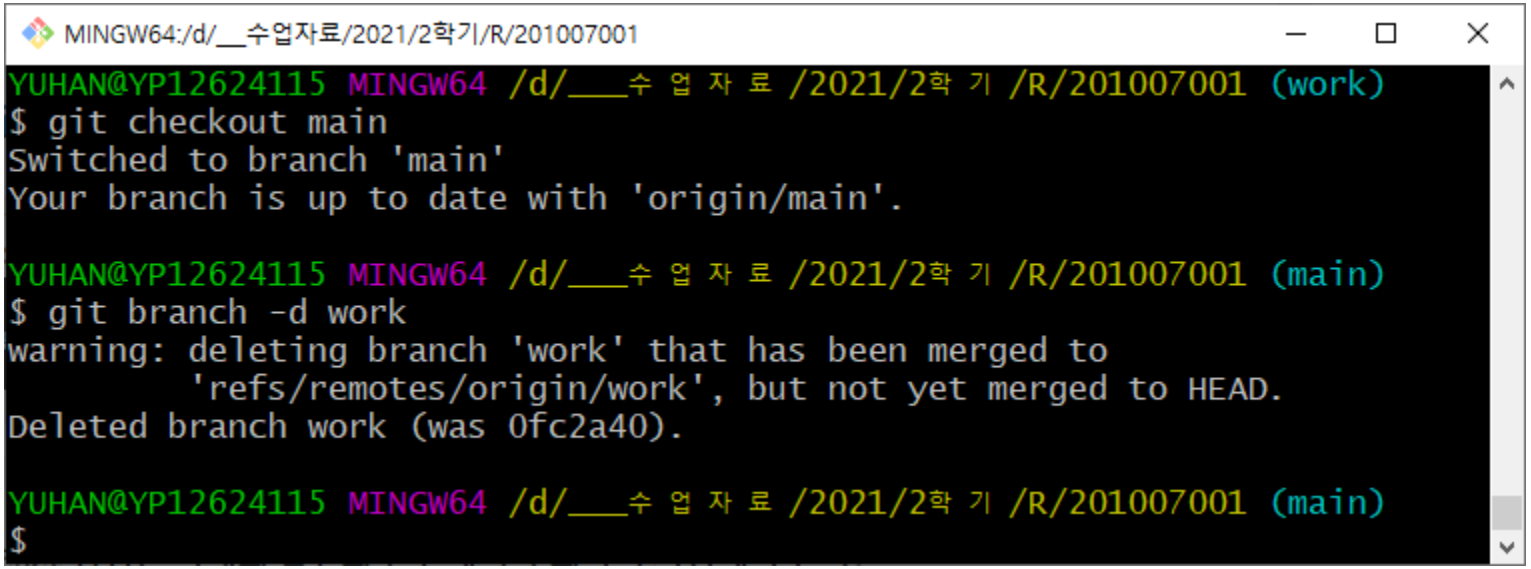
- **업로드 된 브렌치 정보 확인**
  - github.com의 저장소 정보에 새로운 브렌치 확인
  - 상단 메뉴에서 Insights → Network 메뉴로 브렌치 상태 확인
  - 새로운 브렌치 흐름을 볼 수 있음





