

# Git入門1

~Gitの基礎を覚えて使ってみよう~

課外プロジェクト資料 2014 4/21

d8161105 MiraiWatanabe



# ◇ ファイル管理

人は重要なファイル群を管理するときにディレクトリにまとめる。  
さらに、そのファイル群の**変更**と**履歴**が重要視される場合がある。

## 古典的なファイル管理方法

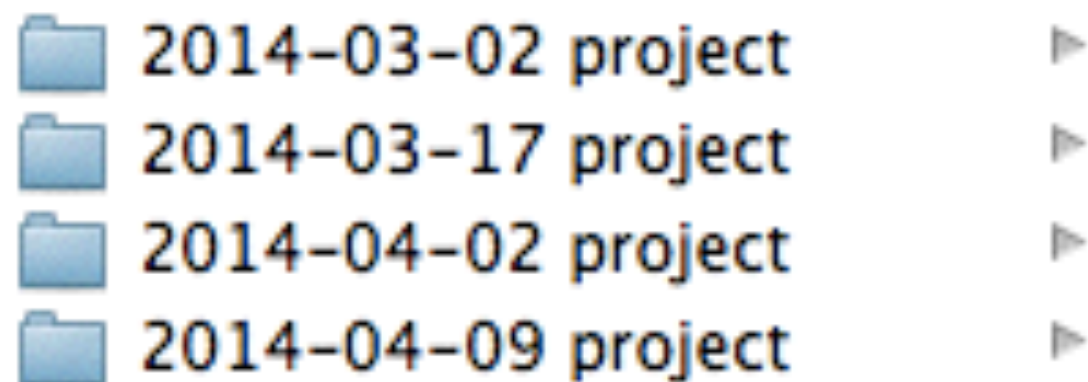
```
2014-04-02 山田 変更ファイル memo: 議事録内容を修正
2014-03-20 ワタナベ 変更ファイル memo: 議事録内容を追加 , main.c: hoge関数を削除
2014-03-16 ワタナベ 変更ファイル memo: 議事録内容を追加 , main.c: hoge関数を追加
2014-03-12 スズキ 変更ファイル memo: 議事録内容を修正 , foo.c: printfのテキストを変更
2014-03-02 ワタナベ 変更ファイル memo: 議事録内容を追加 , main.c: hoge関数を追加
```

### ログを残す方法

ログを残すファイルを用意し、  
管理する。細かく書けるが大変。

### 日付を残す方法

ディレクトリに日付を書いて管理する。  
ただし、どのような変更がされたかは、  
解らない。数が大きくなりすぎて管理  
しきれない。



```
2014-03-02 project
2014-03-17 project
2014-04-02 project
2014-04-09 project
```

# ◇ 「Git」 による管理

Gitは、データの差分を管理するためのツールである。

このようなツールをバージョン管理ツールと呼ぶ。

**add test program**  
master  
ababup1192 authored 3 hours ago

Showing 1 changed file with 8 additions and 0 deletions.

8 hello.c		
...	...	@@ -0,0 +1,8 @@
	1	+ #include <stdio.h>
	2	+
	3	+ int main(){
	4	+
	5	+     printf("hello world!");
	6	+
	7	+     return 0;
	8	+ }

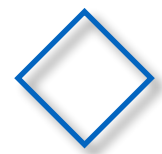


**change test c program**  
master  
ababup1192 authored 3 hours ago

Showing 1 changed file with 4 additions and 1 deletion.

