

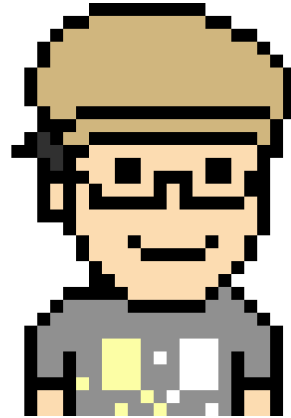
# **Gitとはなにか**

**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



- そのくらい一般的
- テキストものを管理するときは無意識に使っている
  - この資料もGit管理
- （ただし実はGitでなくとも良い）

# 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…

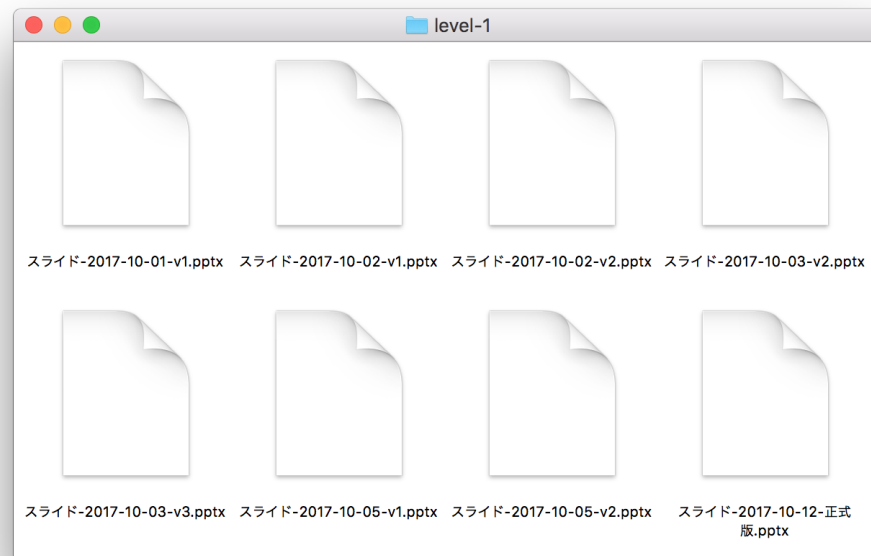
# この



どれ

# バージョン管理レベル1

## 厳密な命名規則の適用



# 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

# 人間は間違える

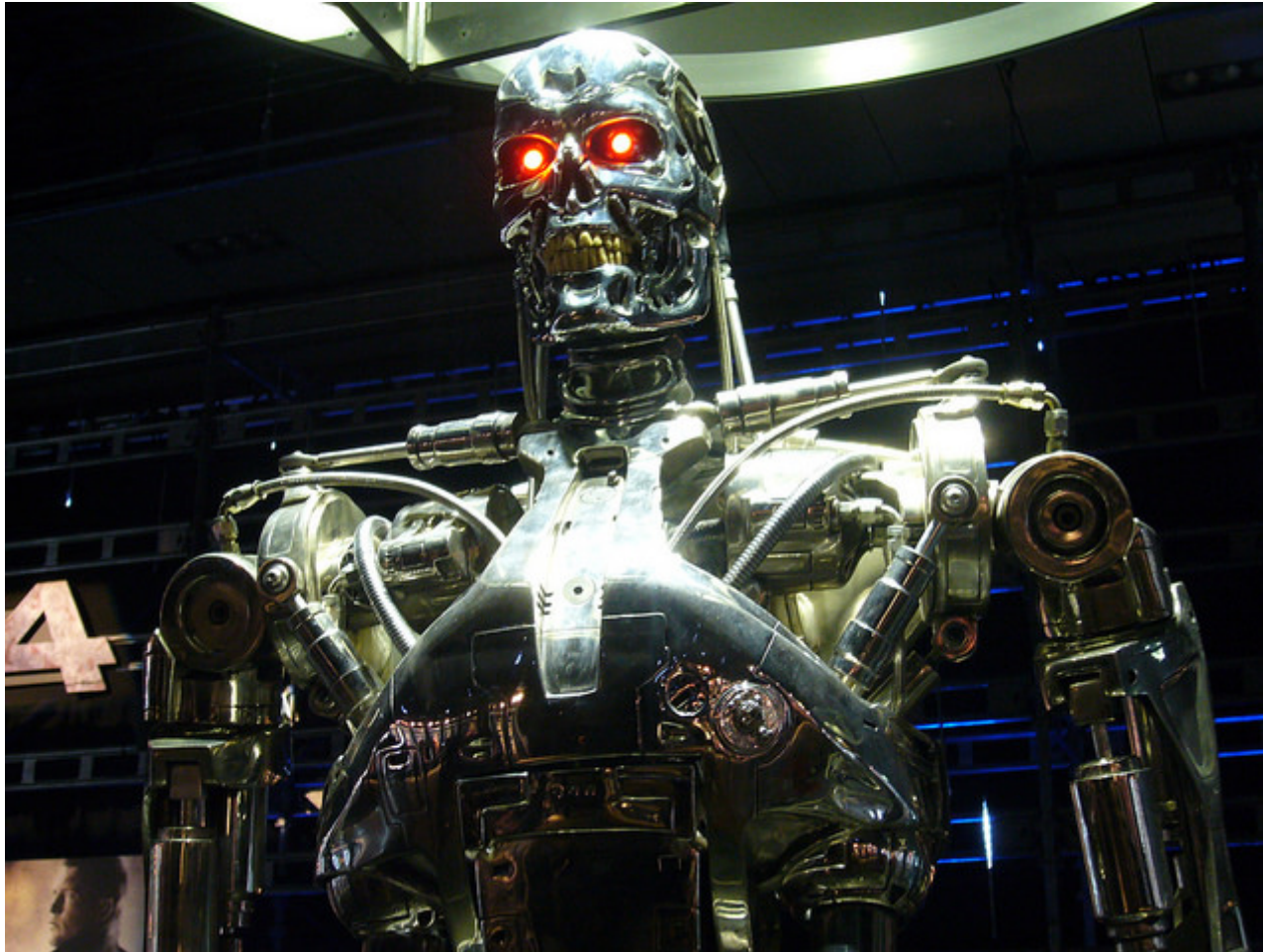
# 人間は怠ける

# 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」
- いちいち書くの面倒だからまとめてやろう
- あれ何変更したっけ

# 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



**modify slide**

skoj committed 5 days ago

この怠惰なコメント



# 変更点は機械的に確認できる

13	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	17	+#(takahashi) 何がうれしいの？
18	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	21	-section.takahashi {
22	22	- h1: どれ
23	23	-}
23	23	+#(takahashi) 最新版を修正...
24	24	+
25	25	+#(takahashi) どれ
26	26	+
24	27	# バージョン管理レベル1
25	28	
26	29	厳密な命名規則の適用
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という機能です（名前が怖い）

		53	<code>&lt;div class='author-profile'&gt;&lt;p&gt;まつだきょうこ&lt;/p&gt;</code>
	著者さんからの要望によりプロフィール修... 6d1ca467 Yasushi Furuta committed 2 years ago	54	<code>&lt;p&gt;京都出身、東京→2011年5月鹿児島へ移住。フリーライター。地域活性化、:</code>
	101号、新米おごじょ cbd306ab Satoshi Kojima committed 2 years ago	55 56	<code>&lt;/div&gt; &lt;/body&gt;</code>
	著者さんからの要望によりプロフィール修... 6d1ca467 Yasushi Furuta committed 2 years ago	57	<code>&lt;/html&gt;</code>