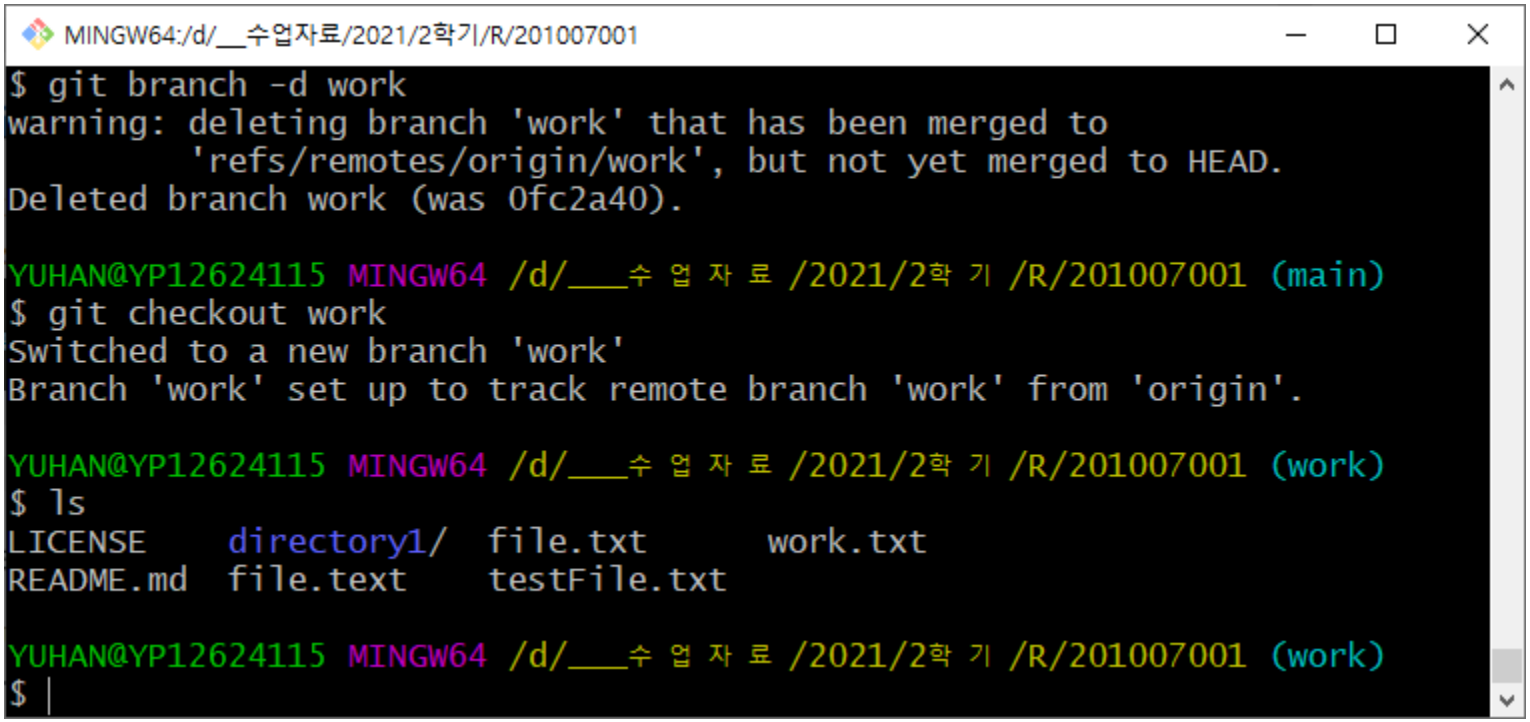
git branch 실습 – 7

- **브렌치 삭제** – 1
  - **로컬에서 브렌치 삭제 명령** : git branch –d "**삭제할 브렌치 이름**"
  - **로컬에서 병합되지 않은 브렌치 삭제**
  - **아직 병합되지 않아** main**에 적용되지 않았음이 경고로 출력**
  - **브렌치 삭제는 정상적으로 이루어짐**
  - **브렌치에서 작업하던 내용이 삭제되므로, 주의해야 함**



