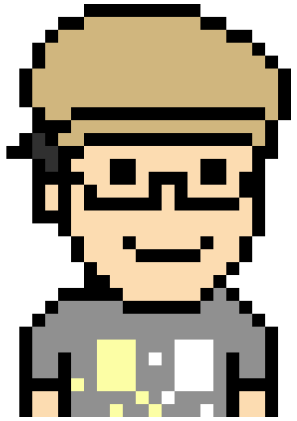
Gitとはなにか

2017-11-22 電書ラボ公開セミナー2017 第3回

小嶋智

自己紹介

- プログラマ
- テキスト処理や電子出版の周辺を漂っています



Gitとはなにか？

空気とか地面

プログラマにとっては

- 空気のように当たり前
- テキストものを管理するときはなんでも使っている
  - この資料もGit管理
- （実はGitでなくても良い）

あらためて

Gitとはなにか？

Git（ギット）とは

バージョン管理システム（VCS）の一種。

ほかにもMercurial（Hg）とかsubversion（svn）とか

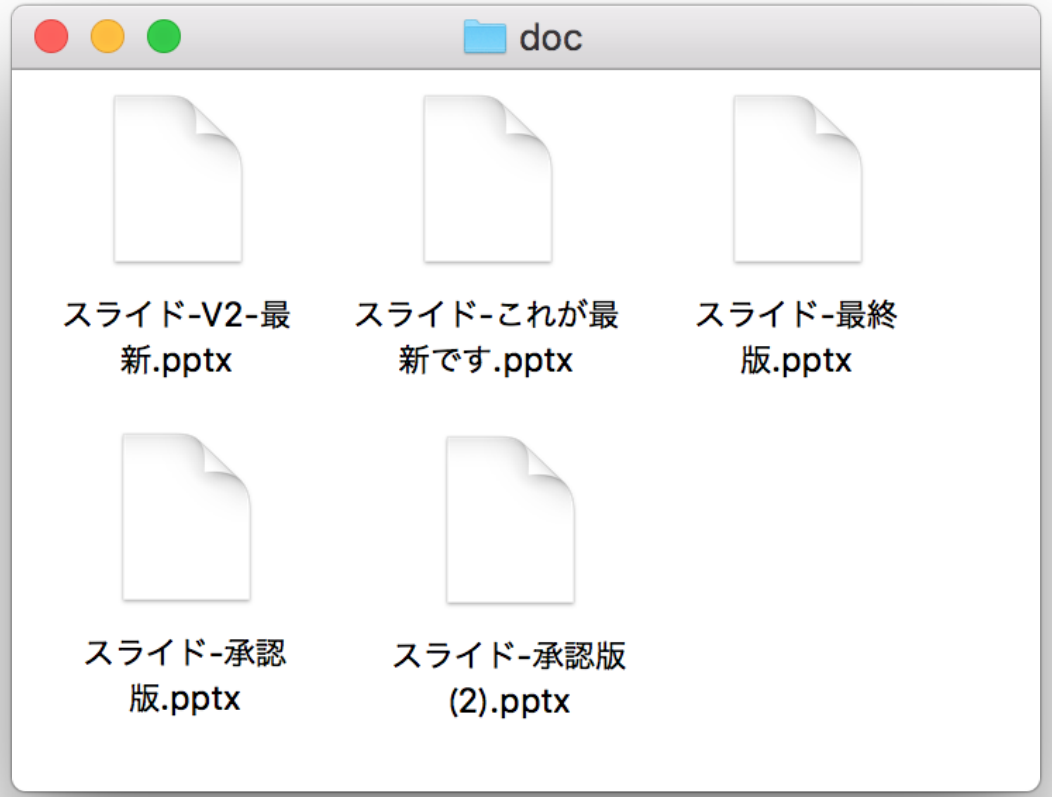
バージョン管理システムの目的は？

バージョン（版）を管理すること。

Photograph: Satoshi Kojima

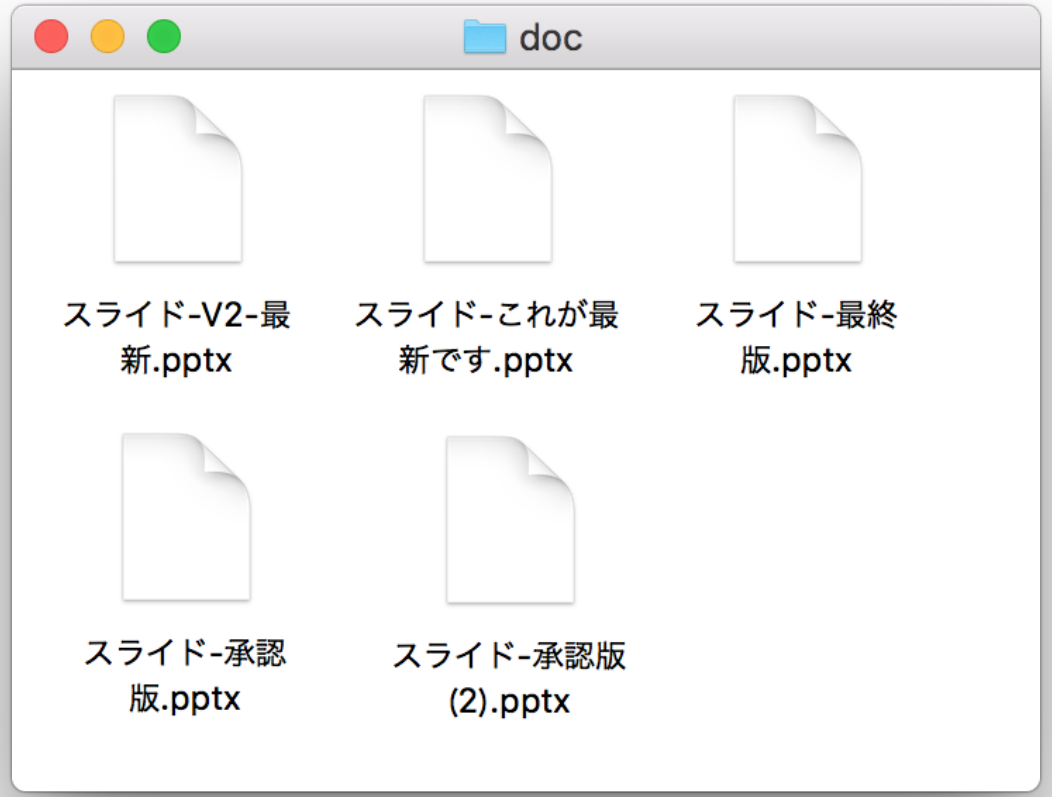
何がうれしいの？

