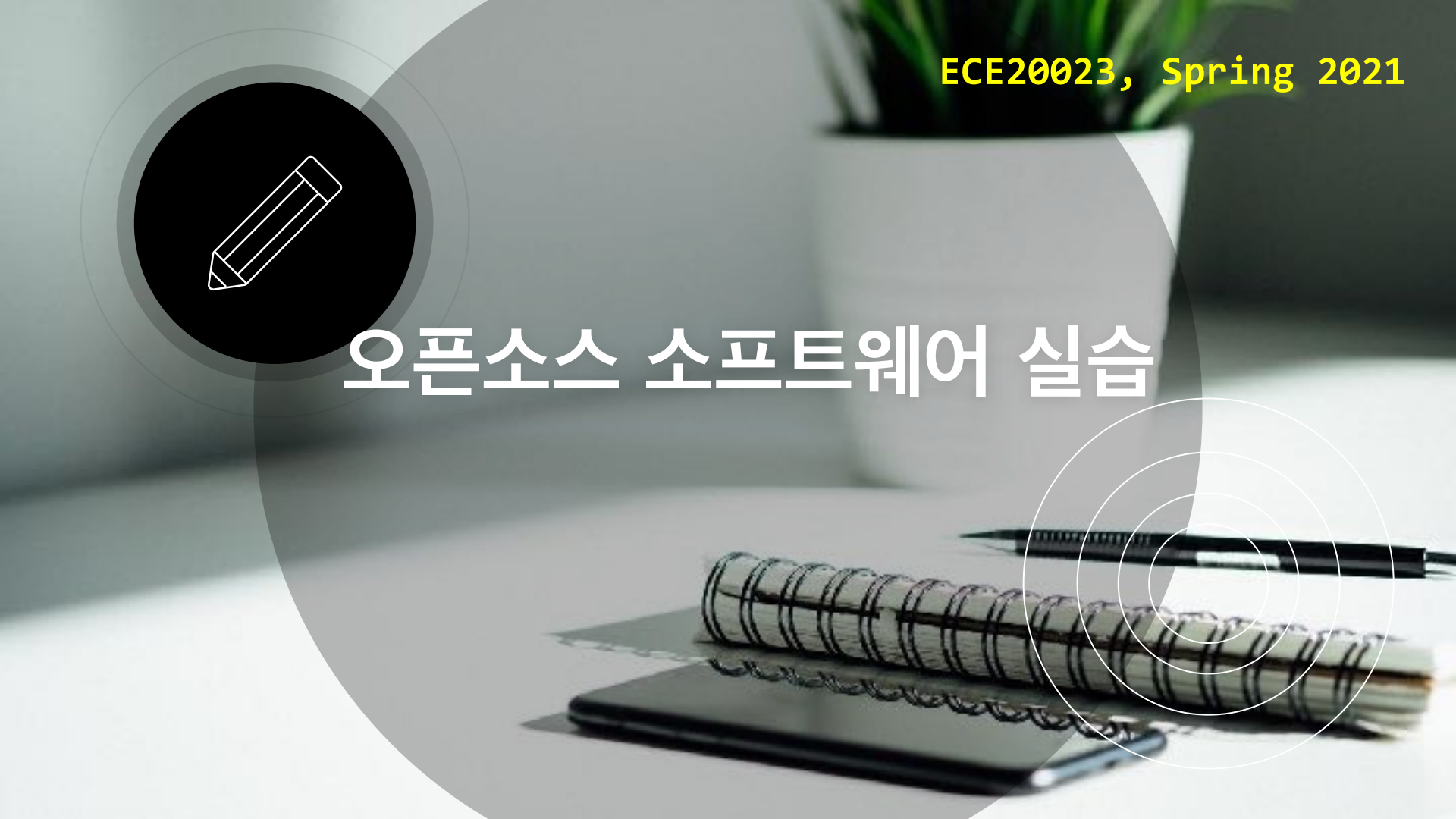
ECE20023, Spring 2021



오픈소스 소프트웨어 실습

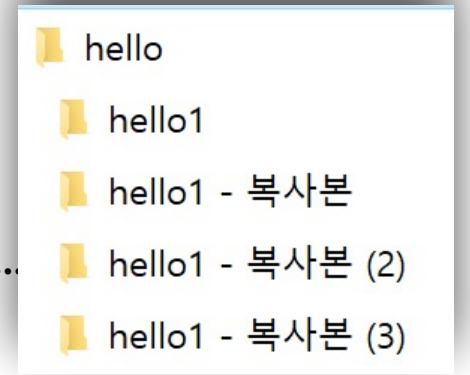
11

Git

VCS (Version Control System)

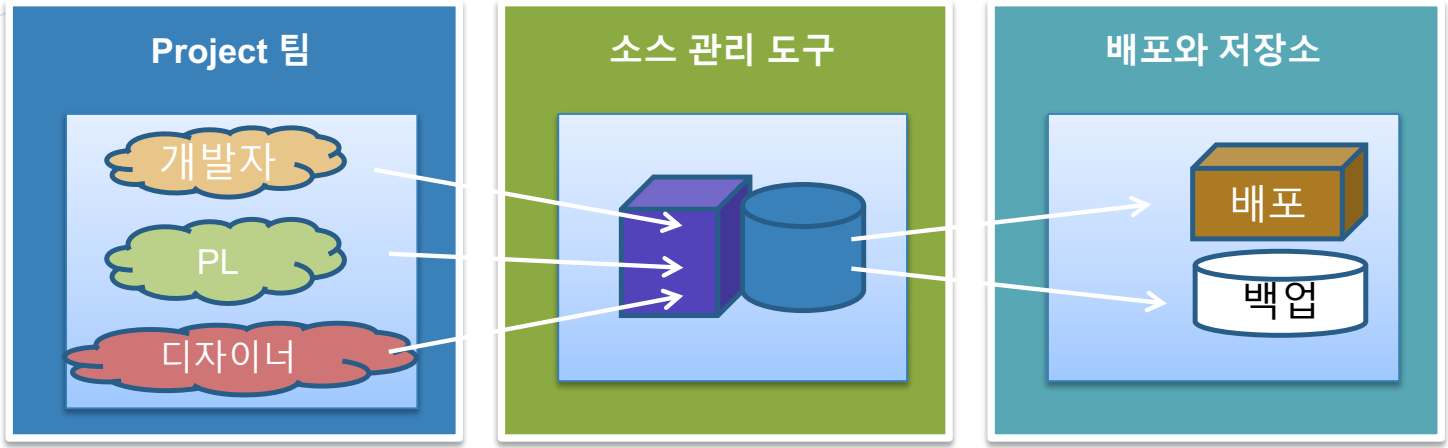
소스 관리는 어떻게?

- 개발할 때 소스 관리는?
  - 각 폴더는 무엇을 위한 버전인가?
  - 왜, 언제 수정했지?
  - 1년 후에 보면 무슨 소스인지 알까?
  - 새 기능 개발하기 전 버전을 다시 가져오려면…
- 협업할 때 소스 관리는?
  - 압축해서 Gmail로?
  - Drive를 이용해서 share?
  - 서버에 올려놓고 같이 접속해서 고치면?
  - 최종본은 누가?



