
Open source Software
오픈소스 소프트웨어
21. [실습] 임시 저장 stash

학습 개요

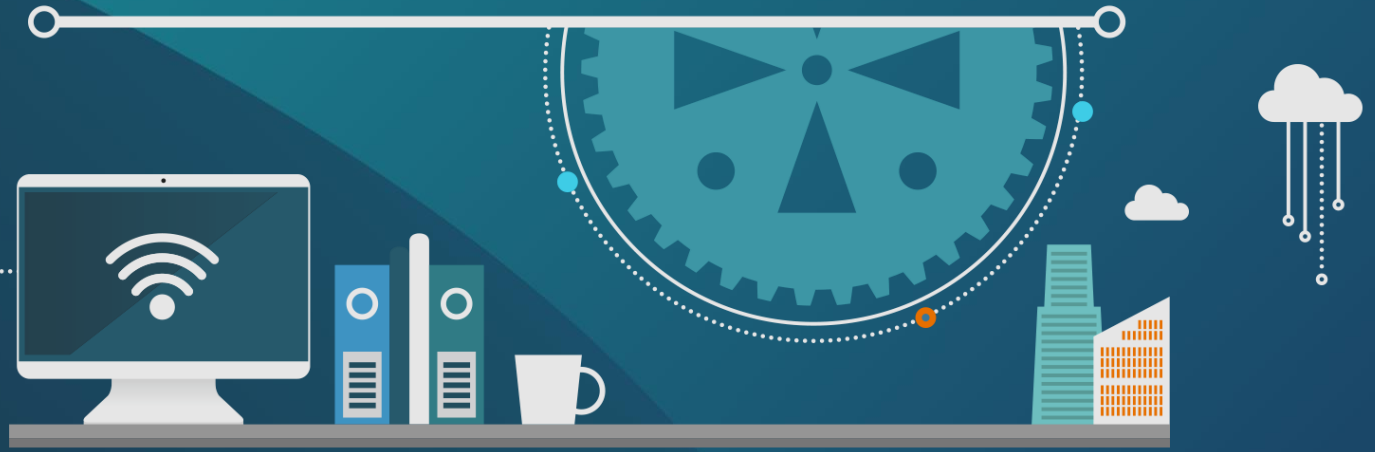
1. stash, stash save
2. stash apply, stash pop

학습 목표

1. 임시 저장을 수행할 수 있다.
2. 임시 저장 목록을 보고 삭제할 수 있다.
3. 임시 저장 목록에서 특정한 목록의 내용으로 작업 디렉토리와 인덱스를 복원할 수 있다.

LESSON 01

임시 저장 기초



저장소 생성 후 커밋

저장소 gstash

```
파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]
$ git init gstash
Initialized empty Git repository in C:/[smart git]/gstash/.git/

파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]
$ cd gstash

파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]/gstash (main)
$ echo 111 > f

파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]/gstash (main)
$ git add f

파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]/gstash (main)
$ git stash
You do not have the initial commit yet

파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]/gstash (main)
$ git commit -m A
[main (root-commit) c56bd01] A
 1 file changed, 1 insertion(+)
 create mode 100644 f

파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]/gstash (main)
$ git lgag1
* c56bd01 (HEAD -> main) A
```

$ git config --global alias.lgag1 'log --all --graph --oneline'

3 영역을 모두 다르게

파일 f가 3 영역이 모두 다르고 추적되지 않은 파일 g도 생성

```
파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]/gstash (main)
$ echo 222 >> f

$ git add f

$ echo aaa > g

$ echo 333 >> f

$ git status -s
MM f
?? g

$ git diff
diff --git a/f b/f
index a30a52a..641d574 100644
--- a/f
+++ b/f
@@ -1,2 +1,3 @@
 111
 222
+333
```

```
$ git diff --staged
diff --git a/f b/f
index 58c9bdf..a30a52a 100644
--- a/f
+++ b/f
@@ -1 +1,2 @@
 111
+222

$ git diff HEAD
diff --git a/f b/f
index 58c9bdf..641d574 100644
--- a/f
+++ b/f
@@ -1 +1,3 @@
 111
+222
+333
```

