

# 計算機科学実験及演習3

## ハードウェア「Gitの使い方」

京都大学情報学科計算機科学コース

## 実験 3 ハードウェアでは、バージョン管理と設計データ・課題提出にGitHubを用います。

### ① Gitの概要

バージョン管理システムとGitの基本概念を紹介

### ② Gitチュートリアル

Gitの基本的使い方を簡単な例を用いて説明

### ③ リモートリポジトリ・GitHubの概要

本実験で使用するリモートリポジトリGitHubの紹介

### ④ GitHubチュートリアル

GitHubの使い方を説明

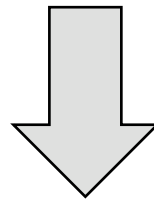
### ⑤ GitHub Classroom

GitHub Classroomの説明

# ①Gitの概要

## PCでの作業で起こりうること

- エラーを含んだ状態でプログラムを保存してしまった。
- レポートのファイルを間違っって削除してしまった
- 1週間ぶりに開くファイル、以前どんな編集をしたのか忘れてしまった。
- 二人で1つのファイルを編集してしまい、一方の人がした編集が反映されなかった。



適切なバージョン管理が必要

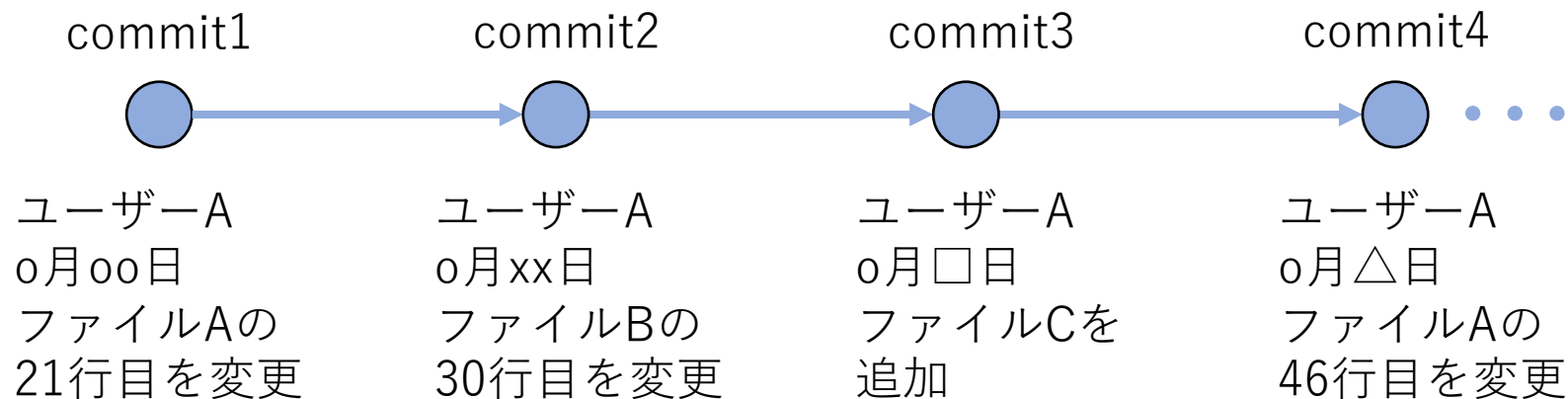
## Gitによるバージョン管理

- Gitは分散型のバージョン管理システム。
- Linuxのソースコードを管理するためにLinus Torvalds自身が開発。
- Gitの機能
  - － ファイルの変更履歴を保存する。
  - － 履歴には、いつ、誰が、どんな変更を行ったかを記録できる。
  - － いつでもファイルを保存した履歴の状態に戻せる。
  - － 他人が編集したファイルを上書きしようとするすると警告を出す。

# Gitの概要

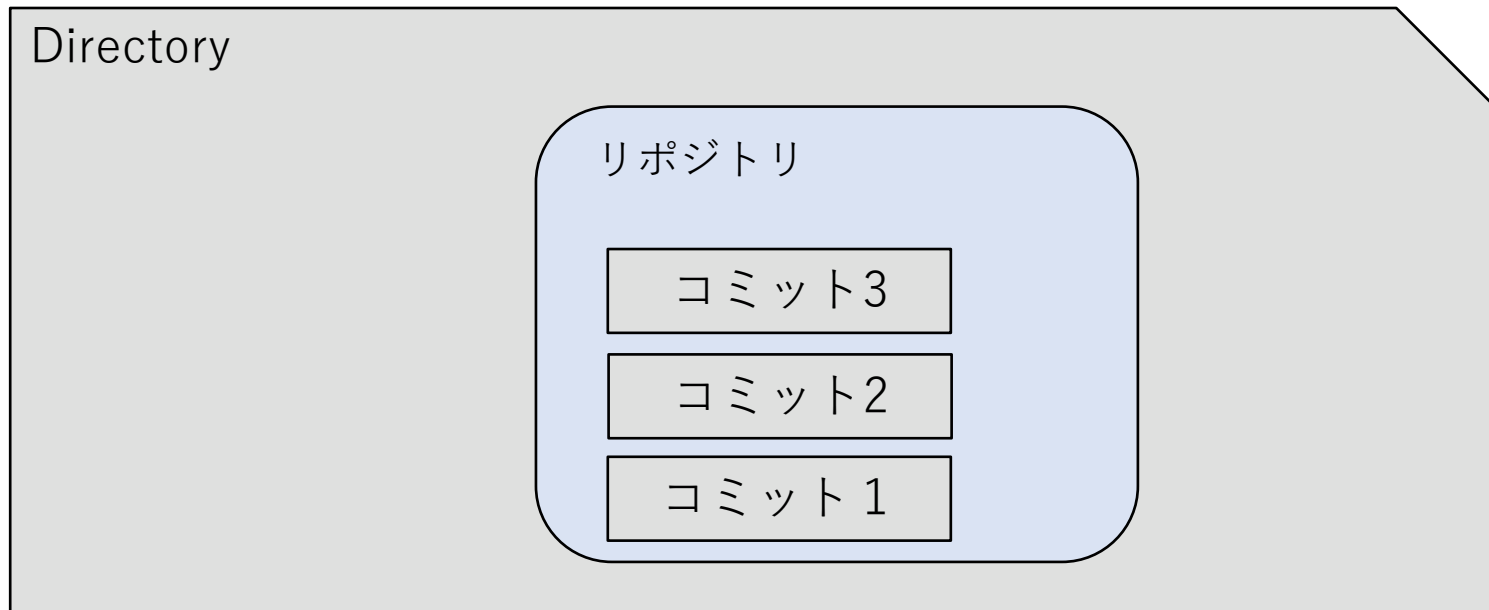
## commit

- Gitでファイルの変更履歴をつけていくことをcommitという。
- commitには、いつ、誰が、どんな変更を行ったかを記録しておける。
- いつでも、任意のcommitに戻ることができる。



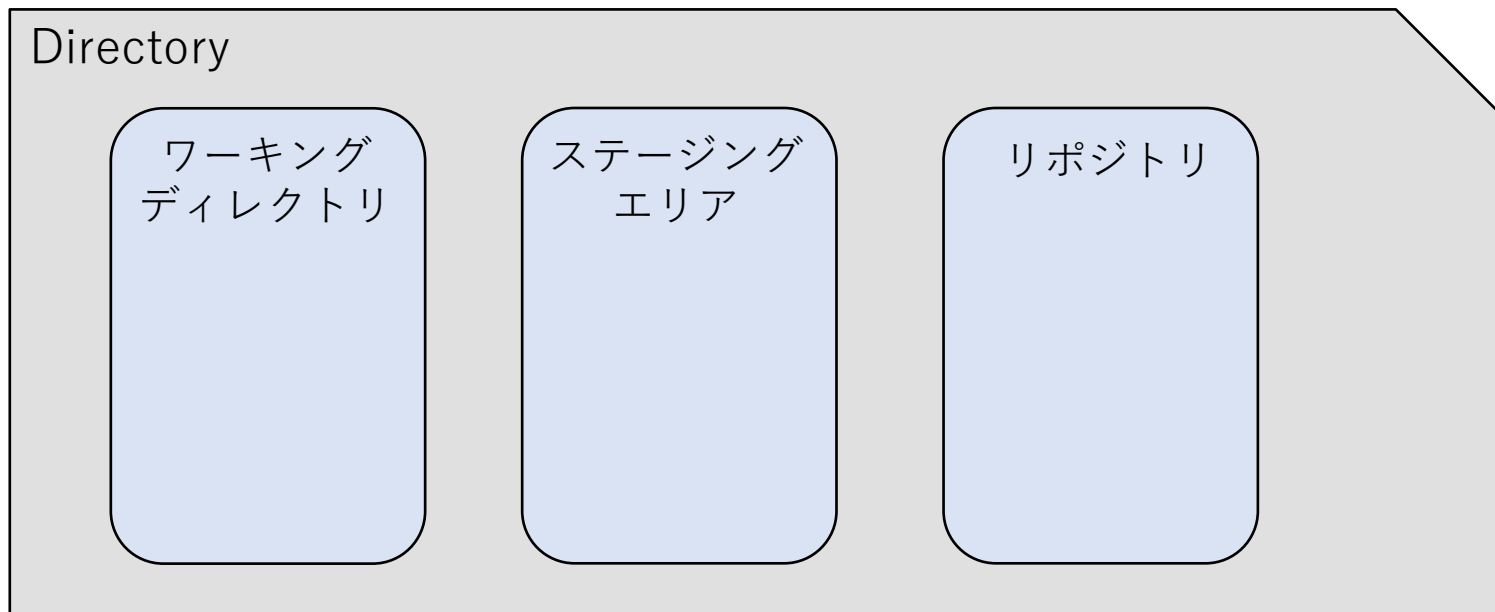
## リポジトリ

- commitは **リポジトリ** と呼ばれる場所に保存されていく。
- commitするたびに、その時のファイルの状態が保存されていく。



## commitの手順

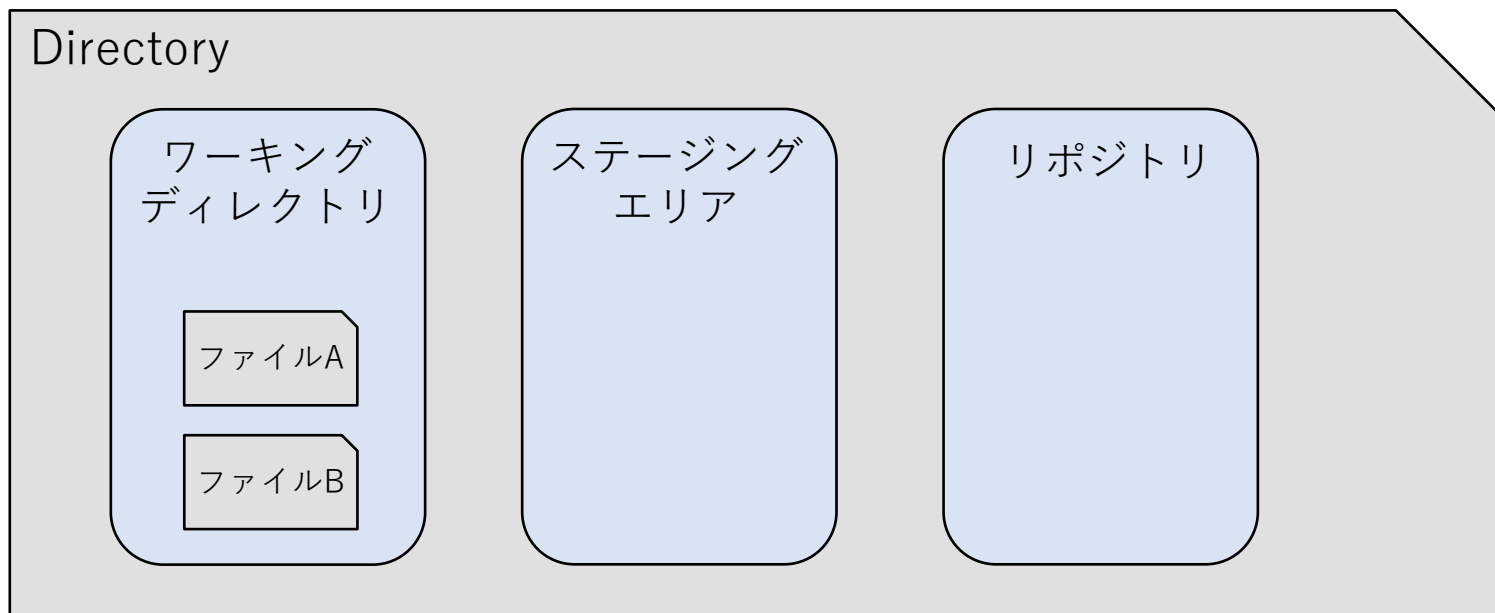
- ディレクトリをGitの管理下に置く（初期化する）とワーキングディレクトリ、ステージングエリア、リポジトリの3つの場所が作られる。





