

GitHub勉強会

2016/3/22

担当者:井上 裕文

1. GitHubとは

GitHubとは

- Gitを使うためのhub(中心・拠点)
 - コードを共有/公開するためのサービス
 - プログラマのためのSNS
 - 様々なプロジェクトのためにGitのリポジトリをホスティングするサービス

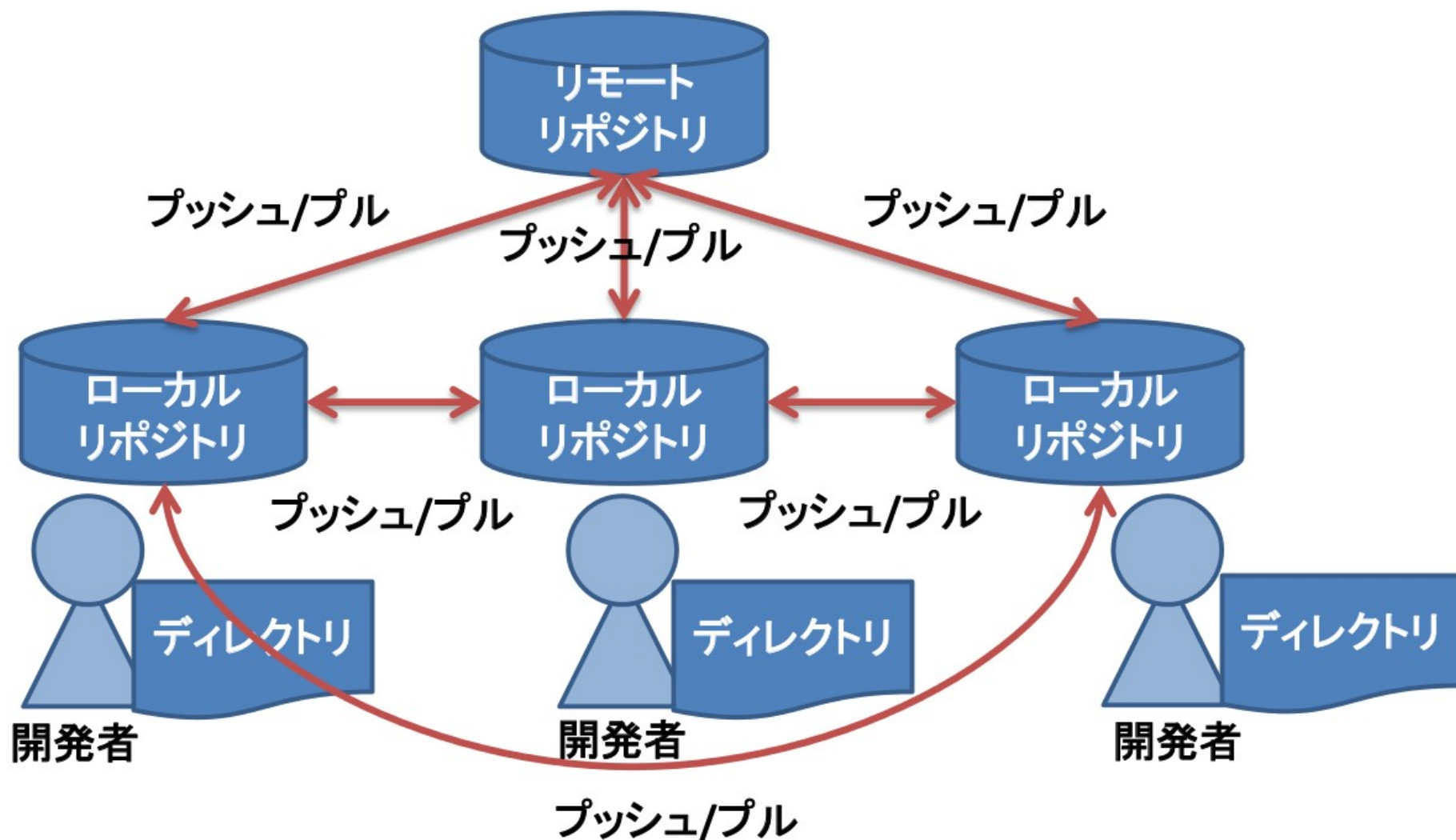
バージョン管理システム

- 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記録し管理するシステム
- バージョン管理システムには主に2種類ある
 - 集中型バージョン管理システム
例) Subversion(SVN), CVS
 - 分散型バージョン管理システム
例) Git, Mercurial

バージョン管理システムを使用しない 場合のデメリット

- 変更が1行の場合でも、ファイルのコピー全体を保存しておく必要がある
- 同じ日に2つの異なるバージョンを保存する必要がある場合、番号付けはより複雑になる
- 2人のユーザーが同じ日にファイルを編集する場合は考えられる
- 多くのバージョンのファイルが保存されるため、プロジェクトフォルダーがいっぱいになる
- ハードドライブが故障した場合、ファイルの履歴全体が消えてしまう
- 同僚に「XとYの間の変更を確認して」と伝える場合、これらのバージョンをそれぞれのユーザーにおくる必要がある

分散型バージョン管理システム

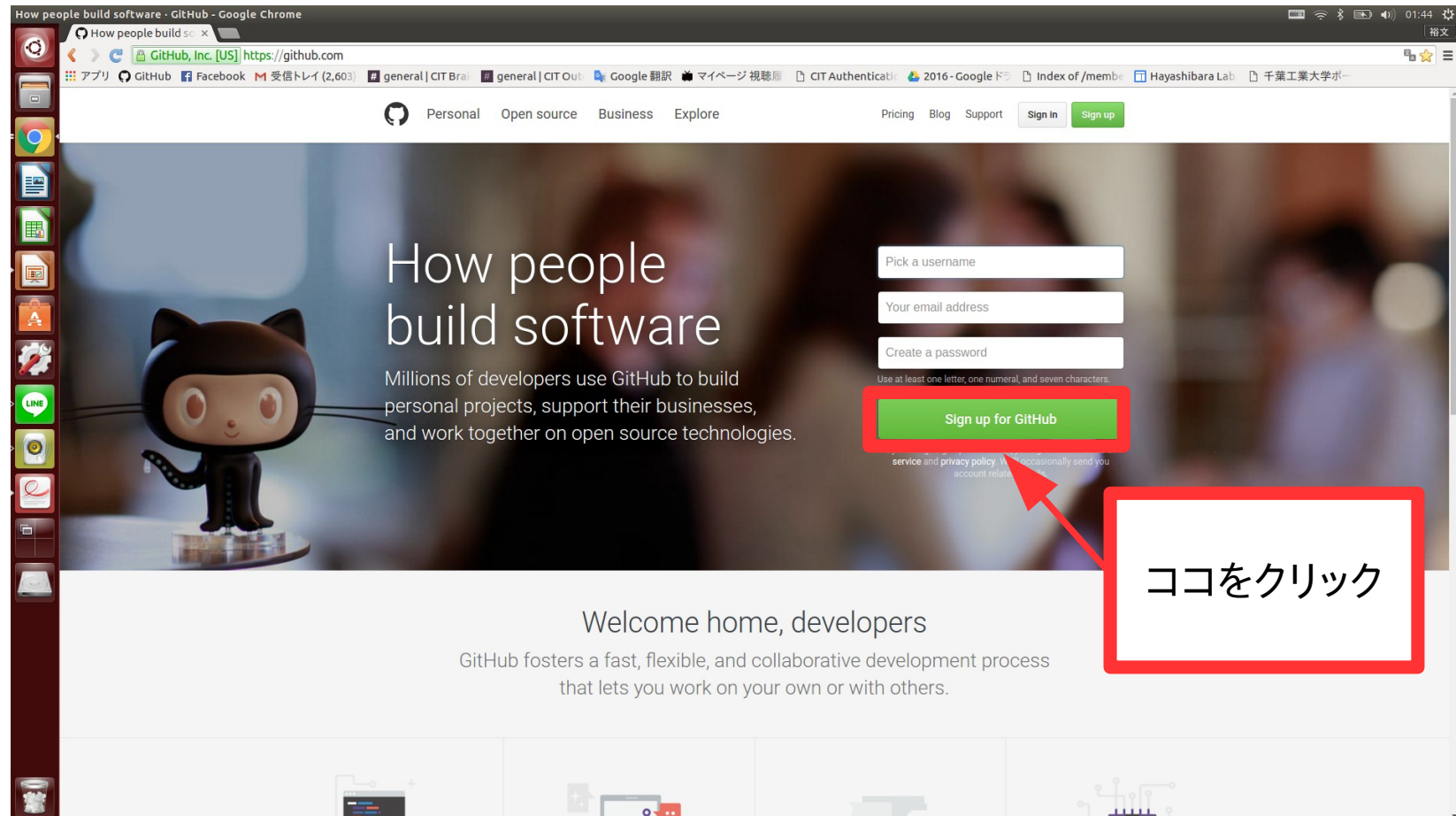


分散型バージョン管理システムの利点

- チェンジセットのプッシュやプル以外の操作の実行は非常に高速
- 誰の目にもさらされることなく、新しいチェンジセットのコミットをローカルで行える
- プッシュとプル以外のすべての操作はインターネット接続なしに行える
- プログラマーはプロジェクト レポジトリの完全なコピーをそれぞれ持っているため、変更を同時に 1 人か 2
- 人のプログラマーと共有してフィードバックを得てから、その変更を全員に公開できる

2. 実際に使いながらバージョン 管理の流れを理解する

GitHubのアカウント作成



GitHubのアカウント作成

Join GitHub · GitHub · Google Chrome

Join GitHub · GitHub · Google 翻訳

GitHub, Inc. [US] https://github.com/join

アプリ GitHub Facebook 受信トレイ (2,603) # general | CIT Bra... general | CIT Out... Google 翻訳 マイページ 視聴... CIT Authentica... 2016 - Google ド... Index of /membe... Hayashibara Lab... 千葉工業大学ボ...

Personal Open source Business Explore Pricing Blog Support Sign in Sign up

Join GitHub

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

Step 1: Set up a personal account Step 2: Choose your plan Step 3: Go to your dashboard

Create your personal account

There were problems creating your account.

Username mob-humio ✓

Email Address kappasetuhirobunbun@yahoo.co.jp ✓

Password ***** ✓

You'll get unlimited public repositories

By clicking on "Create an account" below, you are agreeing to the [Terms of Service](#) and the [Privacy Policy](#).

Create an account

ユーザー名

メールアドレス

パスワード

GitHubのアカウント作成

The screenshot shows the GitHub account creation process. A red box highlights the 'Chosen' button in the 'Free' plan row, with an arrow pointing to it from a red box containing the text 'クリック' (Click). Another red box highlights the 'Finish sign up' button, with an arrow pointing to it from a red box containing the text 'クリック' (Click).

Welcome to GitHub
You've taken your first step into a larger world, @mob-humio.

Completed Set up a personal account Step 2: Choose your plan Step 3: Go to your dashboard

Choose your personal plan

Plan	Cost (view in JPY)	Private repositories	
Large	\$50/month	50	Choose
Medium	\$22/month	20	Choose
Small	\$12/month	10	Choose
Micro	\$7/month	5	Choose
Free	\$0/month	0	Chosen

Each plan includes:

- Unlimited collaborators
- Unlimited public repositories
- Free setup
- HTTPS Protection
- Email support
- Wikis

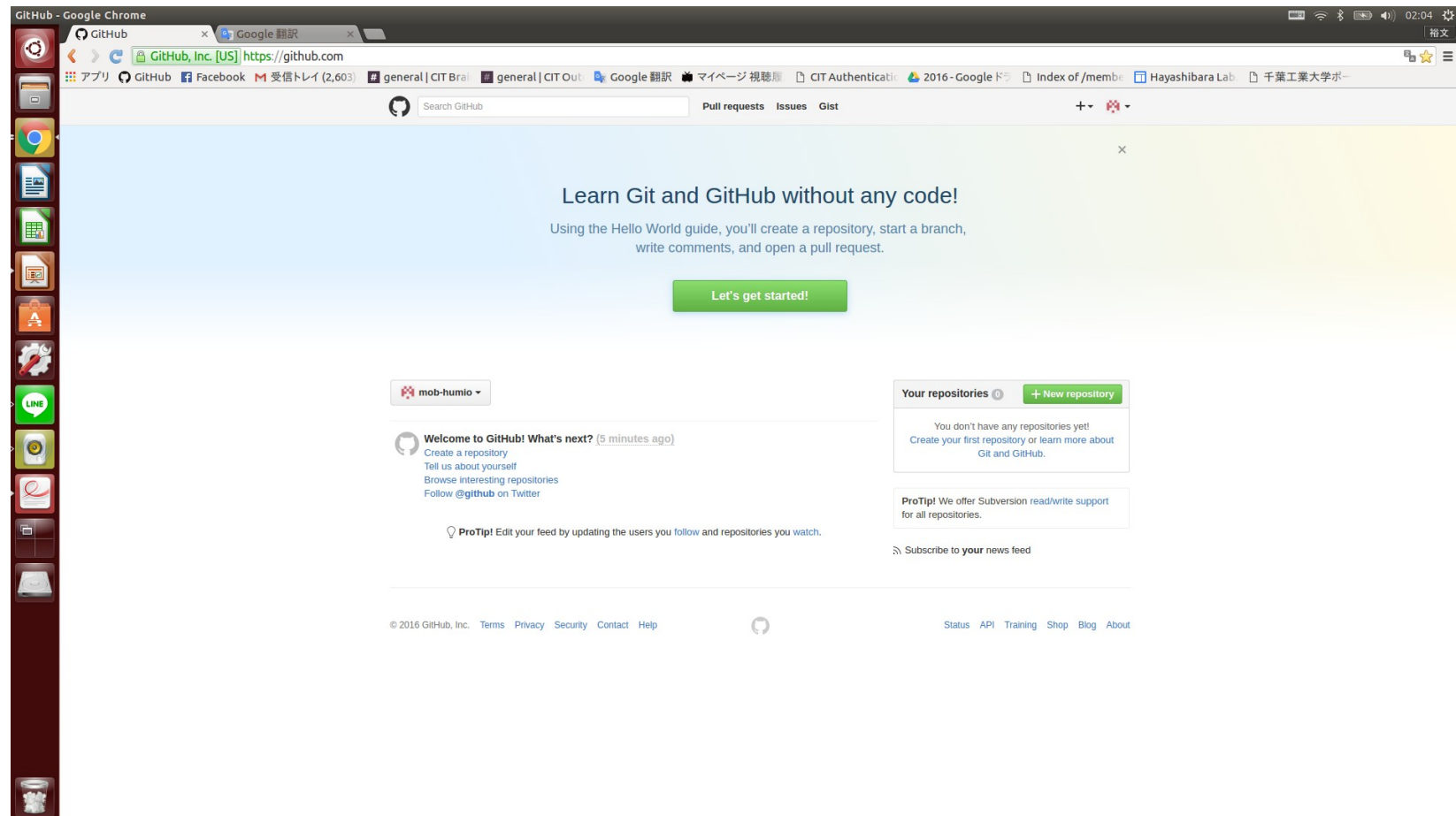
Charges to your account will be made in US Dollars. Converted prices are provided as a convenience and are only an estimate based on current exchange rates. Local prices will change as the exchange rate fluctuates. Don't worry, you can cancel or upgrade at any time.

☐ 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.](#)

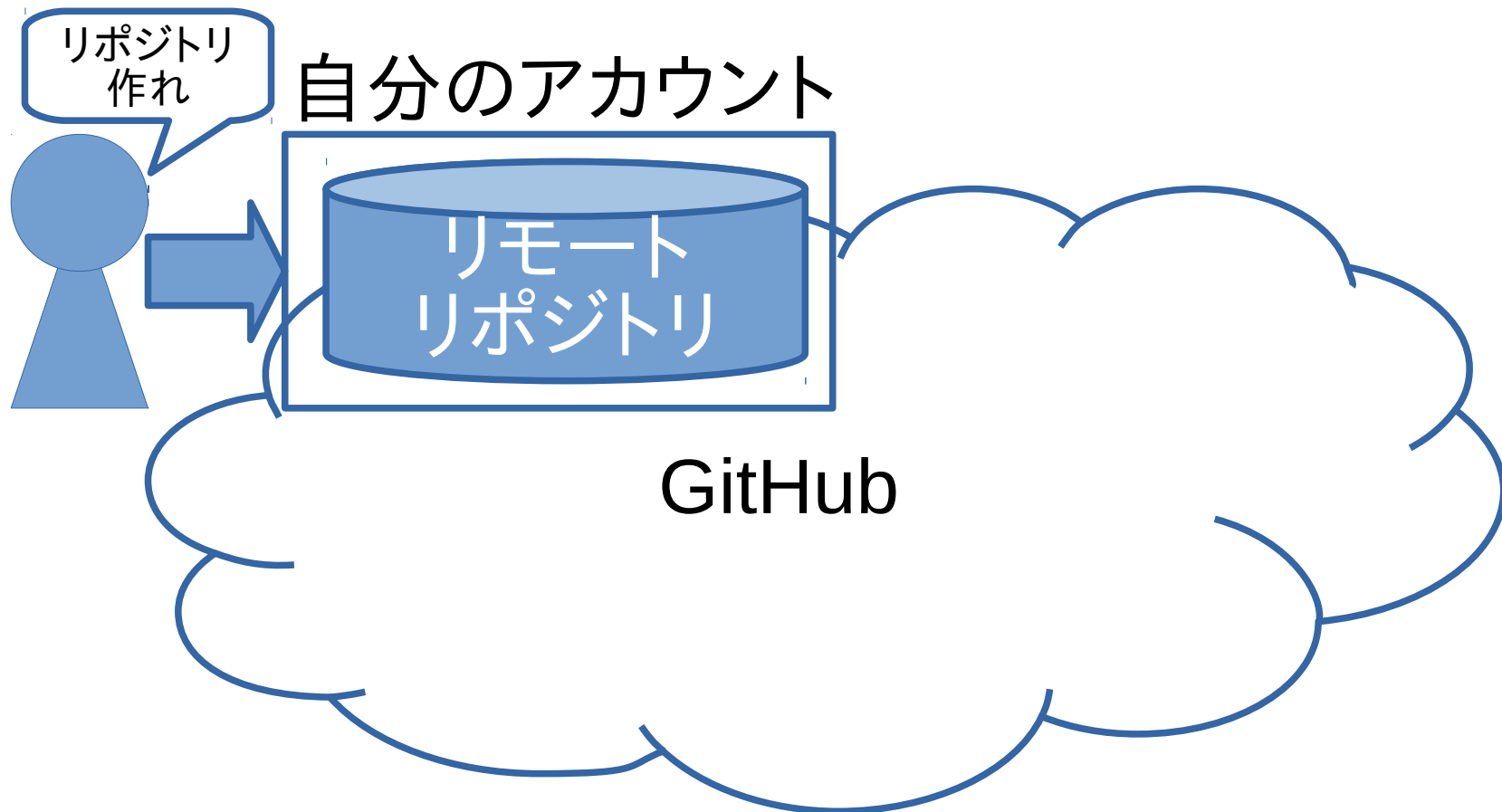
Finish sign up

© 2016 GitHub, Inc. Terms Privacy Security Contact Help Status API Training Shop Blog About

