

# Gitを使ったバージョン管理システム

---

IW-13A-167

朝倉真琴 島田朋幸

# 目次

1. バージョン管理とは
2. Gitの特徴
3. 集中型と分散型の違い
4. 主な流れ
5. Git基本の用語/コマンド

# バージョン管理とは

- バージョン管理システムとは、コンピュータ上で作成、編集されるファイルの変更履歴を管理するシステムです。
- リポジトリと呼ばれるファイル管理用のディレクトリをひとつ以上持ち、そこからファイルを「コミット」したり「チェックアウト」したりすることにより履歴を更新していく。

# Gitの特徴

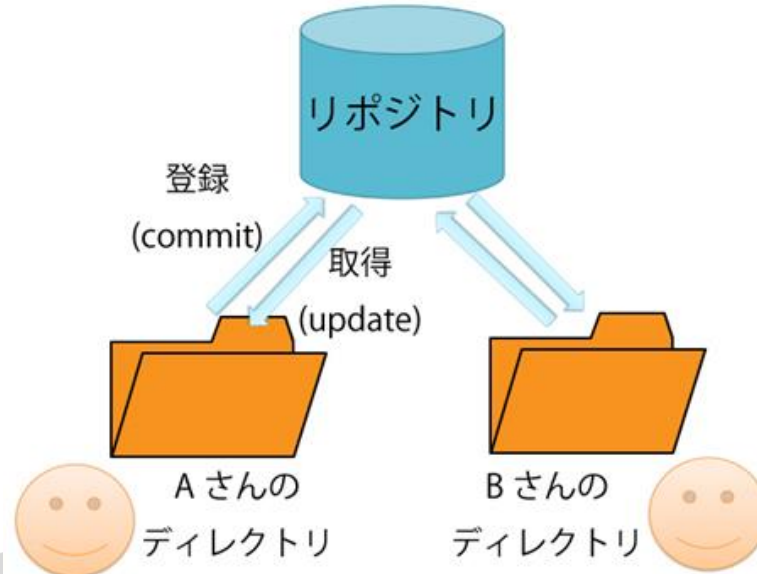
- 分散型バージョン管理
- 履歴管理の柔軟性が高い
  - ・一度コミットしたものは参照されなくなってもしばらくは残り続ける
  - ・過去にさかのぼって履歴を変更することができる
- ブランチの作成/マージが容易にできる

