Download Git Client  
 - <https://git-scm.com/downloads>

SCM Service Site  
 - [GitHub](https://github.com/) : 무료는 public만 가능, 안정적  
 - [BitBucket](https://bitbucket.org/product) : public/private 모두 무료, 안정적인 편  
 - 기타: [GitLab, Codebase, CloudForge, etc](https://www.git-tower.com/blog/git-hosting-services-compared/)

Git에서 origin이라는 것은 Github 저장소의 소스를 이야기함 ex. origin/master, origin/br1

0. commit 규모

.git directory는 히스토리의 창고 -> 작업들을 버전별로 나눠서 관리할 수 있음

application을 만든다면 기능별로 만들어서 commit

ex.

[

Initialise project

Add LoginService module

Add UserRepository module

Add Welcome module

Add About page

Add light theme

]

너무 작은 단위, 혹은 너무 큰 단위로 하는것은 바람직 하지 않음

보통 메세지를 현재형, 동사형으로 작성

주의사항: 수정할때는 하나에 대해서만 수정해야함

ex. crashing을 잡는데 refactoring도하고 다른 버그도 잡은 것을 동시에 commit 하면 안됨

소스에 주석을 남기기보다 github나 gitlens를 이용해서 comment추가

1. 깃 설정

git config --list : 깃 설정 사항 보기

git config --global -e : 깃 설정 사항 보기(에디터로: vim)

git config --global core.editor "code" : config를 에디터로 수정할 때 vs코드로 실행하고 terminal로 다른 수행 가능

git config --global core.editor "code --wait" : config를 에디터로 수정할 때 vs코드로 실행하고 terminal은 수정 끝날때까지 멈춤

//code . : vscode로 현재 설정 열기

git config --global user.name "ymkwon" : 깃 이름 설정

git config --global user.email "ymkwon@hitecis.co.kr" : 깃 이메일 설정

git config user.name : 깃 이름 확인

git config user.email : 깃 이메일 확인

git config --global core.autocrlf true (macOS: true 대신 input) : git에 저장할 때 \r을 삭제하고 올림(다양한 운영체제 사용시 읽지 못하는 오류 방지)

2. 깃 커맨드

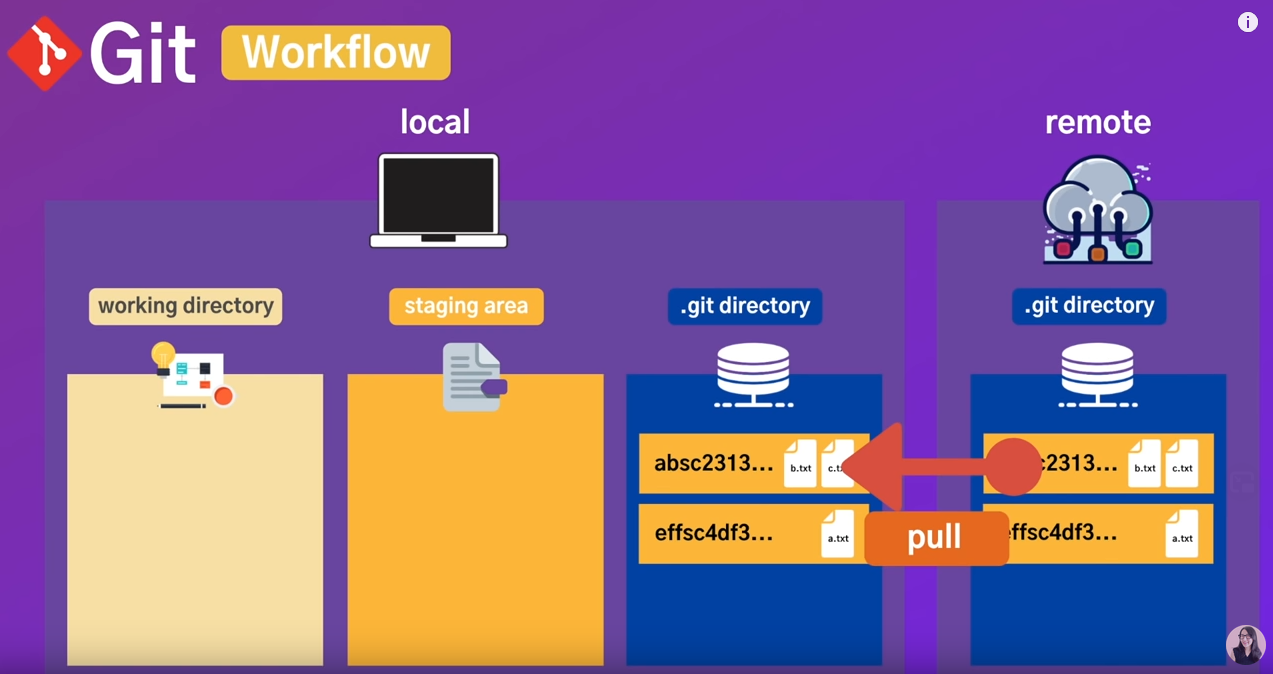
git init : 깃 프로젝트 생성, .git 파일 만들어짐 (rm -rf .git하면 더 이상 깃 프로젝트가 아니게 됨)

