

プログラミング応用

<http://bit.ly/ouyou3d>

ソースコードの管理 (2)

ソフトウェアと知的財産権

前期 第6週

2018/5/22

本日は・・・

- ▶ (改めて) バージョン管理システム
- ▶ ブランチの基礎
- ▶ バージョン管理システムを用いたソースコードの配付
- ▶ バージョン管理システムを用いた共同開発
- ▶ ソフトウェア（ソースコード）と著作権

本日は・・・

- ▶ (改めて) バージョン管理システム
- ▶ ブランチの基礎
- ▶ バージョン管理システムを用いたソースコードの配付
- ▶ バージョン管理システムを用いた共同開発
- ▶ ソフトウェア（ソースコード）と著作権

バージョン管理システム

リポジトリに変更履歴を残していく



```
#include <stdio.h>  
  
int main(void)  
{  
    return 0;  
}
```

コミット1
(新規追加)

```
#include <stdio.h>  
  
int main(void)  
{  
    printf("Hello");  
    return 0;  
}
```

コミット2
(Hello出力)

```
#include <stdio.h>  
int add(int x)  
{  
    return x+10;  
}  
int main(void)  
{  
    printf("Hello");  
    return 0;  
}
```

コミット3
(関数add追加)

リポジトリ
(今回は「prog-st16d??」)

コミットすると変更内容, 変更時間, コメントなど
が自動的に記録される

Gitによるバージョン管理

▶ ステージング…バージョン管理するファイルを指定

◆ 基本コマンド

```
$ git add ファイル名
```

◆ 全てファイルを管理する場合

```
$ git add -A
```

▶ コミット…ステージングしたファイルの変更を記録

◆ 基本コマンド

```
$ git commit -m "コメント"
```

◆ コメントは後から見ても分かるような内容にする

例：課題1完了, 関数addの計算処理を修正

Gitによるバージョン管理

▶ プッシュ…GitHubにコミット内容を同期

◆基本コマンド (origin masterの意味は次週)

\$ git push origin master

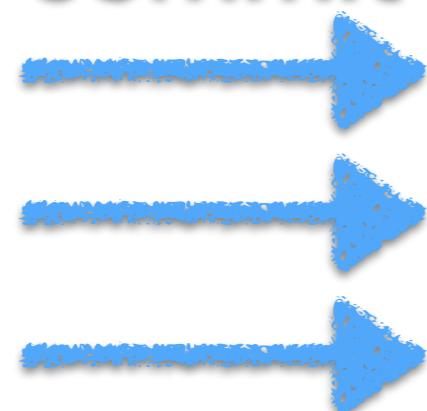
①新規作成時に

add

```
#include <stdio.h>
int add(int x)
{
    return x+10;
}
int main(void)
{
    printf("Hello");
    return 0;
}
```

