소스트리

start a working area (see also: git help tutorial)

clone Clone a repository into a new directory

init Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)

add Add file contents to the index

mv Move or rename a file, a directory, or a symlink

restore Restore working tree files

rm Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)

bisect Use binary search to find the commit that introduced a bug

diff Show changes between commits, commit and working tree, etc

grep Print lines matching a pattern

log Show commit logs

show Show various types of objects

status Show the working tree status

grow, mark and tweak your common history

branch List, create, or delete branches

commit Record changes to the repository

merge Join two or more development histories together

rebase Reapply commits on top of another base tip

reset Reset current HEAD to the specified state

switch Switch branches

tag Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)

fetch Download objects and refs from another repository

pull Fetch from and integrate with another repository or a local branch

push Update remote refs along with associated objects

사용자 설정

git config –global user.name “이름

git config --global user.email [kjk9858@gmail.com](mailto:kjk9858@gmail.com)

확인하는 방법

git config user.name

git config user.email

소스 트리 설치

깃허브 가입

버전 관리 시작하기

현재 작업하는 공간 디렉터리

스테이지

저장할 공간 저장소

로컬 저장소를 작업 디렉터리라 한다

버전을 변경했을 때 1000개의 파일중 일부만 새로운 버전으로 올릴수 있다.,(선별작업가능)

스테이지라는 공간이 이역활을 한다.(전체 컷밋 말고 내가 원하는 부분만 커밋)

저장소는 버전이 만들어지고 관리되는 공간

스테이지에 추가하는 것을 add

스테이지에 있는걸 저장소에 올리는 것을 커밋이라 한다.

(새로운 버전을 만드는 것을 커밋이라한다.)

.커밋할때에는 첫줄은 제목 한줄넘기고 세번째줄부터는 내용이다

수정은 노란색

추가는 초록색

삭제는 빨간색

특정 팔일 스테이지에 올리지 않기

메모장을 만들어서

e.txt

ignore/

=>ignore/ 파일안에 들어간거는 올리지 않는다.

각각의 커밋을 구분할수 있어야한다.

태그는 사용자들에게 보여줄 내용을 명시한다.

커밋 되돌리기

리버트 현재 버전 전버전을 새로운 버전으로 한다 현재 버전은 유지하고

커밋만 되돌리는거는 소프트 리셋

작업 디렉터리 내변경 사항은 유지 그외 모두다 되돌리는거 믹시드 리셋

작업 디렉토리까지 돌리는거 하드 리셋

스태시 :임시저장 기능

-한번 커밋하여 추적된 거에만 사용가능하다.

깃 허브 협업 하기

Ssh키 만들기

ssh-keygen

깃 명령어

경로 확인 pwd

명령어로 저장소 만들기

$ git init

Initialized empty Git repository in C:/Users/KOSTA/Desktop/test/.git/

$ git status

Git status => 현재 상태 보는거 자주 사용한다

스테이지 추가

모든 파일 올리기 git add .

특정 팔이만 올리기 git add 파일명.파일 확장자명

커밋 git commit –m “커밋메세지”

커밋 목록 출력 git log

커밋과 스테이치 추가 동시에 git commit –am “커밋메세지”

인섵트 모드 들어갈려면 스테이지 추가후 git commit 만 해주면 들어가진다

입력모드로 들어갈려먼 a,I 입력 다 작성하고 난후 ese

맨위에 쓴게

:w 하면 내용이 저장된다

나가고 싶으면 :q

동시에 하는 명령어 :wq

커밋 조회하기

Git log **--oneline**

패치

Git log –-p

Git log –graph

-모든 브랜치의 커밋목록을 볼수 있다

Git log –branches

태그 추가 하기 git tag 태그명

만약 다른 커밋에 태그를 추가할려면 (git tag 태그명 커밋해쉬)

