

GitHubの使い方

Alid inc. - Hironao Sekine

Goal

1. gitの基礎的な使い方だけでも扱えるようになる
2. GitHubの位置付けの理解
3. git + GitHubちょっと触ってみる

目次

1. Gitとは

2. GitHubとは

目次

1. Gitとは

1. Gitとは

2. 差分

3. C/S型・分散型

2. GitHubとは

1. GitHubとは

2. gitコマンドGitHubアカウントの準備

3. GitHub開発フロー

4. 実践 git & GitHub

目次

1. Gitとは

2. GitHubとは

Gitとは

「バージョン管理システム」の一つ

Gitとは

「バージョン管理システム」の一つ

Gitとは

「バージョン管理システム」の一つ

その他のバージョン管理システムと歴史…

Gitとは

「バージョン管理システム」の一つ

その他のバージョン管理システムと歴史…

ソフト名	リリース年	方式
RCS	1982	ローカル上のみ
CVS	1990	C/S型
BitKeeper	1998	分散型
Subversion	2000	C/S型
Bazaar	2005	分散型
Git	2005	分散型
Mercurial	2005	分散型

etc…

Gitとは

「バージョン管理システム」の一つ

その他のバージョン管理システムと歴史…

ソフト名	リリース年	方式
RCS	1982	ローカル上のみ
CVS	1990	C/S型
BitKeeper	1998	分散型
Subversion	2000	C/S型
Bazaar	2005	分散型
Git	2005	分散型
Mercurial	2005	分散型

etc…

Gitとは

「バージョン管理システム」の一つ

その他のバージョン管理システムと歴史…

ソフト名	リリース年	方式
RCS	1982	ローカル上のみ
CVS	1990	C/S型
BitKeeper	1998	分散型
Subversion	2000	C/S型
Bazaar	2005	分散型
Git	2005	分散型
Mercurial	2005	分散型

etc…

Gitとは

「バージョン管理システム」の一つ

色々あるけれど、結局目的はただ一つ...

Gitとは

「バージョン管理システム」の一つ

色々あるけれど、結局目的はただ一つ…

ファイルの「差分管理」を便利に行いたい

Gitとは

「バージョン管理システム」の一つ

色々あるけれど、結局目的はただ一つ…

ファイルの「差分管理」を便利に行いたい

差分管理…？

差分

旧ファイルA

- ・ あいうえお
- ・ かきくけこ
- ・ さしすせそ 「さしすせそ」の行が消えた
- ・ たちつてと
- ・ なにぬねの

「はひふへほ」の行が追加された

新ファイルA

- ・ あいうえお
- ・ かきくけこ
- ・ たちつてと
- ・ なにぬねの
- ・ はひふへほ



差分

旧ファイルA

- ・ あいうえお
- ・ かきくけこ
- ・ さしすせそ
- ・ たちつてと
- ・ なにぬねの

新ファイルA

- ・ あいうえお
- ・ かきくけこ
- ・ たちつてと
- ・ なにぬねの
- ・ はひふへほ

- ・ あいうえお
- ・ かきくけこ
-  ・ さしすせそ
- ・ たちつてと
- ・ なにぬねの
-  ・ はひふへほ

旧ファイル→新ファイルにするための差分情報

差分

「diff」 コマンド

```
diff -ur 旧ファイルA 新ファイルA > 差分情報.patch
```

差分

「diff」 コマンド

```
diff -ur 旧ファイルA 新ファイルA > 差分情報.patch
```

```
--- ./old.txt 2017-05-07 16:01:29.000000000 +0900
+++ ./new.txt 2017-05-07 04:55:11.000000000 +0900
@@ -1,5 +1,5 @@
 あいうえお
 かきくけこ
-さしすせそ
 たちつてと
 なにぬねの
+はひふへほ
```

差分

「diff」 コマンド

```
diff -ur 旧ファイルA 新ファイルA > 差分情報.patch
```

```
--- ./old.txt    2017-05-07 16:01:29.000000000 +0900
+++ ./new.txt    2017-05-07 04:55:11.000000000 +0900
@@ -1,5 +1,5 @@
 あいうえお
 かきくけこ
-さしすせそ
 たちつてと
 なにぬねの
+はひふへほ
```

「patch」 コマンド

```
patch -u < 差分情報.patch
```

Gitとは

「バージョン管理システム」の一つ

その他のバージョン管理システムと歴史…

ソフト名	リリース年	方式
RCS	1982	ローカル上のみ
CVS	1990	C/S型
BitKeeper	1998	分散型
Subversion	2000	C/S型
Bazaar	2005	分散型
Git	2005	分散型
Mercurial	2005	分散型

etc…

Gitとは

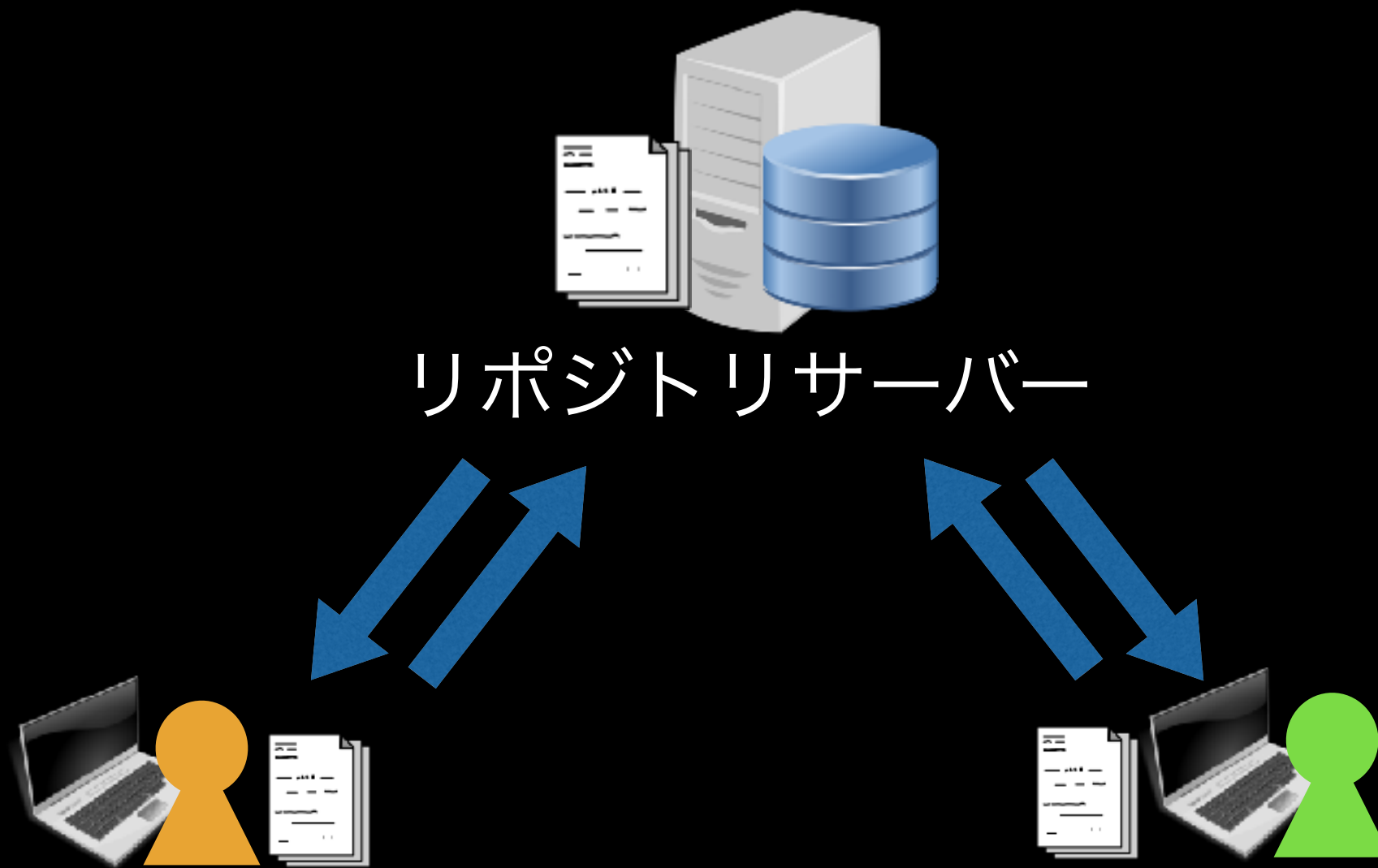
「バージョン管理システム」の一つ

その他のバージョン管理システムと歴史…

ソフト名	リリース年	方式
RCS	1982	ローカル上のみ
CVS	1990	C/S型
BitKeeper	1998	分散型
Subversion	2000	C/S型
Bazaar	2005	分散型
Git	2005	分散型
Mercurial	2005	分散型

