

*Be as proud of Sogang as Sogang is proud of you*

# 블록체인 솔리디티 기초



서강대학교  
SOGANG UNIVERSITY

- Git/GitHub
- GitHub에 Remix 스마트컨트랙트 파일 저장
- 솔리디티 기초
  - 변수
  - 함수
  - 데이터타입
  - 함수
  - 연산자

## ■ Git

- 컴퓨터 파일의 변경사항을 추적하고 여러 명의 사용자들 간에 해당 파일들의 작업을 조율하기 위한 분산 버전 관리 시스템(Distributed Version Control Systems)
- Linus Torvalds가 리눅스의 소스 코드를 관리하기 위해서 만듦

## ■ GitHub

- 깃 저장소 호스팅을 지원
- Git을 기반으로 협업을 할 수 있도록 도와주는 웹서비스
- 그래픽 유저 인터페이스(GUI)를 제공
- <https://github.com/microsoft/vscode>



## ■ Git

### ■ 기존의 버전 관리

그 사이에 뭐가 바뀌었는지  
**차이 (Diff)** 를 알 수 없다.



과제 1\_ 최종 \_2016\_02\_28.zip

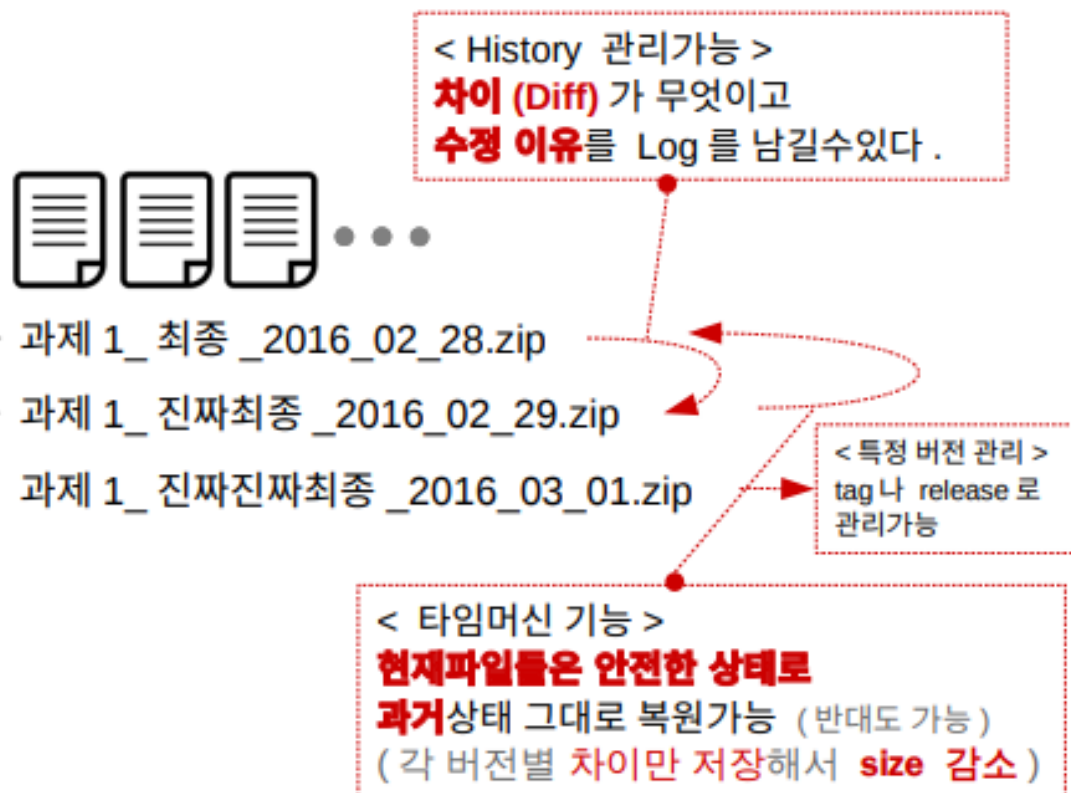
과제 1\_ 진짜최종 \_2016\_02\_29.zip

과제 1\_ 진짜진짜최종 \_2016\_03\_01.zip

Ctrl + c, v 를 할수록  
차지하는 **용량** X 2  
X 3 ... + diff

## ■ Git

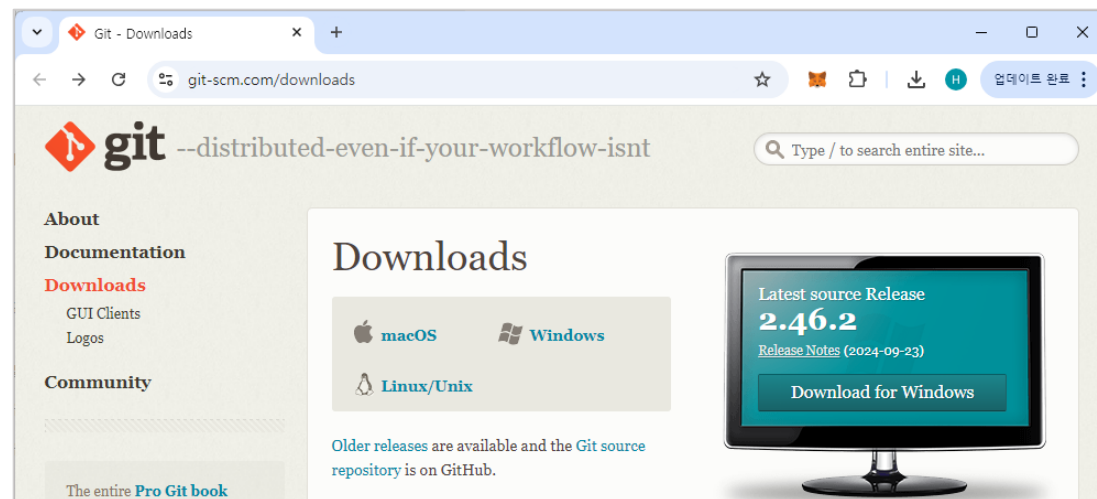
### ■ 버전 관리 도구



## ■ Git 실습

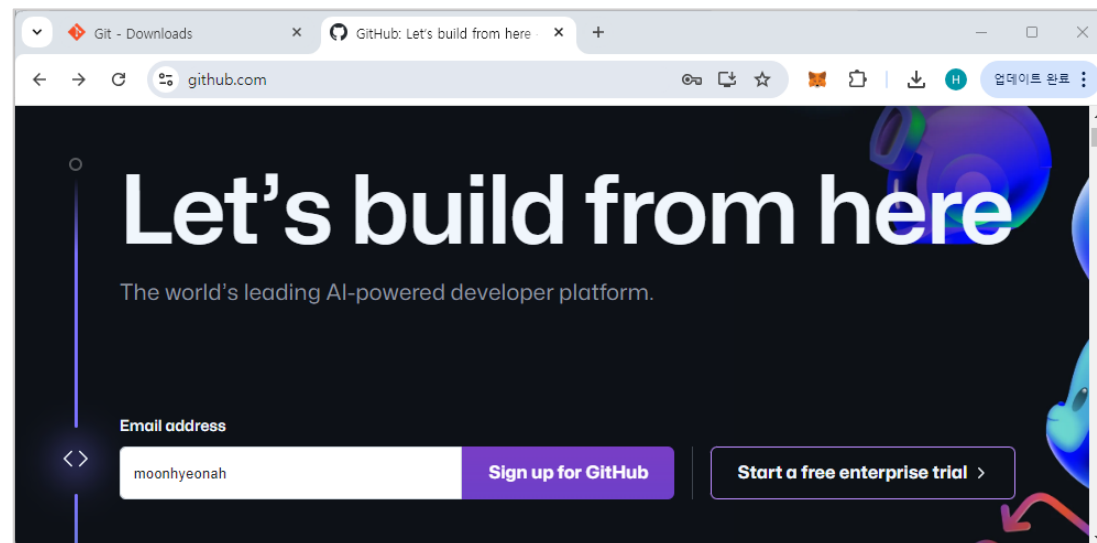
### ■ Git 설치

- <http://git-scm.com/downloads>

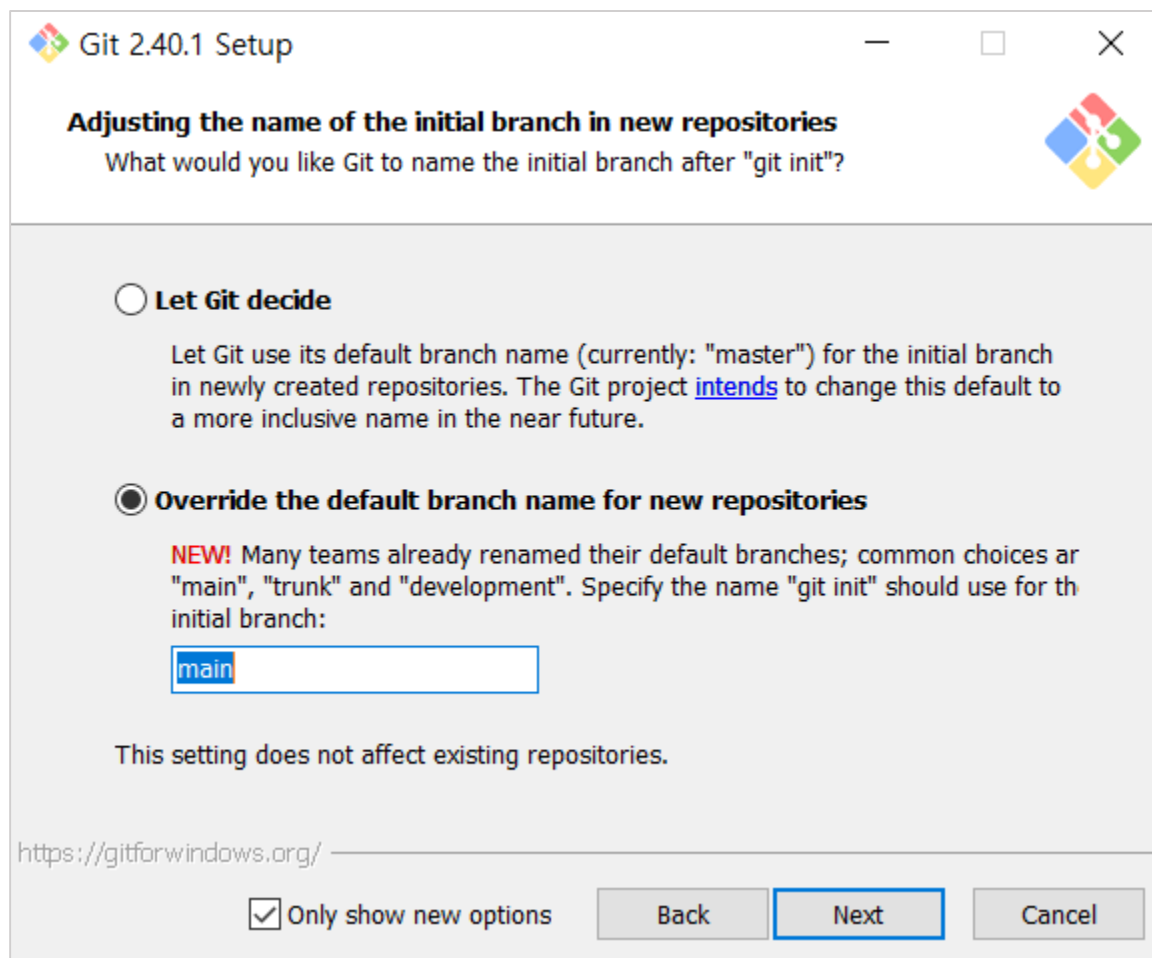


### ■ Github 회원가입

- <https://github.com/>



## ■ 기본 branch name 옵션



Git 2.40.1 Setup

**Adjusting the name of the initial branch in new repositories**

What would you like Git to name the initial branch after "git init"?

☐ **Let Git decide**

Let Git use its default branch name (currently: "master") for the initial branch in newly created repositories. The Git project [intends](#) to change this default to a more inclusive name in the near future.

