

# 俺でもわかる Git/GitHub

## はじめに

この資料は、「著者(俺)がわかる」をベースに書いています。

なのでこの資料を読んで「わかりにくい」と思うかもしれません。

人によって理解の仕方は違うので仕方がないことです。

そういうときは、自分に合う資料を見つけてください。

Git/GitHub に関する資料はインターネット上を探せばいくらでもあります。

自分で探すということも大事です。つまり、何が言いたいかというところの資料を読んでも必ず理解できるわけではないので理解できるよう自分で努力してください、ということです。

それでは、こういう投げやりな資料でも良ければ、読んでいってくださいな。

## Git/GitHub とは

まずはじめに、Git と GitHub は別物です。どれくらい違うかというと、Java と JavaScript、インドとインドネシアくらい違います。

 <b>Git</b>  ・バージョン管理ツール 一人で開発したり 大人数で開発したり するときに起きる 困ったことを 解決してくれる	 <b>GitHub</b>  ・Git 関連サービス  Git をさらに 便利にしてくれる サービスだよ
 <b>インド</b>  英語/ヒンディー語 南アジア ニューデリー 大陸続き	 <b>インドネシア</b>  インドネシア語 東南アジア ジャカルタ 島国

# 俺でもわかる Git/GitHub

## Git とは

Git とはプログラムのソースコードやドキュメントの変更,その履歴を管理,追跡するための分散型バージョン管理ツールです。開発者は Linux カーネルの開発者でおなじみのリーナス・バルズらである。

## バージョン管理ツールとは

名前の通りです。例えば、ソースコードを書きながら「以前の状態のほうが良かった」とか「間違っちゃった」、「新しい機能を試したいからコピーしてそっちで試そう」など思うことがありますよね?そういう開発中の困ったことを解決してくれるのがバージョン管理ツールです。コマンド、または GUI による直感的な操作で、以前の状態に戻したり状態を分岐させて新しい機能を試したりということができます。

バージョン管理ツールは Git だけでなく Apache ソフトウェア財団の SVN(Apache Subversion)や The CVS Team の CVS(Concurrent Versions System)などがあります。