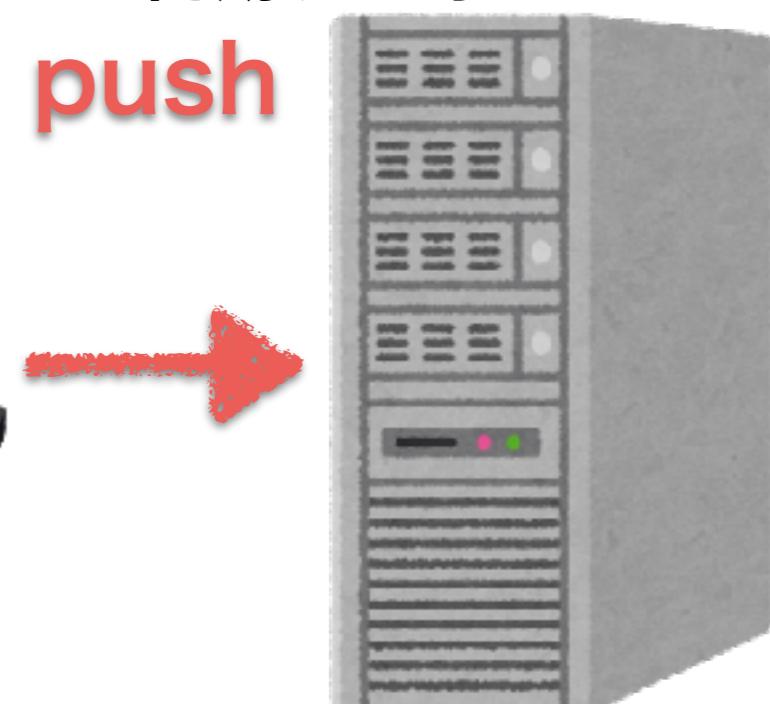
②変更履歴を残す度に

commit



③GitHubと同期する時に

push



リポジトリ

(今回は「prog-st16d??」)

GitHub

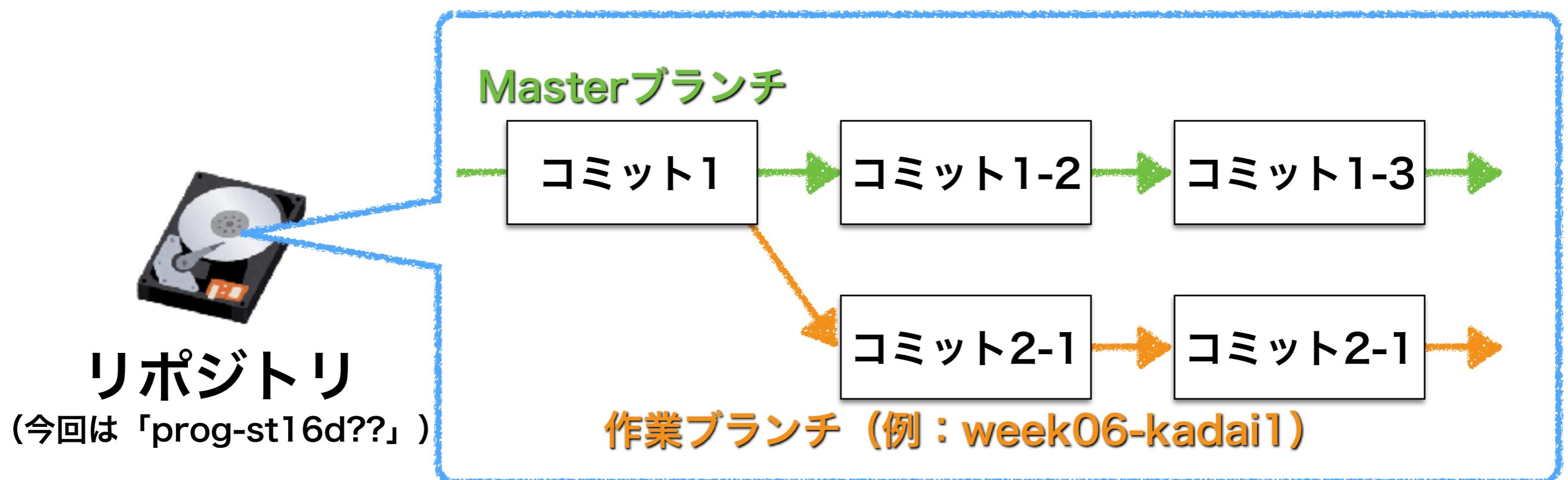
本日は・・・

- ▶ (改めて) バージョン管理システム
- ▶ ブランチの基礎
- ▶ バージョン管理システムを用いたソースコードの配付
- ▶ バージョン管理システムを用いた共同開発
- ▶ ソフトウェア（ソースコード）と著作権

