

## 1. Git&amp;Github 소개

## 실습1-1 Git 설치/설정 실습

Step1. <https://git-scm.com> 이동 및 Git 설치파일 다운로드/설치

Step2. Git Bash 실행

Step3. Git 사용자 등록

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop
$ git config --global user.email "chhak0503@gmail.com"
$ git config --global user.name "chhak0503"
$ git config --list

```

## 실습1-2 Git 기본 명령어 실습

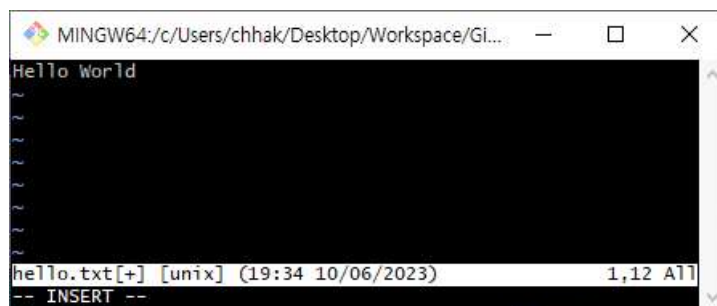
기본 명령어

명령어	사용 예	설명
ls	ls ls -al 또는 ll	현재 디렉터리의 파일 목록을 조회
cd	cd 또는 cd . cd ..	디렉터를 이동 ./ : 현재 디렉터리 ../ : 상위 디렉터리 ~/ : 홈 디렉터리
rm	rm aaa.txt	파일이나 디렉터를 삭제
mkdir	mkdir abc	새로운 디렉터리 생성
cat	cat a.txt	파일의 내용을 화면에 출력(concatenate)
clear	clear	터미널 화면을 깨끗하게 지워줌
vi	vi hello.txt	vi 편집기 실행

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop
$ cd ./Workspace
$ mkdir Git
$ cd ./Git
$ mkdir Ch01
$ cd Ch01
$ vi hello.txt

```



- ❶ i 입력
- ❷ "Hello World" 입력
- ❸ esc 입력
- ❹ :wq 입력

## 2. 버전관리

## 1) 버전관리 기본

## 실습1-1 Git 저장소 생성 및 문서 상태 확인

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop
$ cd ./Workspace/Git/

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git
$ git init

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git (master)
$ ls -al
total 4
drwxr-xr-x 1 chhak 197121 0 Jun 10 14:42 ./
drwxr-xr-x 1 chhak 197121 0 Jun 10 14:42 ../
drwxr-xr-x 1 chhak 197121 0 Jun 10 14:43 .git/
drwxr-xr-x 1 chhak 197121 0 Jun 10 16:03 Ch01/

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git (master)
$ git status
On branch master
No commits yet
nothing to commit (create/copy files and use "git add" to track)

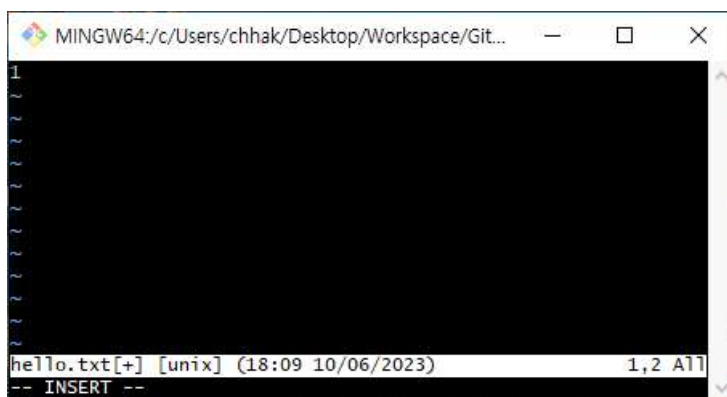
```

## 실습1-2 문서 생성 및 내용 입력

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git (master)
$ mkdir Ch02
$ cd Ch02
$ vi hello.txt

```



- ❶ i 입력
- ❷ 숫자 1 입력
- ❸ esc 입력
- ❹ :wq 입력

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ ll
total 1
-rw-r--r-- 1 chhak 197121 2 Jun 10 16:16 hello.txt

```

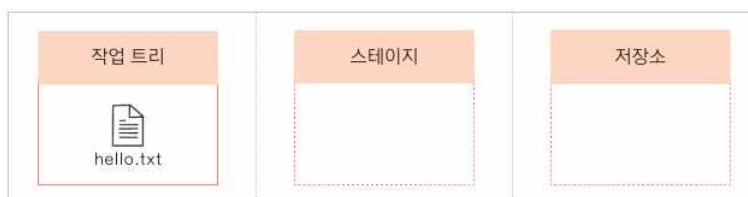
## 실습1-3 문서 현재 상태 확인

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ git status
On branch master

No commits yet

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

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



## 실습1-4 문서 Staging 하기

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ git add hello.txt

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ git status
On branch master

No commits yet

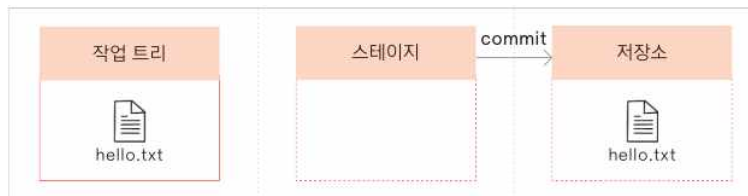
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   hello.txt
```



## 실습1-5 문서 Commit 하기

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ git commit -m "hello.txt 첫 commit"
[master (root-commit) cb8e0f1] hello.txt 첫 commit
1 file changed, 1 insertion(+)
create mode 100644 Ch02/hello.txt

$ git status
On branch master
nothing to commit, working tree clean
```



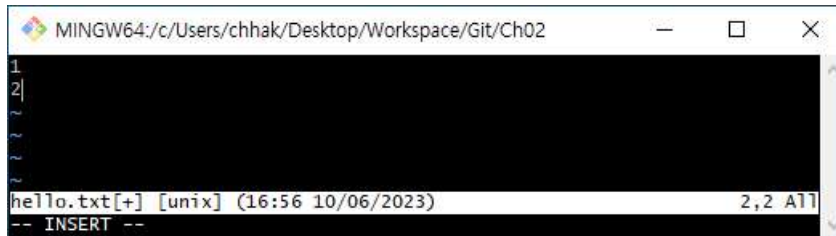
## 실습1-6 문서 버전 이력 확인하기

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ git log
commit cb8e0f15247a5691fdb7e768d6d6ec9360724046 (HEAD -> master)
Author: chhak0503 <chhak0503@gmail.com>
Date: Sat Jun 10 16:32:35 2023 +0900

    hello.txt 첫 commit
```

## 실습1-7 문서 버전 수정하기

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ vi hello.txt
```



- ① i 입력
- ② 숫자 2 추가
- ③ esc 입력
- ④ :wq 입력

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ git status
On branch master
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:   hello.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

## 실습1-9 문서 버전 Staging&amp;Commit 수행 후 이력 확인

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ git add hello.txt
```

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ git commit -m "숫자 2 추가"
[master d9eef59] 숫자 2 추가
1 file changed, 1 insertion(+)
```

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ git log
commit d9eef59c1ad7698b6beee6a14394d67171c55c50 (HEAD -> master)
Author: chhak0503 <chhak0503@gmail.com>
Date: Sat Jun 10 17:00:45 2023 +0900
```

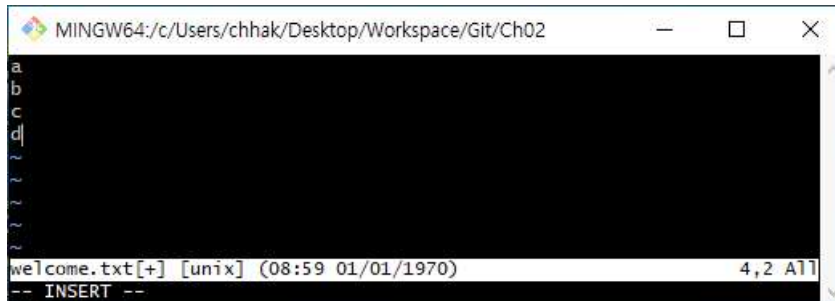
숫자 2 추가

```
commit cb8e0f15247a5691fdb7e768d6d6ec9360724046
Author: chhak0503 <chhak0503@gmail.com>
Date: Sat Jun 10 16:32:35 2023 +0900
```

hello.txt 첫 commit

## 실습1-10 새 문서 버전 추가하기

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ vi welcome.txt
```



- ① i 입력
- ② a, b, c, d 입력
- ③ esc 입력
- ④ :wq 입력

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    welcome.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

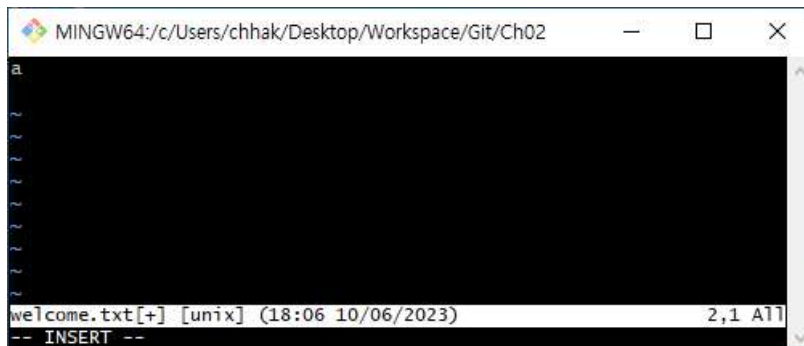
```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ git add welcome.txt
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   welcome.txt
```

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ git commit -m "welcome.txt 추가"
[master 0c9d4b8] welcome.txt 추가
1 files changed, 5 insertions(+)
create mode 100644 Ch02/welcome.txt
```