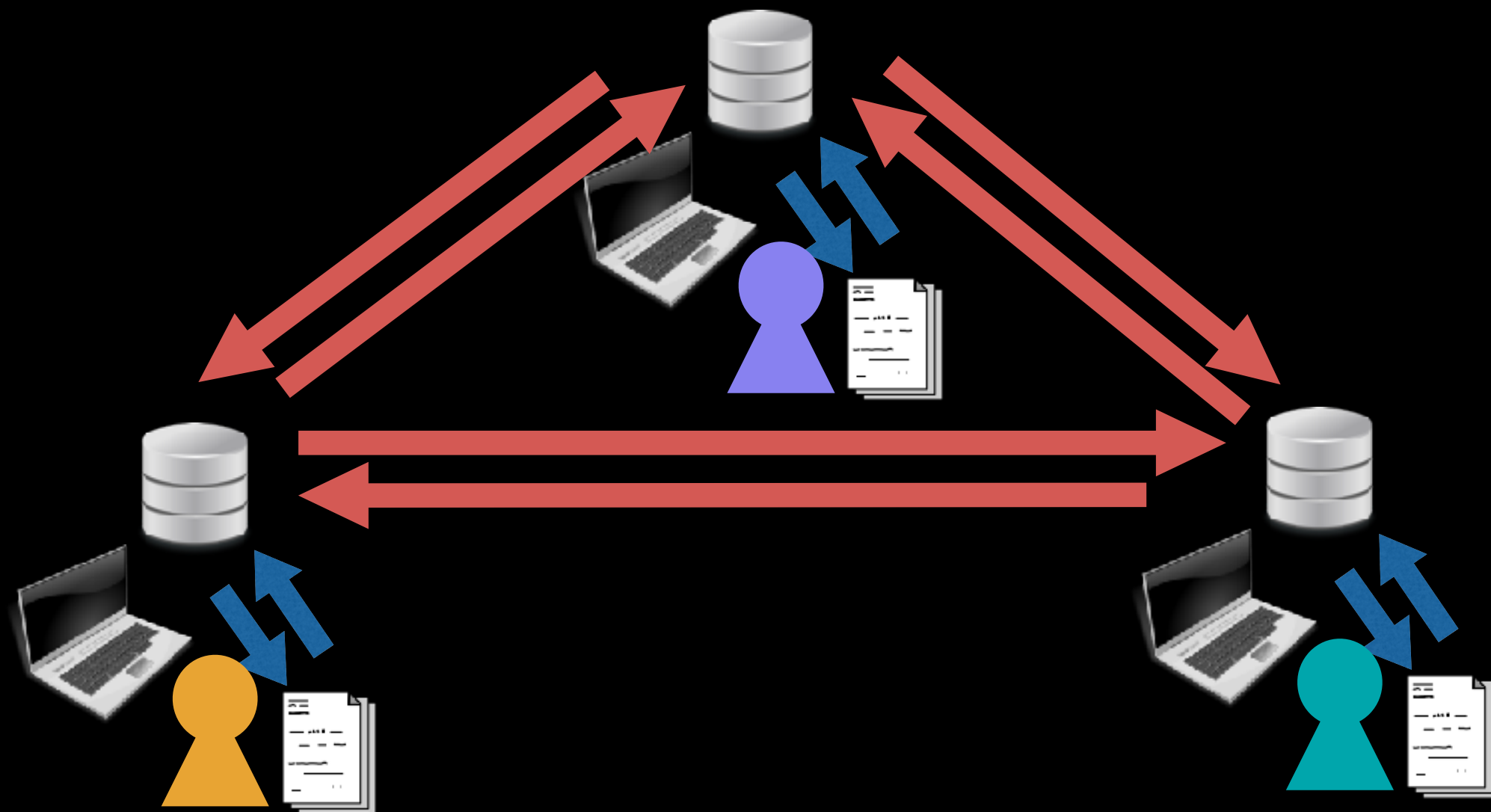
etc…

C/S型



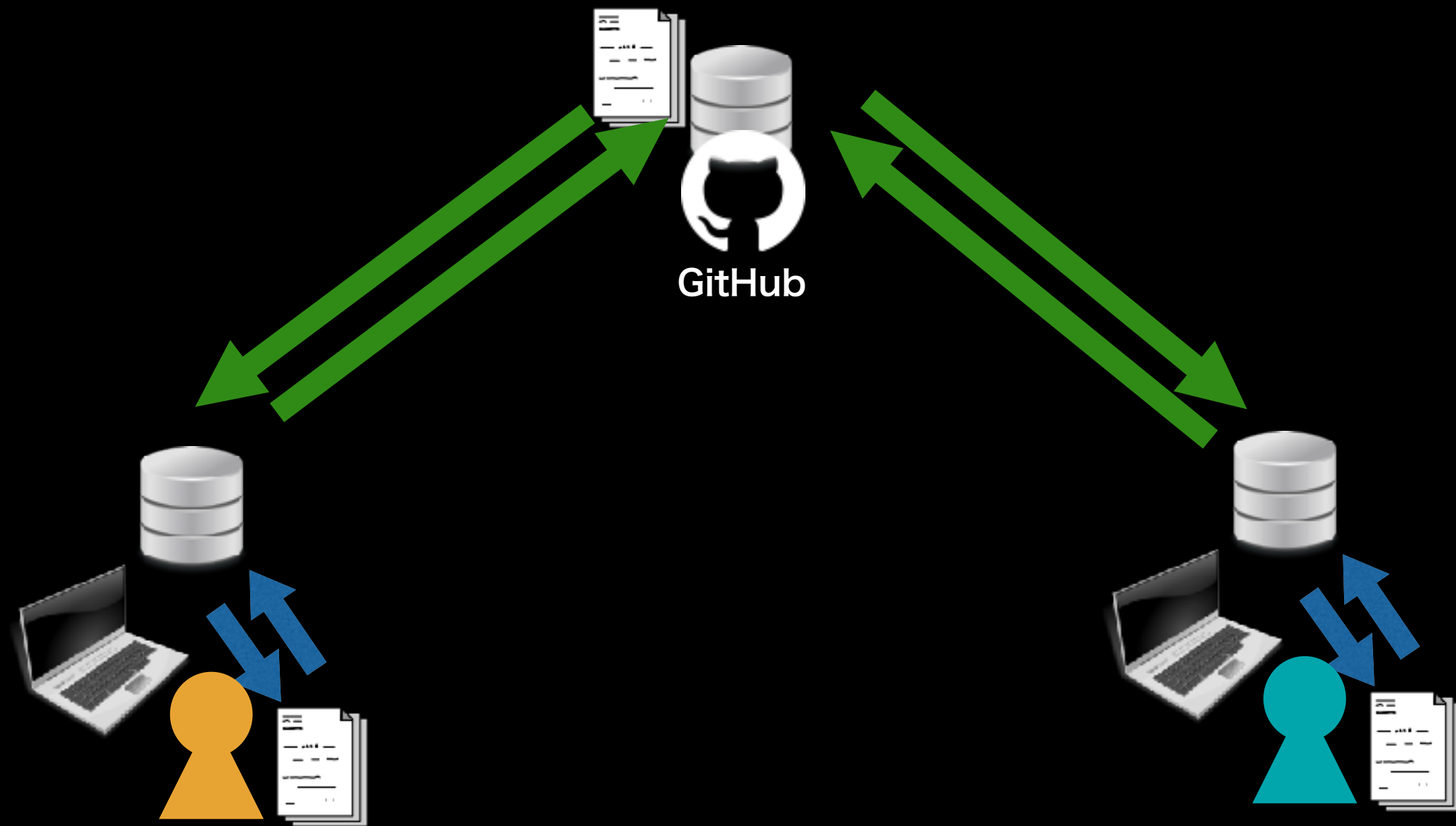
ファイルは各自ローカルにコピー
リポジトリ（履歴情報）はサーバー上に

分散型



リポジトリもファイルも各自ローカルに

分散型 - GitHubの立ち位置



リポジトリもファイルも各自ローカルに

一旦まとめ

1. Gitとはバージョン管理ツールの一つである
2. バージョンを管理するということは、履歴を管理するということ
3. 履歴を管理しているものは「リポジトリ」と呼ばれる

目次

1. ~~Git~~とは

2. GitHubとは

GitHubとは

- ・ gitのリポジトリを公開・閲覧できるサービス
- ・ 公開されているリポジトリを誰もがcloneできる
- ・ issuesとforkとpull request機能により
円滑で高速な共同開発が可能になる

GitHubとは

- ・ gitのリポジトリを公開・閲覧できるサービス
- ・ 公開されているリポジトリを誰もがcloneできる
- ・ issuesとforkとpull request機能により
円滑で高速な共同開発が可能になる

GitHubとは

- ・ GitHub用語
 - ・ issues
 - ・ fork
 - ・ pull request
- ・ git用語
 - ・ clone
 - ・ pull

GitHubとは

- ・ GitHub用語
 - ・ issues
 - ・ fork
 - ・ pull request => pull + を求める
- ・ git用語
 - ・ clone
 - ・ pull

gitを知らないとGitHubもよくわからない…

$$\left(\begin{array}{c} \text{ } \\ \text{ } \end{array} ; \begin{array}{c} \text{ } \\ \text{ } \end{array} \right) \dots$$

というわけで...

- ・ git コマンドを使えるように準備
- ・ GitHub にアカウントを作って準備

gitを使う準備

1. gitコマンドの存在確認

1. `$ git --version`

`git version x.xx.x` みたいな表示が出たらOK

2. ない場合はインストール

`$ brew install git`

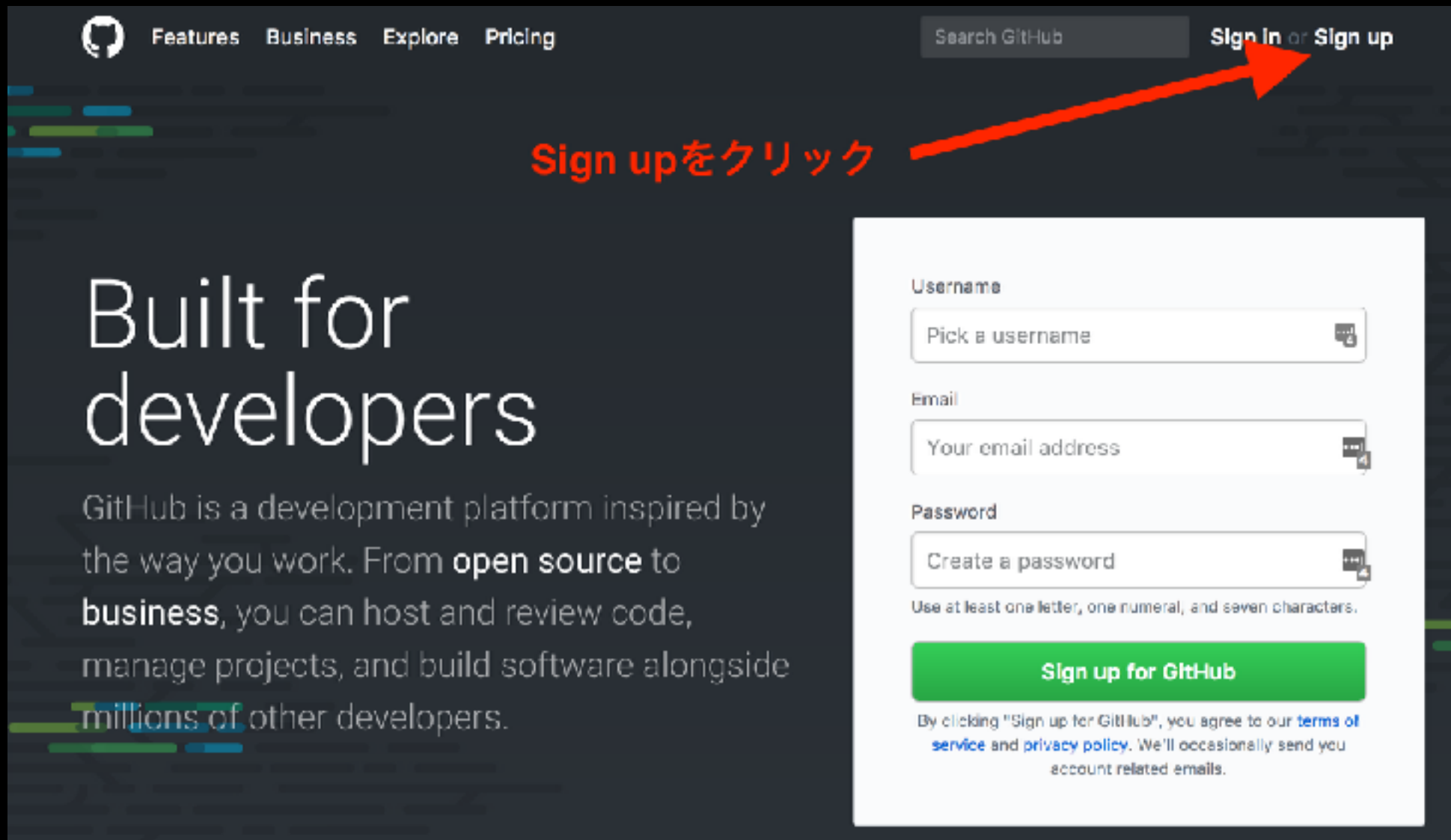
これが一番楽(※ 要homebrew)

GitHubを使う準備

- ・ GitHubにアカウントを作る
- ・ SSH公開鍵をGitHubに登録する

GitHubを使う準備

- <https://github.com/> にアクセス



The image shows the GitHub homepage with a dark header. The header contains the GitHub logo, navigation links (Features, Business, Explore, Pricing), a search bar labeled 'Search GitHub', and a 'Sign In or Sign up' link. A red arrow points from the text 'Sign upをクリック' to the 'Sign up' part of the 'Sign In or Sign up' link. Below the header, the main content area has the text 'Built for developers' and a paragraph about GitHub. A white sign-up form is overlaid on the right side of the page. The form contains fields for 'Username' (with placeholder 'Pick a username'), 'Email' (with placeholder 'Your email address'), and 'Password' (with placeholder 'Create a password'). Below the password field is a note: 'Use at least one letter, one numeral, and seven characters.' At the bottom of the form is a green button labeled 'Sign up for GitHub'. Below the button is a disclaimer: 'By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy policy](#). We'll occasionally send you account related emails.'

Sign upをクリック

Sign In or Sign up

Built for developers

GitHub is a development platform inspired by the way you work. From **open source** to **business**, you can host and review code, manage projects, and build software alongside millions of other developers.

Username
Pick a username

Email
Your email address

Password
Create a password
Use at least one letter, one numeral, and seven characters.

Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy policy](#). We'll occasionally send you account related emails.

GitHubを使う準備

- ・ アカウントを作ります

Join GitHub
The best way to design, build, and ship software.

Step 1:
Create personal account

Create your personal account

Username

This will be your username — you can enter your organization's username next.

Email Address

You will occasionally receive account related emails. We promise not to share your email with anyone.

Password

Use at least one lowercase letter, one numeral, and seven characters.

Annotations:

- 他と被らないUsernameを決めて入力する
このUsernameがGitHubページの[Username]部分になります
`https://github.com/[Username]/[repository]`
- メールアドレスを入力
- GitHubにログインする為のパスワードを決め、入力

You'll love GitHub

- Unlimited collaborators
- Unlimited public repositories
- ✓ Great communication
- ✓ Frictionless development
- ✓ Open source community

GitHubを使う準備

- ・ プランを選びます

Welcome to GitHub

You've taken your first step into a larger world, @hirotest01.

Completed Set up a personal account

Step 2: Choose your plan

Step 3: Tailor your experience

無料プランを選ぶ

Choose your personal plan

☒ Unlimited public repositories for free.

☐ Unlimited private repositories for \$7/month. [\(view in JPY\)](#)

Don't worry, you can cancel or upgrade at any time.

☐ Help me set up an organization next

Organizations are separate from personal accounts and are best suited for businesses who need to manage permissions for many employees. [Learn more about organizations.](#)

Continue

Both plans include:

- ✓ Collaborative code review
- ✓ Issue tracking
- ✓ Open source community
- ✓ Unlimited public repositories
- ✓ Join any organization

GitHubを使う準備

- アンケートには答えず、スキップ...

Welcome to GitHub

You'll find endless opportunities to learn, code, and create, @hirotest01.

✓ Completed
Set up a personal account

📋 Step 2:
Choose your plan

⚙️ Step 3:
Tailor your experience

How would you describe your level of programming experience?

☐ Totally new to programming ☐ Somewhat experienced ☐ Very experienced

What do you plan to use GitHub for? (check all that apply)

☐ Development ☐ Project Management ☐ Design
☐ Research ☐ School projects ☐ Other (please specify)

Which is closest to how you would describe yourself?

☐ I'm a professional ☐ I'm a hobbyist ☐ I'm a student
☐ Other (please specify)

What are you interested in?