태그 조회 git tag –list or git tag –l

태그 삭제 git tag –delete 태그명 or –d 태그명만 해도 된다

디렉터리 비교

Git dff 최근 커밋과 디렉터리 비교하기

Git diff –-staged 최근 커밋과 스테이지 비교

커밋과 커밋으 비교 git diff (비교할 커밋 해쉬코드) (커밋 해쉬코드)

앞에 있는 정보를 기준으로 뒤에있는거와 비교

HEAD 로 사용가능하다 현재 버전이 HEAD, 그앞 버전이 HEAD~1 이런식으로 한다

Reset 은 커밋 없애는거

REVERT는 현재꺼 커밋하고 되돌리는거

예전 커밋으로 되돌악기 git reset <되돌아갈 커밋>

Git reset –-soft 커밋

Git reset 되돌아갈 커밋 자동으로 믹시드 리셋이 됨

하드 리셋는 디렉토리 까지 수정한다. –-hard

Git revert 커밋 => 기존의 커밋을 취소할 새로운 커밋을 만든다

임시저장

Git stash

임시저장에 메모추가 git stash –m “ ”

임시저장 리스트

Git stash list

적용하기

Git stash apply stash@{번호}

스테시 삭제

Git stash drop stash@{번호}

모두 삭제

Git stash clear

브랜치 관리

현재 브랜치 git branch

브랜치 나누기 git branch 브랜치이름

브랜치 바꾸기 git chechkout 브랜치이름

Git checkout –b 브랜치 이름 브랜치를 만들자 마자 이동

브랜치 병합하기

Git merge 병합할브랜치

충돌할경우 수정후 스테이지에 올리고 커밋하면 추가된다

브랜치 삭제 git branch –d 브랜치명

브런치 리베이스

Git rebase 브랜치 명 => 재배치 하기 하기

어디로 재배치할 브랜치 명을 적는다

재배치할 브랜치 명 보다 위로 올라간다.

깃허브 다루기

원격저장소 복제하기 git clone: 주소

원격저장소 추가 죄회 삭제하기

KOSTA@DESKTOP-OVLRB24 MINGW64 ~/Desktop/test\_repo

$ git init

Initialized empty Git repository in C:/Users/KOSTA/Desktop/test\_rep

KOSTA@DESKTOP-OVLRB24 MINGW64 ~/Desktop/test\_repo (main)

$ git remote add origin git@github.com:Aristia0930/test\_repo.git

리모트 잘 되었는지 확인

KOSTA@DESKTOP-OVLRB24 MINGW64 ~/Desktop/test\_repo (main)

$ git remote

Origin

오리진 이름 바꾸기

KOSTA@DESKTOP-OVLRB24 MINGW64 ~/Desktop/test\_repo (main)

$ git remote rename origin changed

KOSTA@DESKTOP-OVLRB24 MINGW64 ~/Desktop/test\_repo (main)

$ git remote

changed

삭제하기

KOSTA@DESKTOP-OVLRB24 MINGW64 ~/Desktop/test\_repo (main)

$ git remote remove changed

KOSTA@DESKTOP-OVLRB24 MINGW64 ~/Desktop/test\_repo (main)

$ git remote

원격 저장소에 푸시하기

$ git remote add origin git@github.com:Aristia0930/test\_repo.git

KOSTA@DESKTOP-OVLRB24 MINGW64 ~/Desktop/test\_repo (main)

$ git remote

origin

현재 브랜치이름을 main으로 바꾸어라

KOSTA@DESKTOP-OVLRB24 MINGW64 ~/Desktop/test\_repo (main)

$ git branch -M main

KOSTA@DESKTOP-OVLRB24 MINGW64 ~/Desktop/test\_repo (main)

$ git push -u origin main

이렇게 한번하고 난후

Git push 만해도 올라간다.

깃허브에 있는거

Git fetch

커밋 가져오기 거는 병합 하면 된다. Git merge origin/main

Git pull 은 병합과 가져오는 한번에 하는거다.