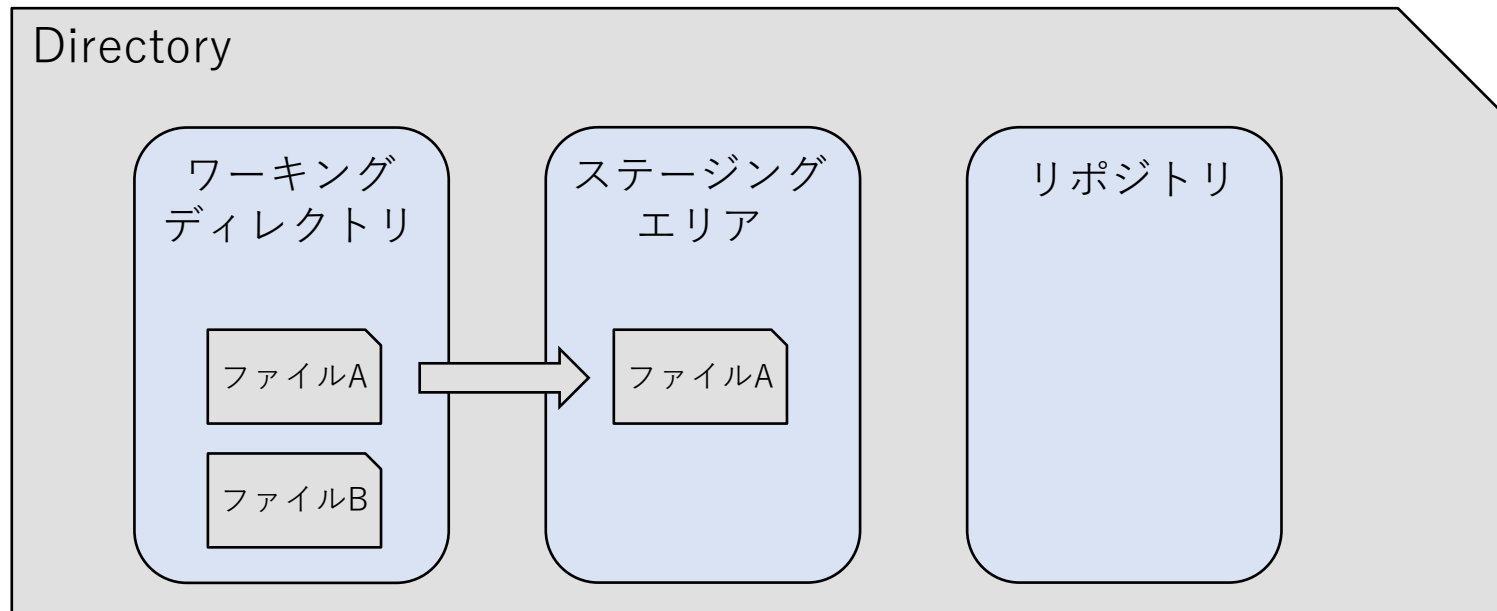
## commitの手順

- Gitで管理している、実際の作業を行うディレクトリのことをワーキングディレクトリという。



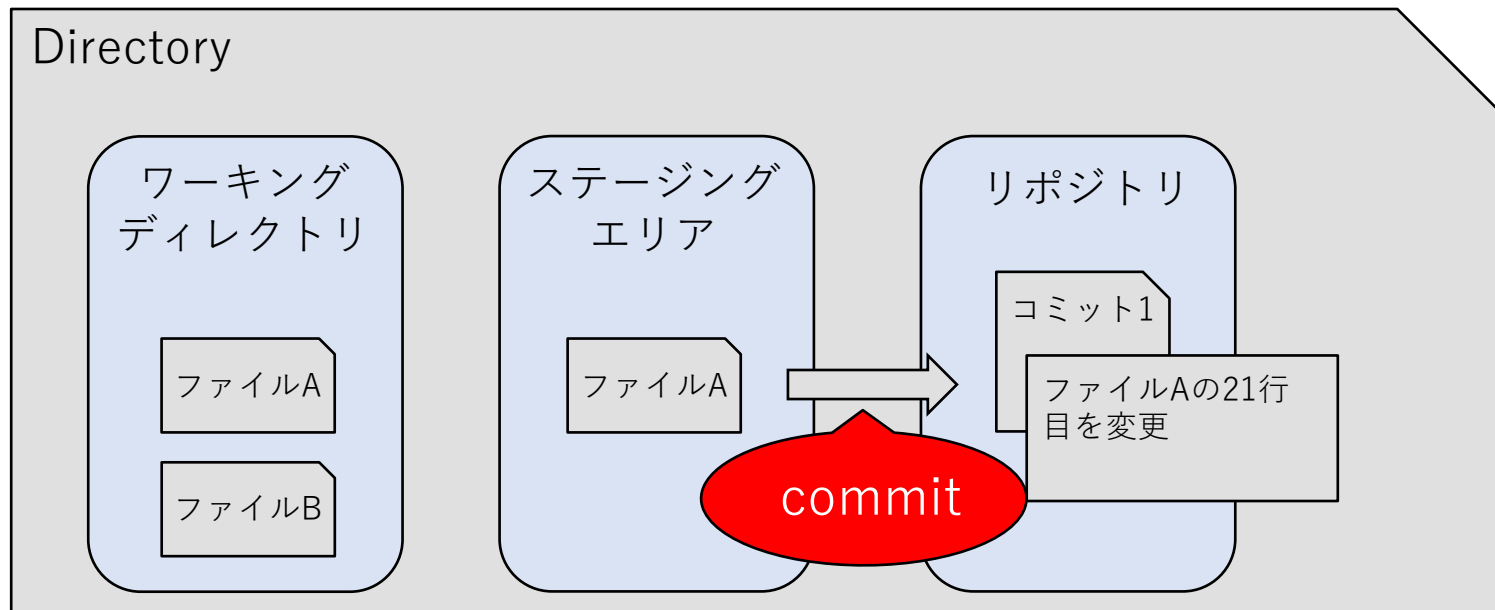
## commitの手順

- 変更履歴として保存するファイルを選択し、置いておく場所を **ステージングエリア** という。
- コミットの前に、ワーキングディレクトリのファイルの中から履歴として保存したいファイルを選びステージングする。



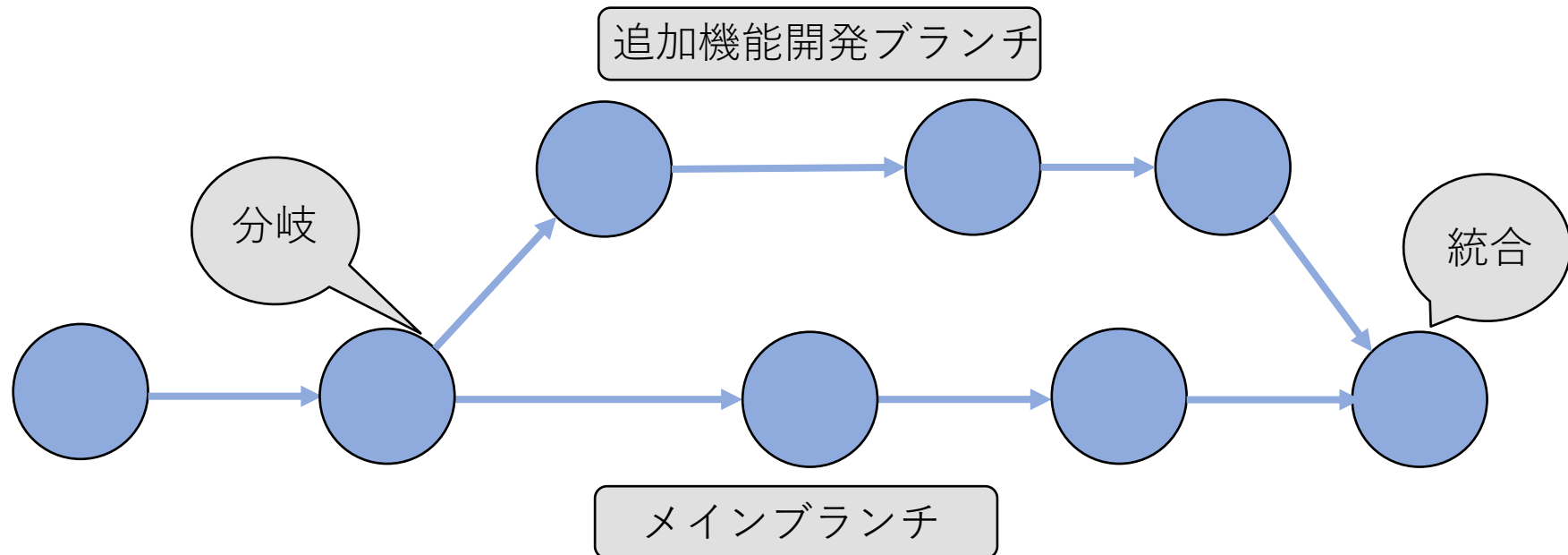
## commitの手順

- commitをすると、ステージングエリアに置かれているファイルの変更履歴が、リポジトリに保存される。
- ステージングエリアがあることで、ワーキングディレクトリの中の保存しておきたいファイルだけを選んで、commitすることができる。



## branch

- 変更履歴の流れを分岐することで並行での編集を支援する機能をbranchという。
- あるブランチへの変更は、他のbranchに影響しない。
- ブランチ同士の統合（merge）も可能。



## ②Gitチュートリアル

## Gitの利用

- Windowsの場合
  - Git for Windows をインストールする
  - WSL (Windows Subsystem for Linux) を利用する
  - Cygwin などのLinuxライクな環境を用意する互換レイヤーをインストールする
- Mac の場合
  - Xcode Command Line Tools をインストールする
    - 「git -version」などとした際に未インストールならインストール画面へ
- Linux の場合
  - 「sudo apt install git-all」 など

## Gitの初期設定

```
$ git config --global user.name "KAZUNARI KATO"  
$ git config --global user.email "foo@bar.com"
```

- Gitの操作はターミナル上で行う。
- `git config --global` コマンドによりユーザー名とメールアドレスを登録する。
- 登録したユーザーがcommitの際に、記録される

