
제품소프트웨어 패키징
(Git과 GitHub를 활용한 소스코드 관리)

# Git push를 활용한 소스코드 적용 : 소스코드 적용

한국기술교육대학교
온라인평생교육원


## 학습내용

- Git push 준비하기
- Tortoise Git에서 Git push

## 학습목표

- 로컬 영역에서 작업한 소스코드를 원격저장소에 저장하기 위한 환경을 설정한다.
- 로컬 영역에서 작업한 소스코드를 원격저장소에 저장하기 위한 Git 명령어를 학습한다.

# Git push 준비하기

## 1. Git push를 위한 환경 설정

### 1) 사용자 정보

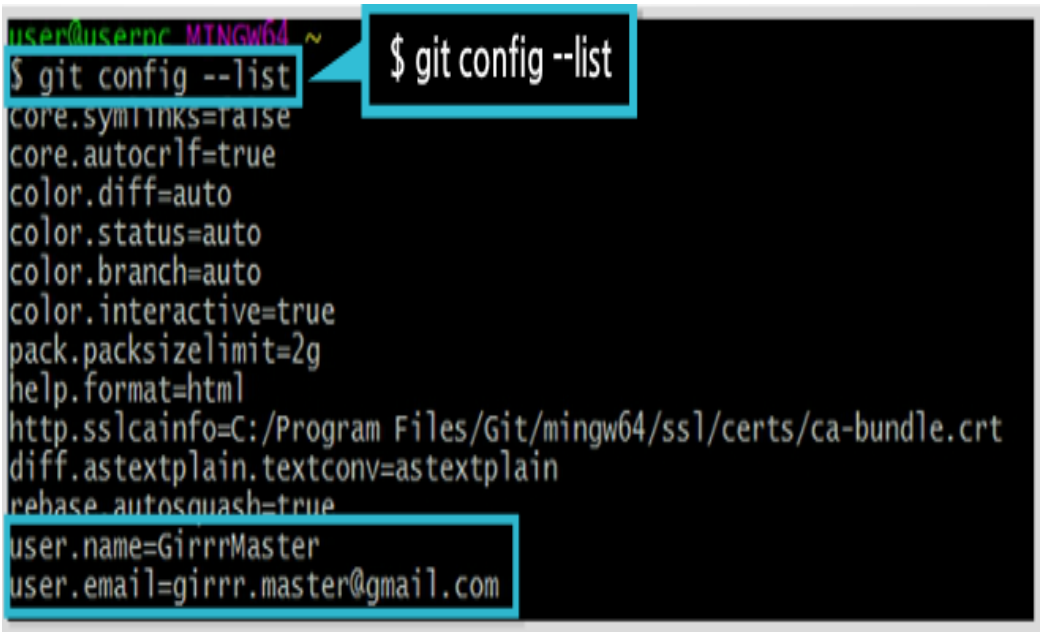
- Git을 사용하기 전, 사용자 정보 및 간단한 몇 가지 환경설정을 해야 함
  - Git config 명령어 이용
- Git config : 사용자의 이름과 이메일 주소 등록
  - 일반 포털 사이트의 아이디 역할
- Git config의 설정 : 사용자 이름은 user.name 이메일 주소는 user.email
- 명령어 입력

