Git을 분산 버전 관리 시스템이라고 표현

분산 버전 관리 시스템 : 여러면의 개발자가 특정프로젝트를 자신의 컴으로 협업하여 개발 하면서 버전관리

Git은 중앙 서버 컴퓨터- 컴퓨터 연결 같은 버전의 데이터 베이스 =컴 A가 중앙 서버에 올리면 업데이트가 되고

컴 B는 중앙 서버 컴으로 최신화 두대 모두 버전과 파일을 유지 할수있다.

GIT 은 CLI (Command line Interface, 명령 줄인터 페이스) 이며 WINDOW에서는 CMD 명령어로 통해 제공

GIT을 통해 사용가능 한 것들

- 소스코드 주고받기가 필요 없고, 같은 파일을 여려 명이 동시에 작업하는 등 병렬 개발이 가능해지며, 버전 관리가 용이해져 생산성이 증가합니다.

- 소스코드의 수정 내용이 커밋 단위로 관리되고, 패치 형식으로 배포할 수 있기 때문에 프로그램의 변동 과정을 체계적으로 관리할 수 있고, 언제든지 지난 시점의 소스코드로 점프(Checkout)할 수 있습니다.

- 새로운 기능을 추가하는 Experimental version을 개발하는 경우, 브랜치를 통해 충분히 실험을 한 뒤 본 프로그램에 합치는 방식(Merge)으로 개발을 진행할 수 있습니다.

- '분산' 버전관리이기 때문에, 인터넷이 연결되지 않은 곳에서도 개발을 진행할 수 있으며, 중앙 저장소가 폭파되어도 다시 원상복구할 수 있습니다.

- 비단 팀 프로젝트가 아닌, 개인 프로젝트일지라도 GIT을 통해 버전 관리를 하면 체계적인 개발이 가능해지고, 프로그램이나 패치를 배포하는 과정도 간단해집니다. (Pull을 통한 업데이트, Patch 파일 배포)



GIT을 이용하면서 일어나는 트랜직션

“작업한 내용을 스테이지에 올려서 로컬저장소에 커밋하고이를 푸시 해서원격 저장소롤 보낸다.

용어 정리

Repository:저장소

소스코드가 저장 되어 있는 여러 개의 브랜치들이 모여 있는 디스크상의 물리적 공간

Git 에는 로컬저장소와 원격저장소(remote repository)로 나눈다.

작업을 시작시 원격 저장소에서 로컬로 소스코드를 복사해서 가져오고 (clone) 이후 소스코드를 변경한 다음 커밋 commit을 합니다. 이 때 커밋한 소스는 로컬 저장 되고 푸시를 하기 전에는원격 저장소에 반영이 되지 않는다.

//이전에는 오픈소스 코드를 배포할 때 버전별 압축파일 형태로 배포했으나, GIT이 등장한 이후 위와 같이 저장소를 통해 배포하는 경우가 많아졌습니다.

위와 같이 임의의 사용자가 마음대로 다운받을 수 있도록 공개된 저장소가 있는 반면, 인증된 사용자만 접근할 수 있는 비공개 저장소도 있습니다. 단, 공개된 저장소일지라도 저장소의 다운로드만 가능하며, 수정된 코드를 저장소에 반영하기 위해서는 저장소 관리자의 허가를 받고 SSH 키를 등록해야 합니다.

Checkout:체크아웃

특정 시점이나 브랜치의 소스코드로 이동하는 것을 의미한다. 체크아웃 대상은 브랜치 커밋 태그 이다. 체크아웃을 통해 과거 여러 시점의 소스 코드로 이동 가능

//**cf. SVN의 체크아웃**

SVN에서는 체크아웃이 원격 저장소의 파일을 작업하기 위해 로컬로 가져오면서 동시에 다른 사람이 수정할 수 없도록 Lock을 거는 과정을 의미하며, GIT에서의 체크아웃과는 전혀 다른 의미입니다.

Commit :커밋

작업한 내용을 로컬로 저장소에 저장하는 과정입니다. 각의 커밋은 의미 있는 변경 단위 이다.

변경에 대한 설명은 커밋 로그로 남는다.

대개 하나의 커밋은 하나의 주제로 묶을 수 있는 변경 단위가 된다. (ex: 검색버그 수정,회원가입 기능 추가 )

프로젝트 팀에 따라 커밋을 하는 단위도 다르고 커밋 로그를 작성하는 형식이 있다.

