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

目的

- 従来のファイル管理方法の**リスク**と**非効率さ**を**認識する**
- **差分によるファイル管理方法**を学ぶ
- 差分管理対象を考える
- **Git**を使った管理方法を学ぶ

従来のファイル管理方法

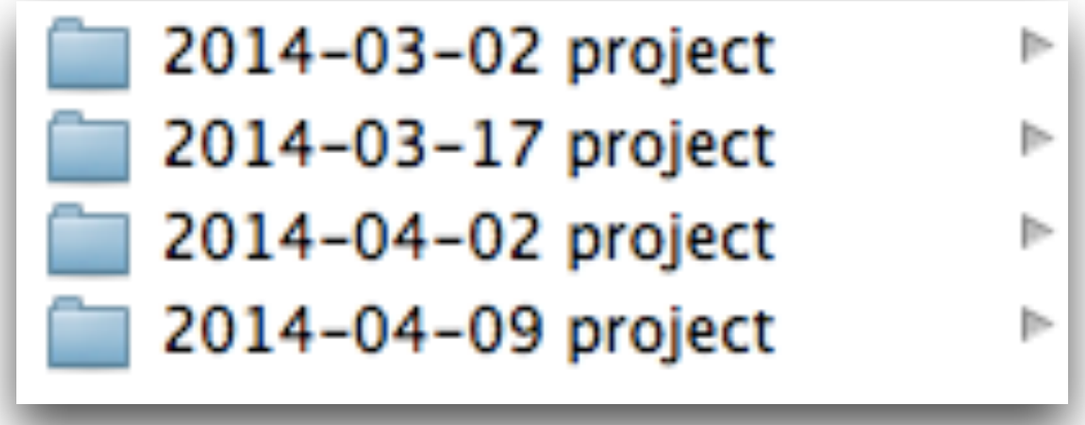
人はファイル群を管理するときにディレクトリ (フォルダ)にまとめる。
さらに、そのファイル群の**変更**と**履歴**が重要視される場合がある。

ログ管理法:

```
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
```

- ログに変更の履歴を残す

日付記入法:



- ディレクトリに日付を残し丸々コピーを残す

従来のファイル管理方法 | 欠点

- 管理にコストが掛かり過ぎる (日付の確認、ディレクトリコピーなど)
- さらにコストとリスクは管理期間が長さに比例する (ログ情報肥大化、ハードディスクの圧迫)
- 管理に強制力がない (ログのつけ忘れ、コピーのしわすれなど)

「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、失敗した場合は破棄できるような実験的な変更をしてみたい