Submit

skip this step

アンケートみたいなものなので入力せず
とりあえず「skip this step」をクリックします

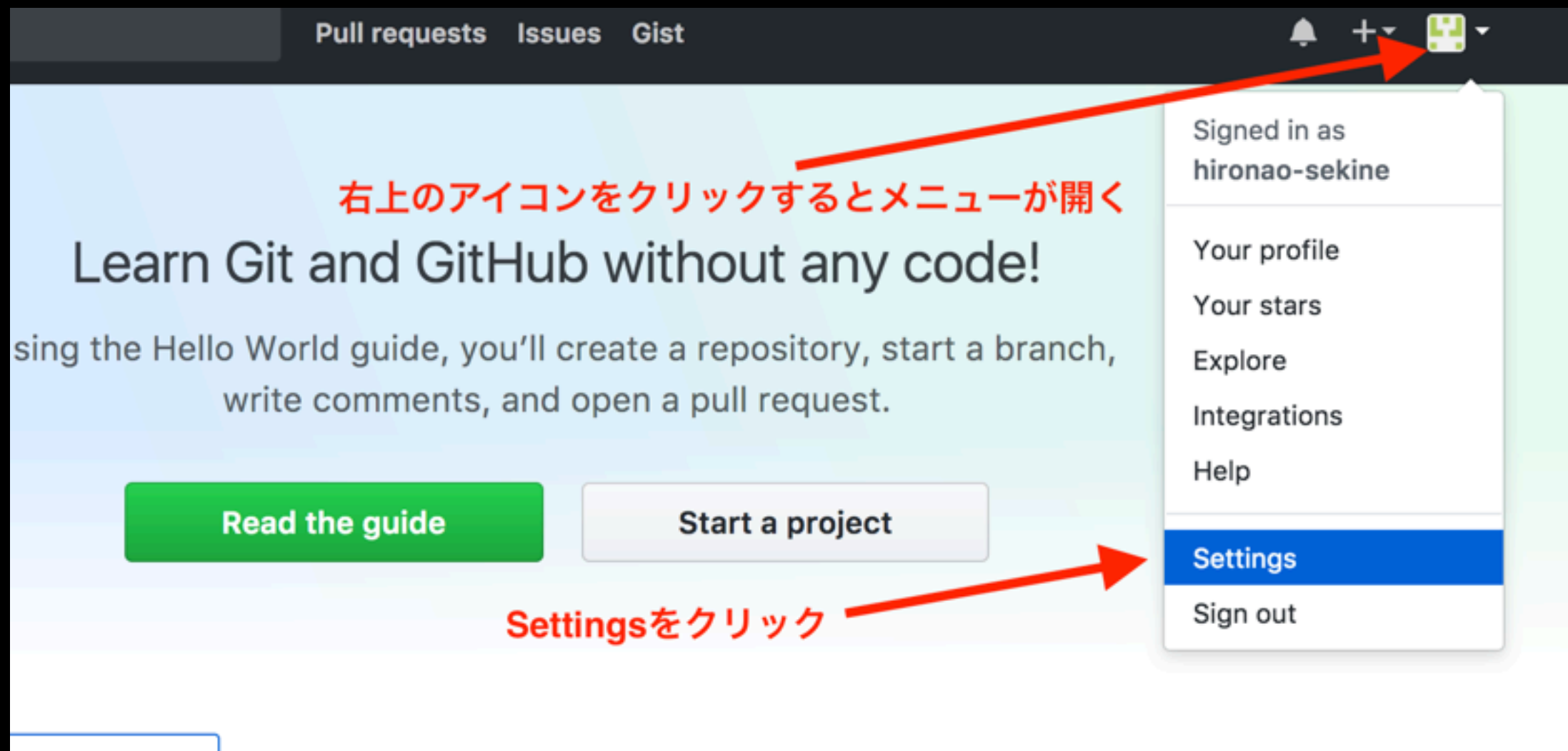
GitHubを使う準備

- 登録に使用したメールアドレスへ
登録確認用のメールが来るのでリンクを開いて登録完了



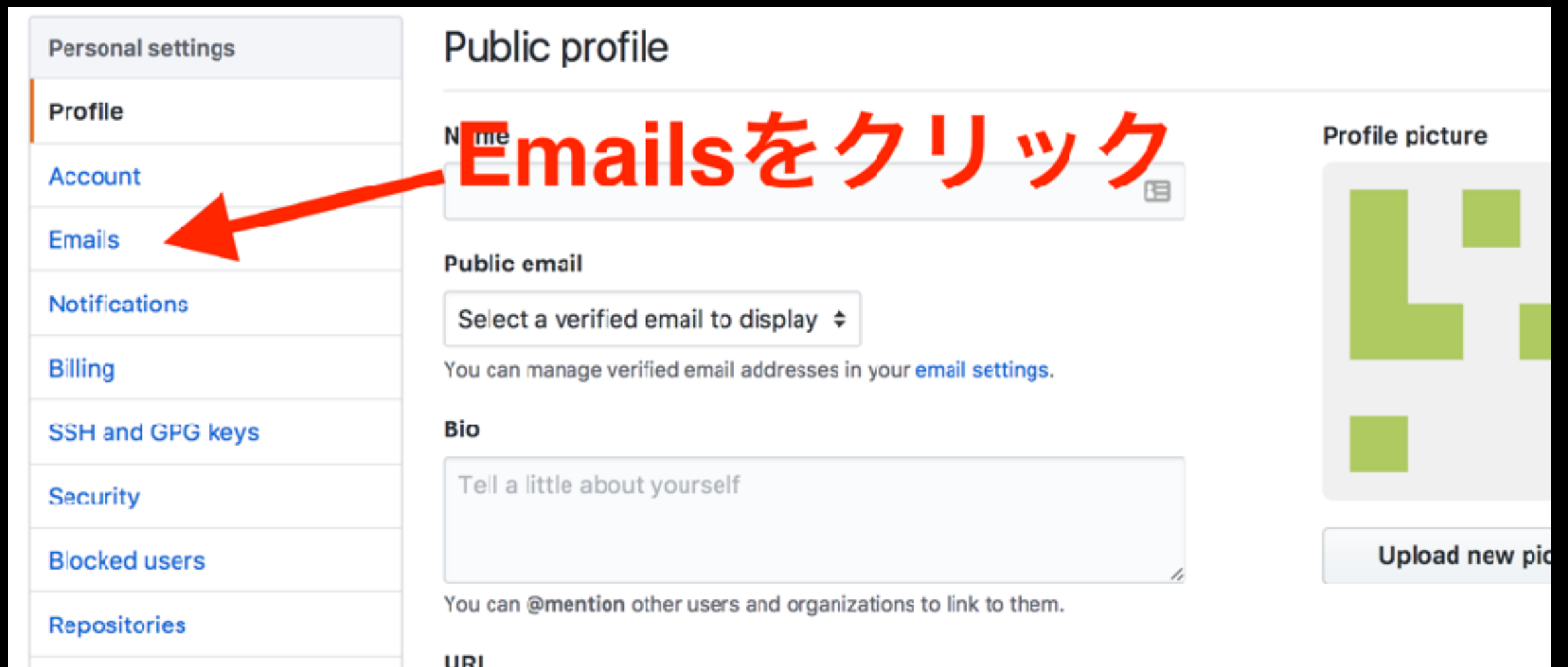
GitHubを使う準備

- ・ 設定ページに移動します



GitHubを使う準備

- Emails 設定ページに移動します



GitHubを使う準備

- ・メールアドレスを非公開に設定します

Allow all verified emails ▼ Save

Please add a verified email, in addition to your primary email, in order to choose a backup email address.

☐ **Keep my email address private**

We'll remove your public profile email and use `hironao-sekine@users.noreply.github.com` when performing web-based Git operations and sending email on your behalf. If you want command line Git operations to use your private email you must [set your email in Git](#).

Email preferences

チェックを入れる

このメールアドレスをあとで使いますので
何処かにコピーしておきます

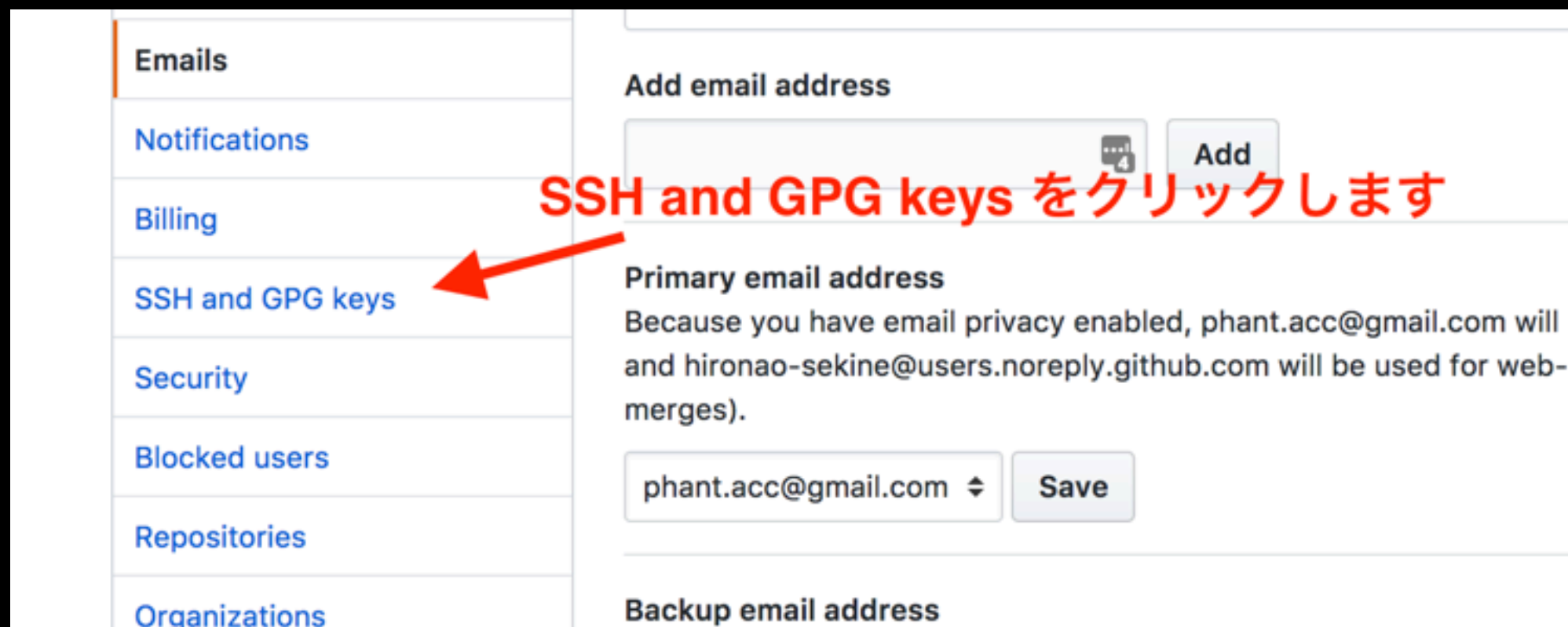
GitHubを使う準備

- ・ gitがcommitした時にEmail情報として記録するものを設定

```
$ git config --global user.email "コピーしたEmailアドレス"
```

GitHubを使う準備

- SSH公開鍵を登録する設定ページに移動します



The screenshot shows the GitHub account settings interface. On the left is a sidebar with links: Emails, Notifications, Billing, SSH and GPG keys, Security, Blocked users, Repositories, and Organizations. The 'SSH and GPG keys' link is highlighted with a red arrow. A red text overlay with the Japanese instruction 'SSH and GPG keys をクリックします' (Click on SSH and GPG keys) points to this link. The main content area is titled 'Add email address' and contains a form for setting a primary email address. It includes a text input field with 'phant.acc@gmail.com', a dropdown arrow, and a 'Save' button. Below this is a section for 'Backup email address'.

SSH and GPG keys をクリックします

Add email address

Primary email address

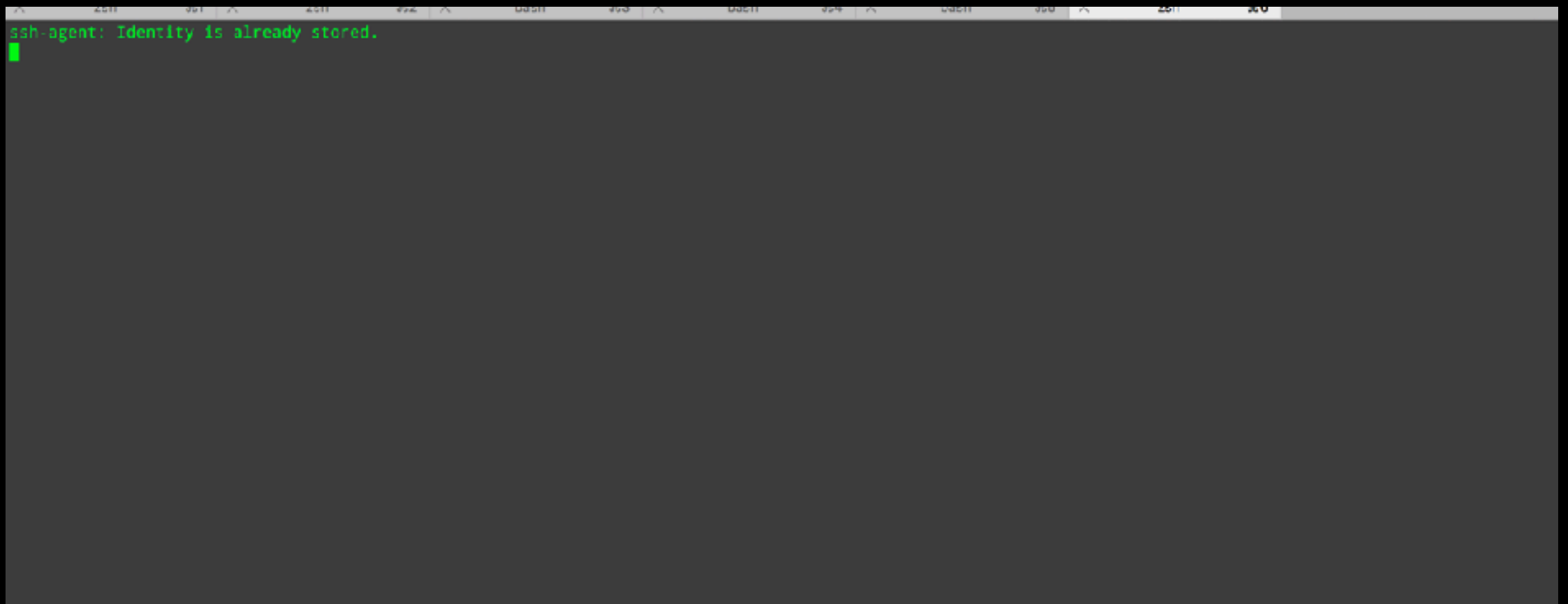
Because you have email privacy enabled, phant.acc@gmail.com will be used for web-based merges).

phant.acc@gmail.com Save

Backup email address

GitHubを使う準備

- ・ ウィンドウをブラウザから、ターミナルに切り替えます

A screenshot of a macOS-style terminal window. The title bar at the top shows several window tabs with icons and numbers. The terminal area is dark gray. The first line of text is green and reads "ssh-agent: Identity is already stored." followed by a green cursor block.

```
ssh-agent: Identity is already stored.
```

GitHubを使う準備

- ・ ホームディレクトリ直下 `.ssh` ディレクトリに移動
無い場合「`mkdir ~/.ssh`」を行い、作ってから下記を行いましょう

```
$ cd ~/.ssh
```

```
$ ls -la
```

一旦ここで`.ssh`フォルダ内に

`id_rsa`や`id_rsa.pub`がない事を確認してください

(あると次のコマンドで上書きしてしまうので…)

GitHubを使う準備

- ・ 秘密鍵・公開鍵のペアを生成するコマンドの実行

```
$ ssh-keygen -t rsa
```

```
Generating public/private rsa key pair.
```

```
Enter file in which to save the key (/Users/hiro/.ssh/id_rsa): [Return]キーを押します
```