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ git status
On branch master
nothing to commit, working tree clean
```

## 실습1-11 문서 버전 최종 수정하기

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ vi welcome.txt
```



- ❶ i 입력
- ❷ b, c, d 삭제
- ❸ esc 입력
- ❹ :wq 입력

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
```

```
$ git status
```

```
On branch master
```

```
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: welcome.txt
```

```
no changes added to commit (use "git add" and/or "git commit -a")
```

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
```

```
$ git commit -am "welcome.txt b, c, d 삭제"
```

```
[master d28ae8c] welcome.txt b, c, d 삭제
```

```
1 file changed, 3 deletions(-)
```

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
```

```
$ git status
```

```
On branch master
```

```
nothing to commit, working tree clean
```

## 실습1-12 문서 버전 이력 상세 확인

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
```

```
$ git log --stat
```

```
commit d28ae8c1a4445518bf3e5b5b52261342e22416db (HEAD -> master)
```

```
Author: chhak0503 <chhak0503@gmail.com>
```

```
Date: Sat Jun 10 18:36:10 2023 +0900
```

```
welcome.txt b, c, d 삭제
```

```
Ch02/welcome.txt | 3 ---
```

```
1 file changed, 3 deletions(-)
```

```
commit d0d07310d747665f0092adfe3cc9257e1b95570c
```

```
Author: chhak0503 <chhak0503@gmail.com>
```

```
Date: Sat Jun 10 18:10:01 2023 +0900
```

```
welcome.txt 추가
```

```
Ch02/welcome.txt | 5 +++++
```

```
1 file changed, 5 insertions(+)
```

```
commit d5510cd46a29b79b0cad2b7b76bf88358a3ecab1
```

```
Author: chhak0503 <chhak0503@gmail.com>
```

```
Date: Sat Jun 10 18:09:48 2023 +0900
```

```
숫자 2 추가
```

```
Ch02/hello.txt | 2 ++
```

```
1 file changed, 2 insertions(+)
```

```
commit 30999baa4210d52f4bf8e6421e59d835a800c60c
```

```
Author: chhak0503 <chhak0503@gmail.com>
```

```
Date: Sat Jun 10 18:05:59 2023 +0900
```

```
hello.txt 첫 커밋
```

```
Ch02/hello.txt | 2 ++
```

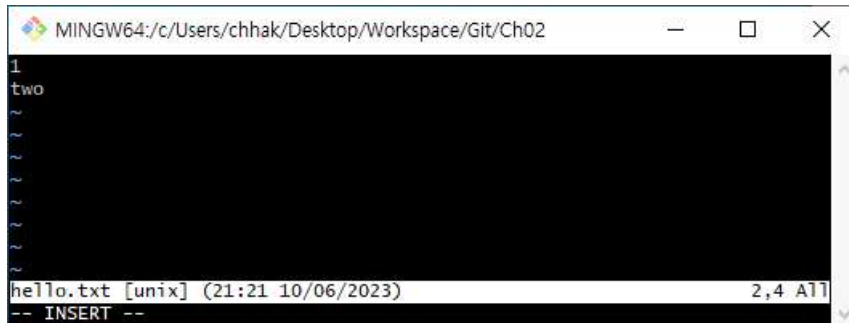
```
1 file changed, 2 insertions(+)
```



## 2) 버전관리 기타

실습2-1 작업 되돌리기 - git checkout -- <file>

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ vi hello.txt
```



- ❶ i 입력
- ❷ two 수정
- ❸ esc 입력
- ❹ :wq 입력

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ git status
On branch master
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:   hello.txt
```

no changes added to commit (use "git add" and/or "git commit -a")

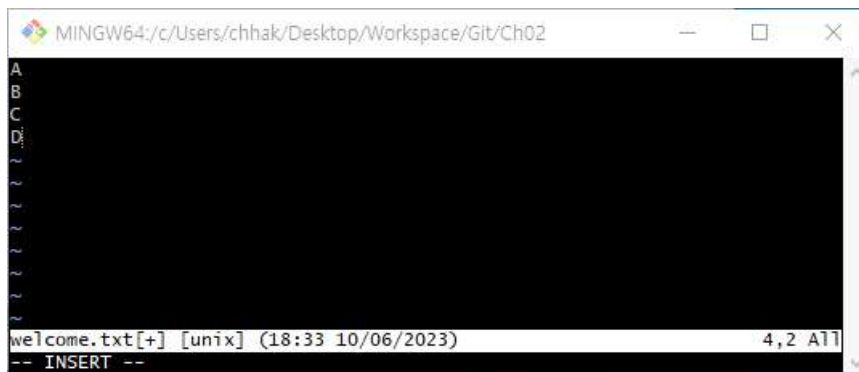
```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ cat hello.txt
1
two
```

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ git checkout -- hello.txt
```

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ cat hello.txt
1
2
```

## 실습2-2 Stage 내리기 - git reset HEAD

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ vi welcome.txt
```



- ❶ i 입력
- ❷ 기존 내용 삭제
- ❸ A, B, C, D 입력
- ❹ esc 입력
- ❺ :wq 입력

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ git add welcome.txt
```

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ git status
```

```
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)
    modified:   welcome.txt
```

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ git reset HEAD
```

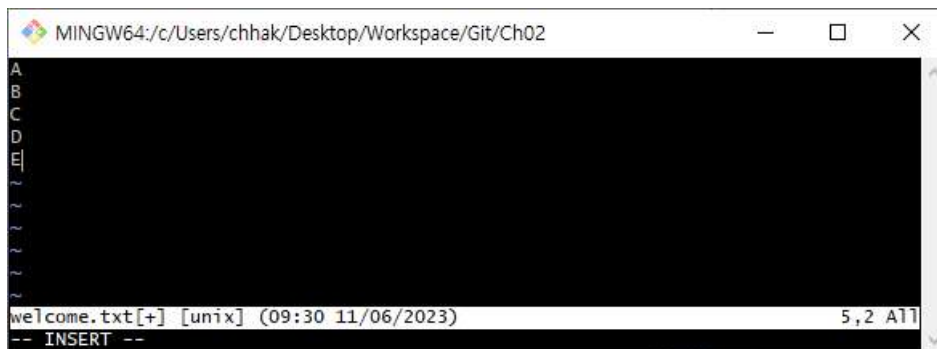
```
Unstaged changes after reset:
M       Ch02/welcome.txt
```

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ git status
```

```
On branch master
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:   welcome.txt
```

## 실습2-3 최신 버전으로 되돌리기 - git reset HEAD^

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ vi welcome.txt
```



- ❶ i 입력
- ❷ 'E' 추가
- ❸ esc 입력
- ❹ :wq 입력

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ git commit -am 'welcome.txt E 추가'
[master df3774e] welcome.txt E 추가
1 file changed, 5 insertions(+), 2 deletions(-)
```

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ git log
commit df3774e41e3614f5575815d007cbf05d9b58e243 (HEAD -> master)
Author: chhak0503 <chhak0503@gmail.com>
Date: Sun Jun 11 09:40:05 2023 +0900
```

```
welcome.txt E 추가
```

```
...
```

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ git reset HEAD^
Unstaged changes after reset:
M Ch02/welcome.txt
```

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ git log
commit d28ae8c1a4445518bf3e5b5b52261342e22416db (HEAD -> master)
Author: chhak0503 <chhak0503@gmail.com>
Date: Sun Jun 11 09:40:05 2023 +0900
```

```
welcome.txt b, c, d 삭제
```

```
...
```

