2020년도 2학기 컴퓨터공학설계및실험Ⅰ

5주차 Github 사용 예비보고서

20161663 허재성

1. 실습 목적

버전 관리 시스템이 무엇인지 이해하고, Git과 Github의 사용법을 익힌다.

2. 관련 이론

(1) 버전 관리 시스템(VCS)

버전 관리(Version Control)는 동일한 정보, 좁게는 동일한 소스 코드에 대해 여러 버전을 관리하는 것이다. 팀 단위로 개발 중인 소스 코드나 청사진 같은 디지털 문서를 관리하는데 사용된다. 문서들의 변경 사항들에 숫자로 이루어진 버전을 부여해서 구분한다. 소프트웨어 개발 프로젝트에서 가장 주된 업무는 소프트웨어의 설계도라 할 수 있는 소스 코드의 작성이다. 작성된 소스 코드는 기업체 또는 연구소의 핵심 역량이 응축된 핵심 자산으로 어떠한 형태로든 소스 코드를 백업하여 분실의 위험에서 보호하고, 소스 코드 수정 전후 내용을 파악하여 추후 발생할지도 모르는 오류를 수정하기 위한 절차가 필요하다. 버전 관리 시스템은 소스 코드의 수정과 백업 절차를 자동화하여 오류 수정을 도와주는 시스템이다.[1

(2) Git

Git은 컴퓨터 파일의 변경 사항을 추적하고 여러 명의 사용자들 간에 해당 파일들의 작업을 조율하기 위한 분산 버전 관리[2] 시스템이다. Git에서는 소스 코드가 변경된 이력을 쉽게 확인할 수 있고, 특정 시점에 저장된 버전과 비교하거나 특정 시점의 소스 코드로 되돌릴 수 있다. 또한 다른 사람들과 협업할 때, 내가 업로드하려는 소스 파일이 협업하는 다른 사람들이 편집한 파일의 내용과 충돌이 발생할 경우 서버에 업로드할 때 경고 메시지를 발생시켜 다른 사람이 작성한 소스 코드를 덮어 써버리는 일을 방지할 수 있다.

(3) Git 이외의 버전 관리 시스템, 다른 버전 관리 시스템과 비교하여 Git의 장단점

버전 관리 시스템(VCS)의 종류에는 로컬 VCS, 중앙집중식 VCS, 분산 VCS가 있다. Git은 위에서 언급한 분산 VCS이다. 분산 VCS와 비교해서 로컬 VCS와 중앙집중식 VCS이 있다.