引用：Git逆引き入門

「Git」による管理 | **管理対象**

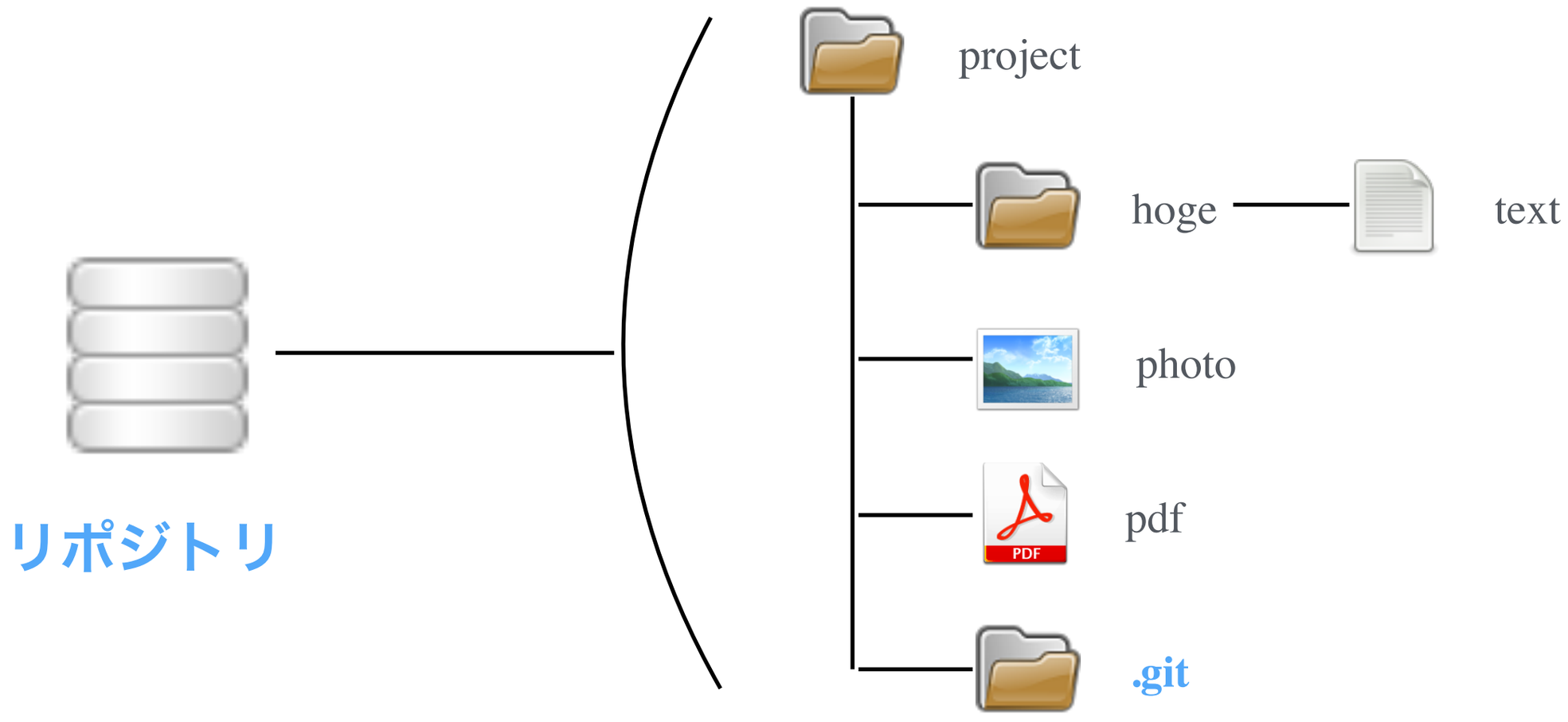
データならば**どんなもの**でも管理可能である。



- 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

- 差分 (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**を管理するための.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ディレクトリによって差分は管理される