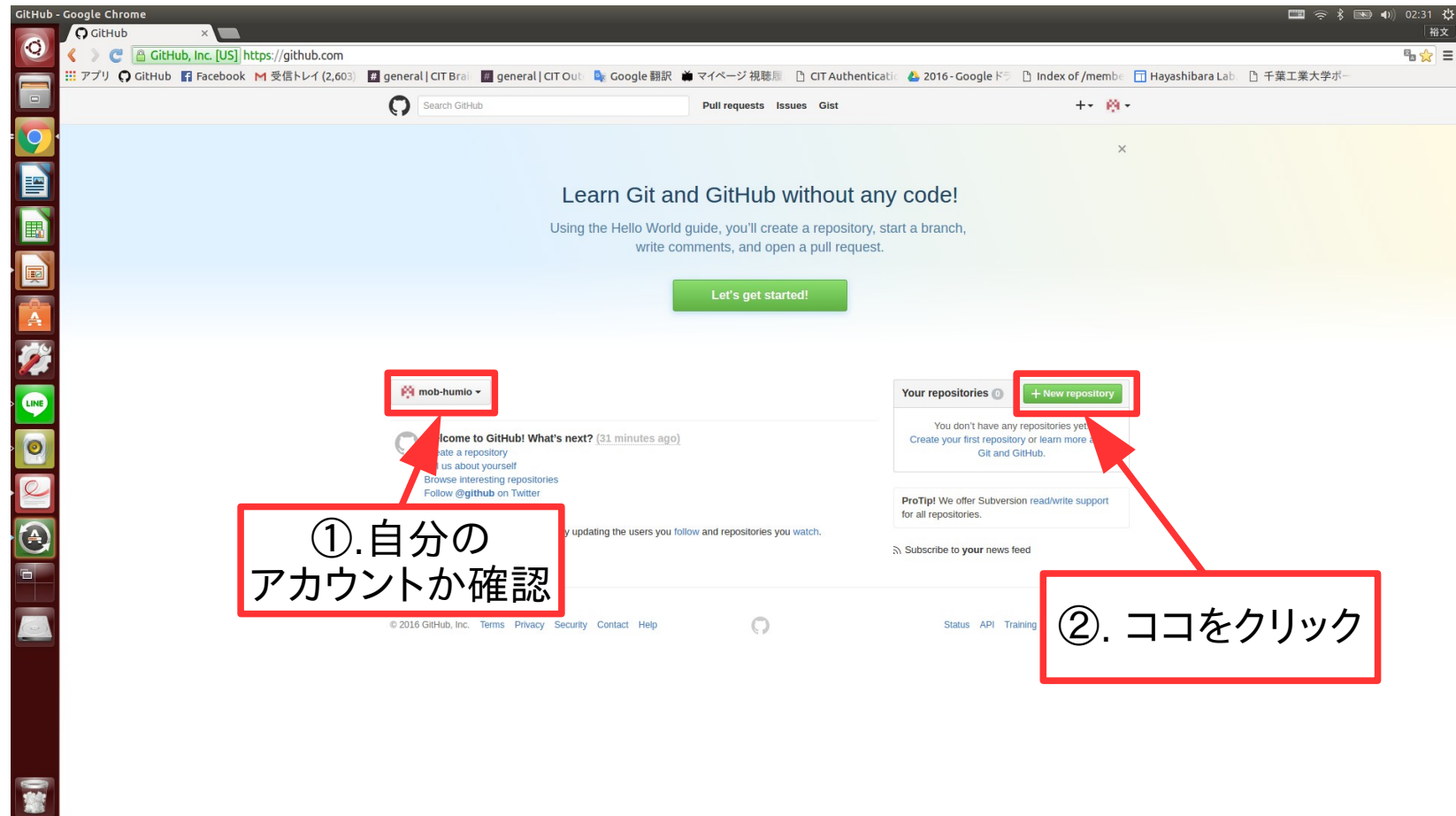
GitHubのアカウント完成



Github上にリポジトリを作成



GitHub上にリポジトリを作成する



GitHubにリポジトリを作成する

Create a New Repository - Google Chrome

Create a New Repository | GitHub, Inc. [US] | https://github.com/new

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: mob-humio

Repository name: test

Description (optional): test

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

Private: You choose who can see and commit to this repository.

☒ Initialize this repository with a README

Add .gitignore: None

Add a license: None

Create repository

①. リポジトリ名

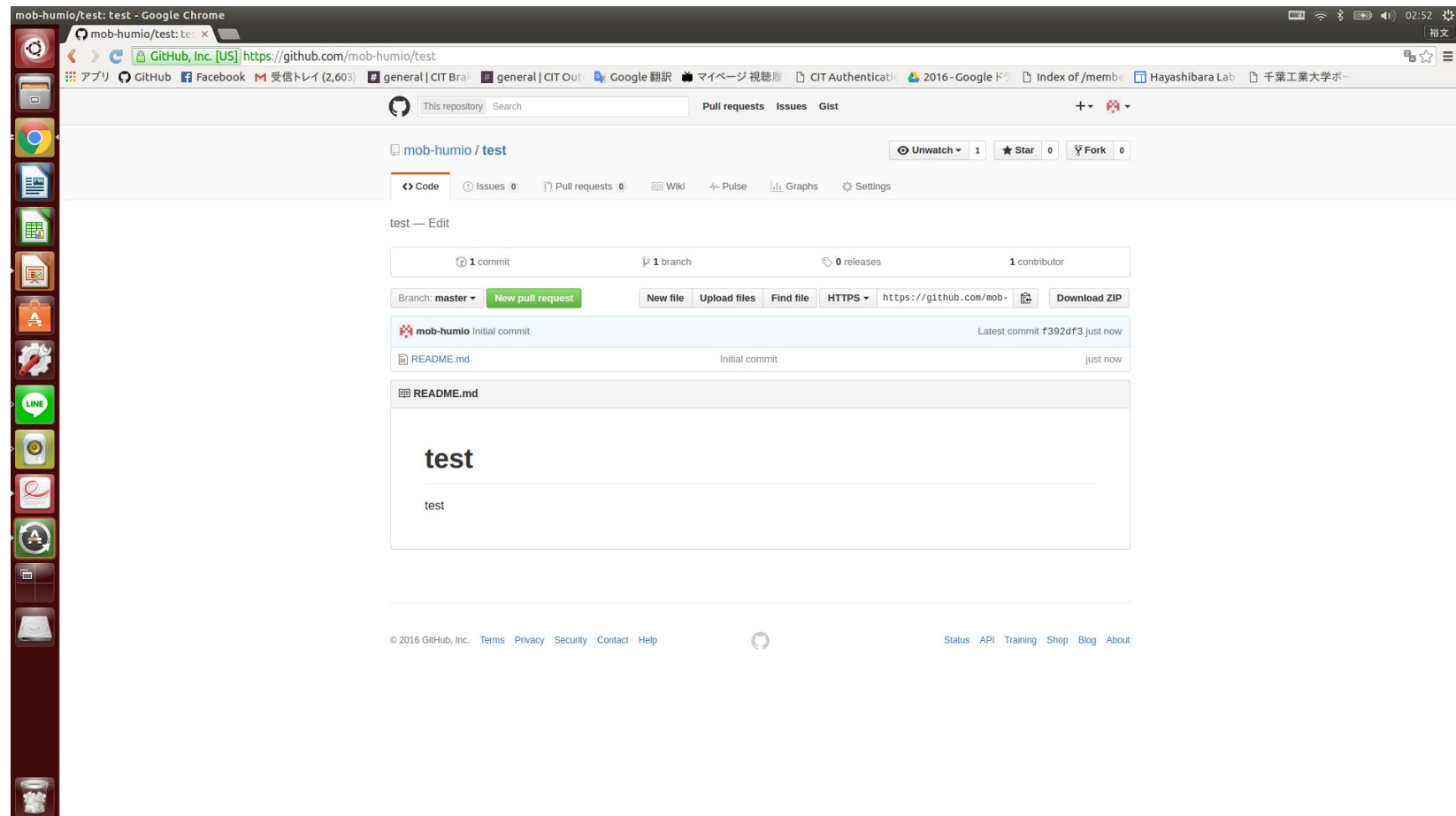
②. 説明

③. 公開か非公開か

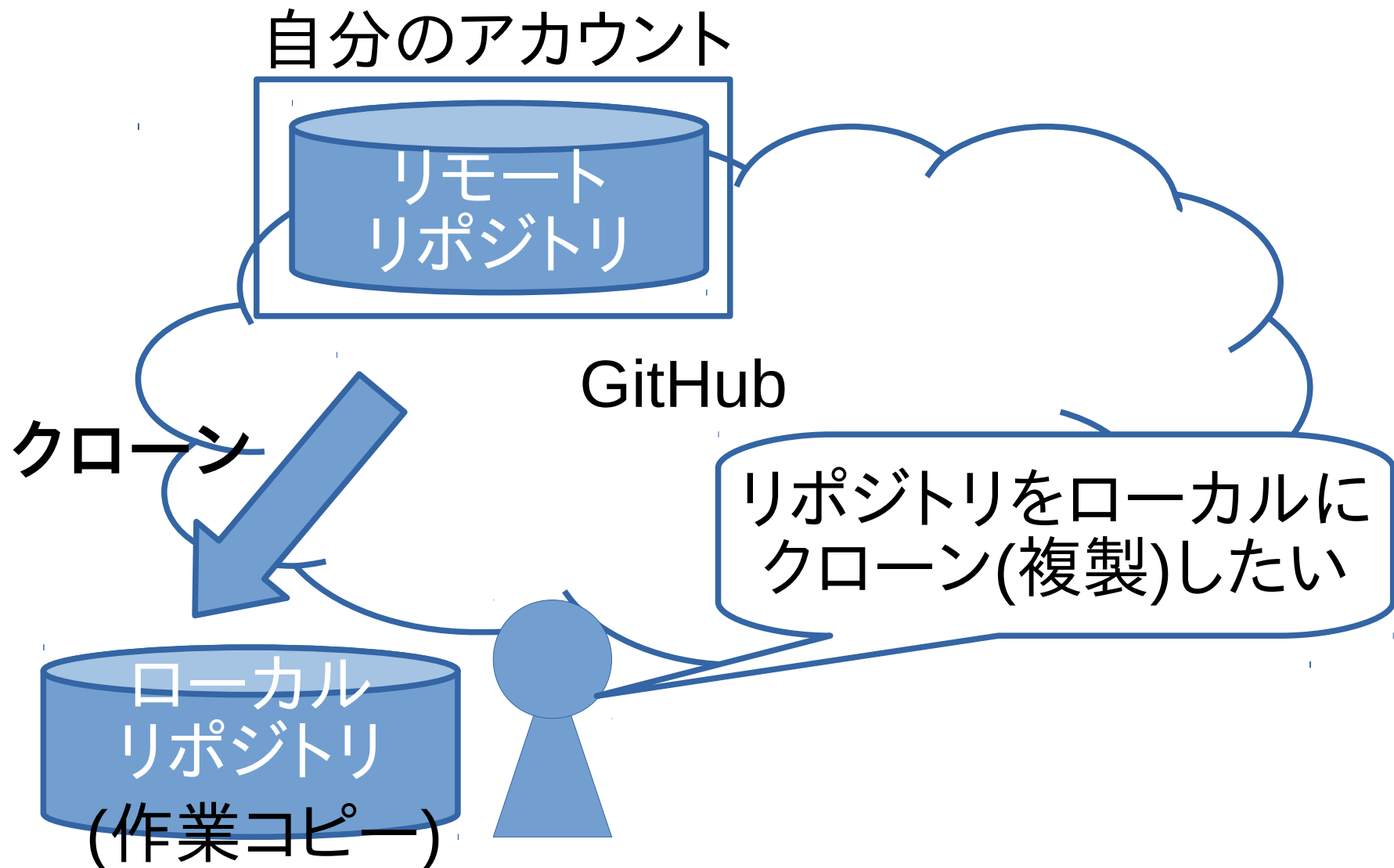
④. とりあえずチェック (READMEが勝手にできます)