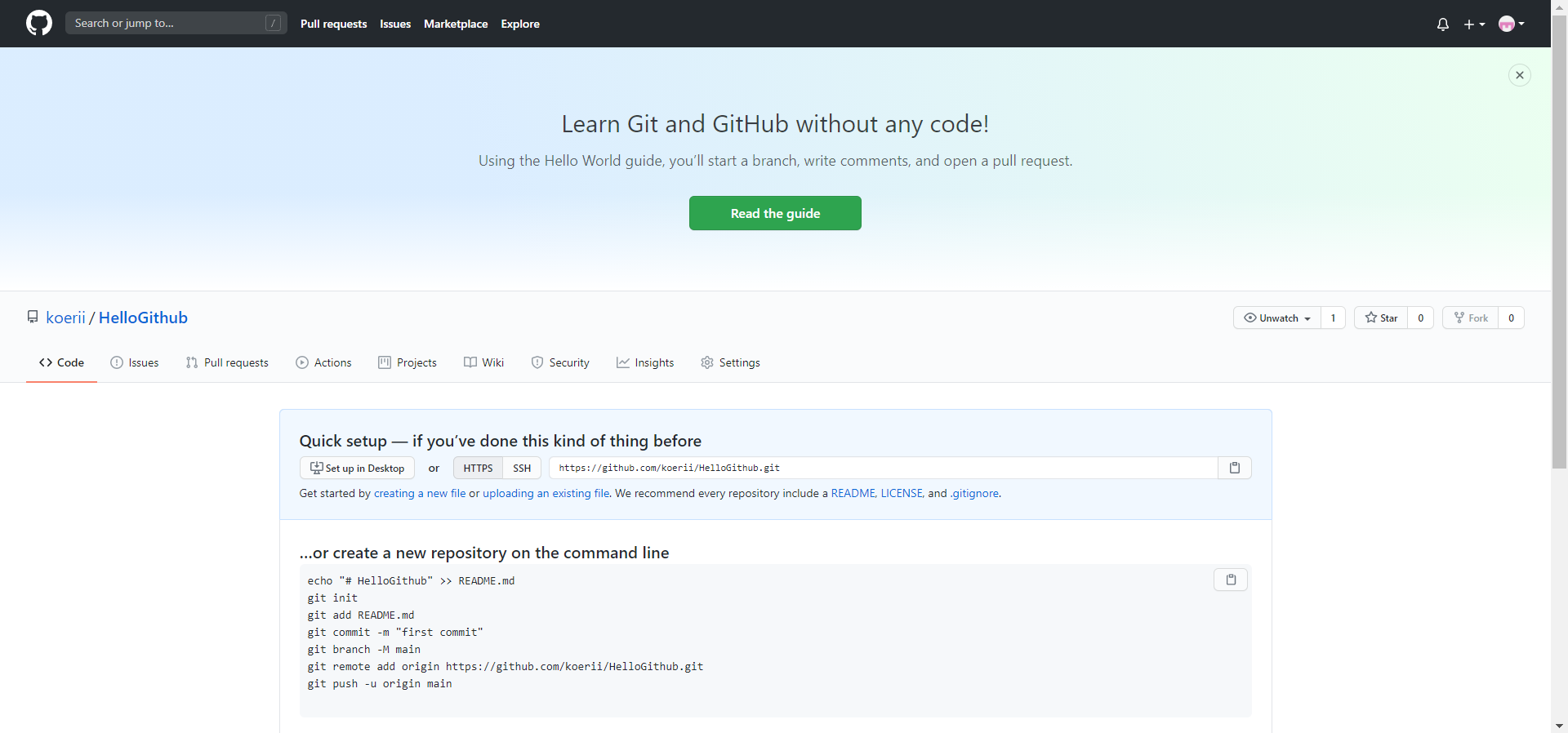
로컬 VCS는 원격 저장소가 없이 로컬(개인 PC)에서 버전 관리를 하는 것이다. 오프라인 상태에서도 버전 관리가 가능하다는 장점은 있지만 로컬에서만 버전 관리를 할 수 있으므로 개인이 아닌 여러 사람의 협업이 불가능하고 로컬 저장소 자체가 사라지면 복구할 수 없다는 단점이 있다.