# Gitの効果

怠惰の補助ではありません

# 過去の時点に戻るのが簡単

手元に複数ファイルがなくても大丈夫。



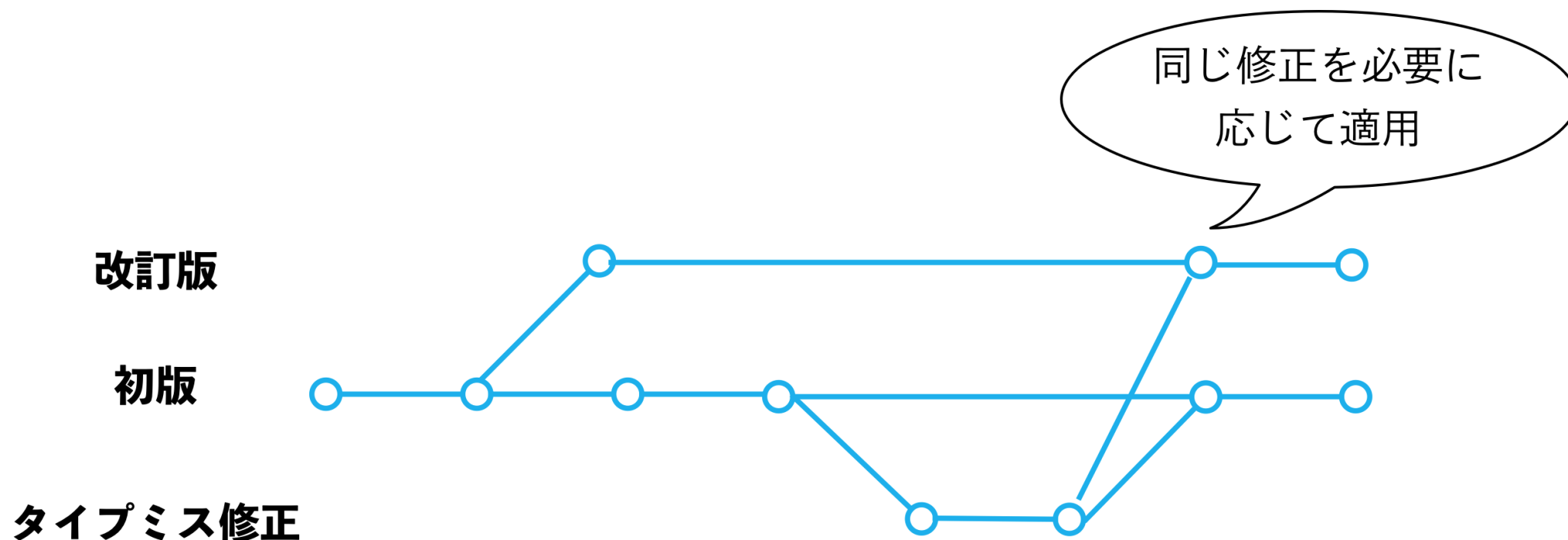
Photograph: Kevin Abato

# 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もちろんあります
  - 同じ行の修正など
  - でも普通は、別の人とは別の場所をさわります
  - 大丈夫じゃないときも、ちゃんと修正する方法が用意されています