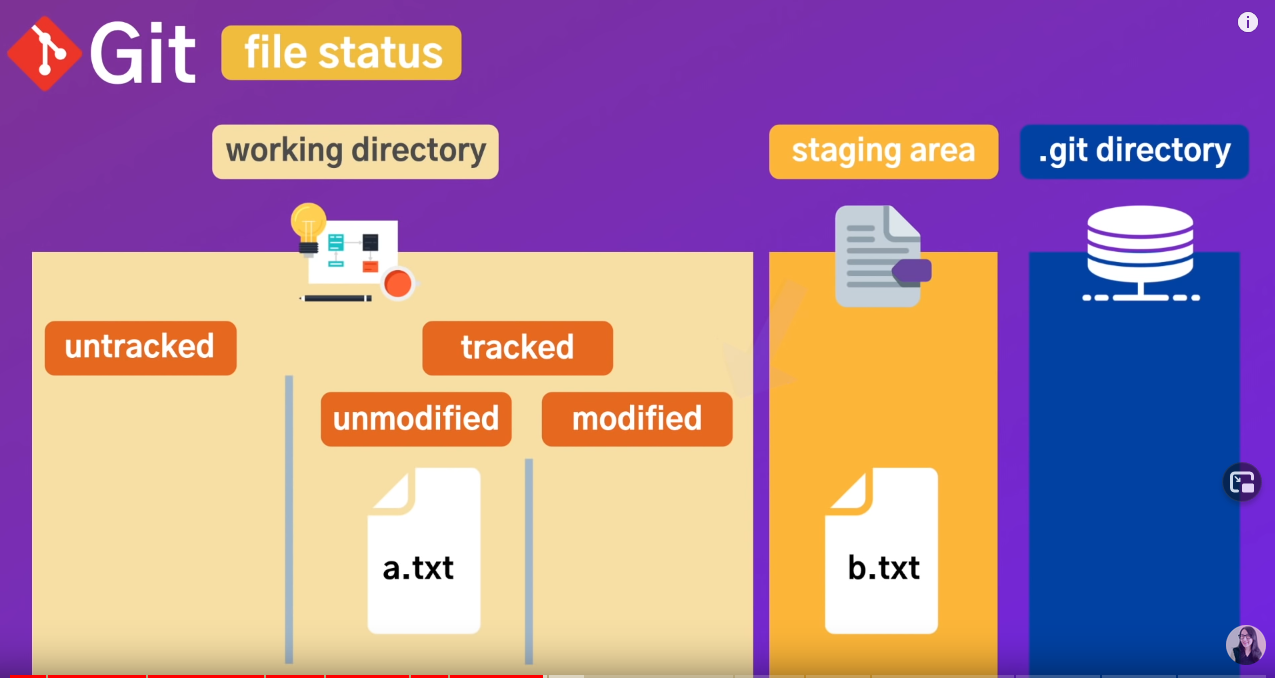
git status: 현재 commit 상태 확인

(만약 간단하게 alias를 설정하고 싶다면 git config --global alias.st status 이런식으로 작성)

git 명령어 -h : 명령어에 대한 도움말 보기

3. 깃 구조





* Modified된 파일만 staging area로 이동할 수 있음

4. 깃 구조 확인

echo hello world > a.txt : a.txt파일에 hello world를 써라

echo hello world >> a.txt : a.txt파일에 hello world를 추가하라

* 파일만 만들어진 경우 git status로 확인하면 untracked된 파일로 되어 있음
* tracked 파일로 만들어 주기 위해 git add 사용

git add a.txt b.txt : 추가된 a.txt, b.txt파일을 untracked file로 만들어줌

git add \*.txt: txt 전체 파일

git status: 확인

만약 여기서 파일을 수정한다면?