バージョン管理レベルゼロ



最新版を修正…

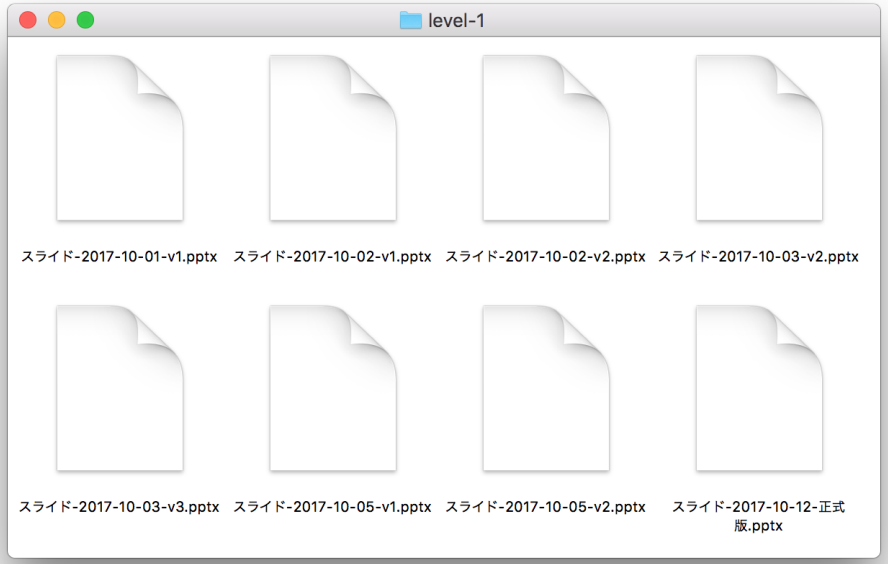
この


doc
スライド-V2-最新.pptx
スライド-これが最新です.pptx
スライド-最終版.pptx
スライド-承認版.pptx
スライド-承認版(2).pptx

どれ

バージョン管理レベル１

厳密な命名ルールの適用



バージョン管理レベル１

変更履歴

変更履歴

日付	変更点	担当
2017/10/1	初版	小嶋
2017/10/3	レビュー指摘事項修正	小嶋
2017/10/5	タイプミス修正	山田
2017/10/12	承認版発行	課長

レベル1の問題点

人間は間違う

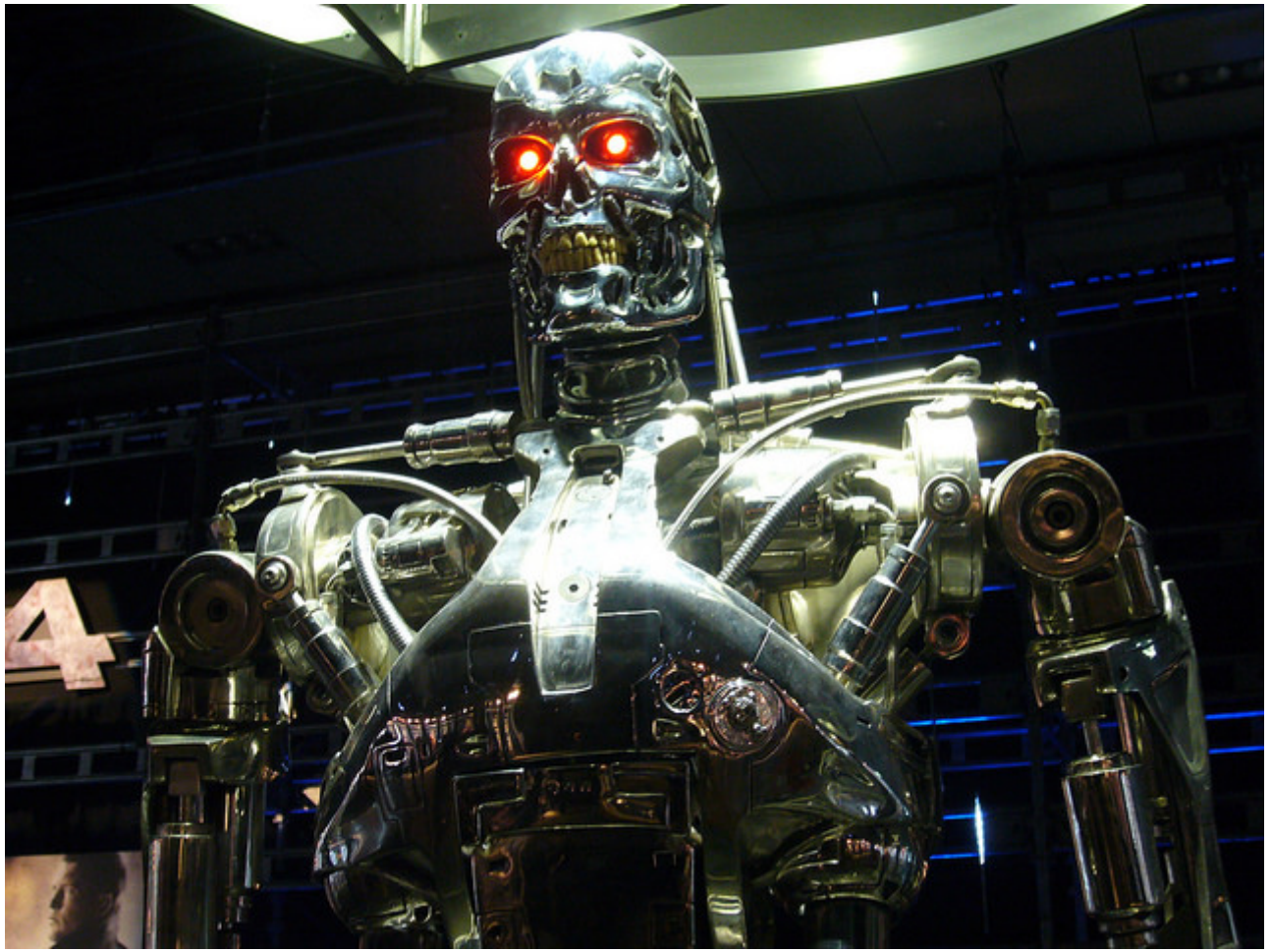
人間は怠ける

人間は忘れる

人間はこうなる

- ルール・日付・変更点を間違える
- 履歴「誤字修正」実際「ついでにこれも編集しちゃえ」
- 履歴「指摘事項修正」…どこ？ なにを変えたの？
- 履歴を書こう。…何変更したんだっけ