ブランチとは

コミットを枝分かれさせる機能

- ▶ ブランチ … 分岐した枝のこと
- ▶ Masterブランチ … デフォルトのブランチ
- ▶ 作業ブランチ … Master以外のブランチ

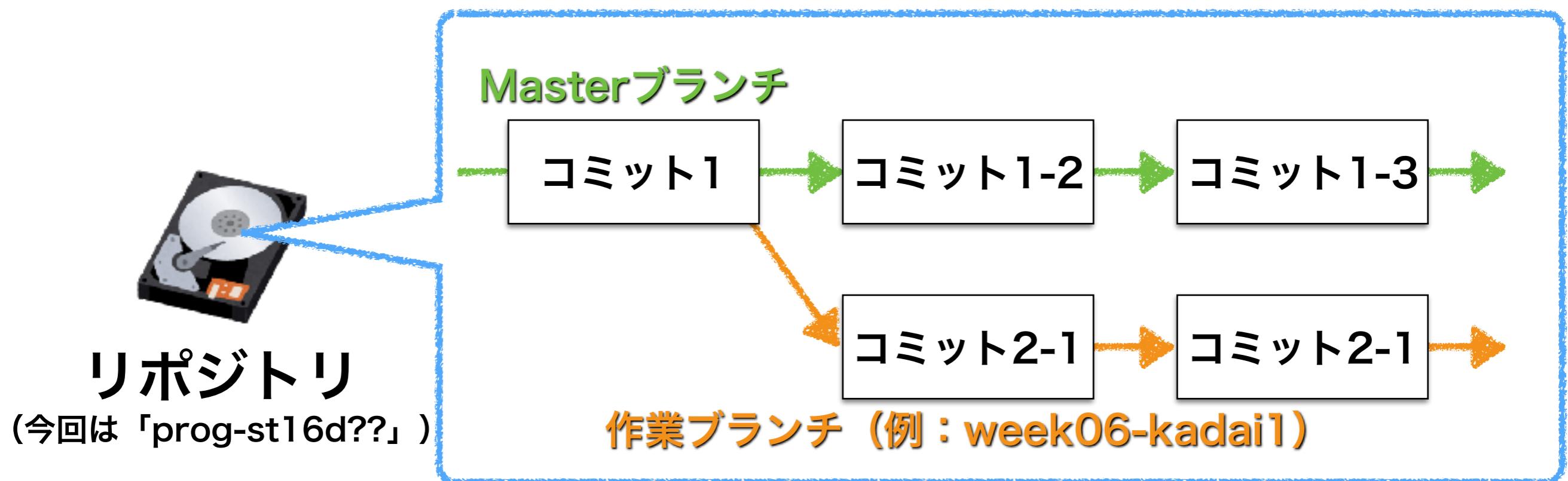


ブランチの利点

ブランチを切り替えてのコミットしても、

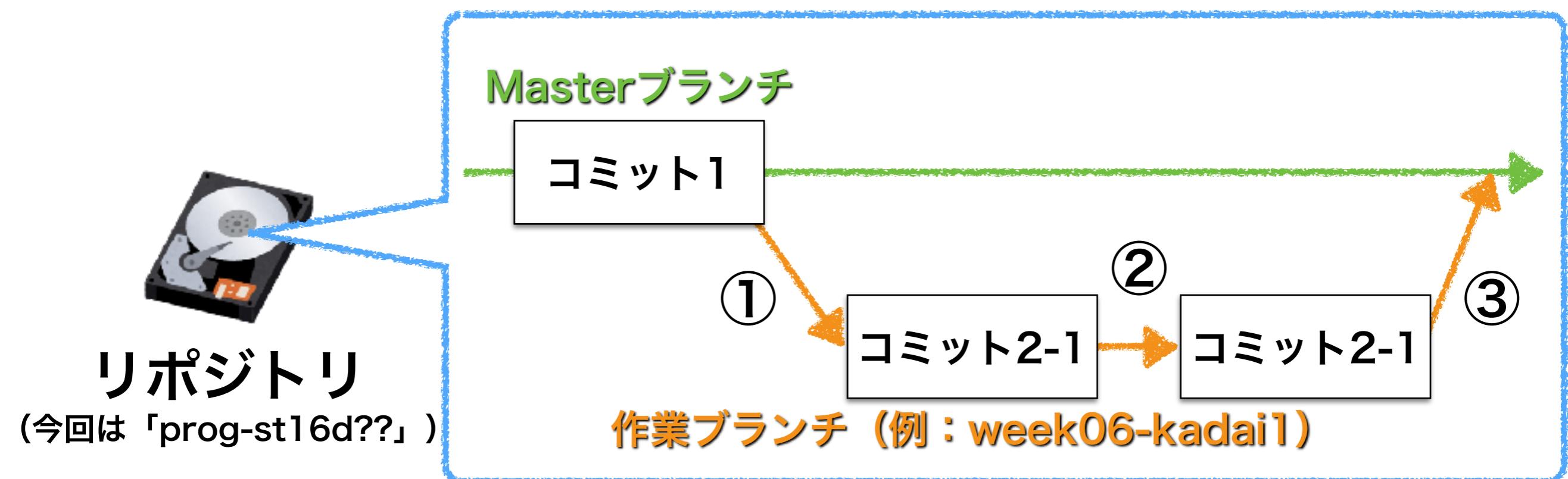
他のブランチには影響しない

- ▶ 例えば、「week06-kadai1」ブランチでコミットしても、
Masterブランチは何も変更されない
- ▶ 「week06-kadai」ブランチでの作業中に、エラーがとれなくなつたとしても、Masterブランチに戻れば元の状態に復元できる



ブランチを利用した作業の流れ

- ① 新しい機能を作る時/新しい課題に取り組む時に、
ブランチを作り、新しいブランチに移動する
- ② 新しいブランチでadd/commitを繰り返しながら作業を進める
- ③ 完成したらMasterブランチにマージする