☒ **Override the default branch name for new repositories**

**NEW!** Many teams already renamed their default branches; common choices are "main", "trunk" and "development". Specify the name "git init" should use for the initial branch:

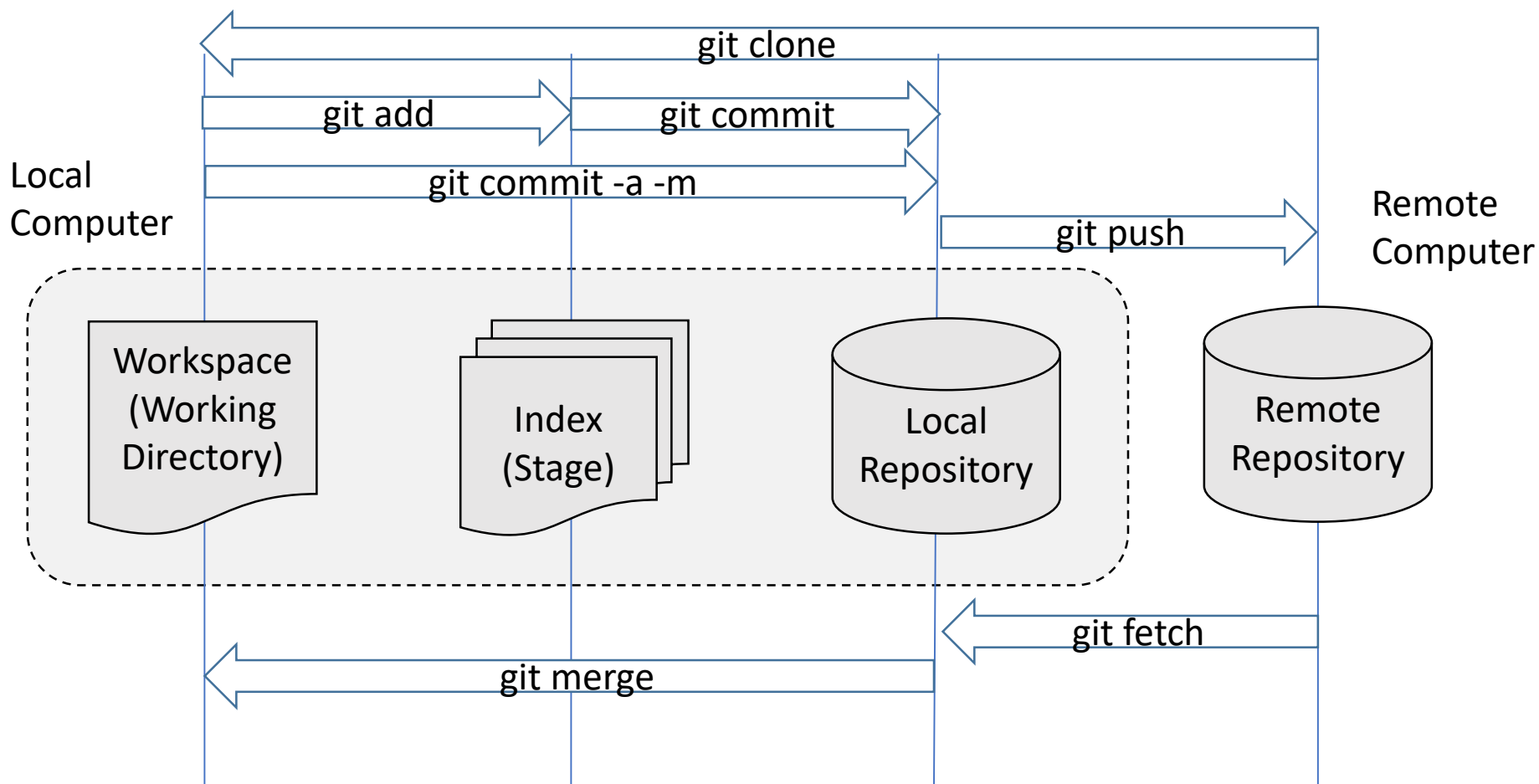
This setting does not affect existing repositories.

<https://gitforwindows.org/>

☒ Only show new options

Back Next Cancel

## ■ Git 실습





## ■ Git 실습

### ■ Git 명령(터미널 실행후) 사용자등록

- git config --global user.email "Github 계정 이메일"
- git config --global user.name "본인 영문 이름"
- git config --list

### ■ 필수 명령어

- add : 커밋할 목록에 추가
- commit : 커밋 ( 히스토리, 버전의 한단위 ) 만들기
- push: 현재까지 커밋을 Github 에 밀어넣기



## ■ Git 실습

### ■ 폴더 만들기

- mkdir git\_training
- cd git\_training

### ■ git init : 해당 폴더 git 초기화, 이 폴더를 git의 Local Repository로 만든다

### ■ example.sol 파일 생성

### ■ git status: 수시로 상태 확인

### ■ git add example.sol : Index(Stage) 목록에 추가

### ■ git commit -m "Add example.sol"

- m옵션은 커밋에 대한 설명

### ■ git log, git shortlog : 커밋 확인

```
// SPDX-License-Identifier: MIT  
pragma solidity ^0.8.7;  
  
contract Example {  
  
}
```

## ■ 명령창

```
C:\Work\git_training>git init
Initialized empty Git repository in C:/Work/git_training/.git/

C:\Work\git_training>git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    example.sol

nothing added to commit but untracked files present (use "git add" to track)

C:\Work\git_training>git add example.sol

C:\Work\git_training>git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   example.sol

C:\Work\git_training>git commit -m "Add example.sol"
[main (root-commit) bea7828] Add example.sol
 1 file changed, 6 insertions(+)
 create mode 100644 example.sol
```

## ■ Git 실습

- example.sol 파일 변경
- git status
- git diff
  - 작업디렉토리와 최신 커밋의 차이 확인

```
// SPDX-License-Identifier: MIT  
pragma solidity ^0.8.7;
```

```
contract Example {  
    //행 단위 주석  
}
```

```
C:\Work\git_training>git status  
On branch main  
Changes not staged for commit:  
  (use "git add <file>..." to update what will be committed)  
  (use "git restore <file>..." to discard changes in working directory)  
        modified:   example.sol  
  
no changes added to commit (use "git add" and/or "git commit -m")
```

```
C:\Work\git_training>git diff  
diff --git a/example.sol b/example.sol  
index cfbeb90..b338fe0 100644  
--- a/example.sol  
+++ b/example.sol  
@@ -2,5 +2,6 @@  
    pragma solidity ^0.8.7;  
  
    contract Example {  
+    //행 단위 주석  
    }
```

## ■ Git 실습

- git add example.sol
- git status
- git commit -m "Update example.sol"
- git log

```
C:\Work\git_training>git add example.sol
```

```
C:\Work\git_training>git status
```

```
On branch main
```

```
Changes to be committed:
```

```
(use "git restore --staged <file>..." to unstage)
```

```
    modified:   example.sol
```

```
C:\Work\git_training>git commit -m "Update example.sol"
```

```
[main 187c376] Update example.sol
```

```
1 file changed, 1 insertion(+)
```

```
C:\Work\git_training>git log
```

```
commit 187c3763c6a7eca5ab50887146c652b41a73d191 (HEAD ->
```

```
Author: Moon HyeonAh <moonhyeonah@gmail.com>
```

```
Date:   Fri Sep 27 16:28:14 2024 +0900
```

```
    Update example.sol
```

```
commit bea7828b16d1fe30e8c350010a601be040c68615
```

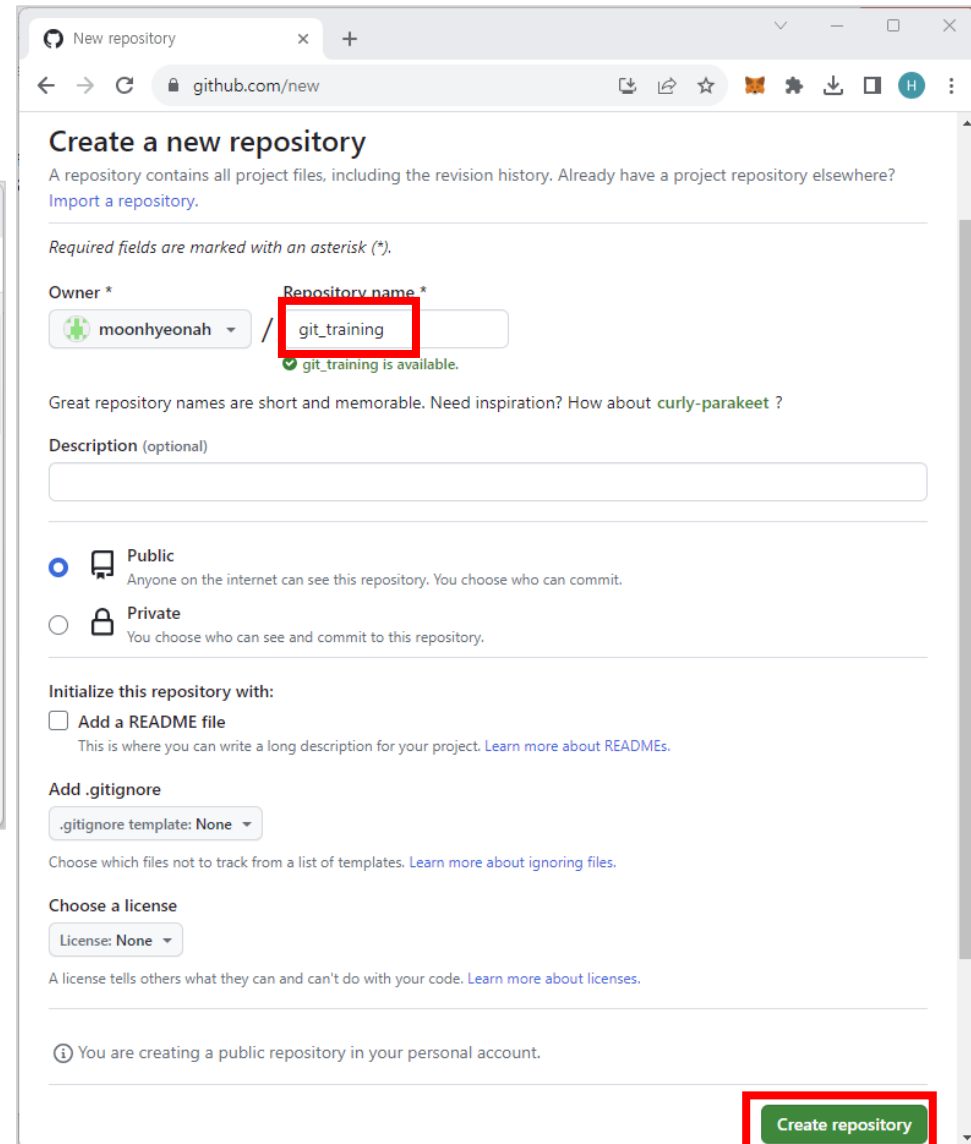
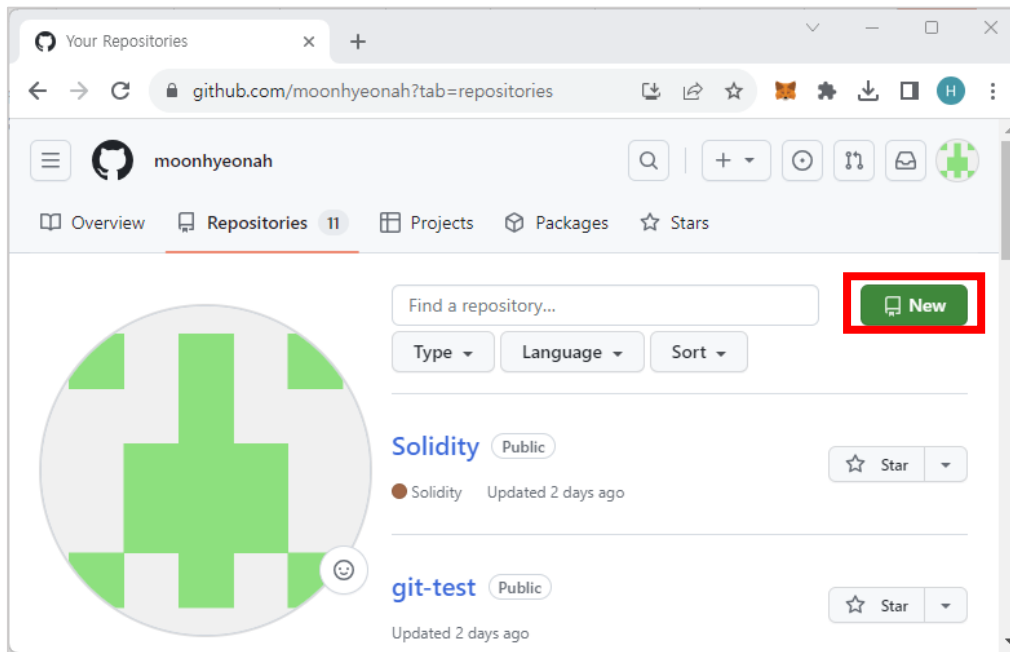
```
Author: Moon HyeonAh <moonhyeonah@gmail.com>
```

```
Date:   Fri Sep 27 16:16:45 2024 +0900
```

```
    Add example.sol
```