人間はあてにならない

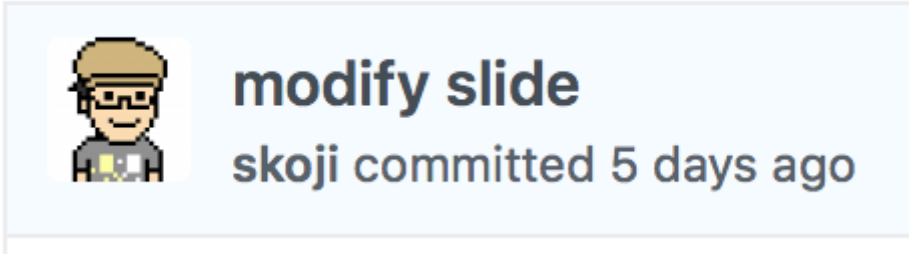


Photograph: Dick Thomas Johnson
https://www.flickr.com/photos/31029865@N06/14997552775

Gitが解決すること

人類の間違いや怠慢を（ある程度）補助する

例えば



この怠惰なコメント

変更点は機械的に確認できる

```
      13  +# バージョン管理システムの目的は？
14    14
15    15   バージョンを管理すること。
16    16
      17  +#(takahashi) 何がうれしいの？
      18  +
17    19   # バージョン管理レベルゼロ
18    20
19    21   image.fig-65(img/level-0.png, 手動バージョン管理):
20    22
21          -section.takahashi {
22          -  h1: どれ
23          -}
      23  +#(takahashi) 最新版を修正…
      24  +
      25  +#(takahashi) どれ
      26  +
24    27   # バージョン管理レベル１
25    28
26    29   厳密な命名ルールの適用
27    30   image.fig-65(img/level-1.png, 手動バージョン管理その2):
```

リンク

誰がいつどこを変更したかも確認できる

blameという機能

Gitの効果

怠惰の補助だけではありません

過去の時点に戻るのが簡単

手元に各バージョンのファイルがなくても大丈夫

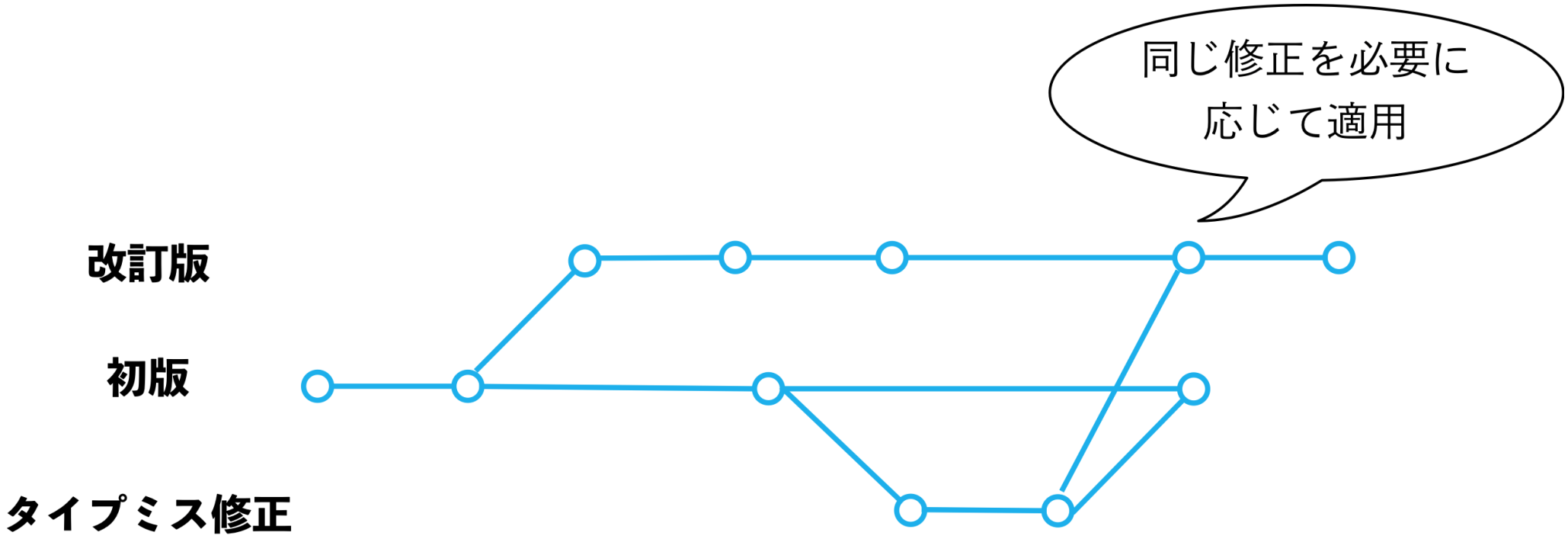
Photograph: Kevin Abato

複数人・同一ファイルへの修正作業

- 同じファイルを複数人で同時に修正しても大丈夫
- 大丈夫じゃないこともあります
  - 例：同じ行を同時に修正したとき
  - あんまり起きない
  - 普通は別の人は別の場所をさわります
  - 大丈夫じゃないときも、なんとかする方法が用意されています