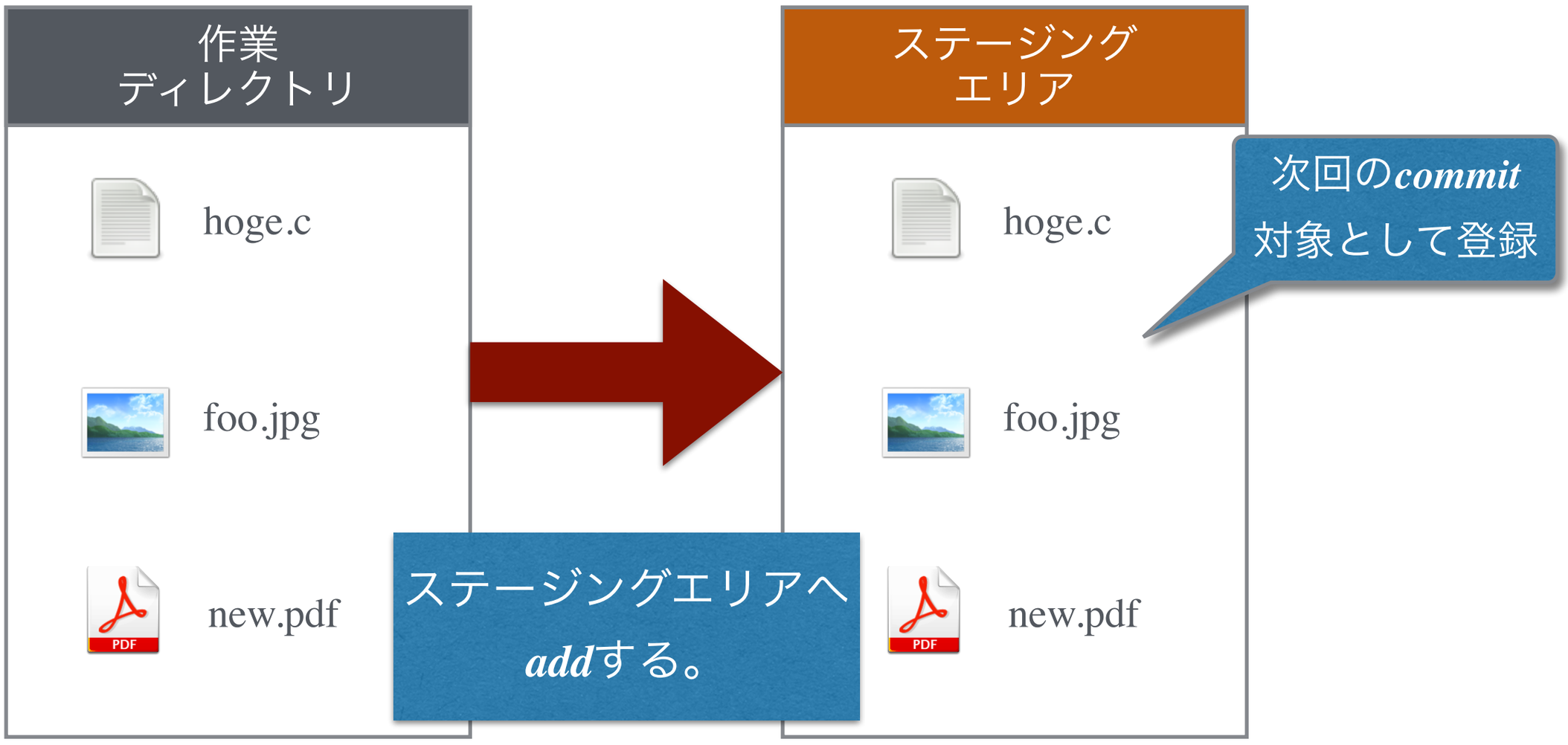
「Git」による管理 | **データ領域**

Gitの差分管理のための３つのデータ領域

作業 ディレクトリ	ステージング エリア	**Git** ディレクトリ
作業を行う ディレクトリ	変更対象を扱う 領域	差分を保存している ディレクトリ

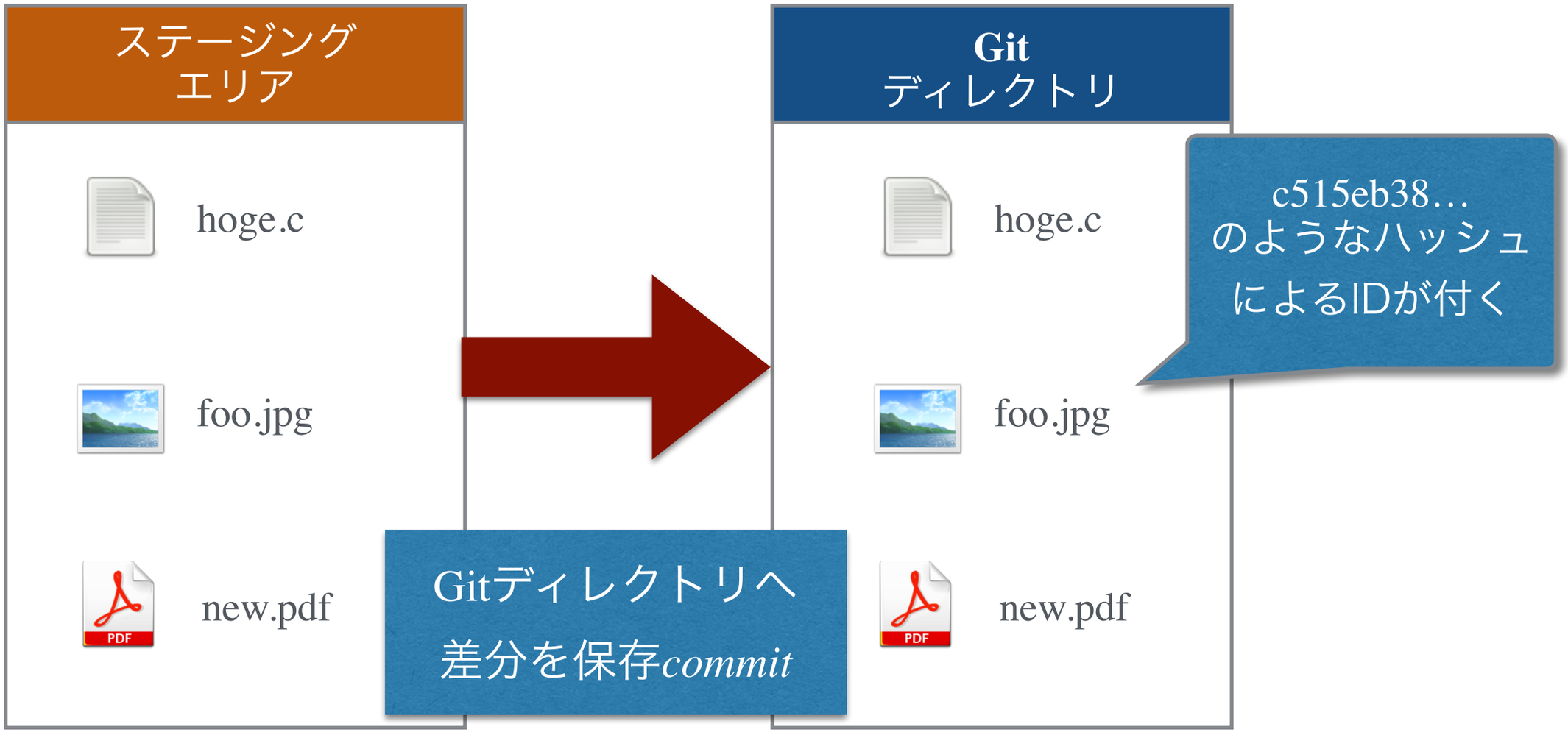
「Git」による管理 | **データ領域**

- 変更対象 (新規作成、移動、名前の変更、削除など) のファイルは、addコマンドによりステージングエリアへと登録される