Gitによるブランチ操作

▶ 新しいブランチの作成

```
$ git branch ブランチ名
```

▶ ブランチの切り替え

```
$ git checkout ブランチ名
```

▶ ブランチを一覧表示する

```
$ git branch
```

▶ ブランチをマージする

```
$ git checkout マージ先のブランチ名
```

```
$ git merge マージ元のブランチ名
```

Gitによるブランチ作業例

① 新しいブランチを切る（作る）

作業内容が分かるブランチ名を付けると良い

```
$ git branch week06-kadai1
```

② 新しいブランチに移動する

```
$ git checkout week06-kadai1
```

③ 完成したらマージする

まずはMasterへ移動する

```
$ git checkout master
```

ブランチweek06-kadai1をMasterへマージする

```
$ git merge week06-kadai1
```

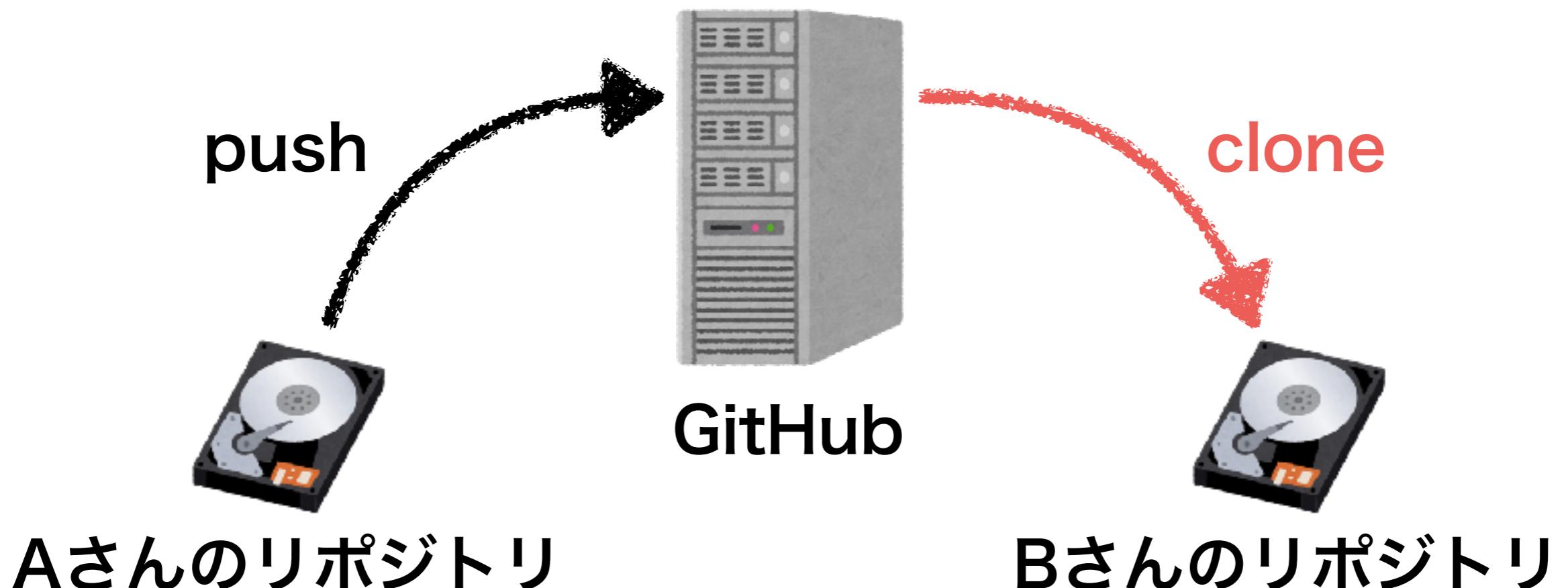
本日は・・・

- ▶ (改めて) バージョン管理システム
- ▶ ブランチの基礎
- ▶ バージョン管理システムを用いた
ソースコードの配付
- ▶ バージョン管理システムを用いた共同開発
- ▶ ソフトウェア（ソースコード）と著作権