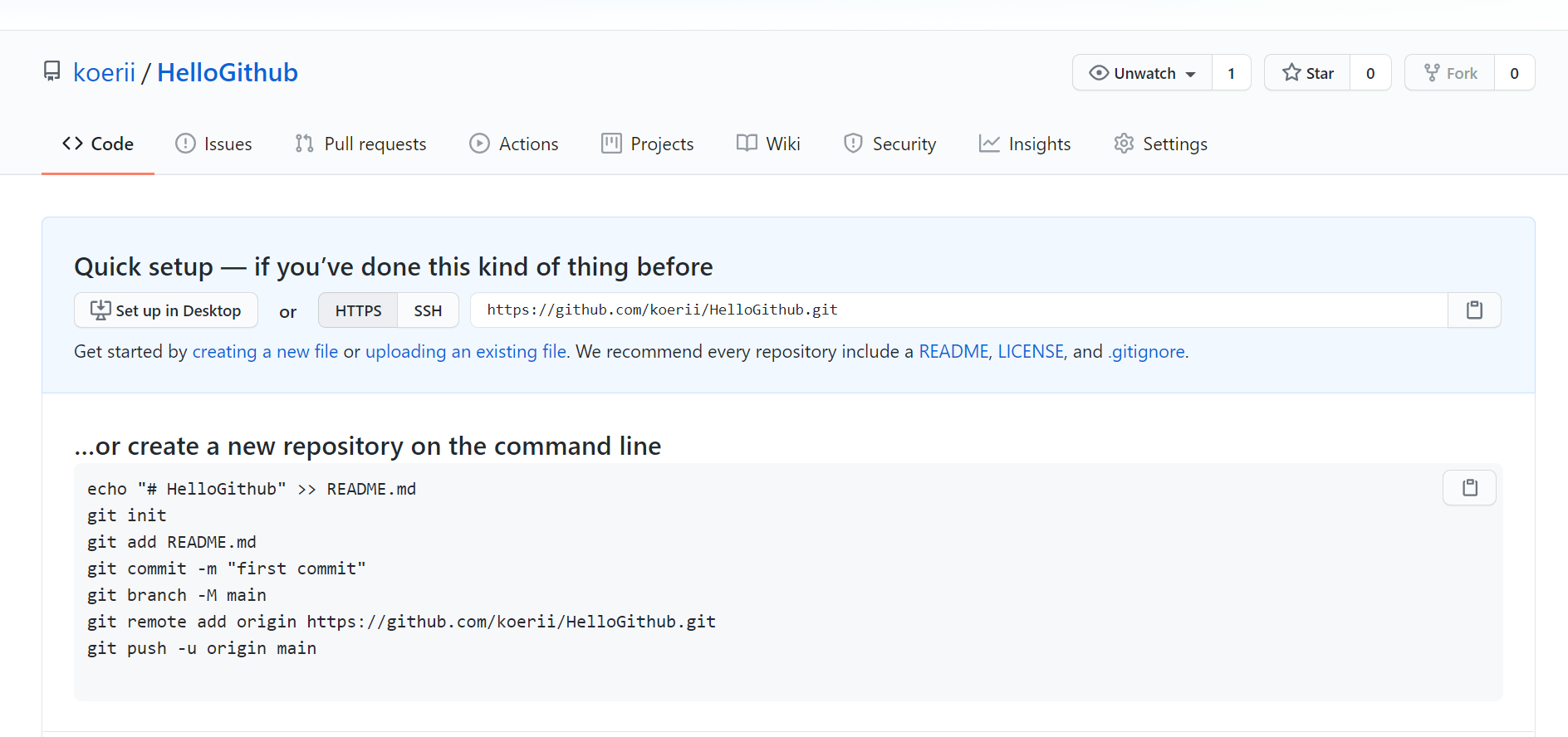
중앙집중식 VCS는 다른 개발자들과 협업하기 위해 소스 파일을 관리하는 별도의 서버의 원격 저장소에 저장한다. 로컬 저장소에 저장하는 것이 아니기 때문에 로컬 VCS에서와 달리 개인 PC에 문제가 생겨도 원격 저장소에 저장한 소스 파일에는 문제가 없다는 장점이 있으며, 다른 사람과 협업이 가능하다는 장점이 있다. 하지만 원격 저장소가 있는 서버에 문제가 발생하여 접속할 수 없다면 그 동안 다른 사람과의 협업도 불가하고 백업할 수도 없다. 그리고 원격 저장소가 있는 서버의 PC가 문제가 생기면 해당 서버를 이용하는 모든 사용자들이 소스 파일을 잃을 수도 있다는 단점이 있다.

Git이 포함된 분산 VCS는 로컬 VCS와 중앙집중식 VCS 각각의 장점을 모아둔 VCS이다. 소스파일의 백업과 버전 관리가 로컬 저장소와 원격 저장소 모두에 저장하여 로컬 저장소에 문제가 생길 경우 원격 저장소를 통해 백업 및 협업이 가능하고, 원격 저장소에 문제가 생길 경우 로컬 저장소에서 협업은 불가능하지만 작업 및 백업이 가능하다. 로컬 저장소에 저장한 소스 파일을 언제든지 원격 저장소에 옮길 수 있고(push), 반대로 원격 저장소에 저장된 소스 파일을 로컬 저장소에 복제(clone) 및 보내기(pull)를 할 수 있다. Git의 장점은 로컬 VCS와 중앙집중식 VCS의 장점을 취합하여 유연한 버전 관리와 협업이 가능하다. 하지만 Git의 경우, 기존에 있던 다른 형상 관리 시스템보다 덜 직관적이고 배우기 어렵다는 단점이 있다.[3]