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch02 (master)
$ git status
On branch master
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:   welcome.txt
```

## 3. 원격 저장소

## 1) 원격 저장소 생성과 연동

## 실습1-1 Github 원격 저장소 생성하기

- ❶ 저장소 이름 입력(필수)
- ❷ 저장소 설명 입력(필수)
- ❸ 저장소 공개 여부(Public 선택)
- ❹ 저장소 README 파일 여부(체크안함)
- ❺ 저장소 ignore 파일 여부(None)
- ❻ 저장소 라이선스 여부(None)

## 실습1-2 지역 저장소와 원격 저장소 연동하기

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git (master)
$ git remote add origin https://github.com/계정명/저장소명.git

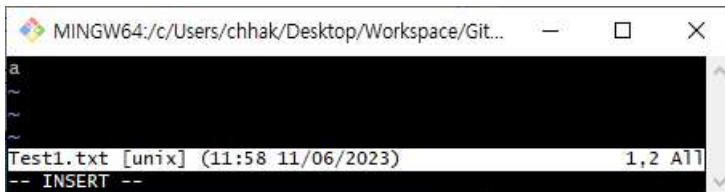
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git (master)
$ git remote -v
origin  https://github.com/계정명/저장소명.git (fetch)
origin  https://github.com/계정명/저장소명.git (push)

```

## 2) 원격 저장소 push와 pull

실습2-1 문서 생성 및 내용 입력(각 파일 내용 동일)

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch03 (master)
$ vi Test1.txt
$ vi Test2.txt
$ vi Test3.txt
```



- ❶ i 입력
- ❷ a 입력
- ❸ esc 입력
- ❹ :wq 입력

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch03 (master)
$ ls -l
total 3
-rw-r--r-- 1 chhak 197121 2 Jun 11 11:58 Test1.txt
-rw-r--r-- 1 chhak 197121 2 Jun 11 11:58 Test2.txt
-rw-r--r-- 1 chhak 197121 2 Jun 11 11:58 Test3.txt
```

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch03 (master)
$ cd ..
```

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git (master)
$ git add Ch03/
```

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git (master)
$ git commit -m 'Ch03 수업예제 추가'
[master 59ca7be] Ch03 수업예제 추가
3 files changed, 3 insertions(+)
create mode 100644 Ch03/Test1.txt
create mode 100644 Ch03/Test2.txt
create mode 100644 Ch03/Test3.txt
```

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git (master)
$ git log
commit 59ca7be2f049f03453de48214a8489555f7d1e38 (HEAD -> master)
Author: chhak0503 <chhak0503@gmail.com>
Date: Sun Jun 11 13:04:58 2023 +0900
```

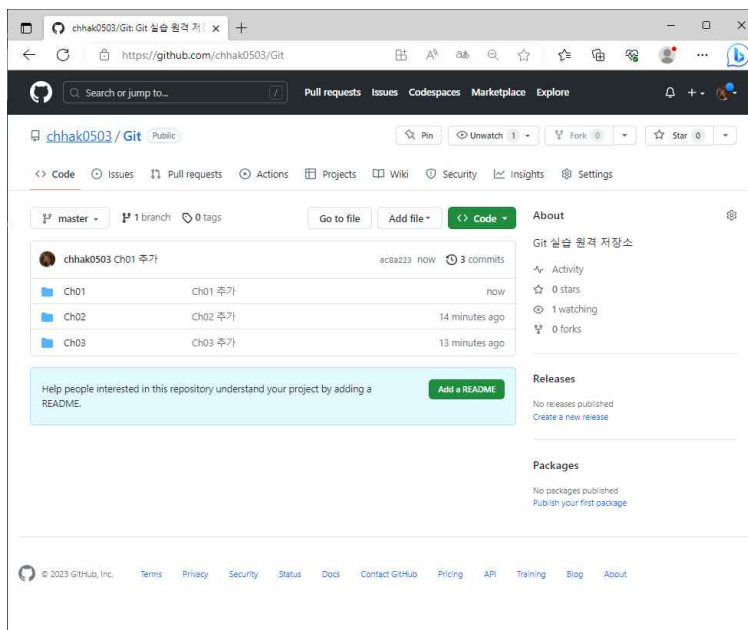
Ch03 수업예제 추가

## 실습2-2 원격 저장소 push 하기

```

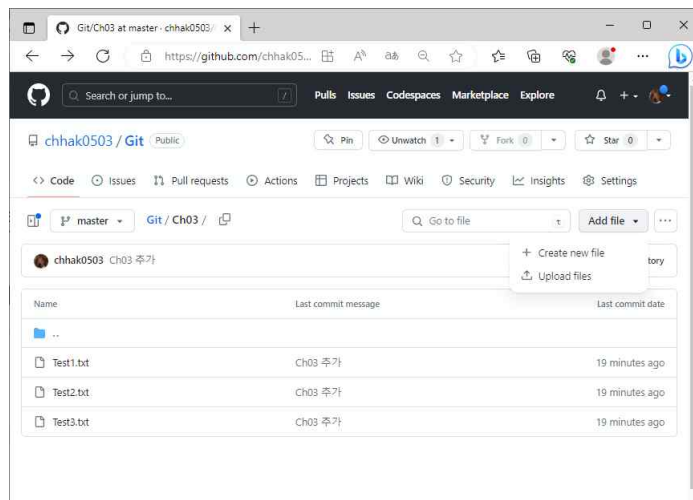
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git (master)
$ git push origin master
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 16 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (13/13), 942 bytes | 471.00 KiB/s, done.
Total 13 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/chhak0503/Git.git
 * [new branch]      master -> master

```

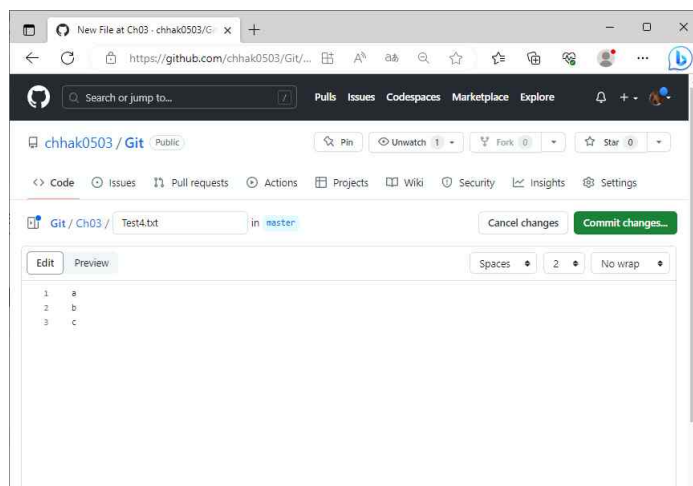


- ① 원격저장소 확인
- ② Commit 이력 확인

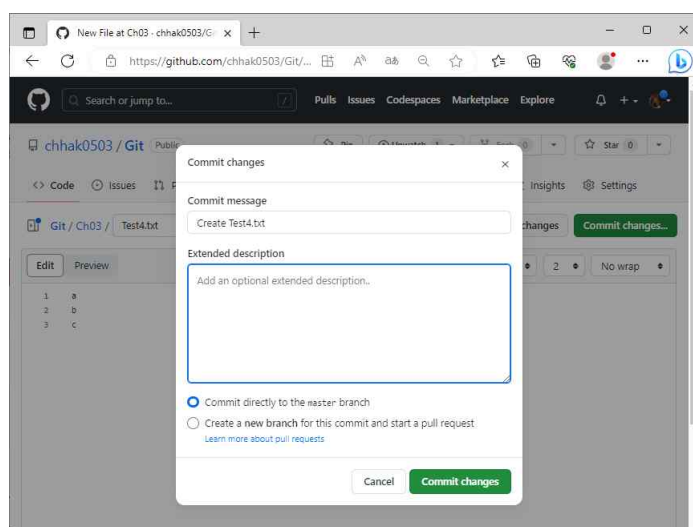
## 실습2-3 원격 저장소 pull 하기



- ① Add file 클릭
- ② Create new file 클릭



- ① 파일명 Test4 입력
- ② a, b, c 입력
- ③ Commit changes... 클릭



- ① Commit message 입력
- ② Commit changes 클릭
- ③ Test4.txt 파일 생성 확인
- ④ Commit 이력 확인

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git (master)
$ git pull origin master
From https://github.com/chhak0503/Git
* branch          master      -> FETCH_HEAD
Updating ac8a223..fa482d1
Fast-forward
 Ch03/Test4.txt | 3 +++
1 file changed, 3 insertions(+)
create mode 100644 Ch03/Test4.txt