# 集中型と分散型の違い

---

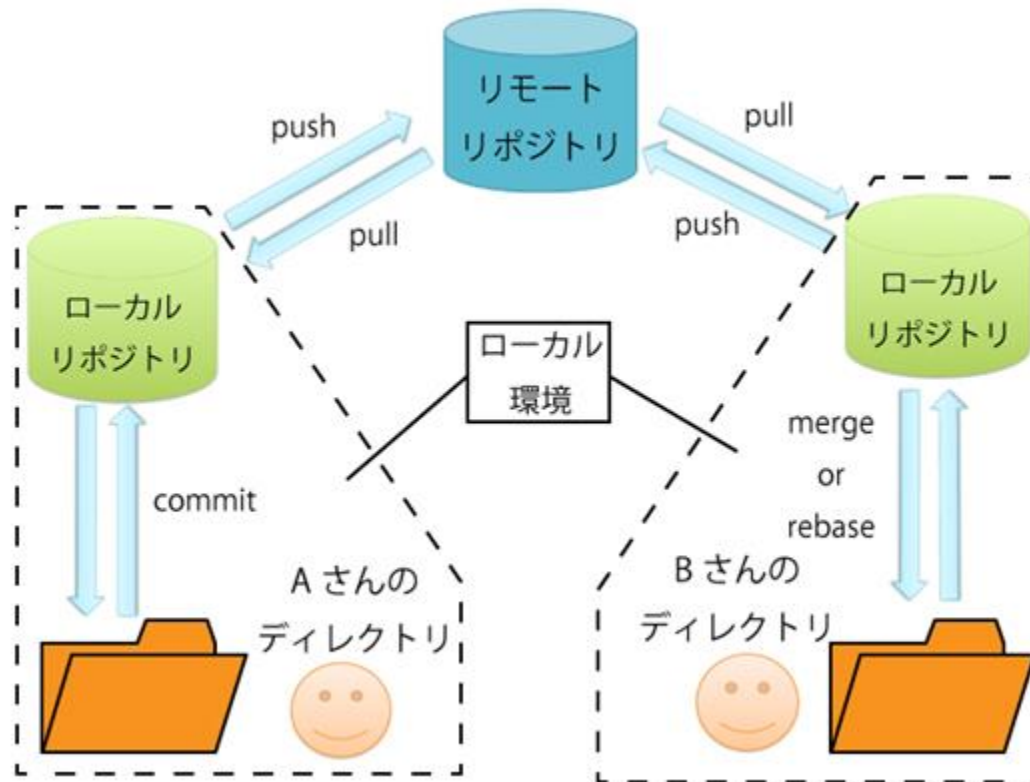
# 集中型バージョン管理とは

- 他のユーザーが行った任意の変更を中央サーバーから取得する。
- 変更を行い、そのコードが正しく動作することを確認する。

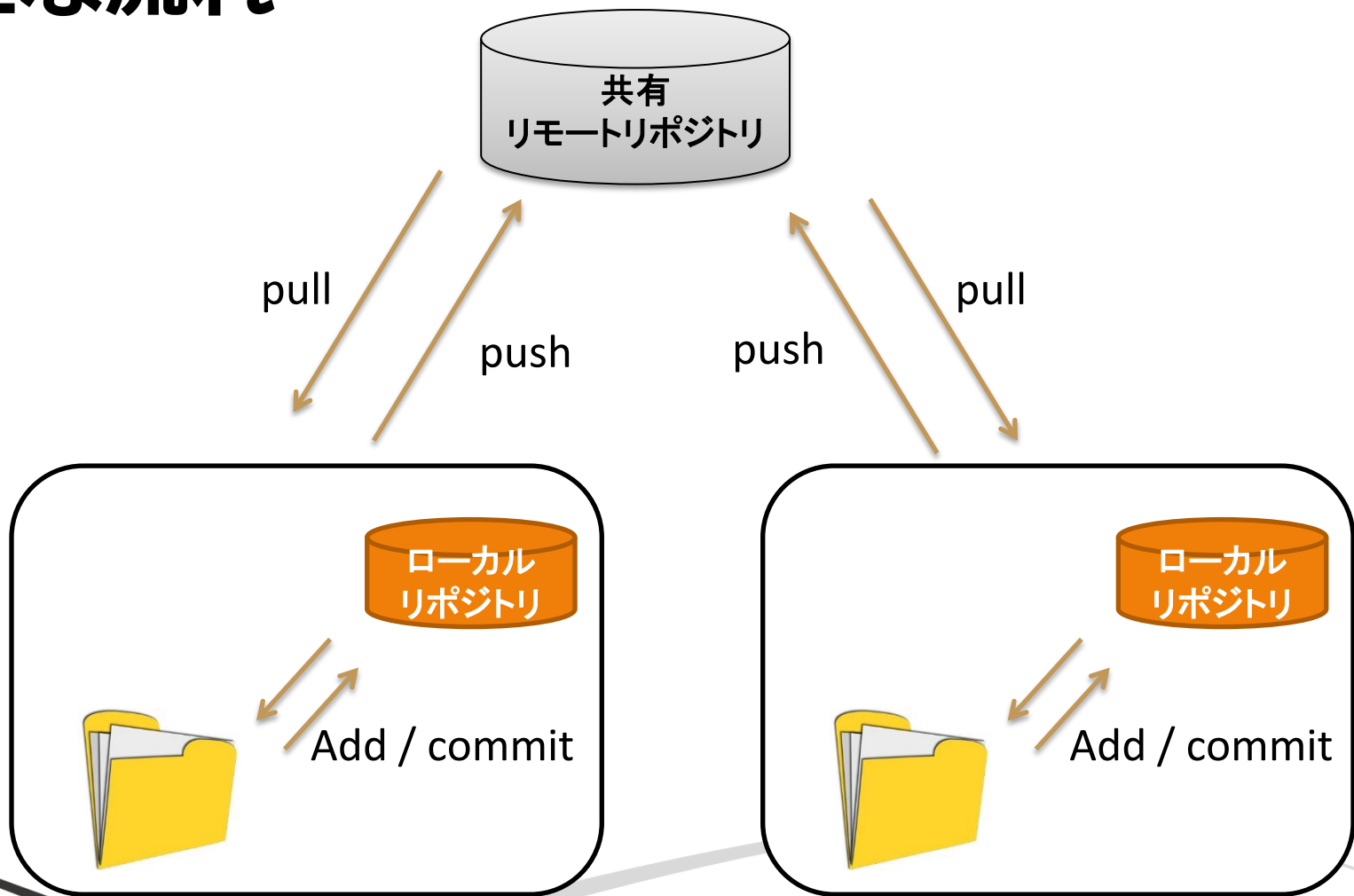