⑤. ココをクリック

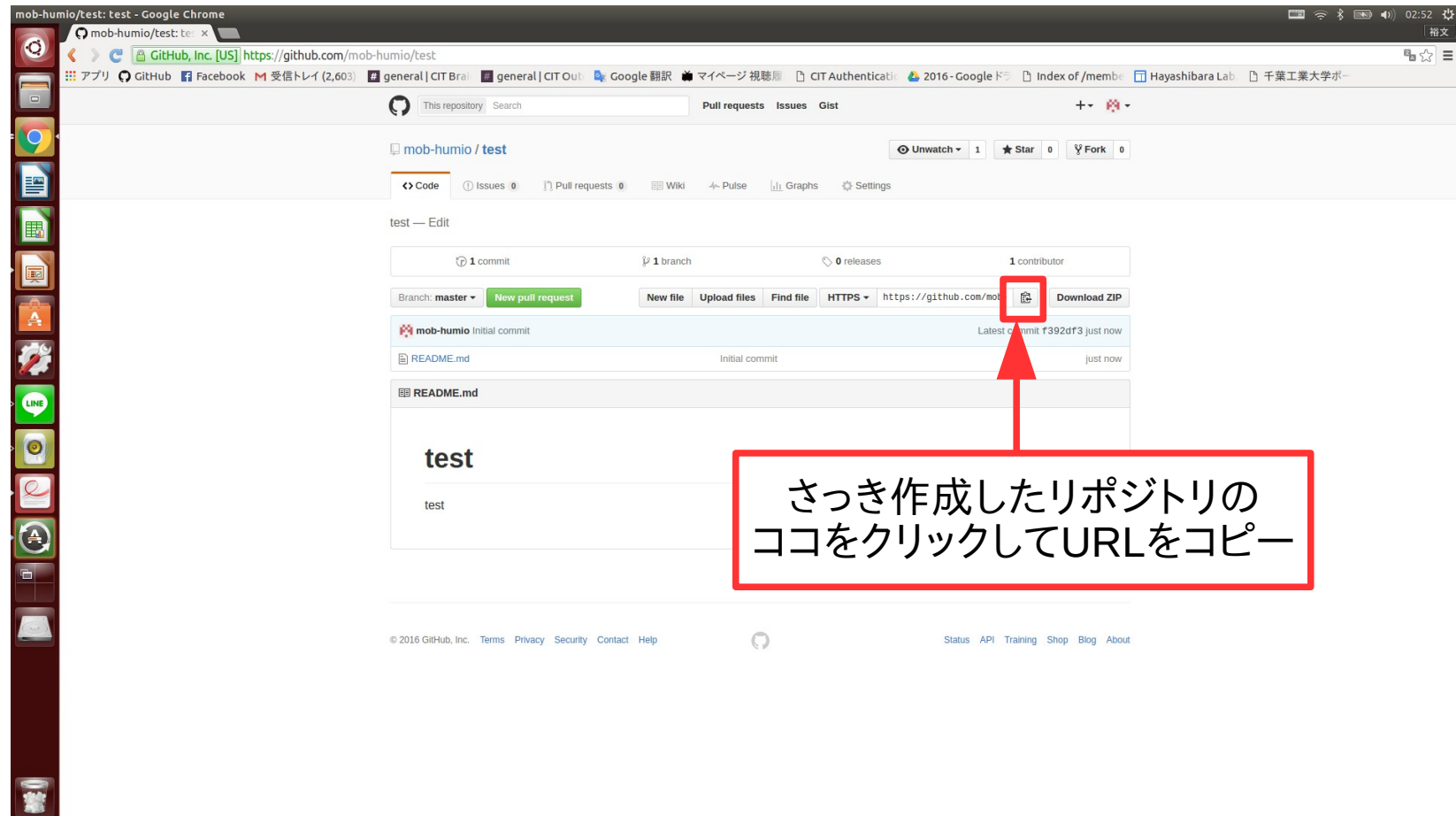
GitHub上にリポジトリが完成



リポジトリをローカルにクローン



リポジトリをローカルにクローン



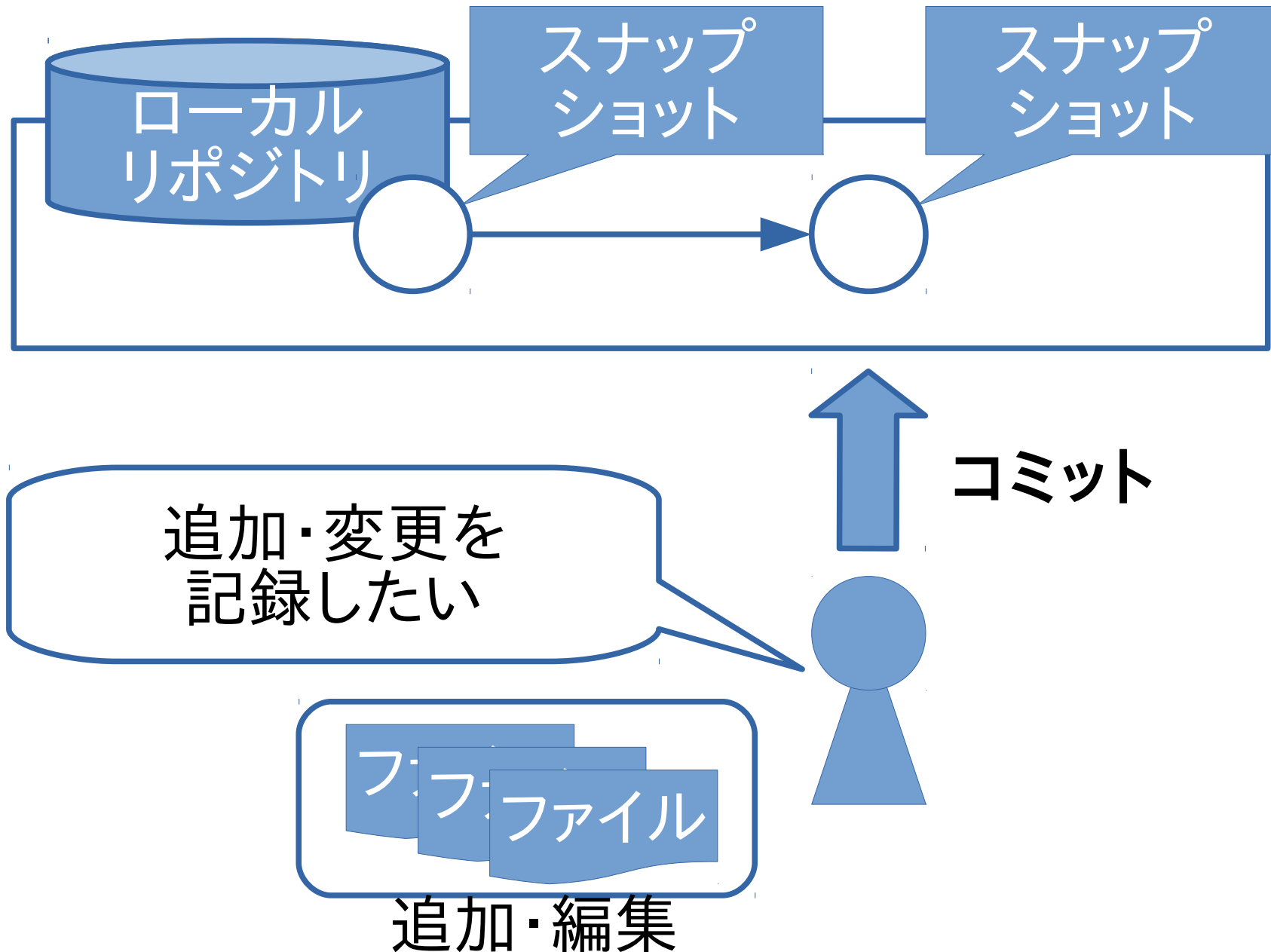
今日の約束

- 頭の'\$'は入力しない
- '␣'は半角スペース
- git が入っていない場合
\$ apt-get␣install␣git

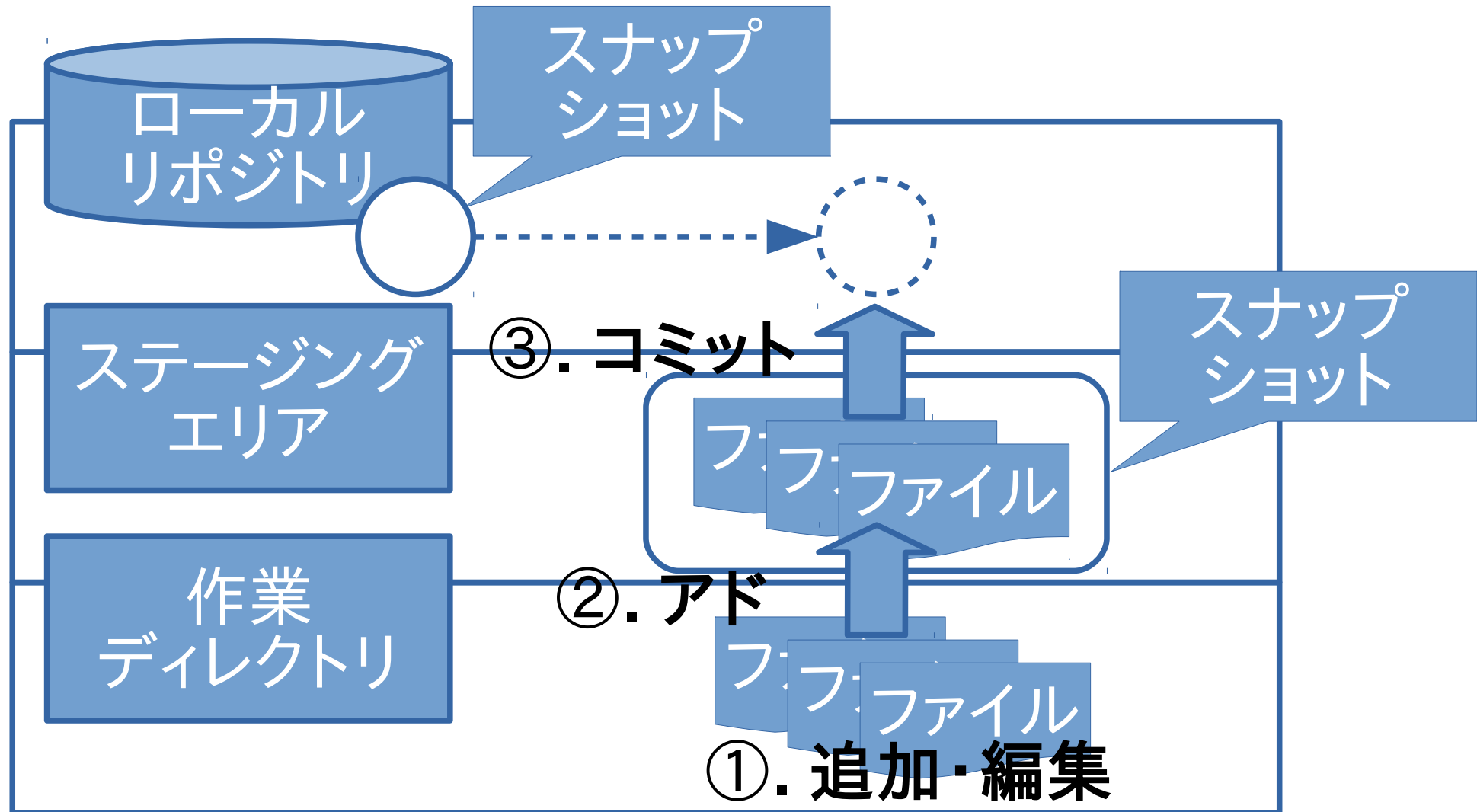
リポジトリをローカルにクローン

- ターミナル起動(CTRL+ALT+t)
- `$ git clone <さっきコピーしたURL>`
- `$ cd <リポジトリ名>`
- `$ ls`
- README.mdとか表示されたらOK

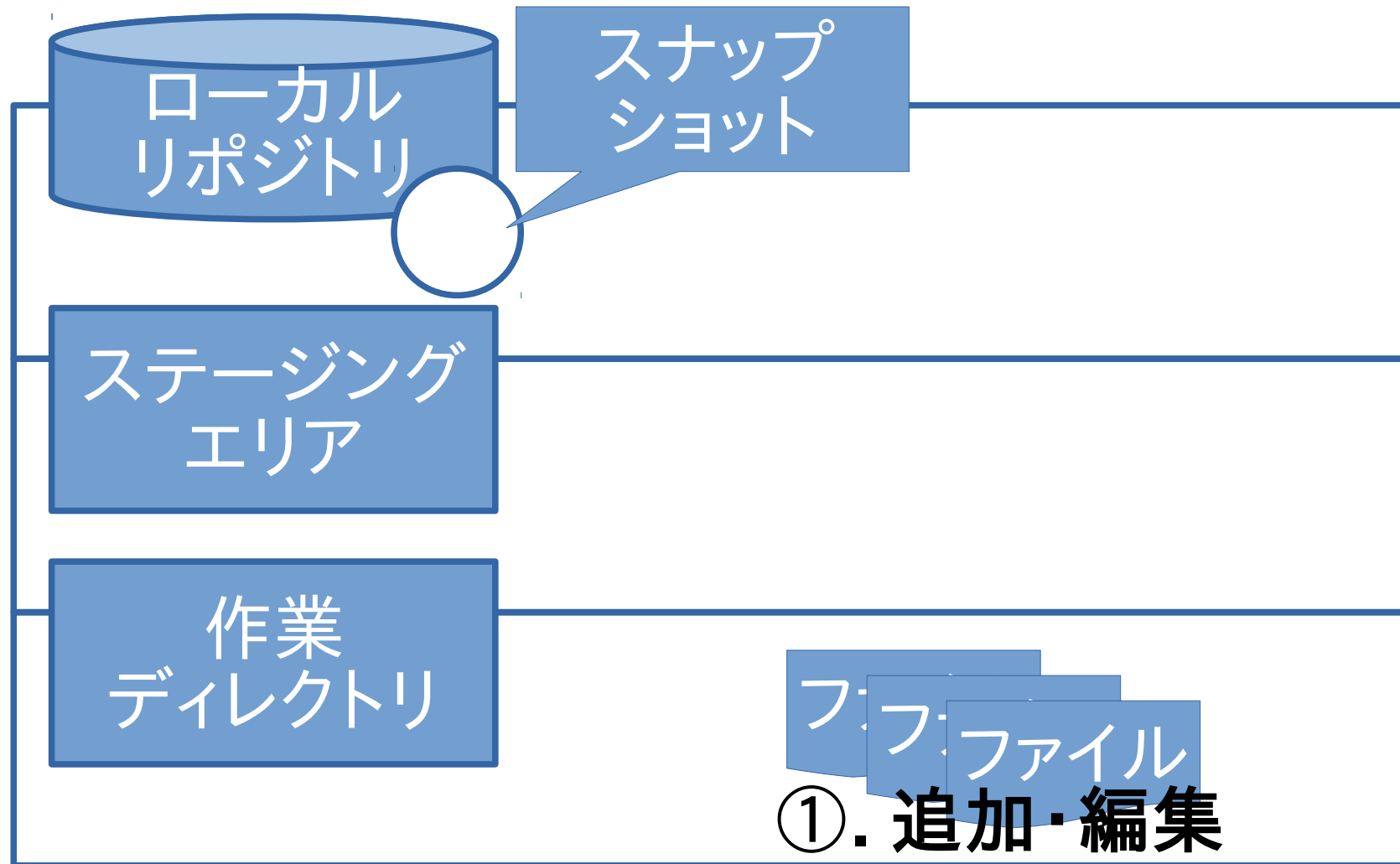
ローカルリポジトリ上でコミット



ローカルリポジトリ上でコミット



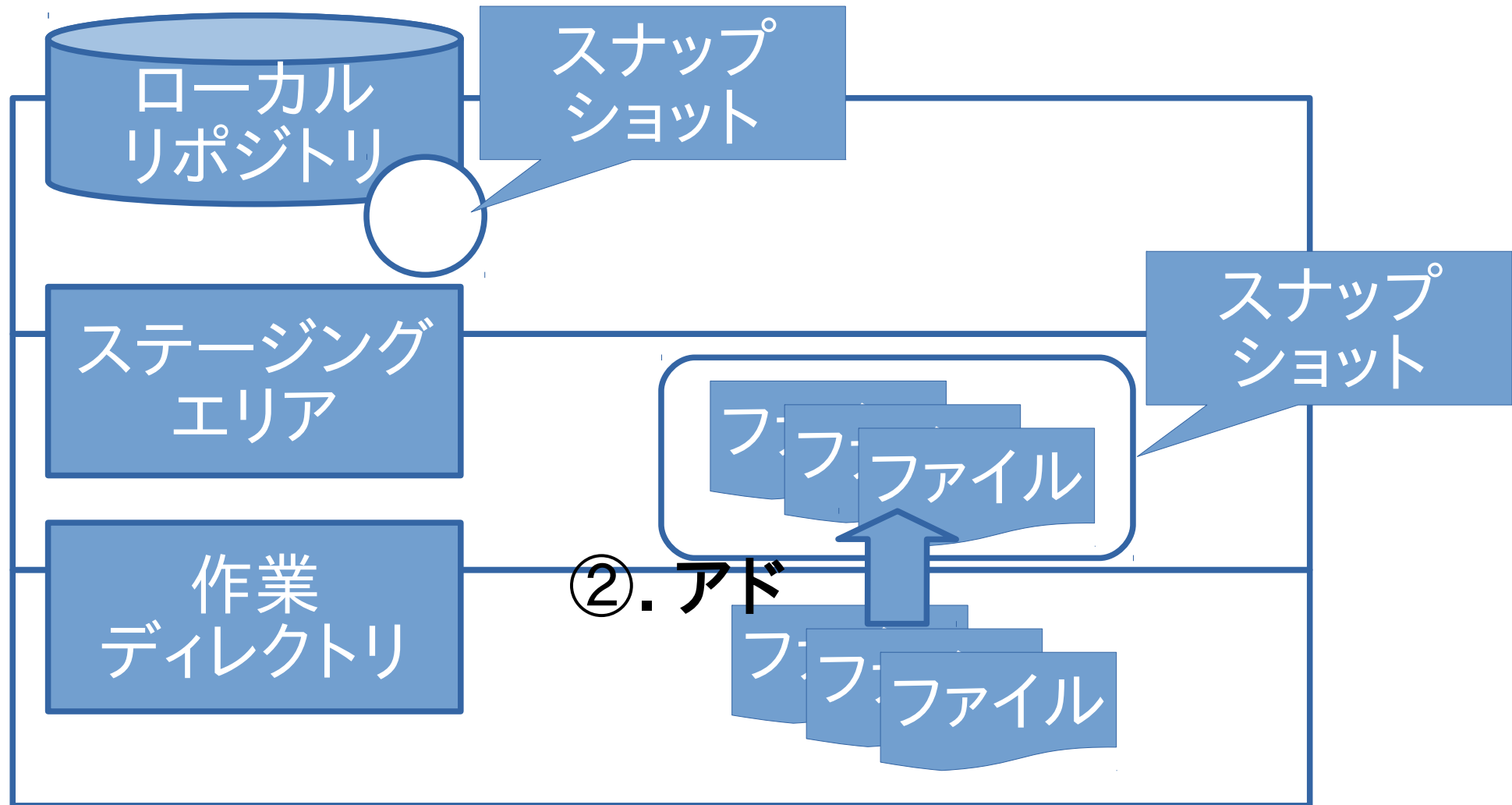
ローカルリポジトリ上でコミット



ローカルリポジトリ上でコミット

- さっきのターミナルで
- `$ touch <新しく追加したいファイル名>`
例)`$ touch test.txt`
- `$ git status`
- ↑ 作業ディレクトリの状態とステージングされたスナップショットの状態を表示するコマンド
- Untracked files(Gitによる追跡の対象外となっているファイル)に新しく追加したファイル名があるはず

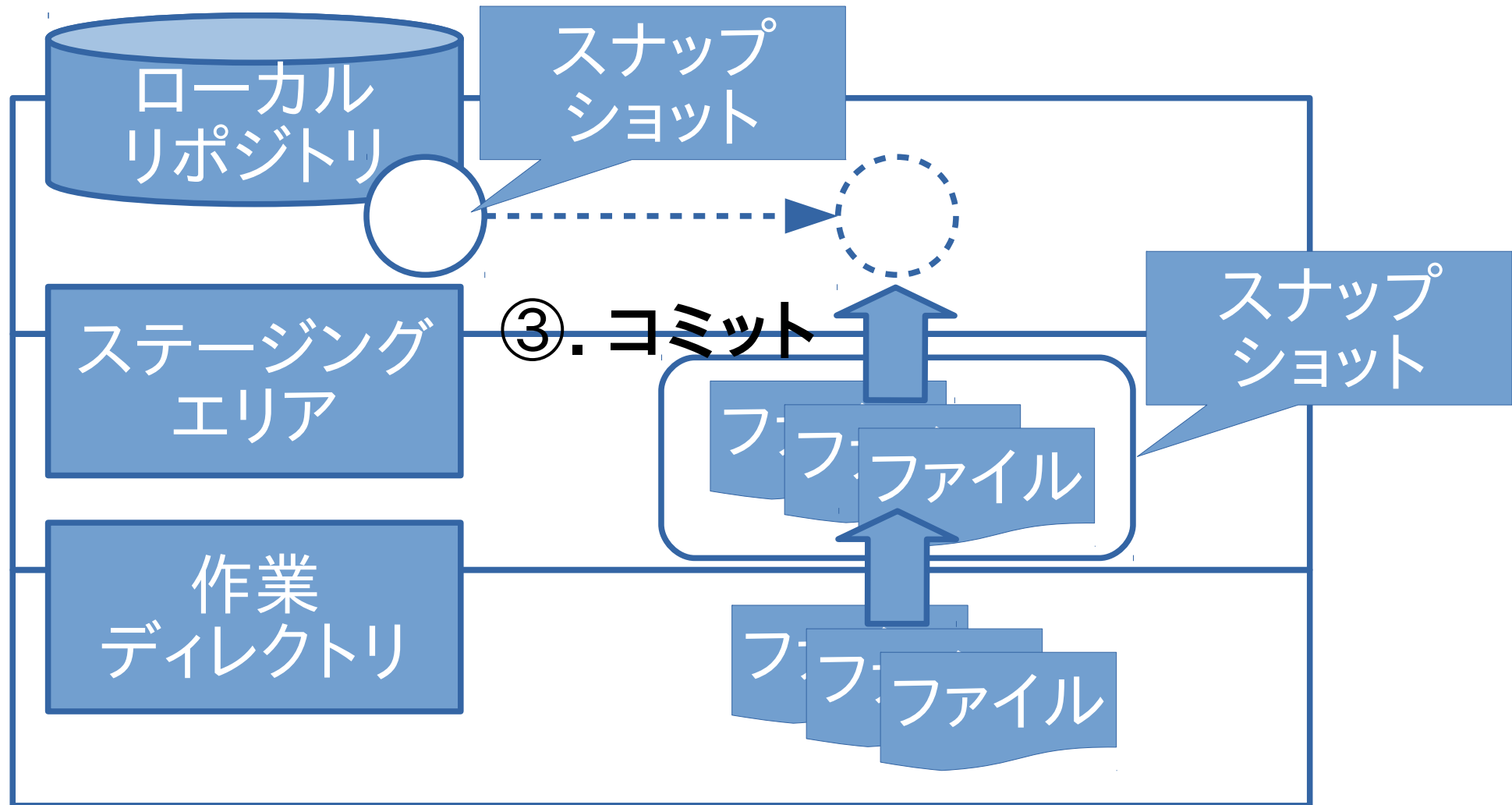
ローカルリポジトリ上でコミット



ローカルリポジトリ上でコミット

- さっきのターミナルで
- `$ git add <新しく追加したいファイル名>`
例) `$ git add test.txt`
- `$ git status`
- `Changes to be committed`(コミットされる予定の変更・ステージされた変更)に新しく追加したファイル名があるはず
- ステージングエリア: 作業ディレクトリとローカルリポジトリとの間のバッファ的存在

