CMDを使わない

Eclipseだけで分かる GitHub入門

コミット / プッシュ / プル / 競合解決

この説明書でできるようになること

1. 自分の変更を安全にGitHubへ共有する
2. 仲間の変更を自分のEclipseへ取り込む
3. 競合しても慌てず、比較して解決する

画面例: Eclipse IDE / EGit　対象: GitHub初心者・グループ制作

まずは4つの言葉だけ

02

操作を暗記する前に「どこから、どこへ動くか」を理解します。

たとえると…

コミット = 自分の作業を「セーブ」する

プッシュ = セーブした記録をチームへ「提出」する

プル = チームの最新記録を自分へ「受け取る」

重要　**コミットしただけでは、GitHubには届きません。**

基本の順番

1. 作業前にプル
2. ファイルを編集
3. コミット
4. プッシュ

毎日の安全な作業手順

03

グループ全員がこの順番を守ると、競合が大幅に減ります。

1

PULL　作業開始前

仲間の最新変更を取得

2

EDIT　編集・動作確認

担当部分だけを変更

3

COMMIT　意味のある単位で記録

何をしたか短く書く

4

PULL　もう一度最新化

送信直前の差分を統合

5

PUSH　GitHubへ共有

成功を確認して完了

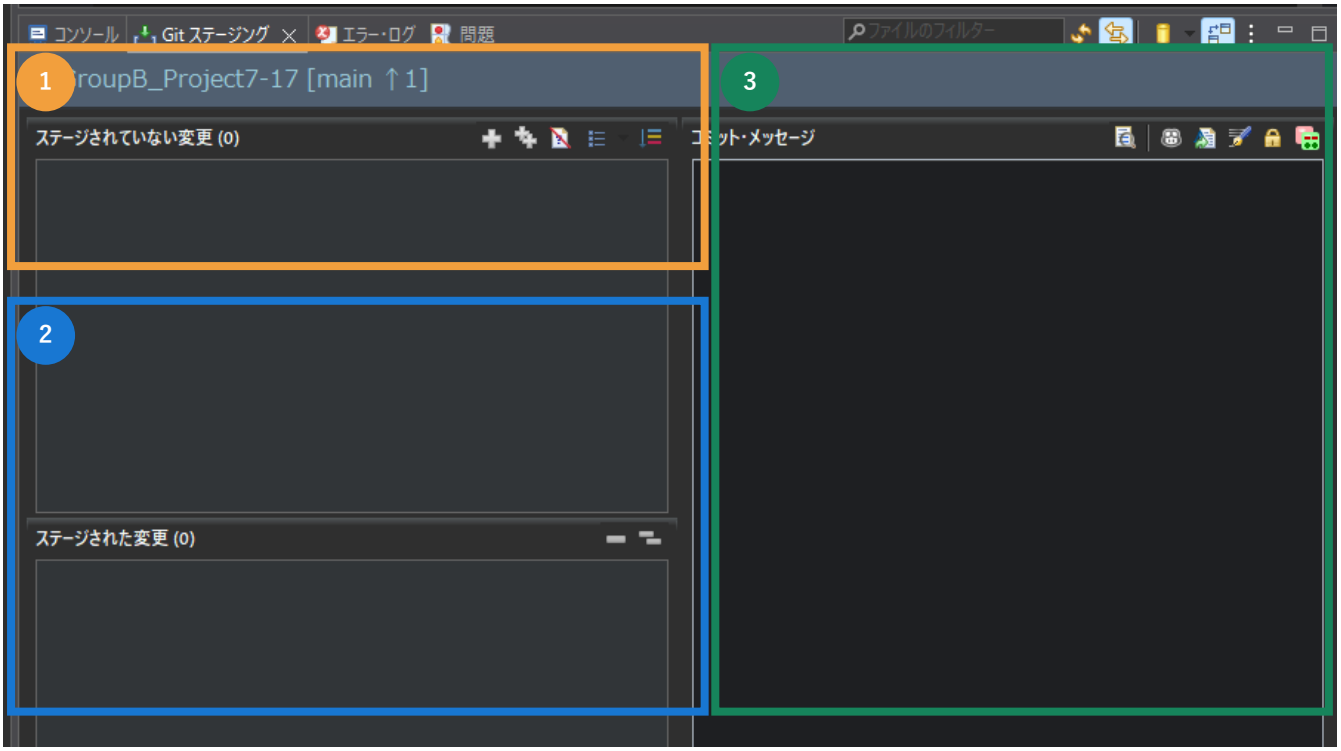
やってはいけない流れ

何日もプルせず編集 → 大量変更を一度にコミット → いきなりプッシュ

画面の見方: Gitステージング

04

いただいたEclipse画面です。下部の「Gitステージング」を主に使います。



① ステージされていない変更

編集したが、次のコミットにまだ入れていないファイル。

② ステージされた変更

次のコミットに入れると決めたファイル。

③ コミットメッセージ