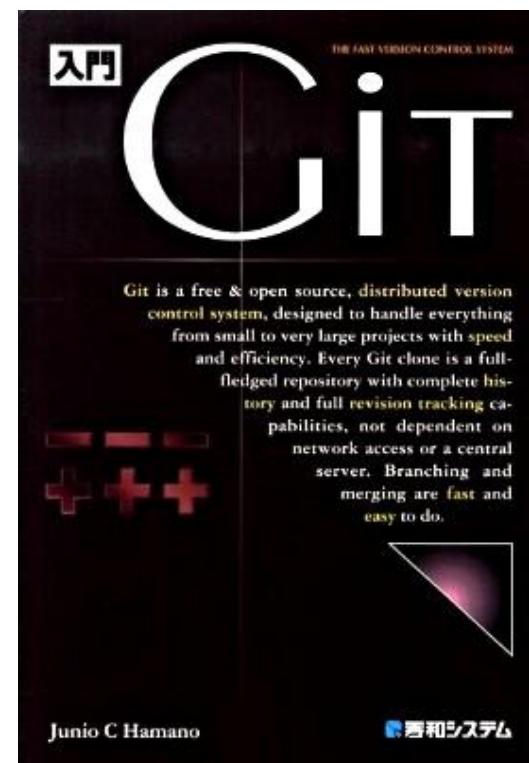
# 分岐した履歴も扱える

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

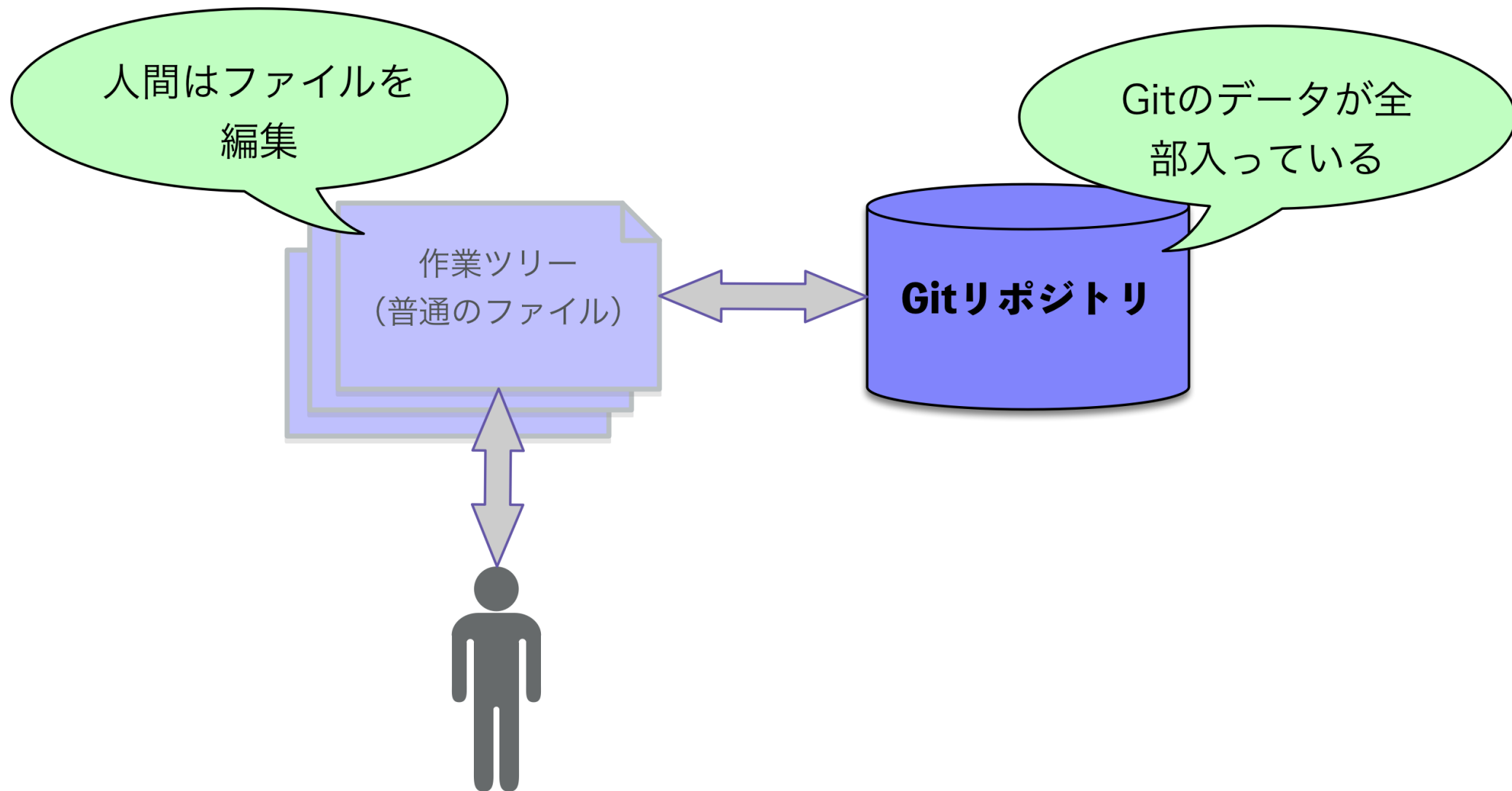