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git (master)
$ cd Ch03

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch03 (master)
$ ls -l
total 4
-rw-r--r-- 1 chhak 197121 2 Jun 11 11:58 Test1.txt
-rw-r--r-- 1 chhak 197121 2 Jun 11 11:58 Test2.txt
-rw-r--r-- 1 chhak 197121 2 Jun 11 11:58 Test3.txt
-rw-r--r-- 1 chhak 197121 9 Jun 11 14:54 Test4.txt

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch03 (master)
$ cat Test4.txt
a
b
c

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch03 (master)
$ git log
commit fa482d13f64903ddfc7e503ecca42ec2e0d447bf (HEAD -> master, origin/master)
Author: chhak0503 <64509878+chhak0503@users.noreply.github.com>
Date:   Sun Jun 11 14:42:38 2023 +0900

    Create Test4.txt
...
```



## 3) 원격 저장소 clone

## 실습3-1 원격 저장소 clone 하기

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop
$ git clone https://github.com/계정명/저장소명.git Git_home
Cloning into 'Git_home'...
remote: Enumerating objects: 17, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 17 (delta 2), reused 11 (delta 0), pack-reused 0
Receiving objects: 100% (17/17), done.
Resolving deltas: 100% (2/2), done.

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop
$ git clone https://github.com/계정명/저장소명.git Git_office
Cloning into 'Git_office'...
remote: Enumerating objects: 17, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 17 (delta 2), reused 11 (delta 0), pack-reused 0
Receiving objects: 100% (17/17), done.
Resolving deltas: 100% (2/2), done.

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop
$ cd Git_home/

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Git_home (master)
$ git log
commit fa482d13f64903ddfc7e503ecca42ec2e0d447bf (HEAD -> master, origin/master,
origin/HEAD)
Author: chhak0503 <64509878+chhak0503@users.noreply.github.com>
Date: Sun Jun 11 14:42:38 2023 +0900

    Create Test4.txt
    ...

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Git_home (master)
$ git remote -v
origin https://github.com/chhak0503/Git.git (fetch)
origin https://github.com/chhak0503/Git.git (push)
```

## 실습3-2 원격 저장소 push 하기

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Git_home/Ch03 (master)
$ vi Test4.txt
```



- ❶ i 입력
- ❷ d, e, f 입력
- ❸ esc 입력
- ❹ :wq 입력

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Git_home/Ch03 (master)
$ git add Test4.txt
```

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Git_home/Ch03 (master)
$ git commit -m 'd, e, f 추가'
[master 7ccd5ba] d, e, f 추가
1 file changed, 3 insertions(+)
```

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Git_home/Ch03 (master)
$ git push origin master
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 326 bytes | 326.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/chhak0503/Git.git
fa482d1..7ccd5ba master -> master
```

## 실습3-3 원격 저장소 pull 하기

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Git_office (master)
$ git pull
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 4 (delta 2), reused 4 (delta 2), pack-reused 0
Unpacking objects: 100% (4/4), 306 bytes | 21.00 KiB/s, done.
From https://github.com/chhak0503/Git
   fa482d1..7ccd5ba  master    -> origin/master
Updating fa482d1..7ccd5ba
Fast-forward
 Ch03/Test4.txt | 3 +++
1 file changed, 3 insertions(+)

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Git_office (master)
$ cd Ch03

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Git_office/Ch03 (master)
$ ll
total 4
-rw-r--r-- 1 chhak 197121  3 Jun 11 15:09 Test1.txt
-rw-r--r-- 1 chhak 197121  3 Jun 11 15:09 Test2.txt
-rw-r--r-- 1 chhak 197121  3 Jun 11 15:09 Test3.txt
-rw-r--r-- 1 chhak 197121 18 Jun 11 15:29 Test4.txt

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Git_office/Ch03 (master)
$ cat Test4.txt
a
b
c
d
e
f
```

## 4. 브랜치

## 1) 브랜치 기본

## 실습1-1 실습 디렉터리 생성

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git (master)
$ mkdir Ch04
$ cd Ch04
$ vi work.txt

```



- ❶ i 입력
- ❷ content 1 입력
- ❸ esc 입력
- ❹ :wq 입력

## 실습1-2 Commit &amp; log 확인

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch04 (master)
$ git add work.txt

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch04 (master)
$ git commit -m 'work 1'

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch04 (master)
$ git log
commit ea2d890285646763f7d20cf98bbaf2811eda0576 (HEAD -> master)
Author: chhak0503 <chhak0503@gmail.com>
Date:   Fri Aug 25 17:31:02 2023 +0900
    work 1
...

```

## 실습1-3 work.txt 수정 후 Commit &amp; log 확인

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch04 (master)
$ vi work.txt ← 'content 2' 추가 입력 후 저장
$ git commit -am 'work 2'

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch04 (master)
$ vi work.txt ← 'content 3' 추가 입력 후 저장
$ git commit -am 'work 3'

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch04 (master)
$ git log
commit 1cc55821e798613e54ced48889d1687282f62a2c (HEAD -> master)
Author: chhak0503 <chhak0503@gmail.com>
Date:   Fri Aug 25 17:31:02 2020 +0900
    work 3
...

```

## 실습1-4 현재 branch 확인

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch04 (master)
$ git branch
* master
```

## 실습1-5 branch aaa 생성 및 확인

```
java@DESKTOP-6PB29LQ MINGW64 ~/Desktop/Workspace/Git/Ch04 (master)
$ git branch aaa

java@DESKTOP-6PB29LQ MINGW64 ~/Desktop/Workspace/Git/Ch04 (master)
$ git branch
aaa
* master

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch04 (master)
$ git log
commit 1cc55821e798613e54ced48889d1687282f62a2c (HEAD -> master, aaa)
Author: chhak0503 <chhak0503@gmail.com>
Date:   Fri Aug 25 17:31:02 2020 +0900
    work 3
    ...
```

## 실습1-6 branch bbb, ccc 생성 및 확인

```
java@DESKTOP-6PB29LQ MINGW64 ~/Desktop/Workspace/Git/Ch04 (master)
$ git branch bbb

java@DESKTOP-6PB29LQ MINGW64 ~/Desktop/Workspace/Git/Ch04 (master)
$ git branch ccc

java@DESKTOP-6PB29LQ MINGW64 ~/Desktop/Workspace/Git/Ch04 (master)
$ git branch
aaa
bbb
ccc
* master

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch04 (master)
$ git log
commit 1cc55821e798613e54ced48889d1687282f62a2c (HEAD -> master, ccc, bbb, aaa)
Author: chhak0503 <chhak0503@gmail.com>
Date:   Fri Aug 25 17:31:02 2020 +0900
    work 3
    ...
```

## 실습1-7 work.txt 수정 후 commit/log 확인

```

java@DESKTOP-6PB29LQ MINGW64 ~/Desktop/Workspace/Git/Ch04 (master)
$ vi work.txt ← 'content 4(master)' 추가 입력 후 저장
$ git commit -am 'master content 4'
$ git log --oneline
45fb5f7 (HEAD -> master) master content4
1cc5582 (ccc, bbb, aaa) work 3
b6cac4c work 2
ea2d890 work 1

```

## 실습1-8 aaa 브랜치 전환 후 내용 확인

```

java@DESKTOP-6PB29LQ MINGW64 ~/Desktop/Workspace/Git/Ch04 (master)
$ git checkout aaa
Switched to branch 'aaa'

java@DESKTOP-6PB29LQ MINGW64 ~/Desktop/Workspace/Git/Ch04 (aaa)
$ git log --oneline
1cc5582 (HEAD -> aaa, ccc, bbb) work 3
b6cac4c work 2
ea2d890 work 1

java@DESKTOP-6PB29LQ MINGW64 ~/Desktop/Workspace/Git/Ch04 (aaa)
$ cat work.txt
content 1
content 2
content 3

```

## 실습1-9 aaa 브랜치에서 commit/log 확인

```

java@DESKTOP-6PB29LQ MINGW64 ~/Desktop/Workspace/Git/Ch04 (master)
$ vi work.txt ← 'content 4(aaa)' 추가 입력 후 저장
$ vi aaa.txt ← 'content 4(aaa)' 입력 후 저장
$ git add work.txt aaa.txt
$ git commit -m 'aaa content 4'
$ git log --oneline --branches --graph
* 83bd291 (HEAD -> aaa) aaa content 4
| * 45fb5f7 (master) master content4
|/
* 1cc5582 (ccc, bbb) work 3
* b6cac4c work 2
* ea2d890 work 1

```

## 2) 브랜치 병합 기본

## 실습2-1 실습 디렉터리 생성 및 파일 생성

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git (master)
$ mkdir Ch05
$ cd Ch05
$ vi work.txt ← '1' 입력 후 저장
$ git add work.txt
$ git commit -m 'work 1'