「何を変更したか」をチーム向けに書く場所。

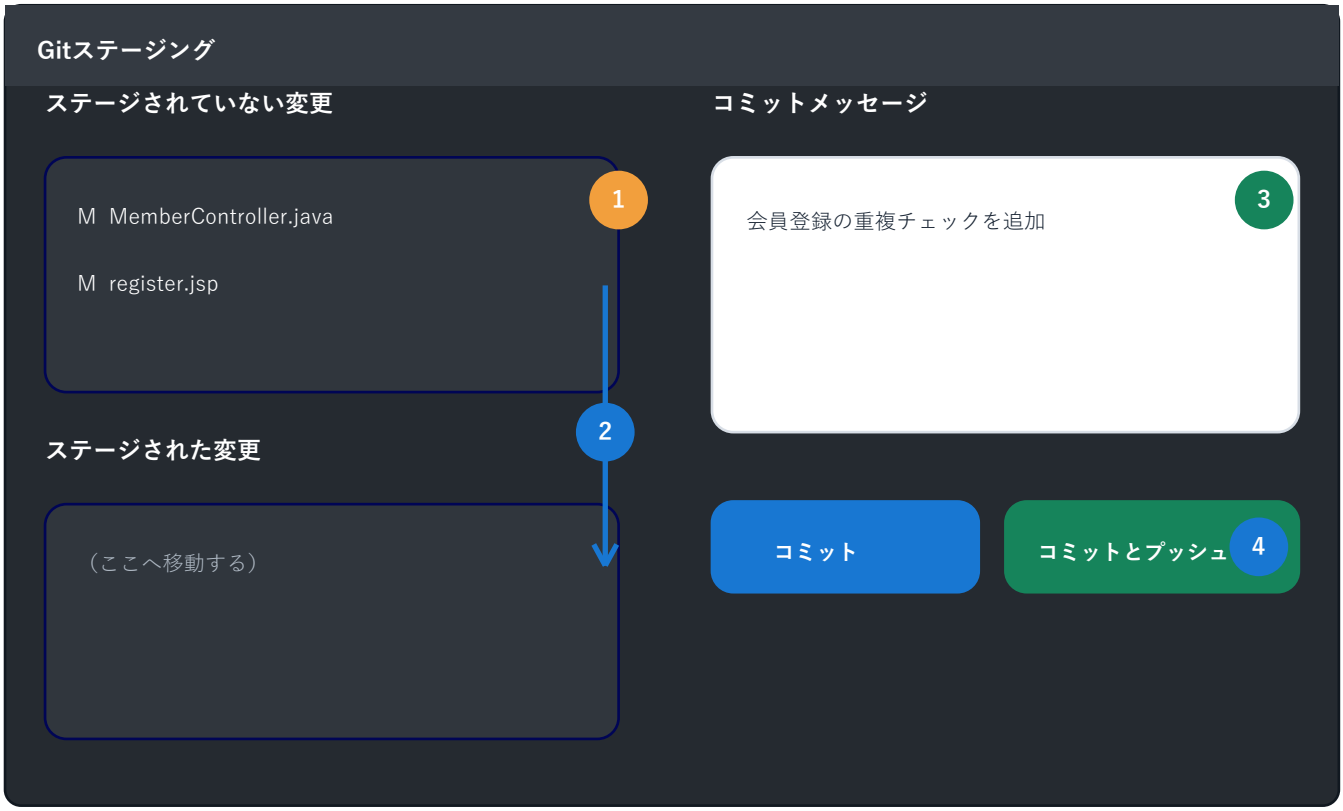
表示方法

メニュー「ウィンドウ」→「ビューの表示」→「その他」→「Git」→「Gitステージング」

コミット: 自分の変更を記録する

05

コミットは「GitHubへの送信」ではなく、自分のPC内への記録です。

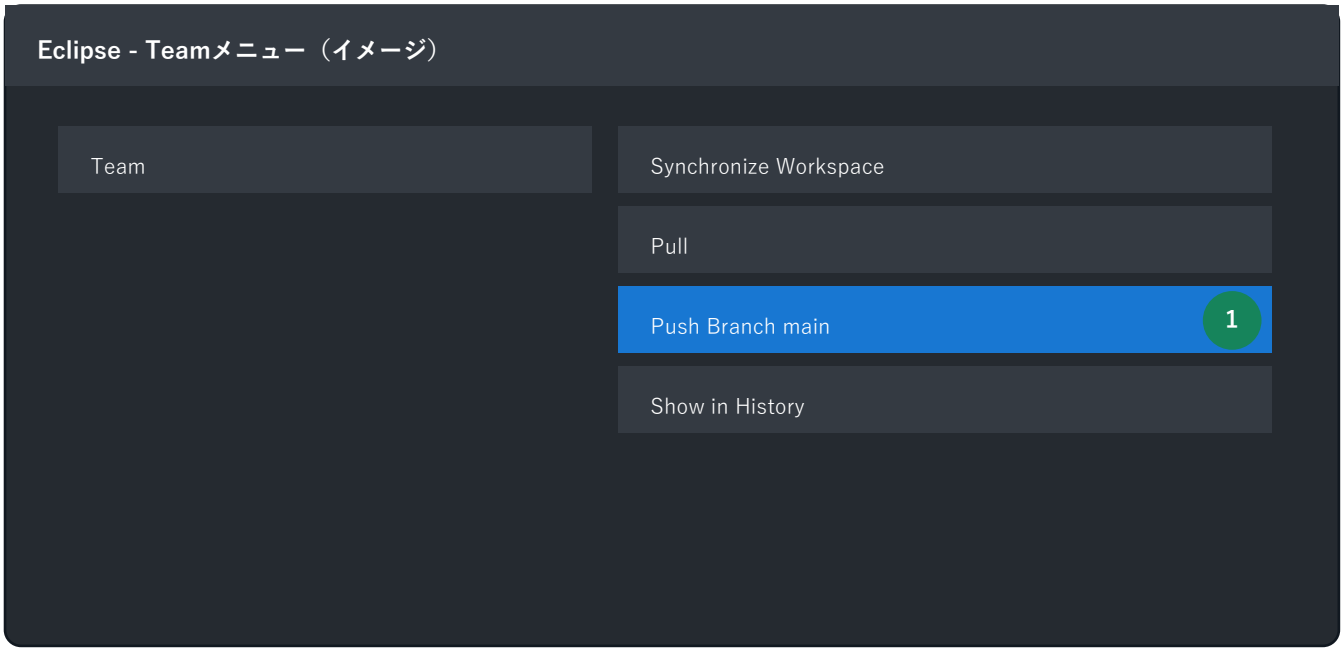


- **1. 変更内容を確認**
  ファイルをダブルクリックし、意図した変更だけか確認します。
- **2. ステージへ移動**
  「+」ボタン、またはファイルを下の欄へドラッグします。
- **3. メッセージを書く**
  例:「会員登録の入力チェックを追加」。曖昧な「修正」は避けます。
- **4. コミット**
  初めは「コミット」を選び、次ページの手順でプッシュするのがおすすめ。

