

Git操作メモ - 新人エンジニア向け完全ガイド

目次

1. `git pull`とローカル変更の競合を `stash` で解決
2. ローカル変更を破棄して `git pull` を成功させる
3. 未追跡ファイルを削除して `git pull` を成功させる

記事①： `git pull` 時のローカル変更と `stash` の活用

背景

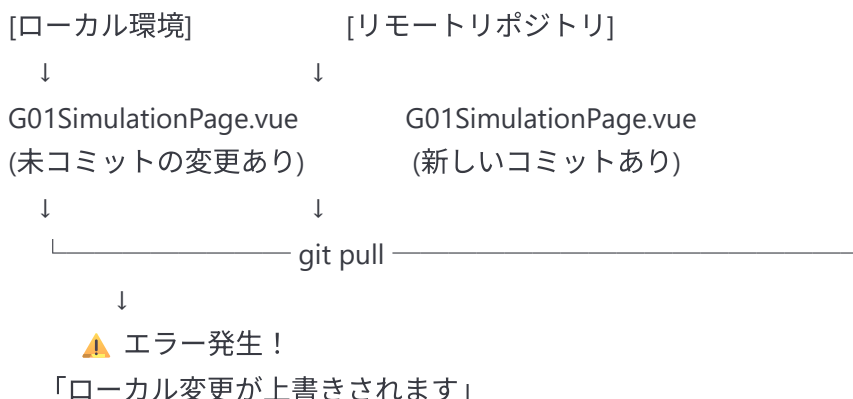
Gitで `git pull` を実行した際、ローカルで**未コミットの変更**があると、リモートの変更と衝突する可能性があるため、Gitはマージを中断します。

！ 発生するエラー例

```
shell

error: Your local changes to the following files would be overwritten by merge:
  simulation/src/pages/G01SimulationPage.vue
Please commit your changes or stash them before you merge.
Aborting
```

状況の図解



✅ 解決方法： `stash` を使った一時退避

`stash` は「一時的な保管庫」のようなもので、作業中の変更を安全に退避できます。

作業フロー

1. git stash → ローカル変更を退避
2. git pull → リモートの変更を取得
3. git stash pop → 退避した変更を復元

実行コマンド

ステップ1：ローカル変更を退避

```
shell  
  
git stash
```

実行後の状態：

```
[stash領域]  
|—— あなたの未コミット変更 (保管中)  
  
[ワーキングディレクトリ]  
|—— クリーンな状態 ✨
```

ステップ2：リモートの変更を取得

```
shell  
  
git pull origin develop
```

ステップ3：退避した変更を復元

```
shell  
  
git stash pop
```

復元後の状態：

```
[ワーキングディレクトリ]  
|—— リモートから取得した最新コード  
|—— あなたの未コミット変更 (復元完了)
```

便利なstashコマンド

コマンド	説明
<code>git stash list</code>	退避した変更の一覧を表示
<code>git stash apply</code>	変更を復元するが、stash領域には残す

コマンド	説明
<code>git stash drop</code>	最新のstashを削除
<code>git stash clear</code>	すべてのstashを削除

⚠ 注意点

- `stash pop` の際にコンフリクト(競合)が発生する可能性があります
- 競合が起きた場合は、手動でファイルを編集して解決する必要があります
- `git stash list` で履歴を確認できます

🗑 記事②：ローカル変更を破棄して `git pull` を成功させる方法

✂ 背景

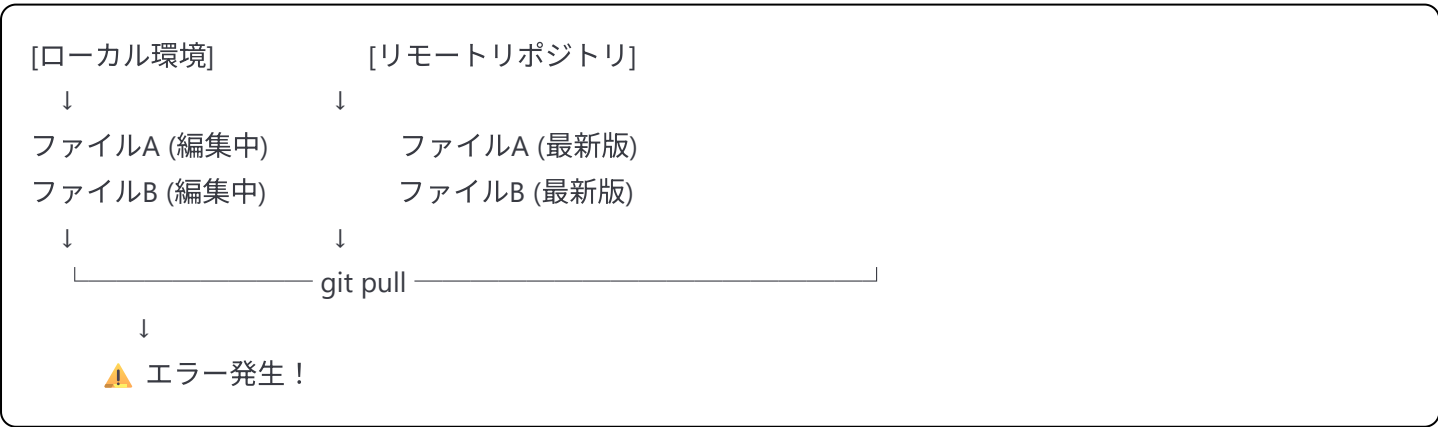
ローカルの変更が**不要な場合**や、リモートの最新状態に完全に合わせたい場合は、変更を破棄することで `git pull` を成功させることができます。