작업 디렉토리	스테이징 영역	깃 저장소
111	111	111
222	222	
333		

임시 저장에 저장

옵션 -u가 없으면 untracked file은 그대로 남음

```
$ git lgag1
* c56bd01 (HEAD -> main) A

$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   f

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:   f

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

$ git stash
Saved working directory and index state WIP on main: c56bd01 A

$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        g

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

$ git stash list
stash@{0}: WIP on main: c56bd01 A
```

작업 디렉토리	스테이징 영역	깃 저장소
111	111	111

stash@{0}

작업 디렉토리	스테이징 영역
111 222 333	111 222

임시 저장을 다시 작업 디렉토리에 복사

저장에서 삭제되지 않고 복사

- 스테이징 영역은 복원이 안 됨
  - $ git stash apply

작업 디렉토리	스테이징 영역	깃 저장소
111 222 333	111	111

```
$ git stash apply
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:   f
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        g
no changes added to commit (use "git add" and/or "git commit -a")
$ cat f
111
222
333
$ git diff
diff --git a/f b/f
index 58c9bdf..641d574 100644
--- a/f
+++ b/f
@@ -1 +1,3 @@
 111
+222
+333
```

stash@{0}

작업 디렉토리	스테이징 영역
111 222 333	111 222

다시 파일 수정한 후 임시 저장 적용

스테이징 영역도 함께 복원

- 다시 처음 상태가 됨

```
$ echo 111 > f
$ git stash list
stash@{0}: WIP on main: c56bd01 A
$ git stash apply --index
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   f
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:   f
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        g
$ git status -s
MM f
?? g
$ git diff
diff --git a/f b/f
index a30a52a..641d574 100644
--- a/f
+++ b/f
@@ -1,2 +1,3 @@
 111
 222
+333
```

```
$ git diff --staged
diff --git a/f b/f
index 58c9bdf..a30a52a 100644
--- a/f
+++ b/f
@@ -1 +1,2 @@
 111
+222
$ git diff HEAD
diff --git a/f b/f
index 58c9bdf..641d574 100644
--- a/f
+++ b/f
@@ -1 +1,3 @@
 111
+222
+333
$ git stash list
stash@{0}: WIP on main: c56bd01 A
```

작업 디렉토리	스테이징 영역	깃 저장소
111	111	111
222	222	
333		

modified (작업 디렉토리 ← 스테이징 영역)

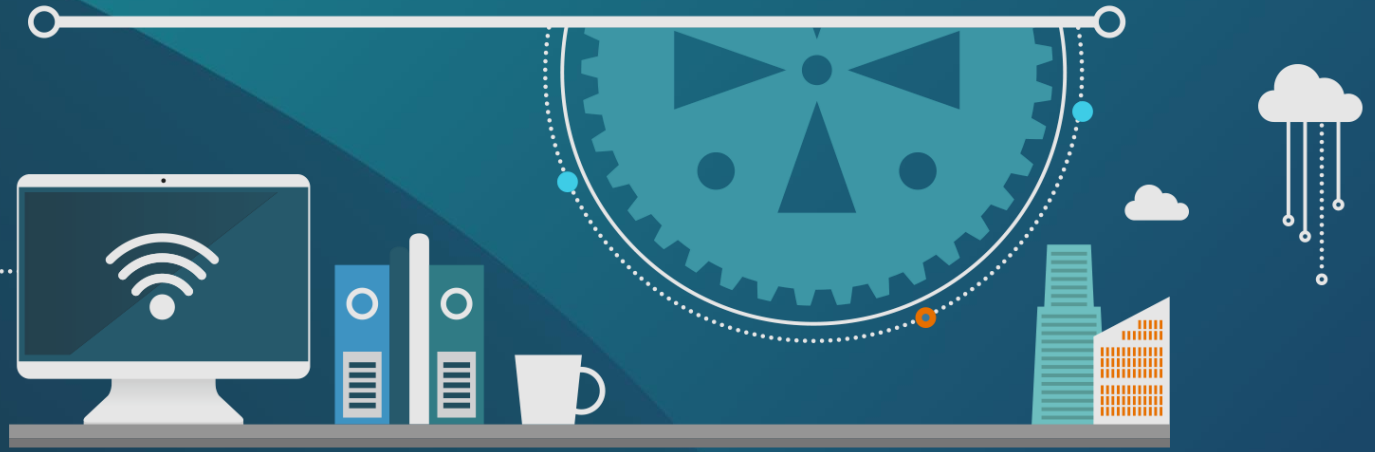
modified (스테이징 영역 ← 깃 저장소)