## バージョン管理ツールの種類

バージョン管理ツールには大きく分けて 2 種類があります。

1 つは CVS や SVN などの集中型バージョン管理システム、もう 1 つは Git の分散型バージョン管理システムです。

集中型バージョン管理システムと分散型バージョン管理システム、これらの違いを説明するにあたり覚えておいてほしい用語があります。

**リポジトリ(repository)** - ファイルの変更履歴や、ファイルそのものなどを保存する場所

**ローカルリポジトリ(local repository)** : 自分のコンピュータ内にあるリポジトリ

**リモートリポジトリ(remote repository)** : 外部のコンピュータ(サーバ)内にあるリポジトリ

**コミット(commit)** : リポジトリに履歴を登録すること

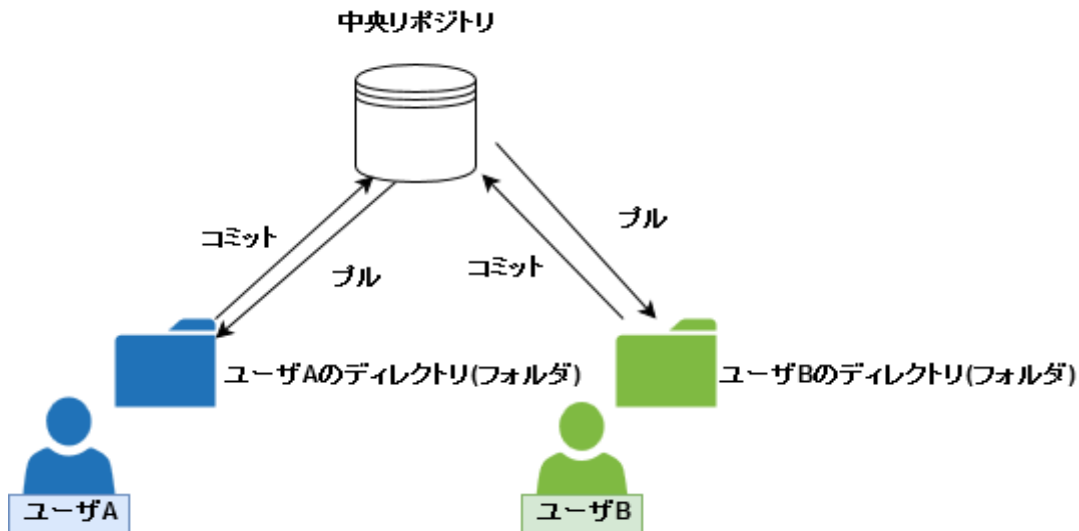
**プル(pull)** : リポジトリから履歴を取得すること

**プッシュ(push)** : ローカルリポジトリをリモートリポジトリに反映すること

# 俺でもわかる Git/GitHub

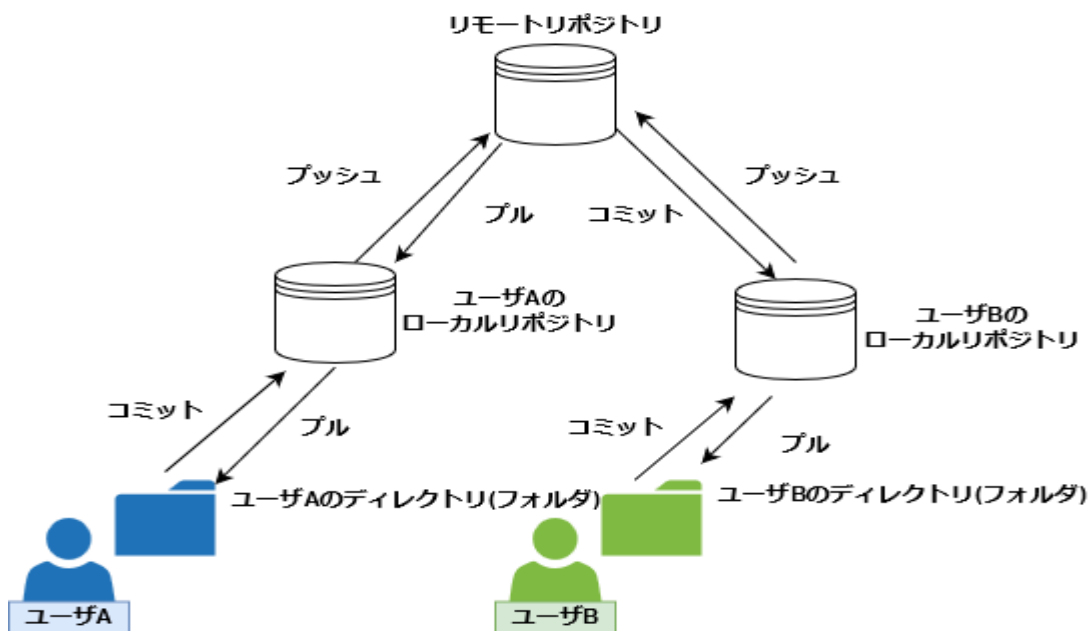
## 集中型バージョン管理システム

集中型バージョン管理システムはいわゆる「クライアント・サーバモデル」を使ったバージョン管理のことです。中央リポジトリ(リモートリポジトリ)から各クライアントがデータを持ってきて編集し、編集が終わったら中央リポジトリを更新するという管理方法です。よってリポジトリは一つとなります。



## 分散型バージョン管理システム

分散型バージョン管理システムは集中型バージョン管理システムと違いリポジトリを複数持つことができます。例えばユーザが自分のリポジトリ(ローカルリポジトリ)を持っていて普段はローカルリポジトリを用いて作業を行い、ある程度作業が完結し作業を他のユーザと共有したい時はリモートリポジトリに反映するといったことが可能です。



# 俺でもわかる Git/GitHub

## GitHub とは

その名の通り Git の Hub(ハブ：中心部)です。リモートリポジトリを運用しているサービスです。所謂、クラウドと呼ばれるもので Google Drive などの Git 版。

類似サービスに

GitLab, Bitbucket などがある。

## Git の導入

公式サイトから <https://git-scm.com/downloads> OS に応じてダウンロード・インストールを行ってください。

## Git を使おう

### 1. リポジトリを作ろう

以下のコマンドを入力するとカレントディレクトリ(今自分が開いているディレクトリ)がリポジトリとして登録されます。おそらく init は initialize(初期化)の略称です。

```
$ git init
```

### 2. ファイルをステージングしよう

Git にコミット(変更履歴を登録)するためにはコミットするファイルを登録してあげなければいけません。以下のコマンドでファイルを登録しましょう。

```
$ git add 登録したいファイル名
```