VCS (Version Control System)

- SCM (Source Control System, Source Configuration Mangement)
- 프로젝트 개발(협업)할 때 소스코드, 문서 등 버전을 관리하는 시스템

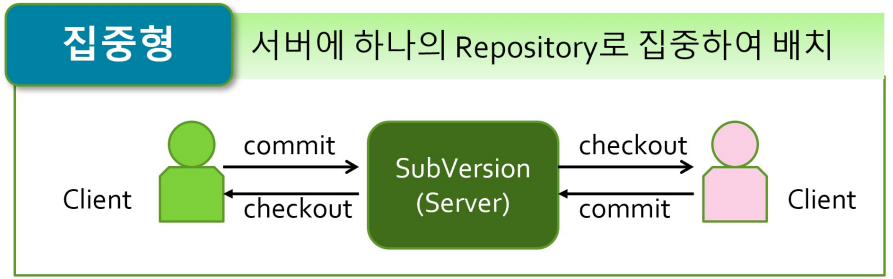


VCS 종류

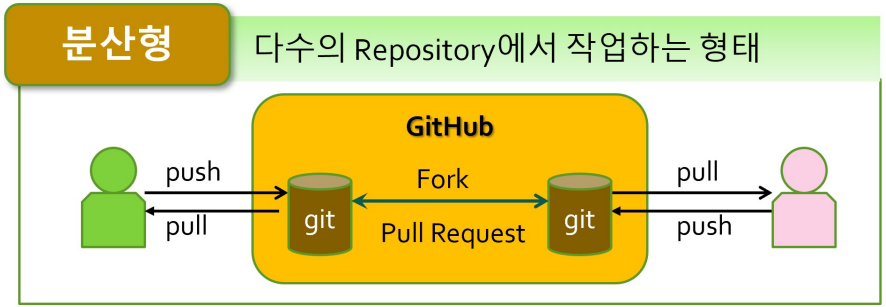
- **CVCS (Centralized VCS)**
  - 중앙집중식 / Client-Server 방식
  - 서버에 소스파일과 버전 히스토리 저장
  - 단점 : 서버가 고장 나면 협업이 어려움
  - CVS (Concurrent Version System), SVN (SubVersion) 등
- **DVCS (Distributed VCS)**
  - 분산 관리 시스템
  - 소스와 히스토리를 여러 PC에 분산 저장
  - 서버 장애시 로컬 저장소를 이용하여 중앙 저장소 복원 가능
  - Git, Mercurial, Bazaar 등

VCS 종류

- CVCS (Centralized VCS)



- DVCS (Distributed VCS)



Git

- 2005년, 리눅스 커널 개발자, 리누스 토르발스에 의해 개발
- DVCS(분산 버전 관리 시스템) : 빠른 수행속도
- 빠른 협업 환경 조성
- 지속적인 버전 관리가 필요한 경우 사용(SW, Design, …)
- 수천 개의 브랜치(branch) 작업 동시 수행
- 누가, 무엇을, 언제, 왜, 어떻게 수정했는지 코드 리뷰가 가능
- 대형 프로젝트의 버전 관리가 가능함.
- 중앙 서버와 독립적으로 Local에서 완벽하게 소스와 이력 관리
- Local에서 소스 버전 관리할 때 사용