## Gitリポジトリを初期化する

```
git init
```

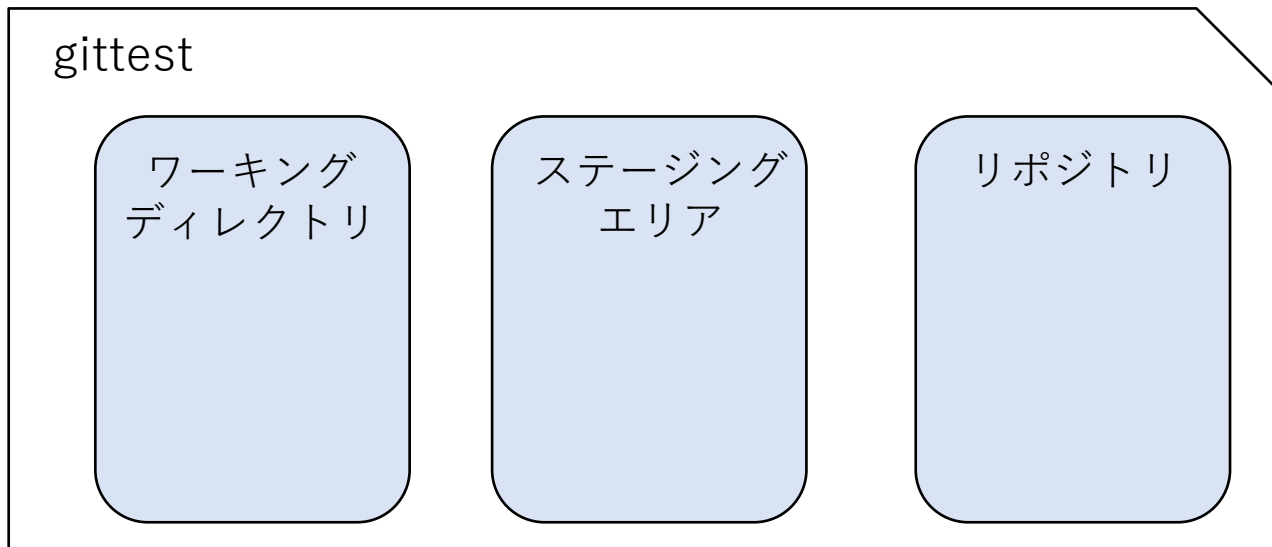
現在のディレクトリをGitリポジトリとして初期化する。



## Gitリポジトリを初期化する

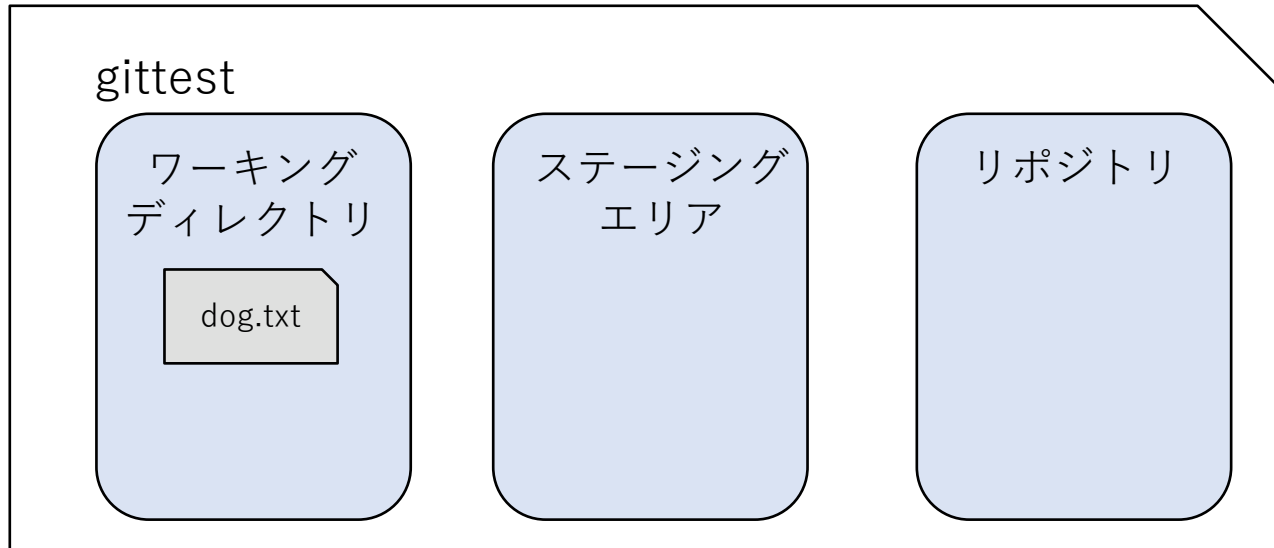
```
$ mkdir gittest
$ cd gittest
$ git init
Initialized empty Git repository in <ディレクトリ>
/gittest/.git
```

- gittestディレクトリがGitリポジトリとして認識される。
- ディレクトリには.gitディレクトリが作成されている。



ディレクトリ内にテキストファイルを作成する。

```
$ echo "bow wow" >> dog.txt
```



ワーキングディレクトリに`dog.txt`が追加される。

### 現在のディレクトリ内の状態を確認する

**git status**

ディレクトリ内の状態を確認する。

## 現在のディレクトリ内の状態を確認する

```
$ git status
On branch main

Initial commit

Untracked files:
(Use "git add<file>..." to include what will committed)

    dog.txt

nothing added to commit but untracked files present(use
"git add" to track
```

- 新しく作成したファイル（dog.txt）がUntracked files:の一覧に表示される。
- Untracked filesには一度もステージングしたことがないファイルが表示される

指定したファイルをステージングエリアへ追加する

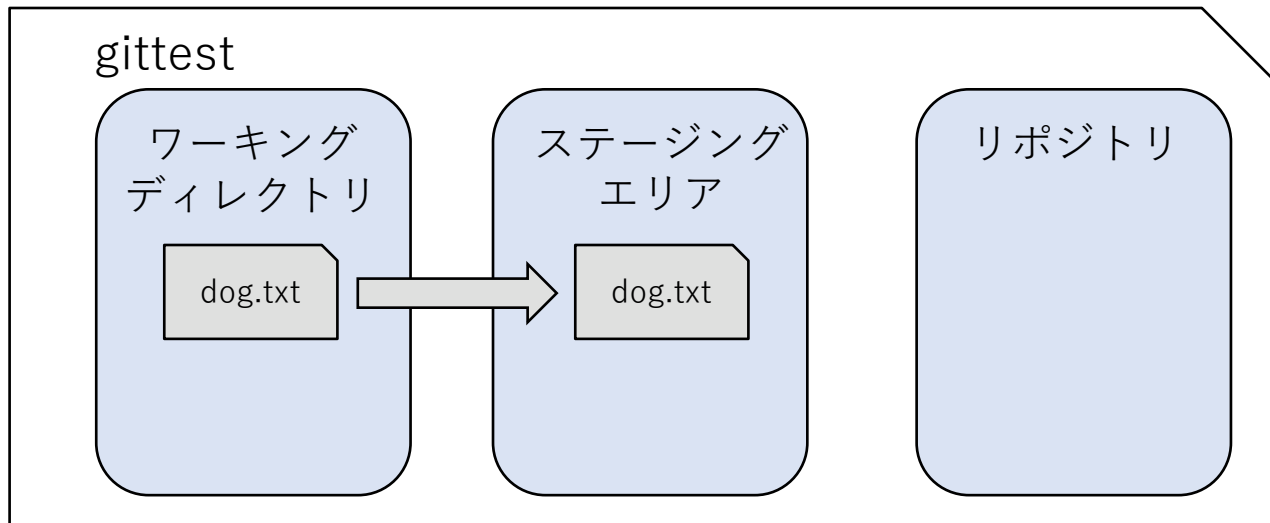
```
git add <ファイル名>
```

<ファイル名>をステージングエリアに追加する。

## 指定したファイルをステージングエリアへ追加する

```
$ git add dog.txt
```

- dog.txtがステージングエリアに追加される。



## ステージングエリア追加後の状態を確認する

```
$ git status
...
Changes to be committed:
(Use "git rm -cached <file>..." to unstage)

    new file:   dog.txt
```

- ステージング後に、再びgit statusコマンドにより状態を確認する。
- Changes to be committed:の一覧にdog.txtがnew fileとして表示されている。ステージングエリアに新たに追加されたことを示している。

ステージングされたファイルをcommitする。

```
git commit -m <コメント>
```

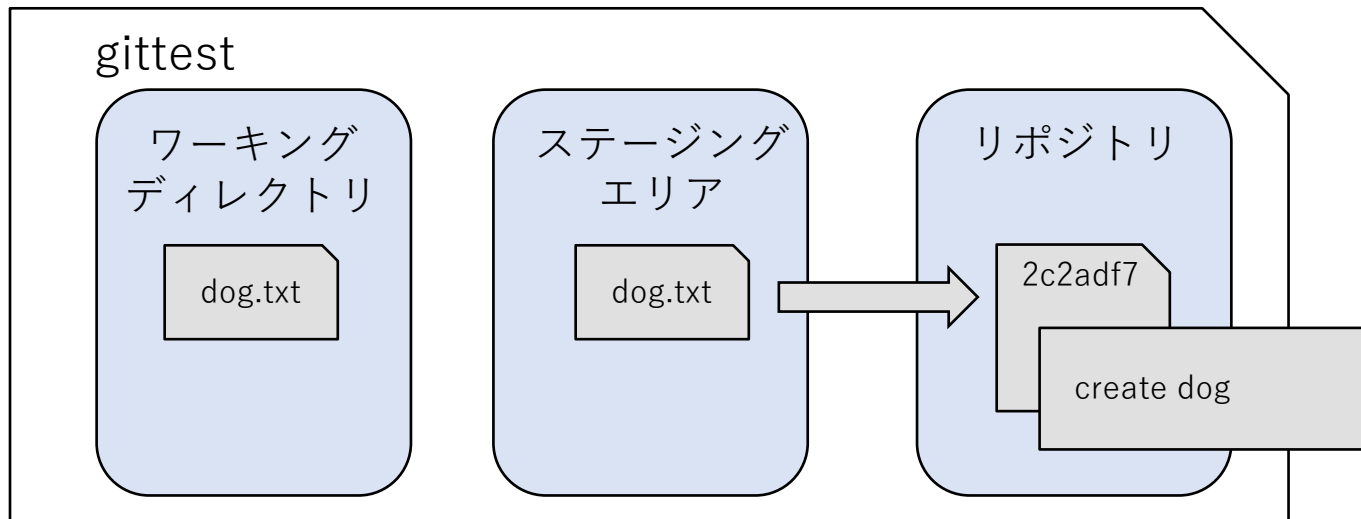
ステージングされているファイルをcommitする。  
-mオプションをつけると1行のコメントをつけることができる。



## ステージングされたファイルをcommitする。

```
$ git commit -m "create dog"  
[master (root-commit) 2c2adf7] create dog  
1 file changed, 1 insertion(+)
```