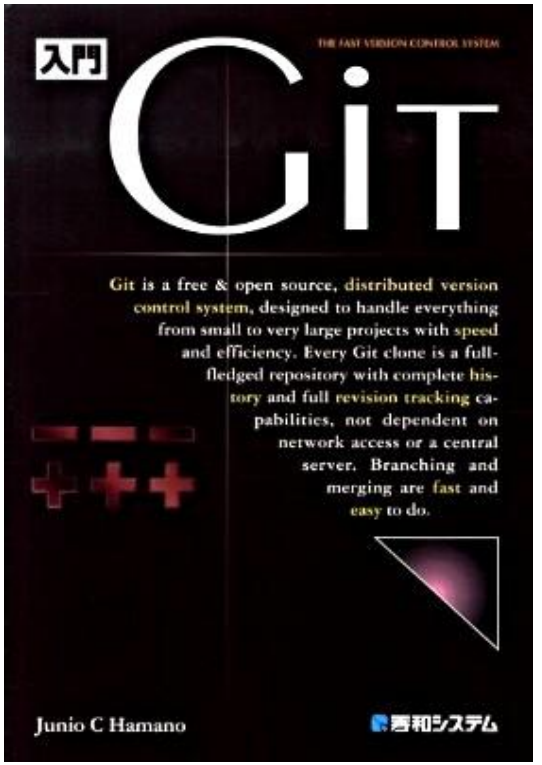
分岐した履歴も扱える

たとえば、改訂版を作りつつ、従来版の間違いを修正するとか。



ほかにもいろいろあります。

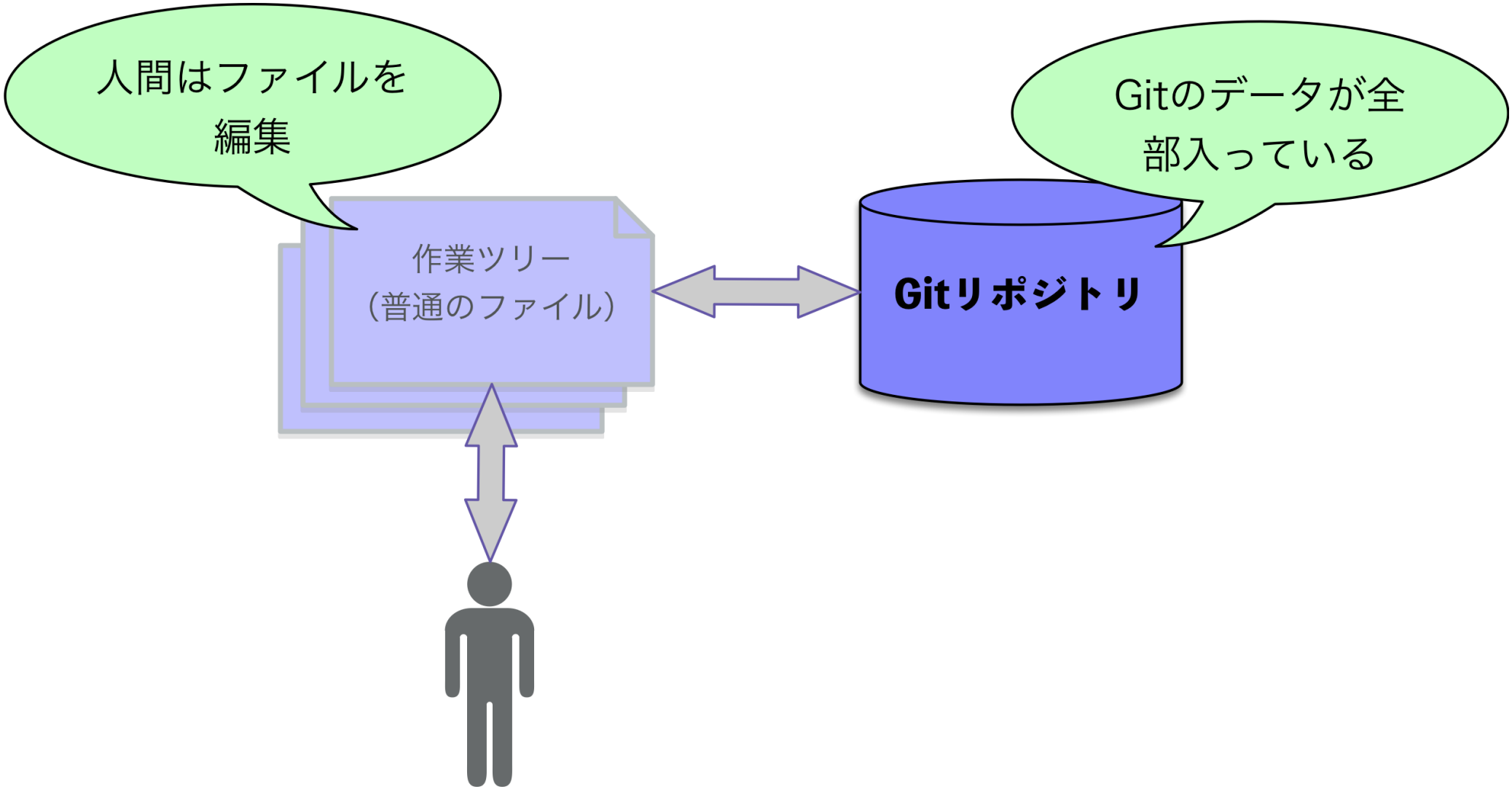
参考文献をご参照くださいませ



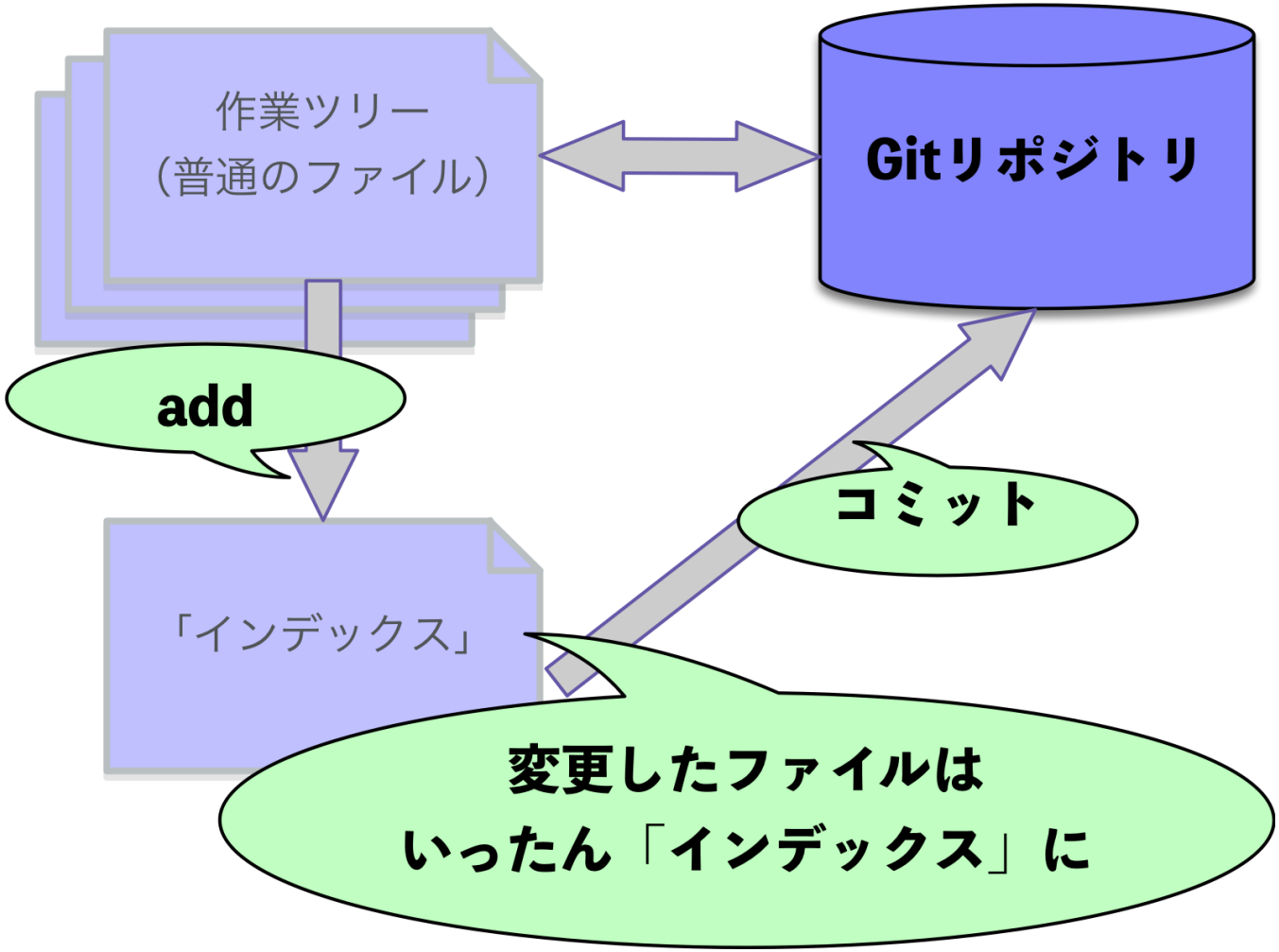
Gitの用語と概念

の基本部分