# Git push 준비하기

## 1. Git push를 위한 환경 설정

### 1) 사용자 정보

- 설정 값 확인
  - 정상적으로 저장
  - Git push를 할 때마다 저장된 정보로 소스코드 반영

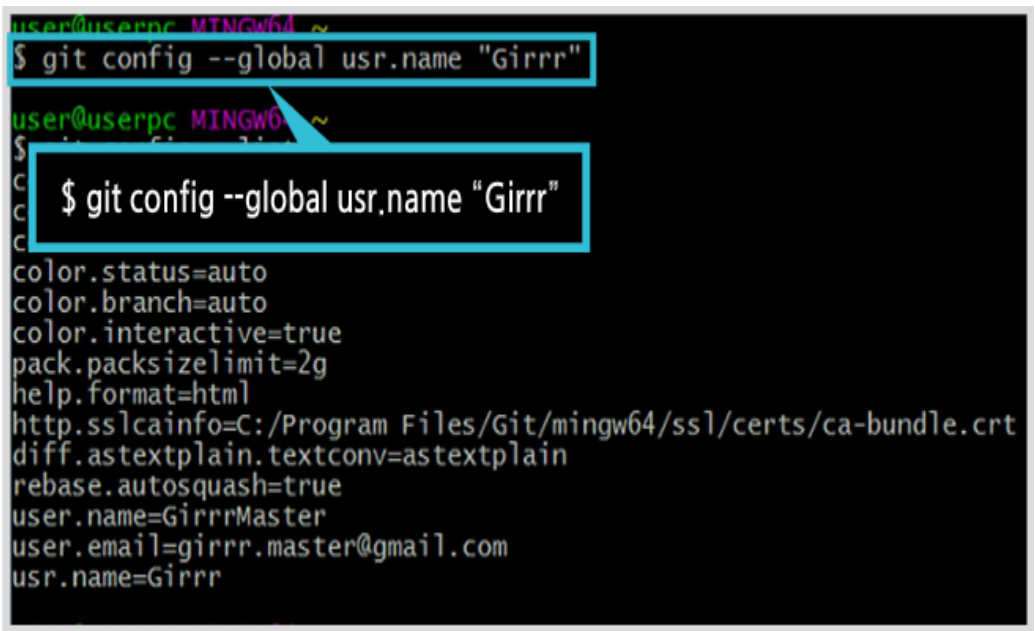


# Git push 준비하기

## 1. Git push를 위한 환경 설정

### 1) 사용자 정보

- 불필요한 설정 값 삭제

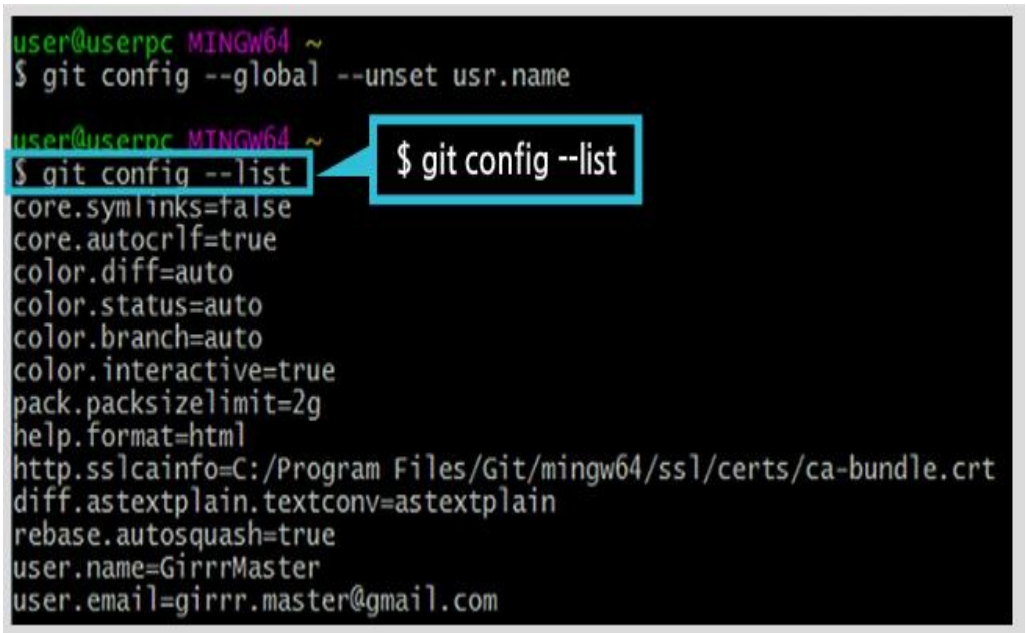



# Git push 준비하기

## 1. Git push를 위한 환경 설정

### 1) 사용자 정보

- 재등록

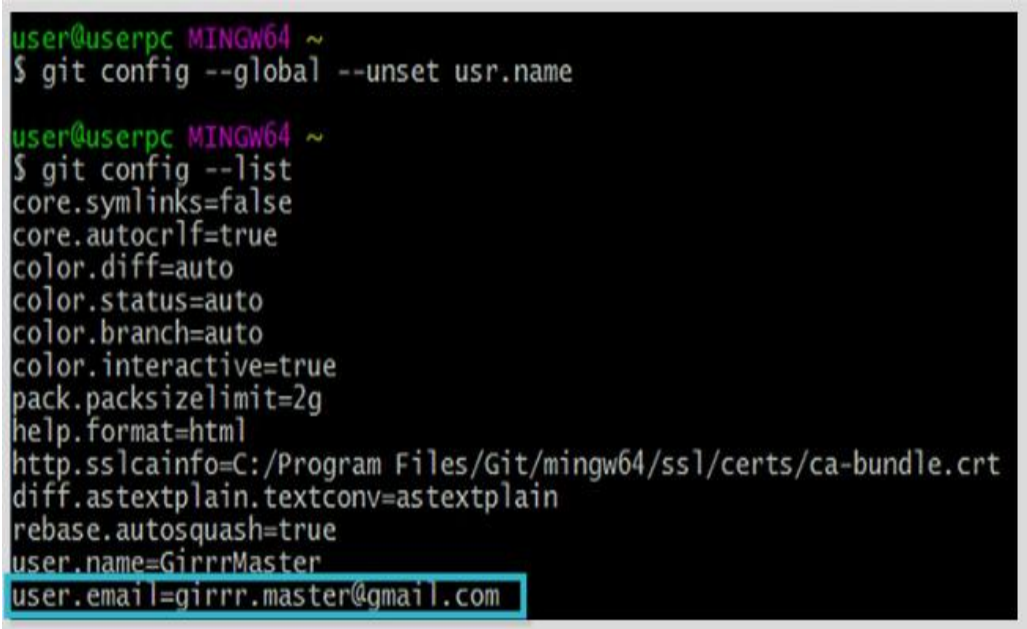



# Git push 준비하기

## 1. Git push를 위한 환경 설정

### 1) 사용자 정보

- 재등록
  - unset의 옵션을 설정하여 삭제 수행



## 1. Git push를 위한 환경 설정

### 2) 기타 환경 설정

- Git의 기본 인코딩은 윈도우 환경에 맞게 인코딩 작업을 해주어야 함
  - Git의 기본 commit 메시지 : utf-8
  - 윈도우의 명령 프롬프트 : cp949
  - commit 메시지가 깨지지 않고 제대로 입력
- 윈도우 사용자를 위한 추가 설정(인코딩)
  - Git commit 메시지의 기본 인코딩 : utf-8
  - 명령 프로프트의 기본 인코딩 : cp949
  - 한글 입력 시 글자가 깨져 보이는 문제를 해결하기 위해 commit 메시지와 log 메시지의 인코딩 방법 변경
- 윈도우 사용자를 위한 추가 설정(인코딩)

```
user@userpc MINGW64 ~
$ git config --global i18n.commitEncoding cp949

user@userpc MINGW64 ~
$ git config --global i18n.logOutputEncoding cp949

user@userpc MINGW64 ~
$ git config --list
core.symlinks=false
core.autocrlf=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
pack.packsizelimit=2g
help.format=html
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
diff.astextplain.textconv=astextplain
rebase.autosquash=true
user.name=GirrrMaster
user.email=girrr.master@gmail.com
i18n.commitencoding=cp949
i18n.logoutputencoding=cp949
```