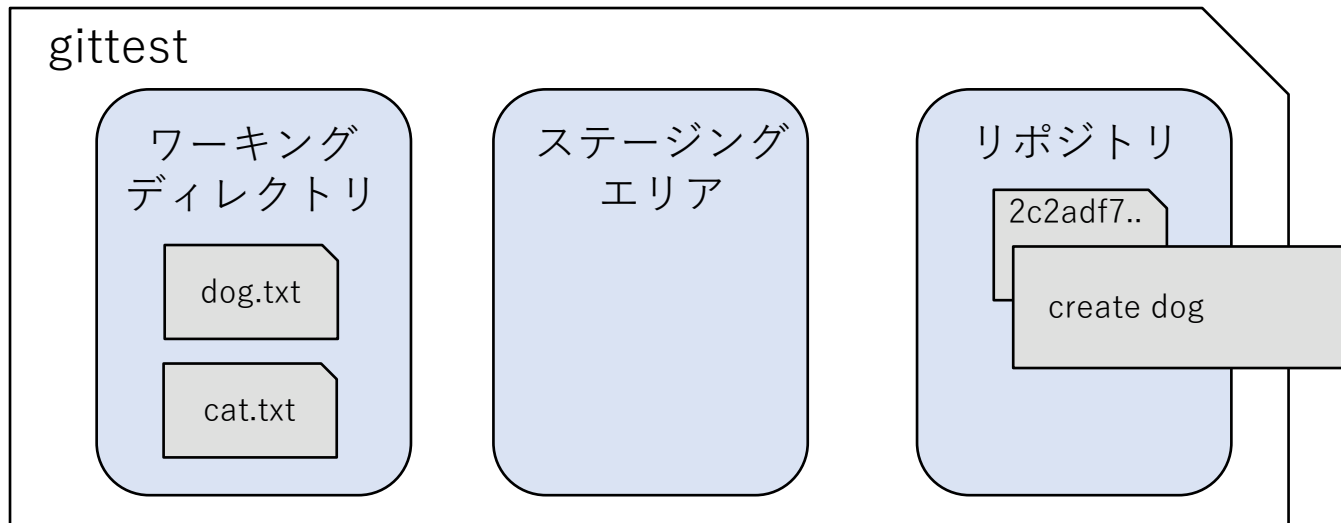
- create dogというコメントをつけてcommitする。
- ステージングされていたdog.txtがcommitされる。



## dog.txtの変更と、新規ファイルの追加。

```
$ echo "mew" >> cat.txt  
$ echo "wan wan" >> dog.txt
```

- dog.txtを編集し、新たにcat.txtを追加する。

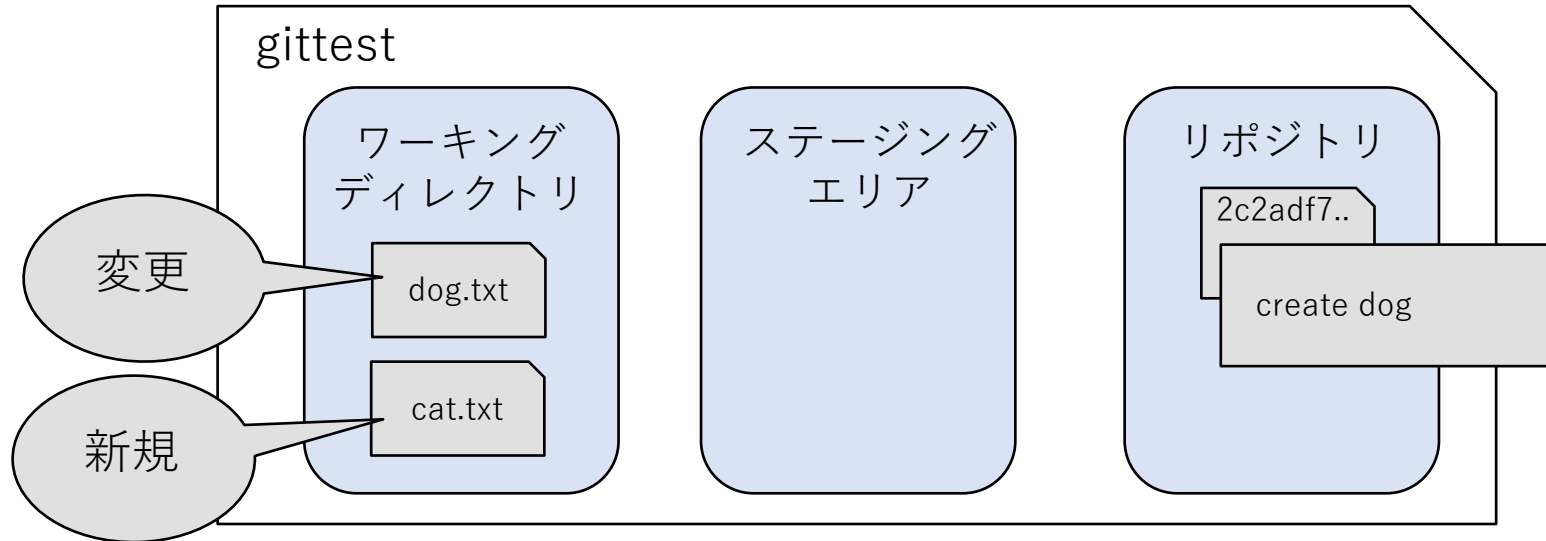


## dog.txtの変更と、新規ファイルの追加。

```
$ git status
On branch main
Changes not staged for commit:
...
    modified:   dog.txt
Untracked files:
...
    cat.txt
```

- Changes not staged for commitには過去にステージングしたことがあり、かつ最新のcommitから変更されているファイルが表示される。
- Untracked filesにはcat.txtが表示される。

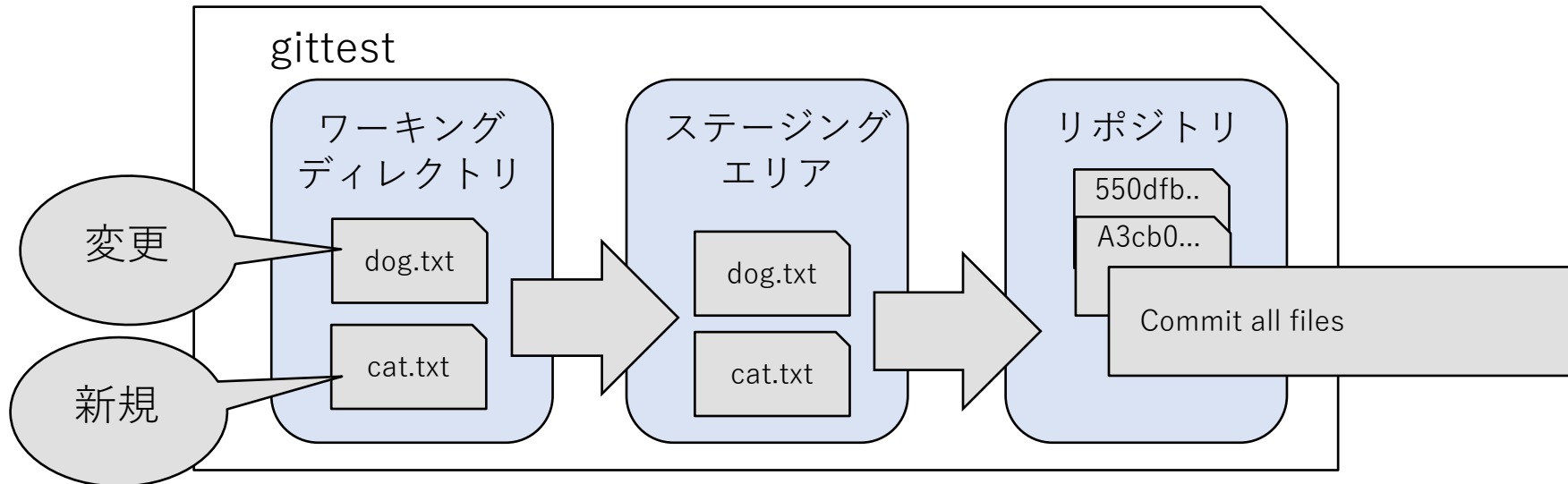
## dog.txtの変更と、新規ファイルの追加。



## 全ての変更をコミット。

```
$ git add *.txt  
$ git commit -m "commit all files"
```

- dog.txtの変更とcat.txtの追加が一つの変更としてコミットされる。
- git addのファイル指定ではワイルドカードが使える。
- "."を指定すると変更があった全てのファイルをステージングする。



## commitの履歴を確認する

**git log**

リポジトリの保存されているcommitの履歴を確認する。

## commitの履歴を確認する

```
$ git log
commit 550dfb1f4de340c3abe04549f52c4a6598ad4cbf
Author: Kazunari Kato<kato@example.com>
Date: Fri Apr 27 14:01:23 2018 +0900

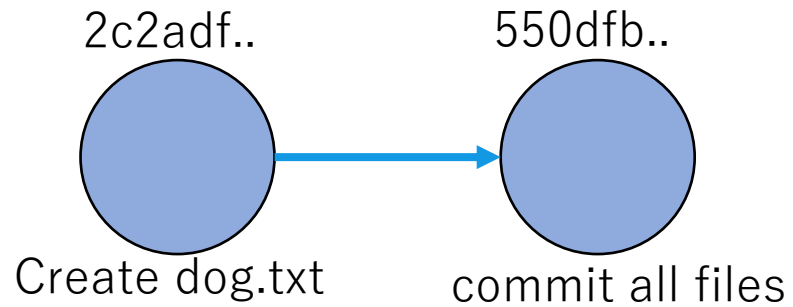
    commit all files

commit 2c2adf7291e16fd4301a625c8ac5589edc703f83
Author: Kazunari Kato<kato@example.com>
Date: Fri Apr 27 13:58:26 2018 +0900

    create dog.txt
```

- commit履歴が新しいものから順番に表示される。
- 1行目の文字列はcommit IDといって、commitを識別するためのもの。

## commitの履歴を確認する





以前のcommitの状態に戻す

```
git reset --hard <commit id>
```

ファイルを<commit id>で指定したcommitの状態に戻す。

## 以前のcommitの状態に戻す

```
$ echo "tweet" >> bird.txt
$ echo "meow meow" >> cat.txt
$ git status
// bird.txtの追加とcat.txtが変更されているメッセージが出る。
$ git reset --hard //コミットIDを省略すると最新のコミットに戻る
$ git status
// 変更が何もないというメッセージがでる。
```

- 新たにbird.txtを追加し、cat.txtを変更する。
- git reset --hard コマンドで、ファイルを最新のcommitの状態に戻す。
- less コマンドなどで、ファイルを確認すると、元に戻っている。

branch一覧を表示する。

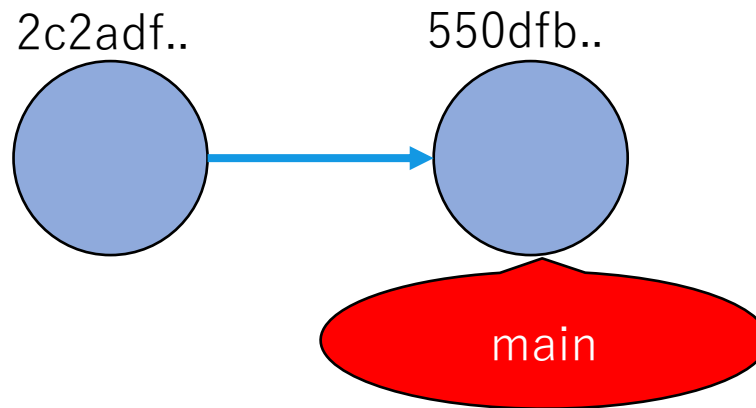
**git branch**

作成されているbranchの一覧を表示する。

## branchを利用する-branch一覧を表示する。

```
$ git branch  
* main
```

- 作成済みのbranchが表示される。
  - main branchは最初から存在している。
    - （デフォルトが master branch の場合も）
- \* は現在選択（checkout）されているbranchを示している。