stash@{0}

작업 디렉토리	스테이징 영역
111 222 333	111 222

LESSON 02

임시 저장 관련 다양한 명령



2 임시 저장 관련 다양한 명령

Save로 저장

--keep-index, -k

- 스테이징 영역은 SA 제외하고, 작업 디렉토리더만 저장

작업 디렉토리	스테이징 영역	깃 저장소
111	111	111
222	222	

--include-untracked, -u

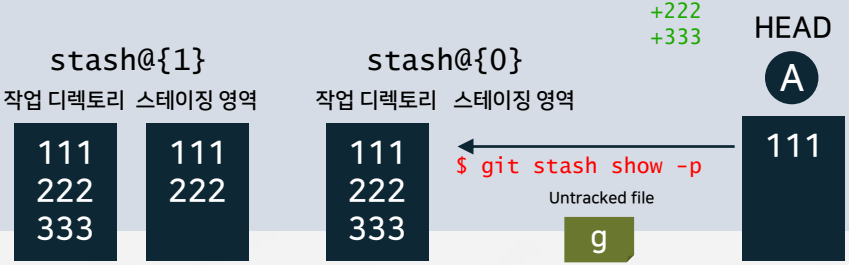
- Untracked 파일도 포함해 저장

```
$ git stash save 'stash -u -k' -u -k
Saved working directory and index state On main: stash -u -k

$ git stash list
stash@{0}: On main: stash -u -k
stash@{1}: WIP on main: c56bd01 A

$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   f

$ cat f
111
222

$ git diff

$ ls
f
```

스테이징 영역의 파일은 임시 저장하지 않으므로 그대로 남음

```
$ git stash show
 f | 2 ++
 1 file changed, 2 insertions(+)

$ git stash show -p
diff --git a/f b/f
index 58c9bdf..641d574 100644
--- a/f
+++ b/f
@@ -1 +1,3 @@
 111
+222
+333
```

stash 생성 시 커밋과 지정한 stash(최신)와의 차이
a/hello.txt: 커밋 내용
b/hello.txt: 임시저장 내용



2 임시 저장 관련 다양한 명령

임시 저장 복원 시 충돌 발생

- 임시 저장 목록에서 제거되지 않음
- 충돌 시 상태 표시
  - Both modified

```
$ git stash list
stash@{0}: On main: stash -u -k
stash@{1}: WIP on main: c56bd01 A

$ git stash pop
Auto-merging f
CONFLICT (content): Merge conflict in f
On branch main
Unmerged paths:
  (use "git restore --staged <file>..." to unstage)
  (use "git add <file>..." to mark resolution)
        both modified:   f

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

no changes added to commit (use "git add" and/or "git commit -a")
The stash entry is kept in case you need it again.

$ git status -s
UU f
?? g
```

```
$ cat f
111
222
<<<<<<< Updated upstream
=======
333
>>>>>>> Stashed changes
```

Updated upstream:
현재 작업 디렉토리 내용
Stashed changed:
임시 저장으로부터 복원될 내용

2 임시 저장 관련 다양한 명령

충돌 해결

파일 수정 후 add

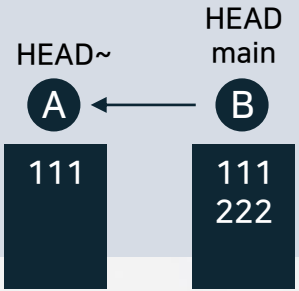
```
$ git status
On branch main
Unmerged paths:
  (use "git restore --staged <file>..." to unstage)
  (use "git add <file>..." to mark resolution)
        both modified:   f

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

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

```
$ code f

$ git add f
```


```
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   f

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

```
$ echo 333 >> f

$ git commit -m B
[main d8e0ca0] B
 1 file changed, 1 insertion(+)

$ git log --oneline
d8e0ca0 (HEAD -> main) B
c56bd01 A
```