5 hello.c		
...	...	@@ -1,8 +1,11 @@
1	1	#include <stdio.h>
2	2	
3	3	int main(){
	4	+     int i;
4	5	
5		-     printf("hello world!");
	6	+     for(i=0; i<10; i++){
	7	+         printf("hello loop world!");
	8	+     }
6	9	
7	10	return 0;
8	11	}

何が増え、何が減ったのか



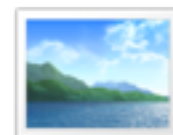
## 「Git」による管理 | Gitができること

- ▶ 誰が何を変更したか確認したい
- ▶ 変更前のファイル状態に戻したい
- ▶ 他の人が変更したファイルを別の内容で上書きしてしまった。  
両方の内容を変更したい
- ▶ ある時点から正しく動かなくなってしまった。変更履歴を比較して問題箇所を見つけたい
- ▶ うまくいったら採用し、失敗した場合ははきできるような実験的な変更を試してみたい

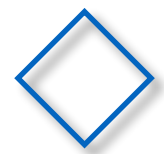
引用：Git逆引き入門

# ◇ 「Git」 による管理 | 管理対象

データならば**どんなもの**でも管理可能である。

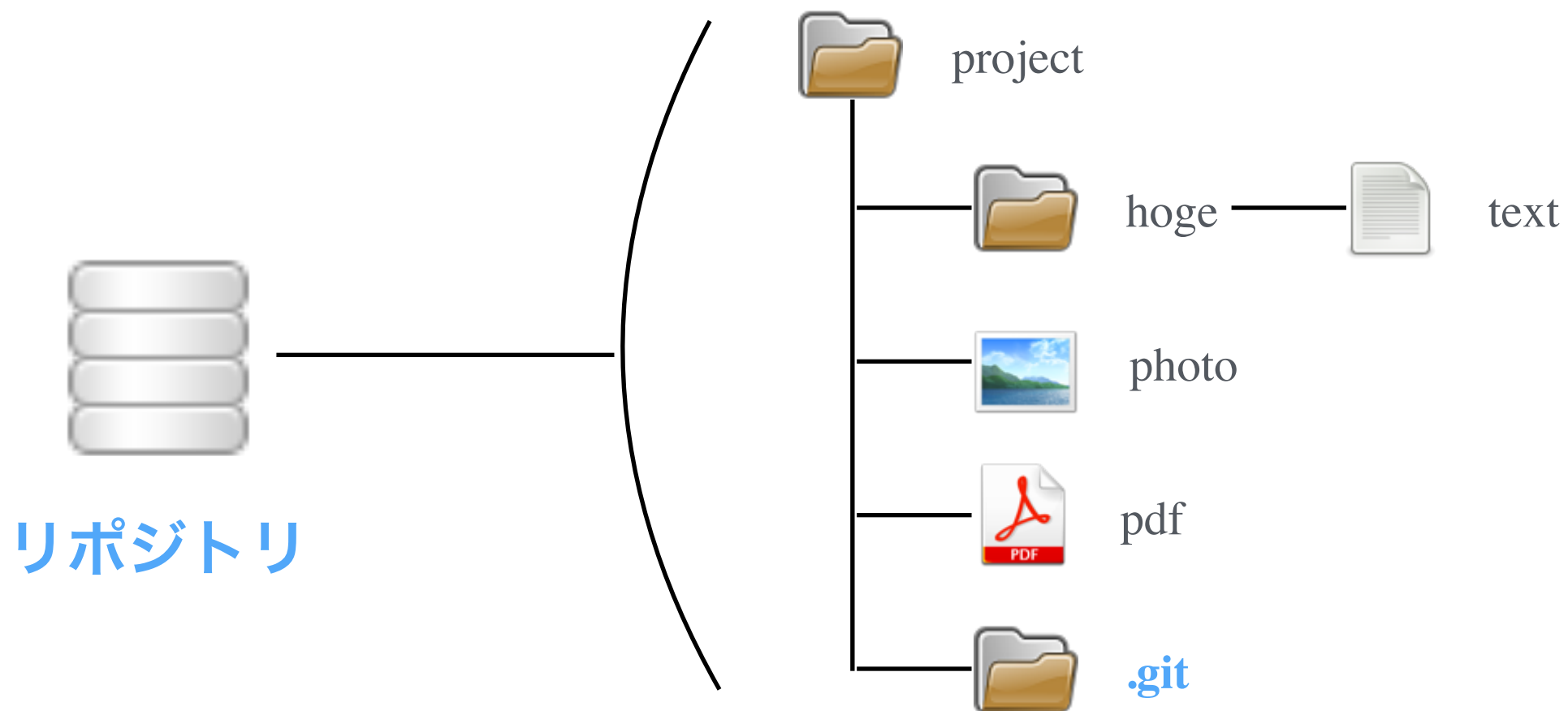


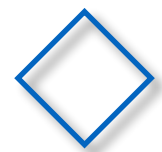
- ▶ 開発のプロジェクト
- ▶ 研究データ・論文執筆
- ▶ シェルやアプリケーションの設定ファイル
- ▶ 授業の課題・宿題



## 「Git」による管理 | リポジトリ

バージョン管理される対象のディレクトリを**リポジトリ**と呼ぶ。  
リポジトリのルートには、**Git**を管理するため`.git`という名前の  
ディレクトリが作られる。これを**Gitディレクトリ**と呼ぶ。**Git**は  
このディレクトリで差分を管理している。

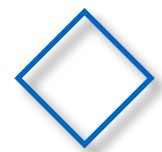




## 「Git」による管理 | データ領域

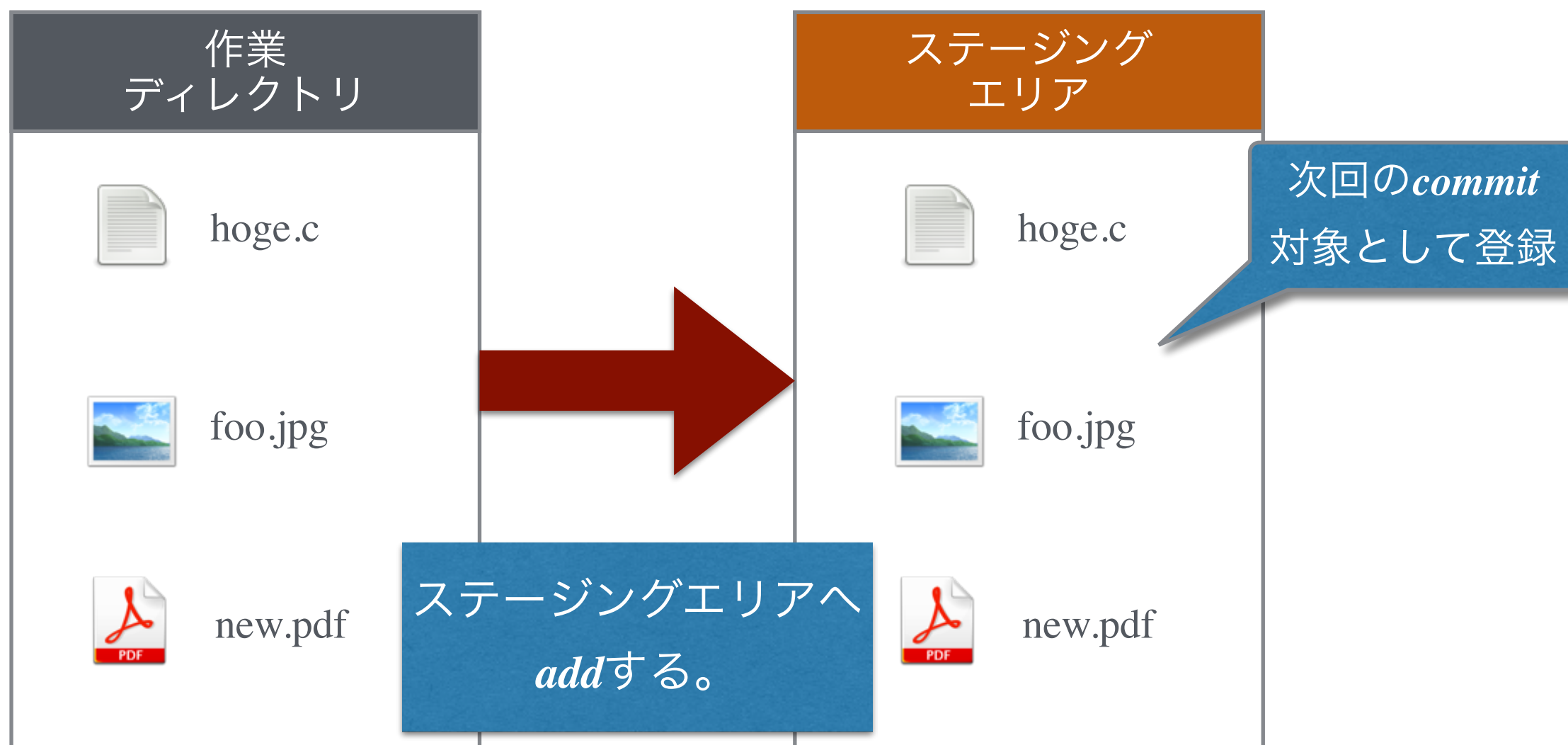
Gitの使い方を覚える上で、Gitの差分を管理する仕組みを少し覚える必要がある。Gitの差分は以下に示すデータ領域の中で管理される。



