설치

OS X용 git 다운로드

Windows용 git 다운로드

Linux용 git 다운로드

새로운 저장소 만들기

폴더를 하나 만들고, 그 안에서 아래 명령을 실행하세요.

```
git init
```

새로운 git 저장소가 만들어집니다.

저장소 받아오기

로컬 저장소를 복제(clone)하려면 아래 명령을 실행하세요.

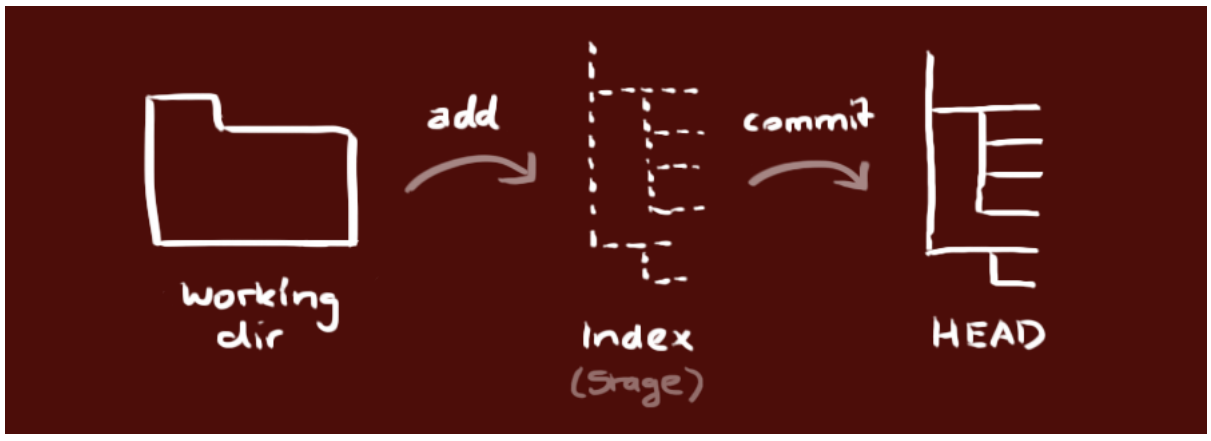
```
git clone /로컬/저장소/경로
```

원격 서버의 저장소를 복제하려면 아래 명령을 실행하세요.

```
git clone 사용자명@호스트:/원격/저장소/경로
```

작업의 흐름

여러분의 로컬 저장소는 git이 관리하는 세 그루의 나무로 구성돼있어요.
첫번째 나무인 `작업 디렉토리(Working directory)` 는
실제 파일들로 이루어져있고, 두번째 나무인 `인덱스(Index)` 는
준비 영역(staging area)의 역할을 하며, 마지막 나무인 `HEAD` 는
최종 확정본(commit)을 나타내요.



추가와 확정(commit)

변경된 파일은 아래 명령어로 (**인덱스**에) 추가할 수 있어요.

```
git add <파일 이름>
git add *
```

이것이 바로 git의 기본 작업 흐름에서 첫 단계에 해당돼요.

하지만 실제로 변경 내용을 확정하려면 아래 명령을 내려야 한답니다.

```
git commit -m "이번 확정본에 대한 설명"
```

자, 이제 변경된 파일이 **HEAD**에 반영됐어요.

하지만, 원격 저장소에는 아직 반영이 안 됐답니다.

변경 내용 발행(push)하기

현재의 변경 내용은 아직 로컬 저장소의 **HEAD** 안에 머물고 있어요.

이제 이 변경 내용을 원격 서버로 올려봅시다. 아래 명령을 실행하세요.

```
git push origin master
```

(다른 가지를 발행하려면 *master*를 원하는 가지 이름으로 바꿔주세요.)

만약 기존에 있던 원격 저장소를 복제한 것이 아니라면,

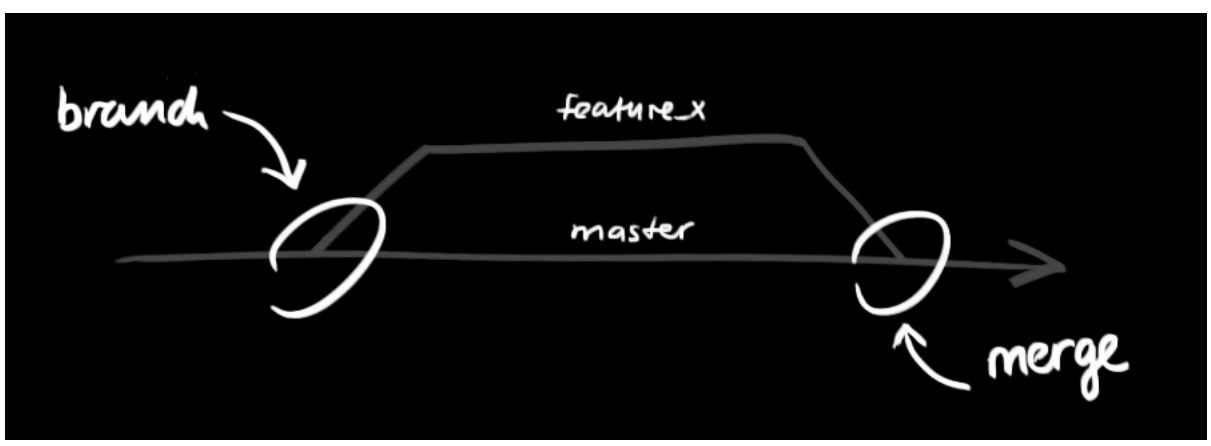
원격 서버의 주소를 git에게 알려줘야 해요.

```
git remote add origin <원격 서버 주소>
```