❗ 発生するエラー例

```
shell

error: Your local changes to the following files would be overwritten by merge:
<ファイル名>
Please commit your changes or stash them before you merge.
Aborting
```

📊 状況の図解



✅ 解決方法：ローカル変更を破棄する

🔄 作業フロー

1. git restore . → すべての変更を破棄
2. git pull → リモートの変更を取得



実行コマンド

ステップ1：変更をすべて破棄する

```
shell  
  
git restore .
```

実行前と実行後：

[実行前]

- ファイルA (編集) 
- ファイルB (編集) 

[実行後]

- ファイルA (最後のコミット状態に戻る) 
- ファイルB (最後のコミット状態に戻る) 


ステップ2：`git pull` を再実行する

```
shell  
  
git pull origin develop
```

特定のファイルだけ破棄したい場合

```
shell  
  
# 特定のファイルのみ破棄  
git restore <ファイル名>  
  
# 例  
git restore src/components/Header.vue
```

注意点

-  `git restore .` は未コミットの変更をすべて破棄します（元に戻せません！）
- コミット済みの変更には影響しません
- 大切な変更がある場合は、事前に別ブランチにコミットするか、stashを使用してください

✂️ 記事③：未追跡ファイルを削除して `git pull` を成功させる方法

✖️ 背景

未追跡ファイル（Gitで管理されていないファイル）がリモートの変更と競合する場合、`git pull` は中断されます。

🤔 未追跡ファイルとは？

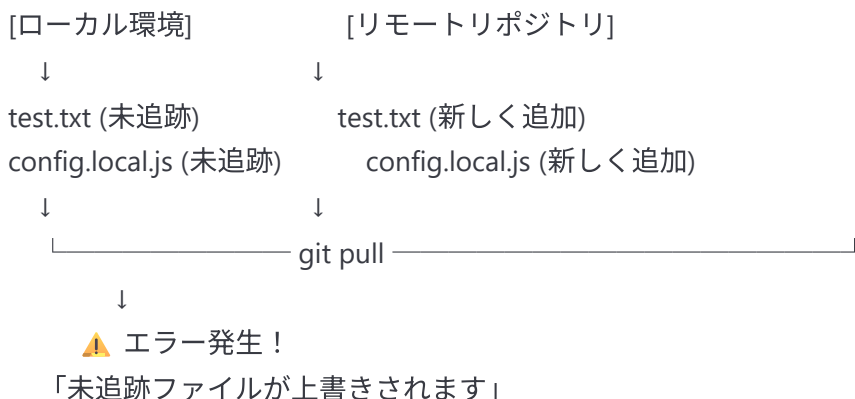
[追跡されているファイル]	[未追跡ファイル]
├── src/App.vue (✅)	├── test.txt (❌)
├── package.json (✅)	├── memo.md (❌)
└── README.md (✅)	└── debug.log (❌)
git add されている Gitが変更を監視	git add されていない Gitが認識していない

❗ 発生するエラー例

```
shell

error: The following untracked working tree files would be overwritten by merge:
    test.txt
    config.local.js
Please move or remove them before you merge.
Aborting
```

📊 状況の図解



✅ 解決方法：未追跡ファイルを削除する

🔄 作業フロー

1. git clean -n → 削除対象を確認（安全確認）
2. git clean -f → 未追跡ファイルを削除
3. git pull → リモートの変更を取得

実行コマンド

ステップ1：削除前に確認する（重要！）

```
shell  
  
git clean -n
```

実行結果の例：

```
Would remove test.txt  
Would remove config.local.js  
Would remove debug.log
```

ステップ2：未追跡ファイルを削除する

```
shell  
  
git clean -f
```

実行結果：

```
Removing test.txt  
Removing config.local.js  
Removing debug.log
```

ステップ3：`git pull` を再実行する

```
shell  
  
git pull origin master
```

より安全なオプション

コマンド	説明
<code>git clean -n</code>	削除されるファイルを表示（実際には削除しない）
<code>git clean -f</code>	未追跡ファイルを削除
<code>git clean -fd</code>	未追跡ファイル + ディレクトリを削除
<code>git clean -fx</code>	<code>.gitignore</code> に書かれているファイルも削除

コマンド	説明
<code>git clean -i</code>	対話モードで削除（1つずつ確認できる）

💡 対話モードの使い方

shell
<code>git clean -i</code>

実行すると：

Would remove the following items: test.txt config.local.js *** Commands *** 1: clean 2: filter by pattern 3: select by numbers 4: ask each 5: quit 6: help What now>

⚠ 注意点


- ⚠ `git clean -f` は元に戻せません！必要なファイルは事前にバックアップしてください
- 本番環境の設定ファイルなど、重要なファイルがないか必ず確認してください
- `-n` オプションで事前確認することを強く推奨します

📌 まとめ：どの方法を使うべき？


状況	おすすめの方法	コマンド
ローカル変更を残したい	<code>stash</code> を使う	<code>git stash</code> → <code>git pull</code> → <code>git stash pop</code>
ローカル変更は不要	<code>restore</code> で破棄	<code>git restore .</code> → <code>git pull</code>
未追跡ファイルが邪魔	<code>clean</code> で削除	<code>git clean -n</code> → <code>git clean -f</code> → <code>git pull</code>

🎓 新人エンジニアへのアドバイス

- 迷ったらまず `git status` で状況確認
- 変更を破棄する前に必ずバックアップ
- `-n` オプションで事前確認する習慣をつける
- わからないときはチームに相談！

 **作成日:** 2025年9月30日

 **対象:** 新人エンジニア

 **タグ:** Git, git pull, stash, restore, clean, トラブルシューティング