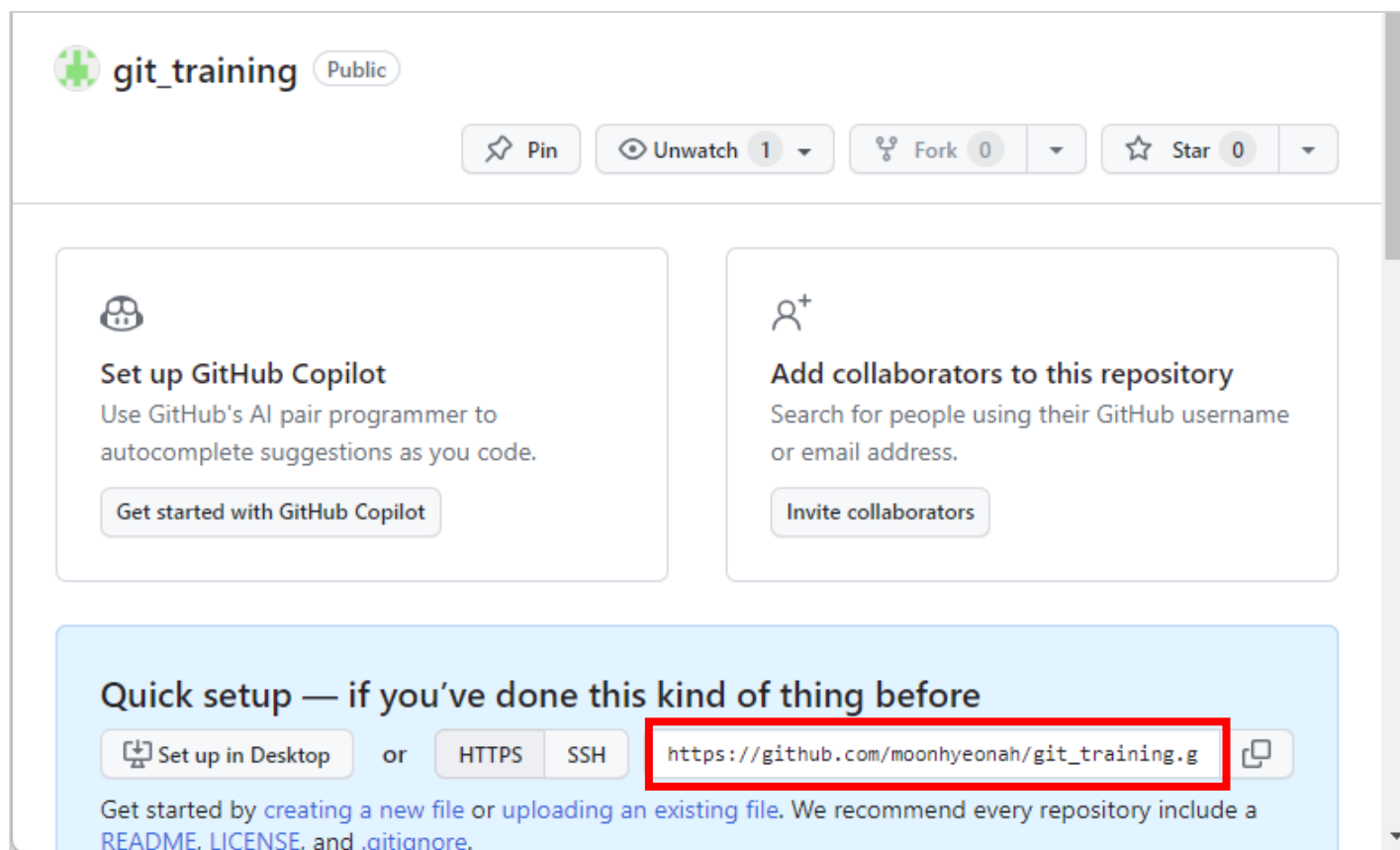
## ■ Git 실습

### ■ Github 에서 원격저장소 만들기



## ■ Git 실습

- Github 에서 원격저장소 만들기
  - 해당 URL 복사



## ■ Git 실습

### ■ `git remote add origin <복사한 URL>`

- 복사한 URL 로 Github 원격저장소 등록
- 현재 Local Repository의 Remote Repository를 지정
- `git remote` : Remote Repository 이름 확인
- `git remote get-url origin`
- `git branch` : Local Repository 이름 확인

### ■ `git push origin main`

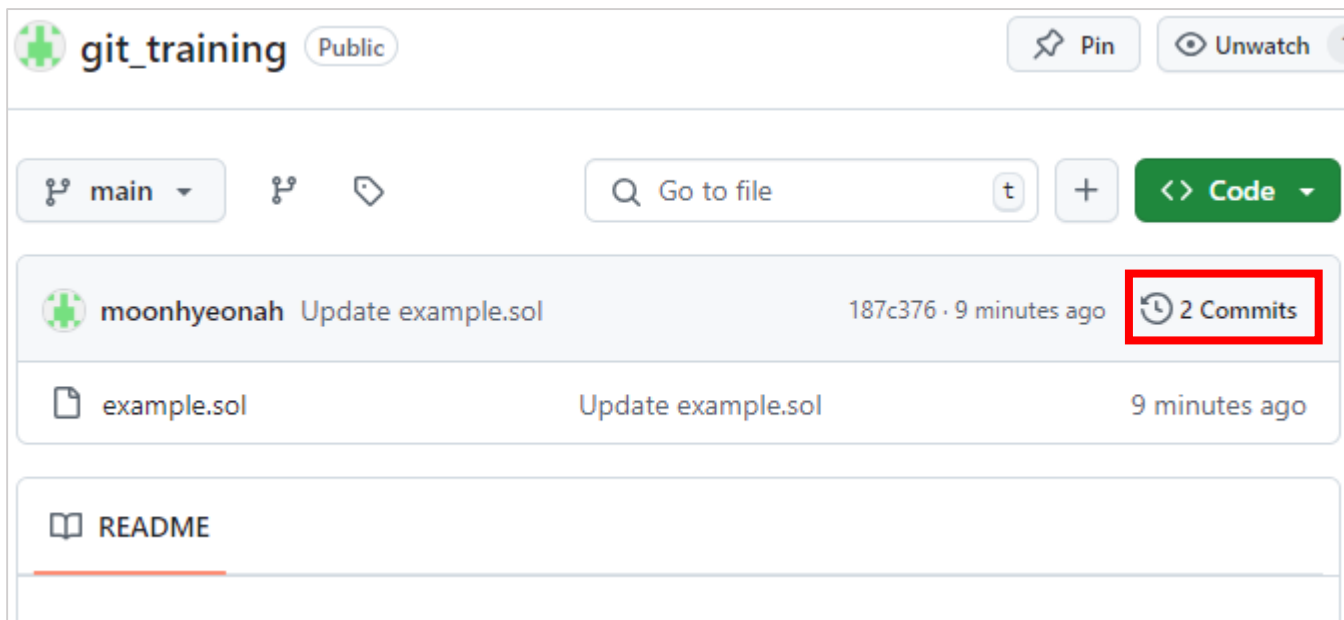
- Github 원격저장소(origin)에다 main(local branch) 밀어 넣기

### ■ Github 웹페이지 열고 확인



```
C:\Work\git_training>git remote add origin https://github.com/moonhyeonah/git_training.git

C:\Work\git_training>git push -u origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 16 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 559 bytes | 559.00 KiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/moonhyeonah/git_training.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```



The screenshot shows the GitHub interface for a repository named 'git\_training' which is public. At the top, there are buttons for 'Pin' and 'Unwatch'. Below this, a navigation bar includes a dropdown for the 'main' branch, a search bar labeled 'Go to file', and a green 'Code' button. The main content area displays a commit by 'moonhyeonah' titled 'Update example.sol', made '187c376 · 9 minutes ago'. A red box highlights the '2 Commits' link next to the commit. Below the commit, a file named 'example.sol' is listed with the same update message and time. At the bottom, a 'README' link is visible.

## ■ Git 실습

- example.sol 파일 변경
- git status
- git diff
- git add example.sol
- git status
- git commit -m "Update Block-level comments"
- git push origin main

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.7;

contract Example {
    //행 단위 주석
    /*
        블록 단위 주석
    */
}
```

### Commits

main	All users	All time
Commits on Sep 27, 2024		
Update Block-level comments		
moonhyeonah committed 1 minute ago	24117b3	<a href="#">📄</a> <a href="#">&lt;&gt;</a>
Update example.sol		
moonhyeonah committed 54 minutes ago	187c376	<a href="#">📄</a> <a href="#">&lt;&gt;</a>
Add example.sol		
moonhyeonah committed 1 hour ago	bea7828	<a href="#">📄</a> <a href="#">&lt;&gt;</a>

## ■ Git 실습

### ■ 커밋 수정

- 파일 수정
- git diff
- git add example.sol
- `git commit --amend` -m "Add new comment"
  - 마지막 커밋을 수정

```
// SPDX-License-Identifier: MIT  
pragma solidity ^0.8.7;
```

```
contract Example {  
    //행 단위 주석  
    /*  
        블록 단위 주석  
    */  
    //행 단위 주석 추가  
}
```

```
C:\Work\git_training>git shortlog  
Moon HyeonAh (3):  
    Add example.sol  
    Update example.sol  
    Update Block-level comments
```

```
C:\Work\git_training>git commit --amend -m "Add new comment"  
[main 89a7896] Add new comment  
Date: Fri Sep 27 17:22:07 2024 +0900  
1 file changed, 4 insertions(+), 1 deletion(-)
```

```
C:\Work\git_training>git shortlog  
Moon HyeonAh (3):  
    Add example.sol  
    Update example.sol  
    Add new comment
```

## ■ Git 실습

### ■ 커밋 수정

- git push origin main
  - 충돌 발생
- git push origin main --force
  - 강제 push
- Github 웹페이지 열고 확인

```
C:\Work\git_training>git push origin main
To https://github.com/moonhyeonah/git_training.git
 ! [rejected]          main -> main (non-fast-forward)
error: failed to push some refs to 'https://github.com/moonhyeonah/git_training.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. If you want to integrate the remote changes,
hint: use 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

C:\Work\git_training>git push origin main --force
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 387 bytes | 387.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/moonhyeonah/git_training.git
+ 24117b3...89a7896 main -> main (forced update)
```

## ■ Git 실습

### ■ add 취소

- 빈파일(test.txt) 생성
- git status
- git add test.txt
- git status
- [git reset](#)
- git status

```
C:\Work\git_training>git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    test.txt

nothing added to commit but untracked files present (use "git add" to track)

C:\Work\git_training>git add test.txt

C:\Work\git_training>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   test.txt

C:\Work\git_training>git reset

C:\Work\git_training>git status
On branch main
Your branch is up to date with 'origin/main'.

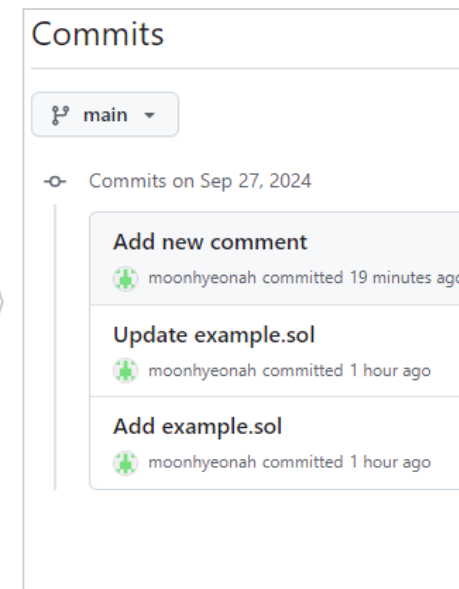
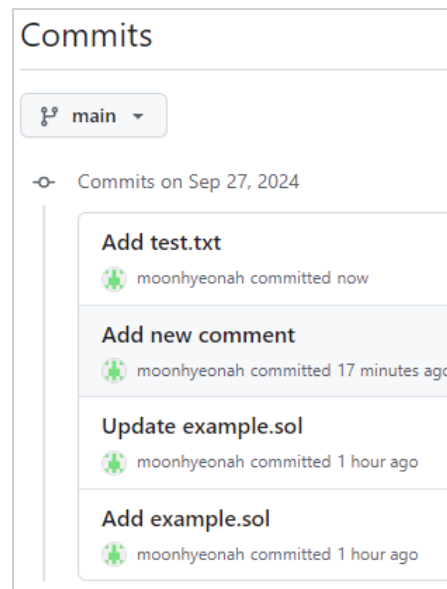
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    test.txt

nothing added to commit but untracked files present (use "git add" to track)
```

## ■ Git 실습

### ■ commit 한거 없애기

- 실수의 commit 만들기
- `git add test.txt`
- `git commit -m "Add test.txt"`
- `git push origin main`
- Github 웹페이지 확인
- `git shortlog`
- `git reset HEAD~1`
  - 가장 최근 commit 지우기
- `git shortlog`
- `git push origin main --force`



```
C:\Work\git_training>git shortlog
Moon HyeonAh (4):
    Add example.sol
    Update example.sol
    Add new comment
    Add test.txt
```

```
C:\Work\git_training>git reset HEAD~1
```

```
C:\Work\git_training>git shortlog
Moon HyeonAh (3):
    Add example.sol
    Update example.sol
    Add new comment
```