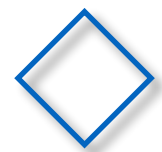


## 「Git」による管理 | データ領域

作業途中や終了時にファイルの変更つまり、ファイルの作成、ファイルの修正、ファイルの削除を差分として保存したいとき、まず変更対象ファイルは**ステージングエリア**へと登録される。

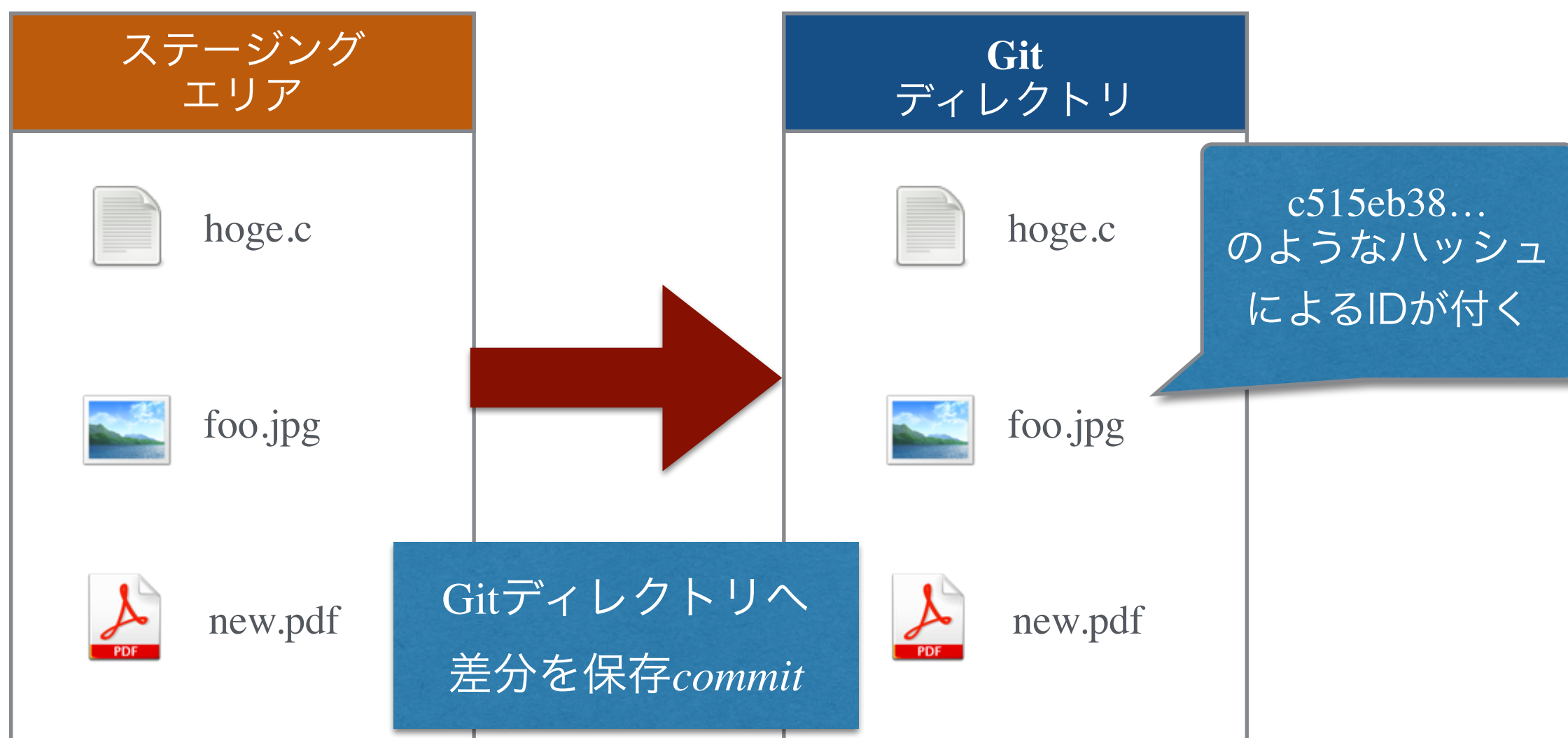