## 新しくbranchを作成する

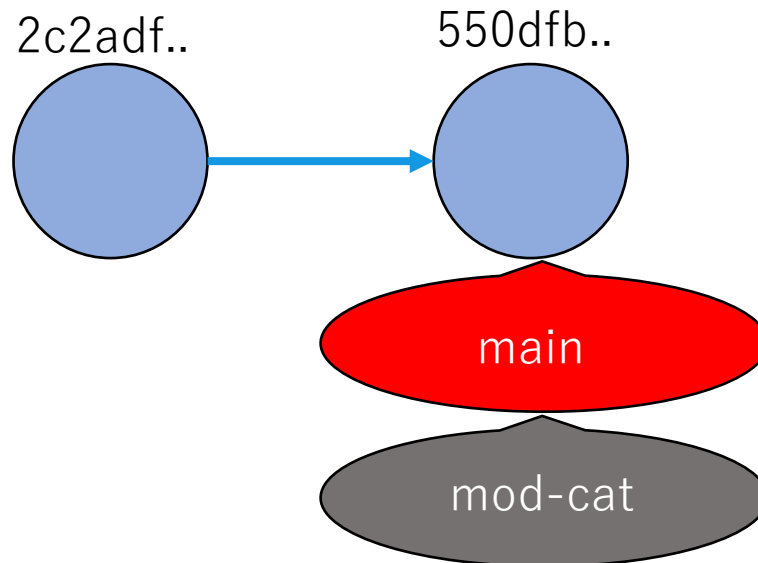
```
git branch <branch名>
```

新しくbranchを作成する。

## 新しくbranchを作成する

```
$ git branch mod-cat  
$ git branch  
* main  
  mod-cat
```

- mod-catというbranchを作成する。
- git branch で確認するとmod-cat branchが作成されているのがわかる。



## branchを選択する

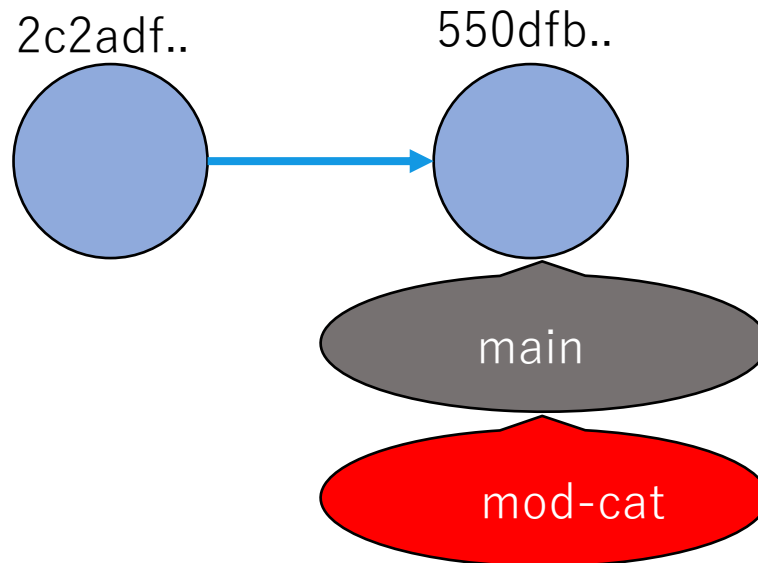
```
git checkout <branch名>
```

<branch名>で指定したブランチを選択する。

## branchを選択する

```
$ git checkout mod-cat  
Switched to branch 'mod-cat'  
$ git branch  
  main  
* mod-cat
```

mod-catブランチを選択して、git branchコマンドでmod-catが選択されていることを確認する。





(参考) branchを作成して選択する

```
git checkout -b <branch名>
```

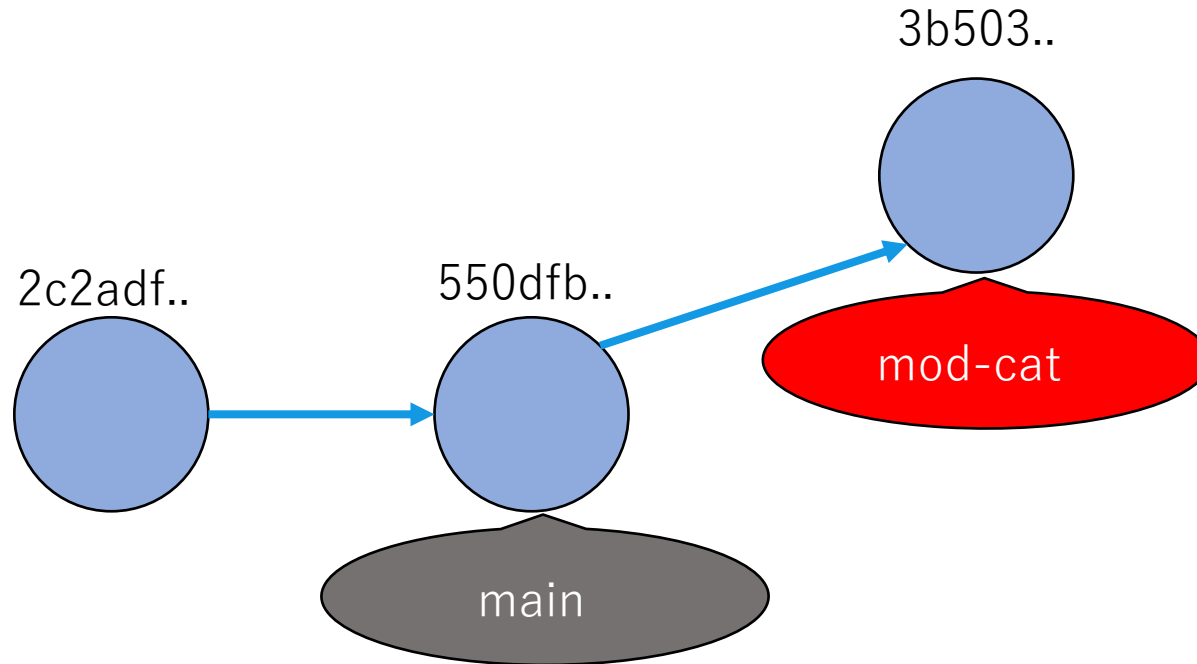
-b コマンドでを使うことで新規ブランチの作成と選択を同時に行うことも出来る。

## mod-catブランチでcommitする

```
$ echo "nya-" >> cat.txt
$ git add cat.txt
$ git commit -m "mod cat"
$ git log --oneline
* 3b503ba mod cat
* 55d0fb1f commit all files
* 2c2adf72 create dog
```

- cat.txtを変更して、commitする。
- git log コマンドに--onelineオプションをつけると履歴を省略形で表示する。
- 3つのcommitがあることを確認できる。

## mod-catブランチでcommitする



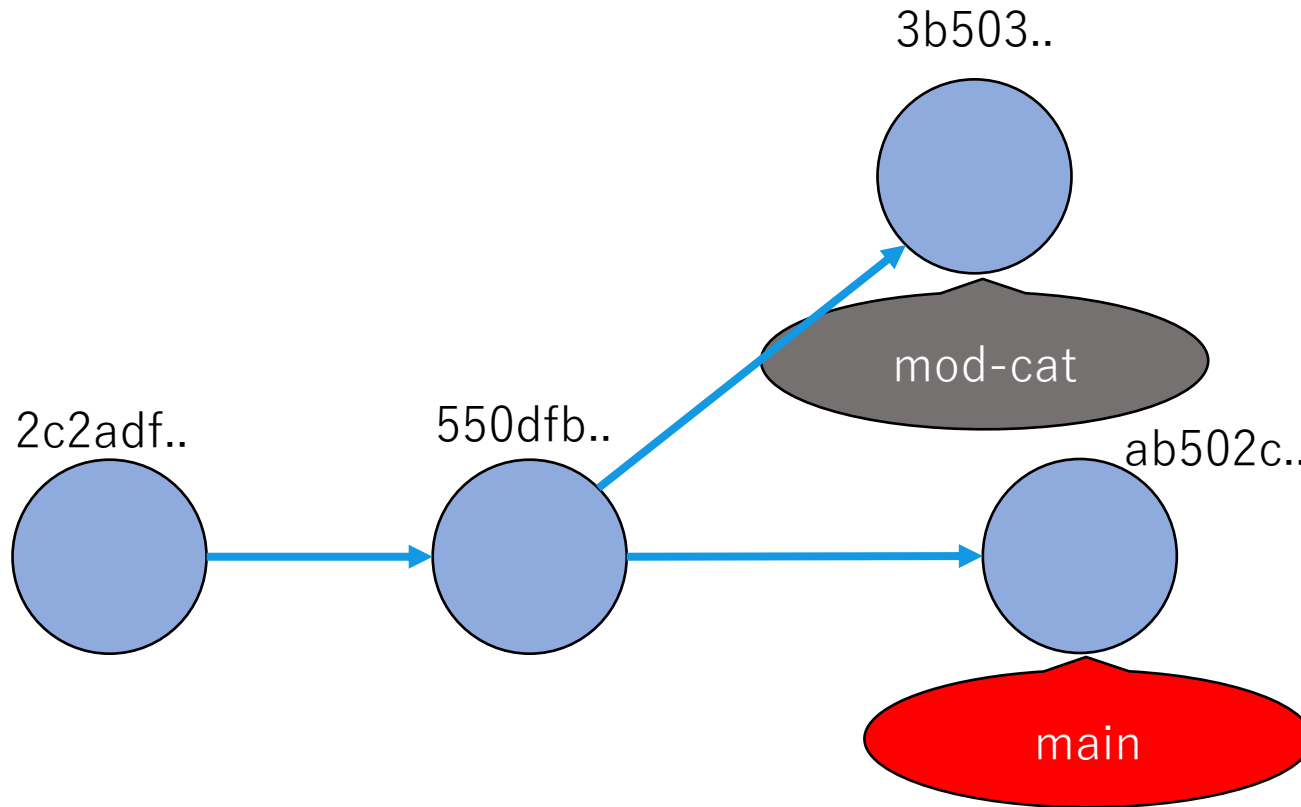
- 作成されたcommitはmod-catブランチで更新される。
- mainブランチは変わらずに550dfb..のcommitを保持している。

## mainブランチを更新する

```
$ git checkout main
$ echo "waon" >> dog.txt
$ git add dog.txt
$ git commit -m "mod dog"
$ git log
* e6fa391 mod dog
* 55d0fb1 commit all files
* 2c2adf7 create dog
```

- mainブランチを選択してdog.txtを変更しcommitする。
- commit履歴を確認するとmod-catブランチでのcommitは含まれておらず、mainブランチで行なったcommitのみが表示される。