GitHubを使う準備

- ・ 鍵のパスフレーズを求められますが
パスフレーズは空でも今回は問題ありません

```
$ ssh-keygen -t rsa
```

```
Generating public/private rsa key pair.
```

```
Enter file in which to save the key (/Users/hiro/.ssh/id_rsa):
```

```
Enter passphrase (empty for no passphrase): [Return]キーを押します
```


GitHubを使う準備

- ・ パスフレーズの確認の為、再入力を促されますが
これも何も入力せず、リターンキーだけを叩きます

```
$ ssh-keygen -t rsa
```

```
Generating public/private rsa key pair.
```

```
Enter file in which to save the key (/Users/hiro/.ssh/id_rsa):
```

```
Enter passphrase (empty for no passphrase):
```

```
Enter same passphrase again: [Return]キーを押します
```

GitHubを使う準備

- ・ ズラズラっと何かが出たらOKです

```
$ ssh-keygen -t rsa
```

```
Generating public/private rsa key pair.
```

```
Enter file in which to save the key (/Users/hiro/.ssh/id_rsa):
```

```
Enter passphrase (empty for no passphrase):
```

```
Enter same passphrase again:
```

```
Your identification has been saved in /Users/hiro/.ssh/id_rsa.
```

```
Your public key has been saved in /Users/hiro/.ssh/id_rsa.pub.
```

```
The key fingerprint is:
```

```
SHA256:pB4l+xSn/badL8pE99Ou/4EJamesP/XjnN8xQj4rjac hiro@hiro-mac-6.local
```

```
The key's randomart image is:
```

```
+---[RSA 2048]-----+
```

```
( ~ 省略 ~ )
```

```
+----[SHA256]-----+
```

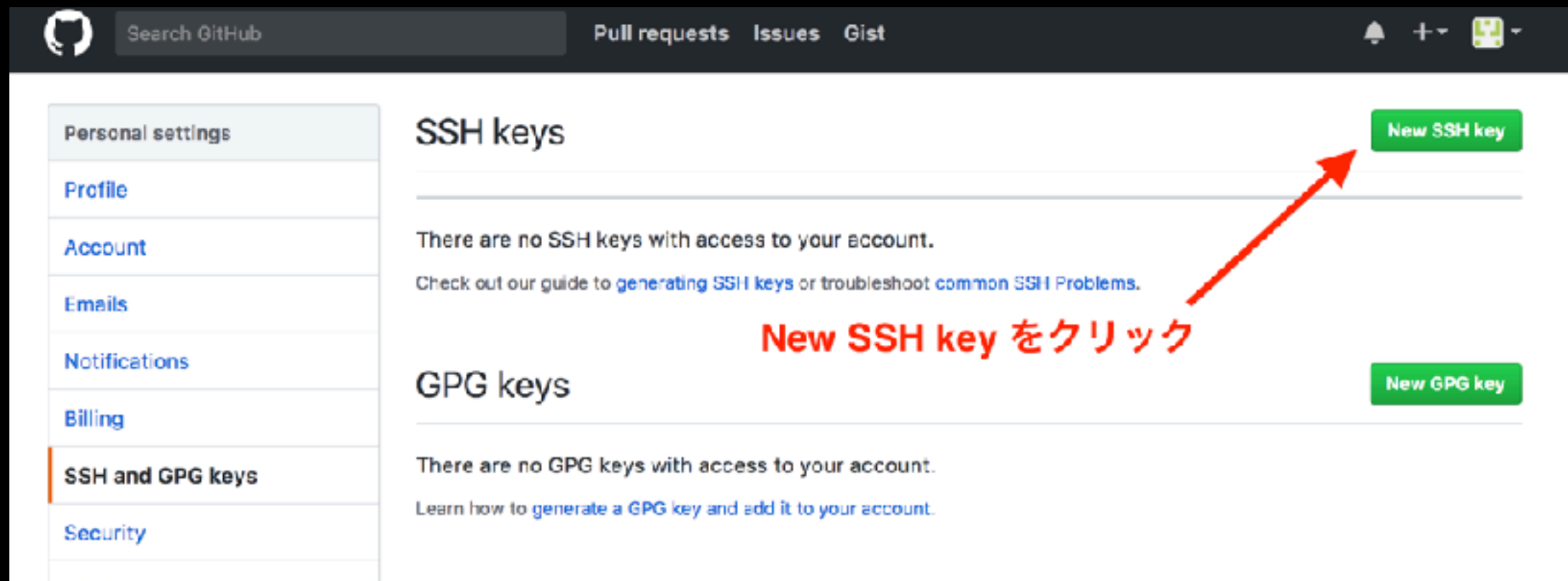
GitHubを使う準備

- ・ ~/.ssh 下に出来た id_rsa.pub の中身を
クリップボードにコピーします

```
$ pbcopy < ~/.ssh/id_rsa.pub
```

GitHubを使う準備

- ・ ブラウザに戻り 「New SSH key」 を押します



GitHubを使う準備

- 名前をつけ、コピーした公開鍵の内容を貼り付けて Add SSH keyをクリックして登録します

The screenshot shows the GitHub 'SSH keys' page. At the top right is a green button labeled 'New SSH key'. Below the header, a message states 'There are no SSH keys with access to your account.' The main form has two sections: 'Title' and 'Key'. The 'Title' section has a text input field containing 'テスト', with a red arrow pointing to it from the annotation '適当に名前を付けます'. The 'Key' section has a large text area containing a long SSH public key, with a red arrow pointing to it from the annotation 'Keyの枠内にペーストします'. At the bottom of the form is a green button labeled 'Add SSH key', with a red arrow pointing to it from the annotation 'Add SSH key をクリック'. At the very bottom of the page, there is a link to a guide on generating SSH keys and troubleshooting common problems.

SSH keys New SSH key

There are no SSH keys with access to your account.

Title

テスト

適当に名前を付けます

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQBAQC33AU1qbiwofc91+QC1XfKZvoWTG+LS2mBCQGTqOSQYmt
+rhOom+wlQI3BvLfdaGFtyb+S5leS6zWK1YeL2jHZrTiel3ZyGvVUMC7gK/z8QIAf/jX0L3wR6nONa/8VhN
Co8sJgN40bcEh7fAYseemacLCRpj1yu3hRnYeiKB00PGL9+Xahg/Y3ajp21KSeZY5K0Hmsal0YOS/7K|BK
xeyzfkvAHgeuBwH6L1UzQv50l/6tuce9JoasHPF7y9+Hq5DzT4GAiQ6lnh4RZzYnhA3L5B5U44KLBScp
BUb8Q/tDy5hcmaiSUHZsWMo9JiaaAWPQgGJyXb95y1rX7zlzzb hiro@hiro-mac-6.local
```

Keyの枠内にペーストします

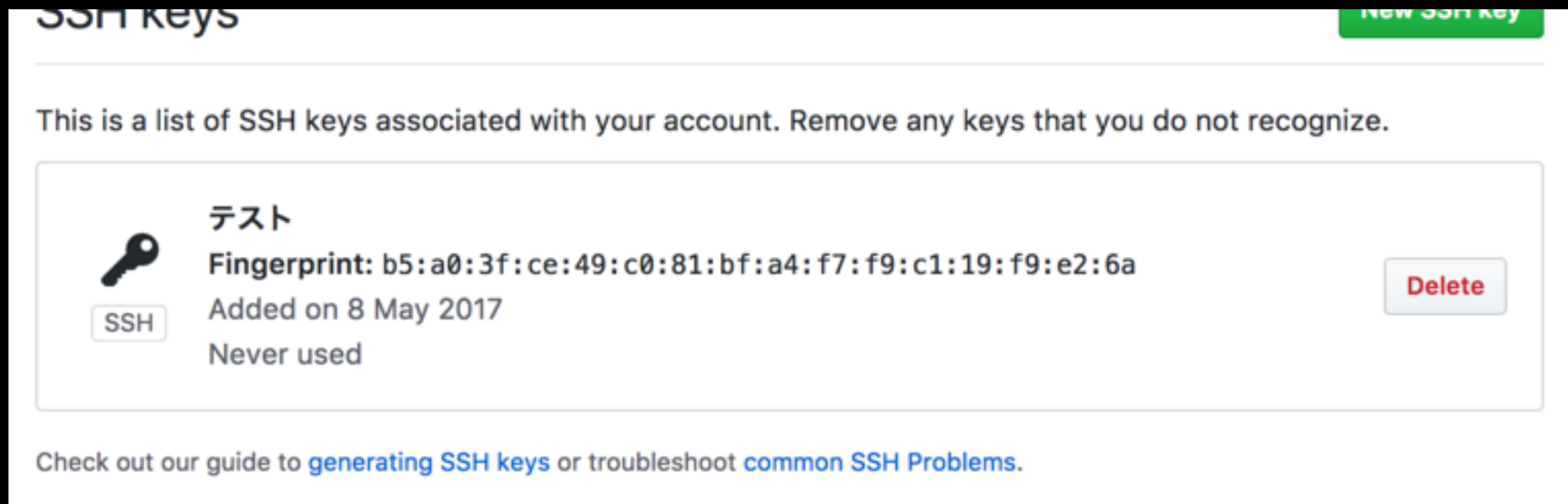
Add SSH key

Add SSH key をクリック

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH Problems](#).

GitHubを使う準備

- 登録が完了するとこんな表示になります



GitHubを使う準備

- ・ SSH接続のテストを行います
初回は確認が入りますので yes と答え、接続を許可します

```
$ ssh -T git@github.com
```

```
The authenticity of host 'github.com (192.30.253.112)' can't be established.
```

```
RSA key fingerprint is
```

```
SHA256:nThbg6kXUpJWGI7E1IGOCspRomTxdCARLviKw6E5SY8.
```

```
Are you sure you want to continue connecting (yes/no)? yes [Return]キーを押します
```

GitHubを使う準備

- ・ SSH接続のテストを行います

You've successfully authenticatedと出れば成功です

```
$ ssh -T git@github.com
```

```
The authenticity of host 'github.com (192.30.253.112)' can't be established.
```

```
RSA key fingerprint is
```

```
SHA256:nThbg6kXUpJWGI7E1IGOCspRomTxdCARLviKw6E5SY8.
```

```
Are you sure you want to continue connecting (yes/no)? yes
```

```
Warning: Permanently added 'github.com,192.30.253.112' (RSA) to the list of  
known hosts.
```