# Git push 준비하기

## 1. Git push를 위한 환경 설정

### 2) 기타 환경 설정

- Git 컬러 사용하기
  - 각 단어나 특정 커멘드에 대한 색상 등을 지정하여 보다 직관적으로 활용
  - $ git config --global color.ui true
- 문서편집기 설정하기
  - Git에서 바로 편집을 하고자 할 때 해당하는 편집기로 소스코드가 열리도록 설정
  - $ git config --global core.editor “editor name”
- diff 툴 설정 하기
  - 기존의 소스코드와 이번에 새로 적용하는 코드의 변경 부분을 한눈에 확인
  - 툴의 종류가 워낙 다양하기 때문에, 어떤 프로그램을 사용할지 설정
  - $ git config --global merge.tool “tool name”

## 2. 소스코드 반영을 위한 Git 명령어

### 1) Git init

- Git 명령어를 통한 Git 사용법
  - 로컬저장소 생성
  - 파일 추가
  - 파일 변경
  - 원격저장소 설정
  - 원격저장소로 로컬저장소 데이터 등록 방법
- 기존 디렉토리를 Git 저장소로 만드는 명령어
  - 저장소로 만들기 위한 위치로 이동 : mkdir dos명령어로 myLocalRepo 디렉토리 생성
  - mkdir myLocalReo 폴더 생성 후 이동
  - $ git init 명령어로 저장소 생성

```
user@userpc MINGW64 /c/gitFolder
$ mkdir myLocalRepo

user@userpc MINGW64 /c/gitFolder
$ cd myLocalRepo/

user@userpc MINGW64 /c/gitFolder/myLocalRepo
$ git init
Initialized empty Git repository in C:/gitFolder/myLocalRepo/.git/
```

- 저장소 생성 완료 : 생성된 디렉토리 구조 확인

```
user@userpc MINGW64 /c/gitFolder/myLocalRepo (master)
$ ls -a
./  ../  .git/
```

- 저장소 생성 완료 : Git에서 파일을 관리하고 추적하는 데이터 저장

```
user@userpc MINGW64 /c/gitFolder/myLocalRepo (master)
$ ls -a
./  ../  .git/
```

## 2. 소스코드 반영을 위한 Git 명령어

### 2) Git add

- myLocalRepo 폴더에 파일 생성 - touch 명령어, 크기가 0인 파일 생성
  - $ touch '파일이름' : 크기가 0인 파일을 생성 : ex) $touch 'myLocalRepo.txt'
- myLocalRepo 폴더에 파일 생성 - touch 명령어 이외의 파일 생성방법
  - $ echo '파일내용' > '파일이름' : ex) $echo 'Add File' > 'myLocalRepo.txt'
  - $ vi '파일이름'(wq 종료) : ex) $ vi myLocalRepo.txt → (wq 종료)
  - 윈도우 상에서 마우스 우 클릭 새로운 만들기 → 텍스트 만들기

```
user@userpc MINGW64 /c/gitFolder/myLocalRepo (master)
$ touch 'myLocalRepo.txt'

user@userpc MINGW64 /c/gitFolder/myLocalRepo (master)
$ ls -a
./  ../  .git/  myLocalRepo.txt
```

- myLocalRepo.txt 파일이 생성되었으면 Git에 추가
  - $ git add를 이용하여 등록하지 않았다면 아직 해당 파일에 대한 변경사항 관리 및 추적 못함. 해당 디렉토리에는 존재하지만 Git에서는 관리하지 않는 상태
  - $ git add '파일이름' : ex) git add myLocalRepo.txt

```
user@userpc MINGW64 /c/gitFolder/myLocalRepo (master)
$ git add myLocalRepo.txt
```

## 2. 소스코드 반영을 위한 Git 명령어

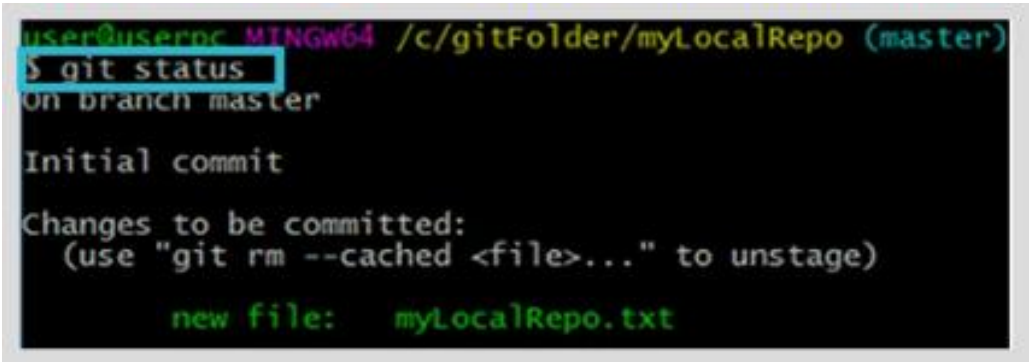
### 2) Git add

- 파일 이름을 생략하면 Git에 add 되지 않은 파일 모두 추가
  - $ git add *.txt → 모든 txt 파일을 add
  - $ git add *.* → 모든 파일을 add
  - 소스코드 중 변경된 사항을 모두 적용하는 경우가 대부분
  - 하나씩만 할 경우 더 헷갈리고 복잡해 질 수 있기 때문

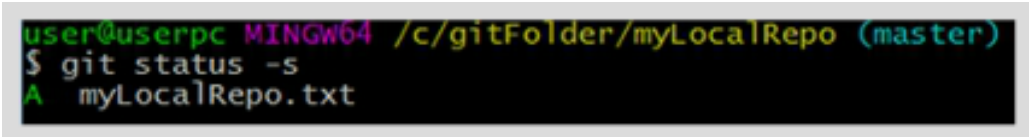
## 2. 소스코드 반영을 위한 Git 명령어

### 3) Git status

- Git에 상태 변화를 확인하는 명령어
  - Git에서 새롭게 파일이 등록되거나 수정된 정보 확인
- Git에 상태 변화를 확인하는 명령어 : $ git status
  - myLocalReop.txt 파일추가 상태 확인