また、以下のコマンドでステージングを取り下げることができます

```
$ git reset ファイル名
```

### 3. ファイルをコミットしよう

Git にコミット(変更履歴を登録)します。以下のコマンドでファイルをコミットしましょう。コミットメッセージとはどんな変更を加えたかなどをわかりやすくするためのラベルです。書かなくともコミットはできますが、できるだけ書くようにしましょう。

```
$ git commit -m "コミットメッセージ"
```

# 俺でもわかる Git/GitHub

## 4. コミットを確認してみよう

以下のコマンドで今までのコミット履歴が確認できます。

```
$ git log
```

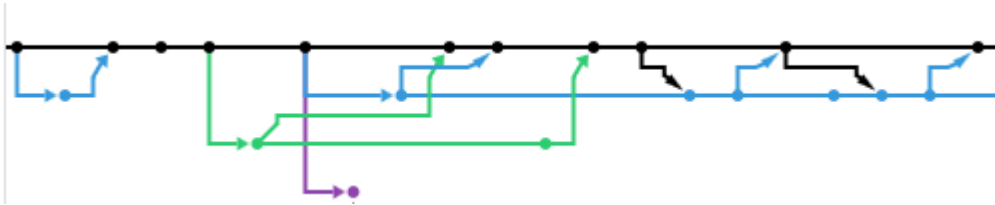
## 5. ブランチを切ってみよう

### ブランチ(Branch)とは

日本語で枝という意味です。

何かプログラムを作成していて別の機能を追加したいけどこのファイルは残しておきたいとか機能追加前のバージョンと機能追加後のバージョン両方とも残しておいて並行して作業を行いたい時に作成します。