```
Hi hironao-sekine! You've successfully authenticated, but GitHub does not provide  
shell access.
```


ここまでやったことのまとめ

- ・ gitコマンドの確認・準備
- ・ GitHubにアカウントを作成
 - ・ GitHub上のメールアドレス非公開設定
 - ・ SSH接続用の秘密鍵・公開鍵のペアを生成
 - ・ GitHubに「SSH公開鍵」を登録

GitHubとは

- ・ GitHubにはシチュエーション毎に使い方が異なる側面がある
 - ・ 公開プログラムのソースを手元で閲覧したい
 - ・ 公開プログラムのソースに個人的な改変を加え、実行・確認したい
 - ・ 随時変更されていく公開プログラムに追従したい
 - ・ 公開プログラムにバグらしきものを見つけたので修正したい
 - ・ 公開プログラムに機能追加をしたい
 - ・ 自作プログラムを公開し複数人で共有しながら開発したい

GitHubとは

- ・ GitHubにはシチュエーション毎に使い方が異なる側面がある

- ・ 公開プログラムのソースを手元で閲覧したい

- ・ 公開プログラムのソースに個人的な改変を加え、
実行・確認したい

GitHubのアカウント = 不要

- ・ 随時変更されていく公開プログラムに追従したい

- ・ 公開プログラムにバグらしきものを見つけたので修正したい

- ・ 公開プログラムに機能追加をしたい

- ・ 自作プログラムを公開し複数人で共有しながら開発したい

GitHubとは

- ・ GitHubにはシチュエーション毎に使い方が異なる側面がある

- ・ 公開プログラムのソースを手元で閲覧したい

- ・ 公開プログラムのソースに個人的な改変を加え、
実行・確認したい

GitHubのアカウント = 不要

- ・ 随時変更されていく公開プログラムに追従したい

- ・ 公開プログラムにバグらしきものを見つけたので修正したい

- ・ 公開プログラムに機能追加をしたい

GitHubのアカウント = 必要
Gitコマンドの利用 = 必要

- ・ 自作プログラムを公開し複数人で共有しながら開発したい

目次

1. ~~Git~~とは

1. ~~Git~~とは

2. 差分

3. C/S型・分散型

2. ~~GitHub~~とは

1. ~~GitHub~~とは

2. ~~gitコマンドGitHubアカウントの準備~~

3. GitHub開発フロー

4. 実践 git & GitHub

目次

1. ~~Gitとは~~

1. ~~Gitとは~~

2. 差分

3. C/S型・分散型

2. ~~GitHubとは~~

1. ~~GitHubとは~~

2. ~~gitコマンドGitHubアカウントの準備~~

3. GitHub開発フロー

4. 実践 git & GitHub

目次

1. ~~Git~~とは

1. ~~Git~~とは

2. 差分

3. C/S型・分散型

2. ~~GitHub~~とは

1. ~~GitHub~~とは

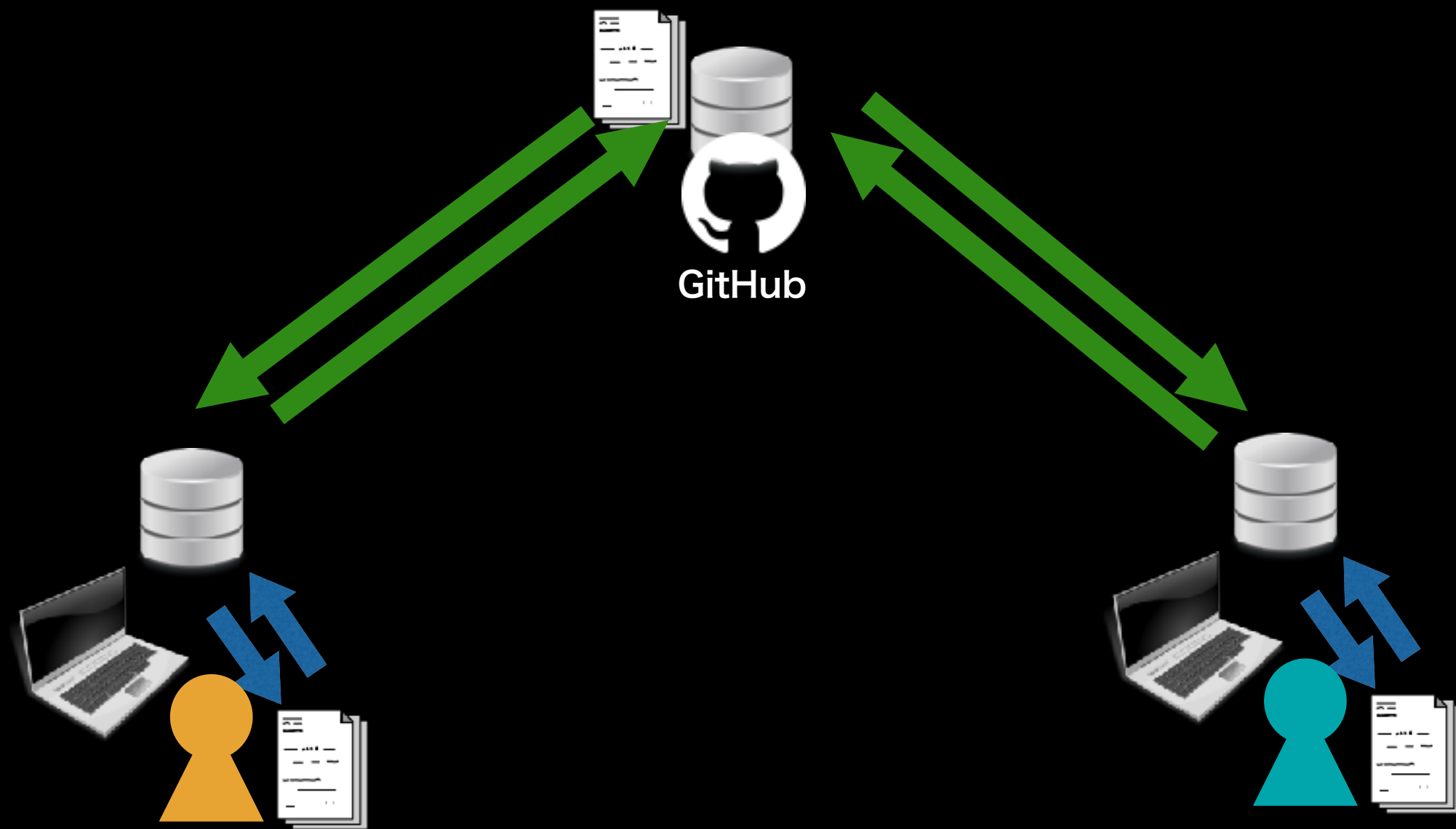
2. ~~gitコマンドGitHubアカウントの準備~~

3. ~~GitHub~~開発フロー

4. 実践 git & GitHub

これぞGitHubが
「Git + Hub」たる所以

GitHub開発フロー



GitHub開発フロー



GitHub開発フロー



github.com/A/リポジトリ1



GitHub



リポジトリ1

Aさん



Aさんのリポジトリ1を
fork !

GitHub開発フロー



GitHub開発フロー



GitHub開発フロー



`git clone /B/リポジトリ1`

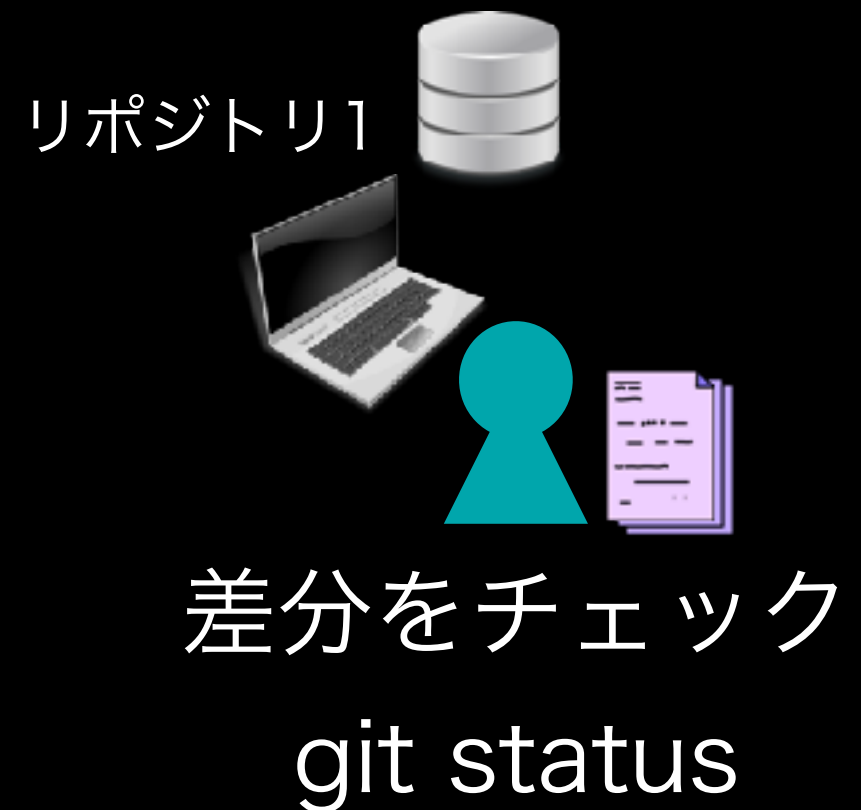
GitHub開発フロー



GitHub開発フロー



GitHub開発フロー



GitHub開発フロー



GitHub開発フロー



GitHub開発フロー



GitHub開発フロー



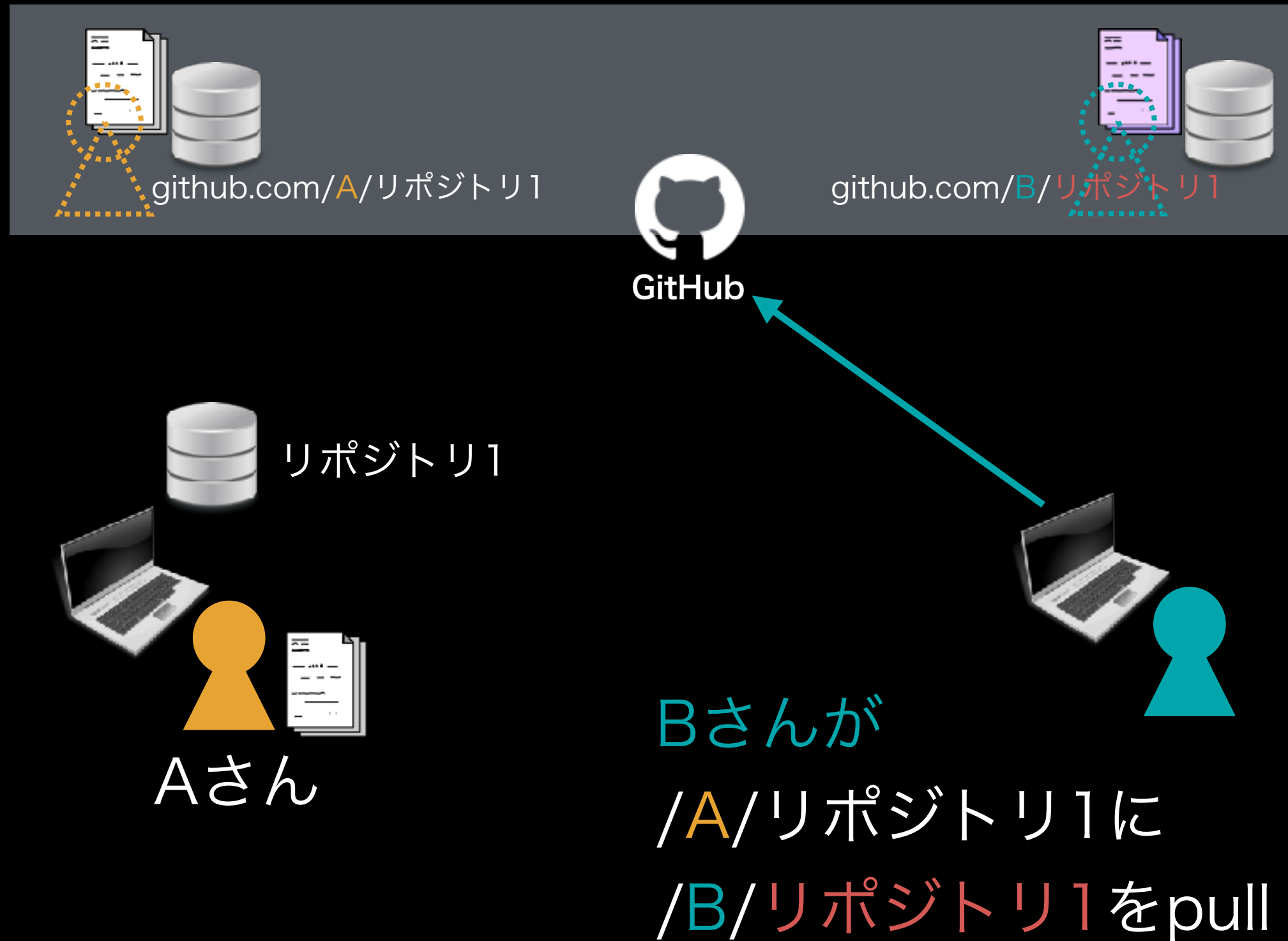
GitHub開発フロー



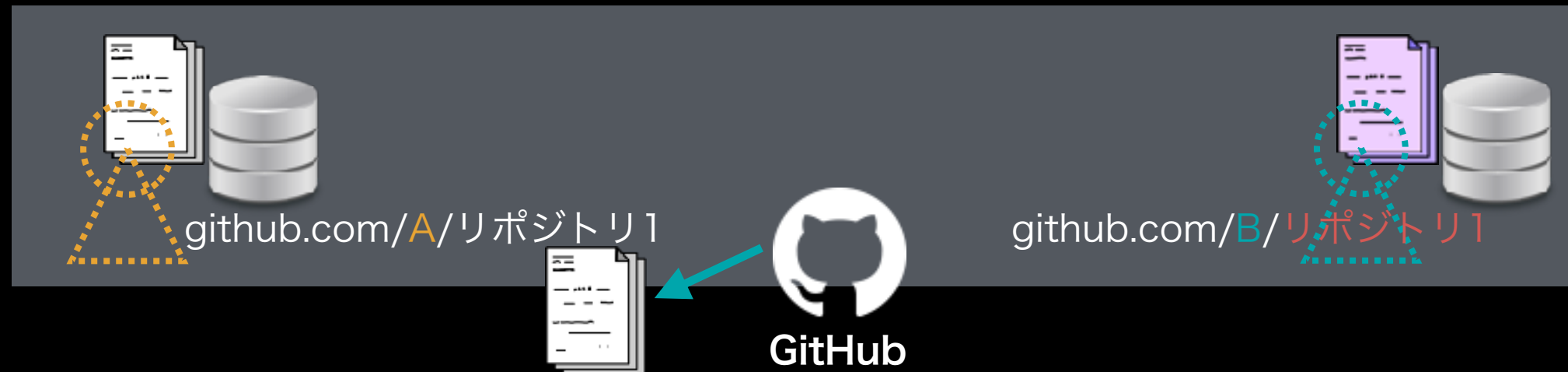
GitHub開発フロー



GitHub開発フロー



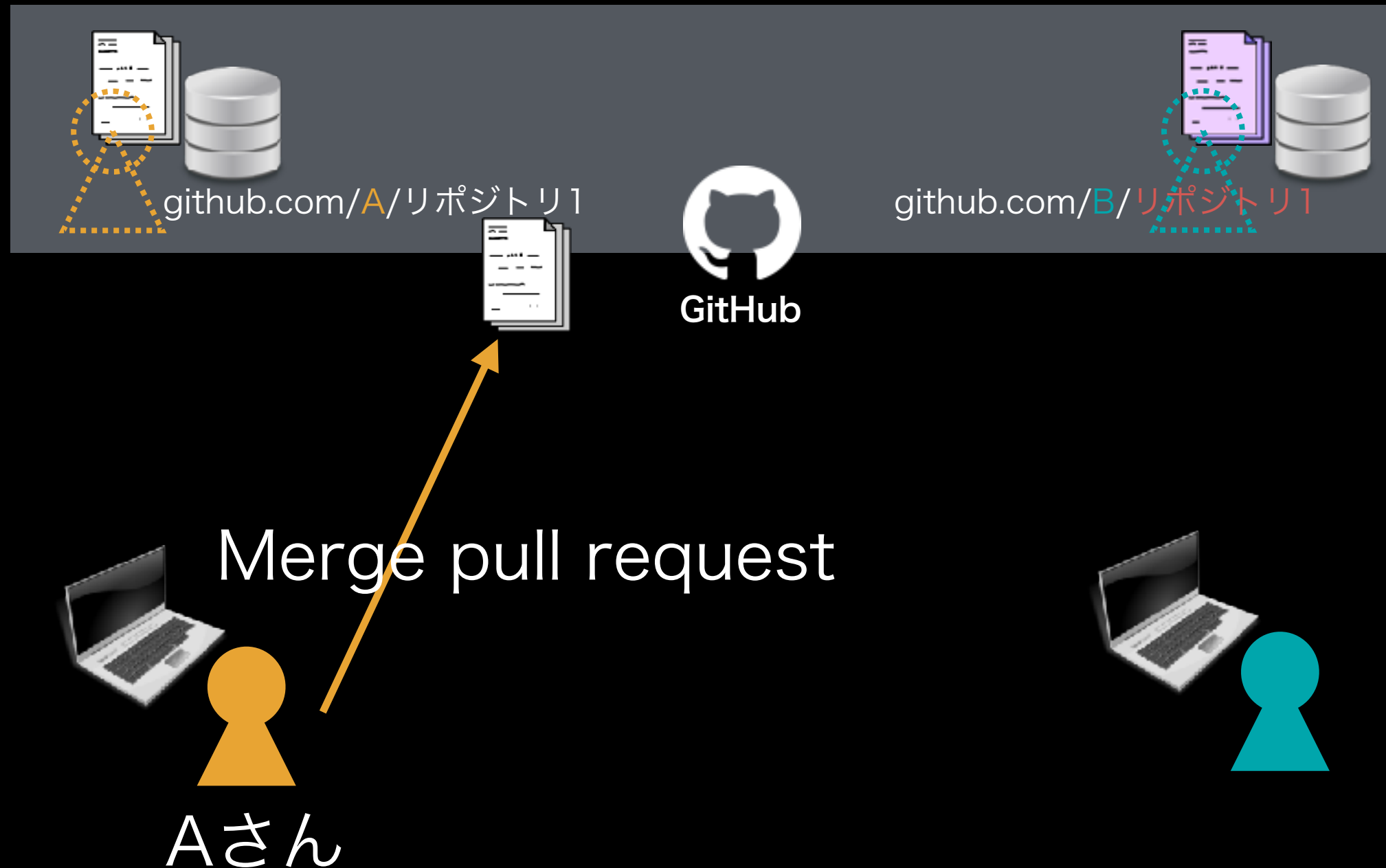
GitHub開発フロー



Aさん



GitHub開発フロー



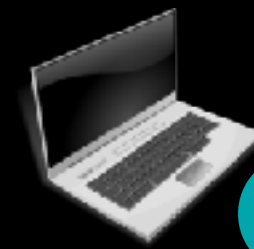
GitHub開発フロー



Merge pull request



Aさん



GitHub開発フロー



Merge pull request



Aさん



GitHub開発フロー



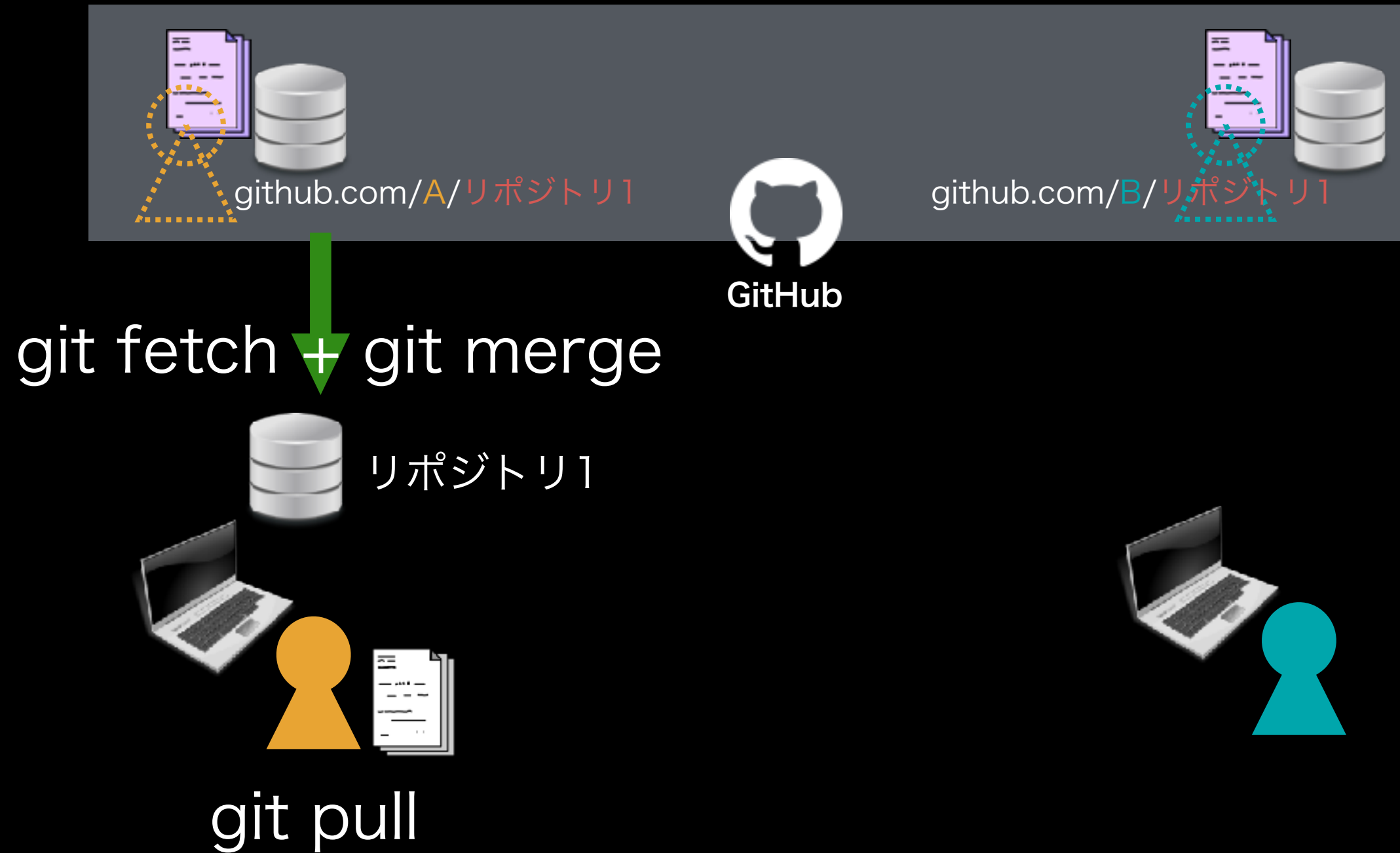
GitHub開発フロー



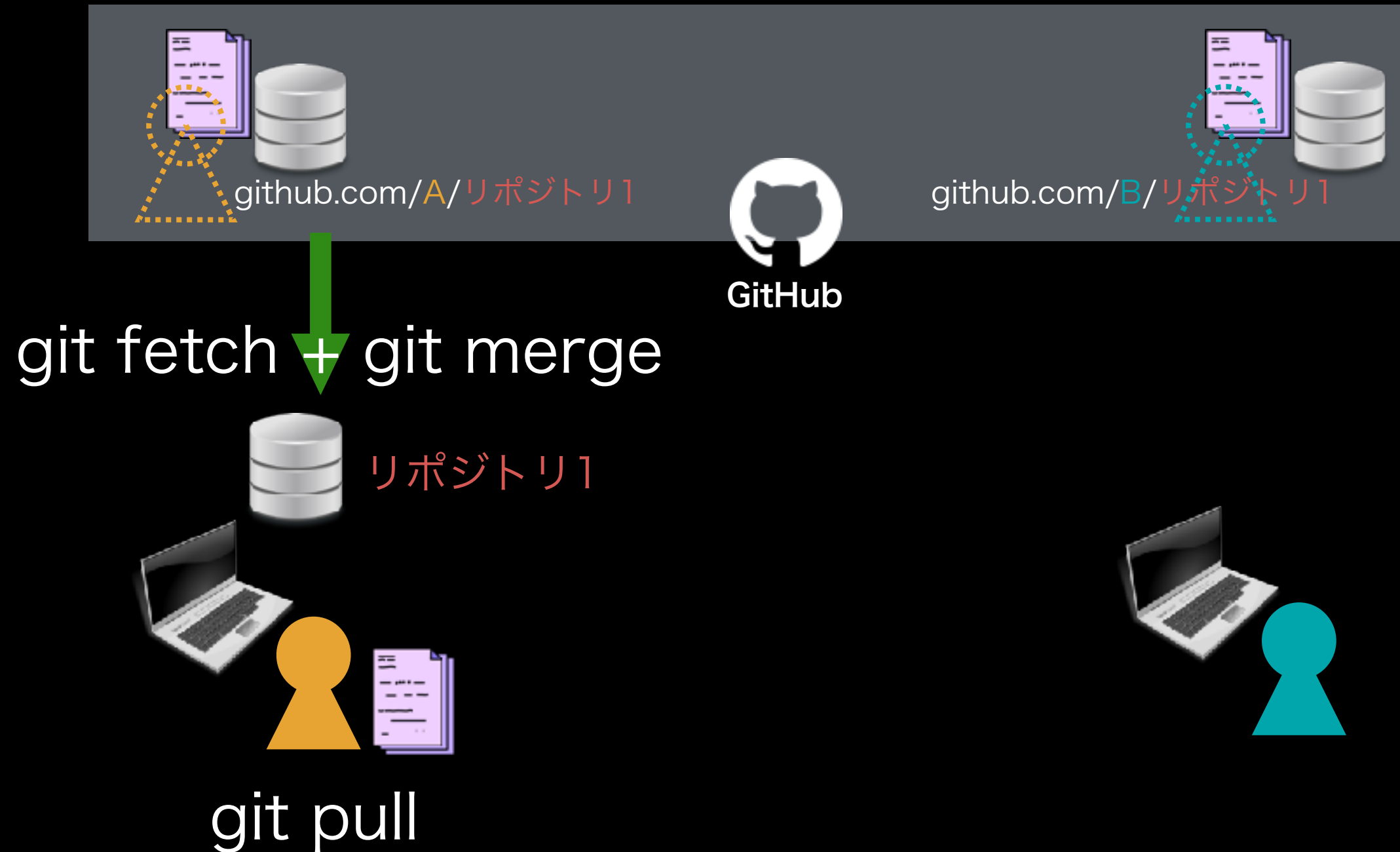
`git pull`



GitHub開発フロー



GitHub開発フロー



GitHub開発フロー



`git pull`



GitHub開発フロー



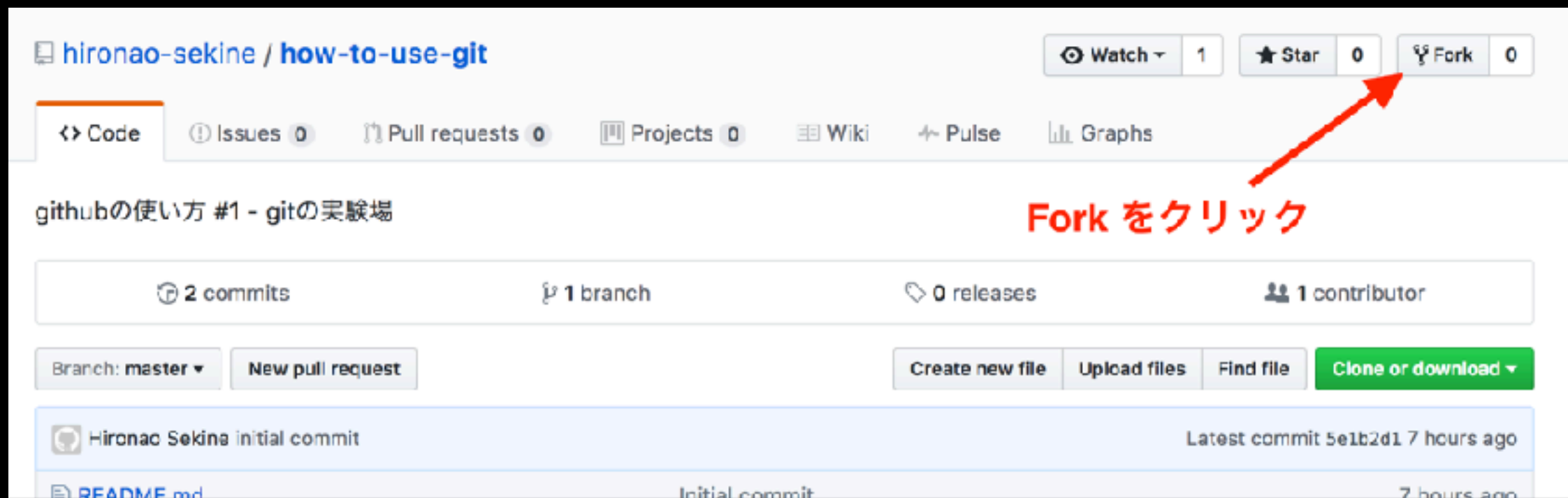
実践 git & GitHub

- ・ fork (特に開発に参加する意思が無い場合、Forkは不要です)
- ・ git clone
- ・ git status
- ・ 変更を加えて…
- ・ git status
- ・ git add
- ・ git commit
- ・ git push
- ・ pull request

Fork

- まずは fork してみましょう