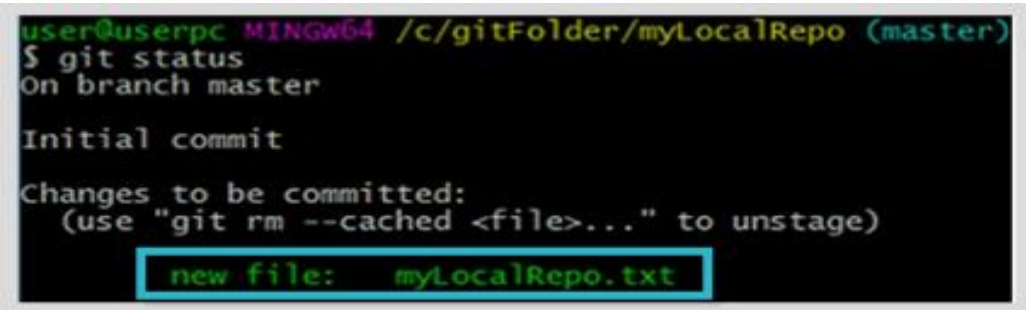

- Git에 상태 변화를 확인하는 명령어 : $ git status -s
  - $ git status -s 옵션으로 간략하게 내용 확인 가능



## 2. 소스코드 반영을 위한 Git 명령어

### 3) Git status

- Git에 상태 변화를 확인하는 명령어 : $ git status
  - 앞에서 추가한 파일이 new file로 생겼다는 것을 표시
  - 파일이 삭제되면 delete file로 기존에 있던 파일이 없어졌다는 것을 표시
  - modified된 파일의 리스트도 제시



## 2. 소스코드 반영을 위한 Git 명령어

### 4) Git commit -m

- git add를 통하여 인텍스에 추가된 사항을 로컬저장소에 등록하는 명령어
- myLocalRepo.txt 파일 commit
  - $ git commit -m “commit 메시지”
- $ git commit -m “commit message 1” 입력하여 myLocalRepo.txt 파일을 등록
  - commit message : Git에 등록할 때 사용자가 해당 등록은 어떤 것인지를 간략하게 설명하는 것
  - Git commit : 변경점을 등록하는 것. 특정 기능을 구현하는 단위나 버그를 수정하는 단위마다 수행

```
user@userpc MINGW64 /c/gitFolder/myLocalRepo (master)
$ git commit -m "commit message 1"
[master (root-commit) 3dc7e61] commit message 1
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 myLocalRepo.txt
```

- $ git status 명령으로 Git 상태 확인

```
user@userpc MINGW64 /c/gitFolder/myLocalRepo (master)
$ git status
On branch master
nothing to commit, working directory clean
```

## 2. 소스코드 반영을 위한 Git 명령어

### 5) Git commit -a -m

- git add 생략하고 추가된 사항을 로컬저장소에 바로 등록
- 파일 생성

```
user@userpc MINGW64 /c/gitFolder/myLocalRepo (master)
$ touch myLocalRepo2.txt
```

- 생성한 파일을 -a 옵션을 이용하여 commit
- 해당 파일을 index에 추가하지 않고 바로 Git 의 Local 저장소에 등록

```
user@userpc MINGW64 /c/gitFolder/myLocalRepo (master)
$ git commit -a -m "commit message2"
On branch master
Untracked files:
        myLocalRepo2.txt

nothing added to commit but untracked files present
```

- Git에서 untracked files을 제공한다는 메시지 출력
- 저장하기 위해서는 git add, commit 명령어를 이용해서 추가

```
$ git add myLocalRepo2.txt        git add

user@userpc MINGW64 /c/gitFolder/myLocalRepo (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   myLocalRepo2.txt
```

- commit 수행

```
user@userpc MINGW64 /c/gitFolder/myLocalRepo (master)
$ git commit -m "commit message3 - add file and commit"
[master 98ad03e] commit message3 - add file and commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 myLocalRepo2.txt
```

## 2. 소스코드 반영을 위한 Git 명령어

### 6) Git rm

- Git 파일삭제 명령어를 통해야만 실제 working directory에서도 파일이 삭제됨
- myLocalRepo2.txt 삭제
  - 기존에 파일 삭제 하는 방법으로 파일 삭제 시, Git : “파일삭제 된 상태”라고 출력
  - Git 자체 등록된 파일을 삭제하지 않음
- delete 키 혹은 rm(dos 명령어)으로 삭제를 해도 Git에서는 삭제가 안됨
  - 단순히 파일을 컴퓨터에서 삭제 → 해당 파일이 삭제된 새로운 변경점이 발생
  - Git rm 명령을 통해 삭제 → 추가되었던 파일의 변경점이 취소되고 삭제전의 상태
  - 사용자의 실수로 인하여 파일이 삭제되어 복구 할 수 없는 상태가 되는 것 방지
- $ git rm myLocalRepo2.txt 명령으로 삭제