リポジトリと作業ツリー



addとコミット

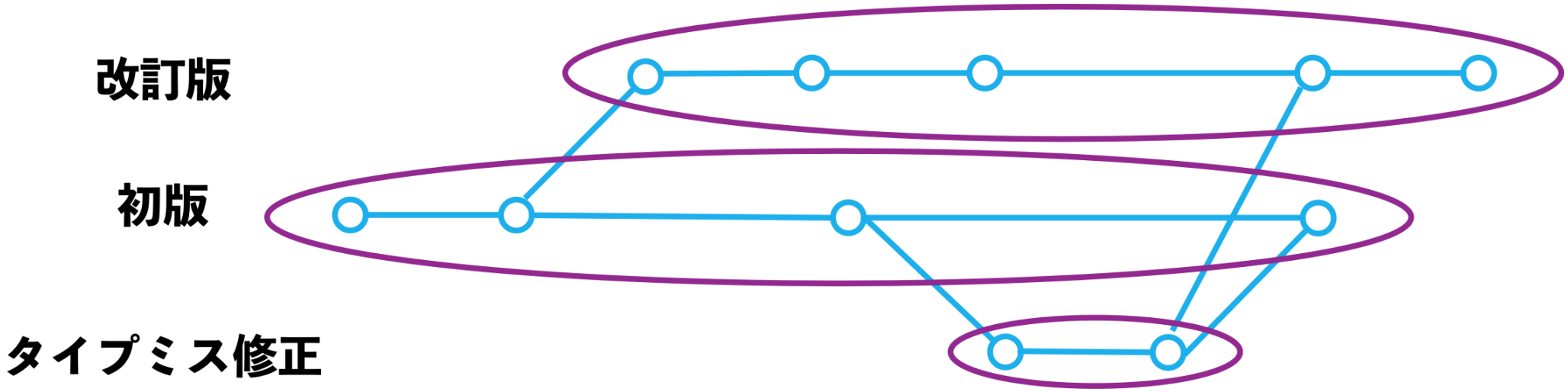

作業ツリー
（普通のファイル）
Gitリポジトリ
add
コミット
「インデックス」
変更したファイルは
いったん「インデックス」に

コミット

- コミットは動詞兼名詞
  - コミットするとコミットができる。
  - 履歴 = 一連のコミット

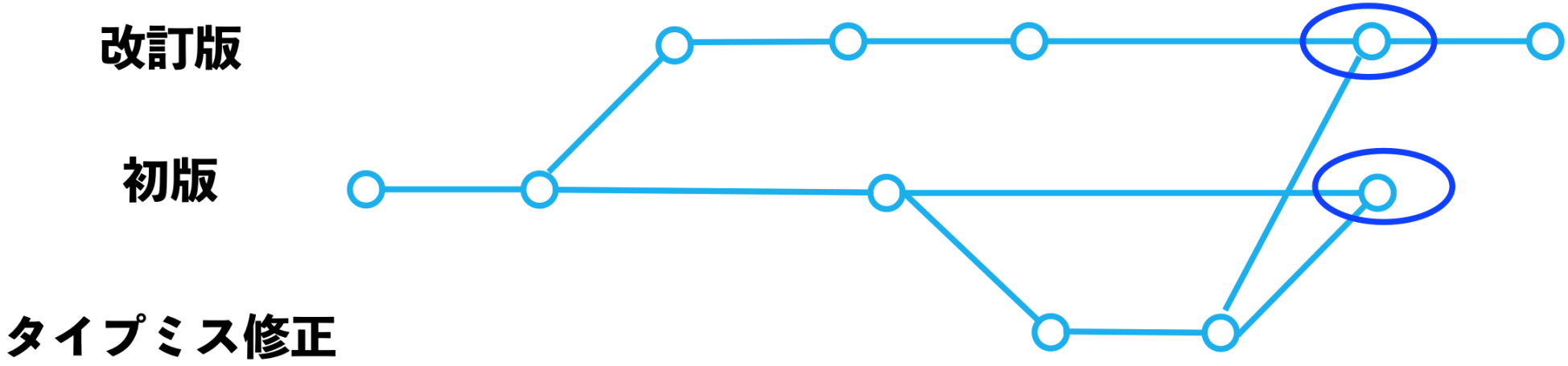