Git vs Github

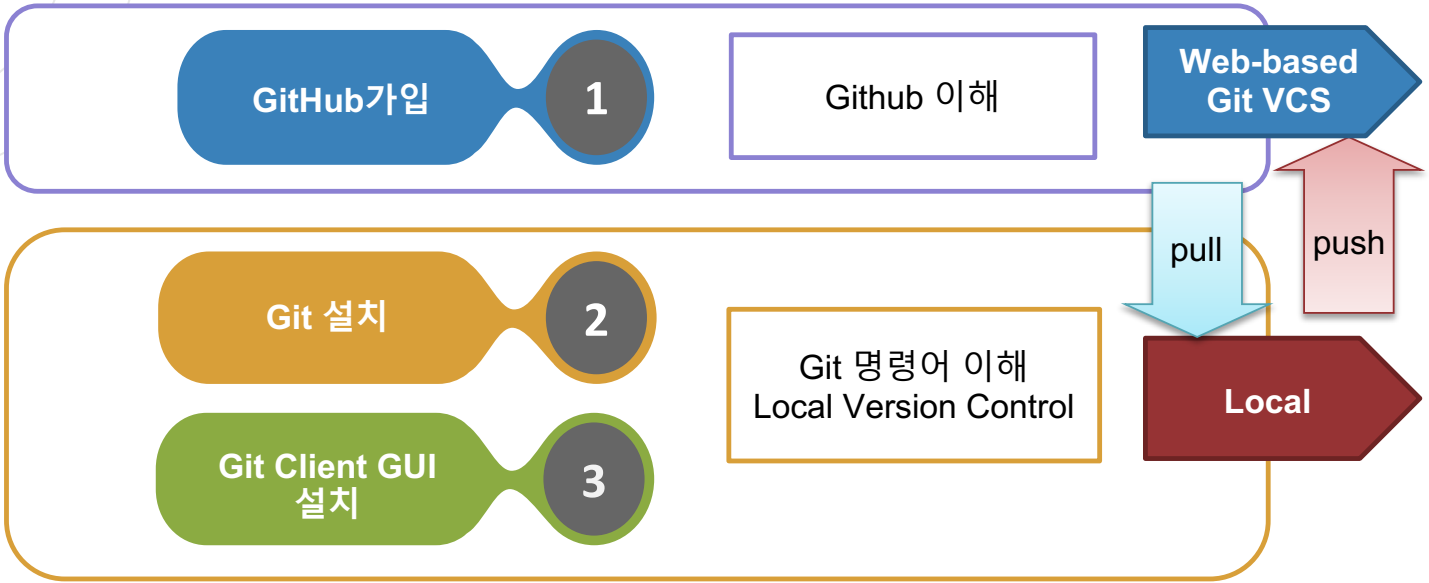
Git

- 분산 버전 관리 시스템 소프트웨어
- 로컬저장소(Local Repository)사용
- Local 내에서 git을 이용하여 버전 관리

GitHub

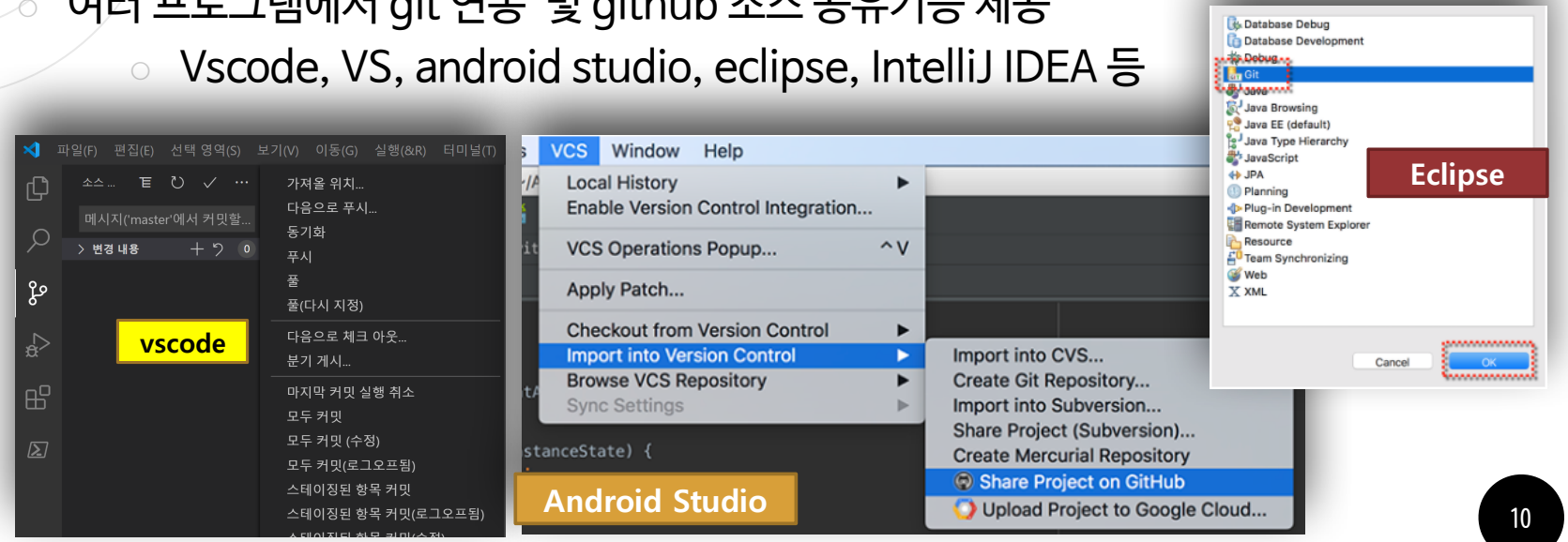
- 원격저장소(Remote Repository) 제공.
- Local에서 관리하는 소스코드를 업로드후 공유
- 다른 개발자와 협업할 때 사용
- https://github.com/