## ■ Git 실습

### ■ clone

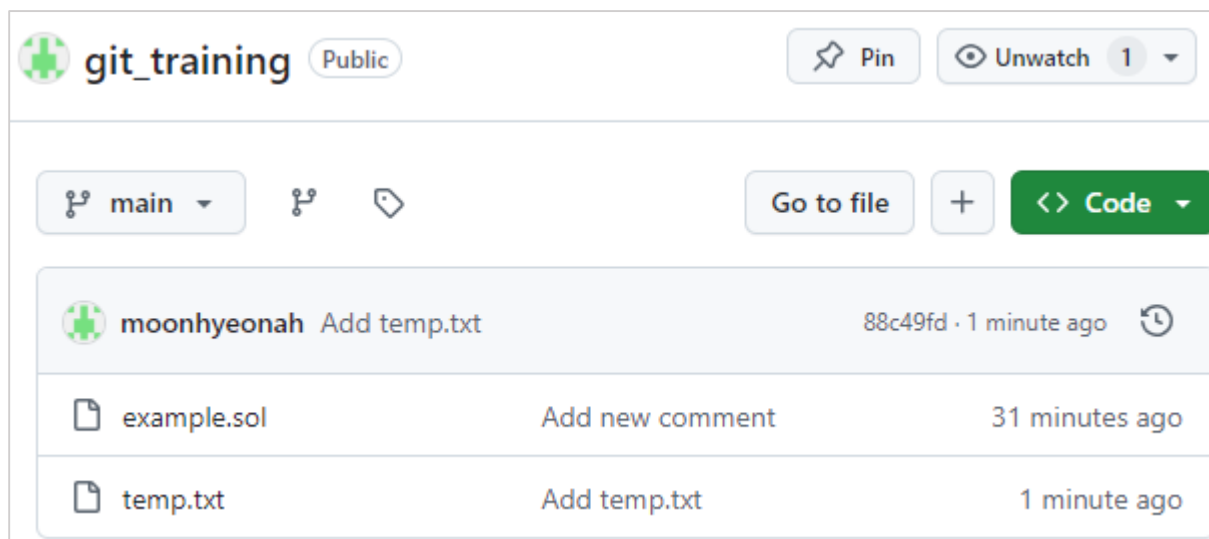
- 작업하던 폴더를 나와서 새로운 폴더 생성 `mkdir temp`
- `git clone` <Remote Repository>
  - Remote Repository 복사

```
C:\Work\git_training>cd ..  
  
C:\Work>mkdir temp  
  
C:\Work>cd temp  
  
C:\Work\temp>git clone https://github.com/moonhyeonah/git_training.git  
Cloning into 'git_training'...  
remote: Enumerating objects: 9, done.  
remote: Counting objects: 100% (9/9), done.  
remote: Compressing objects: 100% (5/5), done.  
remote: Total 9 (delta 2), reused 8 (delta 1), pack-reused 0 (from 0)  
Receiving objects: 100% (9/9), done.  
Resolving deltas: 100% (2/2), done.  
  
C:\Work\temp>cd git_training
```

## ■ Git 실습

### ■ clone

- 복사한 새로운 Local Repository에서 빈 파일 생성 temp.txt
- git status
- git add temp.txt
- git commit -m "Add temp.txt"
- git push origin main

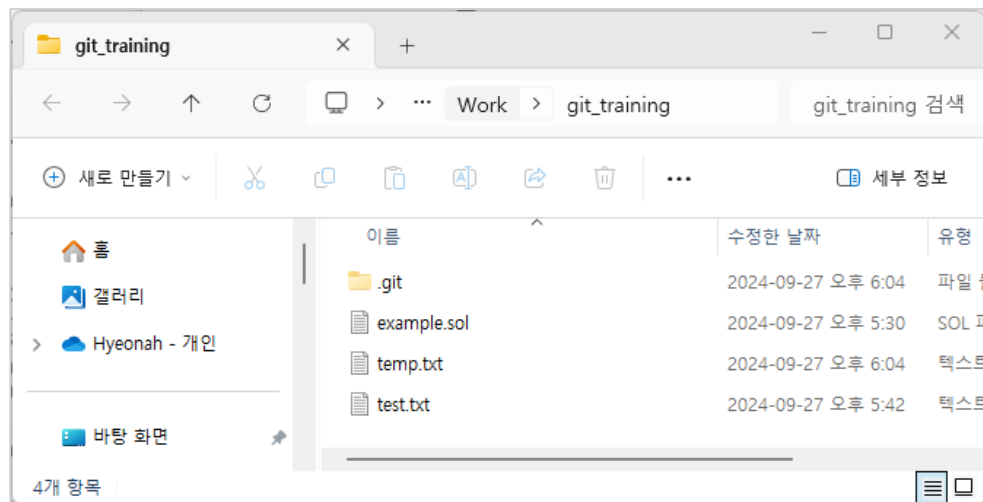




## ■ Git 실습

### ■ fetch/merge

- 원래 처음 작업하던 폴더로 이동
- `git fetch`
  - 이전 슬라이드에서 push 한 결과 fetch
- `git merge origin/main`
  - 원래 작업한 로컬과 merge



```
C:\Work\temp\git_training>cd ..
```

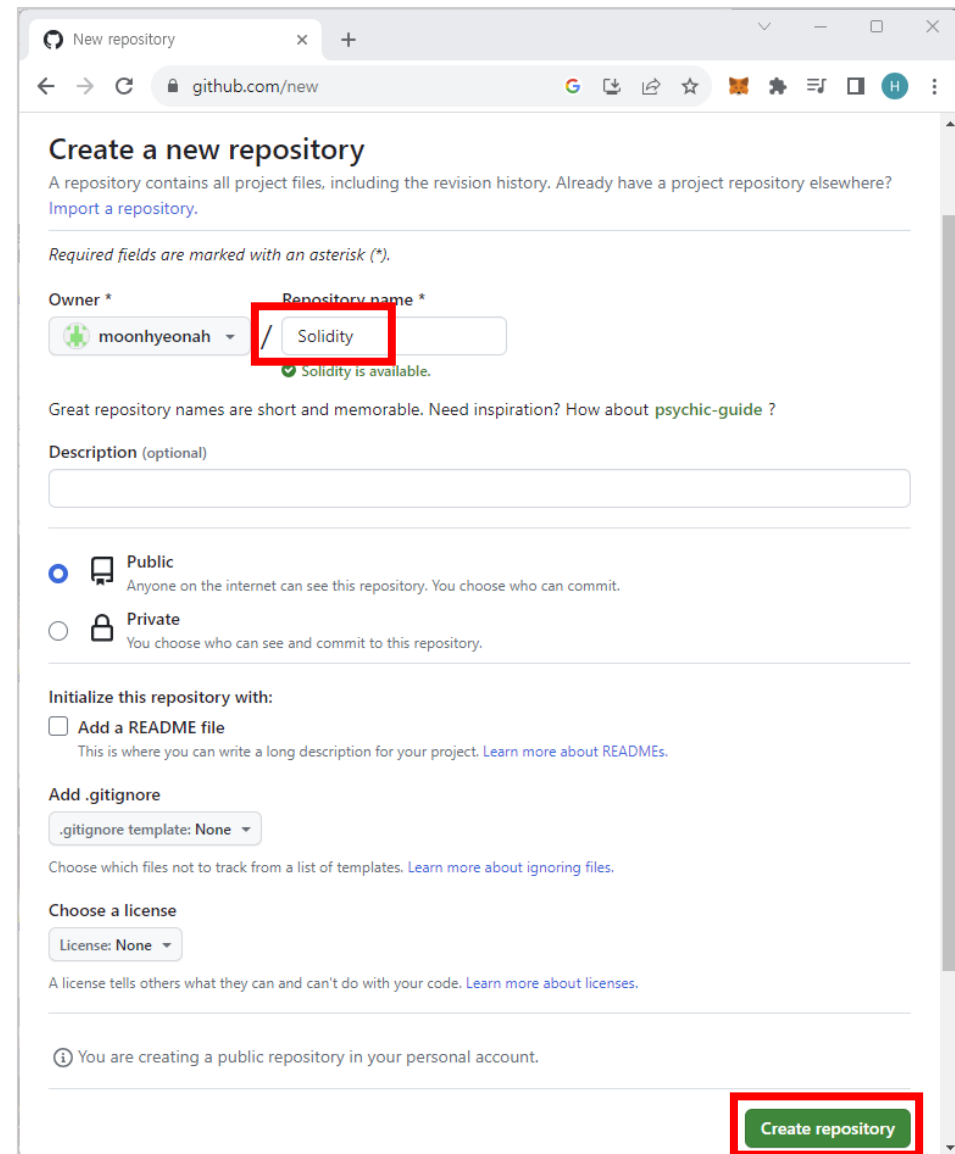
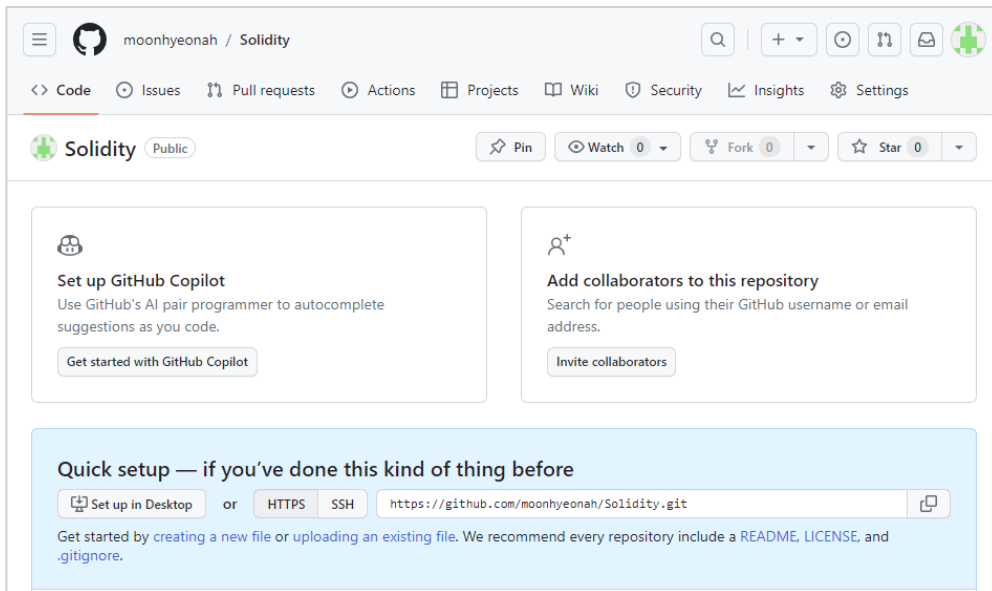
```
C:\Work\temp>cd ..
```

```
C:\Work>cd git_training
```

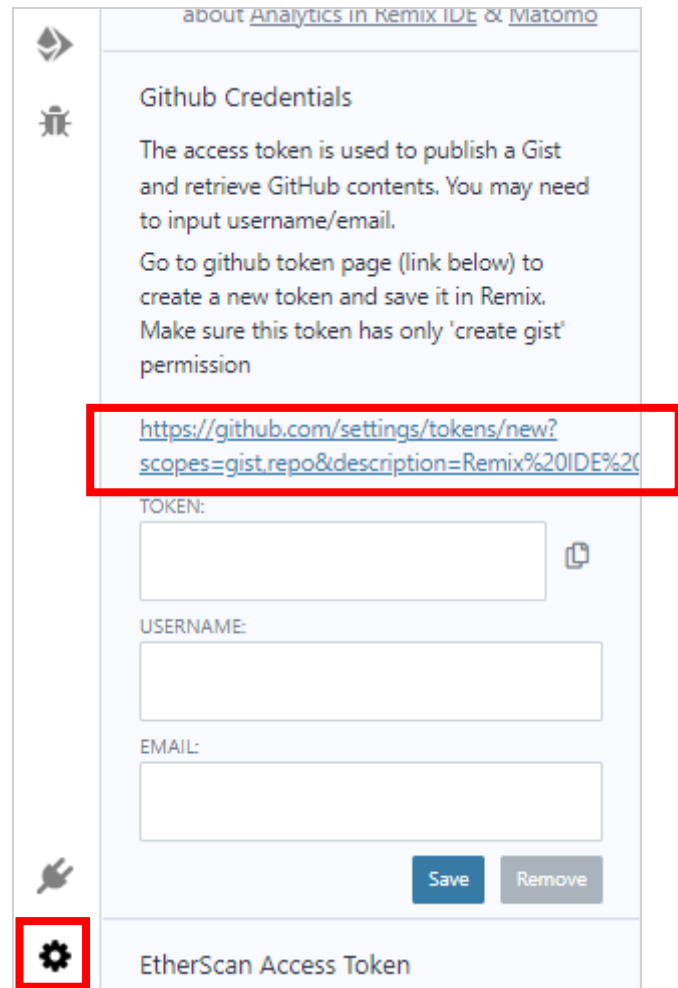
```
C:\Work\git_training>git fetch
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-r
Unpacking objects: 100% (3/3), 262 bytes | 16.00 KiB/
From https://github.com/moonhyeonah/git_training
   89a7896..88c49fd  main       -> origin/main
```

```
C:\Work\git_training>git merge origin/main
Updating 89a7896..88c49fd
Fast-forward
 temp.txt | 0
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 temp.txt
```

- GitHub에 Repository 생성
- Repository 생성



- Remix Settings → access token 만들기로 이동



about [Analytics in Remix IDE & Matomo](#)

### Github Credentials

The access token is used to publish a Gist and retrieve GitHub contents. You may need to input username/email.


Go to github token page (link below) to create a new token and save it in Remix. Make sure this token has only 'create gist' permission

<https://github.com/settings/tokens/new?scopes=gist,repo&description=Remix%20IDE%20>

TOKEN:

USERNAME:

EMAIL:

 EtherScan Access Token

## ■ GitHub의 Generate token 버튼 클릭해서 토큰 생성

GitHub Apps

OAuth Apps

Personal access tokens Beta

Fine-grained tokens

Tokens (classic)

### New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

**Note**

Remix IDE Token

What's this token for?

