
GIT 설치하기

Ubuntu 리눅스에서 git 패키지로 설치.

```
$ sudo apt-get install git-core gitk tig
```

실명과 이메일 주소를 config에 설정. 커밋 시 남는다.

```
$ git config --global user.name "Gildong Hong"
$ git config --globla user.email "big.theaf@insignal.co.kr"
```

색 콘솔을 위해

```
$ git config --global color.ui auto
```

`git config -l` 로 설정 된 config 셋을 확인 할 수 있다.

SSH 키 쌍 생성

SSH Key 생성하기. 개인키, `~/.ssh/id_rsa` 파일이 없는경우 생성

```
$ ssh-keygen -t rsa -C "your_email@company.com"
```

git 관리자에게 전달할 공개키 생성

```
$ ssh-keygen -y > your_id@insignal.pub
Enter file in which the key is (/home/suapapa/.ssh/id_rsa):
Enter passphrase:
```

생성된 공개키 your_id@insignal.pub를 git 저장소 관리자에게 전달.

자주 사용하는 git 명령어들

각 명령어에 대한 도움말은 help 명령어로 확인할 수 있다. 예로 clone 명령어의 도움말을 보려면 다음과 같이 실행한다.

```
$ git help clone
```

init

현재 디렉터리를 git 저장소를 시작. 원격 저장소 없이도 git 저장소를 꾸릴 수 있다.

```
$ git init
```

init 은 저장소 관리를 위해 .git 디렉터리를 만들 뿐이다. 파일들을 추가하고 커밋해 준다.

```
$ git add *
$ git commit -m "Initial commit"
```

clone

원격 저장소로 부터 저장소를 시작할 수 있다.

```
$ git clone <repo_address> [directory_name]
```

diroctory_name 디렉터리에 git 저장소가 clone되며 따로 init 할 필요 없다.

log

현재 브랜치 -git 에서는 브랜치를 일상적으로 사용한다-의 로그를 확인

```
$ git log
```

특정 파일 또는 디렉터리만의 로그 확인

```
$ git log <specific_file.c>
```

다음과 같은 문법으로 브랜치 간의 차이들을 확인해 볼 수 있다.

```
$ git log --oneline <another_branch_01>..<another_branch_02>
```

status

현재 브랜치의 상태 (변경된 파일 및 Staging 상태) 보기

```
$ git status
```

diff

수정된 내역 (Unstaging) 을 확인

```
$ git diff
```

Staging 되었을 때 (커밋 전) 최종 커밋(HEAD) 와의 비교

```
$ git diff HEAD
```

특정 커밋, b48f1b647f0787d3cdeb3 에서부터의 차이 보기

```
$ git diff b48f1b647f0787d3cdeb3..
```

브랜치간 비교

```
$ git diff <branch01>..<branch02>
```

add

Git에서는 변경 (또는 추가) 된 파일들을 커밋 전에 Staging 상태로 등록해야 한다.

```
$ git add <file or directory>
```

-u 옵션을 사용하면 관리 중인 모든 파일(들 중 변경된 것들만)을 Staging 할 수 있다.

```
$ git add -u
```

commit

커밋 전 diff, status를 사용해 변경 내용이 맞는지 확인. git diff는 자주하는 실수 를 명확히 알려준다. add으로 staging 하지 않은 파일들은 커밋되지 않는다.

커밋로그와 함께 커밋.

```
$ git commit -m "log message"
```

> -m 옵션 없이 실행하면 GIT_EDITOR 변수에 설정한 에디터가 실행되고, 커밋 메세지를 입력한 후 저장하면 커밋 된다.

커밋 로그는 첫 줄엔 간단한 설명 다음 줄 한 칸 띄고 긴 설명을 적는다. 각 줄은 80컬럼을 넘지 않도록 작성한다. 예;

```
libmedia: Fixed bug on mix-encoded tag of Korean mp3s.

There are some Korean mp3s which not only one encoding used in their tag,
like album title in UTF-8 and everything else (like artist, title and etc)
in EUC-KR. and, only UTF-8 values was shown correctly.
```

> 현재 프로젝트의 커밋로그 형식이 어떤 관습을 따르는지, git log ./ 로 확인하고 같은 형식으로 커밋하는 것이 좋다.

방금 한 커밋을 다음 명령어로 수정할 수 있다. 자주 사용해야 할 명령

```
$ git commit --amend
```

---

branch 사용하기

checkout

다음 명령어로 로컬 브랜치 간 이동

```
$ git checkout <another_branch>
```

-b 옵션을 주면 새 로컬 브랜치를 생성한 뒤 이동. 자주 쓰는 옵션이므로 기억하자

```
$ git checkout -b <new_branch>
```

--track 옵션을 추가해 remote 의 branch를 track 하는 로컬 브랜치를 만든다.

```
$ git checkout --track -b <new_local_branch> origin/<remote_branch>
```

branch

로컬 브랜치 목록 보기

```
$ git branch
```

원격 브랜치 목록 보기

```
$ git branch -r
```

특정 커밋을 포함하는 브랜치 목록 보기

```
$ git branch --contains b48f1b647f0787d3cdeb3
```

로컬 브랜치 지우기

```
$ git branch -d <branch_name>
```

---

remote 와 함께 작업하기

remote 정보 확인

다음 명령어로 리모트의 목록을 볼 수 있다. clone을 막 마친 상태에서는 기본 alias인 origin 만 표시될 것이다.

```
$ git remote -v
```

다음 명령어로 특정 remote (여기서는 origin)의 주소및 연결된 로컬 브랜치를 확인할 수 있다.

다
```
$ git remote show origin
```

pull

remote의 변경 내용을 적용하려면 pull 명령어를 사용한다.

```
$ git pull
```

위 명령어는 로컬 브랜치 master(clone시에 생성)에 연결된 remote 브랜치의 변경 내용을 가져오는 것으로 특정 브랜치를 땡겨 오려면 다음과 같이 실행하면 된다.

```
$ git pull origin <remote_branch>
```

push

로컬의 작업 내용을 push 하기 전 까지는 서버(remote)에 반영되지 않는다. 연결된 remote 브랜치가 있다면 다음과 같이 push 한다.

```
$ git push
```

특정 remote, origin의 특정 브랜치 <remote_branch>에 push 하려면;

```
$ git push origin remote_branch
```

---

git 팁들

- gitk(GUI) 또는 tig(TUI)를 이용하면 커밋들을 보기 쉽게 열람 가능. 모두 기본 저장소에 있는 패키지.
- 모든 명령어, 명령어의 인자들은 자동완성 된다. 아리송할 때는 TAB키를 눌러보자

Reference

- http://emacs.kldp.net/wiki/doku.php?id=tool_chain:git from emacs.kldp.net
- http://help.github.com/git-cheat-sheets/ from Github