```



## 실습2-2 o2 브랜치 생성

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch05 (master)
$ git branch o2

```

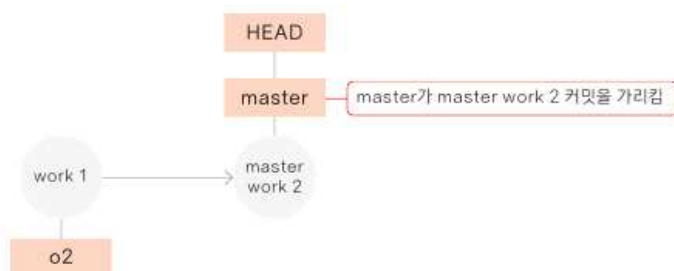


## 실습2-3 master.txt 파일 생성

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch05 (master)
$ vi master.txt ← 'master 2' 입력 후 저장
$ git add master.txt
$ git commit -m 'master work 2'

```

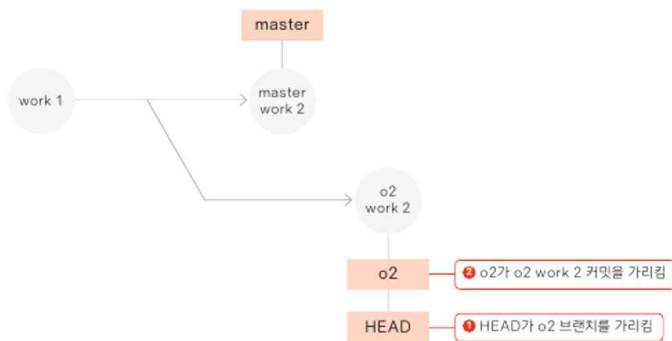


## 실습2-4 o2 브랜치 전환(체크아웃) 후 Commit/Log 확인

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch05 (master)
$ git checkout o2
$ vi o2.txt ← 'o2 2' 입력 후 저장
$ git add o2.txt
$ git commit -m 'o2 work 2'
$ git log --oneline --branches --graph
* b78ffdc (HEAD -> o2) o2 work 2
| * 45af0ac (master) master work 2
| /
* 4b74ad1 work 1

```

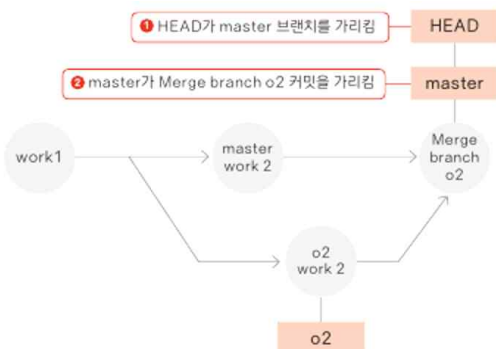


## 실습2-5 master 브랜치 전환(체크아웃) 후 병합(merge)

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch05 (o2)
$ git checkout master
$ git merge o2
$ git log --oneline --branches --graph
* 4246810 (HEAD -> master) Merge branch 'o2'
| \
| * e5d3a9f (o2) o2 work 2
* | 79febae master work 2
|/
* dbd64ba work 1

```



## 실습2-6 o2 브랜치 삭제

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch05 (master)
$ git branch -d o2

```



## 3) 브랜치 병합 - 같은 파일의 다른 위치를 수정 후 병합

## 실습3-1 실습 디렉터리 생성 및 파일 생성/입력

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git (master)
$ mkdir Ch06
$ cd Ch06
$ vi work.txt

```

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch06 (master)
$ git add work.txt

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch06 (master)
$ git commit -m 'work 1'

```

## 실습3-2 o2 브랜치 생성 및 work.txt 수정 후 commit

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch06 (master)
$ git branch o2

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch06 (master)
$ vi work.txt

```

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch06 (master)
$ git commit -am 'master work 2'

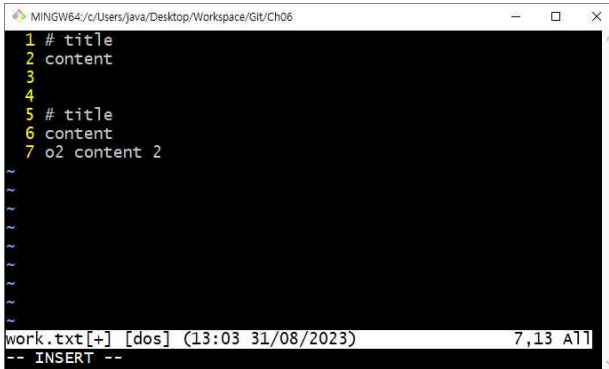
```

## 실습3-3 o2 브랜치 전환 및 work.txt 수정 후 commit

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch06 (master)
$ git checkout o2
Switched to branch 'o2'
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch06 (o2)
$ vi work.txt

```



```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch06 (o2)
$ git commit -am 'o2 work 2'

```

## 실습3-4 master 브랜치 전환 후 병합(merge)

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch06 (o2)
$ git checkout master
Switched to branch 'master'

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch06 (master)
$ git merge o2
Auto-merging Ch06/work.txt
Merge made by the 'ort' strategy.
Ch06/work.txt | 1 +
1 file changed, 1 insertion(+)

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch06 (master)
$ cat work.txt
# title
content
master content 2

# title
content
o2 content 2

```

## 실습3-5 o2 브랜치 삭제

```

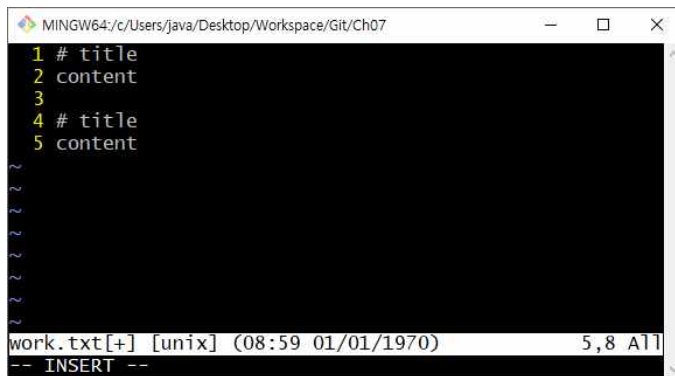
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch06 (master)
$ git branch -d o2

```

## 4) 브랜치 병합 - 같은 파일의 같은 위치를 수정 후 병합

## 실습4-1 실습 디렉터리 생성 및 파일 생성/입력


```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git (master)
$ mkdir Ch07
$ cd Ch07
$ vi work.txt
```



```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch07 (master)
$ git add work.txt
$ git commit -m 'work 1'
```

## 실습4-2 o2 브랜치 생성 및 work.txt 수정 후 Commit

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch07 (master)
$ git branch o2
$ vi work.txt
```



```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch07 (master)
$ git commit -am 'master work 2'
```

## 실습4-3 o2 브랜치 전환 및 work.txt 수정 후 Commit

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch07 (master)
$ git checkout o2
Switched to branch 'o2'

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch07 (o2)
$ vi work.txt

```

```

MINGW64/c/Users/java/Desktop/Workspace/Git/Ch07
# title
content
o2 content 2|
# title
content
~
~
~
~
~
~
~
~
~
~
work.txt[+] [dos] (13:25 31/08/2023) 3,13 All
-- INSERT --

```

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch07 (o2)
$ git commit -am 'o2 work 2'

```

## 실습4-4 master 브랜치 전환 후 병합(merge)

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch07 (o2)
$ git checkout master
Switched to branch 'master'

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch07 (master)
$ git merge o2
Auto-merging Ch07/work.txt
CONFLICT (content): Merge conflict in Ch07/work.txt
Automatic merge failed; fix conflicts and then commit the result.

```

## 실습4-5 파일 내용 확인

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch07 (master)
$ vi work.txt

```

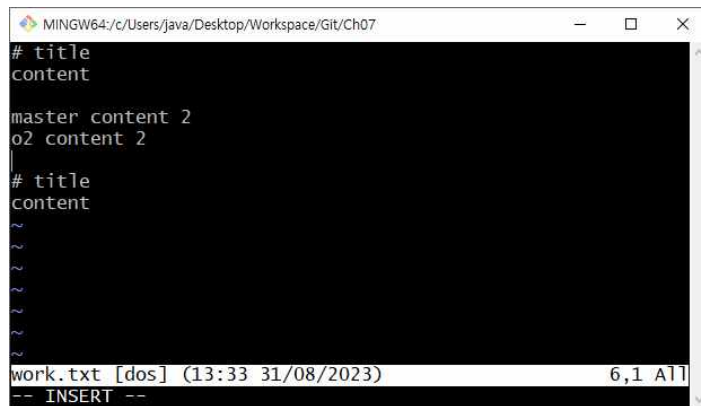
```

MINGW64/c/Users/java/Desktop/Workspace/Git/Ch07
# title
content
<<<<<<< HEAD|
master content 2
=====
o2 content 2
>>>>>>> o2
# title
content
~
~
~
~
~
~
~
~
~
~
work.txt [dos] (13:29 31/08/2023) 3,13 All
-- INSERT --