**Expiration \***

90 days

 The token will expire on Thu, Dec 21 2023

**Select scopes**

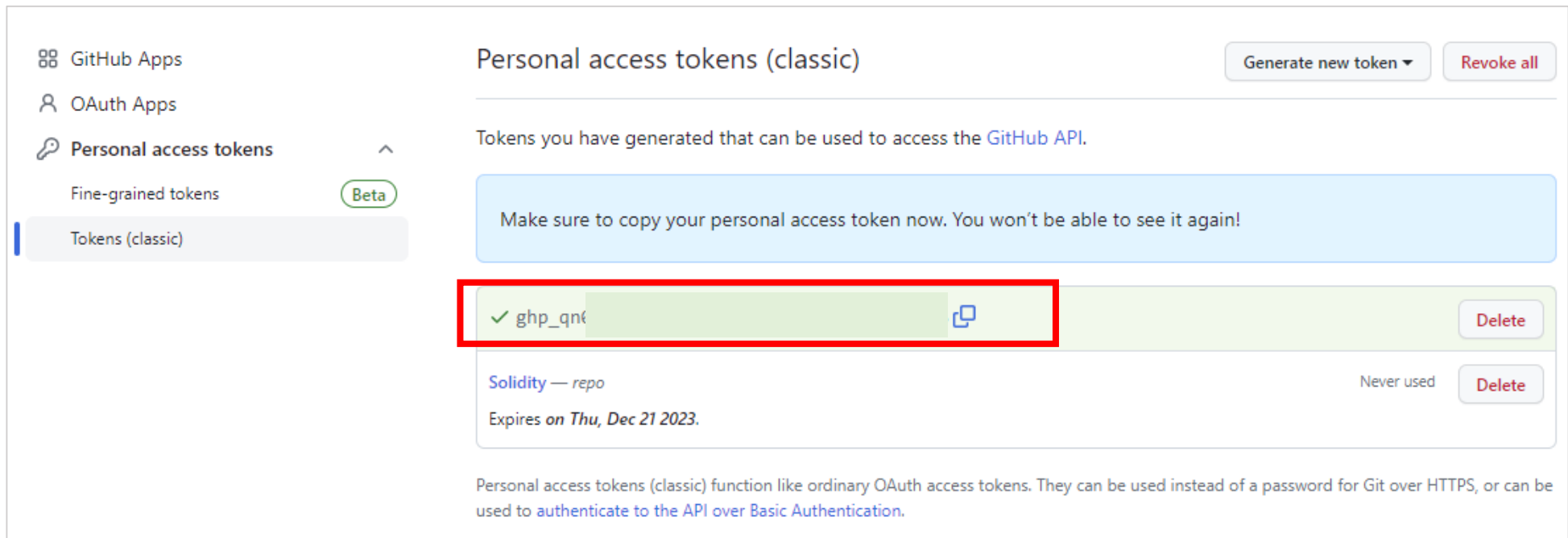
Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input type="checkbox"/> write:gpg_key	Write public user GPG keys
<input type="checkbox"/> read:gpg_key	Read public user GPG keys
<input type="checkbox"/> admin:ssh_signing_key	Full control of public user SSH signing keys
<input type="checkbox"/> write:ssh_signing_key	Write public user SSH signing keys
<input type="checkbox"/> read:ssh_signing_key	Read public user SSH signing keys

Generate token

Cancel

## ■ 생성한 access token 복사




The screenshot shows the GitHub 'Personal access tokens (classic)' interface. On the left, a sidebar contains links for 'GitHub Apps', 'OAuth Apps', 'Personal access tokens' (selected), 'Fine-grained tokens' (marked Beta), and 'Tokens (classic)'. The main area is titled 'Personal access tokens (classic)' and includes buttons for 'Generate new token' and 'Revoke all'. A blue warning box states: 'Make sure to copy your personal access token now. You won't be able to see it again!'. Below this, a table lists generated tokens. The first token, 'ghp\_qn...', is highlighted with a red box and has a copy icon. The second token, 'Solidity — repo', is marked 'Never used'. A footer note explains that these tokens function like OAuth access tokens for Git over HTTPS or API authentication.

Personal access tokens (classic) Generate new token ▼ Revoke all

Tokens you have generated that can be used to access the [GitHub API](#).

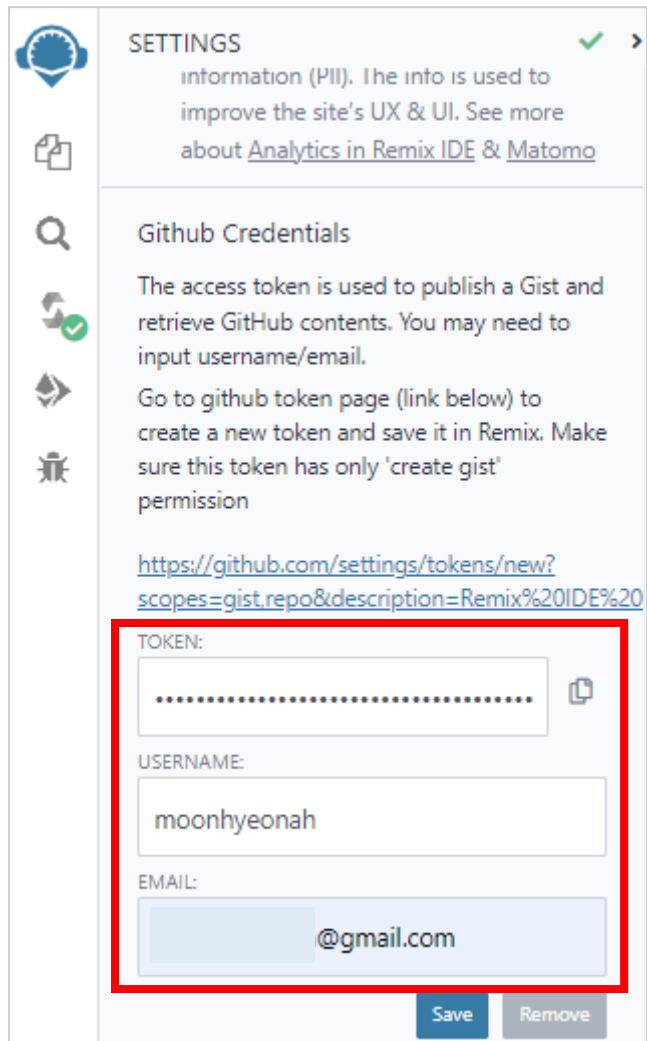
Make sure to copy your personal access token now. You won't be able to see it again!

✓ ghp_qn...		Delete
Solidity — repo	Never used	Delete

Expires on Thu, Dec 21 2023.

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

## ■ Remix Settings에 생성한 access token 저장



**SETTINGS** ✓ >

information (PII). The info is used to improve the site's UX & UI. See more about [Analytics in Remix IDE](#) & [Matomo](#)


**Github Credentials**

The access token is used to publish a Gist and retrieve GitHub contents. You may need to input username/email.

Go to github token page (link below) to create a new token and save it in Remix. Make sure this token has only 'create gist' permission

<https://github.com/settings/tokens/new?scopes=gist.repo&description=Remix%20IDE%20>

**TOKEN:**

..... 

**USERNAME:**

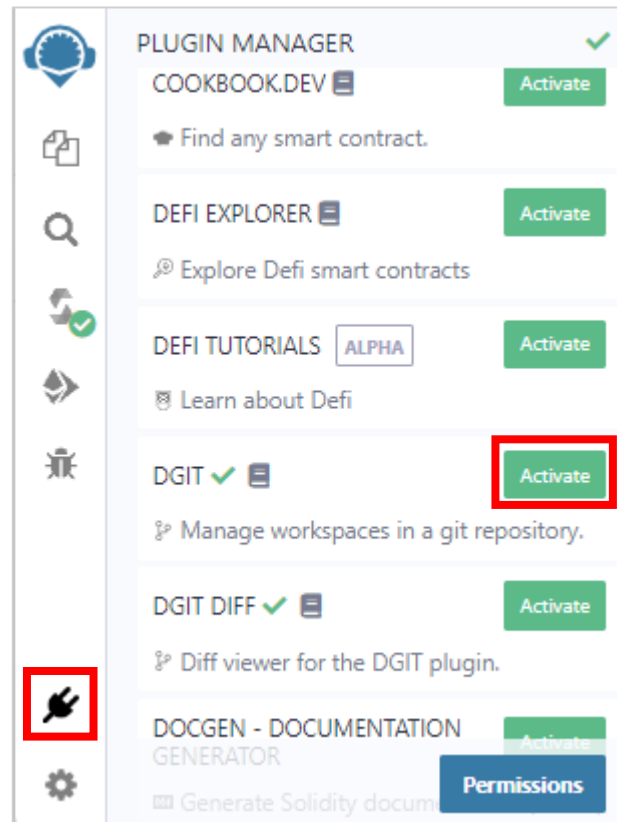
moonhyeonah

**EMAIL:**

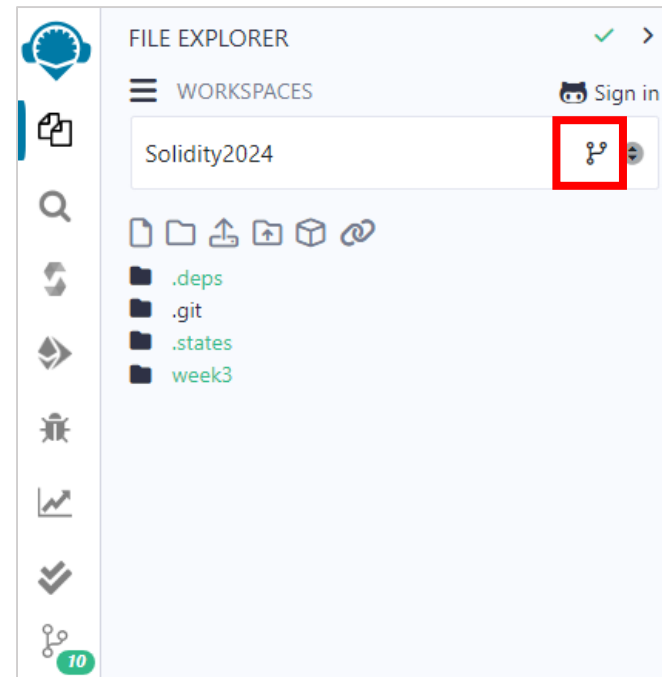
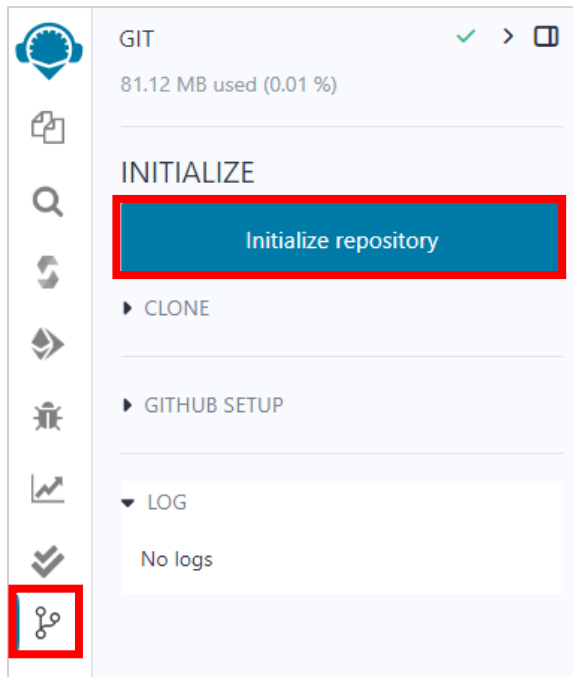
.....@gmail.com