複数人で使う場合（ソースコード配付）

cloneを使って他のリポジトリの複製を作る

【例】AさんからBさんへソースコードを配付する



複数人で使う場合（ソースコード更新）

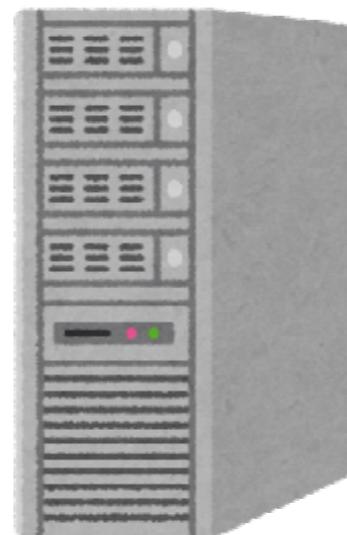
`pull`を使ってリポジトリを最新の状態にする

【例】Aさんがpushした内容を、Bさんに反映する

まずAさんが
pushする



Aさんのリポジトリ



GitHub

そしてBさんが
pullする



Bさんのリポジトリ

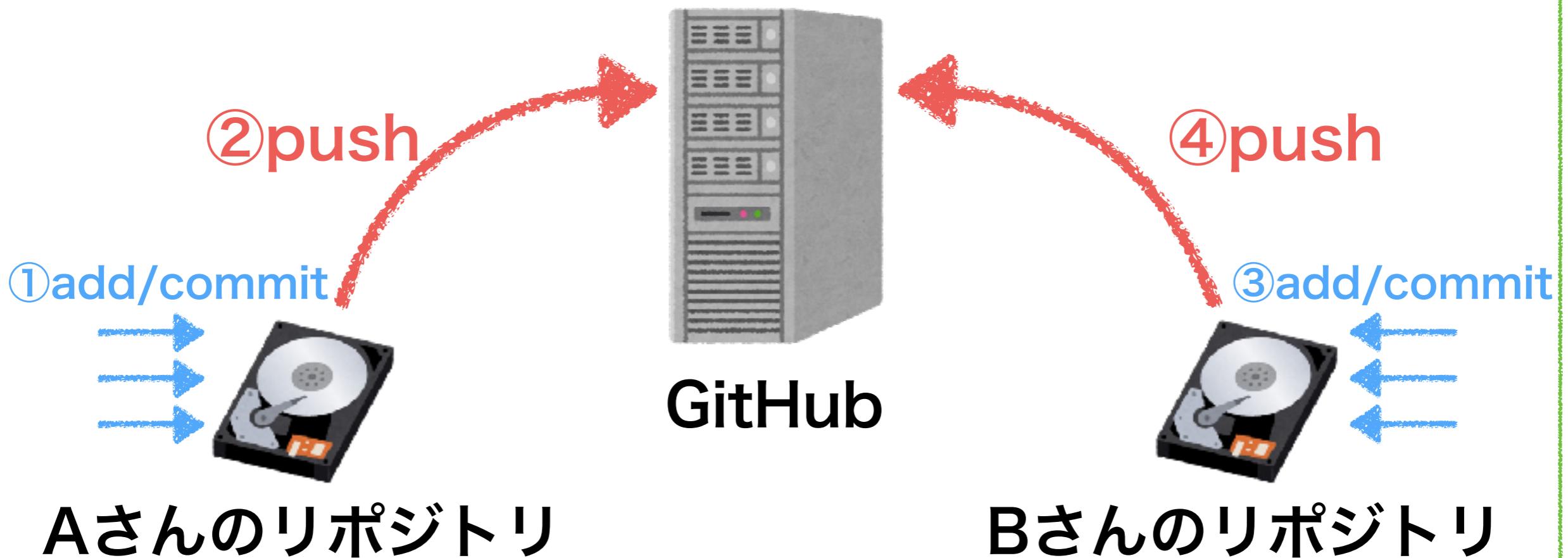
本日は・・・

- ▶ (改めて) バージョン管理システム
- ▶ ブランチの基礎
- ▶ バージョン管理システムを用いた
ソースコードの配付
- ▶ バージョン管理システムを用いた共同開発
- ▶ ソフトウェア（ソースコード）と著作権