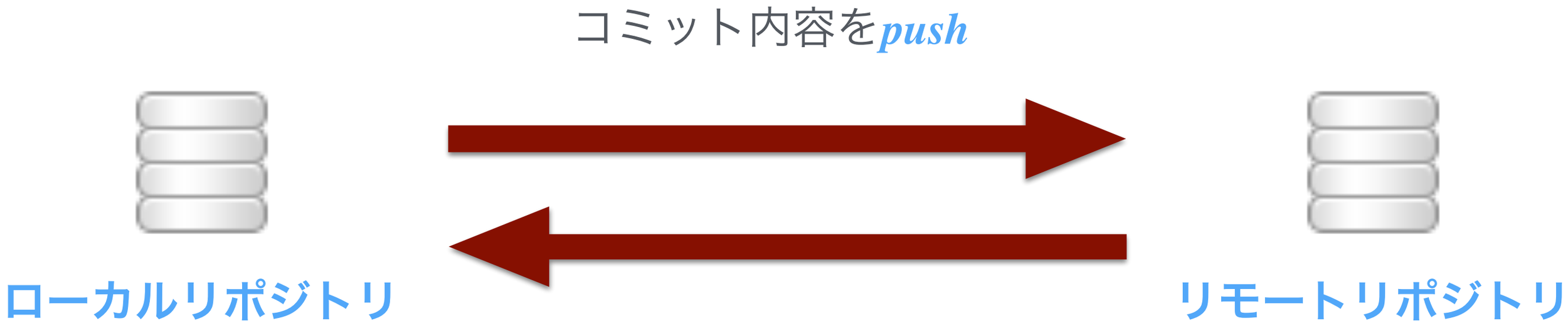
「Git」による管理 | データ領域

- 差分の確定は、commitコマンドを使用する。確定することを**コミット**と言う
- コミット内容には、**変更理由のコメントが求められる**
- また、自動的に**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

- ユーザ個人が作成・クローンしたローカル環境のリポジトリを**ローカルリポジトリ**と呼ぶ
- バックアップ・他人との共有目的で作成される、サーバーなどに設置されるリポジトリを**リモートリポジトリ**と呼ぶ