3. 실습 방법

(1) Github 계정을 생성한 뒤 생성된 계정 화면(Repository)을 촬영할 것

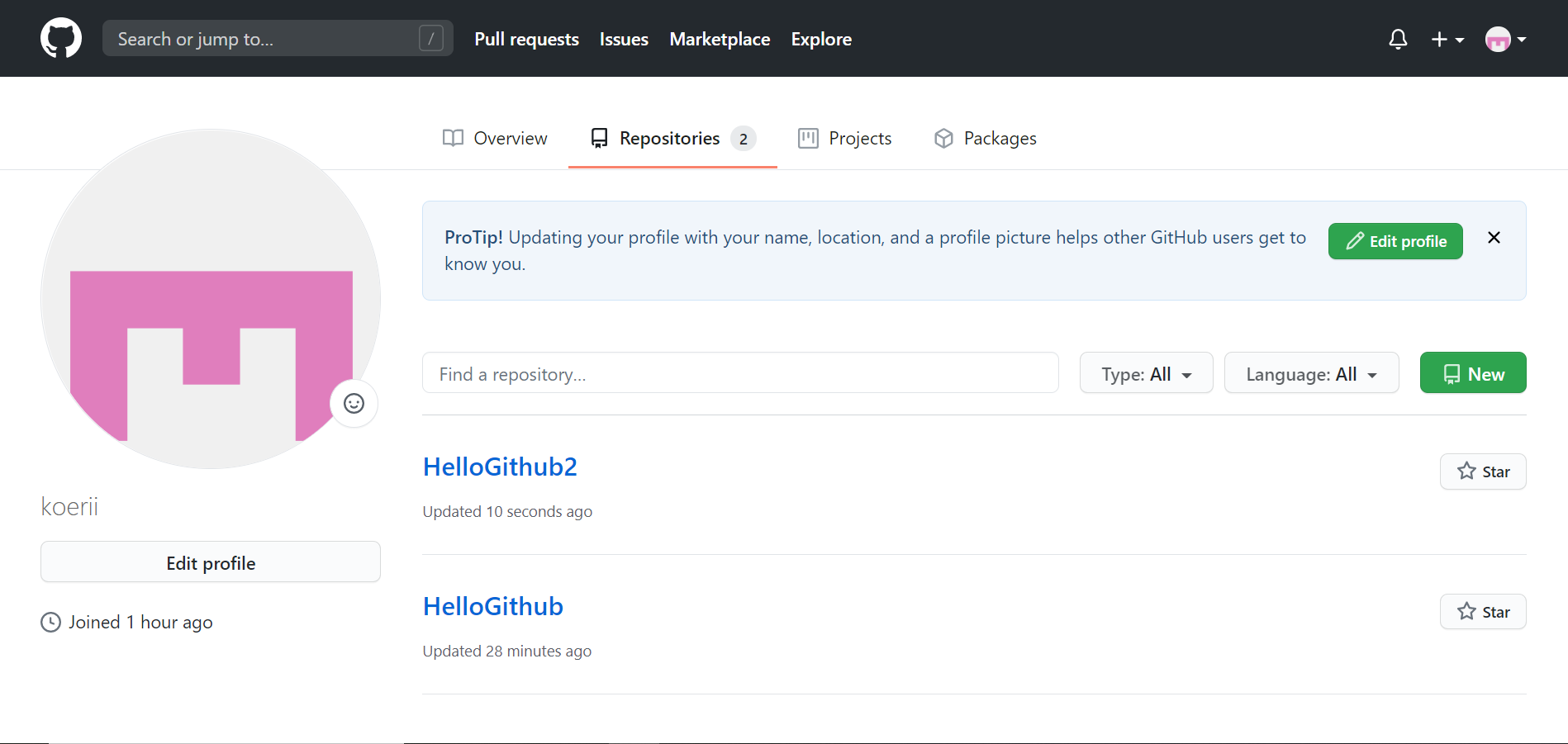
실습을 위해 koerii라는 Github 계정을 만들고 HelloGithub라는 원격 저장소(Repository)를 생성하였다. 사진은 해당 Repository 화면을 캡처한 것이다.



(2) Repository 목록으로 접속할 수 있는 주소를 첨부하라.