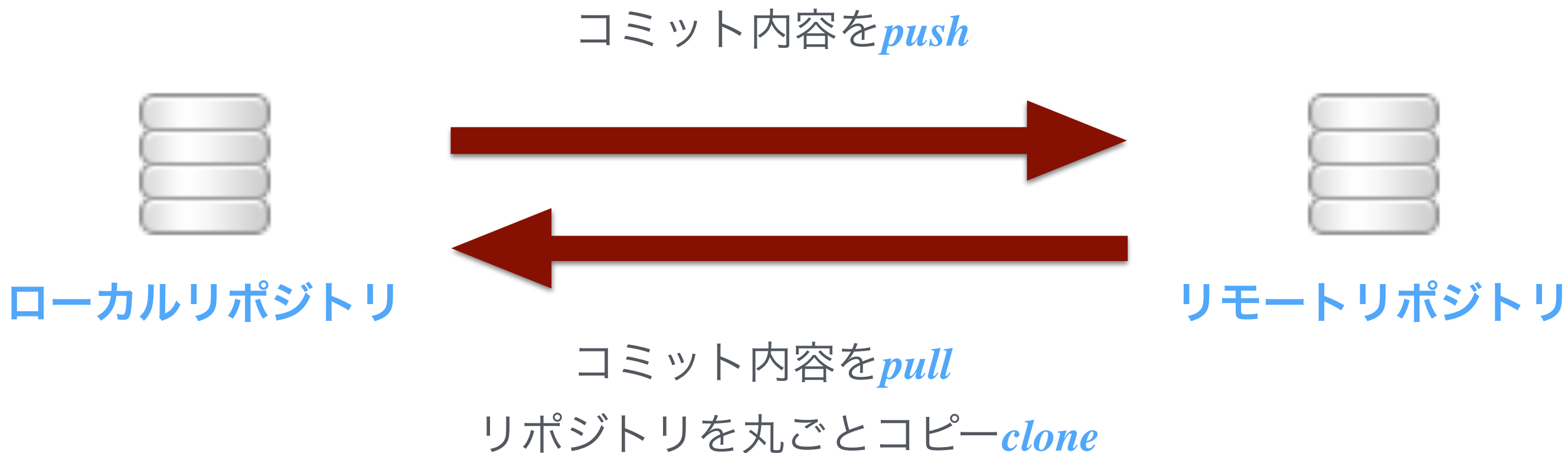
## 「Git」による管理 | データ領域

差分を確定するときは、ステージングエリアから**Git**ディレクトリへと移動する。差分を確定することを**コミット**と言う。コミットは、変更に対するコメントが求められる。また、コミット時間、ユーザ名などが明記される。コミットはハッシュコードにより判別される。