**Save** **Remove**

## ■ 플러그인 DGIT 활성화

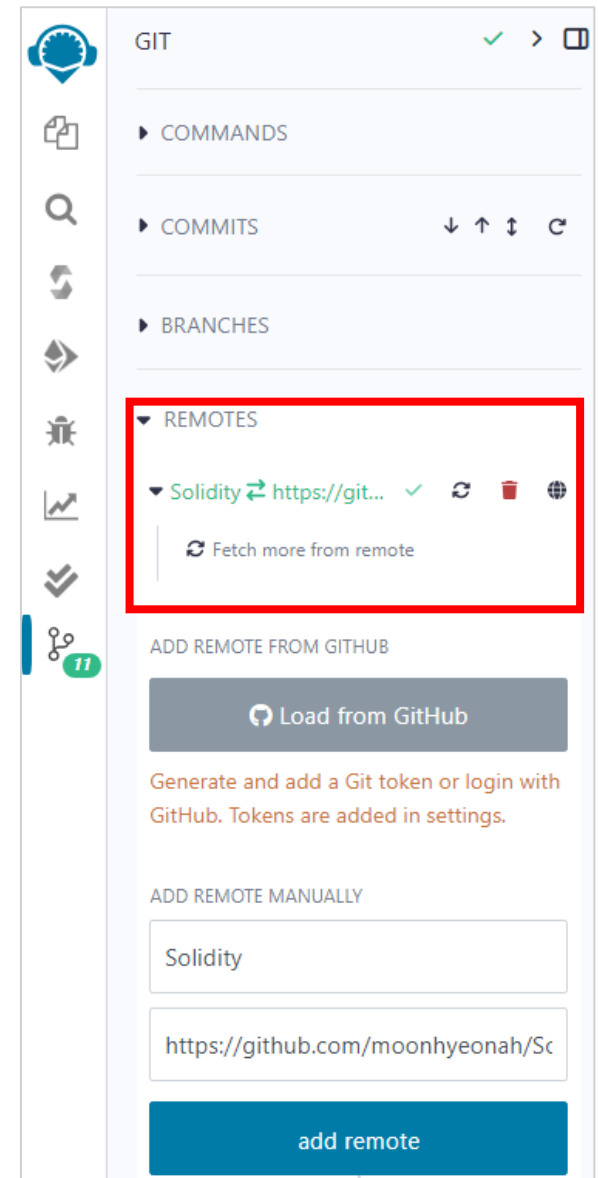
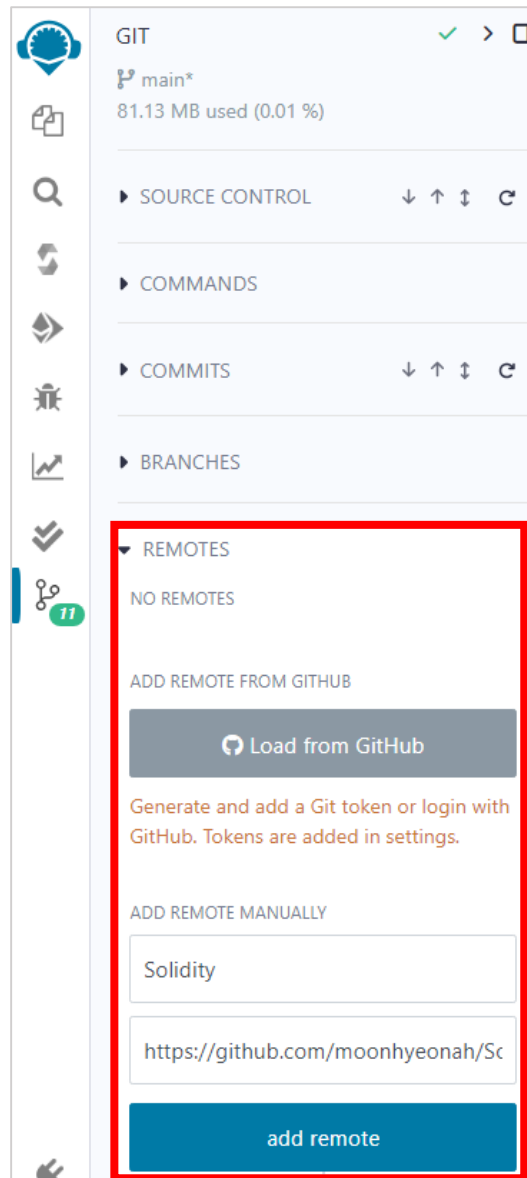


- GitHub 리포지토리 설정
  - git init: Initialize repository

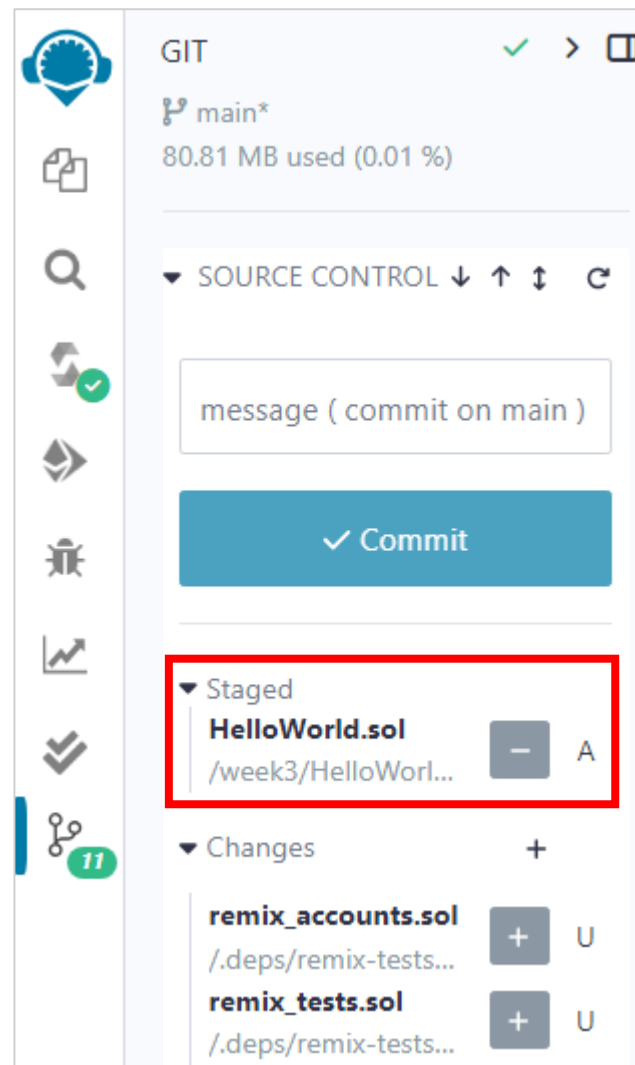
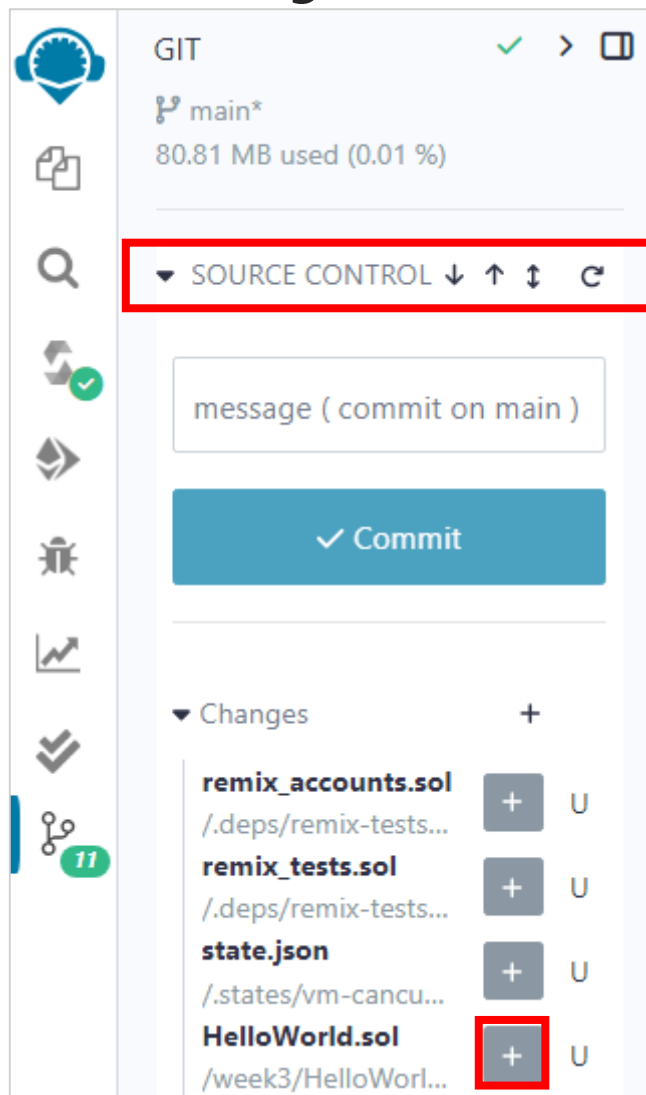




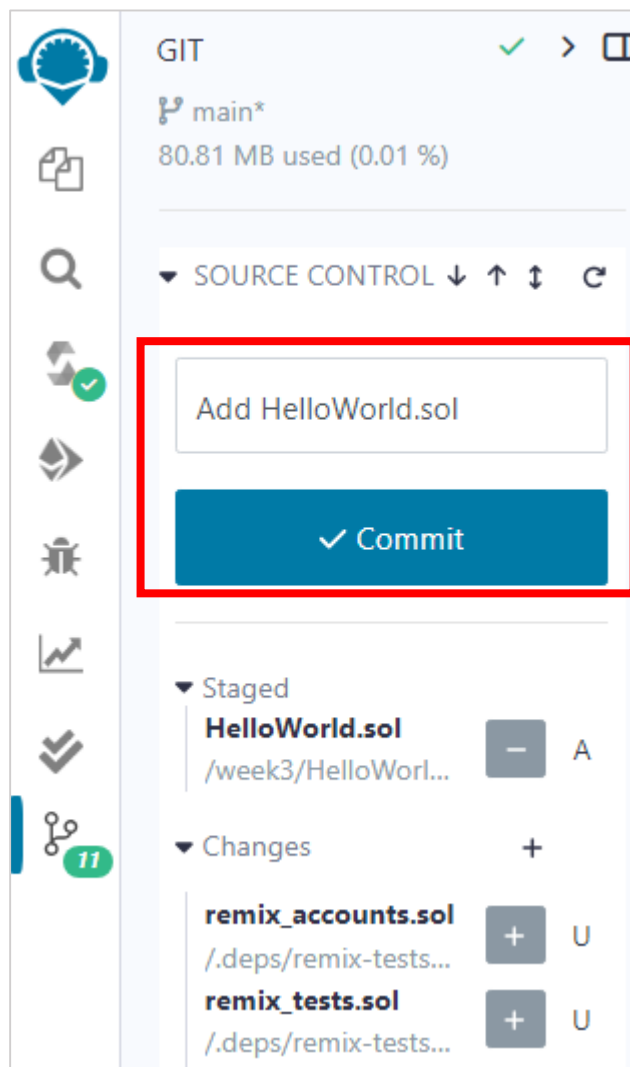
- GitHub 리포지토리 설정
  - GIT REMOTE 설정



## ■ Remix에서 git add

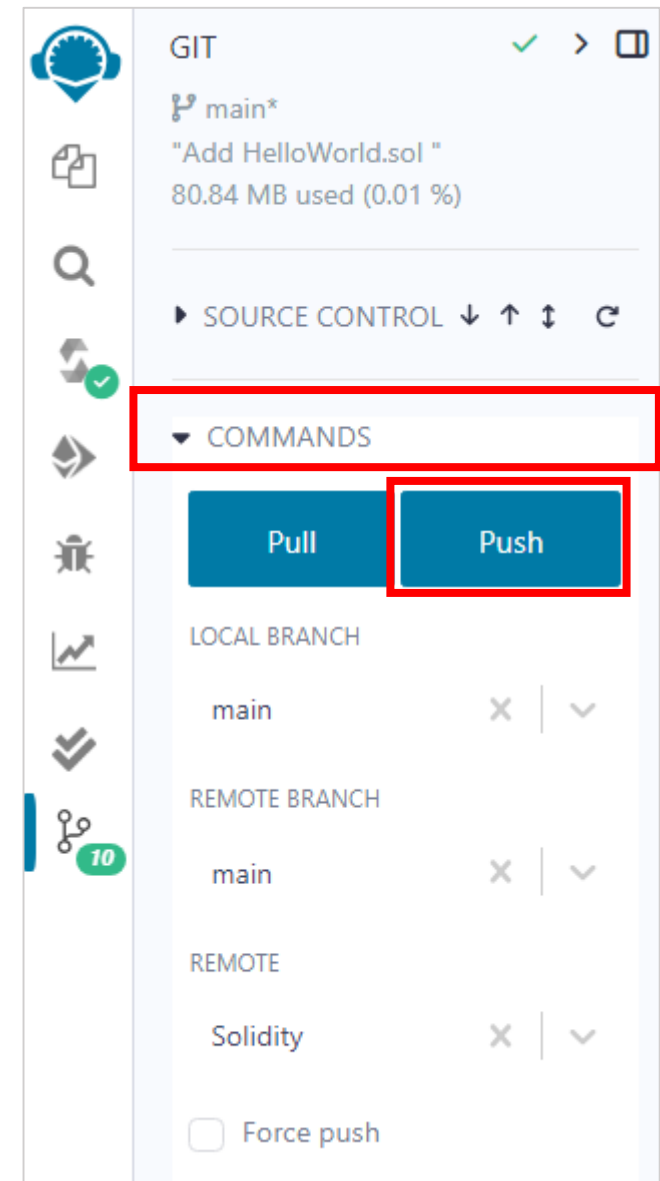
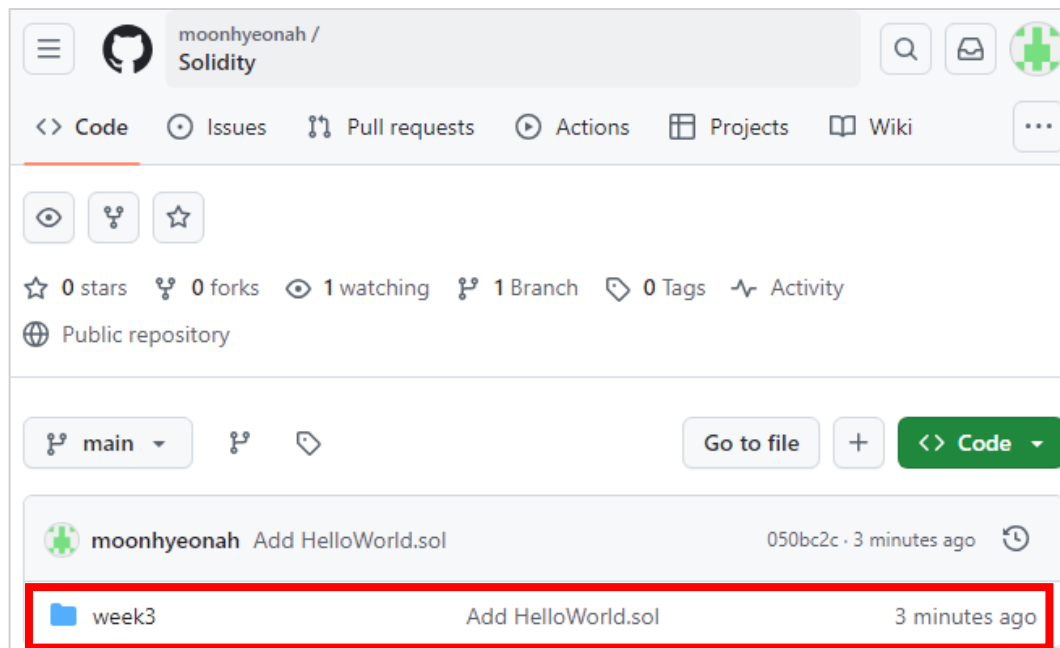


