github

# github

소프트웨어 개발 프로젝트를 위한 소스코드 관리 서비스입니다. 소스 코드를 열람하고 간단한 버그 관리, SNS 기능, 협업까지 갖추고있어 개발자에게 없어서는 안될 서비스입니다.

# 기초 지식

**사전 지식 1: commit과 push**

* 커밋 (commit) : 파일을 추가하거나 변경 내용을 저장소에 저장하는 작업
* 푸시 (push) : 파일을 추가하거나 변경 내용을 원격 저장소에 업로드하는 작업

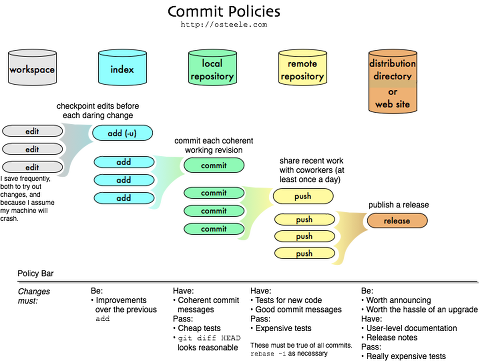
**사전 지식 2: 로컬 저장소와 원격 저장소**

* 로컬 저장소: 자신의 컴퓨터에 있는 파일이나 디렉토리를 저장하는 장소.
* 원격 저장소: 서버 등 네트워크에 있는 파일이나 디렉토리를 저장하는 장소.

**사전 지식 3: branch**

* 현재 출시하고 있는 버전의 유지 보수를 하면서 새로운 기능 추가 및 버그를 수정할 수 있게 병렬로 수행되는 여러 버전 관리 방법

# github 용어 정리



**<GitHub Commit Policy>**

* 저장소 (Repository): 작업자가 변경한 모든 내용을 추적하는 공간.
* 작업 트리 (Working Tree): 저장소를 어느 한 시점을 바라보는 작업자의 현재 시점
* 체크 아웃 (Checkout): 작업자의 작업트리를 저장소의 특정 시점과 일치 하도록 변경하는 작업.
* 스테이징 영역: 저장소에 커밋하기 전에 커밋을 준비하는 위치.
* 브렌치 (branch): 하나의 개발 라인을 의미한다.
* Master: master 브랜치는 복사해온 저장소 내의 HEAD의 복사본.
* Origin: 단지 git가 복사해 온 저장소를 가리키기 위해 기본적으로 사용하는 이름.

# github 사용법

* Github초기 세팅
* Github에 저장소 작성 (git init) 또는 복제 (git clone)
* 파일의 작성 및 편집
* 파일의 생성 / 변경 / 삭제를 git 인덱스에 추가 (git add)
* 변경 결과를 로컬 저장소에 커밋 (git commit)
* 로컬 저장소를 푸쉬해 원격 저장소에 반영 (git push)

# Github 초기 세팅

* Git bash 설치하기
* Git bash를 열고 터미널에 초기 세팅 명령어를 작성한다.
* ‘git config –global user.name 작업자 이름’: 작업자 이름 설정.
* ‘git config –global user.email 작업자 이메일’: 작업자 이메일 설정.
* ‘git config –global –list’: 설정값 확인.

# Github에 저장소 작성 또는 복제

* Repository 생성하기
* Description: 저장소에 대해 설명합니다.
* Public은 모두 사용할 수 있고 / Private은 유료 회원만 작성할 수 있습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**<respository생성 화면>**

* 파일을 생성 및 편집하고 해당 파일이 있는 폴더로 이동합니다.
* Public은 모두 사용할 수 있고 / Private은 유료 회원만 작성할 수 있습니다.
* 해당 폴더에 터미널에 ‘git init’ 명령어 입력하기 (git init: git 저장소를 새로 만드는 명령어로 이 명령을 실행하면 현재 디렉토리를 Git 저장소로 변환합니다.)
* ‘git clone repository주소’: repository에 있는 정보들을 로컬 위치에 저장한다.

# 파일의 생성 / 변경 / 삭제를 git 인덱스에 추가

* ‘git add .’: 현재 모든 파일을 인덱스에 추가하는 명령어입니다. 인덱스는 저장소에 커밋 할 준비를 하기 위해 변경 내용을 임시로 저장하는 위치입니다.

# 변경 결과를 로컬 저장소에 커밋

* ‘git commit -m “커밋에 대한 내용”: 인덱스에 추가 된 파일을 커밋하는 명령어입니다. 커밋ㄷ은 파일이나 디렉토리의 추가 또는 변경을 저장소에 기록하는 작업입니다.
* ‘git status’: 저장소에 파일 추가가 기록되었는지 상태를 확인하는 명령어입니다.
* ‘git remote add origin https://github.com/~~ : 원격 저장소에 반영하기 전에 원격 저장소의 정보를 추가합니다. 기존에 respository를 만들었고 내부 저장소에서 push를 해야하지만 로컬 저장소와 원격 저장소를 이어주는 정보가 없기에 이 정보를 추가하는 명령어입니다.

# 로컬 저장소를 푸쉬해 원격 저장소에 반영

* ‘git push origin master’: 로컬 저장소의 변경 사항을 GitHub에 있는 원격 저장소에 반영하기 위한 명령어입니다.
* ‘git pull origin master’: 다른 사람이 작업한 것을 받아오는 명령어입니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

**<원격 저장소에 문서를 업로드하는 명령어 모음>**

**텍스트, 스크린샷, 모니터, 실내이(가) 표시된 사진

자동 생성된 설명**

**<원격 저장소에 문서를 업로드한 결과>**

# 브랜치 사용하기

* 브랜치의 생성, 이동
* ‘git branch’: 현재 브랜치 목록을 알 수 있는 명령어. (브랜치가 없다면 **\*master**의 결과가 나온다.)
* ‘git branch subdir01’: subdir01이라는 브랜치를 생성하는 명령어.
* ‘git checkout subdir01’ subdir01 브랜치로 이동하는 명령어. (브랜치가 **\*subdir01**로 바뀐다.)
* 브랜치에서의 개발 작업
* 지점을 바꾸게 된다면 어떤 변화를 주어도 다른 지점에는 변화가 없다.
* ‘test2.html’ 파일을 만들어서 브랜치에 푸시를 해본다.
* 브랜치에 푸시
* ‘git add test2.html’
* ‘git commit -m “second commit subdir01”
* ‘git push origin subdir01’

텍스트, 전자기기, 측정기이(가) 표시된 사진

자동 생성된 설명

**<subdir01에 문서를 업로드하는 명령어 모음>**

텍스트, 모니터, 스크린샷, 화면이(가) 표시된 사진

자동 생성된 설명

**<subdir01에 문서를 업로드한 결과>**

* 브랜치 병합
* ‘git checkout master’: master로 지점 변경
* ‘git merge subdir01’: .subdir01 브랜치 결과를 병합
* ‘git push origin master’: 병합된 결과를 푸쉬

텍스트, 스크린샷, 모니터, 검은색이(가) 표시된 사진

자동 생성된 설명

* 브랜치 삭제
* ‘git branch -d subdir01’: subdir01의 브랜치를 삭제.