ブランチ

- 履歴 = 一連のコミット
- 分岐した履歴 = ブランチ
- （厳密には、分岐する前の履歴もそのブランチには含まれます）

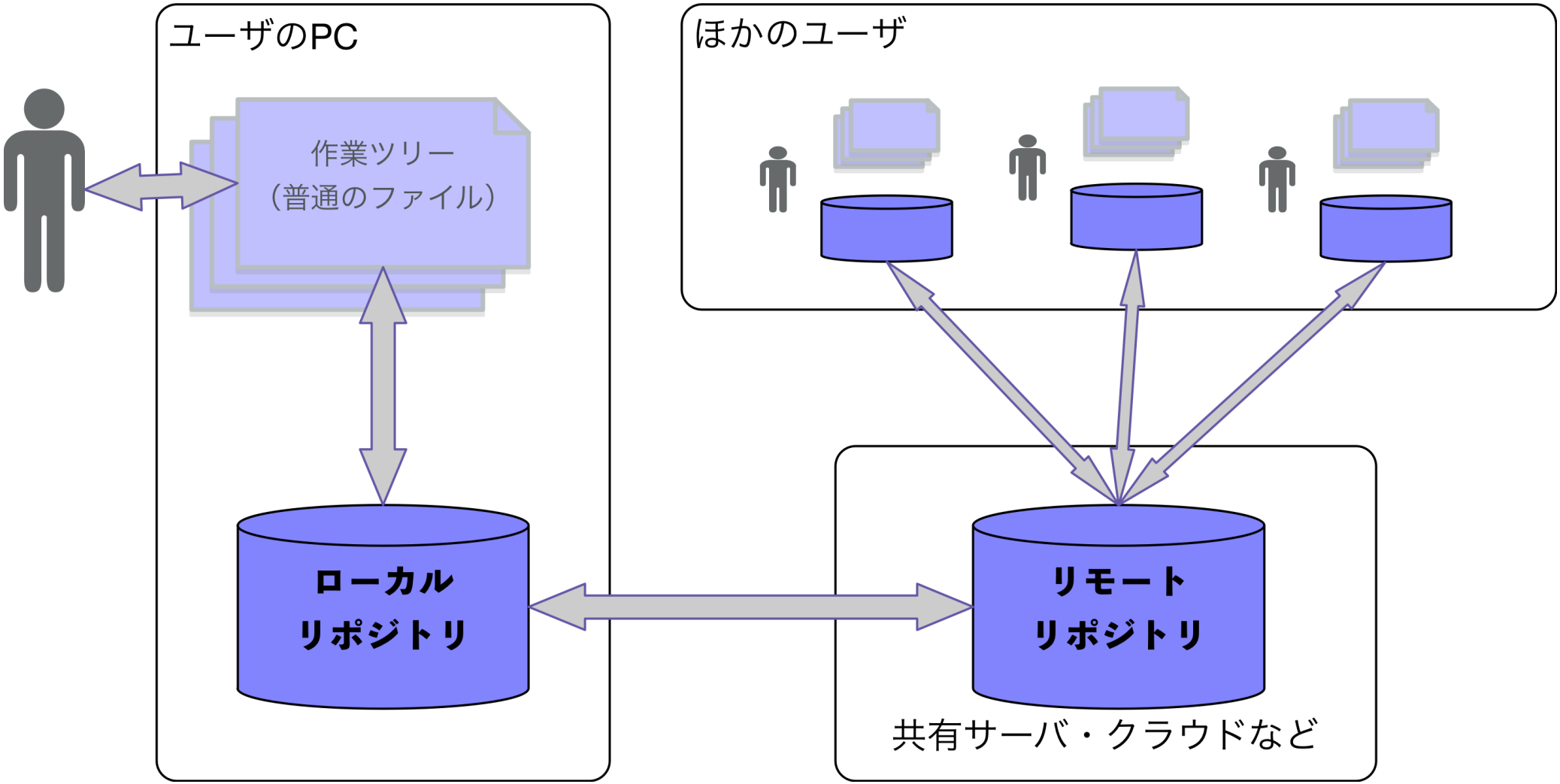


マージ

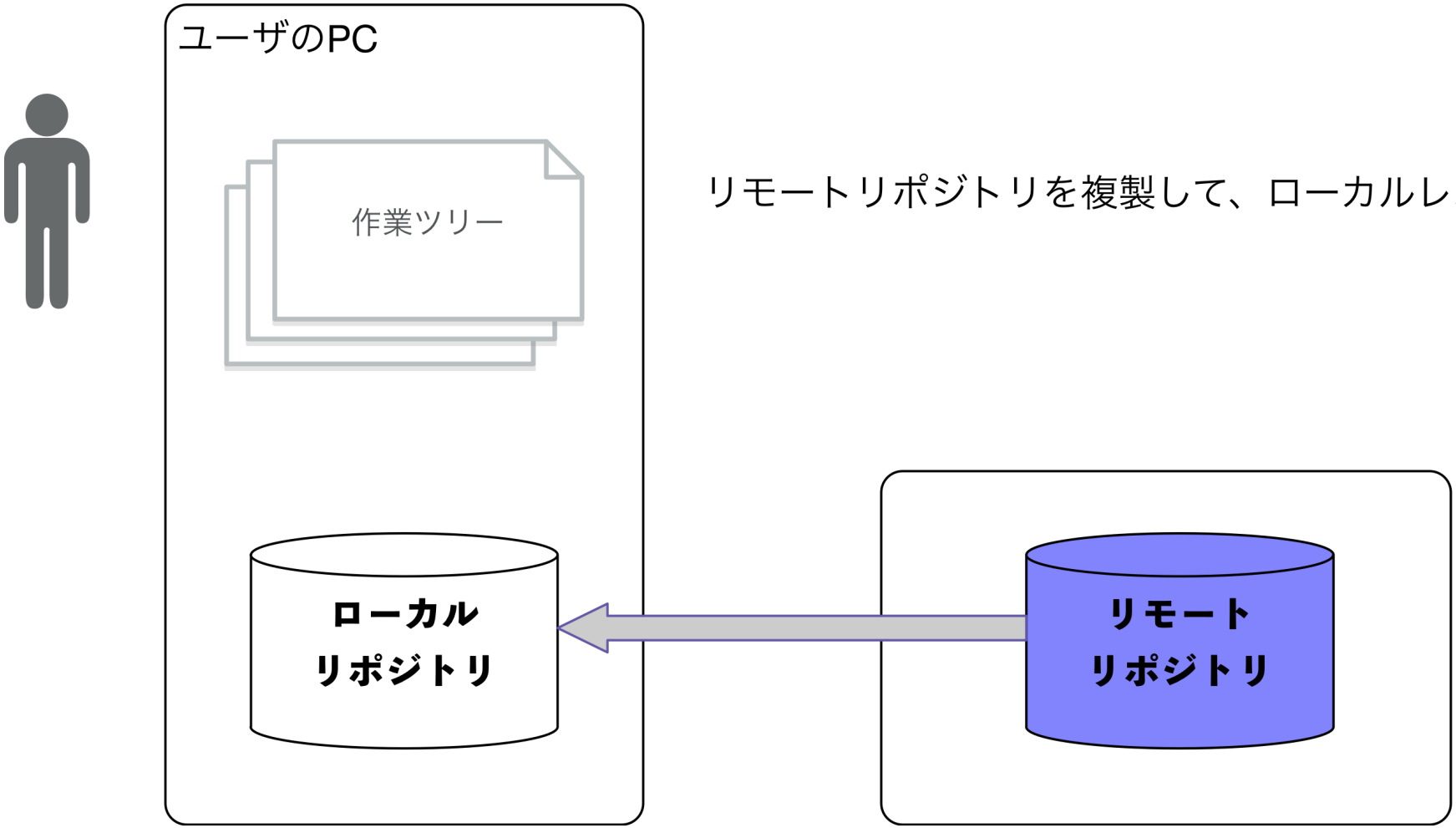
- 分岐した歴史をもう一回混ぜるのがマージ
- 複数人作業ではほぼ必ずマージが必要になります