# Gitの用語と概念

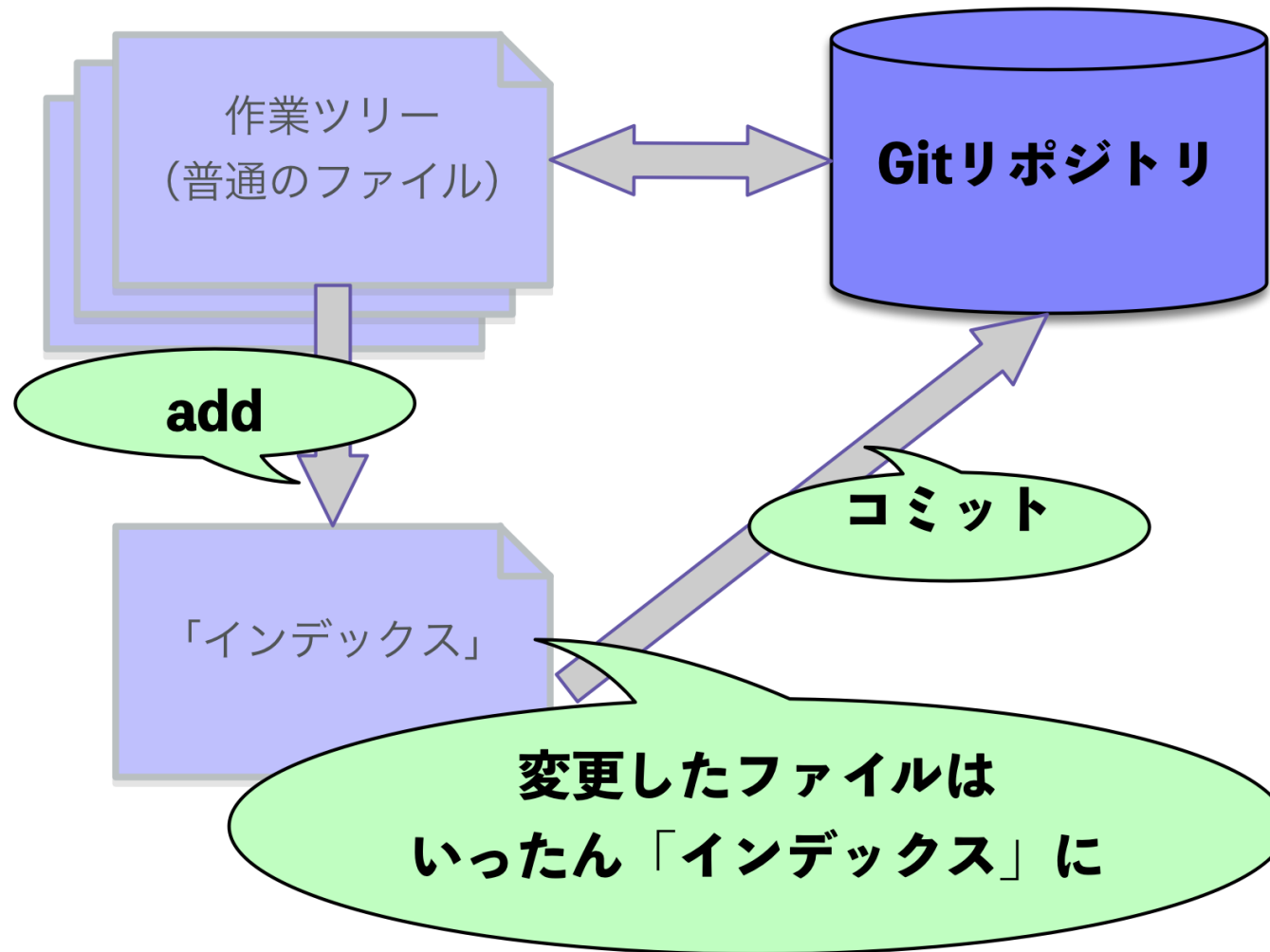
## の基本部分



# リポジトリと作業ツリー



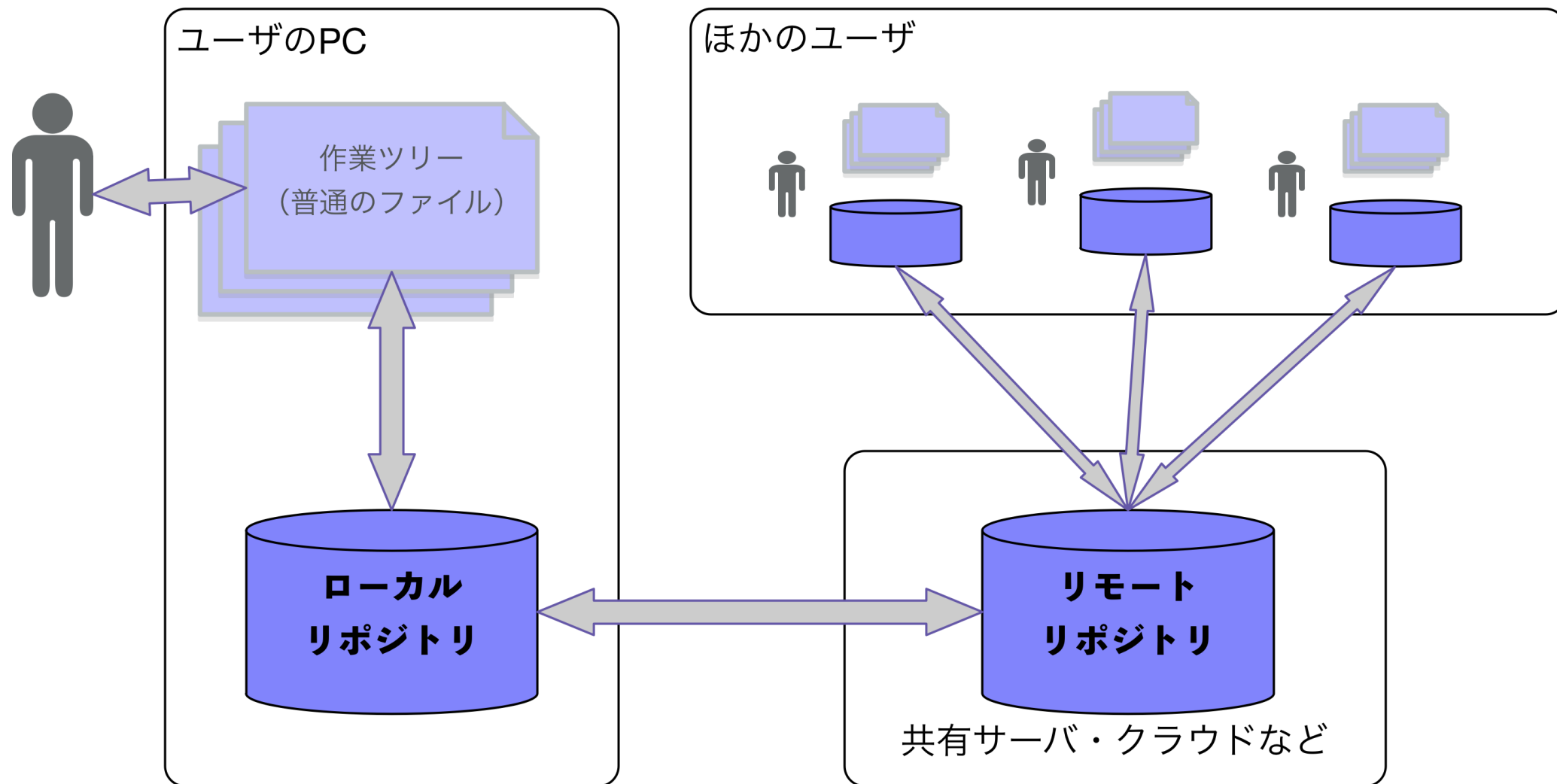
# addとコミット