작업 디렉토리	스테이징 영역	깃 저장소
111 222 333	111 222	111



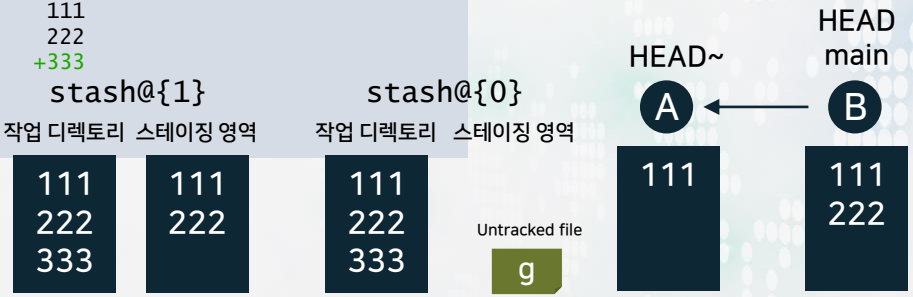
2 임시 저장 관련 다양한 명령

현재 상태

```
$ git log --oneline
d8e0ca0 (HEAD -> main) B
c56bd01 A

$ git stash list
stash@{0}: On main: stash -u -k
stash@{1}: WIP on main: c56bd01 A

$ 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:   f

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

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

$ cat f
```

```
111
222
333

$ git diff
diff --git a/f b/f
index a30a52a..641d574 100644
--- a/f
+++ b/f
@@ -1,2 +1,3 @@
 111
 222
+333

$ git diff --staged

$ git diff HEAD
diff --git a/f b/f
index a30a52a..641d574 100644
--- a/f
+++ b/f
@@ -1,2 +1,3 @@
 111
 222
+333
```

작업 디렉토리	스테이징 영역	깃 저장소
111 222 333	111 222	111 222



2 임시 저장 관련 다양한 명령

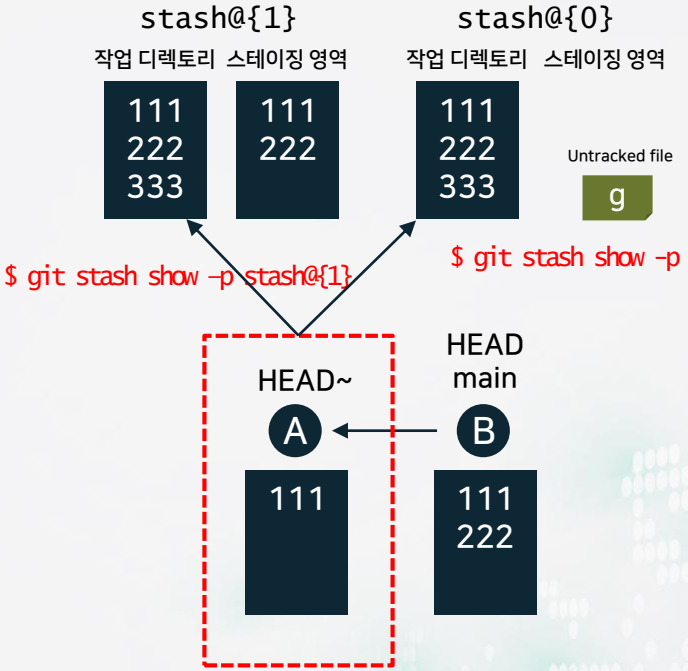
커밋과 임시 저장과의 비교

$ git stash show -p

```
파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]/gstash (main)
$ git stash show -p
diff --git a/f b/f
index 58c9bdf..641d574 100644
--- a/f
+++ b/f
@@ -1 +1,3 @@
 111
+222
+333

파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]/gstash (main)
$ git stash list
stash@{0}: On main: stash -u -k
stash@{1}: WIP on main: c56bd01 A

파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]/gstash (main)
$ git stash show -p stash@{1}
diff --git a/f b/f
index 58c9bdf..641d574 100644
--- a/f
+++ b/f
@@ -1 +1,3 @@
 111
+222
+333
```

stash 생성 시 커밋과 지정한 stash(최신)와의 차이
a/hello.txt: 커밋 내용
b/hello.txt: 임시저장 내용

stash 생성 시 커밋과 지정한 stash(stash@{1})와의 차이
a/hello.txt: 커밋 내용
b/hello.txt: 임시저장 내용



2 임시 저장 관련 다양한 명령

옵션 -m으로 메시지 저장

$ git stash show -p

```
$ 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:   f

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

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

$ git stash -m 'stash -u' -u
Saved working directory and index state On main: stash -u

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

$ git stash list
stash@{0}: On main: stash -u
stash@{1}: On main: stash -u -k
stash@{2}: WIP on main: c56bd01 A
```

