Using git

why git?

- 분산버전관리 (Distributed Version Control System) → 이것이 주는 장점? → 기대효과
- 사용자의 요구사항
  - 추가할 기능 많음; 각 기능별로 작업 이력을 git commit에 저장하므로 작업 이력을 관리 할 수 있음
  - 항상 모든 이슈사항은 응급!; 버그나 수정사항들이 발생하여 프로그램을 수정해야 할 경우, 요청자는 항상 빠르게 대처 가능.
- 개발자의 요구사항
  - 개발과 실제 운용 소스가 다름; 현재 작업 중인 내용이 있다면 현재 작업 내용을 백업하고, 운영(서비스)되는 배포판의 소스를 가져와서 수정한 후 배포하고 수정된 내용을 작업 중이던 소스에도 적용해야 하는 경우에 git은 유연하게 대처 가능.
  - 개발 사항에 대한 이력이 필요; 각 작업별로 언제, 어떤 내용을, 누가 했는지 이력을 관리하면 프로젝트 관리에 이점.

pros. and cons.

pros.

- 강력한 브랜치 기능; 브랜치를 쉽게 만들고 병합 가능, 일시적인 작업에 대한 이력 관리 쉬움
- 속도와 성능; 처리 속도가 빠르고 기능이 많음
- 분산작업; 장소에 구애 받지 않고 협업이 가능(메인 서버에 연결없이 작업 가능), 저장소의 완전한 복사본으로 작업
- 데이터 보장성; 모든 파일을 암호화하여 저장
- 무료; 비용 안 듬.

cons.

- 배우기 어려움.
- 어떻게 사용하는 가가 중요하므로 잘 정의된 개발 프로세스가 필요.
- git hub와 같은 서비스를 이용할 경우 비용 발생. (bitbucket은 5명까지 무료 사용 가능)

setting up a git

- 설치 osx, windows, linux
- global settings

git 시작하기

- 생성 init, clone

- git이 파일(디렉토리)를 관리하는 방법
    - working directory → stage area → git repository
    - working directory; 사용자가 작업하는 디렉토리
    - stage area; git repository에 저장될 파일 목록
    - git repository; git이 파일을 저장하는 저장소
- git repository의 파일 관리 add, remove, commit
- 상태 보기 status
- 뒤로 되돌아 가기 reset
- 버전 되돌아보기 log
- 태그 붙이기 tag
- 이건 저장하지 않을 거야 .gitignore
    - repository에 넣으면 좋은 것들; 스펙문서, 설계서, 매뉴얼, 도움말, 소스코드, 이미지, 외부라이브러리, 빌드스크립트
    - 넣지 말아야 할 것들; 바이너리, 설치본, 영업자료, 마케팅자료, 판매자료 등.
- stash

git branch

- 기준이 되는 메인 프로젝트와 개발 프로젝트 혹은 디버그 프로젝트, 핫픽스 프로젝트 master, branch
    - 가지 치기, 가지 잘라내기
- 지금 작업하고 있는 위치 head
    - 개발 브랜치에서 핫픽스 브랜치로 전환 checkout
- 메인+브랜치⇒새로운 메인, 브랜치1+브랜치2⇒새 버전의 브랜치1 merge
    - 병합(merge)하기 전에 diff
    - rebase

원격 repository

- 서버 repository에서 혹은 서버 repository으로 pull, push

git flow stories

- 도구의 사용법 보다 사용 방법이 더 중요하다.
- 메인 프로젝트는 놔두고 개발 프로젝트로 작업하기
    - master에서 develop 브랜치 만들어 작업하고 develop에서 release 브랜치 생성, release
- 한참 개발 중인데 급한 버그 수정 요청이 들어왔다
    - master에서 hotfix 브랜치를 만들어 작업하고 master와 현재 개발중인 브랜치로 merge
- 하나의 프로젝트 여러 명의 개발자
    - develop 브랜치에서 각 개발자별로 브랜치 생성후 작업 (보통 기능별로 개발자에게 분배) 혹은 develop에서 기능별 브랜치 생성

submodule

fork, pull request

- fork → clone forked repository → make branch and move to branch on forked repository → coding jobs on branch → push → pull request

references of git

- git 간편 안내서
- Pro git 온라인 북 - 한글판
- git-flow
- git 사용자 설명서

Git 사용법 정리

```
*basic concept
workspace -> staging area(index) -> local repository -> remote repository
stash

*file status lifecycle
1. untracked; git에 의해 추적되지 않는 파일. add the file 후 tracked로 변경됨
2. tracked; git에 의해 추적되는 파일. remove the file 하면 untracked로 변경됨
  2.1 unmodified; 수정되지 않은 파일. edit the file 후 modified로 변경됨
  2.2 modified; 수정된 파일. stage the file
  2.3 staged; staging area에 있는 파일. commit 후에는 unmodified로 변경됨


*기본 동작
만들기
(저장소 받아오기)
파일 저장: 생성, 삭제, 수정
로컬 저장소 저장: 생성, 삭제, 수정
원격 저장소 저장: 생성, 삭제, 수정
브랜치: 생성, 삭제, 수정, 이동, 병합

*install git
-Unix/Linux
-Mac
-Windows


*config git
```

```
.gitignore
Glob pattern


*starting w/ git
git init; create local repository
git remote add <repository>; connect local repository with remote
repository
git pull; get data from remote repository


git clone <repository>
ex) git clone /local/repository/path
ex) git clone user@hostname:/remote/repository/path


*basic commands
git add; workspace -> staging area(index)
git commit -a; workspace -> local repository
git push; local repository -> remote repository


git fetch; remote repository -> local repository
git merge; merge

git pull; fetch & merge

git status; modified, staged or unstaged
git diff; show differences between staged and unstaged
git diff --staged or git diff --cached; show differences between staged and
committed

*snapshot

*branch
-create branch
git branch <name>
git checkout <name>

git branch -b <name>; branch & checkout

-delete branch
git branch -d <name>


-merge branch
git checkout master
git merge branch


-list branch
```

```
git branch
git branch --no-merged
git branch -merged

*rebase

*tag
-create tag
git tag <name>
git tag -a <name>


-delete tag
git tag -d <name>


-list tag
git tag


*rollback
git checkout --; rollback unstaged file

git reset HEAD; unstage staged file

git fetch origin; rollback commited file
git rest --hard oring/master

*stashing


 * gitignore 재적용
git rm -r --cached .
git add .
git commit -m "git ignore applied and fixed untracked files"
git push

git pull
git add .
git commit -m "git ignore applied"
git push
references
```

pro git 한글판 http://git-scm.com/book/ko/ git 간편안내서 http://rogerdudler.github.io/git-guide/index.ko.html

From:
http://theta5912.net/ - **reth**

Permanent link:
http://theta5912.net/doku.php?id=public:computer:git&rev=1615795836

Last update: **2021/03/15 17:10**