- Remix에서 git commit
  - 메시지 추가



- Remix에서 git push

- GitHub에서 확인



- **비트코인**

- 결제시스템, 암호화폐의 거래 내역 저장

- **이더리움: 스마트 컨트랙트를 도입해 다양한 분야로 블록체인 확장**

- **스마트 컨트랙트**

- 이더리움의 핵심 기능
- 블록체인 기술을 실용적으로 여러 분야에 적용할 수 있도록 만들어준 전환점
- 스마트컨트랙트 도입에 의해 블록체인1.0과 블록체인2.0으로 구분
- 주로 솔리디티 언어로 작성

## ■ 특징

- 서면계약서가 코드로 구현된 블록체인에서 작동하는 디지털화된 계약서
- 보안성
  - 블록체인에 저장: 조작, 분실이나 해킹 위험 감소
- 신뢰성
  - 중개인 필요성 해소
  - 당사자간의 신뢰성을 높여줌: 스마트 컨트랙트가 자동으로 계약 이행
- 중개인에 대한 수수료 절감
- 코드에 오류가 있을 수 있음

## ■ Solidity

- 스마트 컨트랙트 개발을 위한 프로그래밍 언어
- 객체 지향
- C++, 자바스크립트, 파이썬의 영향
- 가장 많이 사용되는 언어
- 상속 및 라이브러리 사용 가능
- 이더리움 가상 머신(Ethereum Virtual Machine,EVM)에서 실행 가능한 이더리움 바이트코드로 컴파일됨
- sol: Solidity 파일의 확장자

## ■ 스마트 컨트랙트 기본 구조

### ■ SPDX 라이선스

- 스마트컨트랙트에 대한 신뢰를 높이고,
- 저작권 문제 해소 위해 코드 최상단에 작성
- <https://spdx.org/licenses/>

### ■ 솔리디티 컴파일러 버전 정보

### ■ 컨트랙트 선언과 정의

- contract 컨트랙트이름

### ■ 주석: 프로그램의 함수, 변수 등에 대한 설명

- 블록단위
- 행단위

```
// SPDX-License-Identifier: MIT  
Pragma solidity ^0.8.7;
```

```
contract Ex2_1 {
```

```
    //행 단위 주석
```

```
    /*
```

```
        블록 단위 주석
```

```
    */
```

```
}
```



## ■ 변수와 자료형

- 변수: 특정 유형의 값을 담는 그릇(공간), 변하는 값 저장
  - 자료형 명시해야 함

uint	a	=	5
자료형	변수명	대입연산자	값

## ■ 자료형

- 값 타입(value type): 해당 값 자체가 복사
  - bool, int, uint, address
- 참조 타입(reference type): 해당하는 값의 주소만 복사
  - 배열, 매핑, 구조체, 문자열

## ■ 변수와 자료형

### ■ boolean: 참/거짓(true/false)

- 주로 조건문에 사용
- 연산자
  - ! (논리 부정)
  - && (논리 AND, "and")
  - || (논리 OR, "or")
  - == (같음)
  - != (같지 않음)

### ■ 정수: 부호있는 정수 타입(int), 부호없는 정수 타입(uint)

- uint8 에서 uint256 까지, int8 부터 int256 까지 8비트 단위로 존재 → 메모리 공간의 효율적인 사용
- uint 와 int 는 각각 uint256 와 int256 의 별칭

## ■ 변수와 자료형

### ■ 문자열(string)

- 동적 크기 UTF-8(Unicode Transformation Format-8)로 인코딩된 배열

### ■ 주소형(address)

- 계정의 주소를 나타내는 20바이트 자료형
- 유저의 고유 아이디 또는 배포된 스마트 컨트랙트의 아이디

### ■ 바이트 타입(bytes)

- 고정 크기 바이트 배열은 값 타입이며 사용할 바이트를 미리 지정
  - 예) bytes2: 2바이트 크기의 값을 저장할 수 있는 자료형
  - bytes2 b = 0x1020;
- bytes: 정해지지 않은 크기의 바이트 값을 저장

## ■ 변수 유형

### ■ 상태 변수(state)

- 함수 밖에서 선언한 변수
- 블록체인에 영구히 기록

### ■ 지역 변수(local)

- 함수 안에서 선언
- 블록체인에 기록되지 않는다

### ■ 전역 변수(global)

- 블록체인에 관한 정보, 블록이나 트랜잭션 메시지 정보 등

## ■ 가시성 지정자(Visibility): 상태변수와 함수에 적용

### ■ public

- 스마트 컨트랙트 내부 및 외부에서 모두 호출 가능
- public 변수는 자동으로 getter 함수 생성

### ■ private

- 오직 정의한 스마트 컨트랙트 내부에서만 호출 가능
- 상속에서도 private 사용 불가능

### ■ internal

- 스마트 컨트랙트 내부에서만 호출 가능
- 상속받은 컨트랙트에서 사용 가능

### ■ external: 함수에만 적용

- 스마트 컨트랙트 외부에서만 호출 가능

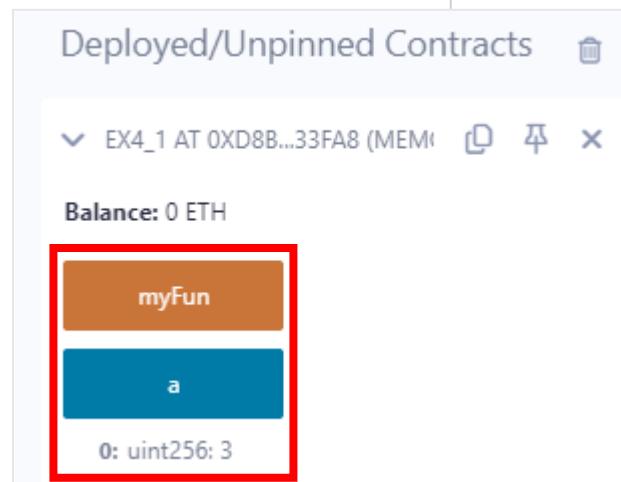
## ■ 함수

- 특별한 작업을 수행하도록 만들어진 코드의 집합체
- 유지보수, 재사용, 가독성

```
function 함수명 () public {    // (public, private, internal, external) 변경가능  
    // 함수 로직  
}
```

## ■ 간단한 함수 정의

```
// SPDX-License-Identifier: GPL-3.0  
pragma solidity >=0.7.0 <0.9.0;  
  
contract Ex4_1 {  
  
    uint public a = 3;  
    function myFun() public{  
        a = 5;  
    }  
  
}
```



## ■ 매개변수(parameter)가 있는 function 정의

```
function 함수명 (자료형 변수명) public {
    // 내용
}
```

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.0 <0.9.0;
```

```
contract Ex4_2 {

    uint public a = 3;
    function myFun(uint b, uint c, uint d) public{
        a = b;
        a = c;
        a = d;
    }

}
```

The image shows two screenshots of the Etherscan interface for a smart contract named 'EX4\_2 AT 0XDDA...5482D (MEMORY)'. The contract's balance is 0 ETH.

The top screenshot shows the 'myFun' function with three input fields for parameters 'b', 'c', and 'd', all set to 'uint256'. A red box highlights the 'myFun' function name. Below the input fields are buttons for 'Calldata', 'Parameters', and 'transact'. A blue button labeled 'a' is also visible.

The bottom screenshot shows the same function with specific values entered: 'b' is 7, 'c' is 8, and 'd' is 9. The 'transact' button is highlighted in orange. Below the input fields, the state change is shown: '0: uint256: 9'.

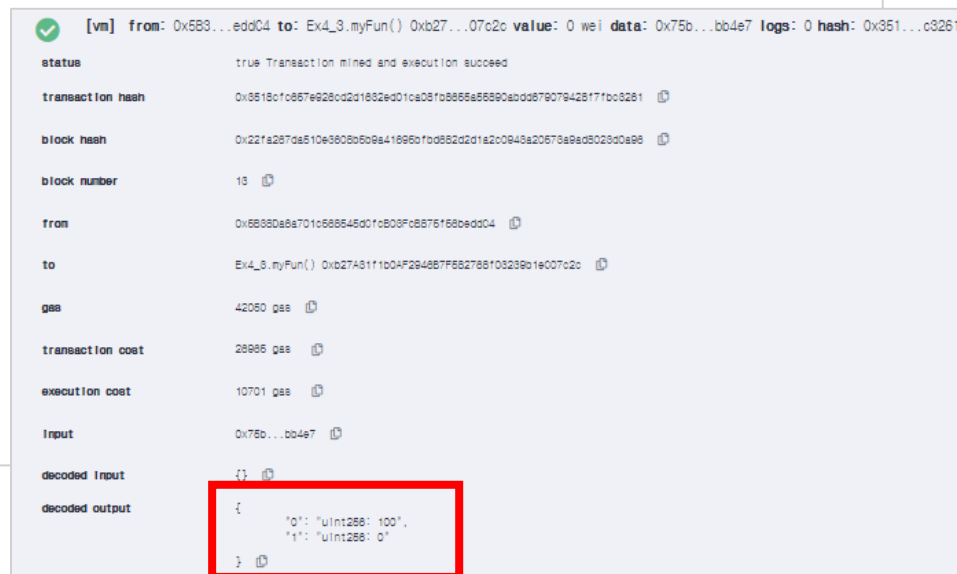
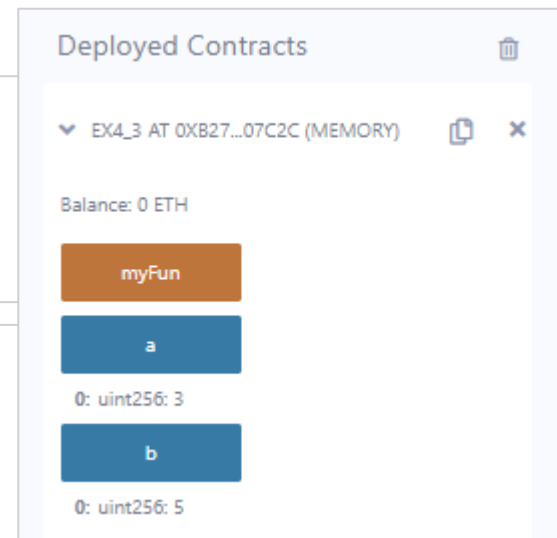
## ■ 반환(return) 값이 있는 function 정의

```
function 함수명 () public returns(반환하고자 하는 자료형) {
    // 내용
    return 반환하고자 하는 자료형 변수;
}
```

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.0 <0.9.0;
```

```
contract Ex4_3 {

    uint public a = 3;
    uint public b = 5;
    function myFun() public returns(uint, uint){
        a = 100;
        b = 0;
        return (a,b);
    }
}
```