Tag : 태그

커밋의 임의 위치를 쉽게 찾아 갈수있게 하는 이정표

태그가 붙혀진 커밋은 태그명으로 입력 하여쉽게 체크 아웃 가능

Push : 로컬 저장소의 내용중 원격 저장소에 반영 되지 않는 커밋을 원격 저장소로 보내는 것

//**cf. SVN의 커밋과 푸시**

SVN에서의 커밋은 변경 사항을 원격 저장소로 저장하는 과정을 의미합니다. GIT에서의 커밋은 로컬 저장소로 변경 사항을 반영하는 것을 의미하며, 원격 저장소로 변경사항을 보내는 과정은 푸시입니다.

즉, 'SVN의 커밋 = GIT의 커밋 + GIT의 푸시' 라고 할 수 있습니다.

Pull :풀

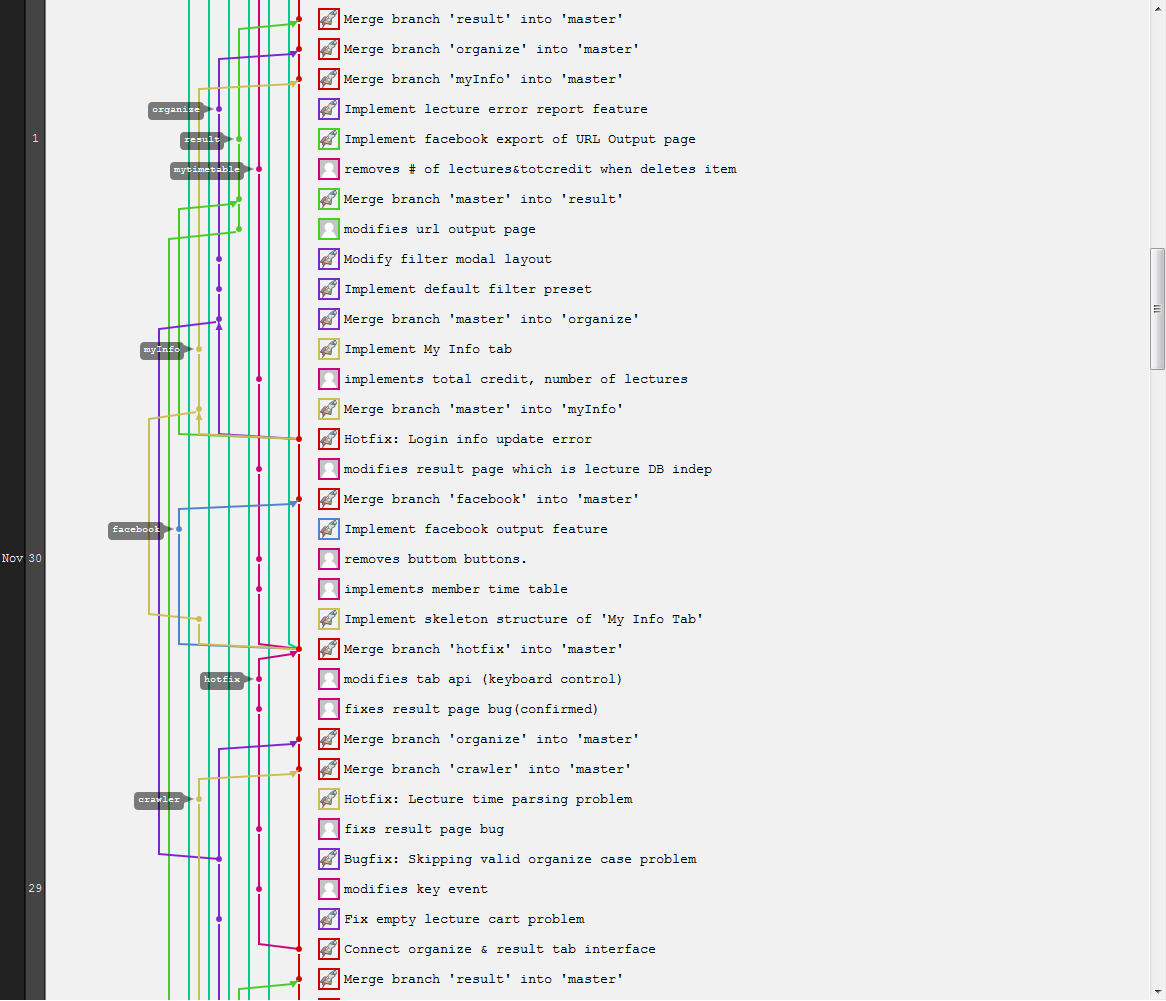
푸시의 반대로 원격 저장소에 있는 내용중 로컬 저장소에 반영 되지 않는 내용을 가져와 로컬 저장소에 저장 하는 것 이를 통해 다른 팀원들이 변경 한 것을 로컬 저장소에 가져올수있다.