複数人で共同開発する場合

それぞれがローカルリポジトリにadd/commitして、
GitHubにPushしながら作業をする

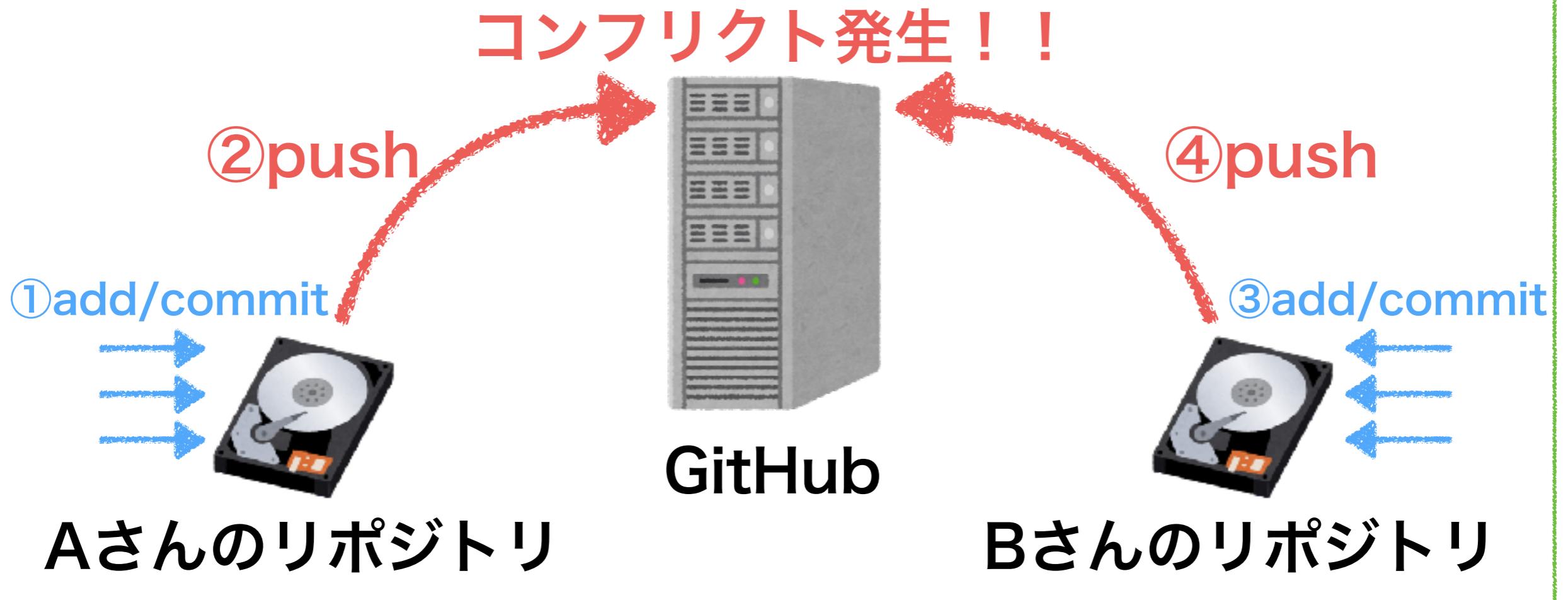
【例】共同開発中のAさんとBさんがそれぞれがpushする



コンフリクト

複数人が同じファイルを編集すると、
競合（コンフリクト）が起こる場合がある

【例】AさんとBさんが同じファイルを編集してpushした



お互に話し合ってコンフリクトを直すまでpushできない

本日は・・・

- ▶ (改めて) バージョン管理システム
- ▶ ブランチの基礎
- ▶ バージョン管理システムを用いたソースコードの配付
- ▶ バージョン管理システムを用いた共同開発
- ▶ ソフトウェア（ソースコード）と著作権

著作権とは

▶ 知的財産権

知的な創作活動から生産されたものを他人が無断で使用して利益を得ることができないようにするための権利

▶ 産業財産権 … 特許や商標など

▶ 著作権 … 著作物を使った人の権利

著作物とは

- ▶ “思想又は感情を創作的に表現したものであって、文芸、学術、美術又は音楽の範囲に属するもの”
- ▶ 著作物には、オリジナリティのある**プログラムコード**も含まれる（ソフトウェアを構成する画像ファイルなども対象となる）

ソースコードが公開されている（オープンソース）ものは…
利用目的や再配布など指定された条件のもとで使う必要がある

今回の演習の前に・・・

- ▶ 本日の内容は皆さんのPCにインストールされているCygwinで作業してください
- ▶ まずは、第2週の内容から進めてください
(必要ならば第1週の内容から)

【練習6-1】

次のように課題用のブランチを作成してください。

