

基本用語

1. ブランチ/コミットを指すポインタ。現在の位置となる。
2. HEAD を移動させ、ローカルを移動後のコミットの状態にする。
 - Untracked files に関しては削除もされず、何も起こらない。Tracked files の追加や変更に関しては破棄されるが、警告が表示される。
 - 実質的に、`git switch` と `git restore --source=<コミット>` を組み合わせたような挙動となる。checkout を行う前にコミットを行うか、stash しておくが良いだろう。
3. HEAD がブランチではなくコミットを直接指している状態。
 - この状態で新規コミットを行うと、どのブランチにも属さないコミットとなる。そのようなコミットは `git log` では表示されないため、コミットハッシュを持ってきたい場合は `git reflog` を利用すると良い。
 - ちなみに通常の HEAD がブランチを指している状態は attached HEAD 状態と呼ぶ。
4. `git switch`
 - checkout コマンドの HEAD の移動の部分だけを取り出したような挙動となる。

表示系

1. `git diff`
2. `git diff --staged`
3. `git log <ブランチA>..
<ブランチB>`
4. `git log --oneline`

復元・削除系

1. `git restore <path>`
2. `git restore --staged <path>` または `git reset HEAD <path>`
3. `git rm --cached <path>`
4. `git rm <path>`
5. 現在のブランチと HEAD の位置をずらすコマンド。
 - checkout はブランチを動かさないが、reset は一緒に動かす。なので detached HEAD 状態にはならない。
 - reset と言っても、**実態はブランチの移動**である。つまり、HEAD -> main から HEAD -> feature に reset することもできるわけで、その場合は未来旅行だ。
6. コミットとファイル。
 - それぞれで全く違う働きをする。--soft,--mixed,--hard はコミットに対して使うときだけ。
 - ファイルに対して使いたいときは `git restore` を使うほうが明確だろう。
7. --soft はインデックスとワーキングツリーを保持する。--mixed はインデックスを破棄し、ワーキングツリーを保持する。--hard はインデックスを破棄し、ワーキングツリーを指定したコミットと同じものにする。
 - ブランチの移動に際してインデックスとワーキングツリーをどうするかを指定するのが soft,mixed,hard だと考えれば良い。

ブランチ

1. `git branch <ブランチ名>`
2. `git switch -c <ブランチ名>` または `git checkout -b <ブランチ名>`
3. `git branch`
4. `git branch -d <ブランチ名>`
5. `git branch -D <ブランチ名>`

6. Orphan Branch (孤児ブランチ)
 - 通常, orphan branch とのマージはエラーになるが, 強制的にマージすることもでき, その場合は全てがコンクリート扱いとなる.
 - GitHub Pages ブランチとか, ドキュメント用ブランチを切るときなどに有効活用されたり.
7. fast-forward マージの場合, マージ先ブランチは一直線の履歴となり, マージコミットは作成されない. non-fast-forward マージは, 分岐した履歴をマージする際に新しくマージコミットが作られる.
 - 当然ながら, ff マージはマージ先ブランチがマージ元ブランチの親になっている場合しか使えない.
8. `git merge --abort`
9. `git rebase <コミット>`
 - これを使えば全部のマージを ff にすることもできる.
10. `-i`
 - 例えば, Feature ブランチで雑にコミットしていた履歴を整えつつ, main ブランチに rebase するときなどに使える.
 - 粒度高->低にするのはできるけど, 逆は難しいのでやっぱりこまめにコミットしておこう.

リモートリポジトリ

1. `git remote add <名前> <URL>`
 - `git clone` した際のリモート名は, デフォルトで origin と付けられる.
2. `git remote set-url <名前> <URL>`
3. `git branch -vv`
4. `git branch -r`
5. `git fetch`
6. • `git log main..origin/main`
 - `git log origin/main..main`
7. `git merge origin/main` または `git rebase origin/main`
8. `git pull`

その他

1. `git stash`
 - いろいろできる. untracked files にも使えて, 一時的な退避領域として便利.
2. `git commit --amend`
3. `git cherry-pick <コミット>`
4. `git revert <コミット>`
5. `git reflog`