ブランチの中でも特に主となるブランチのことを**マスターブランチ(master branch)**と呼びます。下の画像でいうと黒い線がマスターブランチです。



### ブランチを作成する

以下のコマンドでブランチを作成します。

```
$ git branch 作成したいブランチ名
```

### ブランチを確認する

以下のコマンドで現在いるブランチと存在するブランチを確認します。

```
$ git branch
```

### ブランチを移動する

以下のコマンドで現在いるブランチからブランチを移動します。

```
$ git checkout 移動先ブランチ名
```

# 俺でもわかる Git/GitHub

## ブランチをマージする

ブランチを統合することをマージ(merge)といいます。

マージ先のブランチに移動を行ってから以下のコマンドでブランチをマージできます。

```
$ git merge --no-ff マージするブランチ名
```

## 要らないブランチを削除する

以下のコマンドでブランチを削除します。

```
$ git branch -d 削除したいブランチ名
```

## 6. コミットを戻してみよう

git log コマンドで戻したいコミットの id を取得してから以下のコマンドでコミットを戻します。

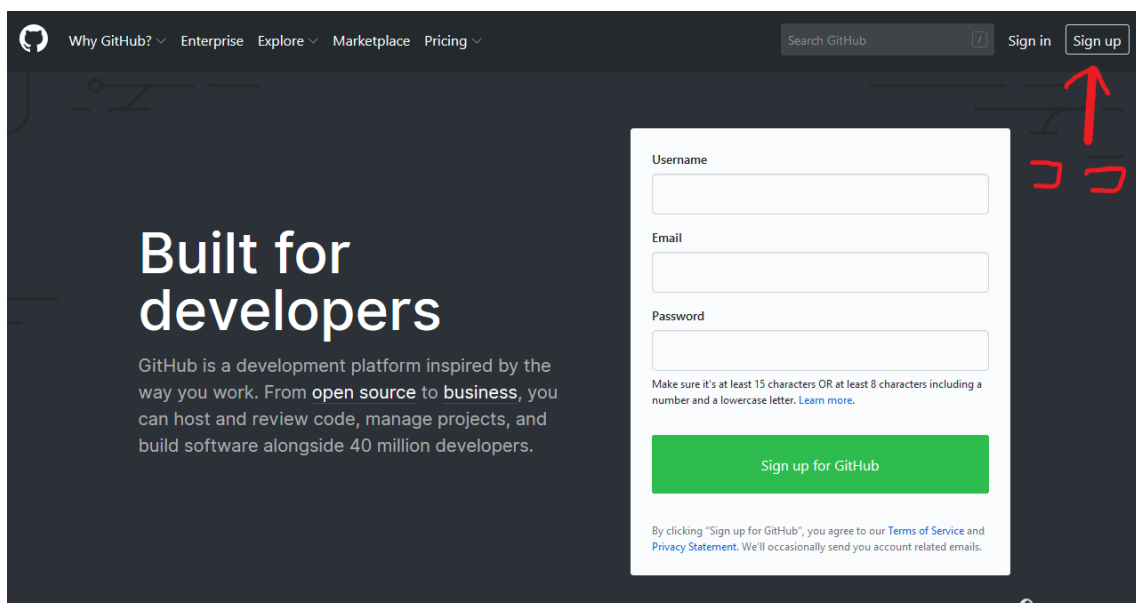
```
$ git reset --hard コミット id
```

## GitHub を使おう

<https://github.com/> にアクセスしてください。

### 1. 登録してみよう

サイトの案内に従って登録してください。



# 俺でもわかる Git/GitHub

## 2. ログインしてみよう

<https://github.com/login> からログインできます

## 3. リポジトリを作成しよう

<https://github.com/new> からリポジトリの作成を行います。


必要事項を埋めて Create repository ボタンを押しましょう。

### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

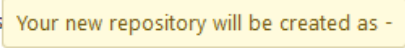
[Import a repository.](#)

Owner

 no1fushi ▾

Repository name \*

/ リポジトリ名 ✓

Great repository names  on? How about **congenial-waffle**?

Description (optional)

リポジトリの説明

☒



Public

Anyone can see this repository. You choose who can commit.

☐



Private

You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☐ Initialize this repository with a README

This will let you immediately clone the repository to your computer.

Add .gitignore: None ▾

Add a license: None ▾



Create repository

# 俺でもわかる Git/GitHub

## 4. リモートリポジトリにプッシュしよう

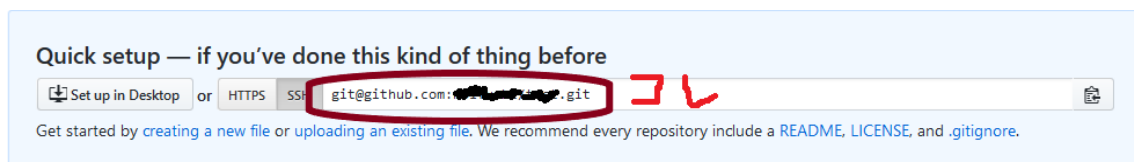
ローカルリポジトリをリモートリポジトリに反映することをプッシュ(Push)と呼びます。

### ローカルリポジトリとリモートリポジトリの関連付けをしよう

この作業は最初の一度だけ行えばいい作業です。以下のコマンドを実行してください。

```
$ git remote add origin リモートリポジトリパス
```

リモートリポジトリパスはリモートリポジトリを作成した際に表示されます。



### リモートリポジトリにプッシュしよう

以下のコマンドを実行してください。

リモートリポジトリのブランチは標準では origin ローカルリポジトリのブランチは master です。

```
$ git push -u リモートリポジトリブランチ ローカルリポジトリのブランチ
```

## 5. リモートリポジトリの状態をローカルリポジトリに反映しよう

以下のコマンドを実行してください

```
$ git pull。 リモートリポジトリブランチ ローカルリポジトリのブランチ
```



# 俺でもわかる Git/GitHub

## 6. プルリクエストを送ってみよう

Push するときにローカルリポジトリのブランチを master 以外のブランチで origin に push するとリモートリポジトリでもそのブランチが作成され、更新内容は origin ブランチではなくその作成されたブランチに反映されます。

なので、そのブランチの更新内容を origin ブランチにマージしてくださいとお伺いを立てる作業をプルリクエスト (pull request) と呼びます。

練習・説明

Edit

Manage topics

12 commits

3 branches

0 packages

0 releases

1 contributor

Your recently pushed branches:

(less than a minute ago)



Compare & pull request

Branch: master

New pull request

Create new file

Upload files

Find file

Clone or download

Compare pull request #4 from

Latest commit 2cd732f on 16 Jul

README.md

Update README.md

5 years ago

test.txt

testtttt

5 months ago

README.md

## pullreq

Pull Requestの練習

## 7. 既にあるリモートリポジトリをローカルに持ってこよう

持ってくるリポジトリ用のディレクトリを作成し以下のコマンドを実行してください。

```
$ git clone リモートリポジトリパス
```