# ◇ 「Git」 による管理 | リモートリポジトリ

Gitのコマンドにより作られ、個人のために作られたリポジトリを**ローカルリポジトリ**と呼ぶ。それに対し、他人と共有するため、もしくはバックアップ用途として外部サーバーに作るリポジトリを**リモートリポジトリ**と呼ぶ。



## ◇ 「Git」による管理 | Gitホスティングサービス

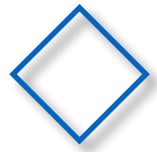
リモートリポジトリを用意するには、サーバーを用意し**Git**本体だけでなく使いやすいWebクライアントなども準備する必要がある。高コストである。そのため、そのようなサーバーを提供してくれるWebサービスなどがある。そのようなサービスを**Gitホスティングサービス**と呼ぶ。



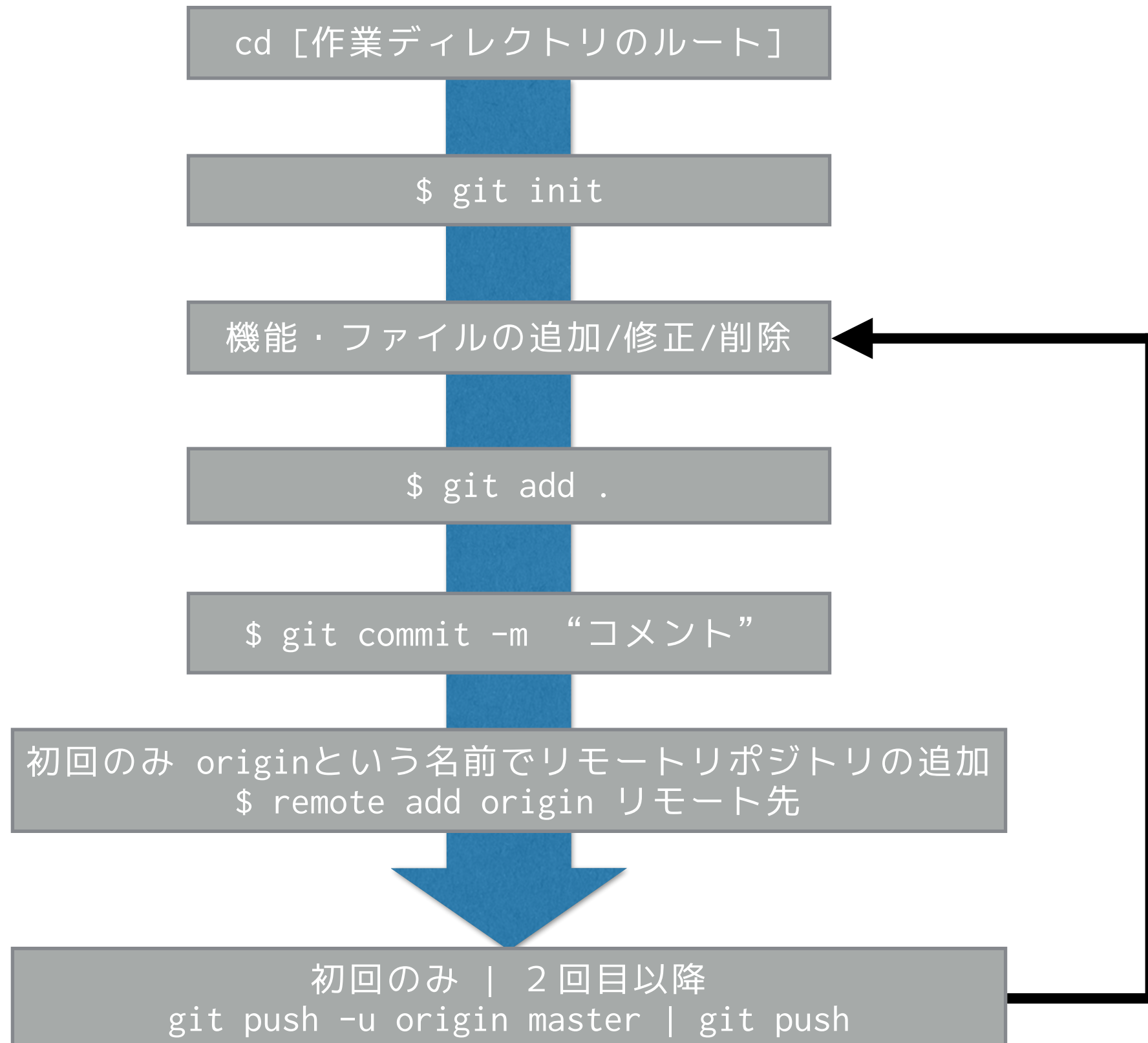
ソーシャルコーディングがスローガンで、オープンソースのプロジェクトが多く存在する。とても使いやすいクライアントで他人のコードが参考として簡単に見ることができるなどの利点が挙げられる。