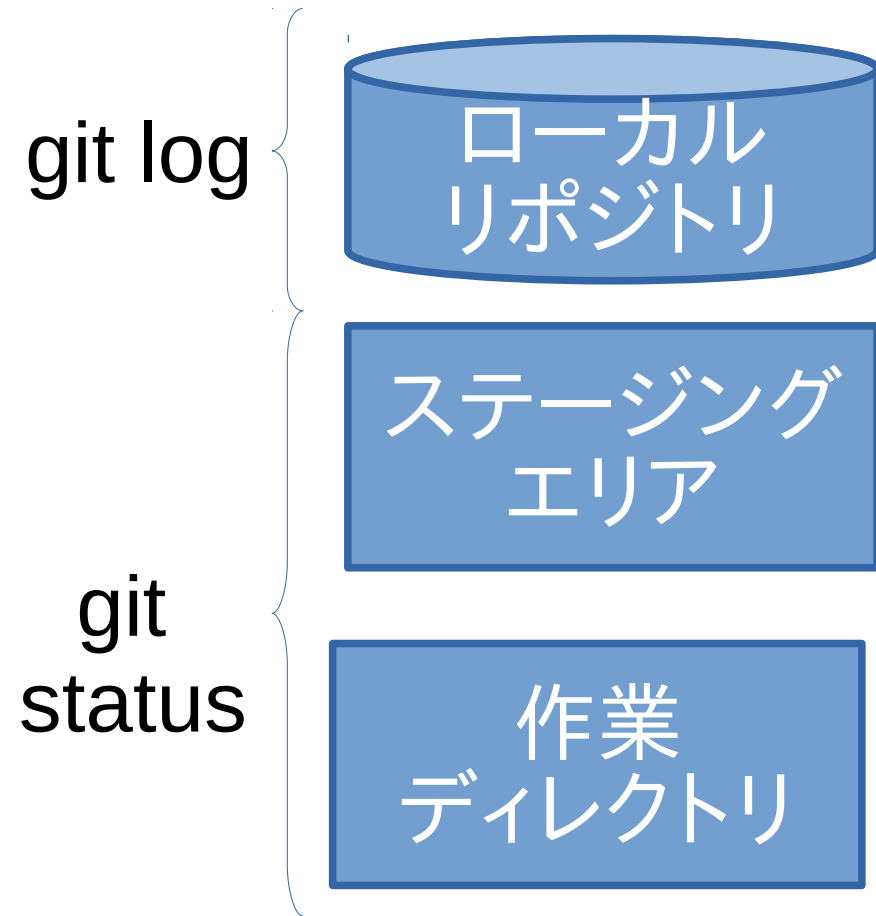
ローカルリポジトリ上でコミット



ローカルリポジトリ上でコミット完了

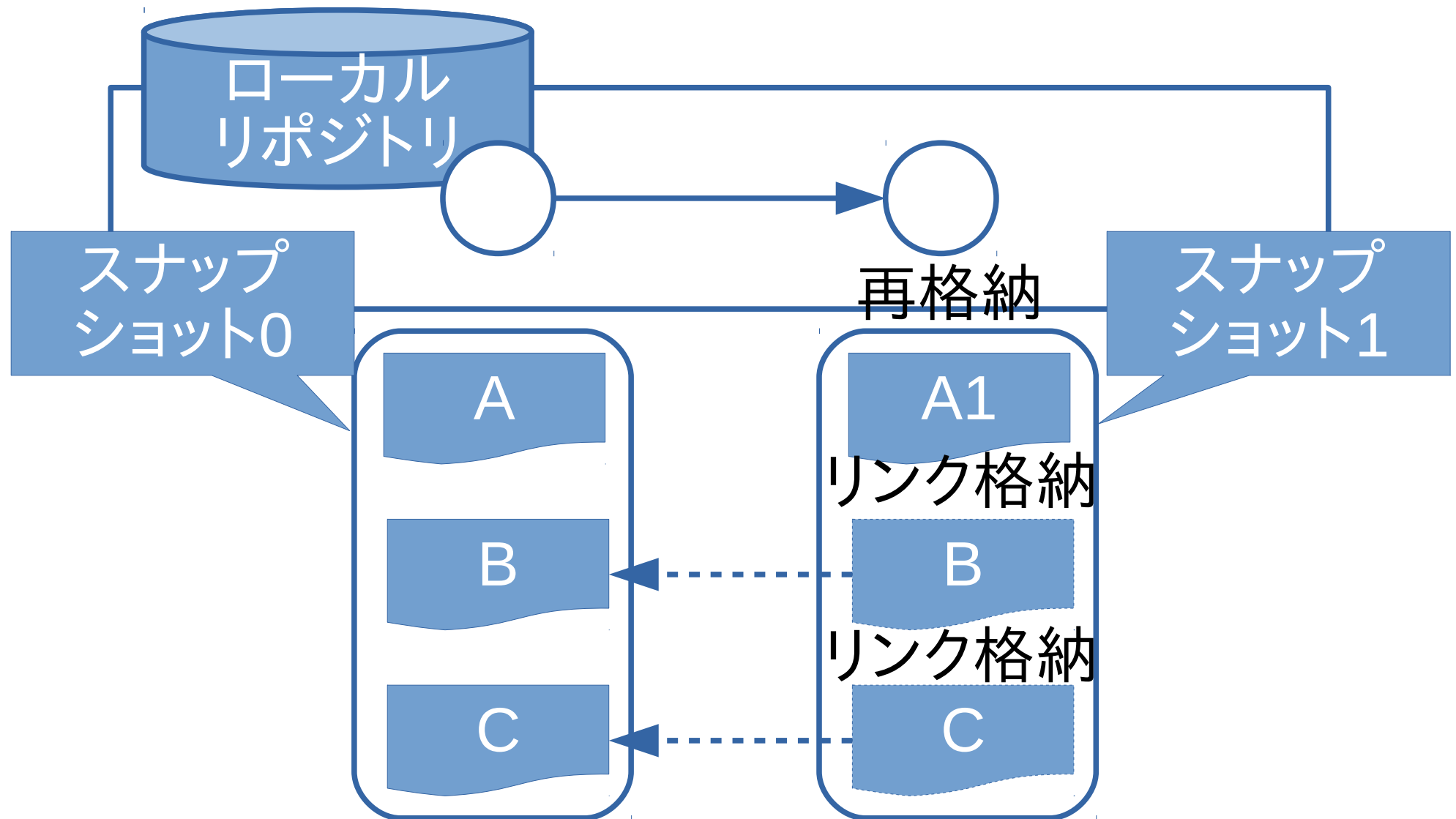
- さっきのターミナルで
- `$ git commit -m "<コミットメッセージ>"`
例)`$ git commit -m "add text"`
- `$ git log`
- ↑ コミット済みの履歴を確認するコマンド
- さっきのコミットが表示されるはず

ローカルリポジトリ上でコミット

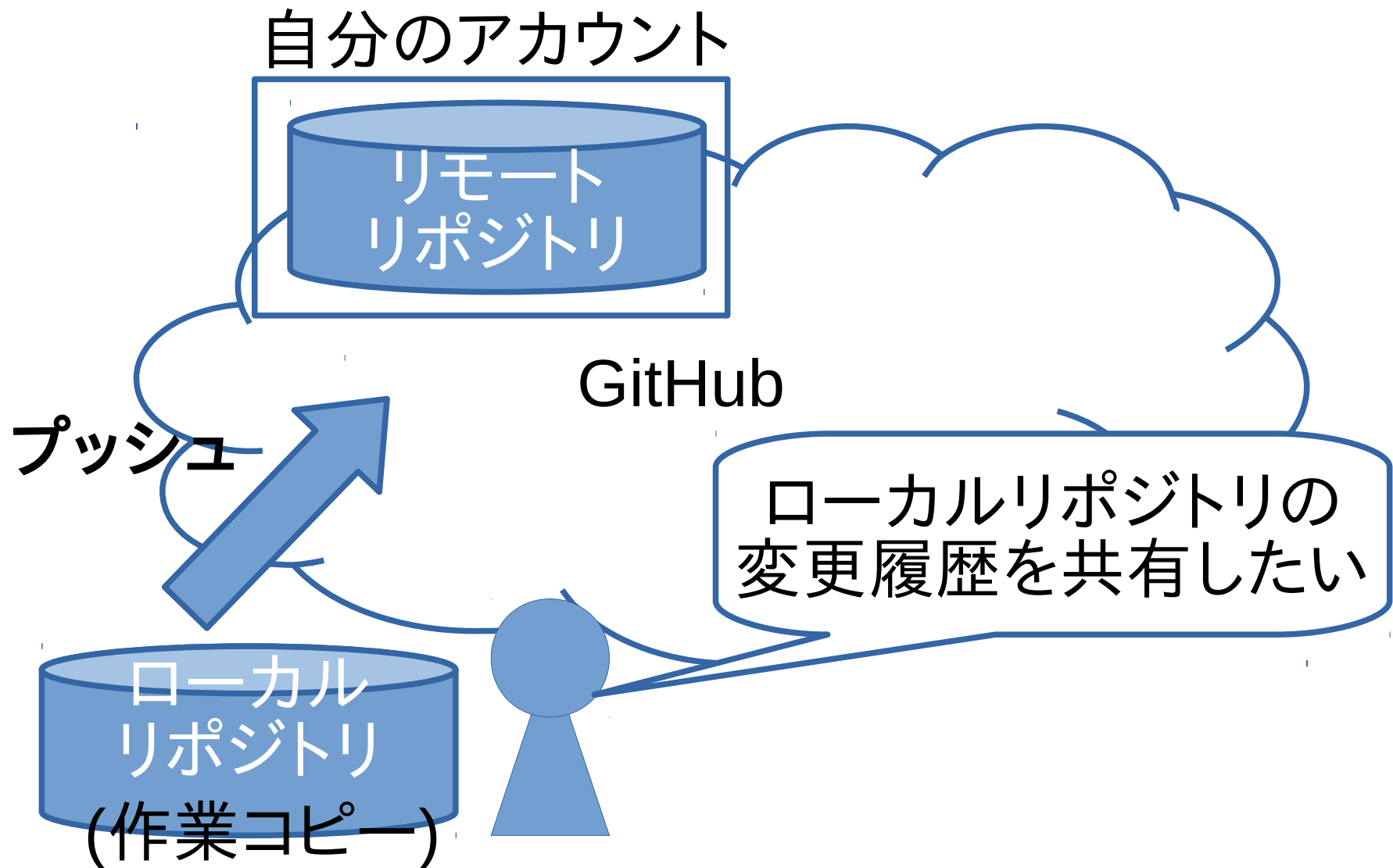


- ファイルには種類がある
 - 修正済み
 - ステージ済み
 - コミット済み
 - (Gitの対象外)

ローカルリポジトリ上でコミット



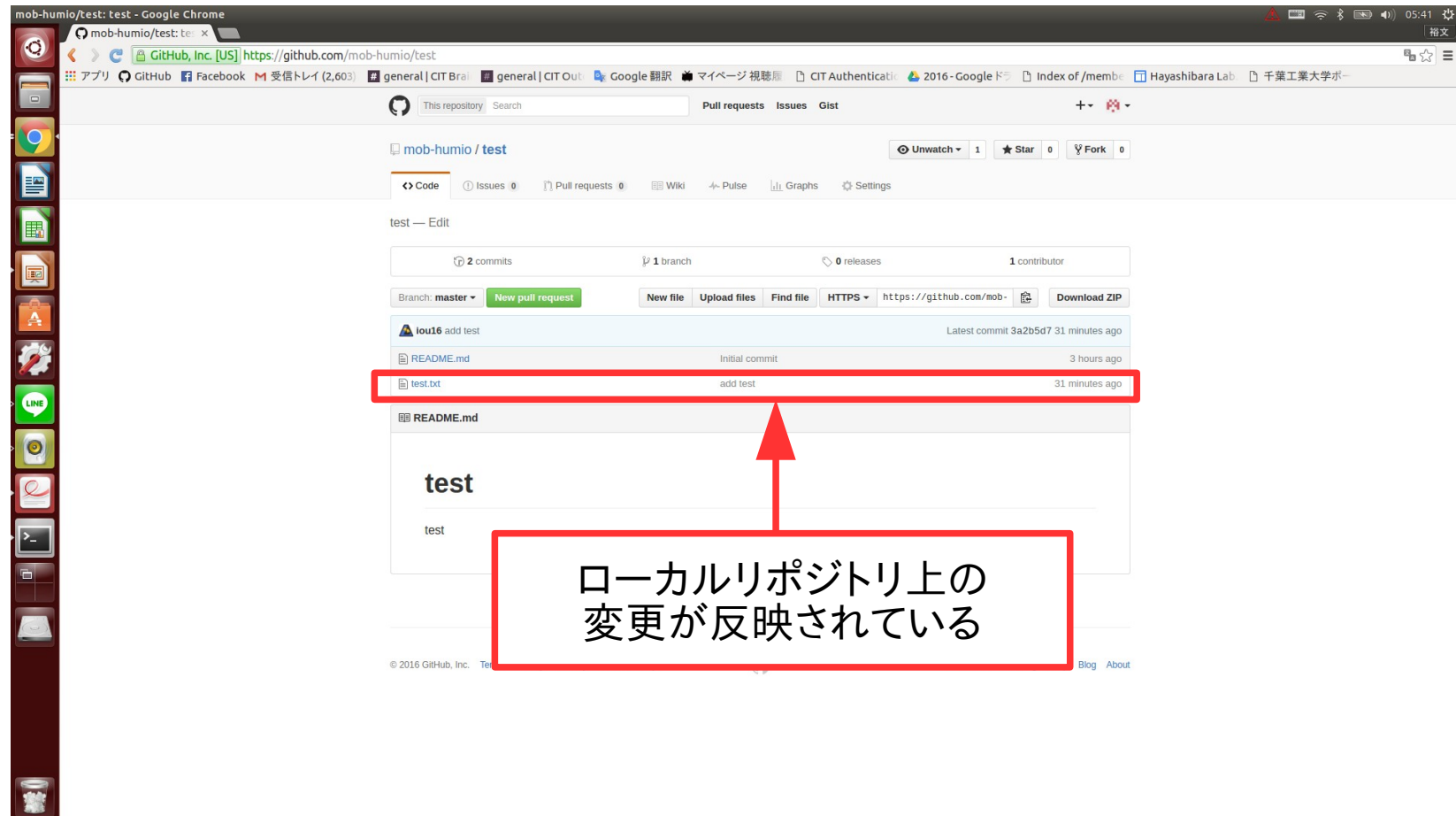
GitHub上のリポジトリにプッシュ



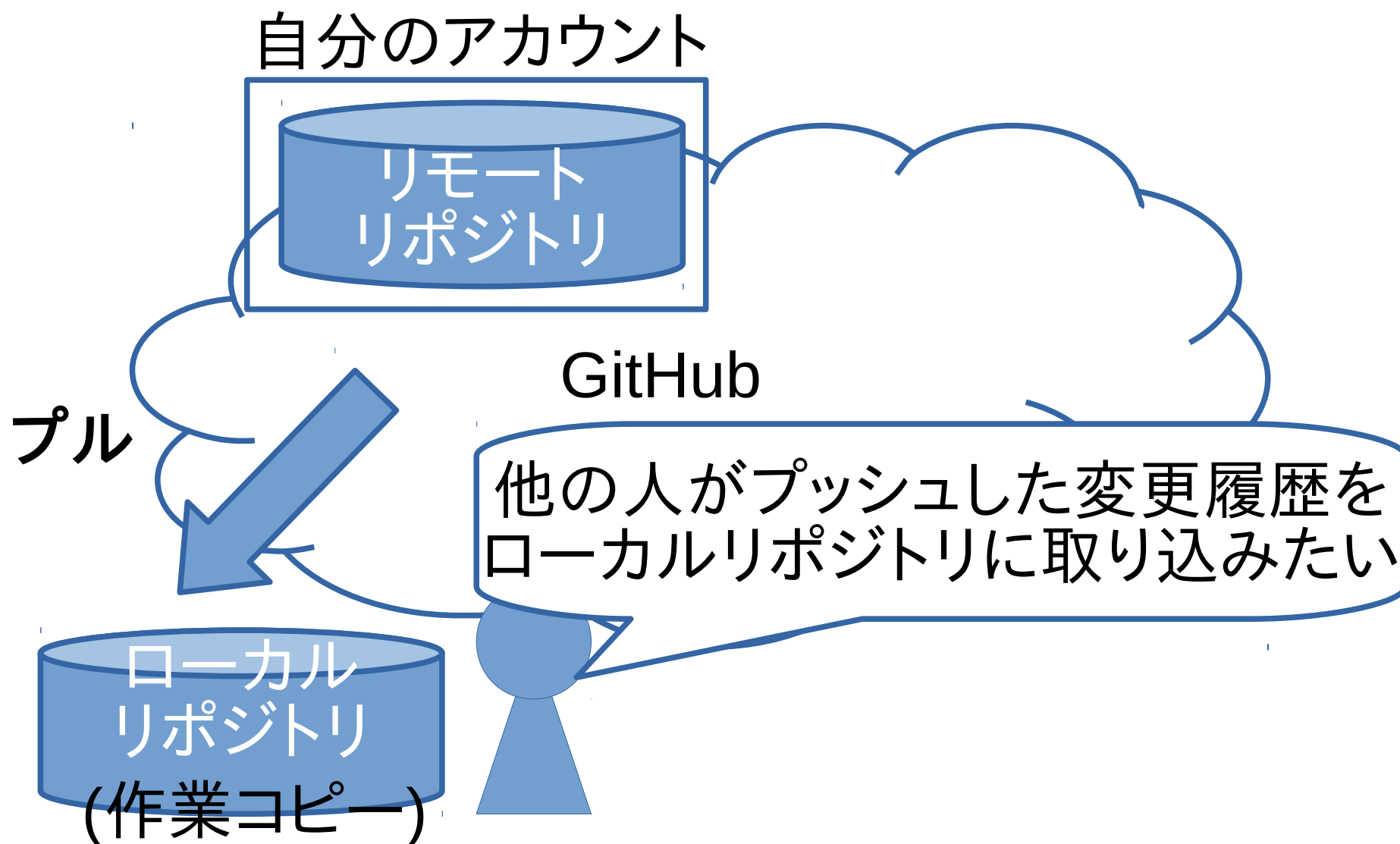
GitHub上のリポジトリにプッシュ

- さっきのターミナルで
- `$ git push origin master`
- ユーザIDとパスワードを確認されるのでそれぞれ入力
- (本番の開発でプッシュする際は, 現在の自分がいるブランチ(後述), プッシュ先のブランチ等をしっかり確認しよう)