echo test > a.txt

git status

이런 경우 a.txt가 modified된 것을 확인할 수 있음

sourcetree로 확인할 경우: 이전 a.txt는 staging area에 올라가있고

최근에 수정한 것은 아직 working directory에 머물러 있는것을 확인할 수 있음

git rm --cached a.txt: staging area에서 다시 working directory로 돌려놓기

track하고 싶지 않은 파일 있다면 .gitignore에 저장하면 됨

(cls: 화면의 대화 다 지우기)

git status -s: 간단하게 보기 (git status --long이 default)

git diff : 내용이 어떻게 수정됬는지 확인하기

(cat c.txt : c.txt 파일 내용 확인)

git diff --staged: staging area에 올라가 있는 파일 어떻게 수정됬는지 확인

(= git diff --cached : staged는 cached랑 동의어로 사용됨)

git commit : .git directory에 올리기

git log: git commit 내역 살펴보기

git commit -m "second commit": 커밋할때 second commit이라는 메세지 입력하면서 커밋

echo add >> c.txt

git commit -a -m "third commit": working directory, stagin area에 있는 파일 모두를 메세지와 함께 커밋

(=git commit -am "third commit")

git log: 커밋 기록 보기

git remote add origin <url> : github repository랑 연결하기

git push u origin <branch name>: .git directory의 commit을 github repository에 전송 (처음에만)

git clone <url>: github repository의 파일 그대로 가져와서 .git directory 만들기

git push: 일반적인 경우 이렇게만 푸시해도 됨(브랜치가 없는 경우)

git pull

git remote rm origin: 서버와 연결된 repo 끊기

git log –graph –oneline : 로그를 그래프 형태로 한줄로 해서 보여줌

4. 충돌 해결법

Revision(log남기기)를 잘해야 원하는 위치로 잘 돌아갈 수 있음 !!!

git reset –[hard | soft | mixed] <hash>: revision만 되돌림, wd소스 그대로 (hard는 흔적 없이 되돌리는 것이기 때문에 사용할 때 매우 주의!!)

* 즉, commit 기록을 다 지우게 됨
* 이런 경우 소스가 변경되어 있지 않기 때문에 소스를 rollback하는 지점과 똑같이 만들어 준 후 push할 수 있음
* git push origin +<branch name> (=git push uf origin <branch name>)

git revert <hash>: revision 남아있음, wd소스 conflict (log로 조회 가능)

* 동작 시키는 경우 에러 뜰 것임
* 소스 확인하고 변경사항 있으면 맞춰준 후 다시 commit 후 푸시 하면 해결

5. branch 생성 및 관리

# 브랜치 기본

git branch: 브랜치 목록 보기

git branch -r: 서버에 있는 브랜치 목록

git branch <name>: name으로 브랜치 만들기

git checkout name: name 브랜치로 이동

# master로 merge하기 (작업폴더에서!!)  
git checkout master    #master로 변경  
git merge <branch-name>  
git branch  
git log

# conflict  (non fast-forward)  
git status

# branch 삭제  
local> git branch -d <branch-name>  
local> git branch  
remote> git push --delete origin <branch-name>

# branch에서 작업하고 merge

git checkout -p <branch-name> <file-name>: branch의 파일을 지금 작업하는곳에 patch 시킴

\*\*

git checkout -t origin/<branch name>: 서버의 특정 브랜치 가져오기

* 서버에서 가져올 때는 기본값은 master에서 밖에 내려올 수 없기 때문에 특정 branch에서 불러올 때는 위의 방법을 써야함
* 브랜치 연결한 이후로는 다음 명령어를 통해 정보 받을 수 있음

# 특정 브랜치 pull 하기  
git pull origin <branch-name>햣

cf. git push origin <branch-name>