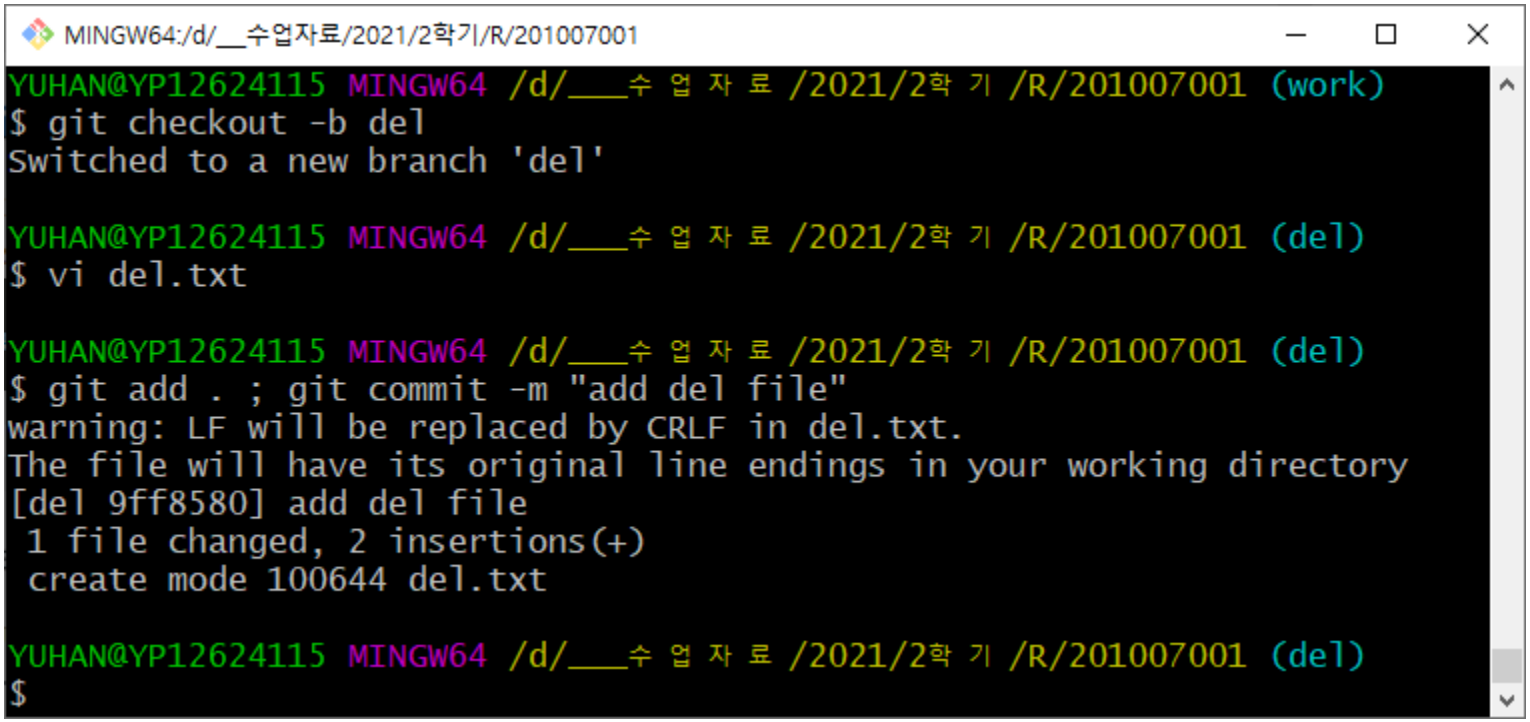
git branch 실습 – 8

- **브렌치 삭제** – 2
  - 서버에 업로드 된 브렌치는 복구가 가능
  - 로컬에서의 삭제는 로컬에서만 발생하였기 때문
  - "git checkout work"로 삭제되었던 브렌치로 이동
  - 원격지(서버)에 동일한 브렌치가 존재하므로, 서버 상태를 가져옴