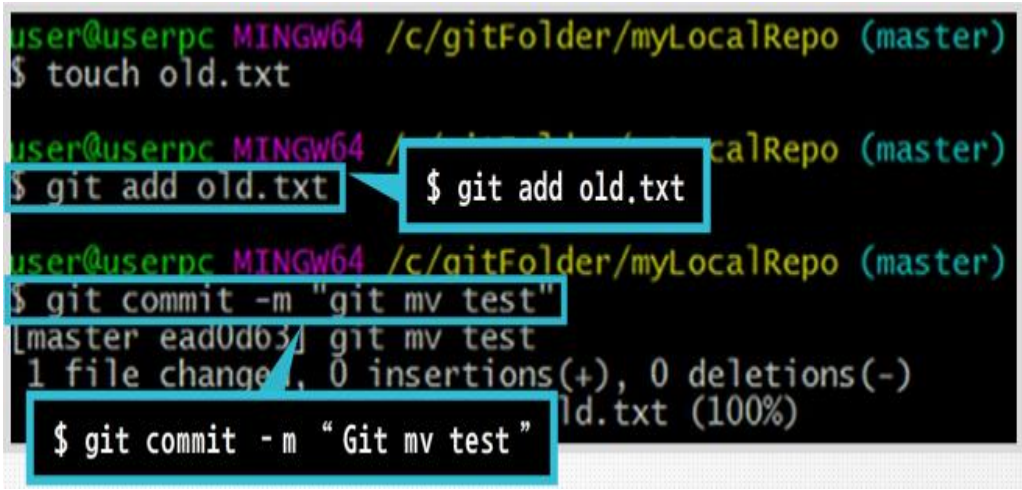
```
user@userpc MINGW64 /c/gitFolder/myLocalRepo (master)
$ git rm myLocalRepo2.txt
rm 'myLocalRepo2.txt'
```

## 2. 소스코드 반영을 위한 Git 명령어

### 7) Git mv

- 파일의 이름을 변경하는 명령어
- 파일 생성

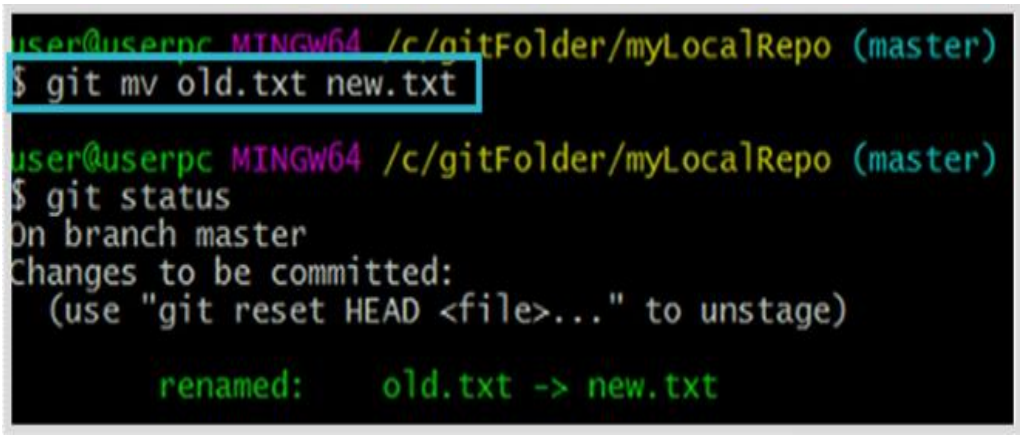

- add, commit 명령어로 저장소에 등록



## 2. 소스코드 반영을 위한 Git 명령어

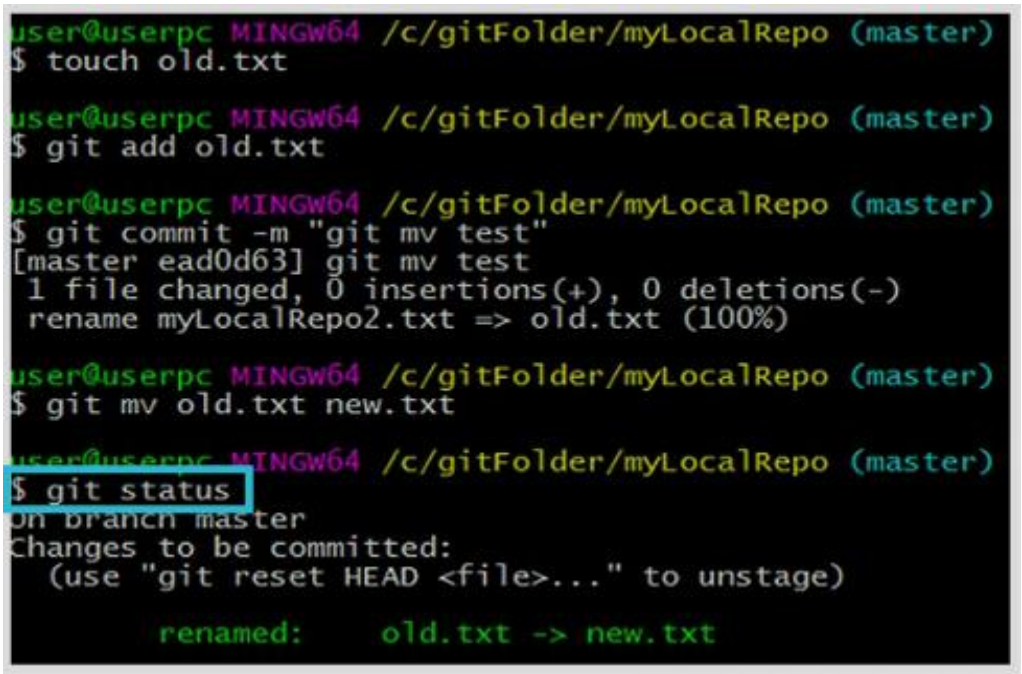
### 7) Git mv

- old.txt 에서 new.txt 로 변경




- old.txt 파일을 new.txt 파일로 복사
- old.txt 파일을 삭제
- new.txt 파일을 add commit 하는 과정을 내부적으로 수행

- $ git status로 상태 확인



## 2. 소스코드 반영을 위한 Git 명령어

### 8) Git log

- Git에서 commit 히스토리를 조회하는 명령어

```
user@userpc MINGW64 /c/gitFolder/myLocalRepo (master)
$ git log
commit ead0d634a43f1f0db0564d1b279c5a5cc71031ea
Author: GirrrMaster <girrr.master@gmail.com>
Date:   Fri Nov 20 14:45:17 2015 +0900

    git mv test

commit 98ad03e09ae806386c6635e7ae20bf7d376bfc16
Author: GirrrMaster <girrr.master@gmail.com>
Date:   Fri Nov 20 14:11:54 2015 +0900

    commit message3 - add file and commit

commit 3dc7e61fcb0286359d2f306efc5c0cdcabfa1d93
Author: GirrrMaster <girrr.master@gmail.com>
Date:   Fri Nov 20 13:46:51 2015 +0900

    commit message 1
```