# 分散型バージョン管理とは

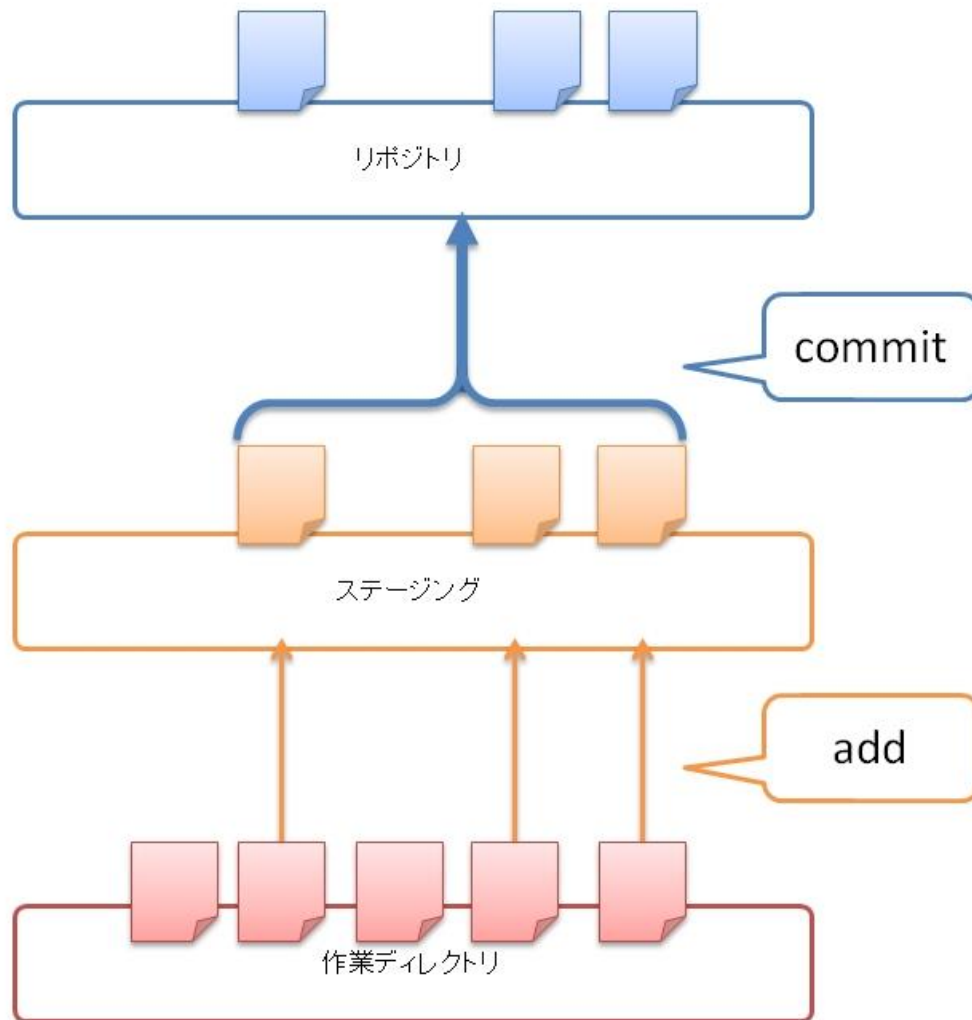
- リポジトリが複数存在する
- プロジェクトメンバーそれぞれが自分のリポジトリを持つ
- ローカルリポジトリへの操作に関してはネットワーク通信を必要としない。