## mainブランチでcommitを行う



- 新しく作成されたcommitはmainブランチで更新される。
- mod-catブランチで行われた変更はmainブランチには影響されない。

## branchをmergeする

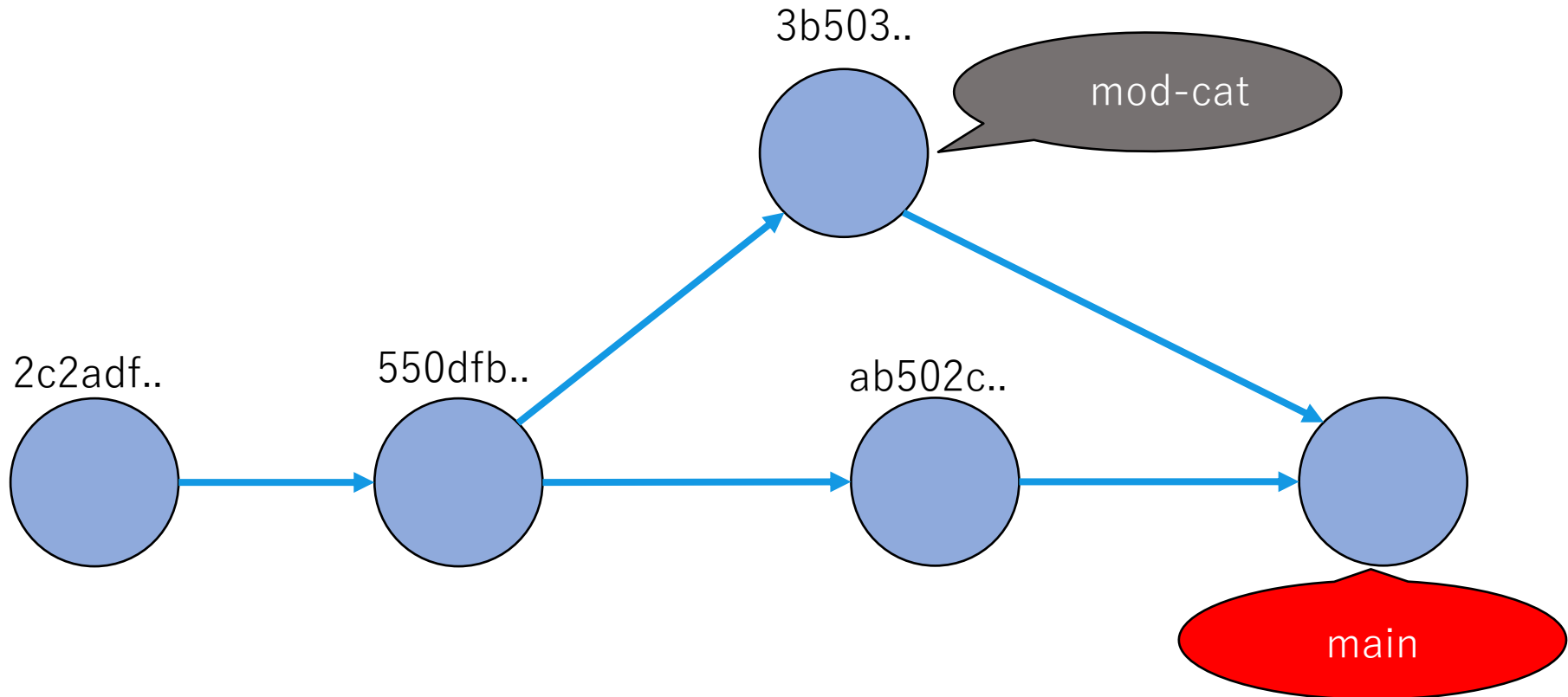
```
git merge <branch 名>
```

現在選択されているbranchに<branch 名>で指定したbranchをmergeする。

## mainブランチにmod-catブランチをmergeする

```
$ git checkout main  
$ git merge mod-cat
```

- mainブランチにcheckoutして、mod-catブランチをmergeする。



## mainブランチにmod-catブランチをmergeする

```
Merge branch 'mod-cat'
```

```
#.....
```

```
//省略
```

- merge commitを行うためのエディタが立ち上がるので、必要に応じてコメントを修正し、保存・終了する。

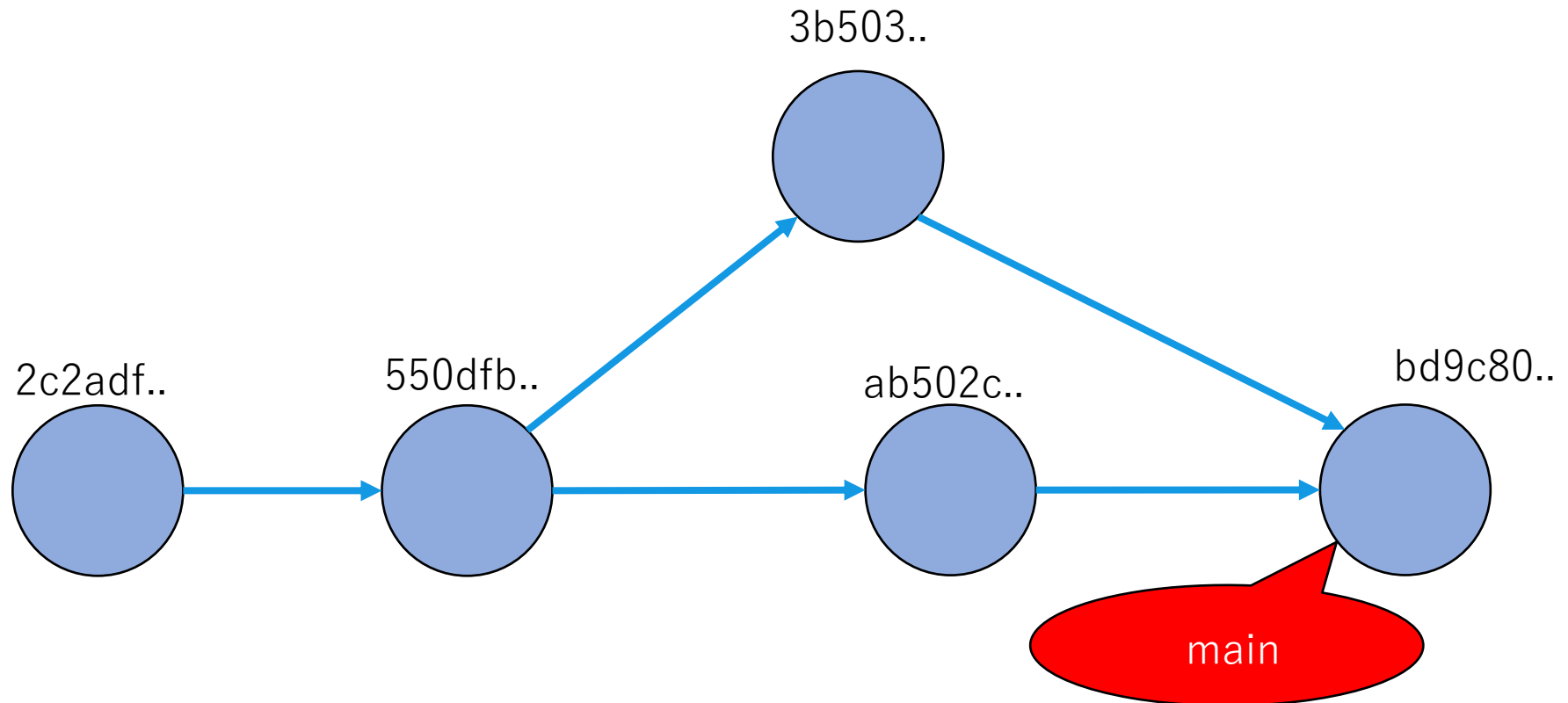


# Gitの使い方

## mainブランチにmod-catブランチをmergeする

```
$ git log --oneline --graph  
//省略
```

- git logコマンドに--graphオプションをつけると、履歴の分岐をグラフィカルに表示する。



## branchを破棄する

```
git branch -d <branch 名>
```

git branchに-dオプションをつけると<branch 名>で指定したbranchを破棄する。

## branchを破棄する

```
$ git branch -d mod-cat  
Delete branch mod-cat (was 23e1793)
```

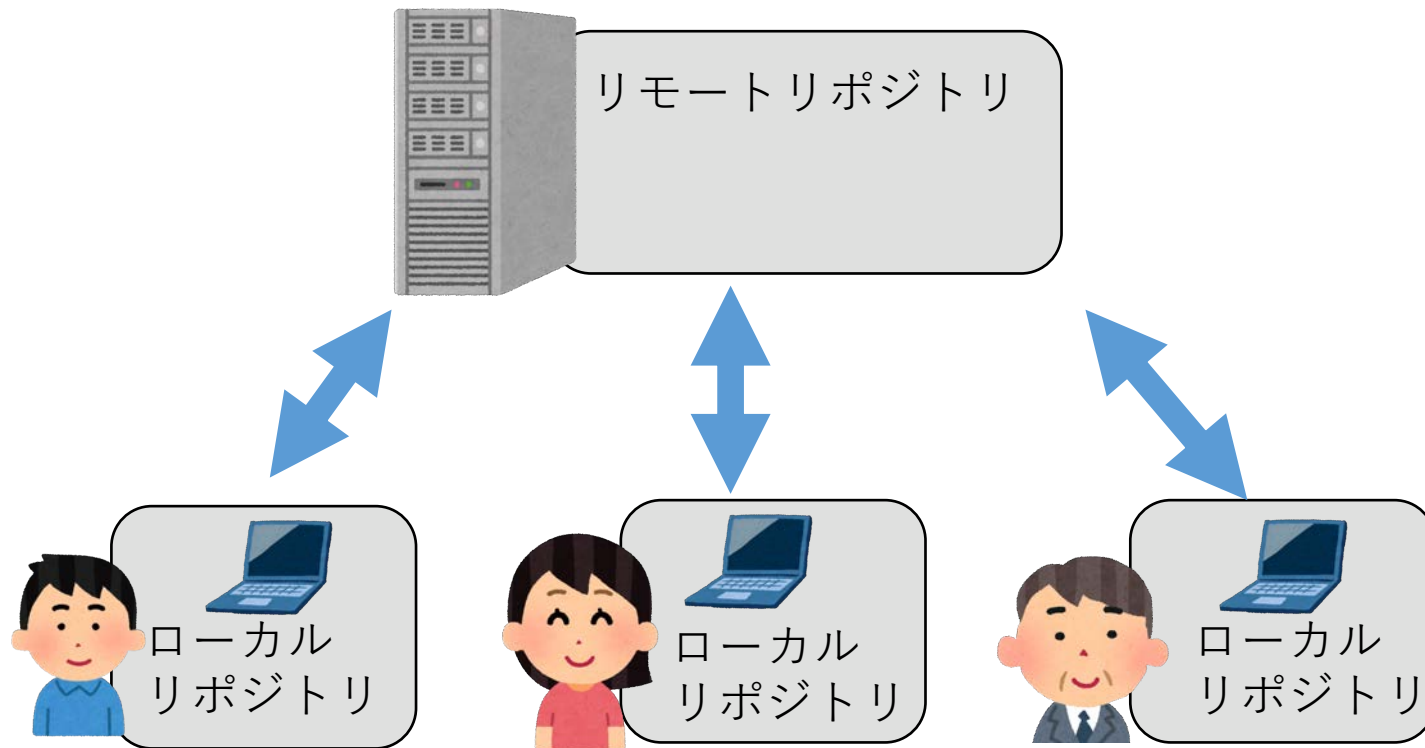