プライベートリポジトリが作り放題な**Git**ホスティングサービスで、外部に公開したくないリポジトリは、こちらのサービスを利用すると良い。



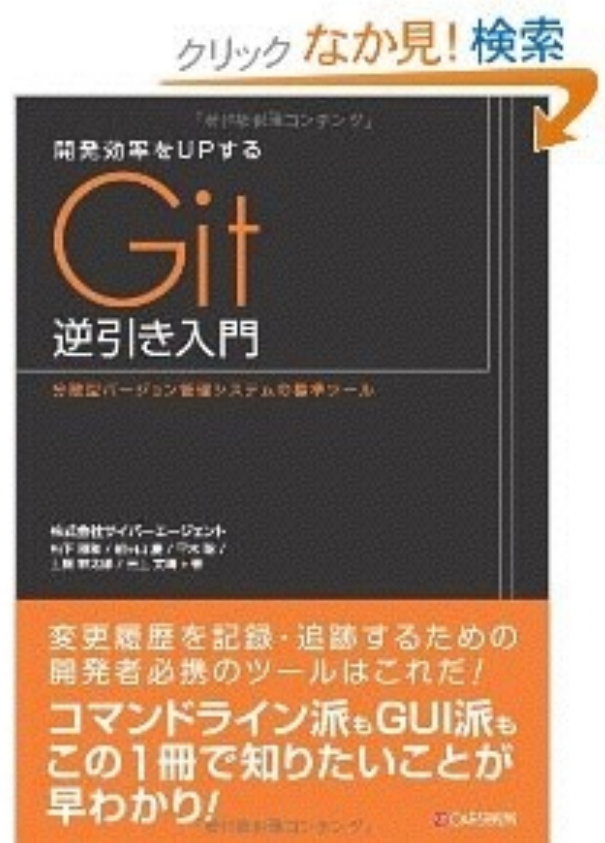
# 「Git」による管理 | Gitのライフサイクル



## ◇ まとめ

- ▶ 差分を管理するバージョン管理ツールGitの使い方を学んだ
- ▶ Gitの簡単な仕組みを抑え、基礎コマンドのライフサイクルとその挙動を抑えた
- ▶ Gitホスティングサービスとそのサービスごとの利点を学んだ

## ◇ 参考文献



### Git逆引き入門



サルでもわかる**Git**

<http://www.backlog.jp/git-guide/>