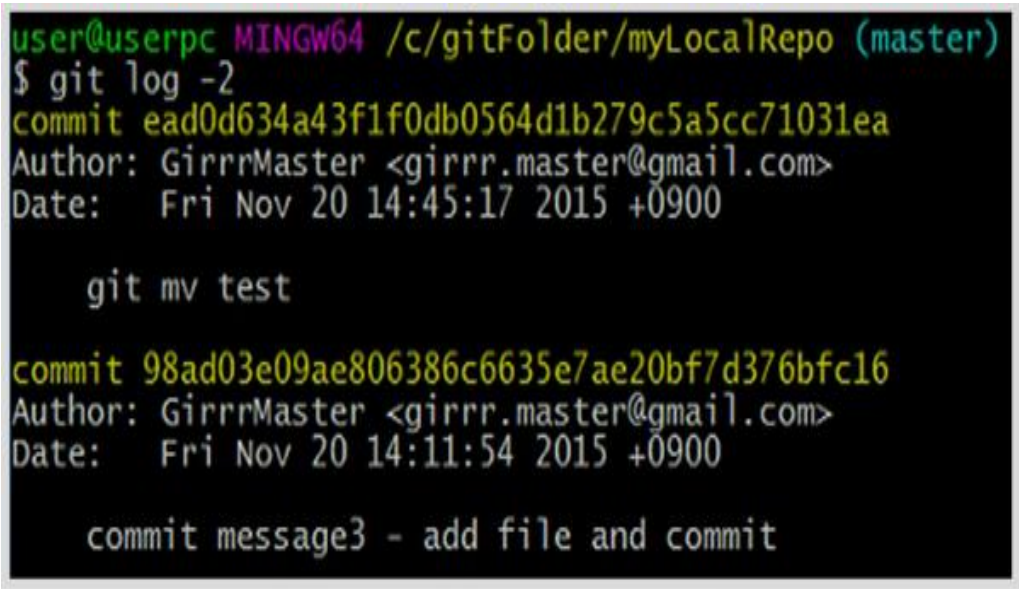
- $ git log -p : 각 commit의 결과를 비교하여 보여줌

```
user@userpc MINGW64 /c/gitFolder/myLocalRepo (master)
$ git log -p
commit ead0d634a43f1f0db0564d1b279c5a5cc71031ea
Author: GirrrMaster <girrr.master@gmail.com>
Date:   Fri Nov 20 14:45:17 2015 +0900

    git mv test

diff --git a/myLocalRepo2.txt b/myLocalRepo2.txt
deleted file mode 100644
index e69de29..0000000
diff --git a/old.txt b/old.txt
new file mode 100644
index 0000000..e69de29

commit 98ad03e09ae806386c6635e7ae20bf7d376bfc16
Author: GirrrMaster <girrr.master@gmail.com>
Date:   Fri Nov 20 14:11:54 2015 +0900

    commit message3 - add file and commit

diff --git a/myLocalRepo2.txt b/myLocalRepo2.txt
new file mode 100644
index 0000000..e69de29

commit 3dc7e61fcb0286359d2f306efc5c0cdcabfa1d93
Author: GirrrMaster <girrr.master@gmail.com>
Date:   Fri Nov 20 13:46:51 2015 +0900

    commit message 1

diff --git a/myLocalRepo.txt b/myLocalRepo.txt
new file mode 100644
index 0000000..e69de29
```

