1. VCS

[VCS, 버전 관리 시스템]

협업을 하며 프로젝트를 진행할 경우 VCS를 이용하여 버전관리를 하면 작업이 용이해진다.
VCS란 버전관리시스템으로 파일의 변화를 기록하고, 어떠한 시점으로 버전을 가져오는 게 가능하다. 파일을 이전 지점으로 복구하고, 다른 사람의 작업 이력을 확인하는 것이 가능하다는 등의 장점이 있다.

2. CVCS와 DVCS

[Server-Client Model, CVCS, 중앙집중식 버전 관리 시스템]

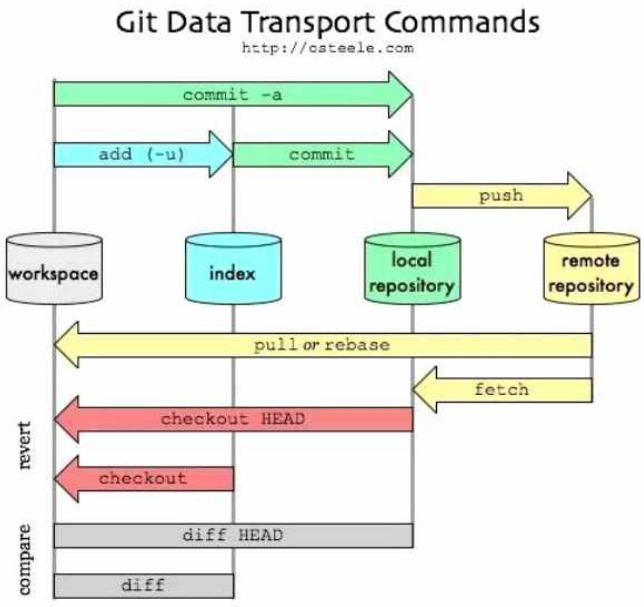
파일을 관리하는 중앙 서버에 데이터베이스가 있어 클라이언트는 서버에서 파일을 받아 사용한다. 중앙에서 관리하기에 누가 어떠한 작업을 하고 있는지 관리하기 쉽다. 단점은 중앙 서버가 작동하지 않으면 각 버전의 정보를 받아올 수 없고, 중앙서버하드에 문제가 생기면 로컬로 가져간 스냅샷(특정 버전, 기록)은 무사하지만 전체적인 히스토리를 잃을 수 있다.
이를 개선한 것이...

[Distributed Model, DVCS, 분산 버전 관리 시스템]

원격 저장소의 히스토리와 모든 것을 복사해 온다. 로컬에서 버전 관리 데이터베이스를 만들어 서버에 올릴 수 있고, 원격 서버에 문제가 생기면 로컬 데이터로 복구해 작업할 수 있다.

3. 리눅스 명령어

-cd: change directory의 약자로 현재 작업 디렉토리 위치를 변경하는 명령어
-mkdir: make directory의 약자로 디렉토리를 생성할 때 사용하는 명령어
-rmdir: remove director의 약자로 빈 디렉토리를 제거하는 명령어
-status: 레퍼지토리의 상태를 확인할 수 있는 명령어
-add: 현재 작업 디렉토리에 있는 내용을 스테이징 영역으로 이동하는 명령어
-commit: 작업한 내용을 로컬 저장소에 기록하는 명령어
-push: 파일을 원격 저장소에 올리는 명령어
-fetch: 원격 저장소의 데이터를 로컬로 가져오는 명령어
-pull: 원격 저장소와 나의 로컬 저장소의 상태를 동일하게 만드는 명령어
-clone: 원격 저장소에 저장된 데이터를 로컬 저장소에 복제하는 명령어



4. 웹 서버와 웹 애플리케이션

[Web Server]
웹 브라우저에서 요청하는 HTML 문서나 파일을 HTTP를 통해 전송해주는 시스템으로 Static Pages(정적 웹페이지)가 웹 서버를 사용한다.

[WAS(Web Application Server)]
HTTP를 통해 애플리케이션을 수행해 주는 미들웨어로 주로 데이터베이스와 같이 수행된다. WAS가 Dynamic Pages(동적 웹페이지)를 사용한다.

[웹 페이지(Web Page)]
크롬, 엣지, 사파리 같은 웹 브라우저에서 보이는 문서이다.

[웹 사이트 (Web Site)]
다양한 방식으로 그룹으로 묶이거나 연결된 웹 페이지들의 모음이다.