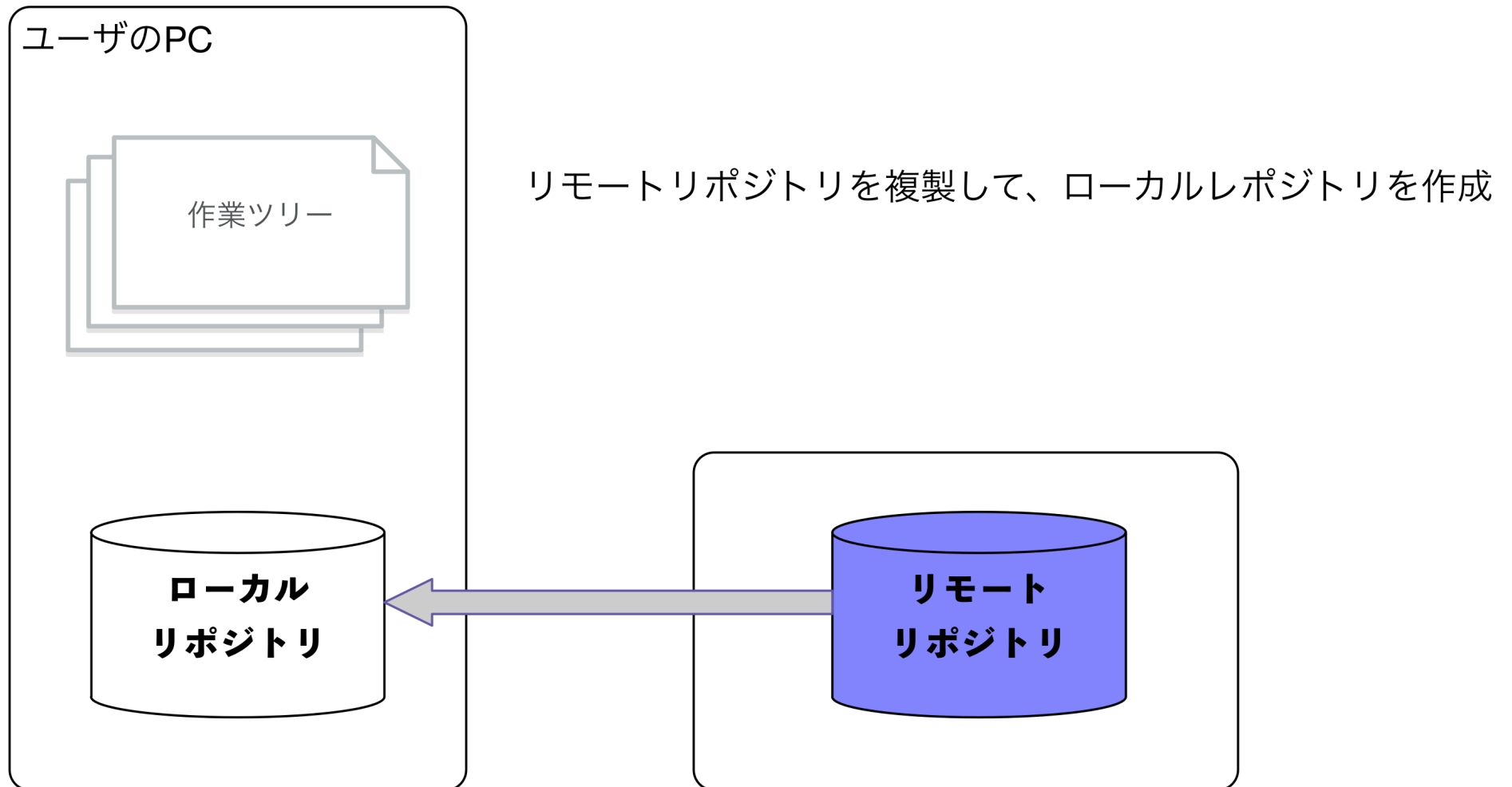
# addとコミット

- addで作業ツリーの内容がインデックスに反映される
- コミットでインデックスの内容がリポジトリに反映される
- コミットは動詞兼名詞。
  - コミットするとコミット（オブジェクト）ができる。
  - 一連のコミット ≡ 履歴。