プッシュ: GitHubへチーム共有する

06

コミット済みの記録を、共有リポジトリへ送信します。



操作

プロジェクトを右クリック →「Team」→「Push Branch “main”」

成功の目印

「Push Results」に rejected や error がなく、完了表示になっている。

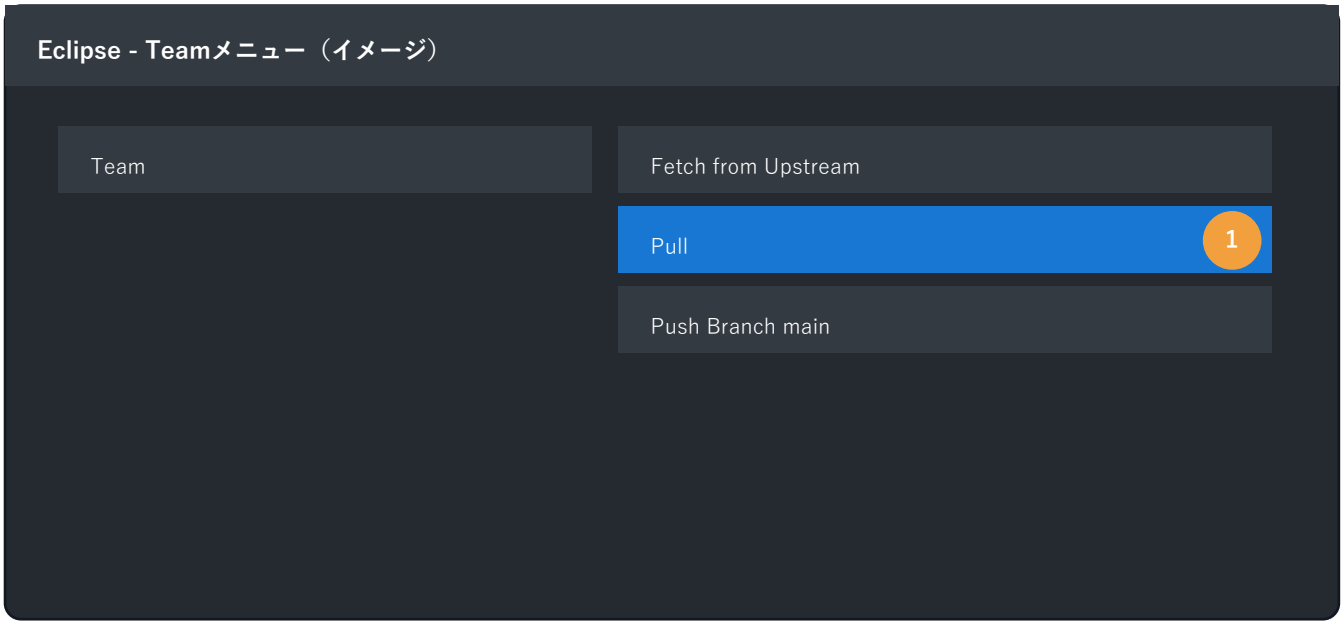
プッシュが拒否されたら

GitHub側が先に進んでいます。無理に上書きせず、いったんプルします。

プル: 仲間の最新変更を取り込む

07

作業開始前と、プッシュ直前に実行します。



操作

プロジェクトを右クリック →「Team」→「Pull」

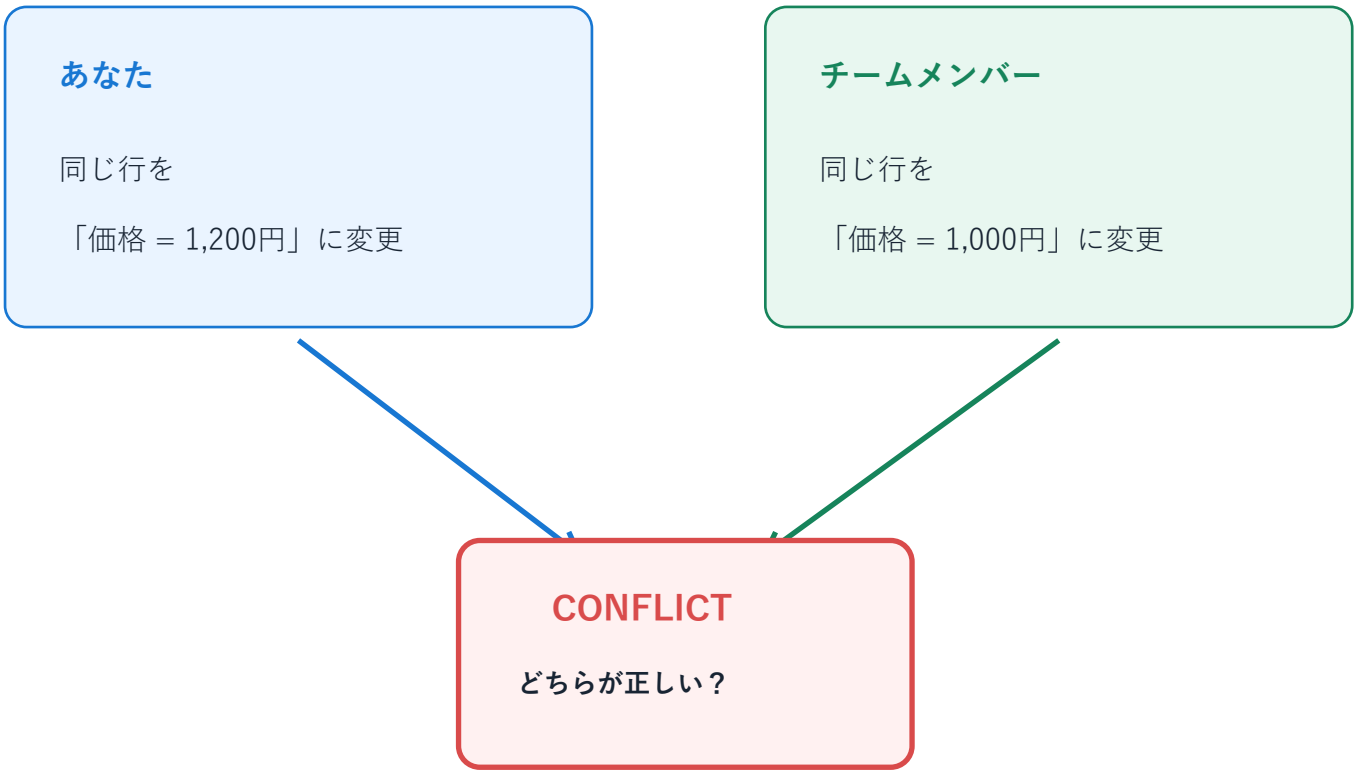
Already up-to-date	すでに最新。作業を始めてOK。
変更ファイルが増えた	仲間の更新を正常に取得。内容を確認。
CONFLICTING	競合発生。次ページ以降で解決。