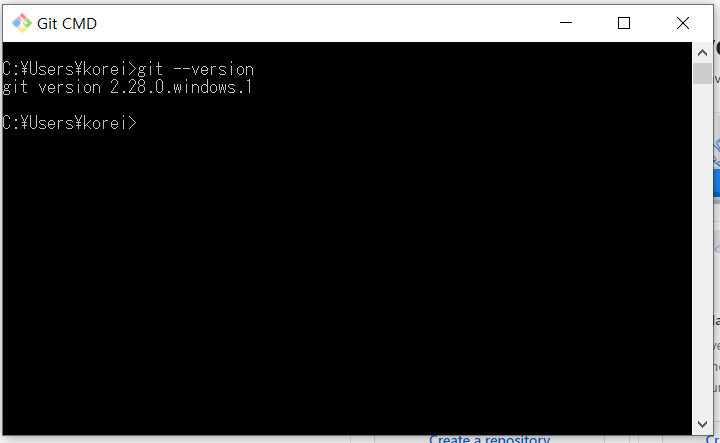
<https://github.com/koerii?tab=repositories>

위의 주소로 접속하면 다음과 같은 페이지로 이동하여 현재 존재하는 repository들을 확인할 수 있다.



. 위에서 생성한 HelloGithub repository뿐만 아니라 임의로 만든 HelloGithub2 repository도 확인할 수 있다

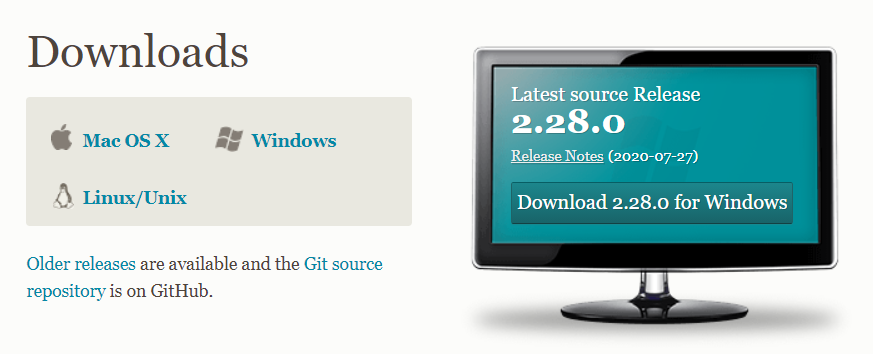
(3) Git을 설치한 뒤 CMD나 bash에서 git –version을 사용하여 설치된 것을 촬영하라.



Git CMD를 실행 후 git --version 명령어를 입력해 현재 설치된 git의 버전을 확인했다. 확인 결과 2.28.0 버전임을 알 수 있다.

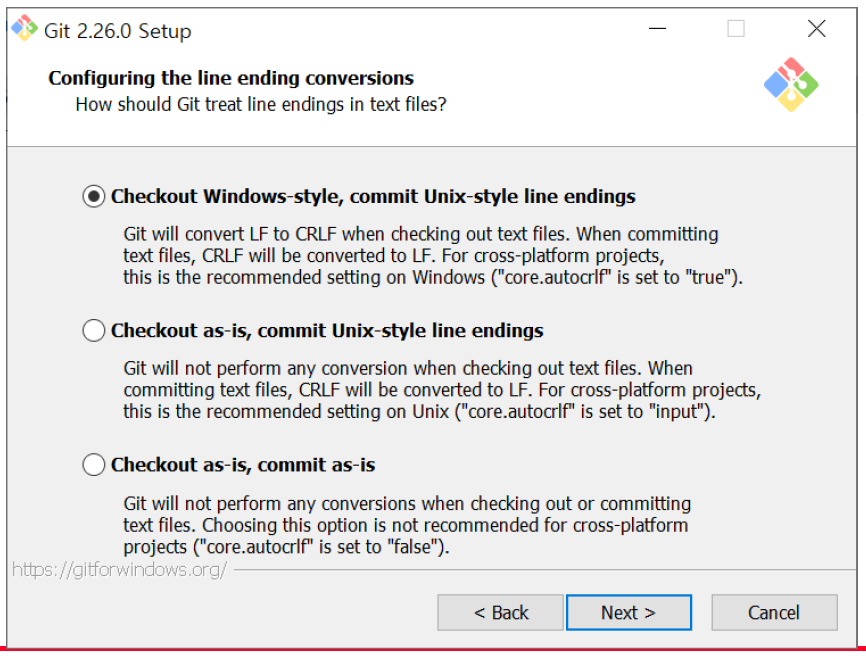
(4) Git을 컴퓨터에 설치해보고 그 과정을 요약하시오.

<https://git-scm.com/downloads> 에 접속하여 해당하는 컴퓨터의 컴퓨터의 운영체제에 맞는 git을 설치한다. 나의 컴퓨터 운영체제는 Windows 10이므로 Windows OS에 해당하는 git을 설치한다.