Git & Github


GitHub가입
1
Github 이해
Web-based
Git VCS
pull
push
Git 설치
2
Git 명령어 이해
Local Version Control
Local
Git Client GUI
설치
3

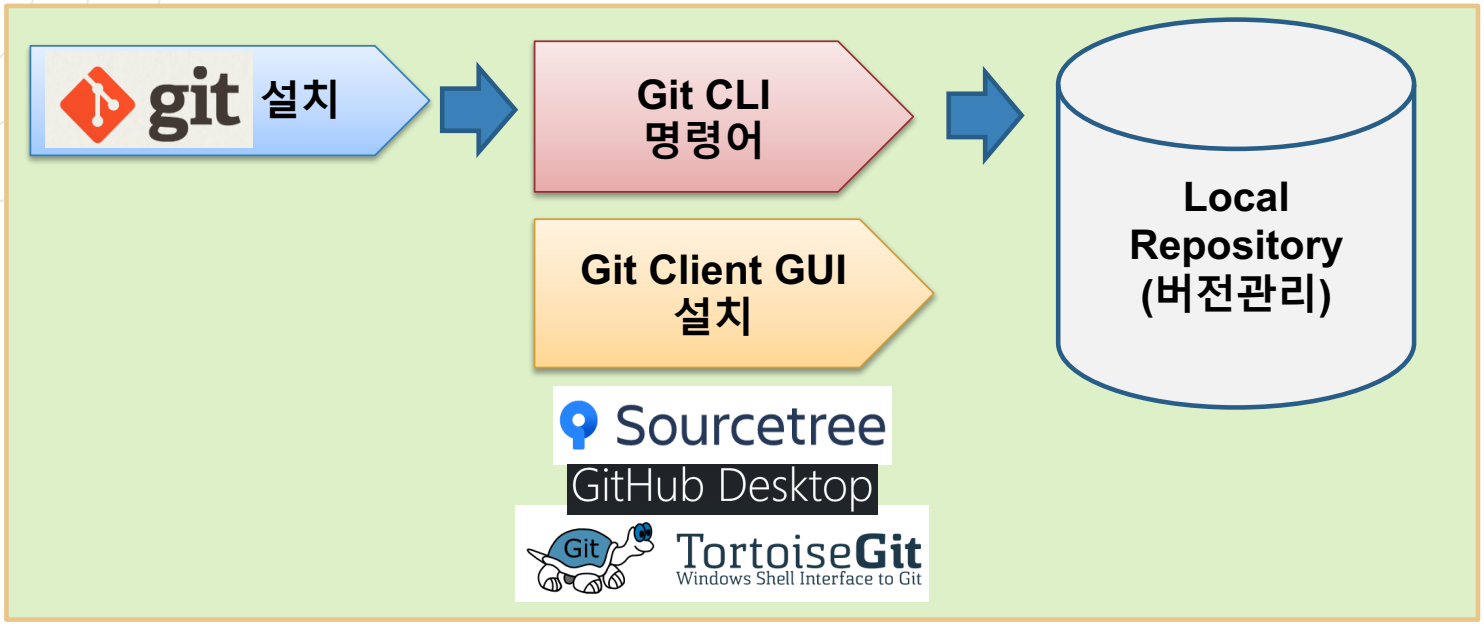
Git 연동

- Windows, Mac OS, 리눅스, 유닉스 등 지원
- 여러 프로그램에서 git 연동 및 github 소스 공유기능 제공
  - Vscode, VS, android studio, eclipse, IntelliJ IDEA 등