# 主な流れ







# Git基本の用語/コマンド

---

# リポジトリ Repository

履歴管理を行う場所

- ・ リモートリポジトリ  
…サーバーにあるリポジトリ。
- ・ ローカルリポジトリ  
…自分のPC上にあるリポジトリ。

# init

リポジトリを作成する。

`$mkdir [directory name]`

…作業領域とするディレクトリを作成する

`$cd [directory name]`

…作業領域に移動する

`$git init`

…このコマンドを実行すると、カレントディレクトリに  
.gitフォルダが作成され、プロジェクトの管理を開始します。

MINGW64:/c/Users/nhs40065/Desktop/myweb

nhs40065@shimada MINGW64 ~/Desktop

\$ mkdir myweb

nhs40065@shimada MINGW64 ~/Desktop

\$ cd myweb

nhs40065@shimada MINGW64 ~/Desktop/myweb

\$ git init

Initialized empty Git repository in C:/Users/nhs40065/Desktop/myweb/.git/

nhs40065@shimada MINGW64 ~/Desktop/myweb (master)

\$ .....

# clone

リモートリポジトリからプロジェクトをローカルに複製する

`$git clone [gitのURL] 又は [ディレクトリ名]`

…リポジトリの履歴情報を含めてリポジトリを複製する

```
MINGW64:/c/Users/nhs40065/Desktop/github

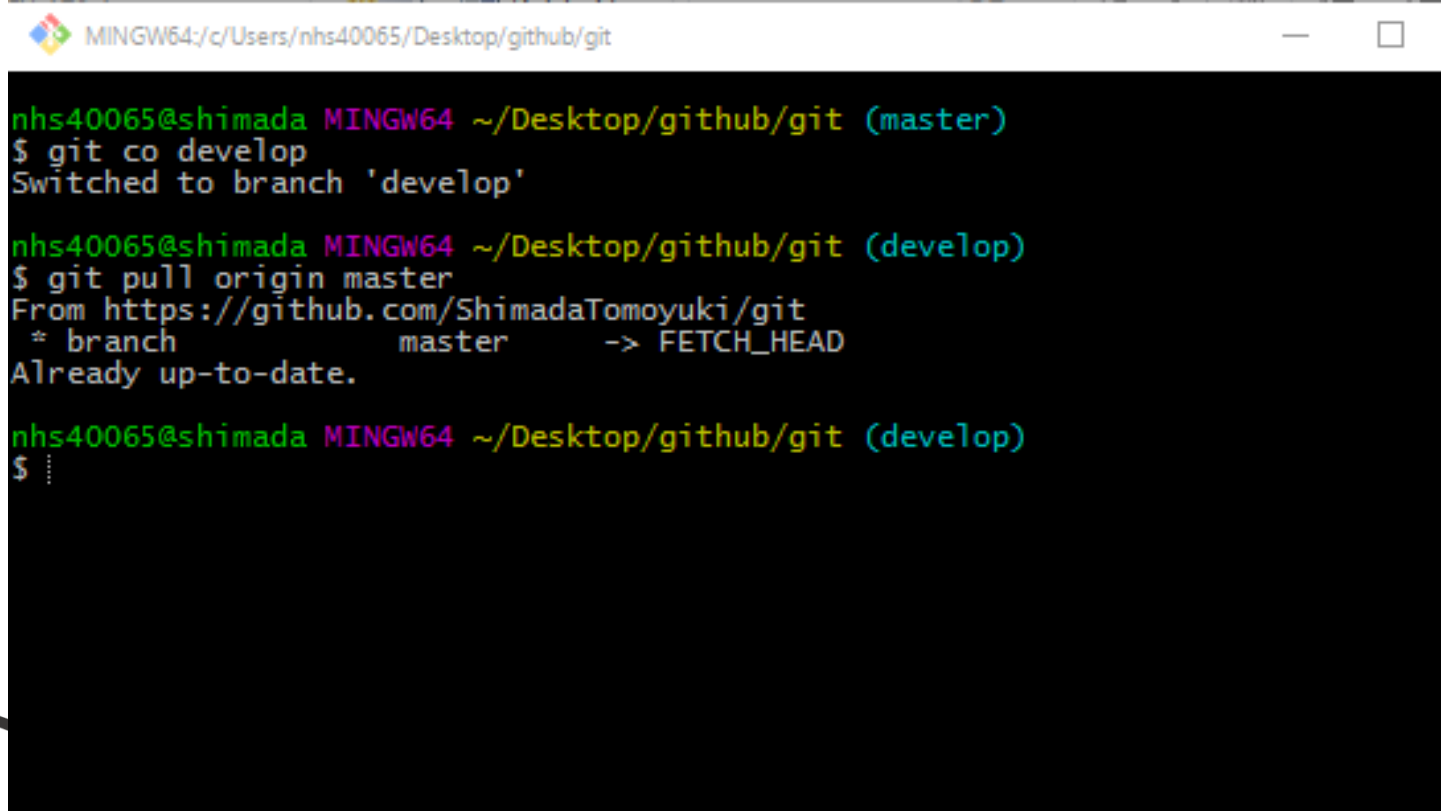
nhs40065@shimada MINGW64 ~/Desktop/github
$ git clone https://github.com/ShimadaTomoyuki/git
Cloning into 'git'...
remote: Counting objects: 27, done.
remote: Total 27 (delta 0), reused 0 (delta 0), pack-reused 27
Unpacking objects: 100% (27/27), done.

nhs40065@shimada MINGW64 ~/Desktop/github
$ :
```