[웹 애플리케이션 (Web Application)]
사용자의 입력을 처리하여 웹 페이지를 동적으로 생성하여 실시간에 상호작용하며 보여준다.

5. 동기/비동기, AJAX/SPA, XML/JSON

*Syncronous(동기): 명령을 호출하고 반환할 때까지 기다리며, 반환 시간 동안 다른 동작을 할 수 없다.
*Asyncronous(비동기): 서버와 클라이언트가 동시에 활동이 가능하다.

[AJAX(Asynchronous JavaScript And XML)]
비동기적 웹 어플리케이션 제작을 위한 웹 개발 기법으로 서버와 통신하기 위해 XMLHttpRequest 객체를 사용한다. JSON, XML, HTML 등을 포함한 다양한 포맷을 주고 받을 수 있다. 특징은 '비동기성'으로 사용자의 이벤트가 있으면 전체 페이지가 아닌 일부분만을 업데이트 할 수 있다.

[SPA(Single Page Application)-단일 페이지 어플리케이션(Dynamic Pages)]
서버로부터 새로운 페이지를 불러오지 않고 현재의 페이지를 '동적'으로 다시 작성하여 사용자와 소통하는 웹 애플레리케이션이나 사이트이다. 모든 정적 리소스를 최초에 한 번만 다운로드하여 이후 새로운 페이지 요청 시, 페이지 갱신에 필요한 데이터만을 전달받아 페이지를 갱신한다.

[XML(eXtensible Markup Language)]
HTML과 매우 비슷한 문자 기반의 마크업 언어로 사람과 기계가 동시에 읽기 편한 구조이다. HTML처럼 데이터를 보여주는 목적이 아닌, 데이터를 저장하고 전달할 목적으로만 만들어졌습니다.

[JSON(JavaScript Object Notation)]
브라우저 통신을 위한 속성-값 또는 키-값 쌍으로 이루어진 데이터 포맷이다.
좀 더 쉽게 데이터를 교환하고 저장하기 위하여 만들어진 텍스트 기반의 데이터 교환 표준이다.

[XML과 JSON]

-공통점: 데이터를 저장하고 전달하기 위해 고안되었고, 기계 뿐 아니라 사람도 쉽게 읽을 수 있다. 계층적인 데이터 구조를 가지며, 다양한 프로그래밍 언어에 의해 파싱될 수 있다.
XMLHttpRequest 객체를 이용하여 서버로부터 데이터를 전송받을 수 있습니다.

-차이점: JSON은 종료 태그를 사용하지 않고, JSON의 구문이 XML의 구문보다 더 짧다.
JSON 데이터가 XML 데이터보다 더 빨리 읽고 쓸 수 있고, XML은 배열을 사용할 수 없지만, JSON은 배열을 사용할 수 있다.
XML은 XML 파서로 파싱되며, JSON은 자바스크립트 표준 함수인 eval() 함수로 파싱된다.

*XML 문서는 XML DOM(Document Object Model)을 이용하여 해당 문서에 접근하지만
SON은 문자열을 전송받은 후에 해당 문자열을 바로 파싱하므로, XML보다 더욱 빠른 처리 속도를 보여준다. 따라서 HTML과 자바스크립트가 연동되어 빠른 응답이 필요한 웹 환경에서 많이 사용되고 있다. 하지만 JSON은 전송받은 데이터의 무결성을 사용자가 직접 검증해야 하기에 데이터의 검증이 필요한 곳에서는 스키마를 사용하여 데이터의 무결성을 검증할 수 있는 XML이 아직도 많이 사용된다.

6. 이벤트 드리븐

<Event Driven>

-Event Driven: 이벤트가 발생함에 따라 정의된 핸들러 메서드가 실행되는 방식

-Event Loop: 이벤트를 매니징해준다. 이벤트가 있는지 없는지 확인(Polling)

-Event: 프로그램에 의해 감지되고 처리될 수 있는 동작이나 사건

*다양한 장치에서 이벤트 발생->파일에 이벤트들이 적힘->이벤트 loop이 파일(이벤트 큐)에서 이벤트 읽어오고 지움->이벤트 핸들러에서 실행

-EventEmitter: 이벤트를 발생시키는 객체

-EventDispatcher: 이벤트를 관리하는 객체. 이미터와 핸들러를 매칭시켜준다.

-EventHandler: 이벤트가 발생했을 때 실행되는 객체(ex.~클릭 시 대화창이 뜨도록 작성)

-EventListener: 이벤트를 감시하는 객체. 이벤트와 이벤트핸들러를 연결시켜준다.