プル後にpom.xmlが変わったら: 右クリック → Maven → プロジェクトの更新

競合はなぜ起こる？

08

競合は失敗ではなく「Gitが勝手に決められないので、人に確認している状態」です。



Gitができること　違うファイルや離れた行なら、自動でまとめる。

Gitができないこと　同じ場所の異なる変更から、業務上の正解を選ぶ。

競合したときの3原則

1. 慌てて削除・上書きしない　2. 相手に変更意図を確認する

3. 比較画面で必要な内容を残し、必ず動作確認する

競合解決: Eclipseの比較画面を使う

09

競合ファイルを右クリックし「Team」→「Merge Tool」を開きます。

1 **左右を比較**

どちらを残すか、または組み合わせるか決めます。

2 **結果側を編集して保存**

必要なコードだけにし、競合記号が残っていないことを確認。

3 **Team → Add to Index**

「この競合は解決済み」とGitへ知らせます。

4 **動作確認 → コミット**

競合解決内容をコミットし、その後プッシュします。

競合記号を直接編集する場合

Merge Toolが使えない場合だけ。3種類の記号をすべて削除します。

```
<<<<<<< HEAD
int price = 1200;   // 自分側
=======
int price = 1000;   // GitHub側
>>>>>>> origin/main
```

完成例（1,200円を残す場合）

```
int price = 1200;
```

確認　<<<<<<<、=======、>>>>>>> が1つも残っていない。

注意　自信がなければチームメンバーに確認してから保存する。

困ったときの早見表

エラー文を消すために操作するのではなく、原因に合う対処を選びます。

org.springframework に赤線

Maven → プロジェクトの更新。その後「プロジェクト → クリーン」。

Push rejected

先にプル。競合が出たら解決して再コミット・プッシュ。

non-fast-forward

GitHub側が先に更新済み。強制プッシュせずプル。

CONFLICTING表示

競合ファイルをMerge Toolで解決し、Add to Index。

変更が一覧に出ない

ファイルを保存。対象プロジェクト/リポジトリも確認。

間違えて変更した

未コミットならTeamの復元系操作。必要な内容を先に退避。

迷ったら止める

「Reset」「Force Push」「Delete」は、経験者と確認してから実行します。

グループ制作チェックリスト

12

この1ページを印刷して、全員で同じ運用にすると安全です。

作業を始める前

- [ ] 担当範囲をチームで確認した
- [ ] プロジェクトを右クリック → Team → Pull
- [ ] プル結果に競合やエラーがない

コミットする前

- [ ] ファイルを保存して動作確認した
- [ ] Gitステージングで差分を確認した
- [ ] 関係ないファイルをステージしていない
- [ ] 内容が分かるコミットメッセージを書いた

プッシュする前

- [ ] もう一度Pullして最新状態にした
- [ ] 競合があれば相手と確認して解決した
- [ ] コミット後の動作確認が通った

プッシュした後

- [ ] Push Resultsが成功になった
- [ ] GitHub上で自分のコミットを確認した
- [ ] チームへ完了・注意点を共有した

合言葉　**「作業前にPull、小さくCommit、共有はPush」**