```

## 실습4-6 파일 내용 최종 수정 후 Commit

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch07 (master)
$ vi work.txt
```



```
MINGW64:~/Desktop/Workspace/Git/Ch07
# title
content

master content 2
o2 content 2

# title
content
~
~
~
~
~
~
~
work.txt [dos] (13:33 31/08/2023) 6,1 All
-- INSERT --
```

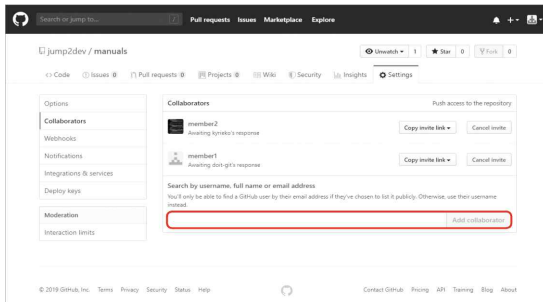
```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch07 (master)
$ git commit -am 'merge o2 branch'
```

## 실습4-7 o2 브랜치 삭제

```
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch07 (master)
$ git branch -d o2
```

## 5) GitHub Flow 브랜치 전략

## 실습5-1 Github 공동 작업자 추가



- ① Github - Setting - Collaborators
- ② Add collaborator
- ③ Collaborators 확인

## 실습5-2 실습 디렉터리 생성 및 파일 생성/입력

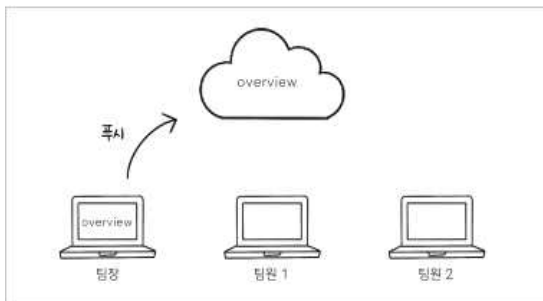
```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git (master)
$ mkdir Ch08
$ cd Ch08
$ vi overview.txt ← 'master 1' 입력 후 저장
  
```

## 실습5-3 작업 파일 Commit &amp; push

```

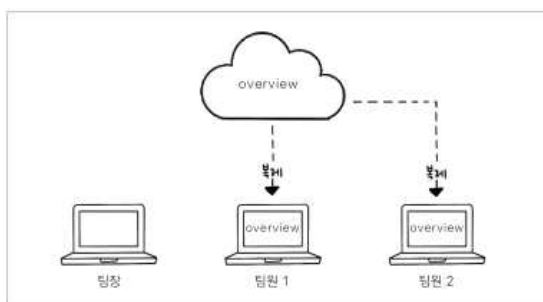
chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Workspace/Git/Ch08 (master)
$ git add overview.txt
$ git commit -m 'overview master 1'
$ git remote add origin 원격_저장소_주소
$ git push -u origin master
  
```



## 실습5-4 협업 작업자 원격저장소 Clone 하기

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop
$ git clone 원격_저장소_주소
  
```



## 실습5-5 feature1 브랜치 생성 및 전환 후 파일 생성 &amp; Commit

```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Git_Team1/Ch08 (master)
$ git branch feature1
$ git checkout feature1
Switched to branch 'feature1'

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Git_Team1/Ch08 (feature1)
$ vi overview.txt ← 2번째 줄 'feature1 1' 입력 후 저장
$ git add overview.txt
$ git commit -m 'feature1'

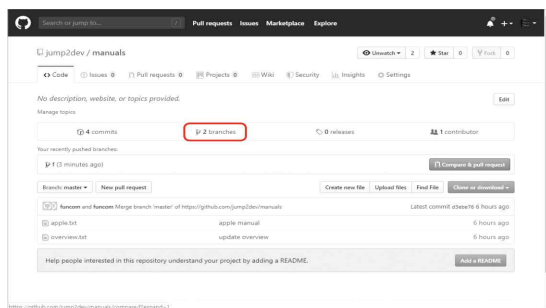
```

## 실습5-6 원격 저장소 feature1 브랜치로 push 후 확인

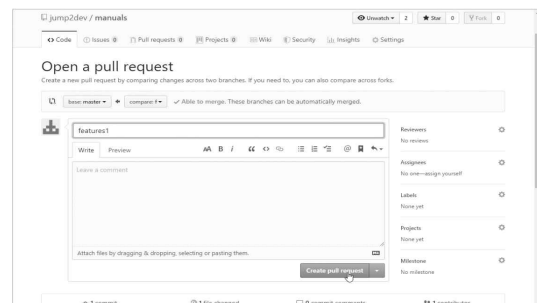
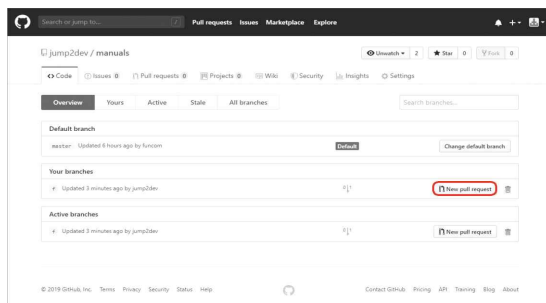
```

chhak@DESKTOP-J3DSA4P MINGW64 ~/Desktop/Git_Team1/Ch08
$ git push origin feature1

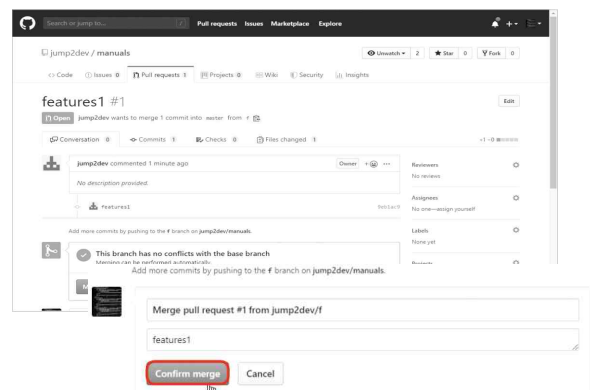
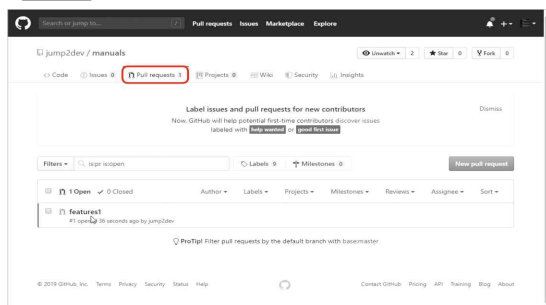
```



## 실습5-7 pull request 요청 및 메시지 작성



## 실습5-8 pull request 요청 확인 및 병합(merge)



## 5. Github Actions

## 1) AWS IAM 설정 - EC2 역할 생성

실습1-1 AWS 전체 서비스 &gt; IAM &gt; 액세스 관리 &gt; 역할 &gt; 역할 만들기

**역할 (3) 정보**  
IAM 역할은 단기간 동안 유효한 자격 증명을 가진 특정 권한이 있는 자격 증명입니다. 신뢰할 수 있는 개체가 역할을 맡을 수 있습니다.

🔍 검색

<input type="checkbox"/>	역할 이름	신뢰할 수 있는 개체	마지막 활동
<input type="checkbox"/>	<a href="#">AWSServiceRoleForAmazonSSM</a>	AWS 서비스: ssm (서비스 연결 역할)	-
<input type="checkbox"/>	<a href="#">AWSServiceRoleForSupport</a>	AWS 서비스: support (서비스 연결 역할)	-
<input type="checkbox"/>	<a href="#">AWSServiceRoleForTrustedAdvisor</a>	AWS 서비스: trustedadvisor (서비스 연결 역할)	-

실습1-2 엔터티 유형 및 사용 사례 선택

**신뢰할 수 있는 엔터티 유형**

☒ **AWS 서비스**  
EC2, Lambda 등의 AWS 서비스가 이 계정에 작업을 수행하도록 허용합니다.

☐ **AWS 계정**  
사용자 또는 서드 파티에 속한 다른 AWS 계정의 엔터티가 이 계정에서 작업을 수행하도록 허용합니다.