이제 변경 내용을 원격 서버로 발행할 수 있어요.

가지(branch)치기

가지는 안전하게 격리된 상태에서 무언가를 만들 때 사용해요.
여러분이 저장소를 새로 만들면 기본으로 *master 가지*가 만들어집니다.
이제 다른 가지를 이용해서 개발을 진행하고, 나중에 개발이 완료되면
master 가지로 돌아와 병합하면 돼요.



아래 명령으로 "feature_x"라는 이름의 가지를 만들고 갈아탑니다.

```
git checkout -b feature_x
```

아래 명령으로 master 가지로 돌아올 수 있어요.

```
git checkout master
```

아래 명령으로는 가지를 삭제할 수 있어요.

```
git branch -d feature_x
```

여러분이 새로 만든 가지를 원격 저장소로 전송하기 전까지는
다른 사람들이 접근할 수 없어요.

```
git push origin <가지 이름>
```

갱신과 병합(merge)

여러분의 로컬 저장소를 원격 저장소에 맞춰 갱신하려면
아래 명령을 실행하세요.

```
git pull
```

이렇게 하면 원격 저장소의 변경 내용이 로컬 작업 디렉토리에
*받아지고(fetch), 병합(merge)*된답니다.
다른 가지에 있는 변경 내용을 현재 가지(예를 들면, master 가지)에
병합하려면 아래 명령을 실행하세요.

```
git merge <가지 이름>
```

첫번째 명령이든 두번째 명령이든, git은
자동으로 변경 내용을 병합하려고 시도해요.
문제는, 항상 성공하는 게 아니라 가끔
*충돌(conflicts)*이 일어나기도 한다는 거예요.

이렇게 충돌이 발생하면, git이 알려주는 파일의 충돌 부분을
여러분이 직접 수정해서 병합이 가능하도록 해야 하죠.
충돌을 해결했다면, 아래 명령으로 git에게
아까의 파일을 병합하라고 알려주세요.

```
git add <파일 이름>
```

변경 내용을 병합하기 전에, 어떻게 바뀌었는지 비교해볼 수도 있어요.

```
git diff <원래 가지> <비교 대상 가지>
```

꼬리표(tag) 달기

소프트웨어의 새 버전을 발표할 때마다 꼬리표를 달아놓으면 좋아요.
(물론 꼬리표는 SVN 등에 이미 존재하는 기능이지요.)
아래 명령을 실행하면 새로운 꼬리표인 *1.0.0*을 달 수 있어요.

```
git tag 1.0.0 1b2e1d63ff
```

위 명령에서 *1b2e1d63ff* 부분은 꼬리표가 가리킬 확정본 식별자입니다.
아래 명령으로 확정본 식별자를 얻을 수 있어요.

```
git log
```

확정본 식별자의 앞부분 일부만 입력해도 꼬리표를 붙일 수 있지만,
그 일부분이 반드시 고유하다는 조건이 필요해요.

로컬 변경 내용 되돌리기

만약 여러분이 (물론 그럴 일은 없겠지만 ;) 실수로 무언가 잘못한 경우,
아래 명령으로 로컬의 변경 내용을 되돌릴 수 있어요.

```
git checkout -- <파일 이름>
```

위 명령은 로컬의 변경 내용을 변경 전 상태(HEAD)로 되돌려줘요.
다만, 이미 인덱스에 추가된 변경 내용과
새로 생성한 파일은 그대로 남는답니다.

만약, 로컬에 있는 모든 변경 내용과 확정본을 포기하려면,
아래 명령으로 원격 저장소의 최신 이력을 가져오고,
로컬 master 가지가 저 이력을 가리키도록 할 수 있어요.

```
git fetch origin
git reset --hard origin/master
```

유용한 힌트

git의 내장 GUI

```
gitk
```

콘솔에서 git output을 컬러로 출력하기

```
git config color.ui true
```

이력(log)에서 확정본 1개를 딱 한 줄로만 표시하기

```
git config format.pretty oneline
```

파일을 추가할 때 대화식으로 추가하기

```
git add -i
```