- 機能追加が完了し、不必要となったbranchは消去する。

### ③ リモートリポジトリ・ GitHubの概要

# リモートリポジトリの概要

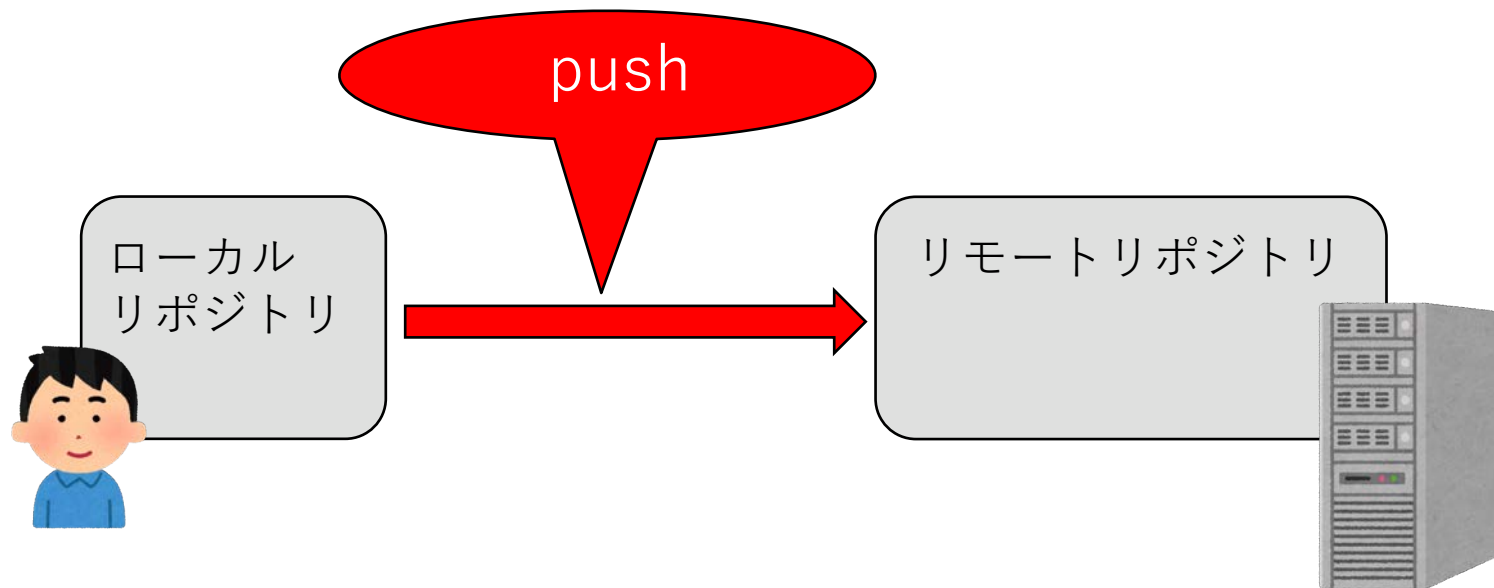
## ローカルリポジトリとリモートリポジトリ

- 自分の端末に存在し、個人で使用するリポジトリをローカルリポジトリ、サーバー上に存在し、複数の人が参照することのできるリポジトリをリモートリポジトリという。



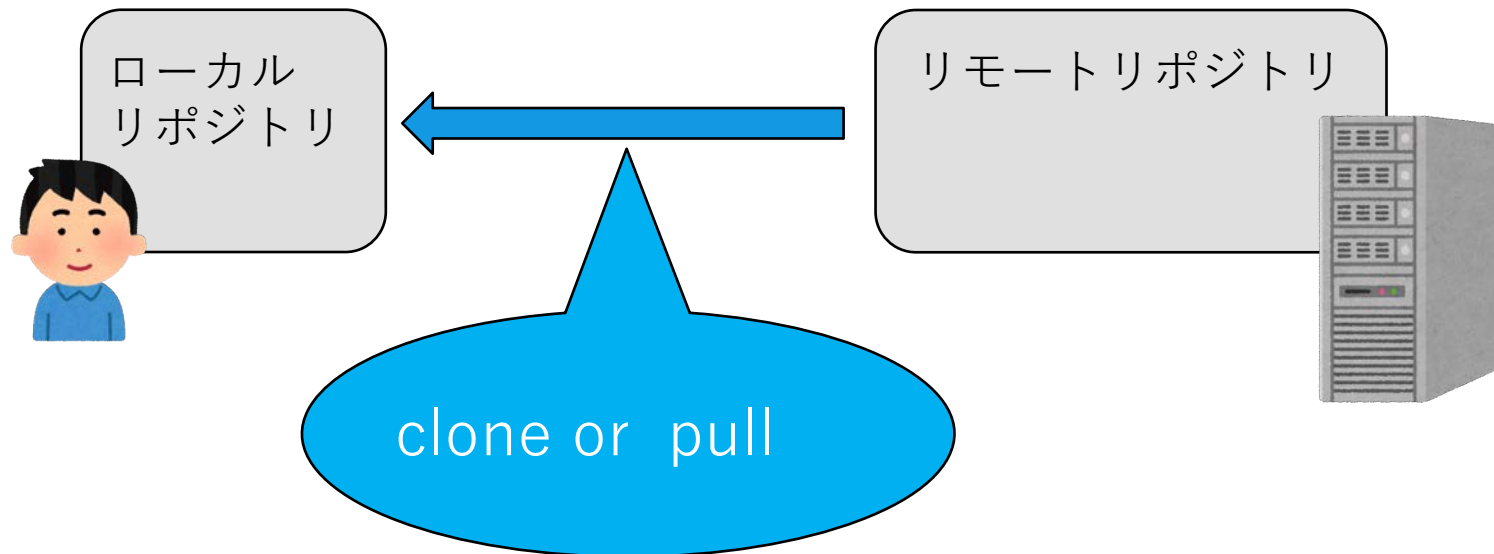
## Push

- リモートリポジトリに自分のローカルリポジトリにある変更履歴をアップロードすることをpushという。



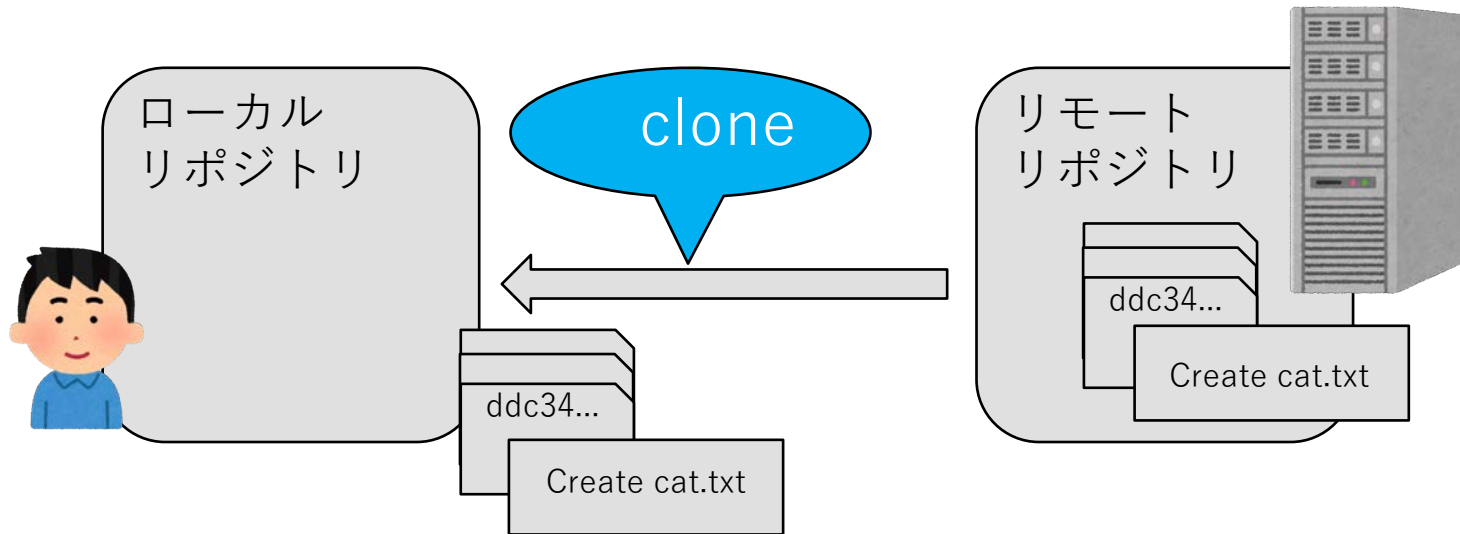
## cloneとpull

- リモートリポジトリにある変更履歴を自分のローカルリポジトリにダウンロードする方法に、**clone**と**pull**がある。



## clone

- **clone**は空のローカルリポジトリにリモートリポジトリのコピーを作成する時に使用。

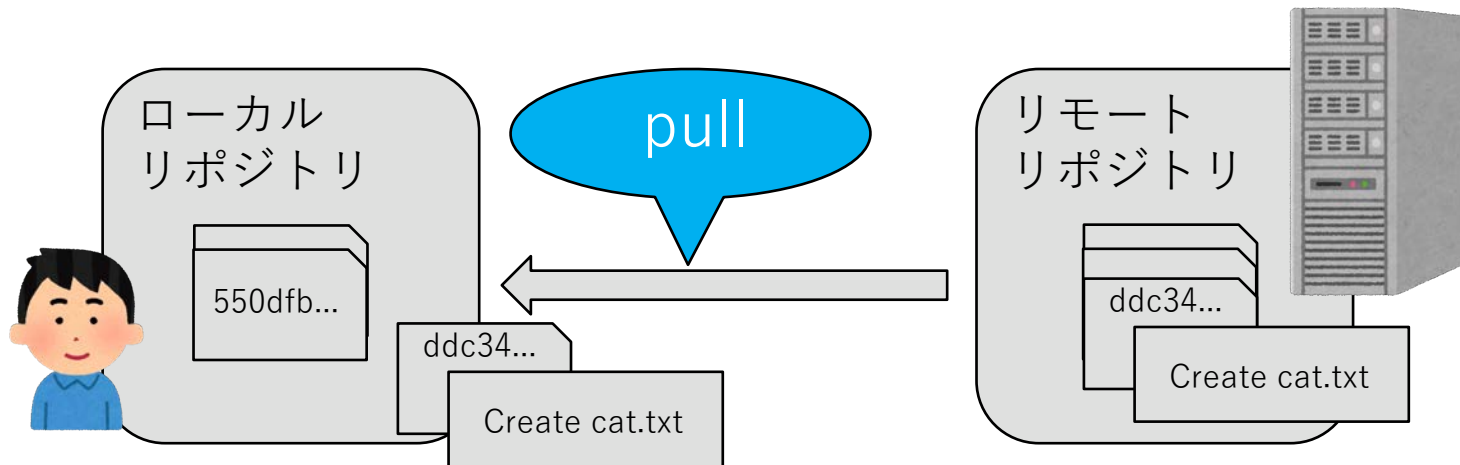




# リモートリポジトリの概要

## pull

- pullはローカルリポジトリをリモートリポジトリの履歴で更新する時に使用。



### GitHub

- GitHubはGitの仕組みを利用して、プログラムのソースコードなどを共有・ホスティングできるサービス。リモートリポジトリとして使用する。

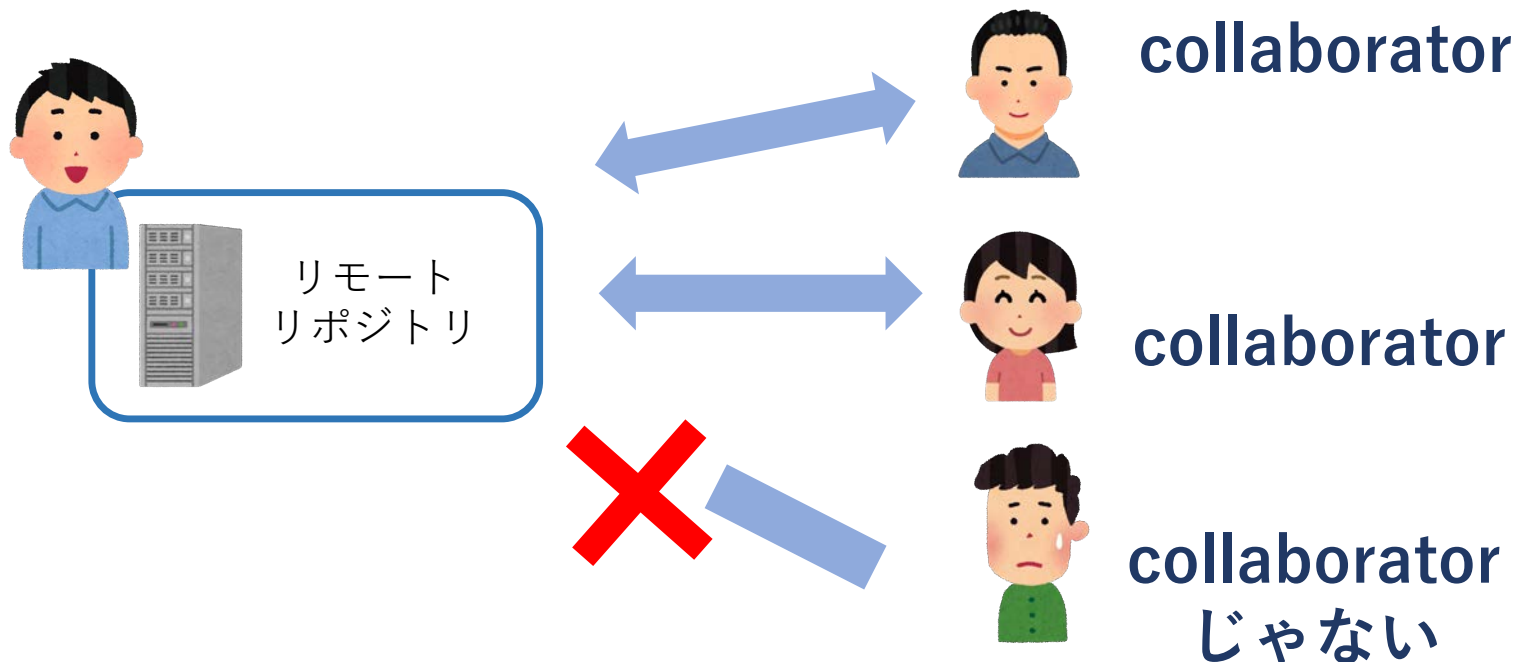
# GitHub

## リポジトリの公開設定

- GitHubでは自分のリポジトリをpublicとprivateいずれかに設定できる。
- Publicでは不特定多数の人がアクセス可能となる。
- Privateでは自分とcollaboratorに設定した人しかアクセスできない。
- 本実験で扱うリポジトリは必ずprivateにすること。