Git 사용

https://git-scm.com/

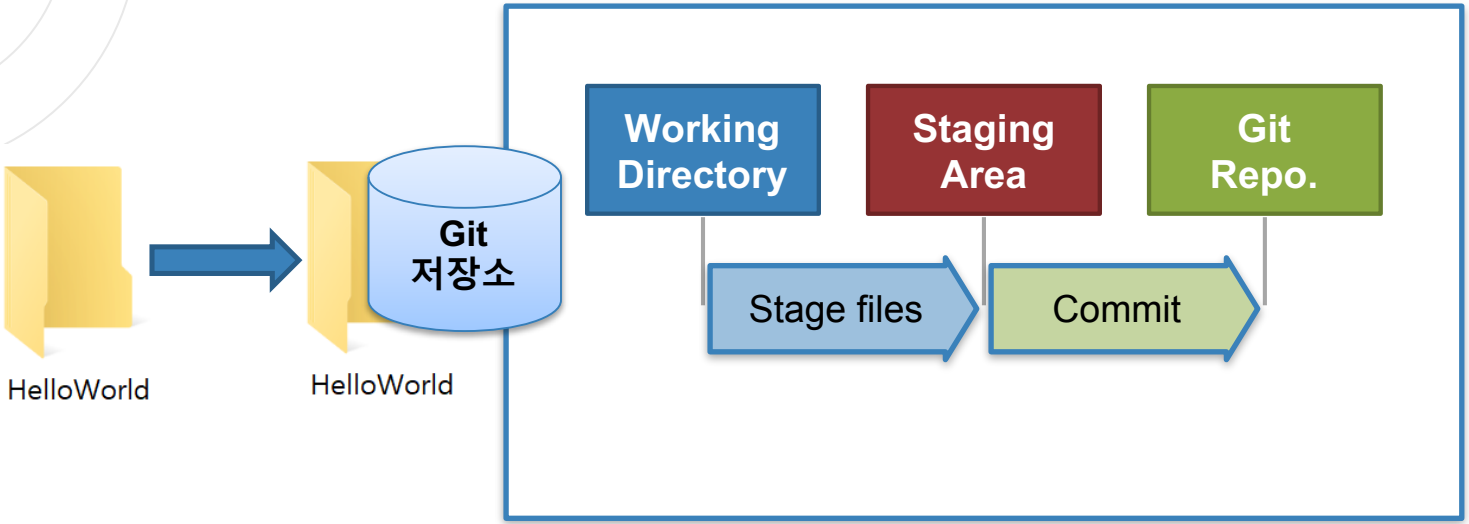


Git : 사용자, 이메일 설정

- Local PC 버전관리
  - Windows : Git Bash 프로그램으로 설정
  - Mac : terminal 사용하여 설정
- Server 에서 버전관리
  - 서버에 접속하여 설정

```
walab-HGU:~:> git config --global user.name "user1"
walab-HGU:~:> git config --global user.email "user1@gmail.com"
walab-HGU:~:> git config --global color.ui auto
walab-HGU:~:> git config --global --list
user.name=user1
user.email=user1@gmail.com
color.ui=auto
```

Git 버전 관리 과정


Working Directory
Staging Area
Git Repo.
Stage files
Commit
Git 저장소
HelloWorld
HelloWorld