<https://github.com/hironao-sekine/how-to-use-git>



Fork

- まずは fork してみましよう



Fork



git clone

- 次にcloneするための情報を表示させて…

The screenshot shows the GitHub interface for the repository `alid-hiro / how-to-use-git`, which is a fork of `hironao-sekine/how-to-use-git`. The repository has 2 commits, 1 branch, 0 releases, and 1 contributor. The `Code` tab is selected, showing the commit history and file list. A red arrow points to the `Clone or download` button, with the text **Clone or download をクリック** (Click Clone or download). Another red arrow points to the `Use SSH` link in the clone dropdown menu, with the text **Use SSH をクリック** (Click Use SSH). The dropdown menu shows the option to clone with HTTPS, with the URL `https://github.com/alid-hiro/how-to-use-git` and buttons for `Open in Desktop` and `Download ZIP`.

alid-hiro / how-to-use-git
forked from hironao-sekine/how-to-use-git

Watch 0 Star 0 Fork 2

Code Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

githubの使い方 #1 - gitの実験場 Edit

Add topics

2 commits 1 branch 0 releases 1 contributor

Branch: master New pull request

Create new file Upload files Find file Clone or download

This branch is even with hironao-sekine:master.

Hironao Sekine initial commit

README.md Initial commit

new.txt initial commit

old.txt initial commit

Clone or download をクリック

Use SSH をクリック

Clone with HTTPS ⓘ Use Git or checkout with SVN using the web URL.

https://github.com/alid-hiro/how-to-use-git

Open in Desktop Download ZIP

Use SSH

git clone

- cloneするための情報をコピーします

The screenshot shows the GitHub interface for the repository 'alid-hiro / how-to-use-git'. The repository is forked from 'hironao-sekine/how-to-use-git'. The main content area shows the commit history with a table of files and their commit messages. A red arrow points from the text 'この文字列をコピー' (Copy this string) to the SSH URL 'git@github.com:alid-hiro/hcw-to-use-gi' in the 'Clone with SSH' dropdown menu. Another red arrow points from the text 'このボタンを押してもコピーできます' (You can also copy by pressing this button) to the 'Download ZIP' button.

alid-hiro / how-to-use-git
forked from hironao-sekine/how-to-use-git

Watch 0 Star 0 Fork 2

Code Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

githubの使い方 #1 - gitの実験場 Edit

Add topics

2 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

This branch is even with hironao-sekine:master.

Hironao Sekine initial commit	
README.md	Initial commit
new.txt	initial commit
old.txt	initial commit

Clone with SSH ⓘ Use an SSH key and passphrase from account.

git@github.com:alid-hiro/hcw-to-use-gi

Open in Desktop Download ZIP

8 hours ago

git clone

- ・ git clone コマンドの実行