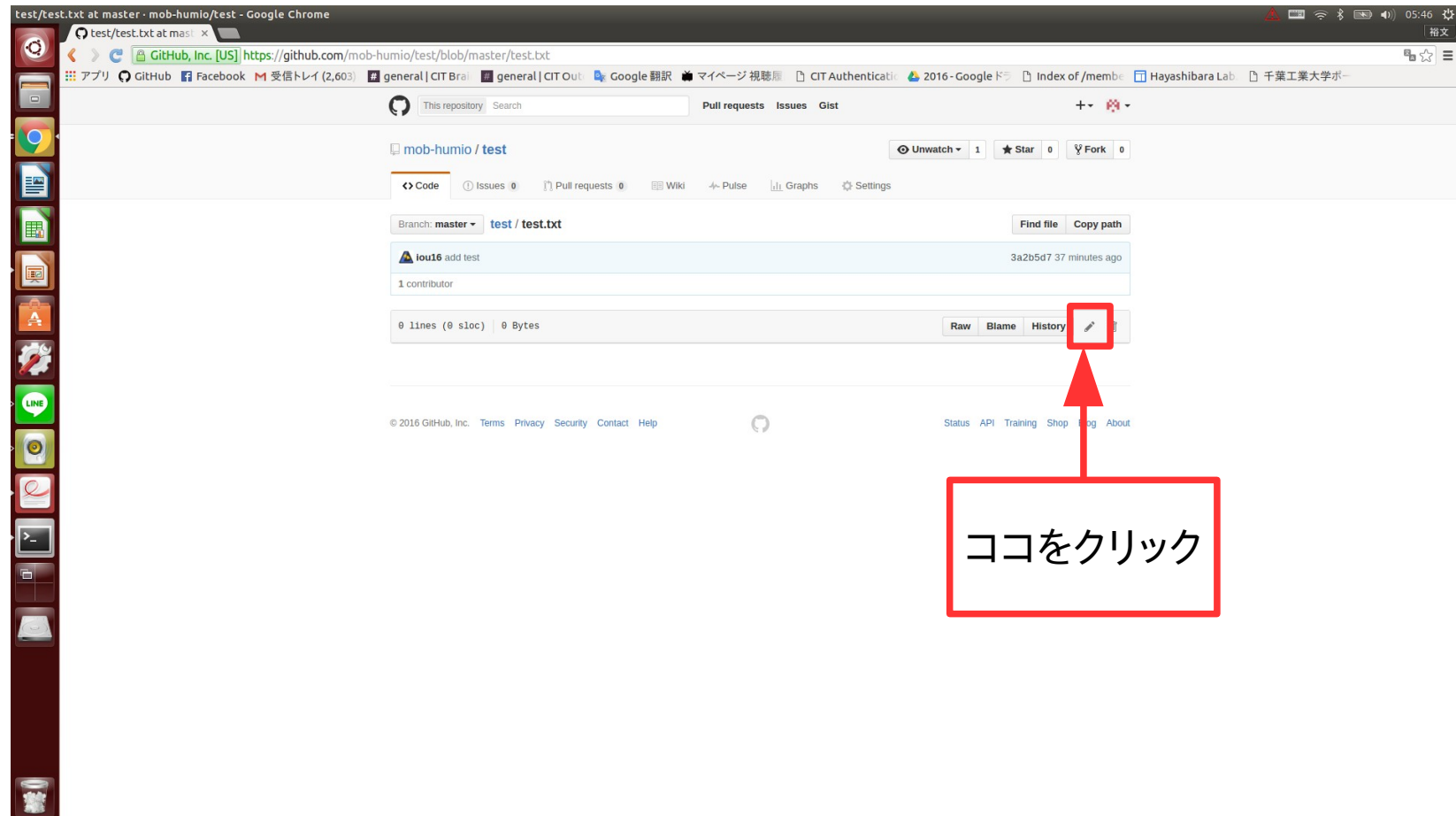
GitHub上のリポジトリにプッシュ完了



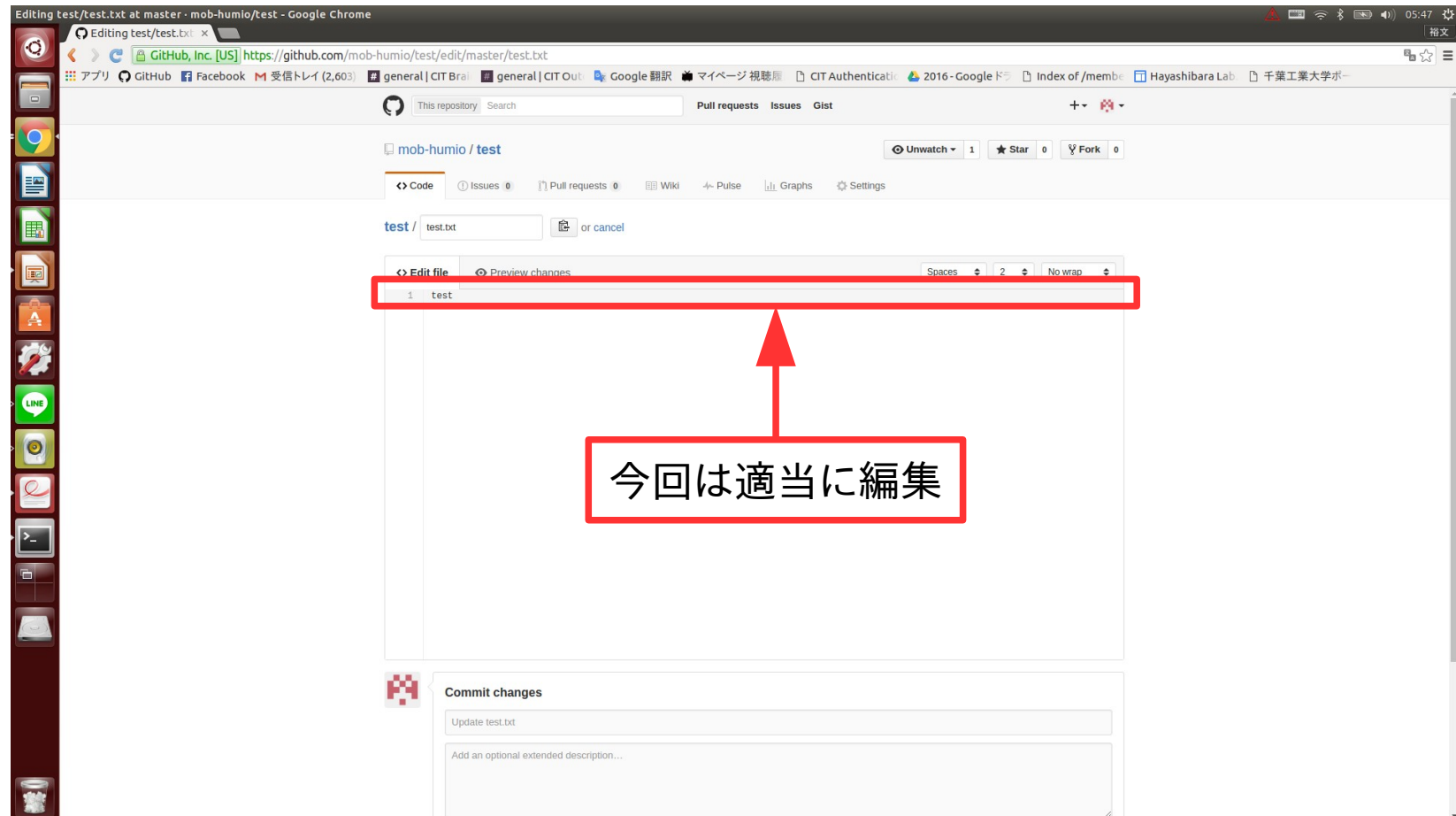
リモートリポジトリからプル



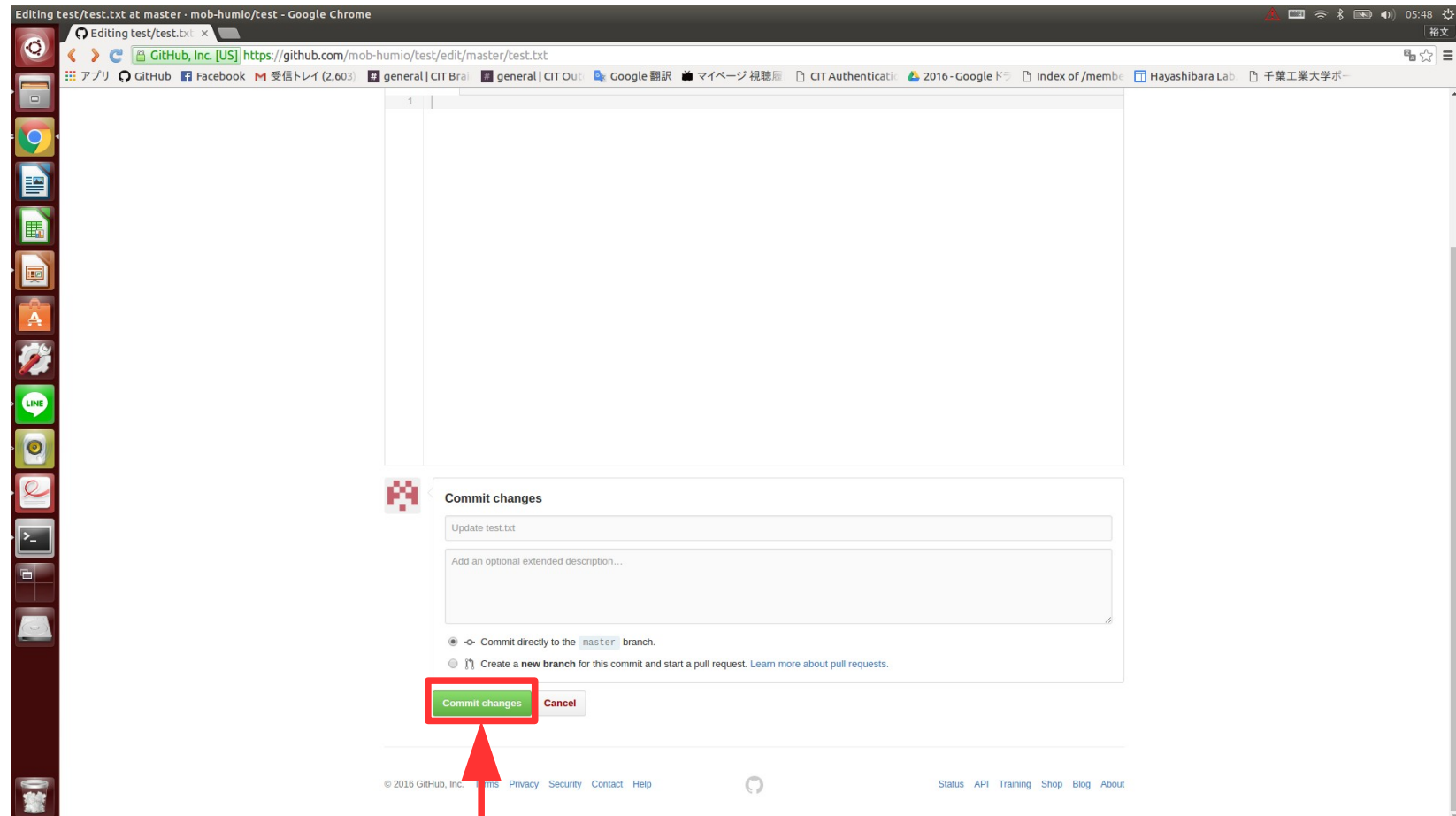
リモートリポジトリからプル



リモートリポジトリからプル

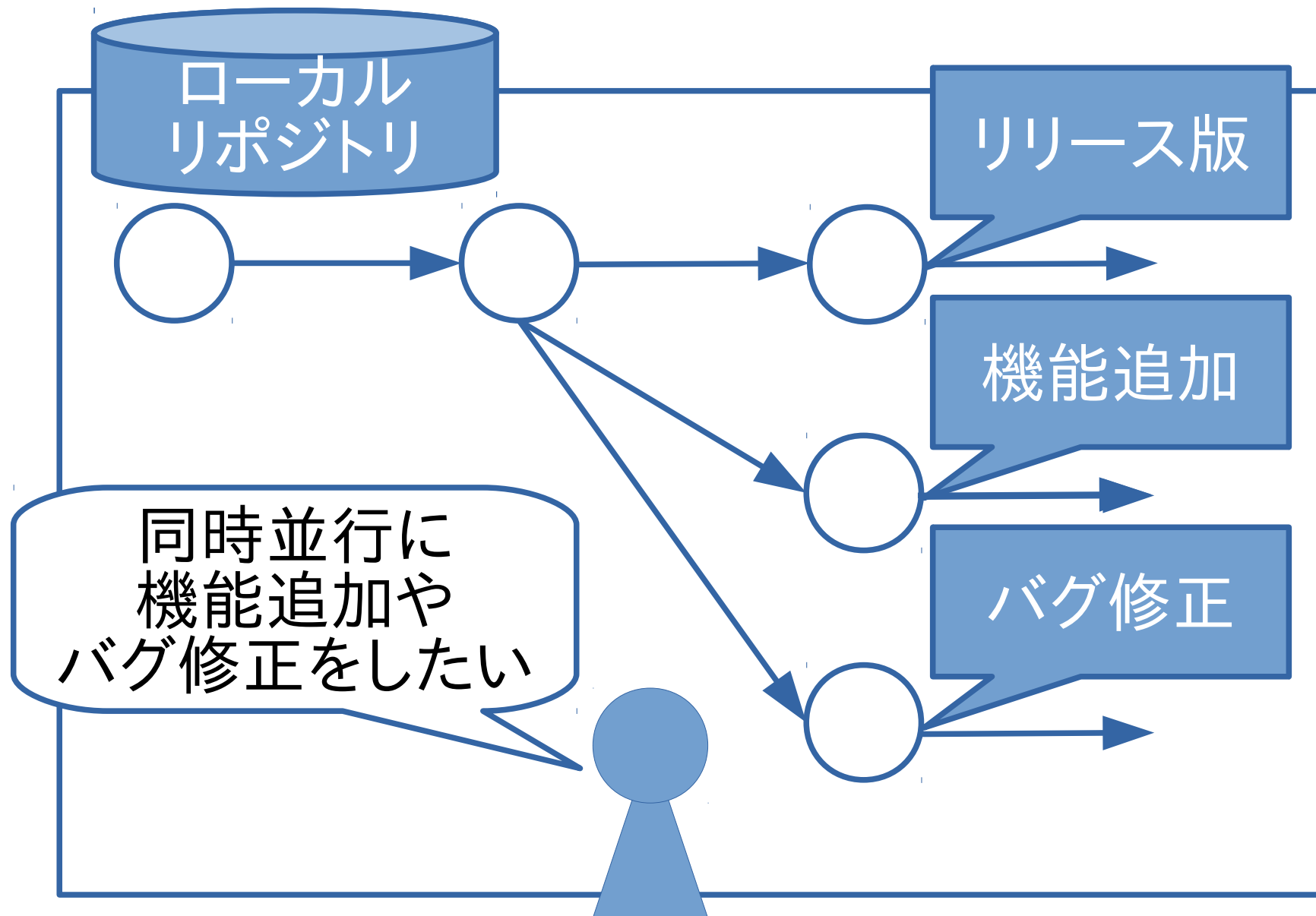


リモートリポジトリからプル



ココをクリック
してコミット

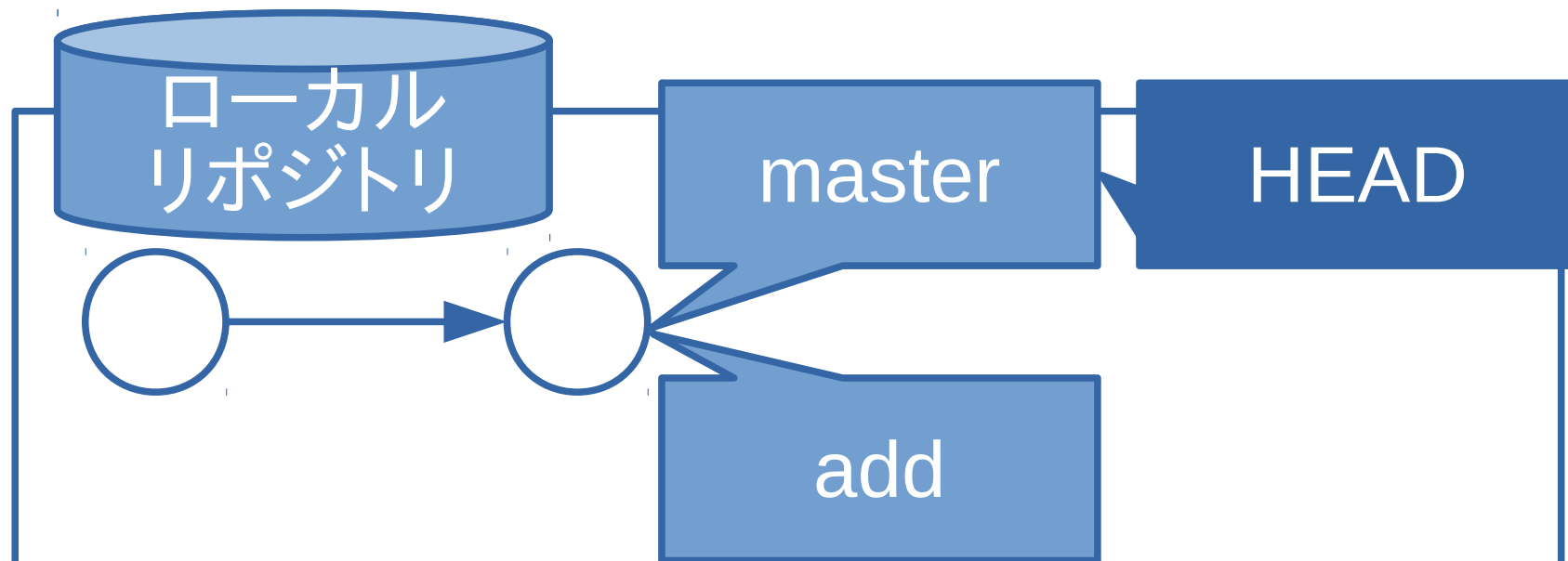
ローカルリポジトリ上でブランチを切る



ローカルリポジトリ上でブランチ を切れた

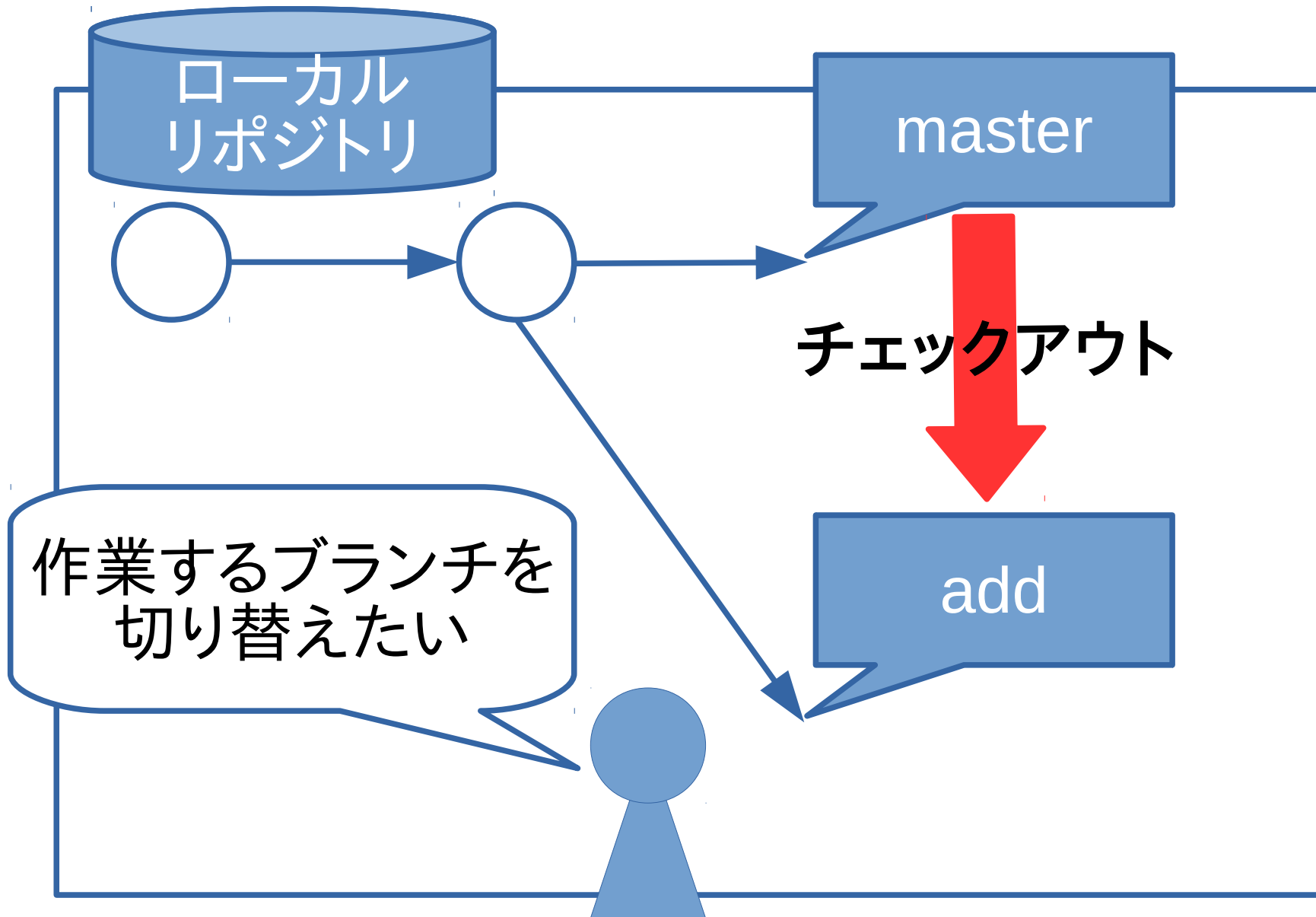
- さっきのターミナルで
- `$ git branch <新しく作成するブランチ名>`
例)`$ git branch add`
- `$git branch`
- ↑リポジトリ内のすべてのブランチを一覧表示(現在自分がいるブランチも表示)
- masterと新しく作成したブランチが表示され
masterに*がついているはず