Repository 생성

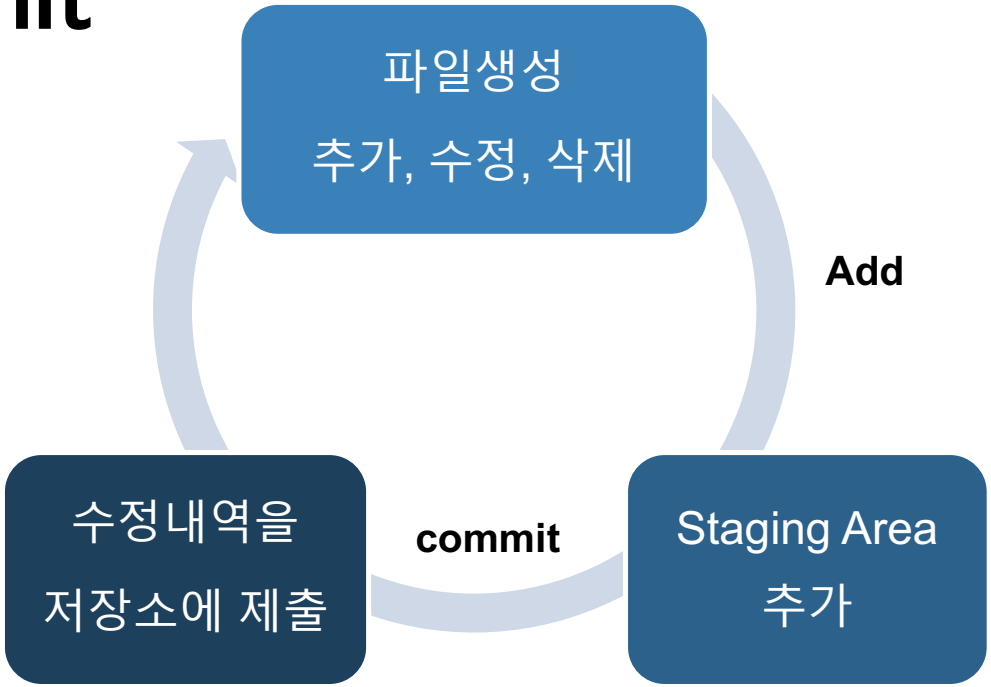
- 로컬 저장소 생성

```
walab-HGU:~/lab6:> mkdir hello
walab-HGU:~/lab6:> cd hello
walab-HGU:~/lab6/hello:> git init
Initialized empty Git repository in /home/User1/2021OSS/lab6/hello/.git/
```

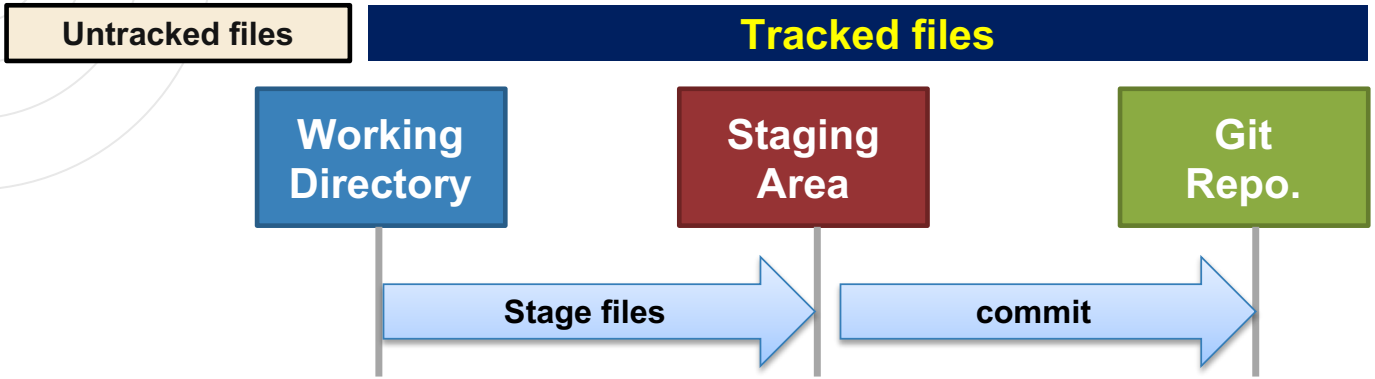
- 원격 저장소 복제
  - 원격 저장소 Repo URL 복사

```
walab-HGU:~/lab6:> git clone <GitHub의 Repository주소>
Cloning into 'simple-chat-client-server'...
remote: Enumerating objects: 7, done.
remote: Total 7 (delta 0), reused 0 (delta 0), pack-reused 7
Unpacking objects: 100% (7/7), done.
walab-HGU:~/lab6:> ls
hello  simple-chat-client-server
```