## 2. 소스코드 반영을 위한 Git 명령어

### 8) Git log

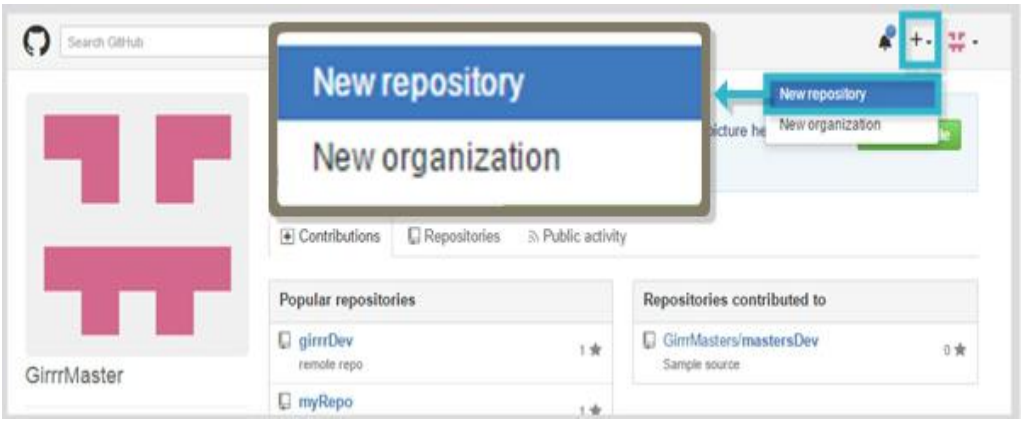
- $ git log -2 : 최근 두 개의 결과만 보여줌



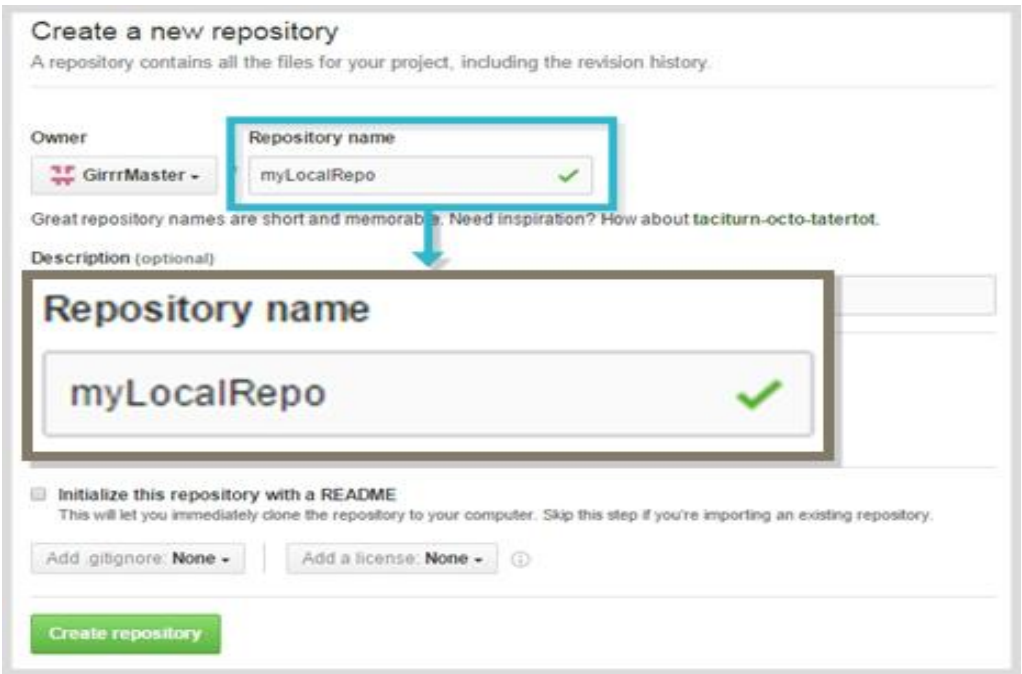
## 2. 소스코드 반영을 위한 Git 명령어

### 9) Git push를 위한 GitHub 원격저장소 만들기

- 원격저장소에 저장소를 만들어 로컬저장소의 데이터 등록
- GitHub 페이지 상단의 + 버튼의 New repository 클릭



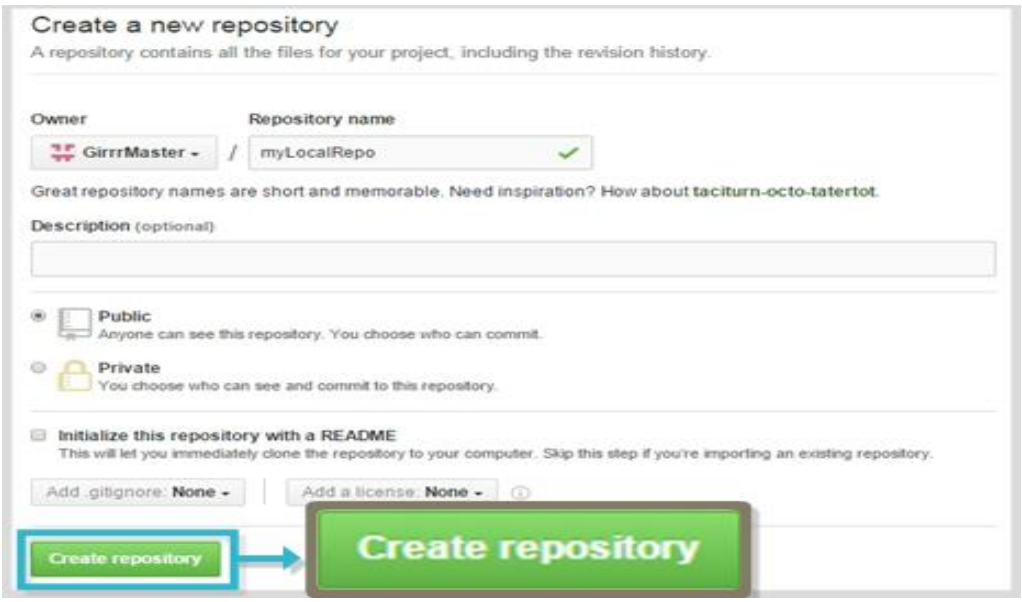

- Repository name에 myLocalRepo 라고 입력하고 생성



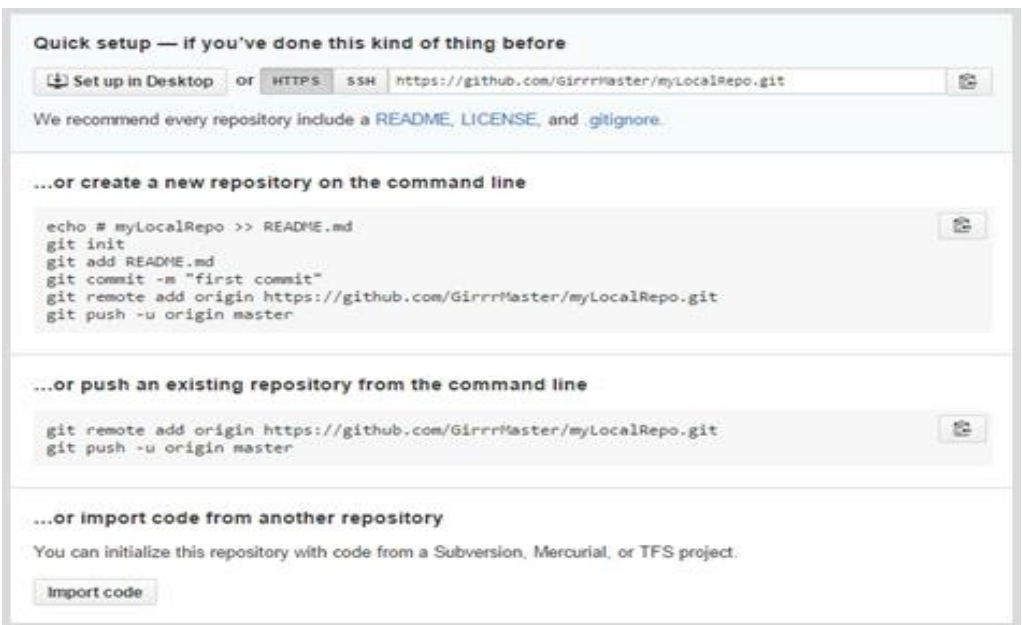
## 2. 소스코드 반영을 위한 Git 명령어

### 9) Git push를 위한 GitHub 원격저장소 만들기

- Repository name에 myLocalRepo 라고 입력하고 생성




- 생성 완료 화면에 기존에 있는 저장소에서 push 하는 방법 확인



## 2. 소스코드 반영을 위한 Git 명령어

### 10) Git remote

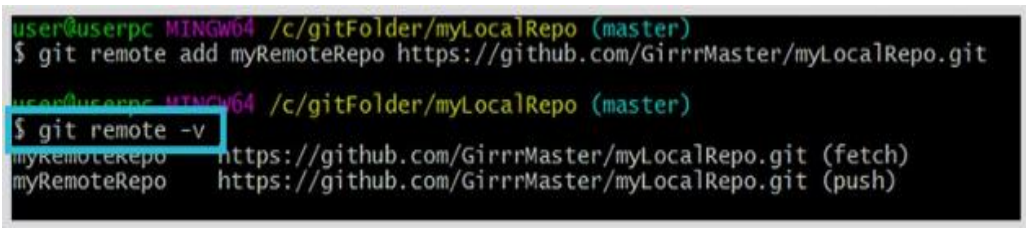
- 저장소의 정보를 추가, 삭제하는 명령어
- Git remote -v

  - $ git remote -v 명령어로 Git 에 등록된 저장소를 확인


  - 원격저장소에 대한 정보 없음

- Git remote add 단축이름 URL

  - $ git remote add myRemoteRepo url 명령으로 remote 저장소 등록



# Git push 준비하기

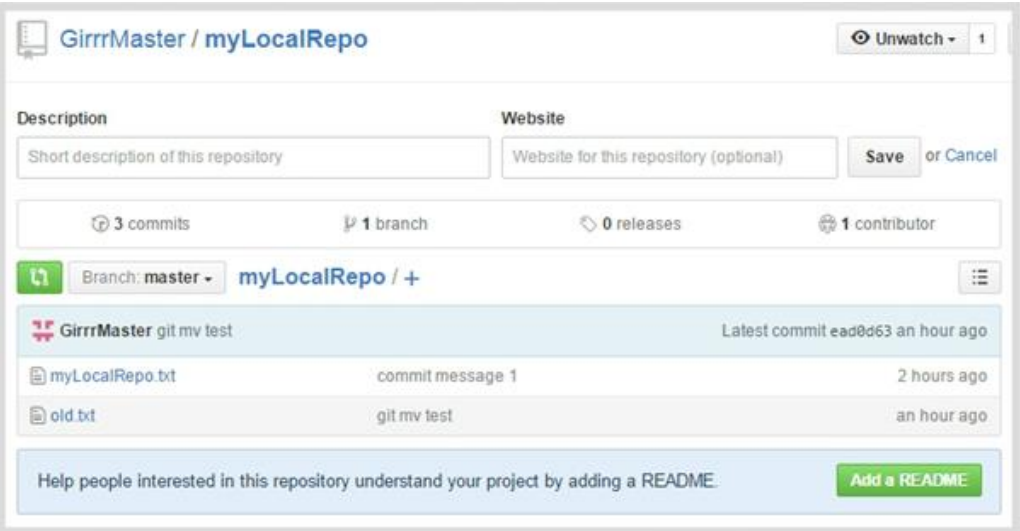
## 2. 소스코드 반영을 위한 Git 명령어

### 11) Git push

- Git push : 로컬저장소의 데이터 → 원격저장소
- 아무 옵션 없이 Git push를 수행하게 된다면?
  - 원격저장소의 오류 메시지가 발생
- Git pull : 비어있는 원격저장소에 기존 원격저장소에 데이터 존재 → 원격저장소의 데이터를 가져와서 동기화 한 후 push 수행