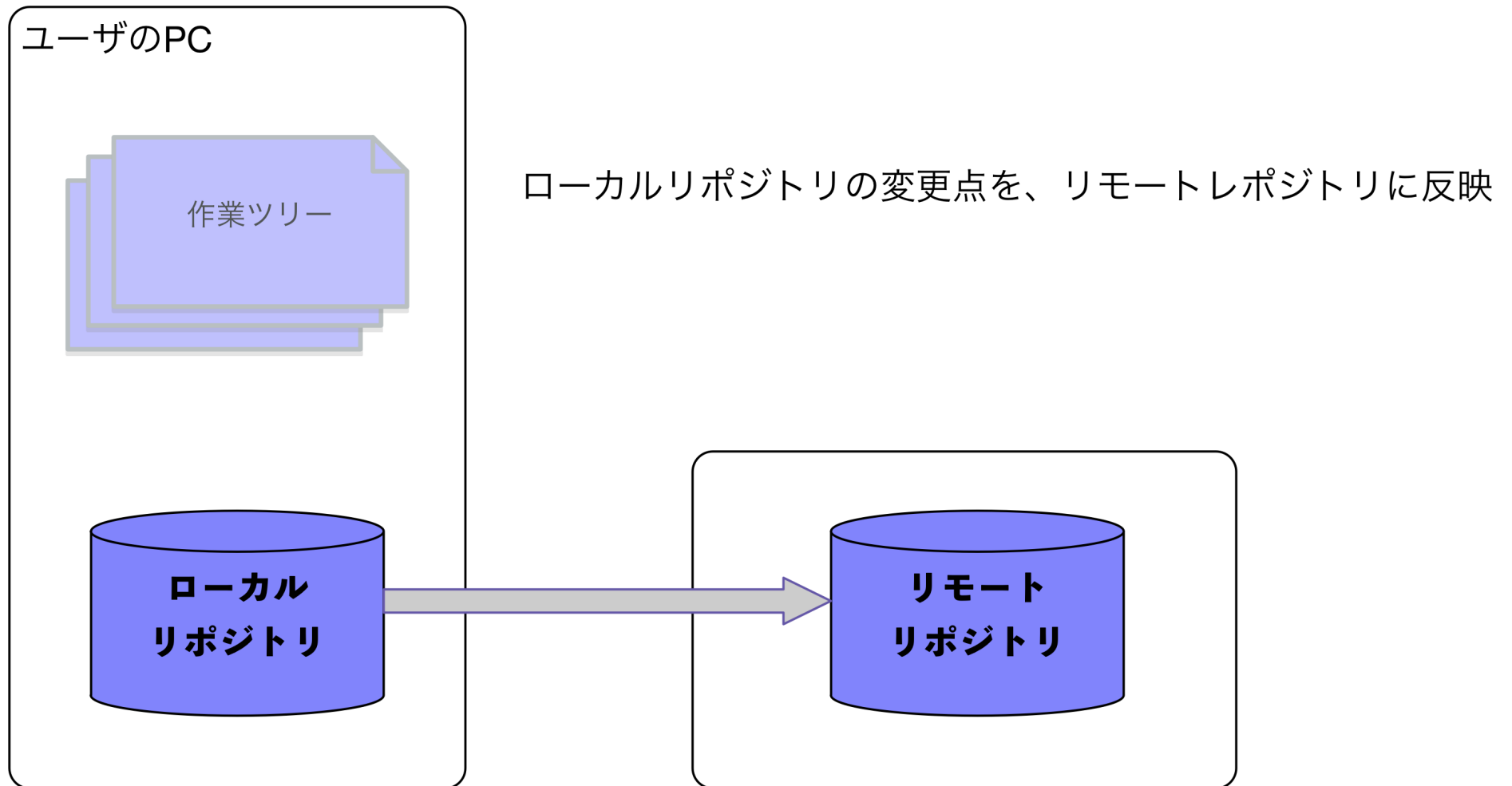
# ローカルレポジトリと共有レポジトリ



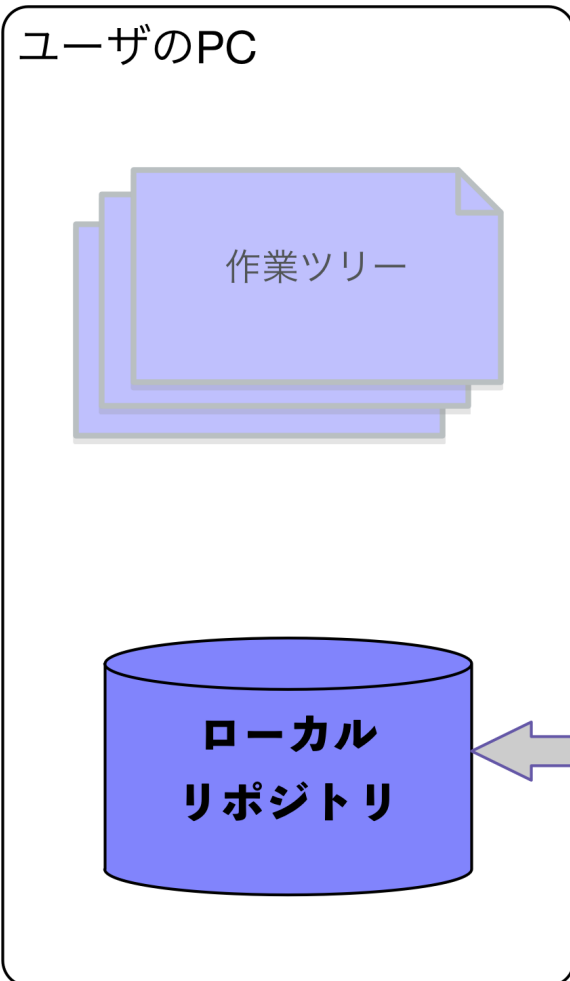
# clone



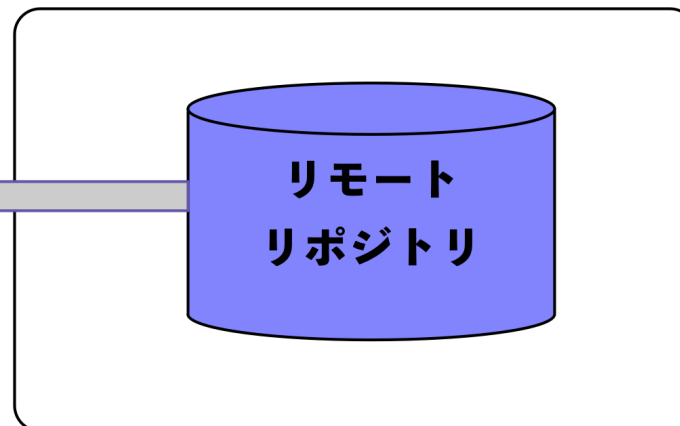
# push



# pull

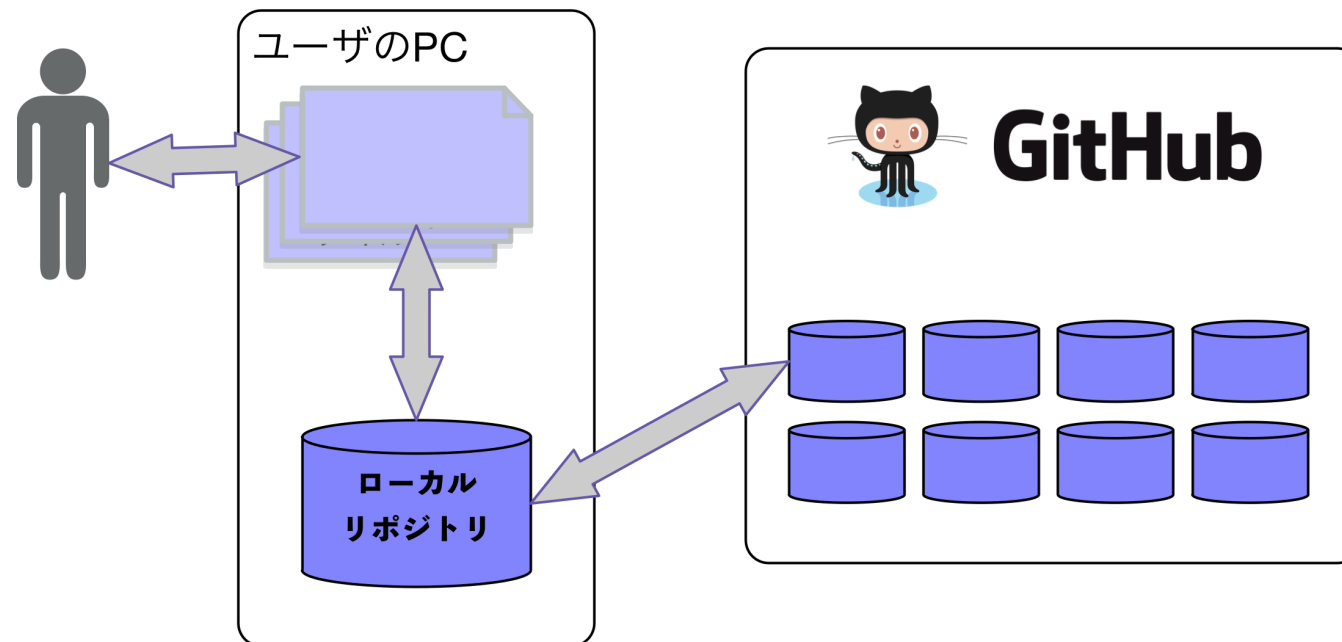


リモートリポジトリの変更点を、ローカルレポジトリに反映



# GitとGitHub

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