1. 現在はMasterブランチを使用していることを確認する
(使用中ブランチに*マークが付く)

```
$ git branch
```

2. 練習用ブランチを作成する

```
$ git branch week06-kadail
```

3. 「week06-kadail」というブランチができていることを確認する
(week06-kadailブランチが新たにできているが、
*マークはまだMasterについていることを確認する)

```
$ git branch
```

【練習6-2】

次のように練習用のブランチに切り替えてください。

1. 「week06-kadai1」ブランチへ切り替える

```
$ git checkout week06-kadai1
```

2. 正しくブランチへ移動できていることを確認する

(*マークがweek06-kadai1についていることを確認する)

```
$ git branch
```

【課題6-1】

練習で作成した「week06-kadai1」ブランチで、
次のようにプログラムを作成してください。

1. 自分のリポジトリ以下に、「week06」フォルダを作成する
2. week06フォルダ内に「hello.c」というファイル作り、
標準出力に “Hello, World.”と出力するCプログラムを作成する

次のスライドに続く

【課題6-1】

「week06-kadai1」 ブランチで、pushして
GitHubのWebサイトで自分のリポジトリの情報を
確認してください。

1. 「week06-kadai1」 ブランチのみpushする

```
$ git add -A  
$ git commit -m "hello.c追加"  
$ git push origin week06-kadai1
```