Windows를 클릭하면 자동으로 Git-2.28.0-64-bit.exe가 설치된다. 32비트 컴퓨터를 사용하면 32비트 설치 파일이 다운로드된다. 설치 파일을 실행하여 설치하면서 원하는 대로 설정을 해준다. 편집기의 경우 어떤 것을 해도 상관없지만 지난 실습에서 사용한 Vim을 사용하도록 설치했다. 원하는 설정을 선택 후 Next를 클릭하여 다음 설정으로 넘어간다.

Next를 클릭하다 보면 다음과 같은 설정 선택 화면이 나온다.



CRLF와 LF의 의미는 다음과 같다. CRLF와 LF 모두 line ending 문자로 한 문장이 끝났을 때 개행하는 문자를 의미한다. LF(Line Feed)는 \n을 의미하고 CR(Carriage-Return)는 \r을 의미한다. Windows 운영체제에서는 line ending으로 CR(Carriage-Return, \r)와 LF(Line Feed, \n)을 사용하고 Unix, MacOS는 LF만 사용한다. [4]

디폴트로 설정된 첫 번째 선택지(Checkout Windows-style,­­­ commit Unix-style line endeings)는 텍스트 파일을 체크할 때, LF를 Windows의 CRLF로 자동으로 변경한다. commit할 때, Line Ending 문자(한 문장이끝났을 때 개행 문자)로 CRLF를 LF로 자동으로 바꾼다. 협업할 때 Windows 환경에서 세팅이 권장된다. 체크할 때 자동으로 변경되면 실제 코드의 변경을 의도하지 않았는데도 변경한 것으로 간주되어 commit이 되어 문제가 될 수 있다.

두 번째 선택지(Checkout as-is, commit Unix-style line endings)는 텍스트 파일을 체크할 때, CRLF로 자동으로 변경하지 않는다. commit할 때, 자동으로 Line Ending문자를 CRLF를 LF로 바꾼다. 협업할 때, Unix 환경에서 세팅하는 것이 권장된다.

마지막 선택지(Checkout as-is, commit as-is)는 텍스트 파일을 체크할 때도, commit할 때에도 어떤 변경을 하지 않는다. 협업 프로젝트에서 권장되지 않는다.

4. 기타

분산 버전 관리(Distributed Revision Control 또는 Distributed Version Control)는 소프트웨어 버전 관리를 위한 시스템으로 각 개발자가 중앙 서버에 접속하지 않은 상태에서도 코드 작업을 할 수 있는 것이 특징이다. 분산 모델은 독립 개발자들과 함께 리눅스 커널 프로젝트와 같은 대형 프로젝트에 적합한데, 개발자들이 독립적으로 작업한 다음에 변경 사항을 병합(merge) 또는 거절할 수 있기 때문에 유연한 버전 관리가 가능하다.[2]

소프트웨어 구성 관리(Software Configuration Management)는 형상 관리라고도 하며 소프트웨어의 변경 사항을 체계적으로 추적하고 통제하는 것으로, 형상 관리는 일반적인 단순 버전 관리 기반의 소프트웨어 운용을 좀 더 포괄적인 학술 분야의 형태로 넓히는 근간을 이야기한다.[5]

5. 참고 문헌

[1] 위키백과, “버전 관리”, https://ko.wikipedia.org/wiki/%EB%B2%84%EC%A0%84\_%EA%B4%80%EB%A6%AC

[2] 위키백과, “분산 버전 관리”, https://ko.wikipedia.org/wiki/%EB%B6%84%EC%82%B0\_%EB%B2%84%EC%A0%84\_%EA%B4%80%EB%A6%AC

[3] git, “1.1 시작하기 - 버전 관리란?” https://git-scm.com/book/ko/v2/%EC%8B%9C%EC%9E%91%ED%95%98%EA%B8%B0-%EB%B2%84%EC%A0%84-%EA%B4%80%EB%A6%AC%EB%9E%80%3F

[4] 위키백과, “새줄 문자”, https://ko.wikipedia.org/wiki/%EC%83%88%EC%A4%84\_%EB%AC%B8%EC%9E%90

[5] 위키백과, “형상 관리”, https://ko.wikipedia.org/wiki/%EA%B5%AC%EC%84%B1\_%EA%B4%80%EB%A6%AC