ローカルレポジトリと共有レポジトリ

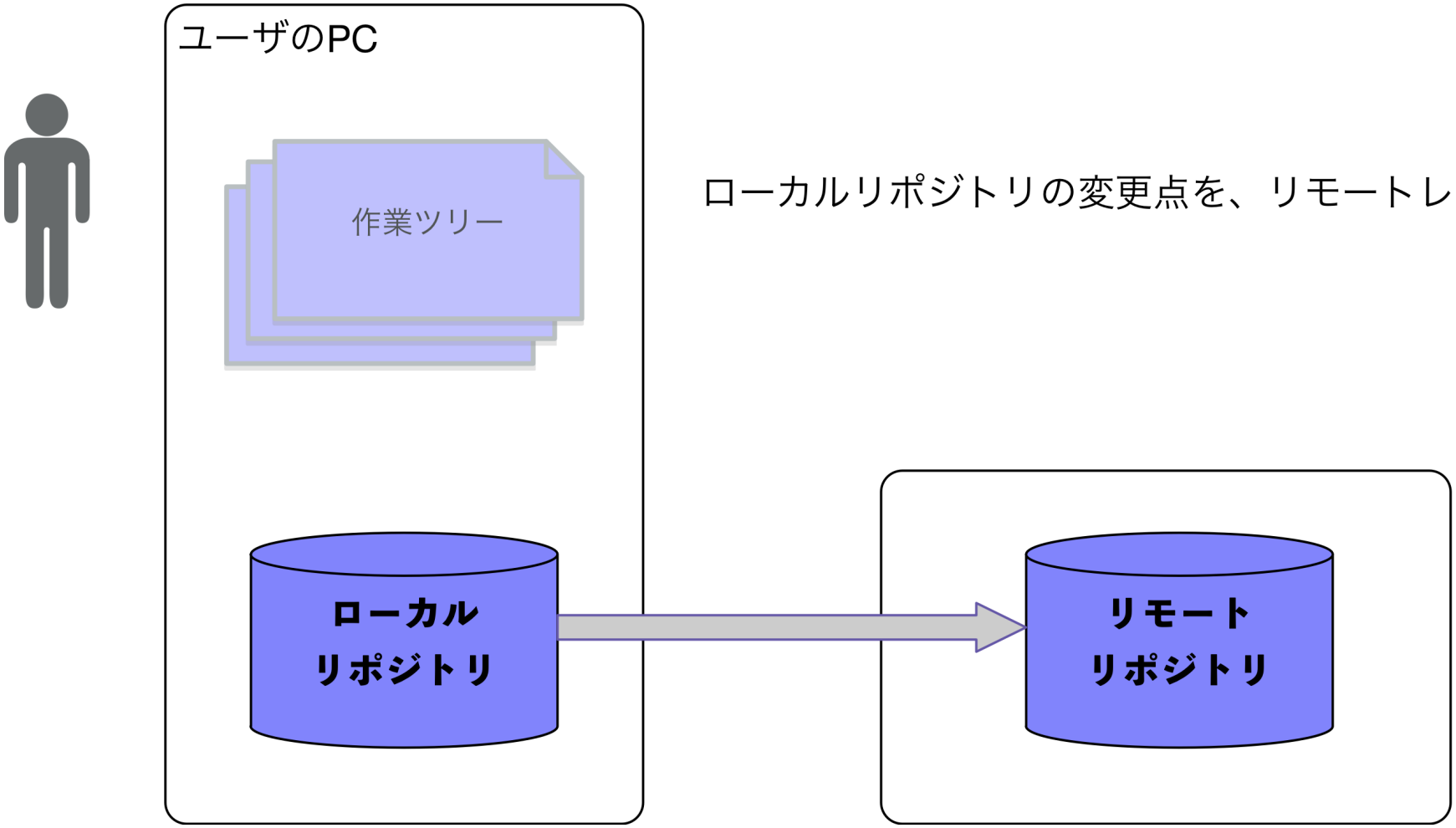


clone



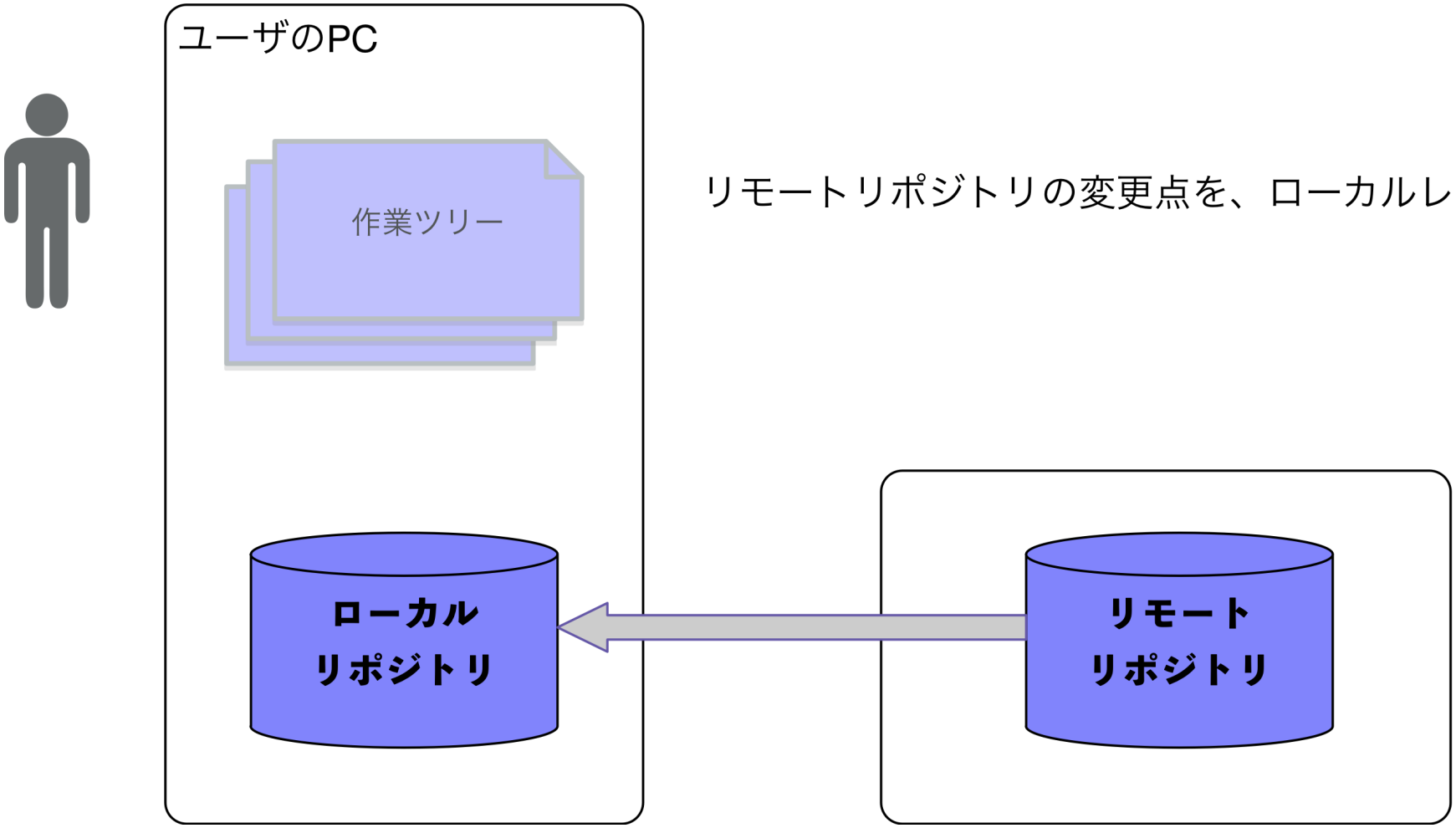
リモートリポジトリを複製して、ローカルレポジトリを作成

push



ローカルリポジトリの変更点を、リモートレポジトリに反映

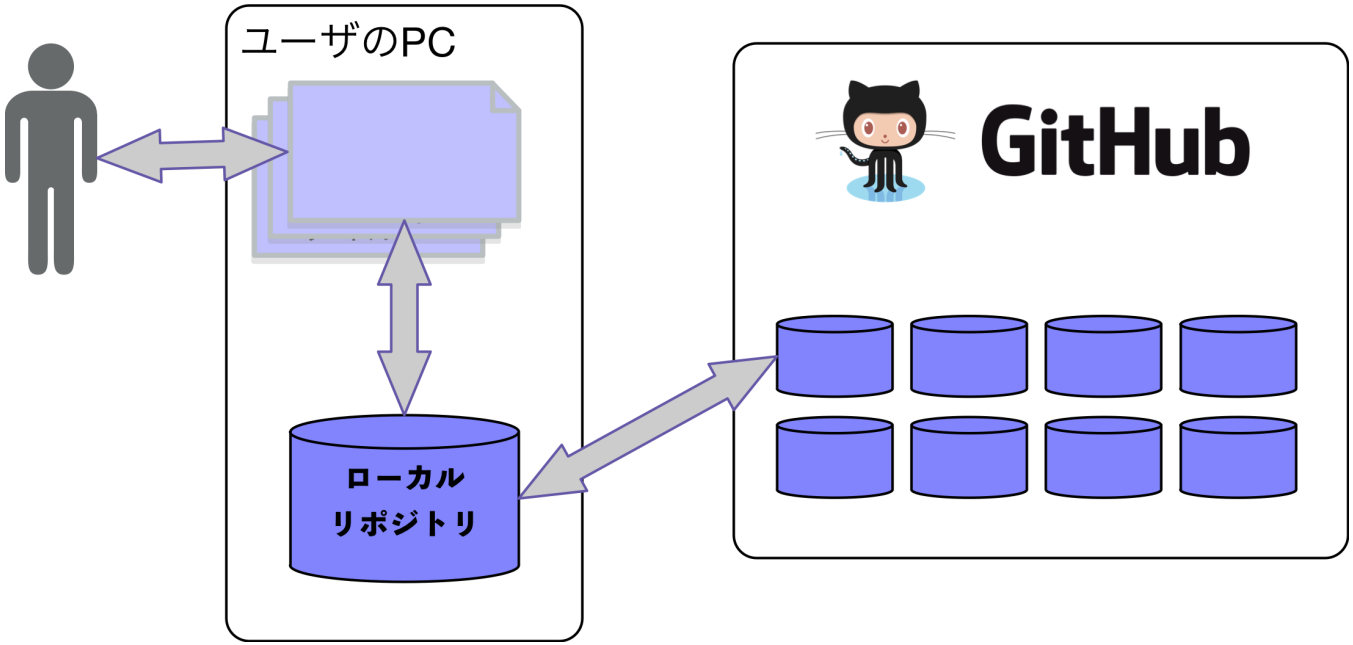
pull



リモートリポジトリの変更点を、ローカルレポジトリに反映

GitとGitHub

- Git ≠GitHub
- GitHubはリモートレポジトリのホスティングサービス



PullRequest

- PullRequestはGitHubの機能
  - Gitの機能ではない