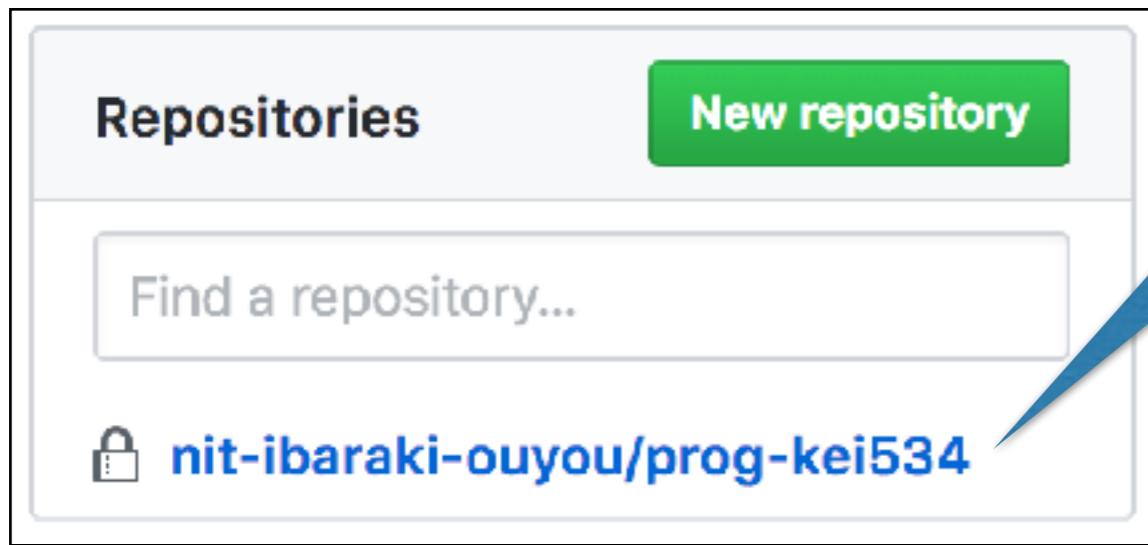
originはリモートリポジトリの名前
(GitHubが指定している)

pushするブランチを指定する

次のスライドに続く

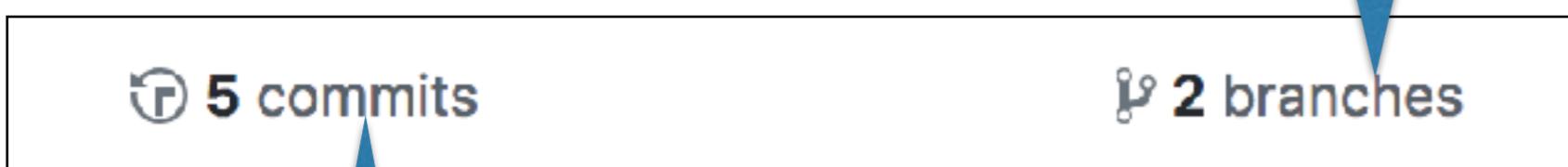
【課題6-1】

GitHubのWebサイトで自分のリポジトリの情報を確認してください。



github.comでログインして、この授業のリポジトリをクリック

リポジトリ内のブランチが確認できます。
masterとweek06-kadai1のブランチがあることを確認してください。



commitの内容が、ブランチごとに閲覧できます。
week06-kadai1のブランチのcommit内容を確認してください。

【課題6-2】

次のように課題用のブランチをマージしてください。

1. 1つ上のフォルダに移動

```
$ cd ..
```

2. Masterブランチに移動

```
$ git checkout master
```

3. Masterブランチに移動できたことを確認

(*マークがMasterについているか確認)

```
$ git branch
```

4. ファイル一覧を表示 (Masterに移動するとweek06ディレクトリが消えるか確認)

```
$ ls
```

5. week06-kadai1ブランチをMasterブランチにマージ

(かならずMasterブランチ上で操作すること!)

```
$ git merge week06-kadai1
```

6. 再度ファイル一覧を表示し、week06ディレクトリが現れたか確認

```
$ ls
```

【課題6-3】

次のように新しいブランチを作成して、プログラムを作り、リポジトリにpushして、マージしてください。

- ▶ ブランチ名は「week06-kadai3」
- ▶ ディレクトリ「week06」以下に「sum.c」というファイル名で、以下のCプログラムを作成する
 - ◆ 配列宣言する `int array[7] = {5, 3, 2, 7, 8, 4, 1}`
 - ◆ `main()`で、上記の配列の合計値を求めて出力する
- ▶ 完成したら、`add`, `commit`, `push`をしてGitHubへ反映する
- ▶ このブランチをMasterへマージする

小テストについて

次回の授業で、前期中間までの内容を含んだ小テストを実施します。