```
$ git clone git@github.com:[自分のユーザー名]/how-to-use-git.git
```


git clone

- ・ git clone コマンドの実行
コマンドを打った場所に how-to-use-git というフォルダが出来ます
場所がわからない時は pwd コマンドを実行して確認してください

```
$ git clone git@github.com:[自分のユーザー名]/how-to-use-git.git
Cloning into 'how-to-use-git'...
remote: Counting objects: 7, done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 7 (delta 0), reused 7 (delta 0), pack-reused 0
Receiving objects: 100% (7/7), done.
```

GitHub開発フロー



GitHub開発フロー



github.com/[hironao-sekine](https://github.com/hironao-sekine/how-to-use-git)/how-to-use-git



github.com/[\[ユーザー名\]](#)/how-to-use-git



hironao-sekine



```
git clone git@github.com:/\[ユーザー名\]/how-to-use-git.git
```

GitHub開発フロー



リポジトリ…？

- ・ 中身を見てみましょう

```
$ cd ./how-to-use-git
```

```
$ ls -la
```

```
total 24
```

```
drwxr-xr-x      6 hiro  staff   204  5  9 01:53 .
drwxr-xr-x+ 205 hiro  staff 6970  5  9 02:12 ..
drwxr-xr-x     12 hiro  staff   408  5  9 01:53 .git
-rw-r--r--      1 hiro  staff    57  5  9 01:53 README.md
-rw-r--r--      1 hiro  staff    80  5  9 01:53 new.txt
-rw-r--r--      1 hiro  staff    80  5  9 01:53 old.txt
```

リポジトリ…？

- ・ 中身を見てみましょう

```
$ cd ./how-to-use-git
```

```
$ ls -la
```

```
total 24
```

```
drwxr-xr-x      6 hiro  staff   204  5  9 01:53 .  
drwxr-xr-x+ 205 hiro  staff 6970  5  9 02:12 ..  
drwxr-xr-x     12 hiro  staff  408  5  9 01:53 .git  
-rw-r--r--      1 hiro  staff   57  5  9 01:53 README.md  
-rw-r--r--      1 hiro  staff   80  5  9 01:53 new.txt  
-rw-r--r--      1 hiro  staff   80  5  9 01:53 old.txt
```

リポジトリ…！

- ・ 中身を見てみましょう

```
$ cd ./how-to-use-git
```

```
$ ls -la
```

```
total 24
```

```
drwxr-xr-x  5 9 01:53 .
```

```
drwxr-xr-x  5 9 02:12 ..
```

```
drwxr-xr-x  3 .git
```

```
-rw-r--r--  1 hiro staff  57 5 9 01:53 README.md
```

```
-rw-r--r--  1 hiro staff  80 5 9 01:53 new.txt
```

```
-rw-r--r--  1 hiro staff  80 5 9 01:53 old.txt
```

この .git フォルダが
「リポジトリ」そのものです

git status

- ・ git status コマンドの実行

```
$ git status
```


git status

- ・ git status コマンドの実行

```
$ git status
```

```
On branch master
```

```
Your branch is up-to-date with 'origin/master'.
```

```
nothing to commit, working tree clean
```

GitHub開発フロー



github.com/[hironao-sekine](https://github.com/hironao-sekine)/how-to-use-git



GitHub



github.com/[\[ユーザー名\]](#)/how-to-use-git

how-to-use-git



差分をチェック

git status

変更を加えて…

- ・ new.txt を開いて適当に編集して保存してから
もう一度 git status を実行してみましょう

```
$ git status
```

また git status

- ・ new.txt を開いて適当に編集して保存してから
もう一度 git status を実行してみましょう

```
$ git status
```

```
On branch master
```

```
Your branch is up-to-date with 'origin/master'.
```

```
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)
```

```
modified: new.txt
```

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

GitHub開発フロー



github.com/[hironao-sekine](https://github.com/hironao-sekine)/how-to-use-git



GitHub



github.com/[\[ユーザー名\]](#)/how-to-use-git

how-to-use-git



差分をチェック

git status

status メッセージの見方

- ・ new.txt を開いて適当に編集して保存してから
もう一度 git status を実行してみましょう

```
$ git status
```

```
On branch master
```

```
Your branch is up-to-date with 'origin/master'.
```

```
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)
```

```
modified: new.txt
```

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

status メッセージの見方

- ・ new.txt を開いて適当に編集して保存してから
もう一度 git status を実行してみましょう

```
$ git status
```

```
On branch master
```

```
Your branch is
```

リポジトリの情報と実体ファイルの間に差異が検出されたよ
(ステージされていない差分がある)

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)

modified: new.txt

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

status メッセージの見方

- ・ new.txt を開いて適当に編集して保存してから
もう一度 git status を実行してみましょう

```
$ git status
```

```
On branch master
```

```
Your branch is
```

リポジトリの情報と実体ファイルの間に差異が検出されたよ
(ステージされていない差分がある)

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)

```
modified: new.txt
```

どんな差異かというと、new.txt が変更されているよ

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

status メッセージの見方

- ・ new.txt を開いて適当に編集して保存してから
もう一度 git status を実行してみましょう

```
$ git status
```

```
On branch master
```

```
Your branch is
```

リポジトリの情報と実体ファイルの間に差異が検出されたよ
(ステージされていない差分がある)

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)

```
modified:   new.txt
```

どんな差異かというと、new.txt が変更されているよ

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

コミットに含める変更は無いよ

git diff

- ・ 変更の詳細をチェックする

```
$ git diff
```

```
diff --git a/new.txt b/new.txt
```

```
index 648b66f..d554e88 100644
```

```
--- a/new.txt
```

```
+++ b/new.txt
```

```
@@ -1,5 +1,4 @@
```

```
あいうえお
```

```
かきくけこ
```

```
たちつてと
```

```
-なにぬねの
```

```
-はひふへほ
```

```
+ほげほげほげ
```

git add

- ・ 「ステージ」 にnew.txtを登録します

```
$ git add new.txt
```

またまた git status

- ・ もう一度 git status を実行してみましよう

```
$ git status
```

またまた git status

- ・ もう一度 git status を実行してみましょう

```
$ git status
```

```
On branch master
```

```
Your branch is up-to-date with 'origin/master'.
```

```
Changes to be committed:
```

```
(use "git reset HEAD <file>..." to unstage)
```

```
modified:   new.txt
```

またまた git status

- ・ もう一度 git status を実行してみましょう

```
$ git status
```

```
On branch master
```

```
Your branch is up to date with 'master'.
```

```
Changes to be committed:
```

```
(use "git reset HEAD <file>..." to unstage)
```

```
modified:   new.txt
```

new.txt が変更されたという情報です

git commit

- ・ コミットしましょう！

```
$ git commit -m “このコミットの概要など”
```

git commit

- ・ コミットしましょう！

```
$ git commit -m “このコミットの概要など”  
[master 6505754] このコミットの概要など  
1 file changed, 1 insertion(+), 2 deletions(-)
```


GitHub開発フロー



github.com/[hironao-sekine](https://github.com/hironao-sekine)/how-to-use-git



GitHub



github.com/[\[ユーザー名\]](#)/how-to-use-git

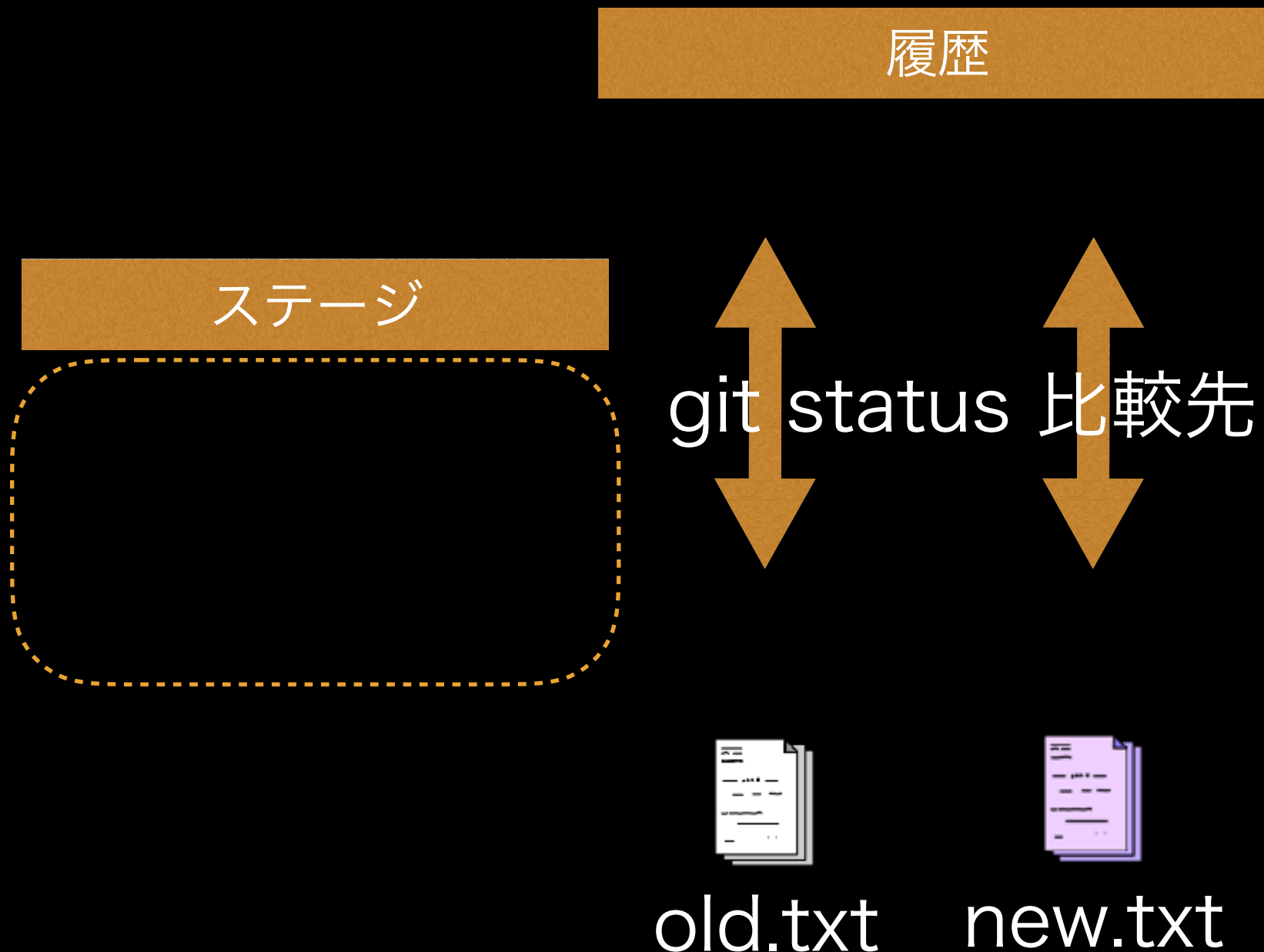
how-to-use-git



git add + git commit

ステージとは…

- git の 気持ち



ステージとは…

- ・ git の 気持ち

履歴

ステージ

git add new.txt



old.txt



new.txt

ステージとは…

- ・ git の 気持ち

履歴

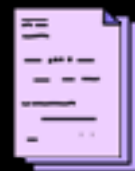
ステージ

new.txtをコミットに含める