ローカルリポジトリ上でブランチを切る



- ブランチ: コミットの先頭を指すポインタ
- コミットを繰り返すたびに, ポインタも自動的に進む
- HEAD自分が作業しているローカルブランチへのポインタ

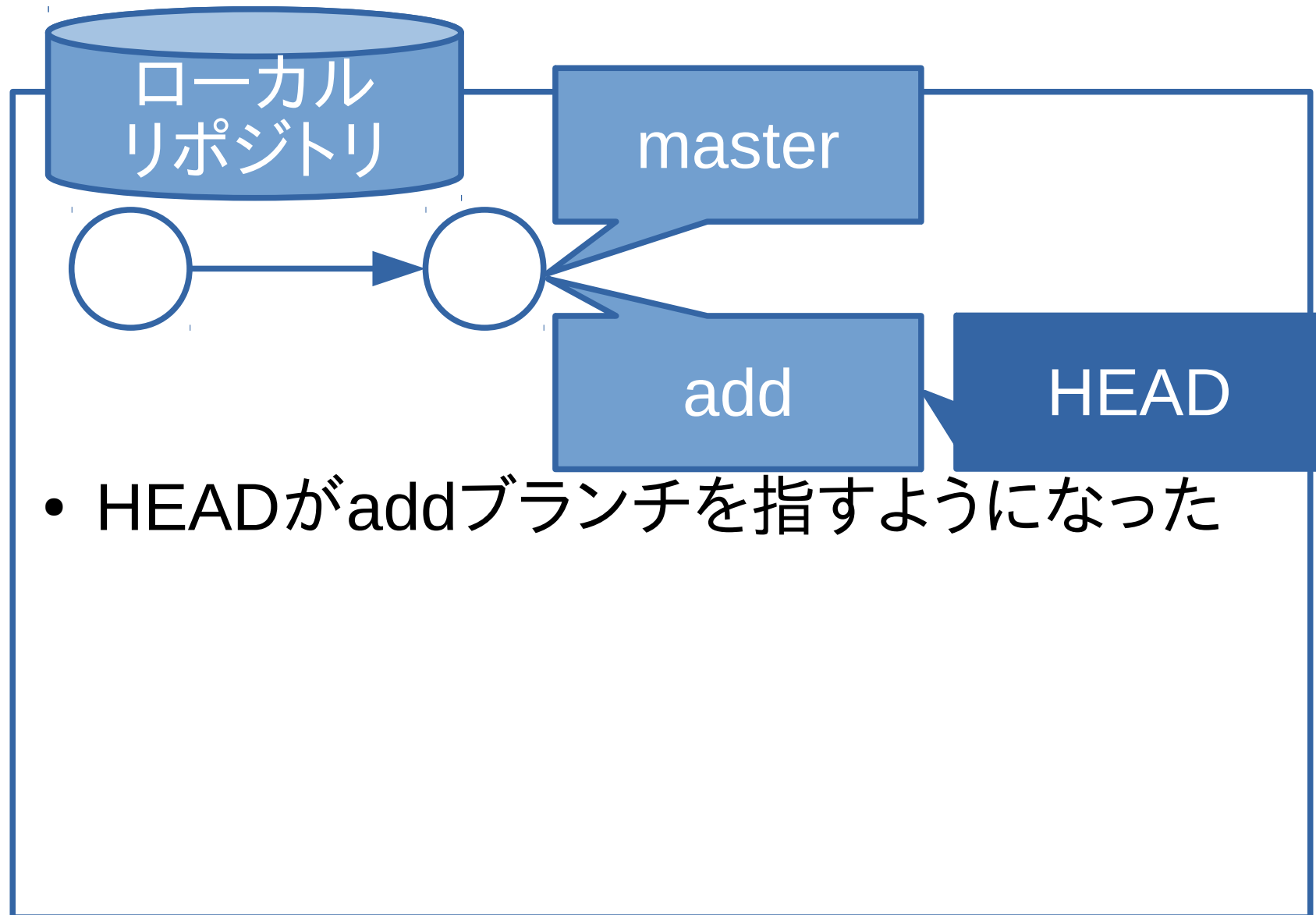
チェックアウトしてブランチ を切り替える



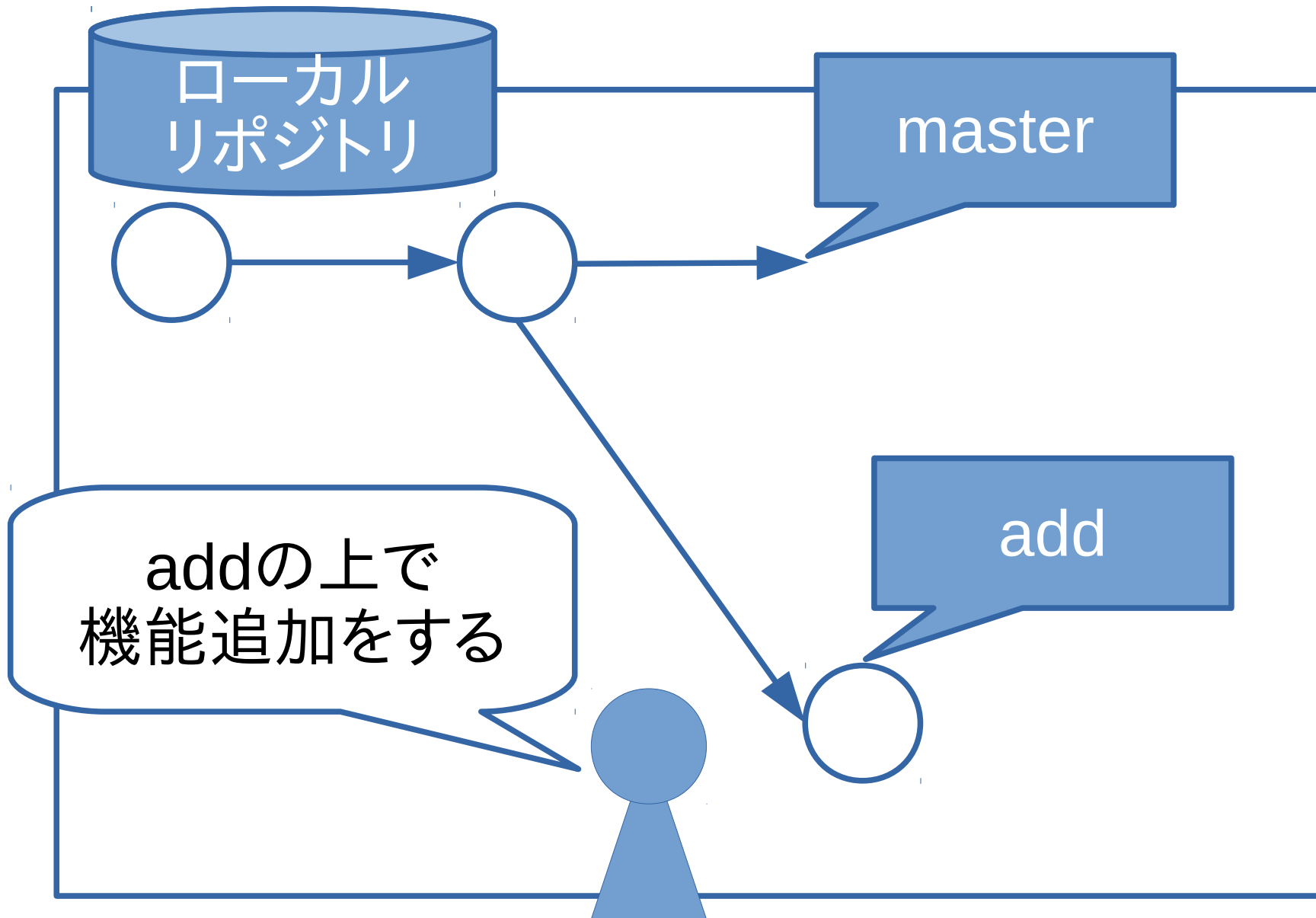
ローカルリポジトリ上でブランチ を切り替えた

- さっきのターミナルで
- `$ git checkout <新しく作成したブランチ名>`
例)`$ git checkout add`
- Switched to branch '<新しく作成したブランチ名>'のように表示されるはず
- `git branch`で確認し, ちゃんと<新しく作成したブランチ>に*がついているか確認

ローカルリポジトリ上でブランチを切る



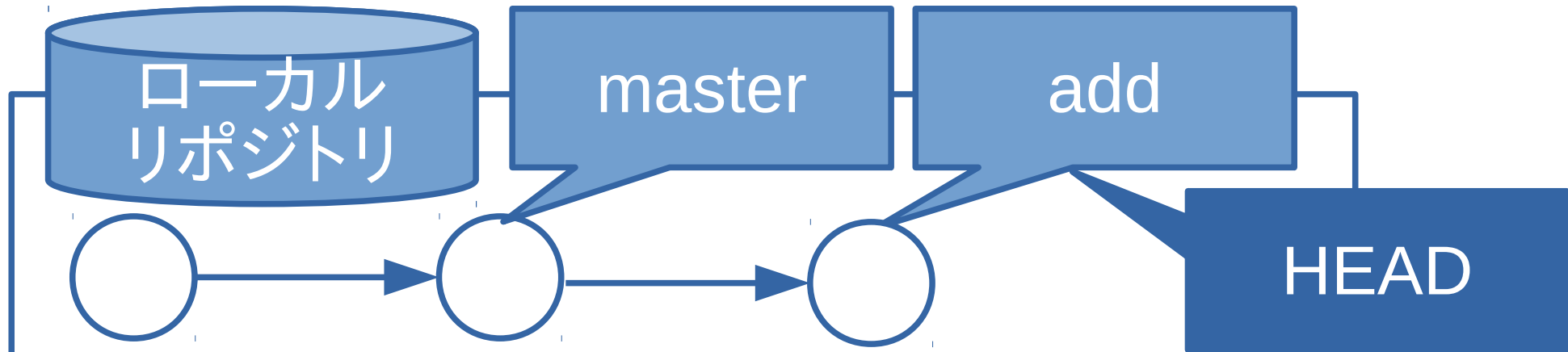
ブランチ上での開発



ブランチ上での開発

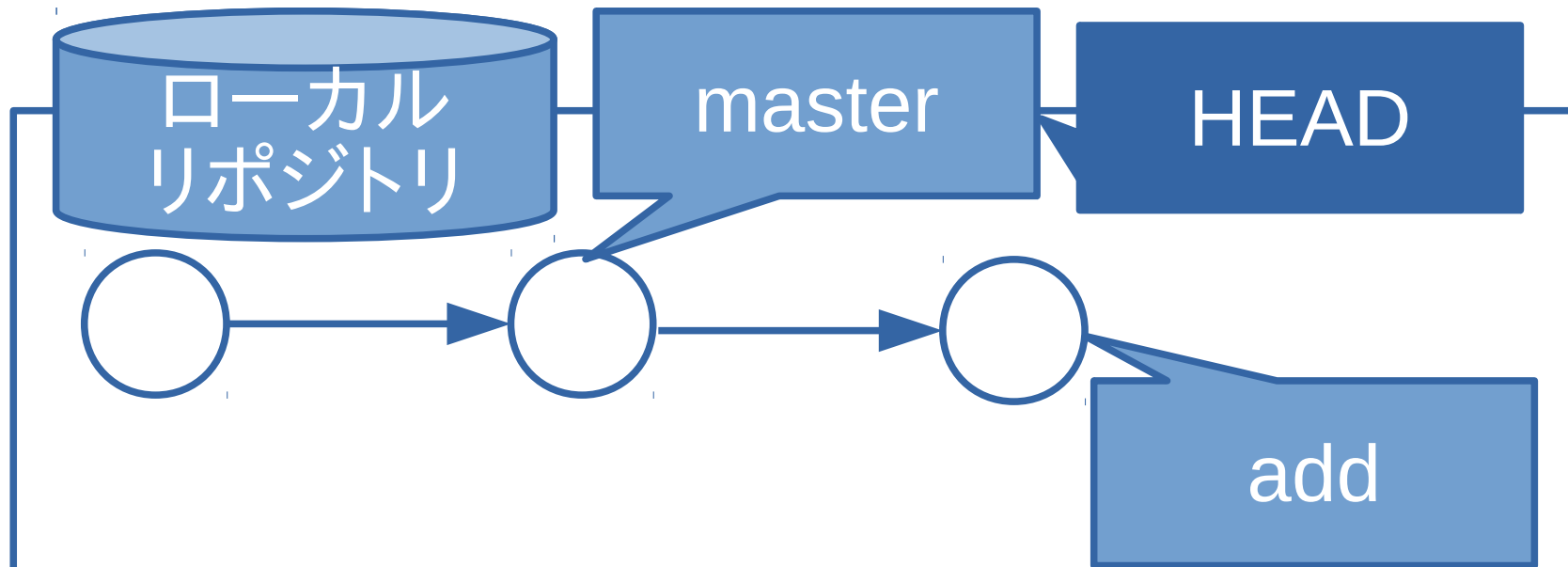
- さっきのターミナルで
- 以前追加したファイルを編集する
例)\$ vim test.txt
- git statusでChanges not staged for commitに以前追加したファイルがあるか確認
- git addで編集したファイルをステージし, git commit -m “<コミットメッセージ>”でコミット
- git logで確認

ブランチ上での開発



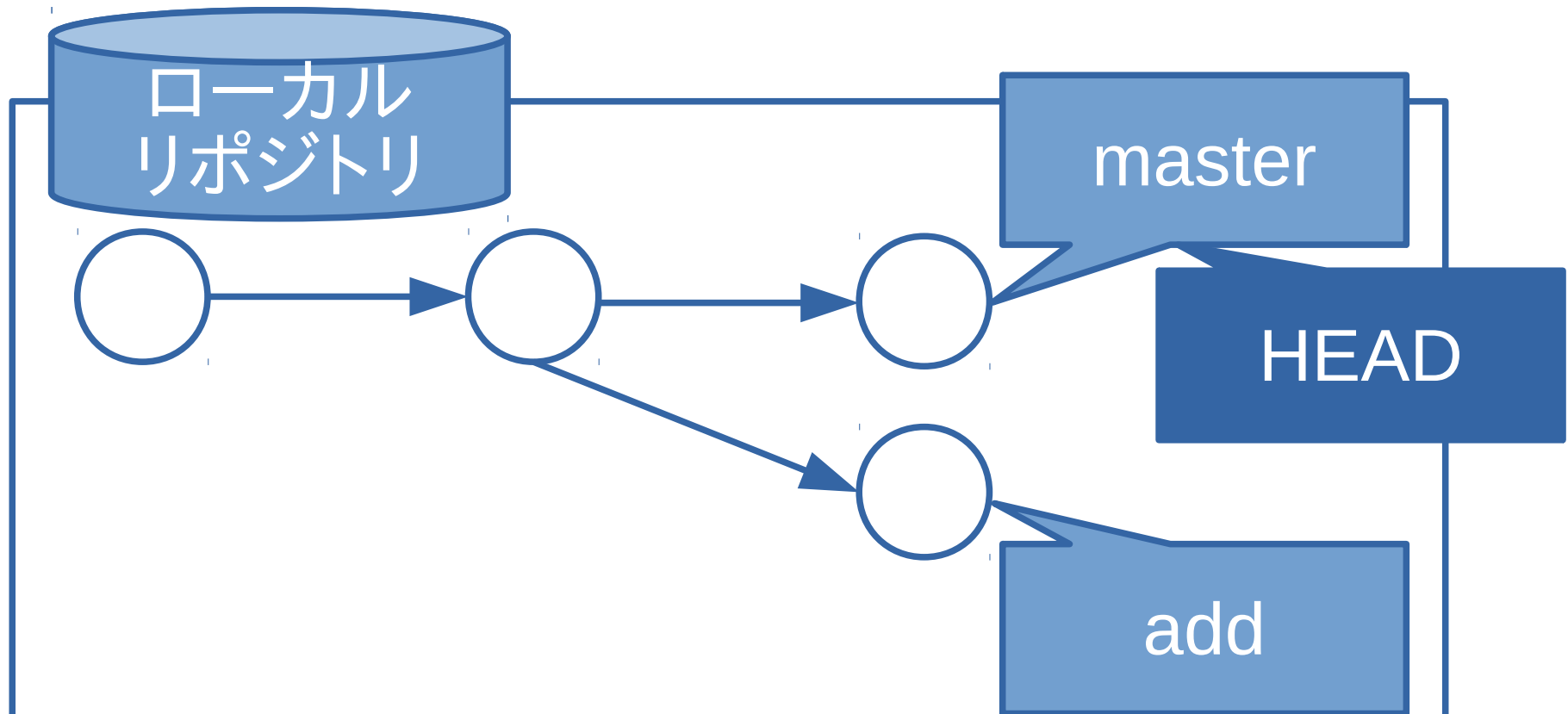
- HEADが指すブランチがコミットによって移動
- masterブランチは移動しない

ブランチ上での開発



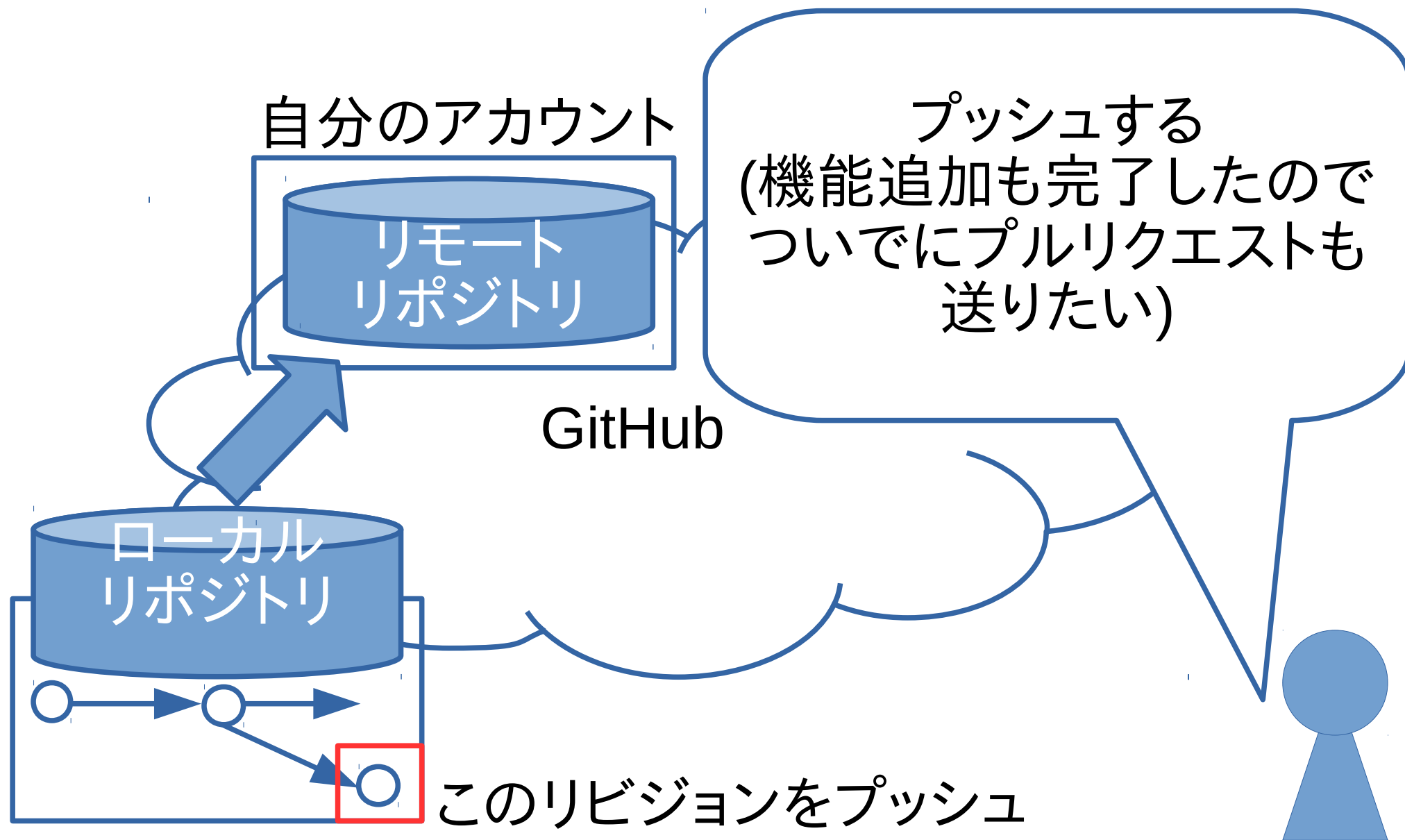
- ここでmasterブランチにチェックアウトするとHEADポインタが指す先がmasterブランチに戻り, 作業ディレクトリ内のファイルがmasterが指すスナップショットの状態に戻る