add/commit



add/commit




```
git add index.html  index2.html
git add *.*
git add *.html
```

```
git commit
git commit -m “commit msg”
git commit -am “commit msg”
```

git rm

- Untracked file 삭제
  - rm 명령어를 사용하여 삭제
- Tracked file 삭제
  - 로컬 디렉터리와 git 저장소에서 모두 삭제
  - git rm ⟨filename⟩
- git에서만 삭제, 로컬 디렉터리에는 삭제하지 않음
  - git rm --cached ⟨filename⟩

git status

- 현재 파일 상태를 확인하기 위해 사용하는 명령
  - Untracked file 상태
  - Staged 전 상태
  - commit 전 상태

```
walab-HGU:~/lab6/hello:> touch hello.c
walab-HGU:~/lab6/hello:> git status
walab-HGU:~/lab6/hello:> git add *.c
walab-HGU:~/lab6/hello:> git status
walab-HGU:~/lab6/hello:> git commit -m "create hello.c"
[master (root-commit) 0d01bae] create hello.c
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 hello.c
walab-HGU:~/lab6/hello:>
```

git log

- Commit history를 볼 수 있음

```
git log
git log -3
git log -p -1
git log -2 --oneline
git log --author=Brandon
git log - - pretty=oneline -graph
```

git diff

- 커밋 내역을 비교하거나 commit과 working tree 변경 내역 확인
- 워킹 디렉터리와 Staging area 비교

```
walab-HGU:~/lab6/hello:> git diff
```

- Commit내용과 Staging area 비교

```
walab-HGU:~/lab6/hello:> git diff --cached
walab-HGU:~/lab6/hello:> git diff --staged
```

- Commit과 다른 Commit 비교

```
walab-HGU:~/lab6/hello:> git diff commit1_checksum commit2_checksum
```

gitignore 파일

- Git 저장소에서 관리가 필요없는 파일이나 폴더 지정
  - 실행파일, 목적파일
  - Hidden file
  - Id/password 등 보안 정보
  - 개인 파일
  - 실행파일 저장된 폴더(/bin, /out)
- Git 저장소 폴더에 .gitignore 파일 생성
- .gitignore 파일 생성 사이트

  https://www.gitignore.io/

과제설명

3. git 명령어 정리(dynalist or blog …)
   - git add index.html index2.html
   - git add *.*
   - git add *.html

4. 아래 git 실습과제를 수행하고, 실습내용을 text로 복사 붙여넣기 + 마지막 단계 화면캡처
   a. 원하는 Repository선택 및 URL복사(https://github.com/jerry10004/calculator-2)
   b. walab 서버 접속
   c. ~/2021OSS 로 이동
   d. git clone **https://github.com/jerry10004/calculator-2**
   e. ls -al 실행
   f. calculator 폴더 생성되었는지 확인 및 이동
   g. Makefile을 myMakefile로 복사
   h. myMakefile 첫줄에 본인의 학번과 이름 주석문 추가
   i. cat myMakefile 실행
   j. git status 확인
   k. git add 사용하여 staging area에 추가
   l. 메세지와 함께 git commit
   m. git log -5 --oneline 실행
   n. myMakefile 이용한 컴파일
   o. 프로그램 실행파일 실행
   p. ls -al 실행화면