# pull

リモートリポジトリの変更をローカルリポジトリに反映させる

`$git pull [remote] [branch]`

A screenshot of a Windows terminal window with a black background and white text. The window title bar shows the path 'MINGW64:/c:/Users/nhs40065/Desktop/github/git'. The terminal output shows a user switching to the 'develop' branch and then pulling from the 'origin master' branch, which results in 'Already up-to-date.'.

```
mingw64 ~/Desktop/github/git (master)
$ git co develop
Switched to branch 'develop'

mingw64 ~/Desktop/github/git (develop)
$ git pull origin master
From https://github.com/ShimadaTomoyuki/git
 * branch            master       -> FETCH_HEAD
Already up-to-date.

mingw64 ~/Desktop/github/git (develop)
$ ..
```

# add

ステージング領域に追加し、次回のコミット対象にする

`$git add .`

…カレントディレクトリ以下のすべての変更がステージング領域に追加されます。

`$git add [file name]`

…指定されたファイルのみ追加されます。

スペース区切りで複数ファイル指定することもできます。



```
MINGW64: c:/Users/nhs40065/Desktop/github/git

nhs40065@shimada MINGW64 ~/Desktop/github/git (develop)
$ git status
On branch develop
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)
  ① modified:   test.txt

no changes added to commit (use "git add" and/or "git commit -a")

nhs40065@shimada MINGW64 ~/Desktop/github/git (develop)
$ git add . ②

nhs40065@shimada MINGW64 ~/Desktop/github/git (develop)
$ git status
On branch develop
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)
  ③ modified:   test.txt

nhs40065@shimada MINGW64 ~/Desktop/github/git (develop)
$
```

- ①git statusで変更点の確認  
Changes not staged for commit:  
ステージングされていない
- ②git add . 実行  
ステージング領域に追加
- ③もう一度git statusを実行して、  
追加されたことを確認