ブランチ上での開発



- プロジェクトの歴史が2つに分岐

GitHub上のリポジトリにプッシュ



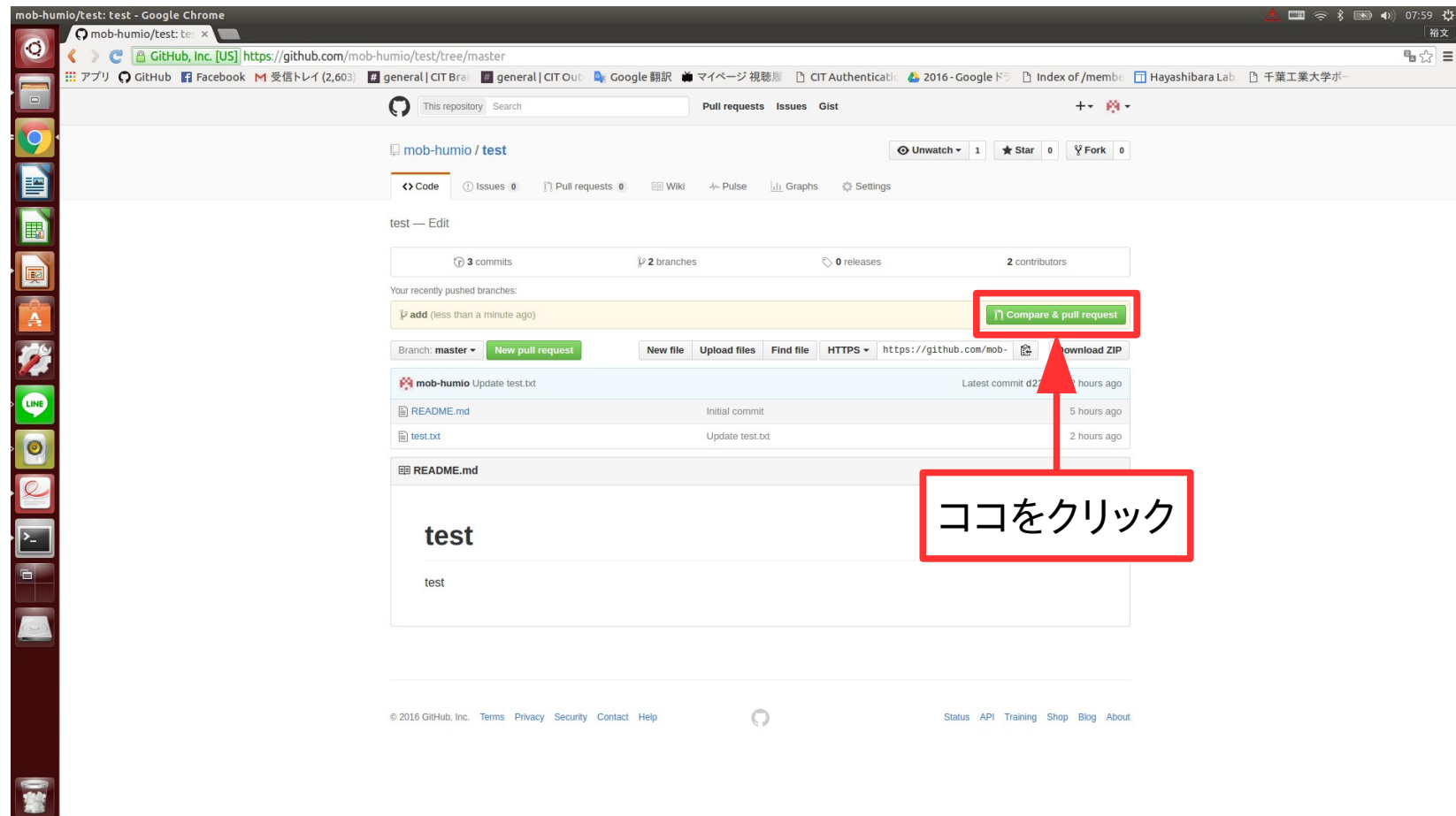
ブランチ上での開発

- さっきのターミナルで
- `$ git push origin add`

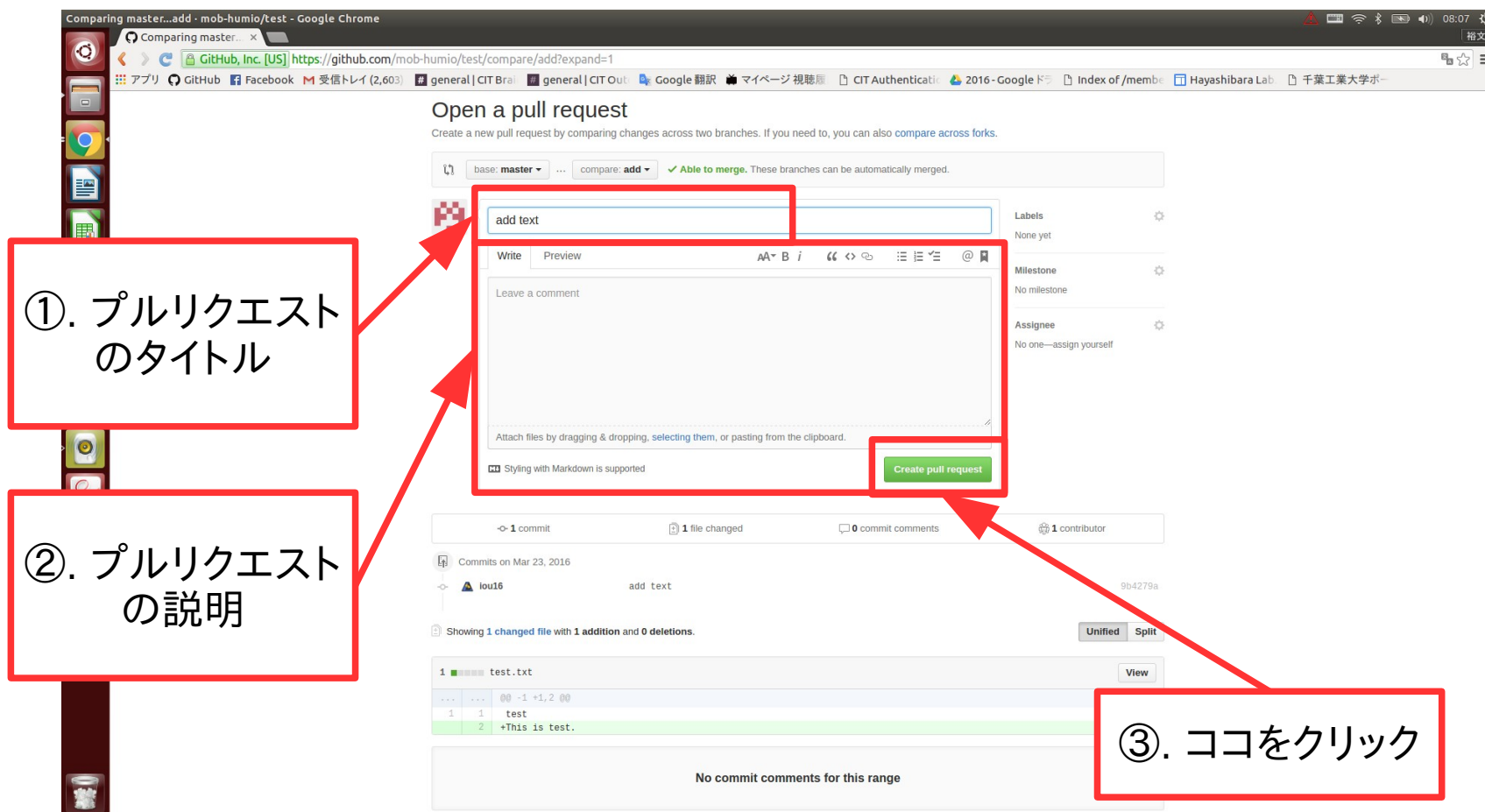
プルリクエスト

- 機能追加が完了した時
 - 分岐元のブランチに対して自分の変更を取り込んでもらう時
 - マージ
- アドバイス等が欲しい時
 - 必ずしもマージする訳ではない