「Git」による管理 | リモートリポジトリ

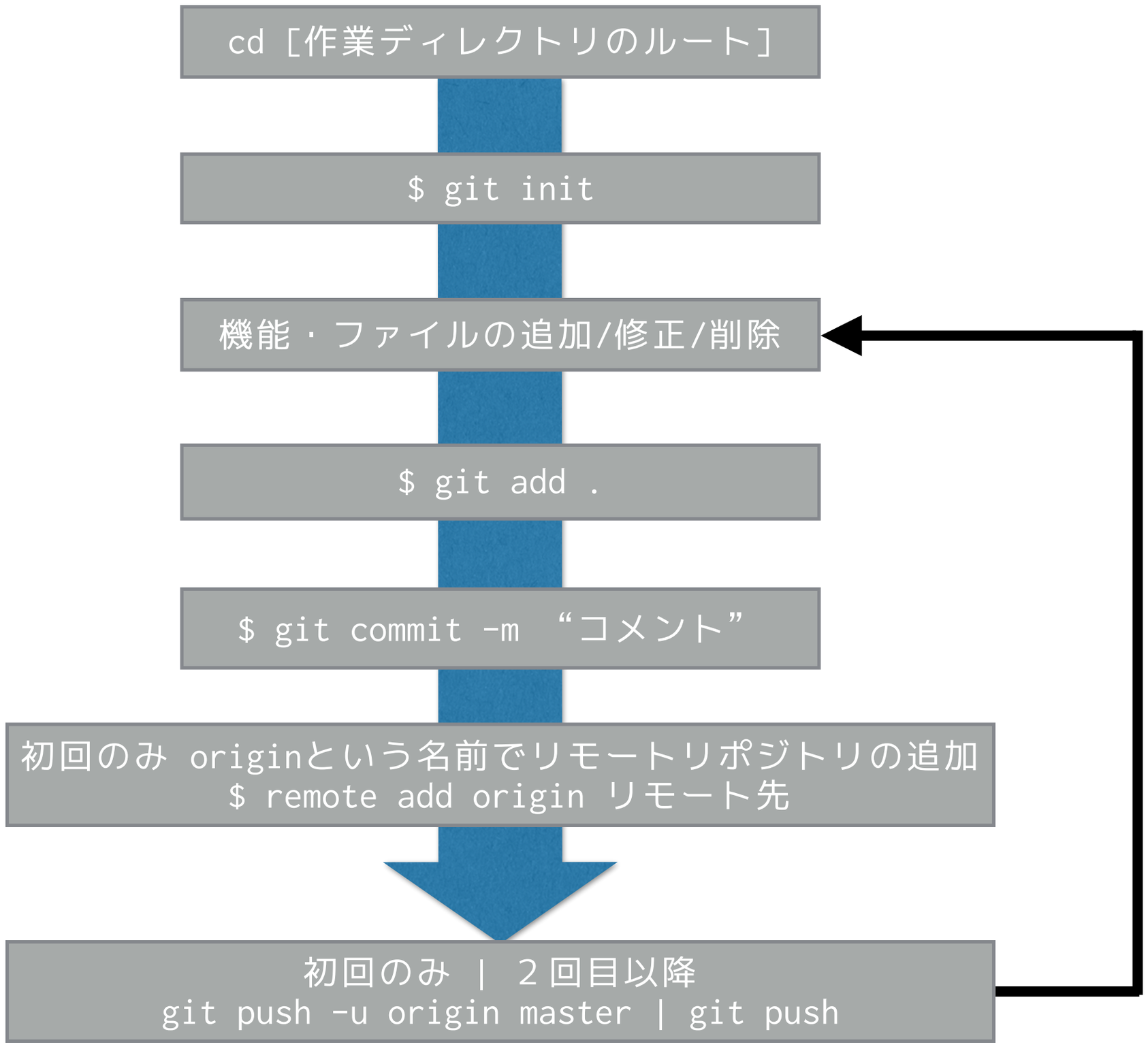
- リモートリポジトリ (サーバー)を用意・管理・維持するのは、**高コスト**である
- さらにGitのために使いやすいWebクライアントを準備する必要がある
- そのような負担を担い、リモートリポジトリを提供してくれるサービスを**Gitホスティングサービス**と呼ぶ。

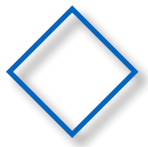
「Git」による管理 | Gitホスティングサービス

- **ソーシャルコーディング**がスローガン
- オープンソースのプロジェクトが非常に多く存在する
- 他人のコードが参考として簡単に見ることができる

- **プライベートリポジトリ**が作り放題
- Github (プライベートリポジトリは、課金対象)と対象的に、外部公開したくないリポジトリを自由に置くことができる

「Git」による管理 | Gitのライフサイクル





まとめ

- 従来のファイル管理方法の欠点を学んだ
- 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/