☐ **웹 자격 증명**  
지정된 외부 웹 자격 증명 공급자와 연동된 사용자가 이 역할을 맡아 이 계정에서 작업을 수행하도록 허용합니다.

☐ **SAML 2.0 연동**  
기업 디렉터리에서 SAML 2.0과 연동된 사용자가 이 계정에서 작업을 수행할 수 있도록 허용합니다.

☐ **사용자 지정 신뢰 정책**  
다른 사용자가 이 계정에서 작업을 수행할 수 있도록 사용자 지정 신뢰 정책을 생성합니다.

- AWS 서비스 선택
- 사용 사례 : EC2 선택




## 실습1-3 2단계 권한 추가

**권한 정책 (1/882) 정보** 🔄

새 역할에 연결할 정책을 하나 이상 선택합니다.

필터링 기준 유형

🔍 S3Full ✕ 모든 유형 ▼ 1 개 일치 < 1 > ⚙️

<input checked="" type="checkbox"/>	정책 이름 <span>🔗</span>	유형 <span>▼</span>	설명
<input checked="" type="checkbox"/>	 AmazonS3FullAccess	AWS 관리형	Provides full access to all buckets via the AWS Management Console.

▶ 권한 경계 설정 - 선택 사항

취소 이전 다음

- "S3Full" 검색
- AmazonS3FullAccess 권한 체크

## 실습1-4 3단계 이름 지정, 검토 및 생성

**역할 세부 정보**

**역할 이름**  
이 역할을 식별하는 의미 있는 이름을 입력합니다.

chhak2023-iam-ec2

최대 64자입니다. 영숫자 및 '+', '@', '\_' 문자를 사용하세요.

**설명**  
이 역할에 대하여 간단한 설명을 추가합니다.

Allows EC2 instances to call AWS services on your behalf.

최대 1000자입니다. 영숫자 및 '+', '@', '\_' 문자를 사용하세요.

- 역할 이름 : 사용자계정-iam-for-ec2
- 작성 후 역할 생성

## 2) AWS IAM 설정 - CodeDeploy 역할 생성

## 실습2-1 AWS 전체 서비스 &gt; IAM &gt; 액세스 관리 &gt; 역할 &gt; 역할 만들기

**역할 (3) 정보**  
IAM 역할은 단기간 동안 유효한 자격 증명을 가진 특정 권한이 있는 자격 증명입니다. 신뢰할 수 있는 개체가 역할을 맡을 수 있습니다.

Q 검색

<input type="checkbox"/>	역할 이름	신뢰할 수 있는 개체	마지막 활동
<input type="checkbox"/>	<a href="#">AWSServiceRoleForAmazonSSM</a>	AWS 서비스: ssm (서비스 연결 역할)	-
<input type="checkbox"/>	<a href="#">AWSServiceRoleForSupport</a>	AWS 서비스: support (서비스 연결 역할)	-
<input type="checkbox"/>	<a href="#">AWSServiceRoleForTrustedAdvisor</a>	AWS 서비스: trustedadvisor (서비스 연결 역할)	-

## 실습2-2 엔터티 유형 및 사용 사례 선택

**신뢰할 수 있는 엔터티 유형**

☒ **AWS 서비스**  
EC2, Lambda 등의 AWS 서비스가 이 계정에 서 작업을 수행하도록 허용합니다.

☐ **AWS 계정**  
사용자 또는 서드 파티에 속한 다른 AWS 계정의 엔터티가 이 계정에서 작업을 수행하도록 허용합니다.

☐ **웹 자격 증명**  
지정된 외부 웹 자격 증명 공급자와 연동된 사용자가 이 역할을 맡아 이 계정에서 작업을 수행하도록 허용합니다.

☐ **SAML 2.0 연동**  
기업 디렉터리에서 SAML 2.0과 연동된 사용자가 이 계정에서 작업을 수행할 수 있도록 허용합니다.

☐ **사용자 지정 신뢰 정책**  
다른 사용자가 이 계정에서 작업을 수행할 수 있도록 사용자 지정 신뢰 정책을 생성합니다.

- AWS 서비스 선택
- 사용 사례 : CodeDeploy 선택

## 실습2-3 2단계 권한 추가

**권한 정책 (1) 정보**  
 선택한 역할 유형에는 다음 정책이 필요합니다.

정책 이름	유형
AWSCodeDeployRole	AWS 관리형

▶ 권한 경계 설정 - 선택 사항

취소 이전 다음

- AWSCodeDeployRole 기본권한 확인 후 다음

## 실습2-4 3단계 이름 지정, 검토 및 생성

**역할 세부 정보**

**역할 이름**  
 이 역할을 식별하는 의미 있는 이름을 입력합니다.

chhak2023-iam-for-codedeploy

최대 64자입니다. 영숫자 및 '+', '@', '-' 문자를 사용하세요.

**설명**  
 이 역할에 대하여 간단한 설명을 추가합니다.

Allows CodeDeploy to call AWS services such as Auto Scaling on your behalf.

최대 1000자입니다. 영숫자 및 '+', '@', '-' 문자를 사용하세요.

- 역할 이름 : 사용자계정-iam-for-codedeploy
- 작성 후 역할 생성

## 3) AWS IAM 설정 - Github Actions 사용자 생성

## 실습3-1 AWS 전체 서비스 &gt; IAM &gt; 액세스 관리 &gt; 사용자 &gt; 사용자 생성

## 실습3-2 1단계 사용자 세부 정보 지정

- 사용자 이름 : 사용자계정-iam-user-for-github-actions

- 작성 후 다음

## 실습3-3 2단계 권한 설정

- 직접 정책 연결 선택

**권한 정책 (2/1132)** 정책 생성

새 사용자에게 연결할 정책을 하나 이상 선택합니다.

필터링 기준 유형

CodeDeployFull 모든 유형 1 개 일치

☒ 정책 이름 유형 연결된 엔터티

<input checked="" type="checkbox"/>	AWSCodeDeployFullAccess	AWS 관리형	1
-------------------------------------	-------------------------	---------	---

▶ 권한 경계 설정 - 선택 사항

취소 이전 다음

- “S3Full” 검색 후 AmazonS3FullAccess 체크
- “CodeDeployFull” 검색 후 AWSCodeDeployFullAccess 체크
- 다음 클릭

#### 실습3-4 3단계 검토 및 생성

### 검토 및 생성

선택 사항을 검토합니다. 사용자를 생성한 후 자동 생성된 암호를 보고 다운로드할 수 있습니다(활성화된 경우).

**사용자 세부 정보**

사용자 이름 chhak2023-iam-user-for-github-actions	콘솔 암호 유형 None	암호 재설정 필요 아니요
---	------------------	------------------

**권한 요약** < 1 >

이름	유형	다음과 같이 사용
<a href="#">AmazonS3FullAccess</a>	AWS 관리형	권한 정책
<a href="#">AWSCodeDeployFullAccess</a>	AWS 관리형	권한 정책

- 검토 후 사용자 생성 후 사용자 목록 이동

#### 실습3-5 사용자 정보 이동

[IAM](#) > [사용자](#) > chhak2023-iam-user-for-github-actions

**chhak2023-iam-user-for-github-actions** 정보 삭제

**요약**

ARN arn:aws:iam::509886355954:user/chhak2023-iam-user-for-github-actions	콘솔 액세스 비활성화됨	엑세스 키 1 <a href="#">엑세스 키 만들기</a>
생성됨 October 20, 2023, 12:28 (UTC+09:00)	마지막 콘솔 로그인 -	

- 액세스 키 만들기 클릭

## 실습3-6 액세스 키 모범 사례 및 대안

액세스 키 모범 사례 및 대안 정보

보안 개선을 위해 액세스 키와 같은 장기 자격 증명을 사용하지 마세요. 다음과 같은 사용 사례와 대안을 고려하세요.

**사용 사례**

☐ Command Line Interface(CLI)  
AWS CLI를 사용하여 AWS 계정에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.

☐ 로컬 코드  
로컬 개발 환경의 애플리케이션 코드를 사용하여 AWS 계정에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.

☒ AWS 컴퓨팅 서비스에서 실행되는 애플리케이션  
Amazon EC2, Amazon ECS 또는 AWS Lambda와 같은 AWS 컴퓨팅 서비스에서 실행되는 애플리케이션 코드를 사용하여 AWS 계정에 액세스할 수 있도록 이 액세스 키를 사용할 것입니다.

☒ 위의 권장 사항을 이해했으며 액세스 키 생성을 계속하려고 합니다.