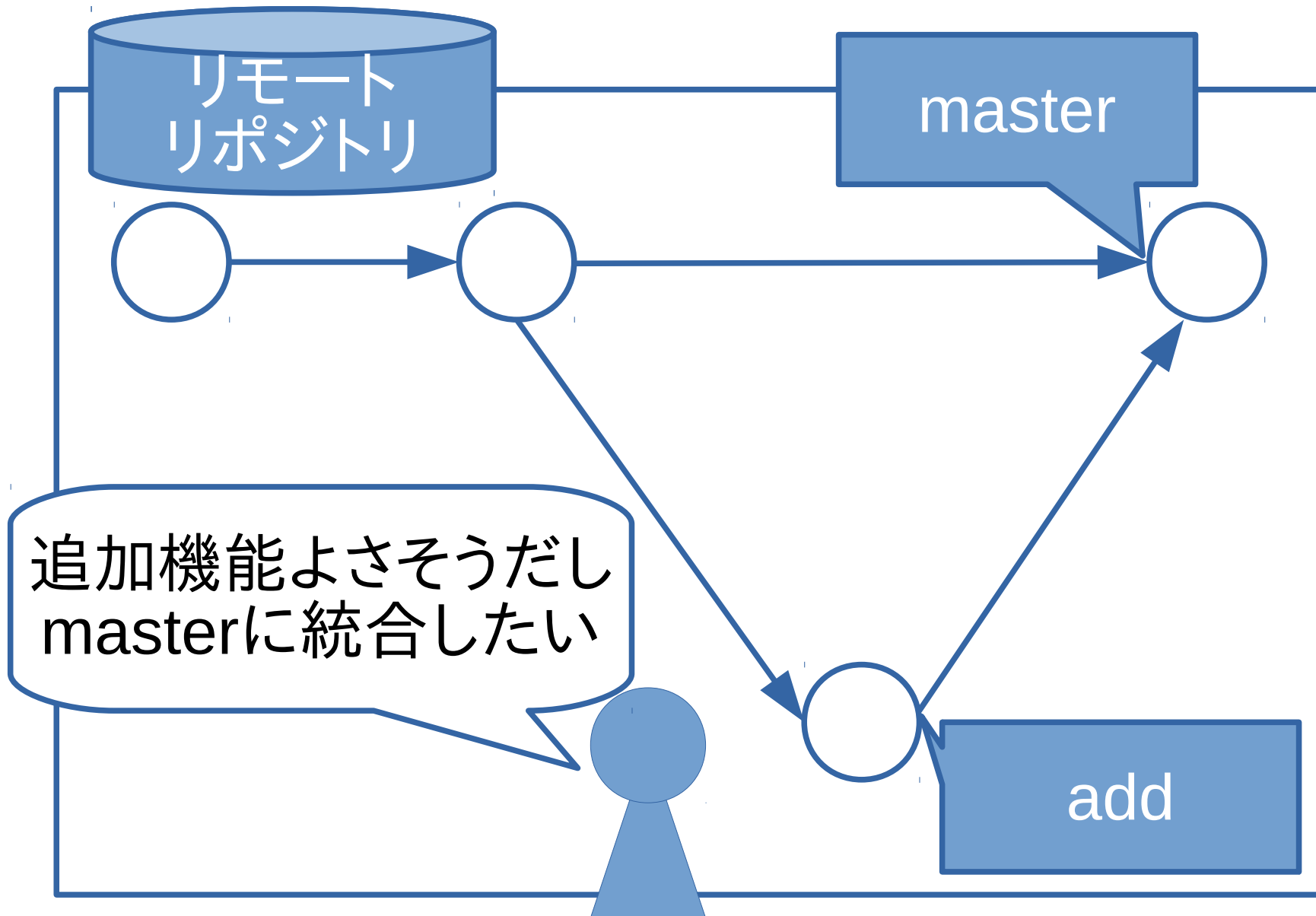
プルリクエストを送信



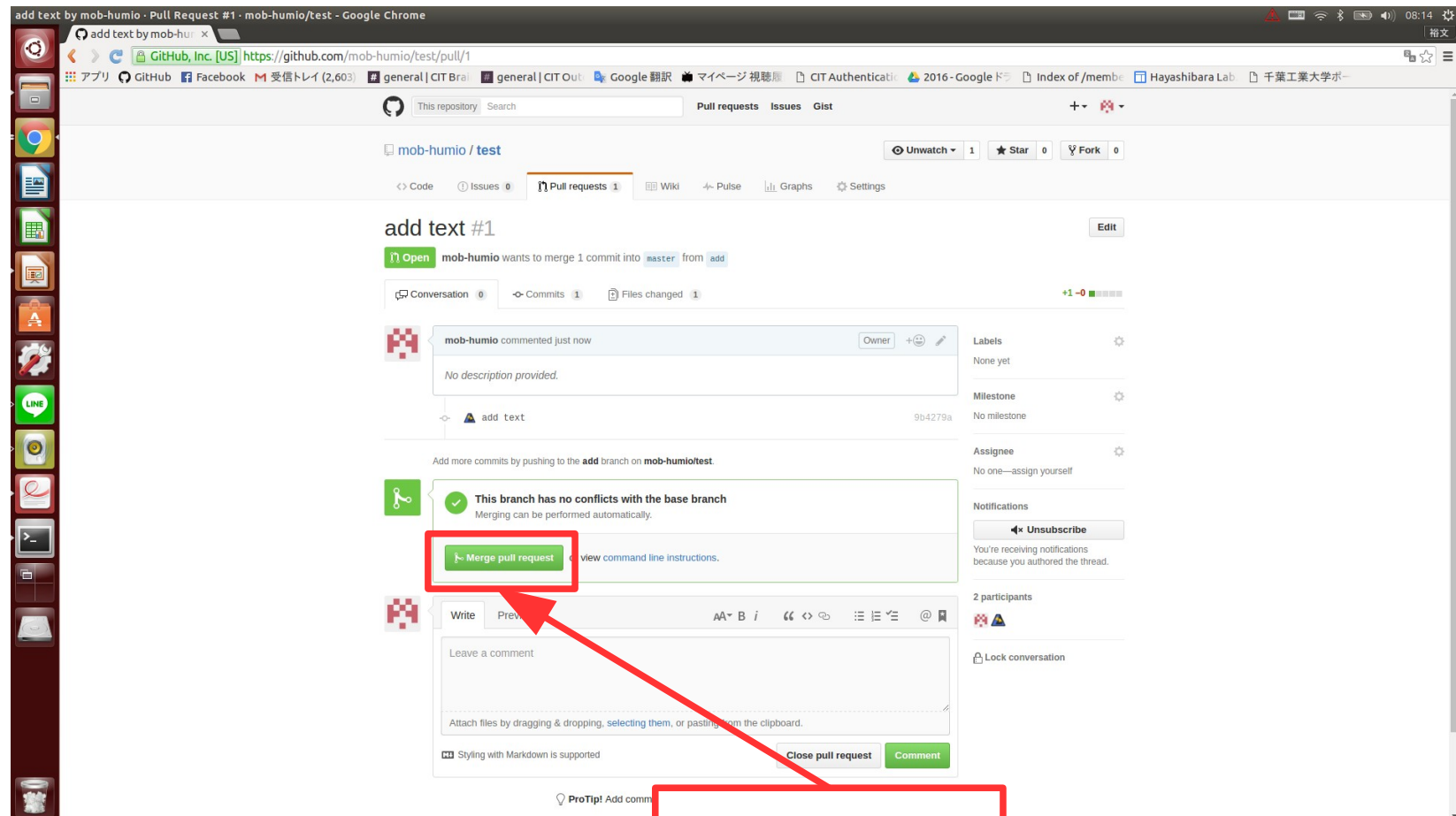
プルリクエストを送信



プロリクエストをもとにマージ

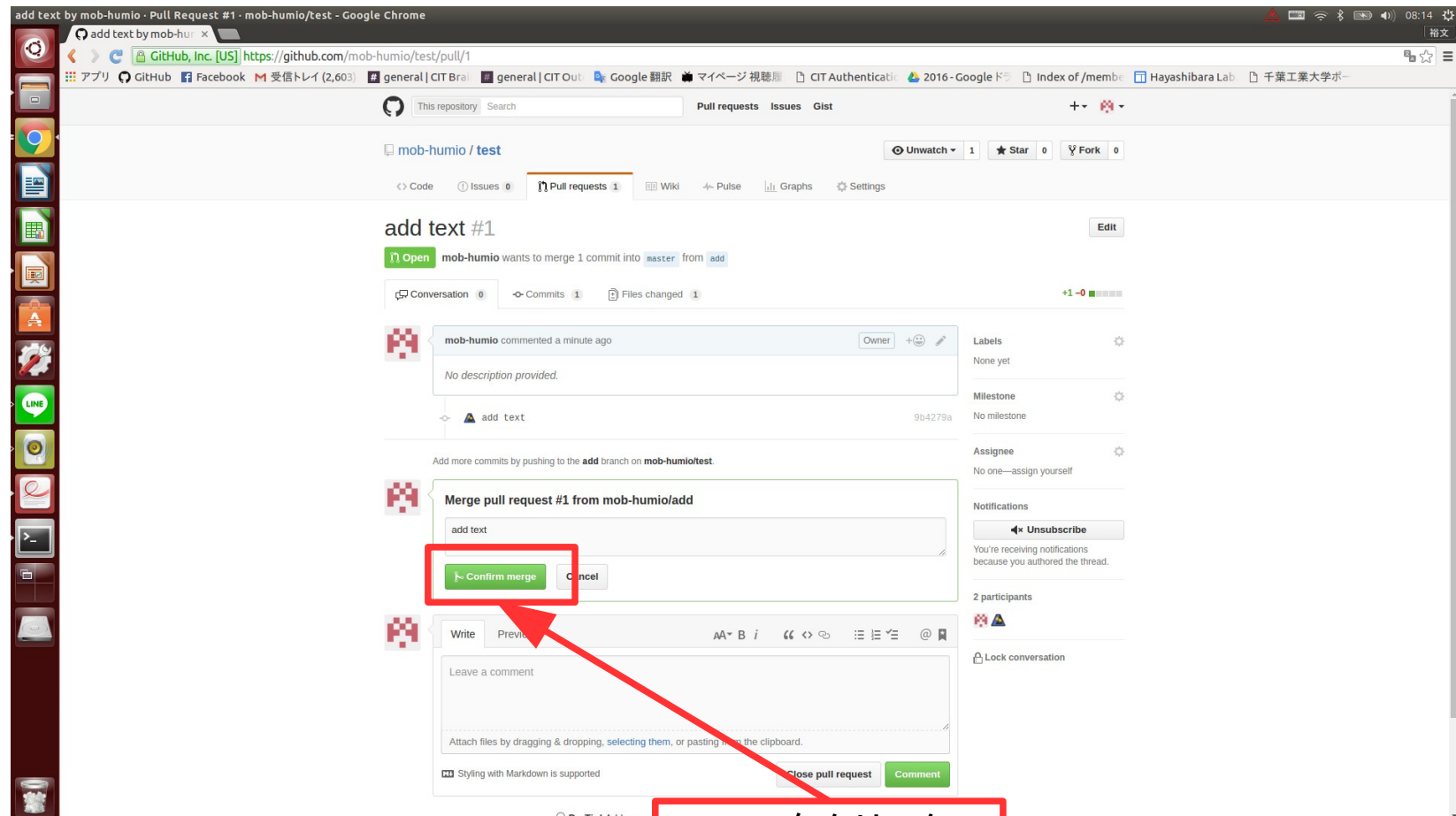


リモートリポジトリ上でのマージ



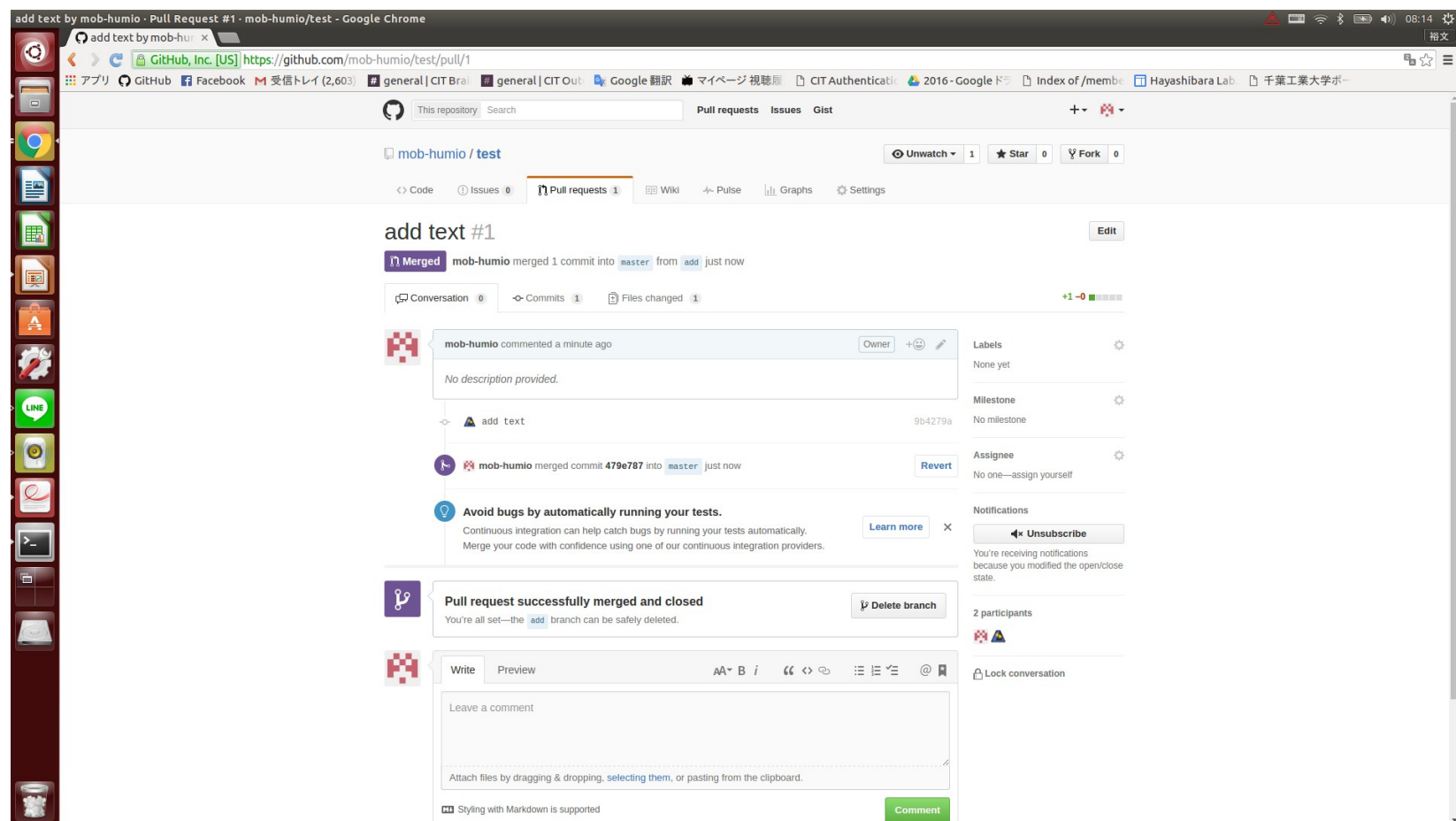
ココをクリック

リモートリポジトリ上でのマージ

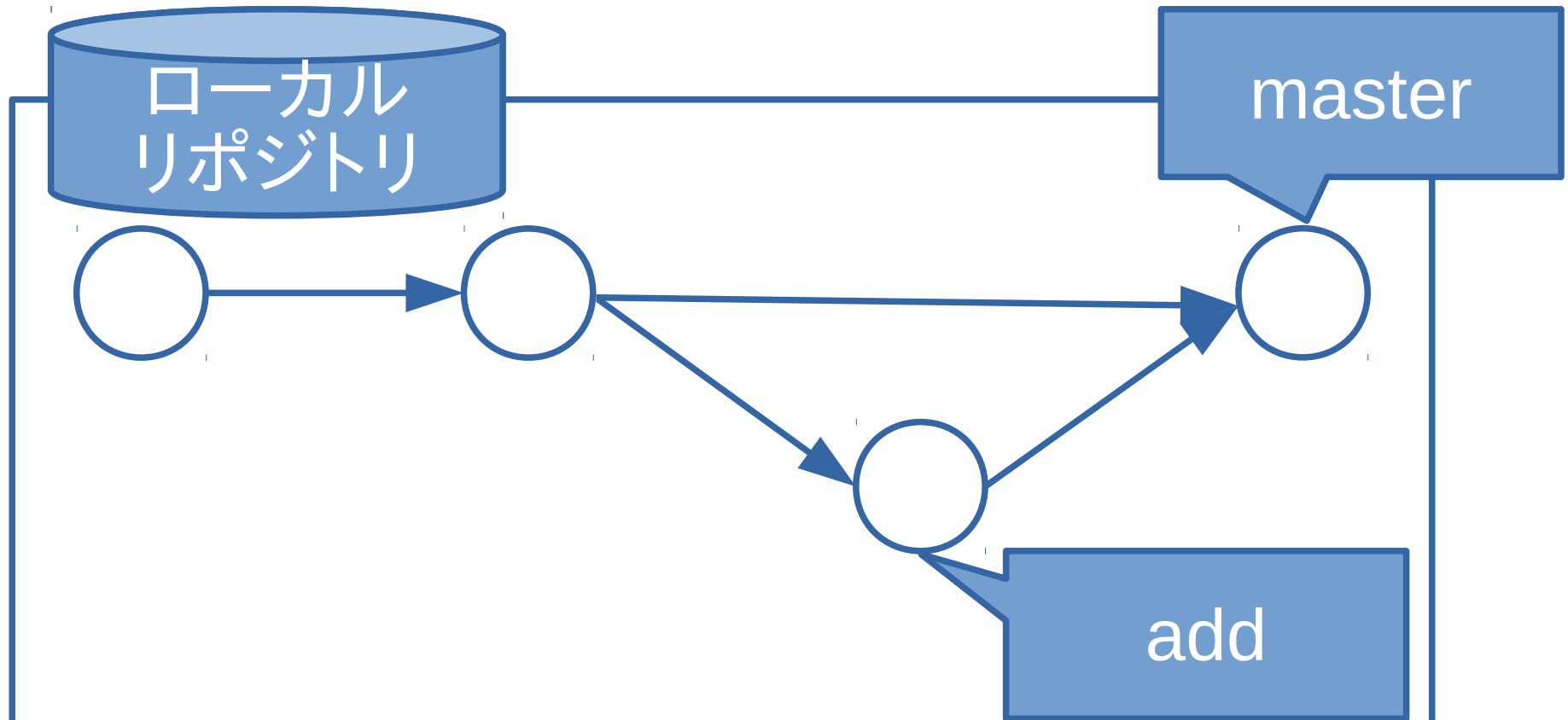


ココをクリック
(最終確認)

マージ完了の画面



プルリクエストをもとにマージ

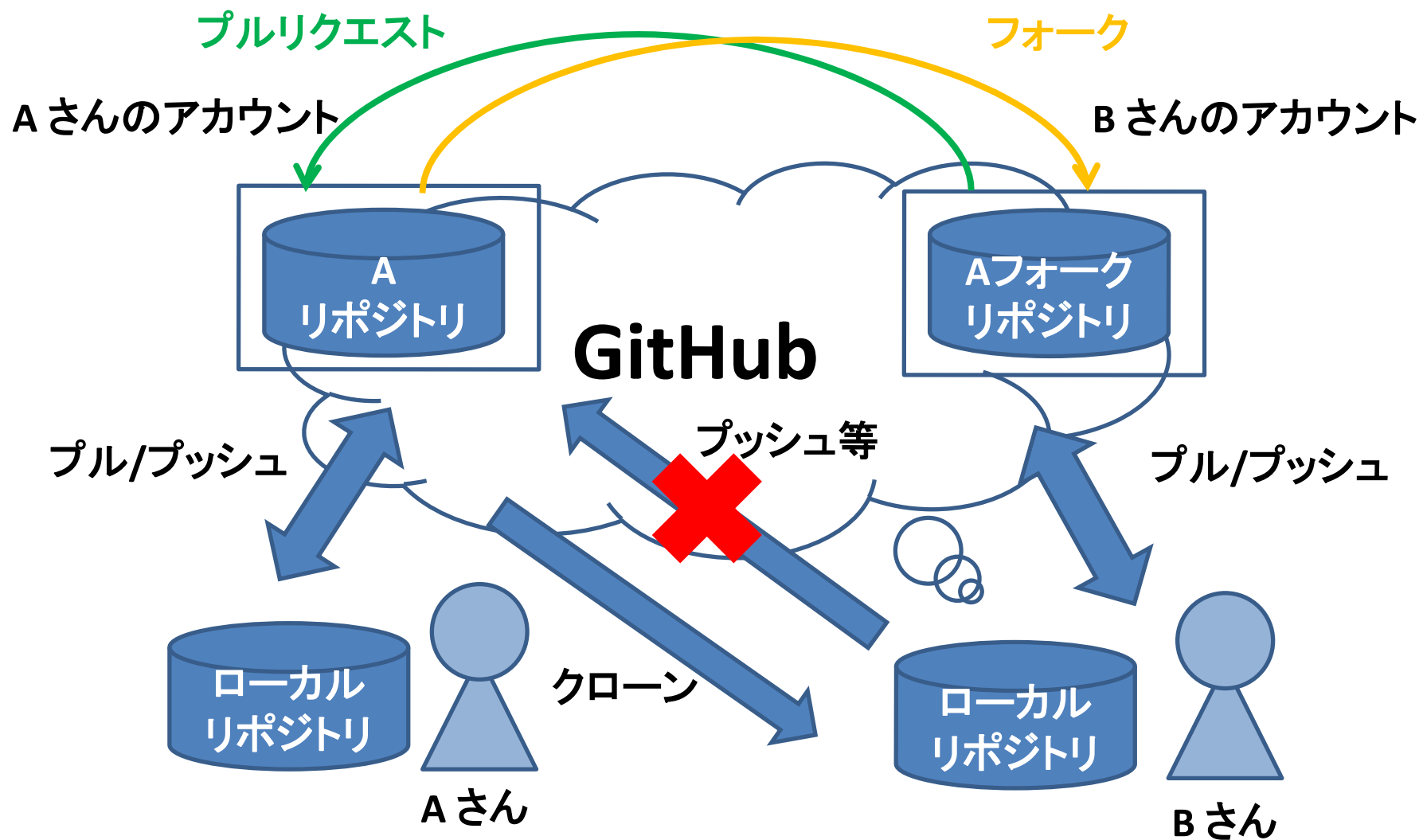


- 新たなスナップショットが作成され, それを指すコミットが自動生成
- masterブランチのポインタも移動

マージされた結果のプル

- さっきのターミナルで
- `$ git checkout master`
- `$ git pull origin master`
- `git log`で確認
- Merge pull request #1 from ユーザー名/ブランチ名のようなコミットメッセージのコミットがあるはず

フォーク



Issuesの例

AMCLのパラメータ調整 - Issue #113 - open-rdc/orne_navigation - Google Chrome

AMCLのパラメータ調整 #113

open-rdc / orne_navigation

Issues 113

AMCLのパラメータ調整 #113

Nasupl opened this issue on 23 Oct 2015 · 8 comments

Nasupl commented on 23 Oct 2015

- チューニング前にランドマークが開けた場所でロボットを走行させてパーティクルの広がりを見視したほうがいい
- 障害物や自己位置のスレでパーティクルの広がりが多いに早い場合はオドメトリの信頼度を上げる

iou16 was assigned by Nasupl on 23 Oct 2015

iou16 commented on 23 Oct 2015

@DaikiMaekawa オドメトリ信頼度を上げるということですが、これはサンプリングステップの誤差パラメータ(具体的にはamclのodom_alpha1, odom_alpha2, odom_alpha3, odom_alpha4)を低く設定することでパーティクルの広がりを抑えるようにするという形ですよろしいでしょうか？

DaikiMaekawa commented on 26 Oct 2015

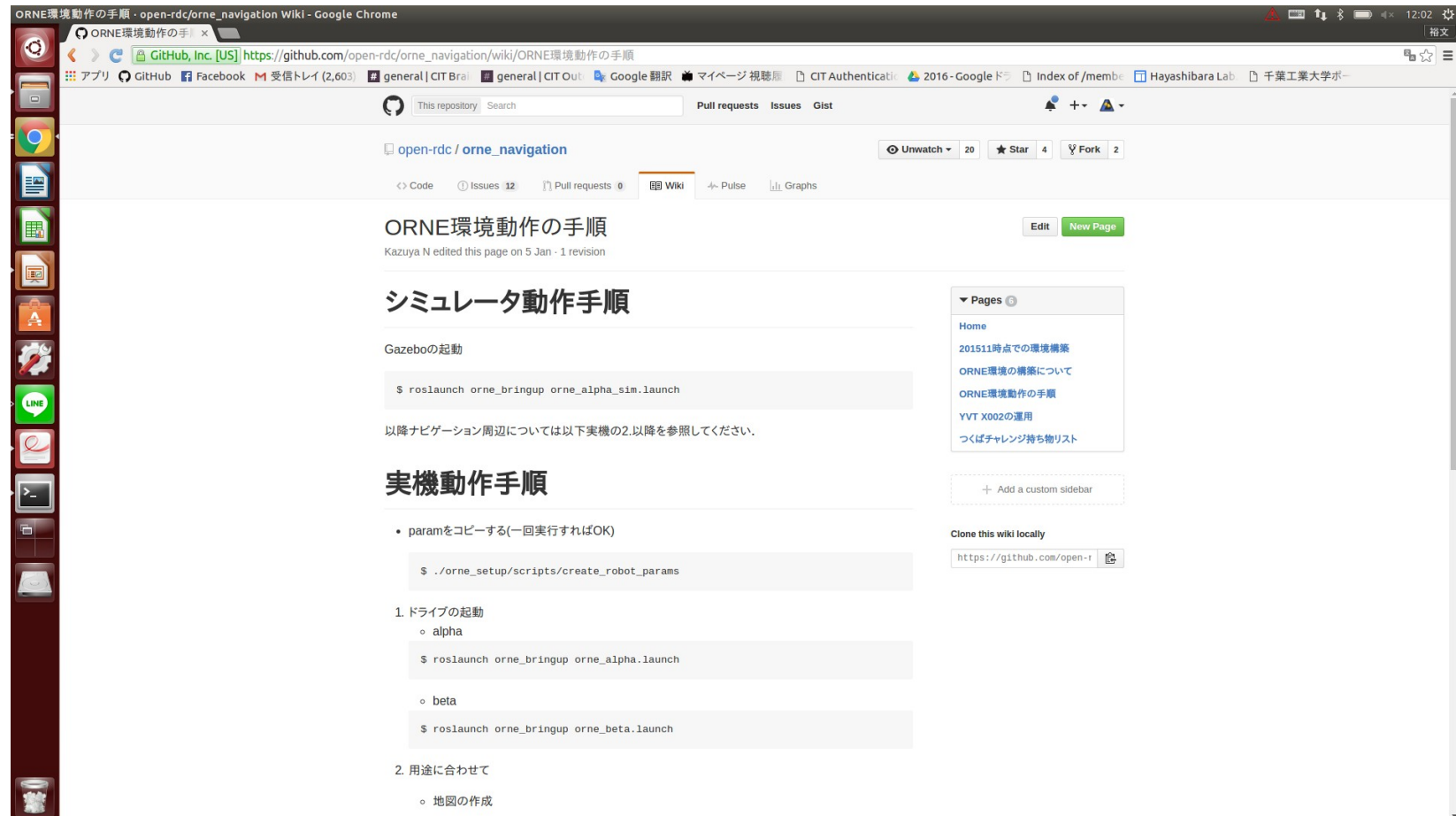
はい、差し当たりはその方法でノイズの期待値を下げて試してみてください。こういったパラメータの調整は結構厄介な作業なので動作中やバグファイルで挙動を何度も確認し、必要に応じて他の該当パラメータもいじる必要が出てくるかもしれません。とにかく検証実験は数をこなすしかありません。

iou16 commented on 27 Oct 2015

パラメータ調整の現状を報告いたします。

- 今回は、公園内においてスタート地点(テント地帯から見て右・トイレ付近)の直線からスタートし、そのまま開けた場所(公園の道路側・地図に記録されるような障害物なし)に突入した場合に発生したパーティクルの発散を学内の環境で疑似的に再現し検証しました。
- odom_alpha4(回転方向の誤差を基準に回転方向の分散を決めるパラメータ)を低くする(信頼度を上げる)と、

Wikiの例



ORNE環境動作の手順 · open-rcd/orne_navigation Wiki · Google Chrome

GitHub, Inc. [US] https://github.com/open-rcd/orne_navigation/wiki/ORNE環境動作の手順

アプリ GitHub Facebook 受信トレイ (2,603) general | CIT Brai general | CIT Out Google 翻訳 マイページ 視聴 録 CIT Authentica 2016 - Google ド Index of /membe Hayashibara Lab 千葉工業大学ポ

This repository Search Pull requests Issues Gist

open-rcd / orne_navigation Unwatch 20 Star 4 Fork 2

<> Code Issues 12 Pull requests 0 Wiki Pulse Graphs

ORNE環境動作の手順

Kazuya N edited this page on 5 Jan · 1 revision

Edit New Page

シミュレータ動作手順

Gazeboの起動

```
$ roslaunch orne_bringup orne_alpha_sim.launch
```

以降ナビゲーション周辺については以下実機の2以降を参照してください。

実機動作手順

- paramをコピーする(一回実行すればOK)

```
$ ./orne_setup/scripts/create_robot_params
```

1. ドライブの起動
 - alpha

```
$ roslaunch orne_bringup orne_alpha.launch
```
 - beta

```
$ roslaunch orne_bringup orne_beta.launch
```
2. 用途に合わせて
 - 地図の作成

▼ Pages 3

- Home
- 201511時点での環境構築
- ORNE環境の構築について
- ORNE環境動作の手順
- YVT X002の運用
- つくばチャレンジ持ち物リスト

+ Add a custom sidebar

Clone this wiki locally

https://github.com/open-rcd/orne_navigation/wiki

まとめ

- Gitは分散型バージョン管理システムの1つ
- GitHubはGitをベースにチーム開発を支援するツール
- 実際に手を動かしてGitとGitHubの使い方を理解した