- Git push -u https://Github.com/Girrr Master/myLocalRepo.Git master 실행

```
user@userpc MINGW64 /c/gitFolder/myLocalRepo (master)
$ git push -u myRemoteRepo master
Username for 'https://github.com': GirrrMaster
Password for 'https://GirrrMaster@github.com':
Counting objects: 7, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (7/7), 702 bytes | 0 bytes/s, done.
Total 7 (delta 0), reused 0 (delta 0)
To https://github.com/GirrrMaster/myLocalRepo.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from myRemoteRepo.
```

- GitHub 원격저장소 확인




# 핵심정리

## Git push 준비하기

### 1. Git push를 위한 환경설정

- 사용자 정보 등록

  - 사용자 이름 및 이메일 설정 → 불필요한 설정 값 삭제

- 윈도우 사용자를 위한 추가 설정

  - Git 컬러 사용하기, 문서편집기 설정하기, diff 툴 설정하기

### 2. 소스코드 반영을 위한 Git 명령어

- Git init, Git add, Git status, Git commit -m, Git commit -a -m, Git rm, Git mv, Git log, Git remote, Git push


# 핵심정리

## Tortoise Git에서 Git push

### 1. 기존 파일 내용 수정하기

- 현재 작업중인 Repository의 내용 수정
- 상위폴더로 이동하여 commit창 실행
- commit 메시지 입력하여 commit
- Push 수행
- Tortoise 인증하기
- 원격저장소에서 확인하기

### 2. 새로운 파일 추가

- 새로운 파일에 내용 추가
- commit창 실행하여 메시지 입력
- 추가 파일 체크 후 반영
- Push 수행
- 원격저장소에서 확인하기