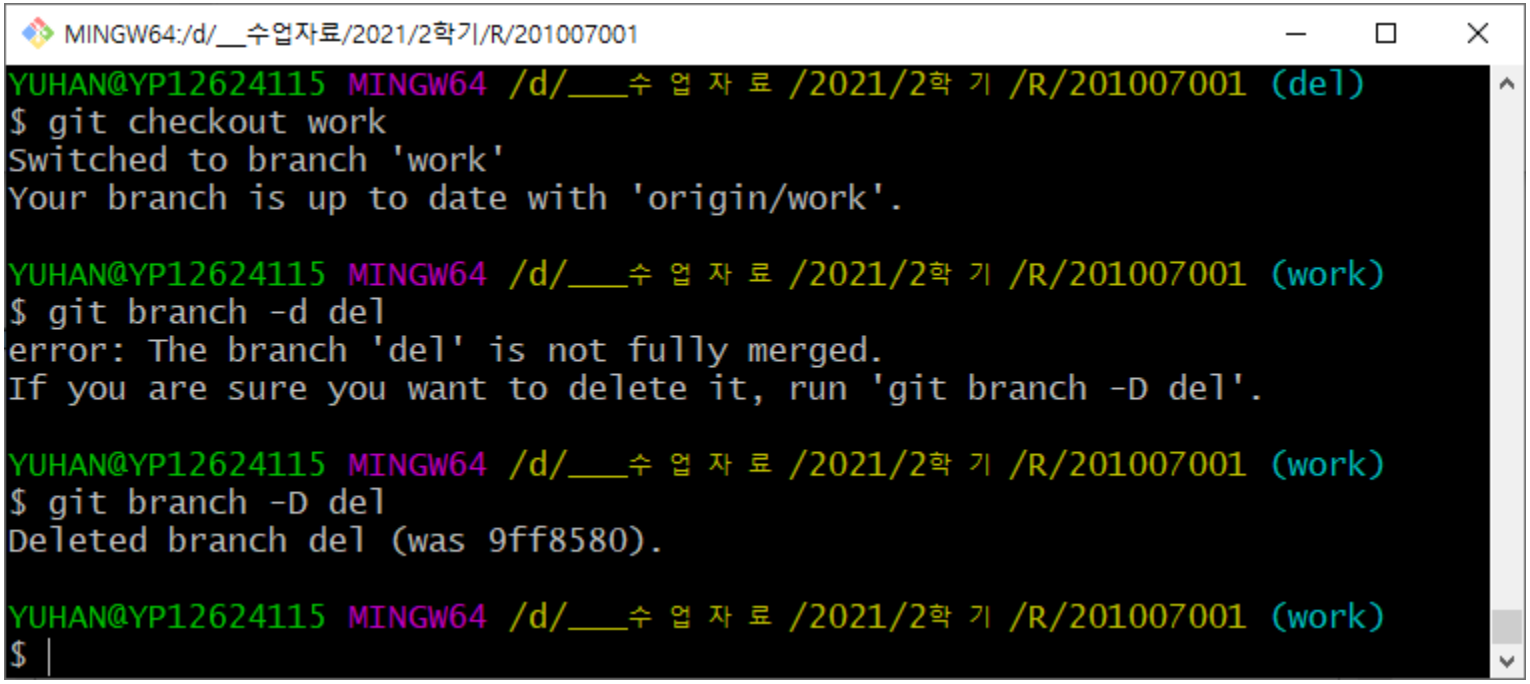
git branch 실습 – 9

- **브렌치 삭제 – 3**
  - **서버로는 업로드하지 않고, 병합하지 않은 브렌치 생성과 삭제**
  - **로컬에서만 작업하고 삭제할 새로운 브렌치 생성**
  - "git checkout –b del" **로 새로운 브렌치를 생성**
  - **새로운 파일을 추가하고**, commit **수행**