# commit

インデックスに登録してある変更対象をローカルリポジトリに反映する。

```
$git commit -m '<message>'
```

…<message>をコミットメッセージとしてローカルリポジトリにコミットします。

```
$git commit -a -m '<message>'
```

…オプション -a をつけて実行すると、変更されたファイルを自動検出して、コミットできる。

```
MINGW64:~/Desktop/github/git
nhs40065@shimada MINGW64 ~/Desktop/github/git (develop)
$ git commit -m 'test.txt edited'
[develop 0d099e2] test.txt edited
1 file changed, 1 insertion(+), 1 deletion(-)
nhs40065@shimada MINGW64 ~/Desktop/github/git (develop)
$ ..
```

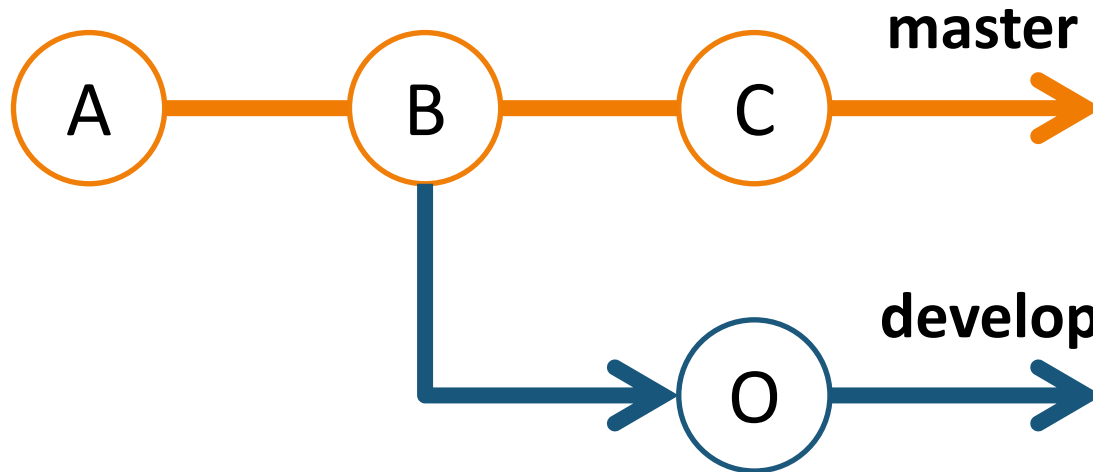
# branch/checkout

branchとは履歴の流れを分岐して記録するもの。  
他のブランチの影響を受けないため、並行作業が可能になる

`$git branch [branch名]`

`$git checkout [branch名]`  
…作業するブランチに切り替える

`$git checkout -b [branch名]`  
…上の2つの操作を同時にやるコマンド



---

 MINGW64:/c:/Users/nhs40065/Desktop/github/git

```
nhs40065@shimada MINGW64 ~/Desktop/github/git (master)
$ git branch develop

nhs40065@shimada MINGW64 ~/Desktop/github/git (master)
$ git checkout develop
Switched to branch 'develop'

nhs40065@shimada MINGW64 ~/Desktop/github/git (develop)
$ ..
```

# Push

ローカルリポジトリの変更をリモートリポジトリに反映させる。

`$git push origin [送信するbranch] : [送信先branch]`

…送信先branchが同じ場合は省略可能。

また、送信先リポジトリにbranchが存在しない場合はそのbranchが作成される。

```
MINGW64: c:/Users/nhs40065/Desktop/github/git
nhs40065@shimada MINGW64 ~/Desktop/github/git (develop)
$ git commit -m 'test.txt edited'
[develop 0d099e2] test.txt edited
1 file changed, 1 insertion(+), 1 deletion(-)

nhs40065@shimada MINGW64 ~/Desktop/github/git (develop)
$ git push origin develop
Counting objects: 6, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 529 bytes | 0 bytes/s, done.
Total 6 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 1 local objects.
To https://github.com/ShimadaTomoyuki/git
 * [new branch]      develop -> develop

nhs40065@shimada MINGW64 ~/Desktop/github/git (develop)
$ .....
```

\$git pushを実行するとリモートリポジトリに、追加したブランチが反映される。