- 「この変更良かったらマージしてね！」という依頼。
- （GitLabではMergeRequestと呼んでいる）
- ちょっとやってみますね

最後に

Gitは多機能

説明しきれません

でも

ごく一部でも便利

まずは使ってみよう

参考文献

- 非エンジニアも対象
- 内容が浅いわけではない
- むしろ濃い

わかばちゃんと学ぶGit使い方入門

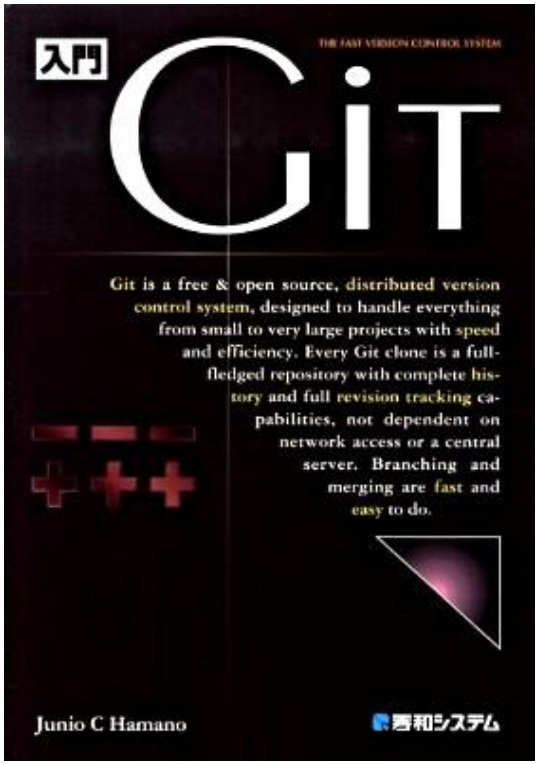
参考文献

- 概要・コマンドリファレンス・開発フローまで幅広くカバー。
- ややエンジニア向け

【改訂新版】Gitポケットリファレンス

参考文献

- Gitのメンテナによる書籍
- いわゆる入門書ではない
- Gitがどんなデータ構造をどう扱っているか知りたくなったらおすすめ



入門Git