취소 다음

- AWS 컴퓨팅 서비스에서 실행되는 애플리케이션 선택
- 동의 체크 후 다음

## 실습3-7 설명 태그 설정

설명 태그 설정 - 선택 사항 정보

이 액세스 키에 대한 설명은 이 사용자에게 태그로 연결되고, 액세스 키와 함께 표시됩니다.

**설명 태그 값**  
이 액세스 키의 용도와 사용 위치를 설명합니다. 좋은 설명은 나중에 이 액세스 키를 자신있게 교체하는 데 유용합니다.

최대 256자까지 가능합니다. 허용되는 문자는 문자, 숫자, UTF-8로 표현할 수 있는 공백 및 `_` `:` `/` `=` `+` `-` `@`입니다.

취소 이전 액세스 키 만들기

- 설명 태그 값 비워두고 액세스 키 만들기 클릭

## 실습3-8 액세스 키 확인 및 완료

액세스 키 검색 정보

액세스 키

분실하거나 잊어버린 비밀 액세스 키는 검색할 수 없습니다. 대신 새 액세스 키를 생성하고 이전 키를 비활성화합니다.

액세스 키	비밀 액세스 키
<div>AKIA&gt;NN4YLHZO73EKL7Z</div>	<div>***** 표시</div>

액세스 키 모범 사례

- 액세스 키를 일반 텍스트, 코드 리포지토리 또는 코드로 저장해서는 안됩니다.
- 더 이상 필요 없는 경우 액세스 키를 비활성화하거나 삭제합니다.
- 최소 권한을 활성화합니다.
- 액세스 키를 정기적으로 교체합니다.

액세스 키 관리에 대한 자세한 내용은 [AWS 액세스 키 관리 모범 사례](#)를 참조하세요.

.csv 파일 다운로드

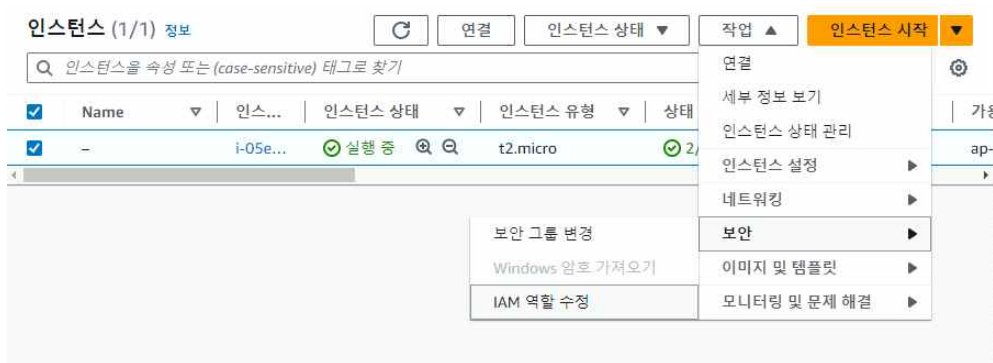
완료

- 액세스 키, 비밀 액세스 키 보관 후 완료

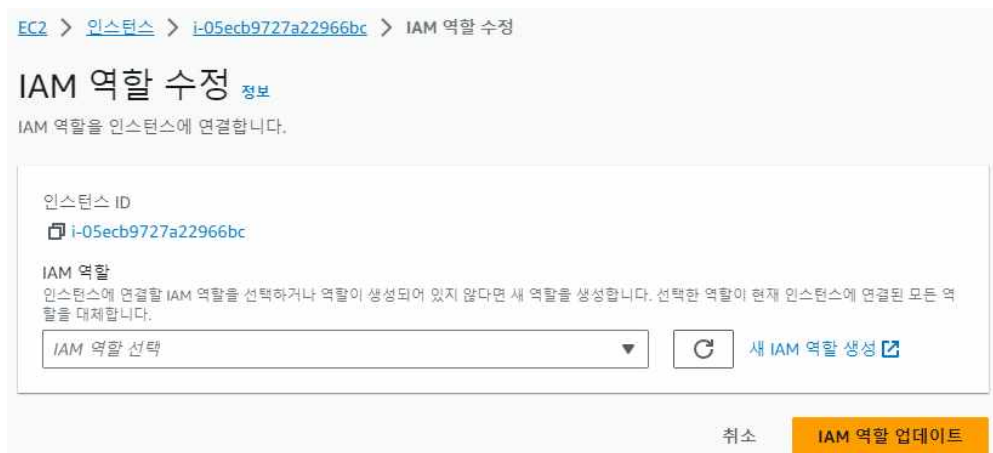




## 실습4-3 EC2 &gt; 작업 &gt; 보안 &gt; IAM 역할 수정



## 실습4-4 IAM 역할 선택 및 IAM 역할 업데이트



- 사용자계정-iam-for-ec2 선택 후 IAM 역할 업데이트 클릭

## 실습4-5 CodeDeploy Agent 설치(EC2 SSH 접속)

```
# 패키지 업데이트
$ sudo yum update

# 의존 패키지 루비 설치
$ sudo yum install -y ruby

# CodeDeploy Agent 패키지 파일 다운로드
$ wget https://aws-codedeploy-ap-northeast-2.s3.amazonaws.com/latest/install

# 실행 권한 부여
$ chmod +x ./install

# Agent 설치
$ sudo ./install auto

# Agent 실행 확인
$ sudo service codedeploy-agent start && sudo service codedeploy-agent status

잠시 후... 아래 결과 출력되면 끝
Starting codedeploy-agent:The AWS CodeDeploy agent is running as PID xxxxx
```

## 5) AWS S3 설정

## 실습5-1 AWS 전체 서비스 &gt; S3 &gt; 버킷 &gt; 버킷 만들기



**버킷 정보**

버킷은 S3에 저장되는 데이터의 컨테이너입니다. [자세히 알아보기](#)

이름으로 버킷 찾기

버킷 없음  
버킷이 없습니다.  
[버킷 만들기](#)

## 실습5-2 버킷 이름 입력 후 버킷 생성



**일반 구성**

버킷 이름

chhak2023-s3-bucket-github-actions

버킷 이름은 글로벌 네임스페이스 내에서 고유해야 하며 버킷 이름 지정 규칙을 따라야 합니다. [버킷 이름 지정 규칙 보기](#)

AWS 리전

아시아 태평양(서울) ap-northeast-2

기존 버킷에서 설정 복사 - 선택 사항  
다음 구성의 버킷 설정만 복사됩니다.

[버킷 선택](#)

- 버킷 이름 : 사용자계정-s3-bucket-github-actions
- 나머지 구성 기본값 설정 후 버킷 생성 클릭

## 6) AWS CodeDeploy 설정

실습6-1 AWS 전체 서비스 &gt; CodeDeploy &gt; 배포 &gt; 애플리케이션 생성

[개발자 도구](#) > [CodeDeploy](#) > 애플리케이션

애플리케이션

알림 세부 정보 보기 애플리케이션 배포 애플리케이션 생성

애플리케이션 이름 | 컴퓨팅 플랫폼 | 생성됨

결과 없음  
표시할 결과가 없습니다.

실습6-2 애플리케이션 구성

## 애플리케이션 생성

애플리케이션 구성

애플리케이션 이름  
애플리케이션 이름을 입력합니다.  
chhak2023-codedeploy-app  
100자 이내

컴퓨팅 플랫폼  
컴퓨팅 플랫폼 선택  
EC2/온프레미스

태그  
태그 추가

취소 애플리케이션 생성

- 애플리케이션 이름 : 사용자계정-codedeploy-app 입력
- 컴퓨팅 플랫폼 : EC2/온프레미스 선택
- 애플리케이션 생성 클릭

## 실습6-3 CodeDeploy &gt; 배포 &gt; 애플리케이션 &gt; 배포그룹생성

배포 배포 그룹 개정

배포 그룹

세부 정보 보기 편집 배포 그룹 생성

Q

< 1 > ⚙

이름	상태	최근 시도한 배포	최근 성공한 배포	트리거 개수
<p>배포 그룹 없음</p> <p>CodeDeploy를 사용하여 애플리케이션을 배포하기 전에 배포 그룹을 생성해야 합니다.</p> <p>배포 그룹 생성</p>				

## 실습6-4 배포그룹 생성

- 배포 그룹 이름 입력 : chhak2023-codedeploy-deploy-group
- 서비스 역할 입력 : chhak2023-iam-for-codedeploy (IAM 역할 입력한 값 선택)
- 배포 유형 : 현재 위치 선택
- 환경구성 : Amazon EC2 인스턴스 선택
- 태그 그룹 키 : chhak2023-ec2-tag-key 선택(처음 생성한 EC2 Tag)
- 배포 설정 : CodeDeployDefault.AllAtOnce 선택
- 로드 밸런서 : 로드 밸런싱 활성화 체크 해제
- 배포 그룹 생성 클릭