2 임시 저장 관련 다양한 명령

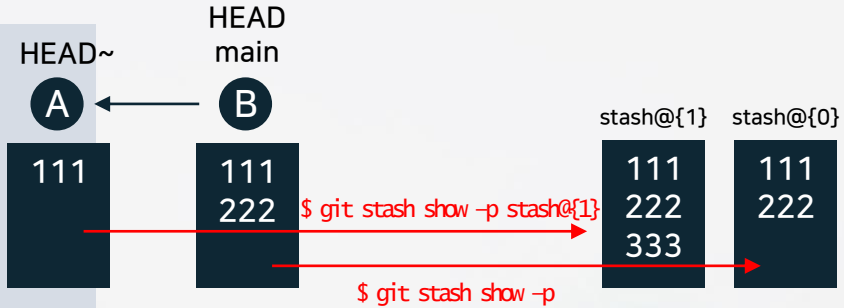
커밋과 임시 저장과의 비교

$ git stash show -p

```
파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]/gstash (main)
$ git stash show -p
diff --git a/f b/f
index a30a52a..641d574 100644
--- a/f
+++ b/f
@@ -1,2 +1,3 @@
 111
 222
+333

파이썬@DESKTOP-8TN3J1L MINGW64 /c/[smart git]/gstash (main)
$ git stash show -p stash@{1}
diff --git a/f b/f
index 58c9bdf..641d574 100644
--- a/f
+++ b/f
@@ -1 +1,3 @@
 111
+222
+333
```

stash 생성 시 커밋과 지정한 stash(최신)와의 차이
a/hello.txt: 커밋 내용
b/hello.txt: 임시저장 내용

stash 생성 시 커밋과 지정한 stash(stash@{1})와의 차이
a/hello.txt: 커밋 내용
b/hello.txt: 임시저장 내용



2 임시 저장 관련 다양한 명령

임시 저장 목록 삭제

- drop
  - 최신이나 지정한 임시 저장 삭제
- clear
  - 모든 임시 저장 삭제

```
$ git stash list
stash@{0}: On main: stash -u
stash@{1}: On main: stash -u -k
stash@{2}: WIP on main: c56bd01 A

$ git stash drop stash@{1}
Dropped stash@{1} (aedfded4177658e18dc55f8b047f2fad12945a73)

$ git stash list
stash@{0}: On main: stash -u
stash@{1}: WIP on main: c56bd01 A

$ git stash drop
Dropped refs/stash@{0} (e9dc3e557dbeece9a9eeabab982d5ccda1db650a)

$ git stash list
stash@{0}: WIP on main: c56bd01 A

$ git stash clear

$ git stash list
```

2 임시 저장 관련 다양한 명령

Untracked 파일 삭제

새로 생성한 파일 삭제

```
$ ls
f

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

$ touch g h i

$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        g
        h
        i

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

$ git clean
fatal: clean.requireForce defaults to true and neither -i, -n,
nor -f given; refusing to clean
```

```
$ git clean -i
Would remove the following items:
  g  h  i
*** Commands ***
    1: clean                2: filter by pattern    3: select
by numbers
    4: ask each             5: quit                 6: help
What now> 4
Remove g [y/N]? n
Remove h [y/N]? n
Remove i [y/N]? y
Removing i

$ ls
f  g  h

$ git clean -f
Removing g
Removing h

$ ls
f
```

번호와 첫 문자
모두 입력 가능

Summary

작업 디렉토리와 스테이징 영역을 숨김(stash)에 저장하고 작업 폴더를 정리

- $ git stash
- $ git stash –m ‘메시지’
- $ git stash save
- $ git stash save ‘메시지’

최근 또는 지정된 임시 저장소 내용을 가져와 반영하고 삭제

- $ git stash pop
- $ git stash pop stash@{n}

Summary

» 최근 또는 지정된 임시 저장소 내용을 가져와 반영, 작업 디렉토리에 반영, stash 목록은 그대로

- $ git stash apply
- $ git stash apply stash@{n}

» 최근 또는 지정된 임시저장소 내용을 가져와 반영, 작업 디렉토리와 스테이징 영역도 반영, stash 목록은 그대로

- $ git stash apply --index
- $ git stash apply --index stash@{n}