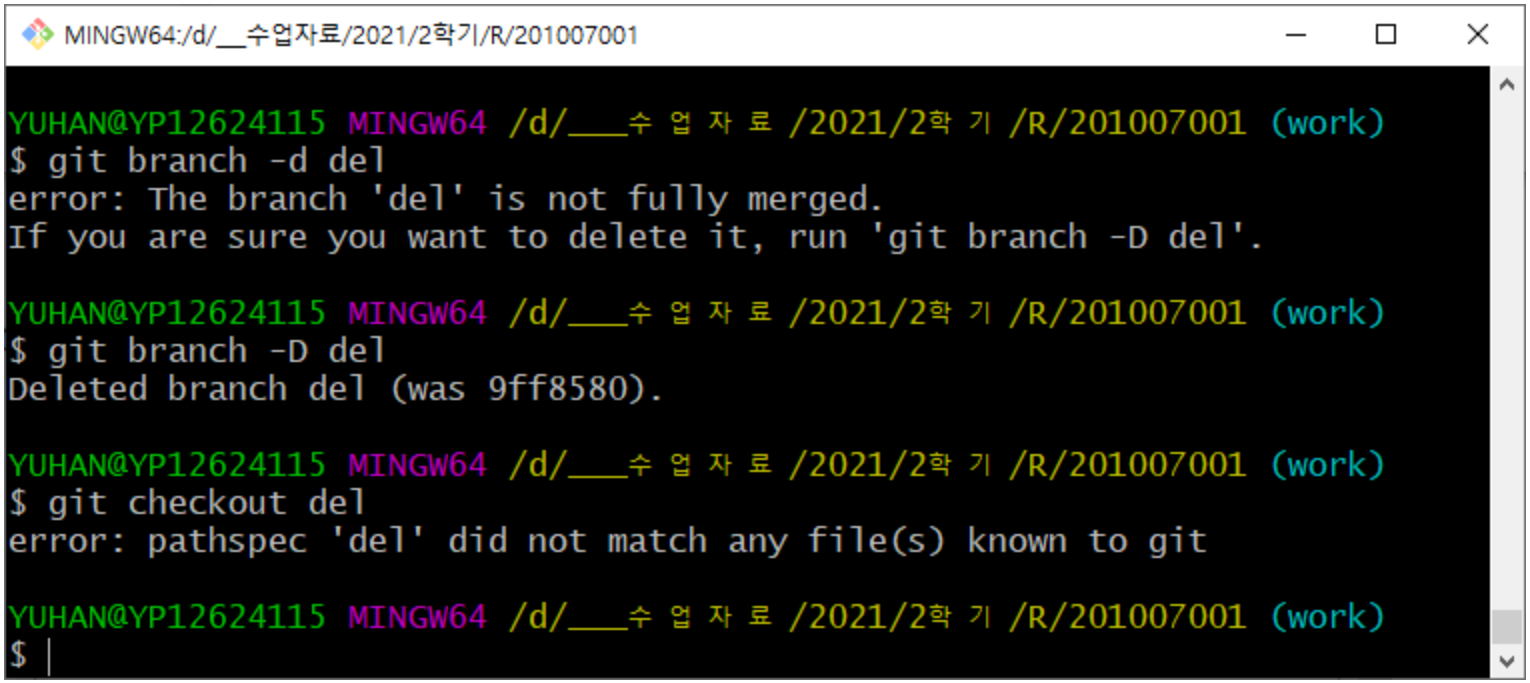
git branch 실습 – 10

- **브렌치 삭제** – 4
  - **다른 브렌치로 이동하여** del **브렌치를 삭제**
  - **병합되지 않은 로컬 브렌치 삭제에 대한 오류 메시지 출력**
  - **로컬에만 존재하는 브렌치는 완전히 삭제되기 때문에 오류로 처리**
  - **완전히 제거하려면** "git branch –D **삭제브렌치명"으로 삭제해야 함**



git branch 실습 – 11

- **브렌치 삭제 – 5**
  - **로컬에만 존재하는 브렌치의 삭제는 복구가 불가능**
  - **서버에 존재하는 것과 달리 "git checkout del"로 이동할 수 없음**
  - **기존 작업 내용은 모두 삭제되고, 복구가 불가능**



git 기초 명령어

- **저장소**
  - **git clone** : 서버 저장소를 로컬로 복제
  - **git pull** : 서버 저장소의 변경된 정보를 로컬로 다운로드
  - **git push** : 로컬 저장소를 서버로 업로드
- **파일 관리**
  - **git add** : 추가 또는 수정된 파일/디렉터리의 버전 관리 시작
    - Database에서 commit 전의 상태로 임시 저장된 staging 상태
  - **git commit** : stagin된 파일들을 commit 하고, 정보를 입력
  - **git commit –m** "commit 메시지" : 프로그램 없이 간단한 메시지 입력
  - **git reset** : commit 된 상태를 취소할 때의 명령
- **브렌치 관리**
  - **git branch** : 브렌치 관련 정보 출력/삭제 등의 명령 수행
  - **git checkout** : 존재하는 브렌치 간의 이동