//푸시과정에서 충동이 일어 나서 푸시가 거절된 경우 풀을 통해 원격 저장소의 변경 내용을 반영한후 다시 푸시 해야 합니다.

Branch : 커밋을 단위로 구분된 소스코드 타임 라인에서 분기해서 새로운 커밋을 쌓을 수 있는 가지를 만드는 것 그 가지르 브렌치라합니다.

개발의 주축 : 마스터 브랜치

모든 브랜치는 마스터 브랜치에 분기 되어 최종으로 다시 마스터 브랜치에 병합 되어 개발



서로 다른 색상으로 구분된 선들이 하나의 브랜치라 의미함

가장오른쪽 빨간색 선 마스터 브랜치 마스터 브랜치에서 분기된 후 최종적으로 다시 마스터 브랜치에 병합된다. 브랜치를 항상마스터 브랜치에서 해야할 제약은없다. 브랜치에서 브랜치로 분기가 가능 하다 . 대개 브랜치는 프로그램 전체를 몇 개의 단위로 나눈 토픽 단위로 생성 각 토픽 브랜치에서 개발자 별로 새로운브랜치를 생성 한다. 그 이후 최종적으로 마스터 브랜치로 병합되어 하나의 토픽이 종료

//**cf. Fork**

Branch와 유사한 용어로 Fork가 있습니다. GitHub와 같은 Social GIT 사이트에 보면 Fork 버튼이 있는 것을 많이 보셨을 것입니다.

리눅스 프로그램을 해 보신 분이라면 Fork라는 용어를 어딘가에서 들어보셨을 것입니다. (돈까스 먹을때 쓰는 포크 말구요..;;) 리눅스에서 새로운 프로세스를 실행하는 방법 중 하나인 Fork는 이미 동작중인 프로세스를 복제해서 새로운 프로세스를 만드는 것을 의미합니다.

GIT에서의 Fork도 이와 유사한 개념이라고 할 수 있습니다. GIT의 Fork는 저장소를 복제해서 새로운 독립된 저장소를 만드는 작업을 의미합니다.

오픈 소스 프로젝트일지라도 소스코드의 메인스트림에 커밋을 반영하는 커밋터(Commiter)의 권한을 아무에게나 주지 않고, 대개 일정 기준을 통과해야 하는 경우가 많습니다. 대규모 오픈소스 프로젝트일수록 이 기준은 더 까다롭습니다.

공개된 오픈 소스 프로젝트를 자신이 프로젝트 매니저가 되어 입맛대로 수정하고 싶을 때 사용하는 기능이 바로 Fork입니다. 원격 저장소를 로컬 저장소로 복제한 뒤 새로운 원격 저장소에 푸시하는 과정을 한큐에 해결할 수 있도록 도입한 기능이 바로 Fork입니다.

Merge : 병합

브랜치와 반대 되는 개념으로 하나의 브랜치를 다른 브랜치와 합치는 과정

두개의 브랜치를 합쳐서 하나의 브랜치로 만드는 3-way merge가 모든 병합 작업이 기본

병합의 대상이되는 두 브렌치는 주종 관계가 성립 A 브랜치와 B 브랜치를 병합한다.는 애매한 말

A와 B를 병합 한다 와 B와 A를 병합한다 는 말은 다른말이다.

병합과정에서 두개의 브랜치에서 파일의 같은 부분을 서로 다르게 수정하면 충돌이 발생

병합이 일시정시 이경우 충돌이 발생한 부분을 직정 수정 Merge tool등을 이용해 해결 한후

병합을한다. 보통 PM이 주로 관리 한다,.

GIT 관련 웹 기반 솔루션

GITHUB :웹기반 오픈소스 프로젝트에서사용

GITLAB 기업체 인트라넷 설치하여 사용