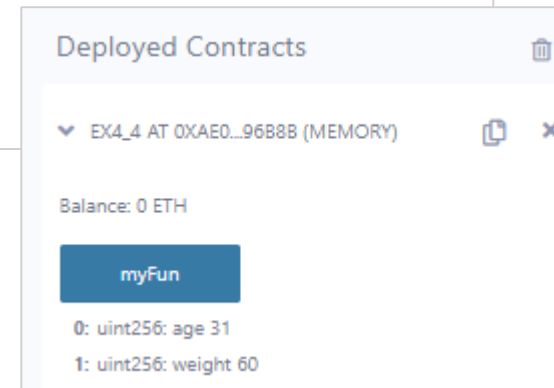
## ■ 반환(return) 값이 있는 function 정의

### ■ 반환자료형과 함께 변수명 선언

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.0 <0.9.0;

contract Ex4_4 {

    function myFun() public pure returns(uint age, uint weight) {
        age = 31;
        weight = 60;
    }
}
```



## ■ 반환(return) 값이 있는 function 정의

### ■ 함수의 반환값 활용

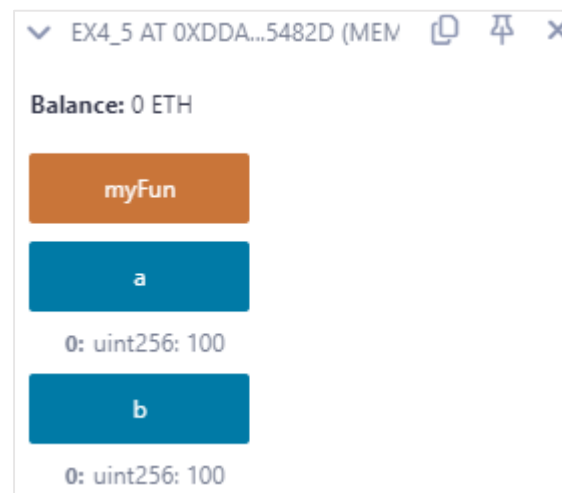
```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.0 <0.9.0;

contract Ex4_5 {

    uint public a = 3;
    uint public b = myFun();

    function myFun() public returns(uint){
        a = 100;
        return a;
    }

}
```

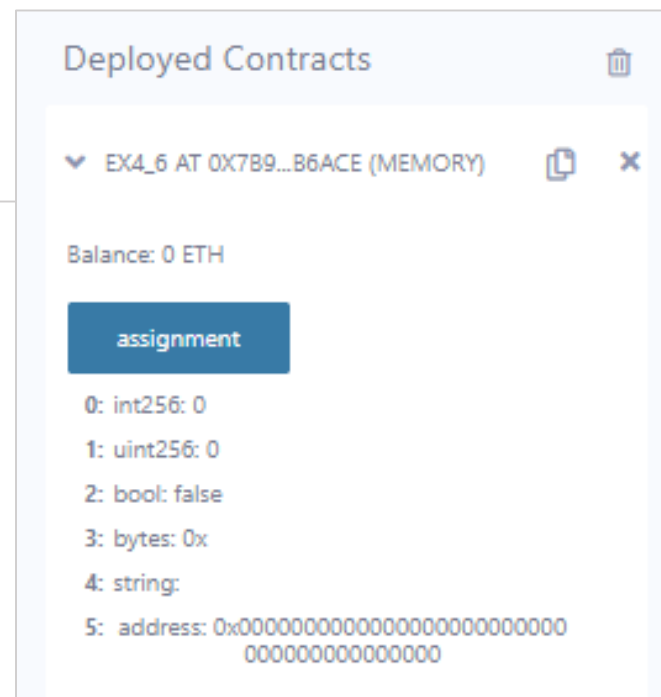


## ■ 자료형에 따른 각 변수의 기본값

```
// SPDX-License-Identifier: GPL-3.0  
pragma solidity >=0.7.0 < 0.9.0;
```

```
contract Ex4_6 {  
    int a;  
    uint b;  
    bool c;  
    bytes d;  
    string e;  
    address f;
```

```
    function assignment() public view returns(int, uint, bool, bytes memory, string memory, address) {  
        return(a, b, c, d, e, f);  
    }  
}
```



## ■ 산술 연산자

```
// SPDX-License-Identifier: GPL-3.0  
pragma solidity >=0.7.0 < 0.9.0;
```

```
contract Ex4_7 {
```

```
    uint a = 5+2;
```

```
    uint b = 5-2;
```

```
    uint c = 5*2;
```

```
    uint d = 5/5;    // 나눗셈
```

```
    uint e = 5%2;    // 나머지
```

```
    uint f = 5**2;
```

```
    function arithmetic() public view returns(uint, uint, uint, uint, uint, uint) {  
        return(a, b, c, d, e, f);  
    }  
}
```

```
}
```

### Deployed Contracts

▼ EX4\_7 AT 0X332...D486D (MEMORY)

Balance: 0 ETH

arithmetic

0: uint256: 7

1: uint256: 3

2: uint256: 10

3: uint256: 1

4: uint256: 1

5: uint256: 25

## ■ 복합할당연산자(대입 + 산술연산)

```
// SPDX-License-Identifier: GPL-3.0  
pragma solidity >=0.7.0 < 0.9.0;
```

```
contract Ex4_8 {
```

```
    uint a = 5;
```

```
    uint b = 5;
```

```
    uint c = 5;
```

```
    uint d = 5;
```

```
    uint e = 5;
```

```
    function assignment() public returns(uint, uint, uint, uint, uint) {
```

```
        a += 2; // a = a + 2
```

```
        b -= 2; // b = b - 2
```

```
        c *= 2; // c = c * 2
```

```
        d /= 5; // d = d / 5
```

```
        e %= 2; // e = e % 2
```

```
        return(a, b, c, d, e);
```

```
    }
```

```
}
```

```
decoded output
```

```
{
```

```
  "0": "uint256: 7",
```

```
  "1": "uint256: 3",
```

```
  "2": "uint256: 10",
```

```
  "3": "uint256: 1",
```

```
  "4": "uint256: 1"
```

```
}
```



## 증감연산자: 변수 값을 1씩 증가 또는 감소

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.0 < 0.9.0;
```

```
contract Ex4_9 {
```

```
    uint a = 5;
```

```
    function justA() public view returns(uint){
        return a;
    }
```

```
    function prePlus() public returns(uint){
        return ++a; // a = a+1
    }
```

```
    function postPlus() public returns(uint){
        return a++; // a = a+1
    }
}
```

Deployed Contracts

EX4\_9 AT 0X5A8...C4D01 (MEMORY)

Balance: 0 ETH

postPlus

prePlus

**justA**

0: uint256: 5

Balance: 0 ETH

postPlus

prePlus

**justA**

0: uint256: 6

Input: 0xd7c...f1638

decoded Input: {}

decoded output: {"0": "uint256: 6"}

logs: []

val: 0 wei

Balance: 0 ETH

postPlus

prePlus

**justA**

0: uint256: 7

Input: 0xb7a...8fa3d

decoded Input: {}

decoded output: {"0": "uint256: 6"}

logs: []

val: 0 wei

- 증감연산자: 변수 값을 1씩 증가 또는 감소

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.0 < 0.9.0;

contract Ex4_10{

    uint a = 5;

    function justA() public view returns(uint){
        return a;
    }

    function preMinus() public returns(uint){
        return --a; // a = a-1
    }

    function postMinus() public returns(uint){
        return a--; // a = a-1
    }

}
```

## ■ 비교연산자

```
// SPDX-License-Identifier: GPL-3.0  
pragma solidity >=0.7.0 < 0.9.0;
```

```
contract Ex4_11 {
```

```
    bool a = 3>4;  
    bool b = 3<4;  
    bool c = 5>=2;  
    bool d = 5<=5;  
    bool e = 3==2;  
    bool f = 3!=2;
```

```
    function comparison() public view returns (bool, bool, bool, bool, bool, bool) {  
        return(a, b, c, d, e, f);  
    }  
}
```

### Deployed Contracts

▼ EX4\_11 AT 0X049...A1FD3 (MEMORY)

Balance: 0 ETH

comparison

0: bool: false

1: bool: true

2: bool: true

3: bool: true

4: bool: false

5: bool: true



## ■ 논리연산자 (AND,OR,NOT)

```
// SPDX-License-Identifier: GPL-3.0  
pragma solidity >=0.7.0 < 0.9.0;
```

```
contract Ex4_12 {
```

```
    bool a = true&&true;  
    bool b = true&&false;  
    bool c = false&&false;  
    bool d = true||true;  
    bool e = true||false;  
    bool f = false||false;  
    bool g = !false;
```

```
    function logical() public view returns (bool, bool, bool, bool, bool, bool, bool) {  
        return(a, b, c, d, e, f, g);  
    }
```

```
}
```

▼ EX4\_12 AT 0X38C...24C73 (MEMORY)

Balance: 0 ETH

logical

0: bool: true

1: bool: false

2: bool: false

3: bool: true

4: bool: true

5: bool: false

6: bool: true

## ■ 상수(constant)

- 변수와 달리 저장한 값을 바꿀 수 없다

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.0 < 0.9.0;

contract Ex4_13 {

    uint constant A = 13;

    function plusA() public pure returns (uint) {
        return A+10;
    }
    /*
    function changeA() public {
        A = 99; // 에러
    }
    */
}
```

```
from solidity:
TypeError: Cannot assign to a constant variable.
--> week3/Hello.sol:13:9:
   |
13 |         A = 99;
   |         ^
```

- pure 함수: 함수밖에 선언된 변수를 함수 내부에서 읽거나 변경 불가

```
// SPDX-License-Identifier: GPL-3.0  
pragma solidity >=0.7.0 <0.9.0;
```

```
contract Ex4_14 {  
    function myFun(uint a) public pure returns(uint){  
        return a;  
    }  
}
```

```
from solidity:  
Warning: Function state mutability can be restricted to pure  
--> week3/Hello.sol:5:5:  
|  
5 |     function myFun(uint a) public returns(uint){  
|     ^ (Relevant source part starts here and spans across multiple lines).
```

```
// SPDX-License-Identifier: GPL-3.0  
pragma solidity >=0.7.0 <0.9.0;
```

```
contract Ex4_15 {  
  
    uint a = 3;  
    function myFun() public pure returns(uint){  
        a = 4;        //오류  
        return a;      //오류  
    }  
}
```

```
TypeError: Function cannot be  
declared as pure because this  
expression (potentially)  
modifies the state.
```

```
--> week4/Ex4_15.sol:8:9:  
|  
8 |     a = 4; //오류  
|     ^
```

```
TypeError: Function declared as  
pure, but this expression  
(potentially) reads from the  
environment or state and thus  
requires "view".
```

```
--> week4/Ex4_15.sol:9:16:  
|  
9 |     return a; //오류  
|     ^
```

## ■ view 함수

- 함수밖에 선언된 변수를 함수 내부에서 읽을 수 있으나 변경 불가

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.0 <0.9.0;

contract Ex4_16 {
    uint public a = 4;

    function myFun() public view returns(uint){
        uint b = a + 5;
        return b;
    }
}
```

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.0 <0.9.0;

contract Ex4_17 {

    uint public a = 3;

    function myFun() public view returns(uint){
        a = 4; //오류
        return a;
    }
}
```

```
TypeError: Function cannot be
declared as view because this
expression (potentially)
modifies the state.
--> week4/Ex4_17.sol:9:9:
|
9 | a = 4; //오류
| ^
```

## ■ 솔리디티의 저장영역

### ■ storage

- 영속적으로 읽기/쓰기가 가능한 저장공간
- 함수 외부에 정의된 변수, 함수
- 키와 값 쌍의 매핑 구조

### ■ memory

- 휘발성 저장공간, 함수 실행 시 이용되다가 실행 후 지워진다
- 읽기/쓰기가 가능한 저장공간

### ■ calldata

- 트랜잭션의 매개변수를 위한 읽기 전용 공간
- external 함수의 참조 타입 매개변수

### ■ stack: EVM에서 사용하는 휘발성 공간

## ■ 솔리디티의 저장영역

- 참조 타입이 함수의 매개변수, 반환 값 또는 내부 변수로 정의될 때는 저장공간을 명시해야 한다. → 에러 발생

```
contract Ex4_18 {
    function myFun(string memory str)
        public
        pure
        returns(
            uint,
            string memory,
            bytes memory
        )
    {
        uint num = 99;
        bytes memory byt = hex"01";
        return (num, str, byt);
    }
}
```

```
TypeError: Data location must be
"memory" or "calldata" for
parameter in function, but none
was given.
```

```
--> week4/Ex4_18.sol:5:20:
|
5 | function myFun(string str)
| ^^^^^^^^^^^
```

```
TypeError: Data location must be
"memory" or "calldata" for
return parameter in function,
but none was given.
```

```
--> week4/Ex4_18.sol:10:13:
|
10 | string ,
| ^^^^^^
```

```
TypeError: Data location must be
"storage", "memory" or
"calldata" for variable, but
none was given.
```

```
--> week4/Ex4_18.sol:15:9:
|
15 | bytes byt = hex"01";
| ^^^^^^^^^^^
```

## ■ calldata 지정

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.0 <0.9.0;
```

```
contract Ex4_19 {
```

```
    function myFun(string calldata str) external pure returns(string memory){
        return str;
    }
}
```

```
from solidity:
TypeError: Data location must be "memory" or "calldata" for parameter in external function,
--> week3/Hello.sol:6:20:
   |
6 |     function myFun(string str) external pure returns(string memory){
   |                        ~~~~~
```

## ■ 상태변수와 지역변수

- 지역변수는 정의된 함수 밖에서 사용할 수 없다

```
contract Ex4_20 {  
  
    uint public a = 3;  
  
    function myFun1() external view returns(uint,uint) {  
        uint b = 4;  
        return (a,b);  
    }  
  
    /*  
    function myFun2() external pure returns(uint) {  
        return b;  
    }  
    */  
}
```

DeclarationError: Undeclared  
identifier.

--> week4/Ex4\_20.sol:15:16:

```
|  
15 | return b;  
| ^
```