```
git add new.txt
```



old.txt



new.txt

ステージとは…

- ・ git の 気持ち

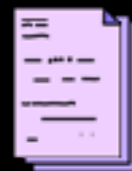
履歴

ステージ

new.txtをコミットに含める



old.txt



new.txt

ステージとは…

- ・ git の 気持ち

履歴

ステージ

new.txtをコミットに含める

git commit



コミット箱



old.txt



new.txt

ステージとは…

- ・ git の 気持ち

履歴

ステージ

git commit



コミット箱



old.txt



new.txt

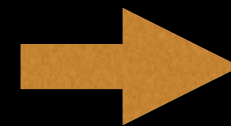
ステージとは…

- git の 気持ち

履歴

ステージ

git commit



new.txt

コミット箱



old.txt



new.txt

ステージとは…

- ・ git の 気持ち

履歴

ステージ

git commit



コミット箱



old.txt



new.txt

ステージとは…

- ・ git の 気持ち

履歴

ステージ

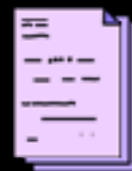
git commit



コミット箱



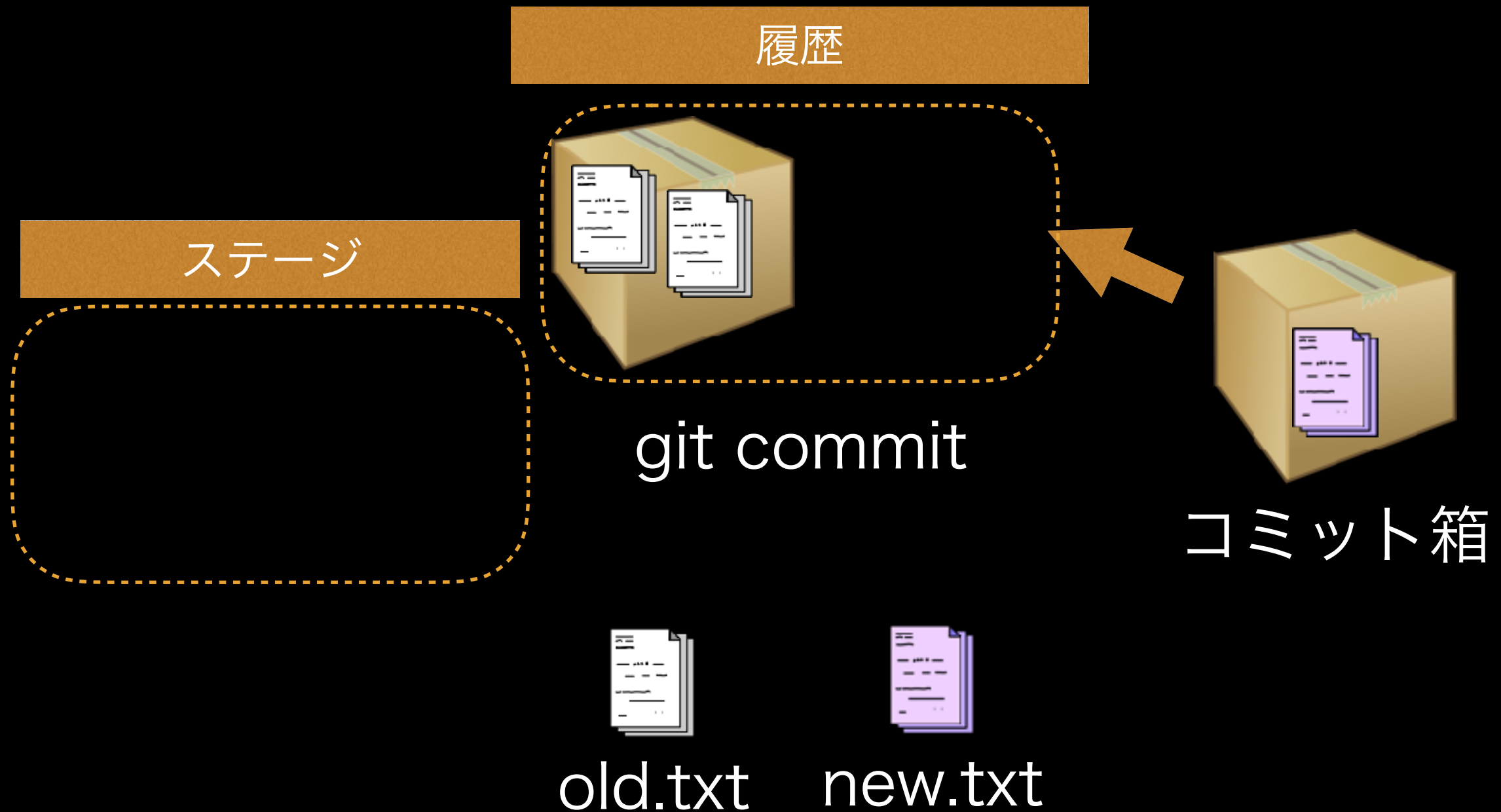
old.txt



new.txt

ステージとは…

- git の 気持ち

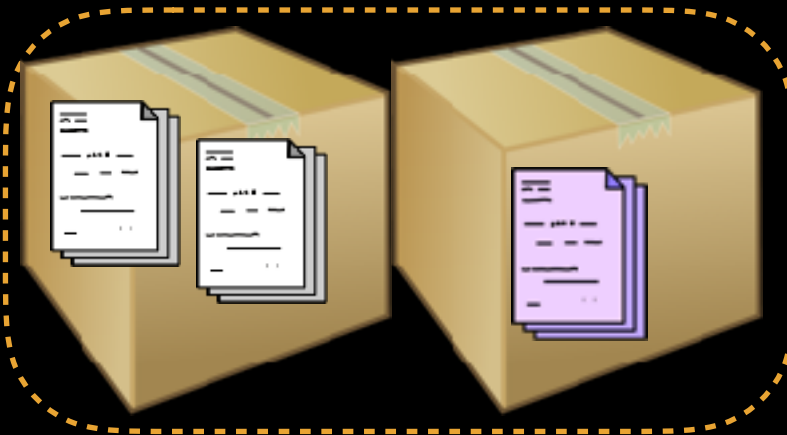


ステージとは…

- git の 気持ち

履歴

ステージ



git commit



old.txt



new.txt

git push

- ・ git pushしてGitHub上の自分のリポジトリに反映しましょう！

```
$ git push
```

git push

- ・ git pushしてGitHub上の自分のリポジトリに反映しましょう！

```
$ git push
```

```
Counting objects: 3, done.
```

```
Delta compression using up to 4 threads.
```

```
Compressing objects: 100% (3/3), done.
```

```
Writing objects: 100% (3/3), 398 bytes | 0 bytes/s, done.
```

```
Total 3 (delta 0), reused 0 (delta 0)
```

```
To github.com:alid-hiro/how-to-use-git.git
```

```
5e1b2d1..6505754 master -> master
```

GitHub開発フロー



実践 git & GitHub

- ・ ~~fork (特に開発に参加する意思が無い場合、Forkは不要です)~~
- ・ ~~git clone~~
- ・ ~~git status~~
- ・ ~~変更を加えて...~~
- ・ ~~git status~~
- ・ ~~git add~~
- ・ ~~git commit~~
- ・ ~~git push~~
- ・ pull request

実践 git & GitHub

- ・ ~~fork (特に開発に参加する意思が無い場合、Forkは不要です)~~

- ・ ~~git clone~~

- ・ ~~git status~~

- ・ ~~変更を加えて...~~

- ・ ~~git status~~

- ・ ~~git add~~

- ・ ~~git commit~~

- ・ ~~git push~~

- ・ pull request

ローカルのgitリポジトリと
GitHubの接合点は clone & push (& pull)
の3つ！

実践 git & GitHub

- ・ ~~fork (特に開発に参加する意思が無い場合、Forkは不要です)~~

- ・ ~~git clone~~

- ・ ~~git status~~

- ・ ~~変更を加えて...~~

- ・ ~~git status~~

- ・ ~~git add~~

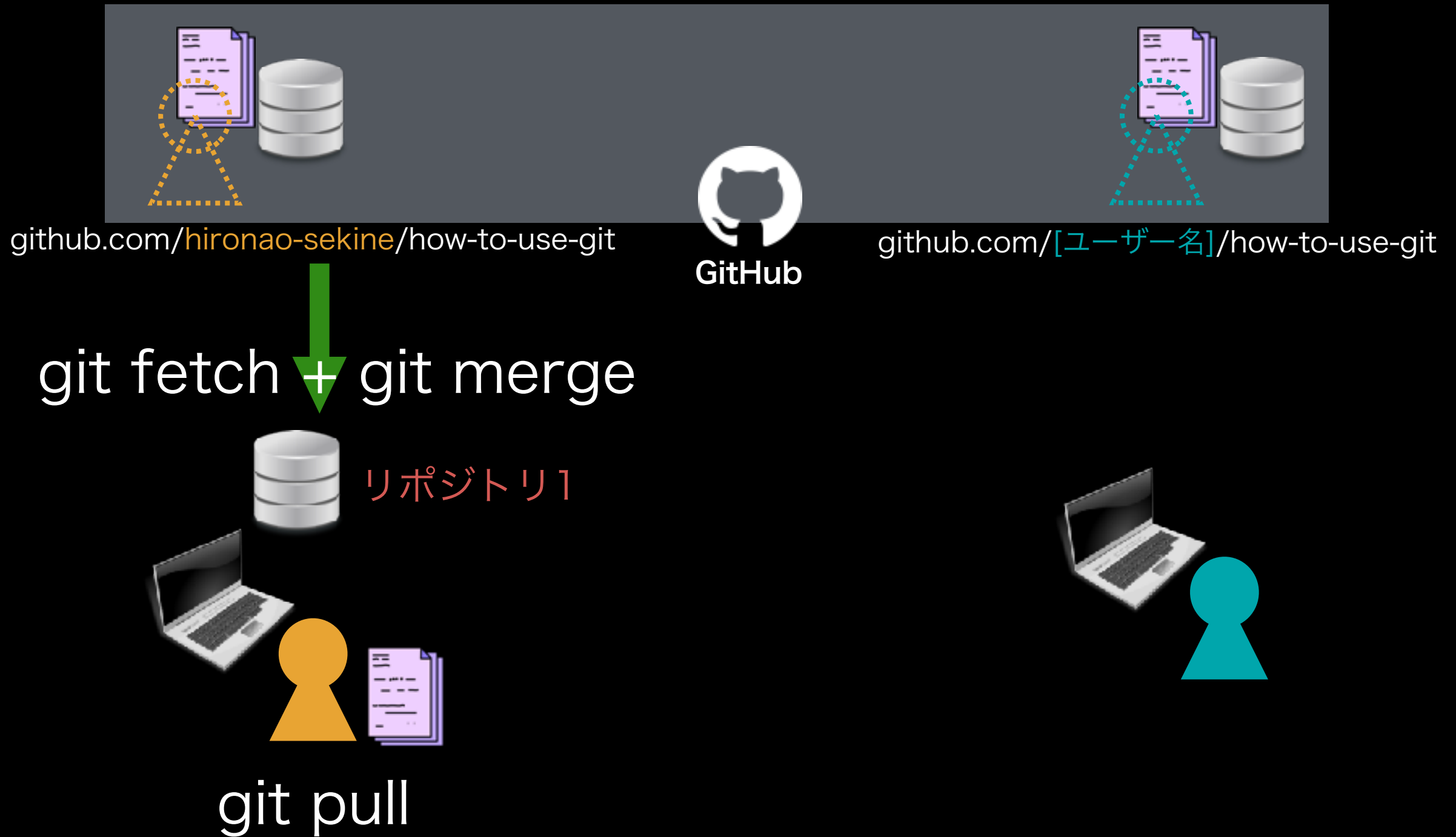
- ・ ~~git commit~~

- ・ ~~git push~~

- ・ pull request

ローカルのgitリポジトリと
GitHubの接合点は clone & push (& pull)
の3つ！

GitHub開発フロー



実践 git & GitHub

- ・ ~~fork (特に開発に参加する意思が無い場合、Forkは不要です)~~

- ・ ~~git clone~~

- ・ ~~git status~~

- ・ ~~変更を加えて...~~

- ・ ~~git status~~

- ・ ~~git add~~

- ・ ~~git commit~~

- ・ ~~git push~~

- ・ pull request

ローカルのgitリポジトリと
GitHubの接合点は clone & push (& pull)
の3つ！

その他 git コマンド

- ・ git init
- ・ git log
- ・ git branch
- ・ git checkout
- ・ git revert
- ・ git rebase
- ・ git reset
- ・ git reflog

git の操作に慣れるなら

- <http://learngitbranching.js.org/>

GUIでgitを操作するなら

- ・ Macなら SourceTree 一択
- ・ Win でも沢山あるけれど...
- ・ gitコマンドそのものに「どんなものがあって」「どうなるのか？」が分かっていないとつらい...

間に合いませんでした

- ・ 最後に、GitHubのIssuesを…。

総まとめ

1. Gitとは

1. Gitとは
2. 差分
3. C/S型・分散型

2. GitHubとは

1. GitHubとは
2. ~~gitコマンド~~GitHubアカウントの準備
3. GitHub開発フロー
4. ~~実践 git & GitHub~~

総まとめ

1. Gitとは

1. Gitとは => バージョン管理ツールの一種である
2. 差分
3. C/S型・分散型

2. GitHubとは

1. GitHubとは
2. ~~gitコマンドGitHubアカウントの準備~~
3. GitHub開発フロー
4. ~~実践 git & GitHub~~

総まとめ

1. Gitとは

1. Gitとは => バージョン管理ツールの一種である
2. 差分 => ファイル同士の「差」を表すデータ
3. C/S型・分散型

2. GitHubとは

1. GitHubとは
2. ~~gitコマンドGitHubアカウントの準備~~
3. GitHub開発フロー
4. ~~実践 git & GitHub~~

総まとめ

1. Gitとは

1. Gitとは => バージョン管理ツールの一種である
2. 差分 => ファイル同士の「差」を表すデータ
3. C/S型・分散型 => git は 分散型であり各々がリポジトリのフルコピーを持てる

2. GitHubとは

1. GitHubとは
2. ~~gitコマンドGitHubアカウントの準備~~
3. GitHub開発フロー
4. ~~実践 git & GitHub~~

総まとめ

1. Gitとは

1. Gitとは => バージョン管理ツールの一種である
2. 差分 => ファイル同士の「差」を表すデータ
3. C/S型・分散型 => git は 分散型であり各々がリポジトリのフルコピーを持てる

2. GitHubとは

1. GitHubとは => gitのリポジトリを公開/Fork/Pull Reqできる場所
2. ~~gitコマンドGitHubアカウントの準備~~
3. GitHub開発フロー
4. ~~実践 git & GitHub~~

総まとめ

1. Gitとは

1. Gitとは => バージョン管理ツールの一種である
2. 差分 => ファイル同士の「差」を表すデータ
3. C/S型・分散型 => git は 分散型であり各々がリポジトリのフルコピーを持てる

2. GitHubとは

1. GitHubとは => gitのリポジトリを公開/Fork/Pull Reqできる場所
2. ~~gitコマンドGitHubアカウントの準備~~
3. GitHub開発フロー => forkしてcloneしてgit使って開発してpushしてPull Req
4. ~~実践 git & GitHub~~