【Git】還原相關

以下是關於 Git 還原（恢復）功能的完整說明

Git 還原功能總表

指令	還原範圍	是否會刪除暫存區	是否會刪除未追蹤檔案	適用情境
`git checkout .`	工作目錄	否	否	還原已追蹤檔案
`git restore .`	工作目錄	否	否	推薦用法，等同於 checkout .
`git restore --staged`	暫存區	是	否	還原 `git add` 的檔案
`git reset`	暫存區	是	否	退回 staging 區
`git reset --hard HEAD`	工作目錄 + 暫存區	是	是（若搭配 clean）	徹底放棄所有變更（⚠危險）
`git clean -fd`	未追蹤檔案/資料夾	無關	是	清空未納入 Git 管控的檔案
`git revert <commit>`	版本歷史	無關	否	安全回復 commit（保留紀錄）

Git 還原功能完整整理與使用說明

在日常開發中，Git 提供多種指令用來還原工作目錄、暫存區，或是回復錯誤的 commit。根據不同的需求，選擇合適的還原方式可避免資料遺失或版本混亂。

---

1. `git checkout .`

還原所有「已追蹤檔案」回到最新 commit 的狀態。

- **作用範圍**：僅限工作目錄（Working Directory）
- **不會還原**：未追蹤（untracked）檔案與暫存區（Staging Area）

適用情境：

- 編輯了多個檔案，想快速放棄變更，但未進行 `git add`。

---

2. `git restore`（推薦用法）

`git restore` 是 Git 2.23+ 提供的新指令，取代過去 `checkout` 的混用狀況。

範例：

```
git restore .                # 還原所有檔案
git restore path/to/file.js # 還原單一檔案
```

- 與 `git checkout .` 行為一致，語意更清楚
- 可搭配 `--staged` 還原暫存區（如下）

---

3. `git restore --staged`

還原暫存區的內容，讓已 `git add` 的檔案退回至工作目錄狀態。

```
git restore --staged path/to/file.js
```

- **不會影響工作區的內容**
- 常用於誤將檔案加入 staging 區的情況

---

4. git reset

還原 staging 區（暫存區），讓所有已 `git add` 的檔案回到修改中狀態。

```
git reset
```

- 與 `git restore --staged` 功能類似
- 不會更動工作目錄內容

---

5. git reset --hard HEAD

將 **工作目錄與暫存區** 都還原成最近一次提交（HEAD）的狀態。

```
git reset --hard HEAD
```

- **非常危險**，會清除未提交的所有變更
- 包含 `git add` 過的內容也會消失

適用情境：

- 變更混亂，想完全清空所有未提交的東西

---

6. git clean -fd

清除所有「未被追蹤的檔案與資料夾」。

```
git clean -fd
```

- `-f`：強制執行
- `-d`：包括未追蹤的資料夾

適用情境：

- 清除 build 檔案或不小心留下的臨時檔案

---

7. git revert

對某個已提交的 commit 產生一個「反向操作的 commit」。

```
git revert <commit_hash>
```

- 保留歷史紀錄
- 安全的還原方式，尤其在多人開發時

適用情境：

- **上線後發現某次 commit 有錯誤，需在版本歷史中保留修正記錄**

修訂版本 #3
★由 treeman 建立於 6 ﷺ 2023 01:04:15
由 treeman 更新於 14 ﷺ 2025 10:21:49