# 最後に

# Gitは多機能

**ごく一部しか説明で  
きませんでした**

でも

**ごく一部だけ知って  
ても便利**

**まずは使ってみるの  
がおすすめです**

# 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

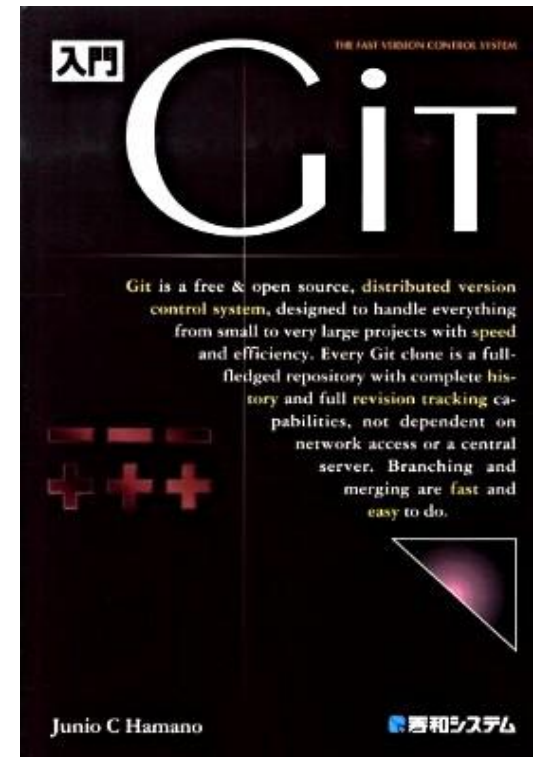


【改訂新版】Gitポケットリファレンス



# 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がどのようなデータ構造をどのように扱っているか知りたくなったらおすすめ



[入門Git](#)