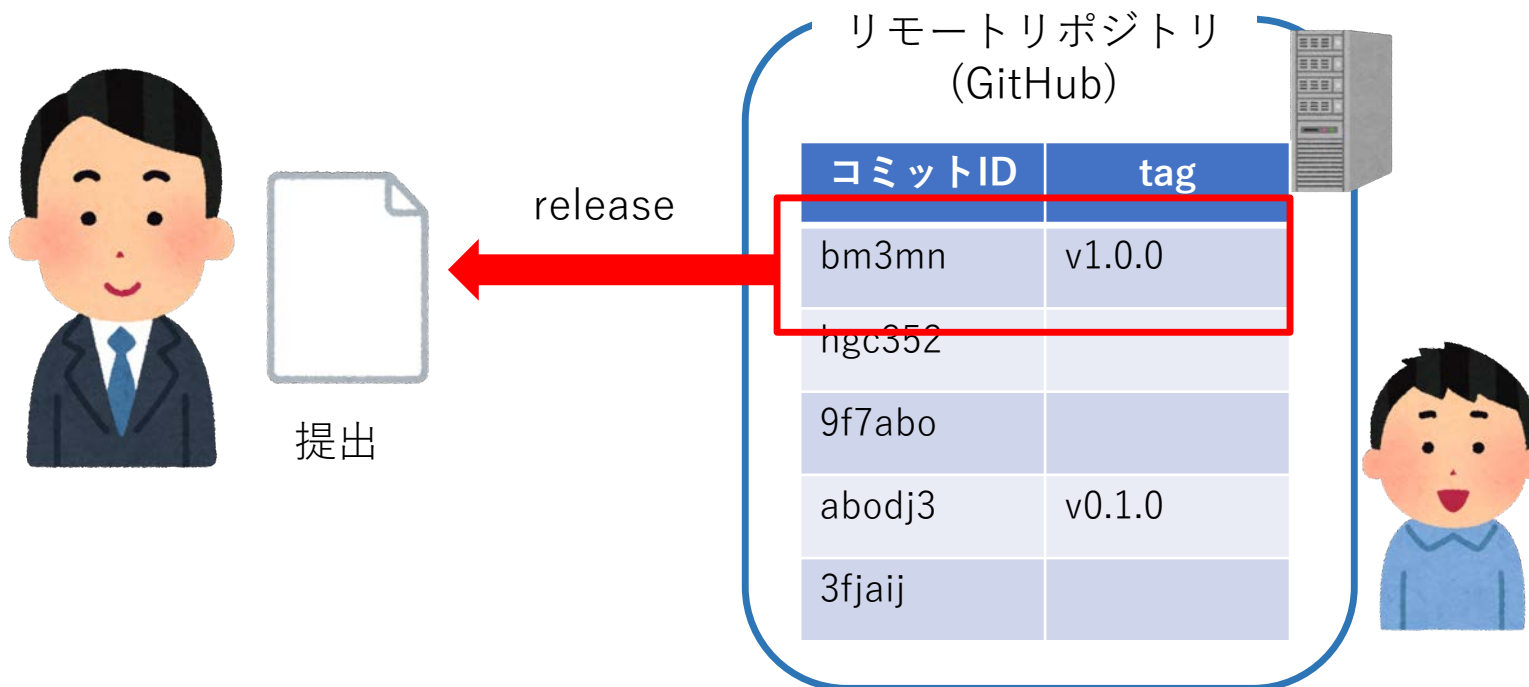
## collaborator

- 自分のprivateリポジトリに他のユーザを**collaborator**として登録すると、リポジトリを共有することができる。



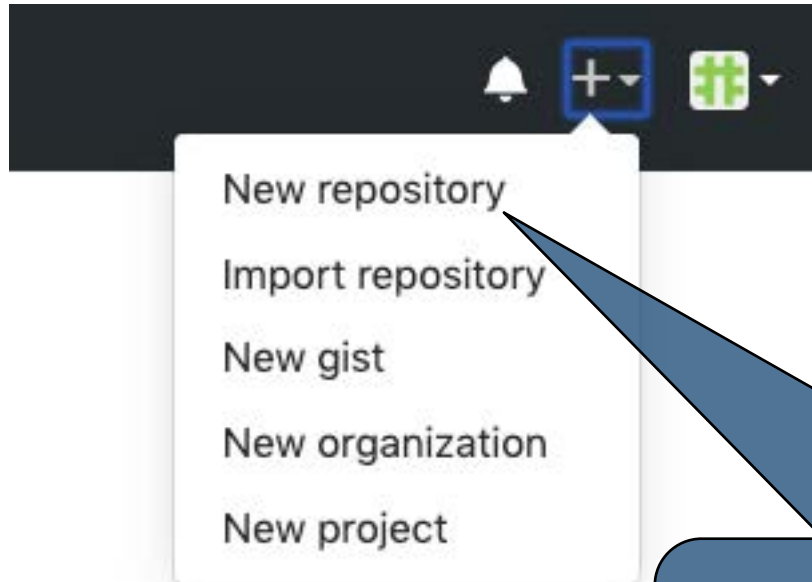
## tagとrelease

- 重要なコミットに分かりやすい名前をつけるための機能を **tag** という。
- tagがつけられたcommitを、他の人が使用できるように提供する機能を**release**という。本実験ではreleaseにより課題の提出を行う。



## ④GitHubチュートリアル

## リモートリポジトリの作成




ページ右上の「+」 → 「New repository」を  
クリック

## リモートリポジトリの作成

### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner

 **kazunarikato** ▾

Repository name \*

/ **le3-test** ✓

任意のリポジトリ  
の名前を入力

Great repository names are short and memorable. Need inspiration? How about [crispy-octo-succotash](#)?

Description (optional)

Privateにチェック  
を入れる

☐ **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 repo

☐ **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」  
をクリック

Create repository



## リモートリポジトリの作成

The screenshot shows the GitHub interface for creating a new repository. The repository name is 'kazunarikato/le3-test' and it is set to 'Private'. The page offers a 'Quick setup' for users who have done this before, providing a URL and instructions to create a new file or upload an existing one. It also provides command-line instructions for creating a new repository, pushing an existing one, or importing code from another repository. The 'le3-test' repository is highlighted with a green border.

リポジトリの名前と“Private”になっていることを確認。

リポジトリのアドレス

説明に従って、ターミナルでコマンド。

```
...or create a new repository on the command line

echo "# le3-test" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/kazunarikato/le3-test.git
git push -u origin master

...or push an existing repository from the command line

git remote add origin https://github.com/kazunarikato/le3-test.git
git push -u origin master

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code
```

## リモートリポジトリのアドレス登録

```
git remote add <エイリアス>  
               <URL>
```

<URL>を<エイリアス>という名前でリモートリポジトリとして登録する。

## リモートリポジトリのアドレス登録

kazunarikato / le3-test Private

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Actions Projects 0 Wiki Security Insights Settings

**Quick setup — if you've done this kind of thing before**

Set up in Desktop or **HTTPS** SSH `https://github.com/kazunarikato/le3-test.git`

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

**...or create a new repository on the command line**

```
echo "# le3-test" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/kazunarikato/le3-test.git
git push -u origin master
```

**...or push an existing repository from the command line**

```
git remote add origin https://github.com/kazunarikato/le3-test.git
git push -u origin master
```

**...or import code from another repository**

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

GitHubのリポジトリページに  
リポジトリのURLがある

## リモートリポジトリのアドレス登録

```
$ git remote add origin <リモートリポジトリのURL>
```

- 一般的にエイリアス名には“origin”という名前をつける。
- リモートリポジトリとしてURLをoriginというエイリアス名で設定する。

## リモートリポジトリへのプッシュ

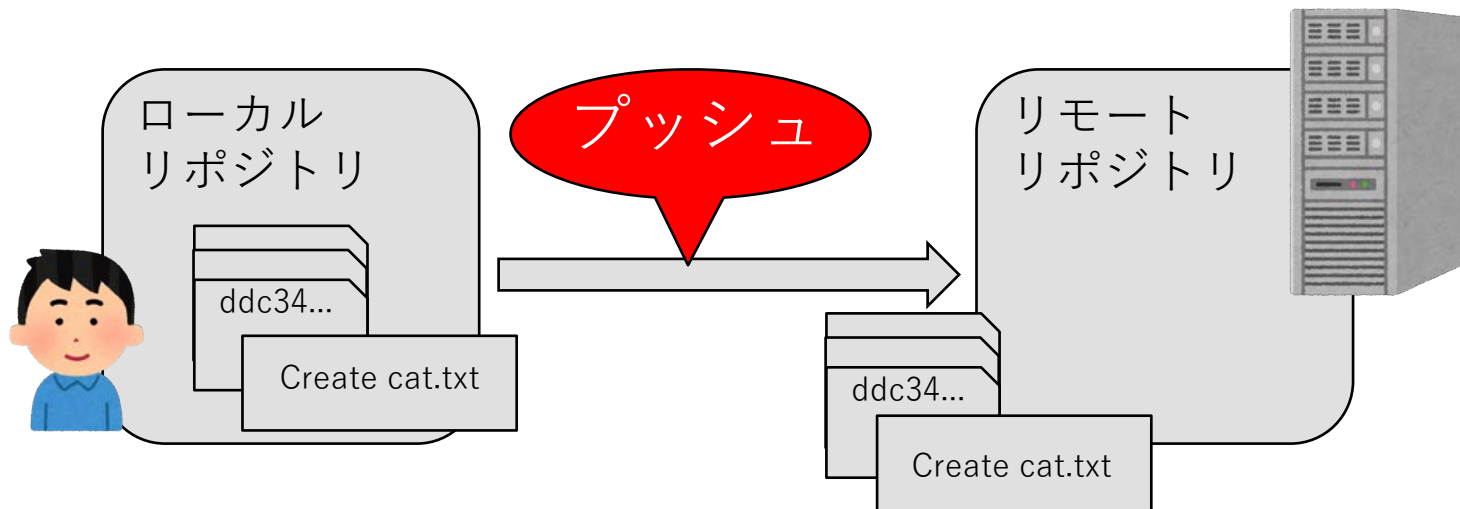
```
git push <エイリアス> <ブランチ>
```

<エイリアス>のリモートリポジトリにローカルの<ブランチ>をpushする。

## リモートリポジトリへのプッシュ

```
$ git push origin main
```

- originのリモートリポジトリにmainブランチをプッシュする。



## リモートリポジトリからのpull

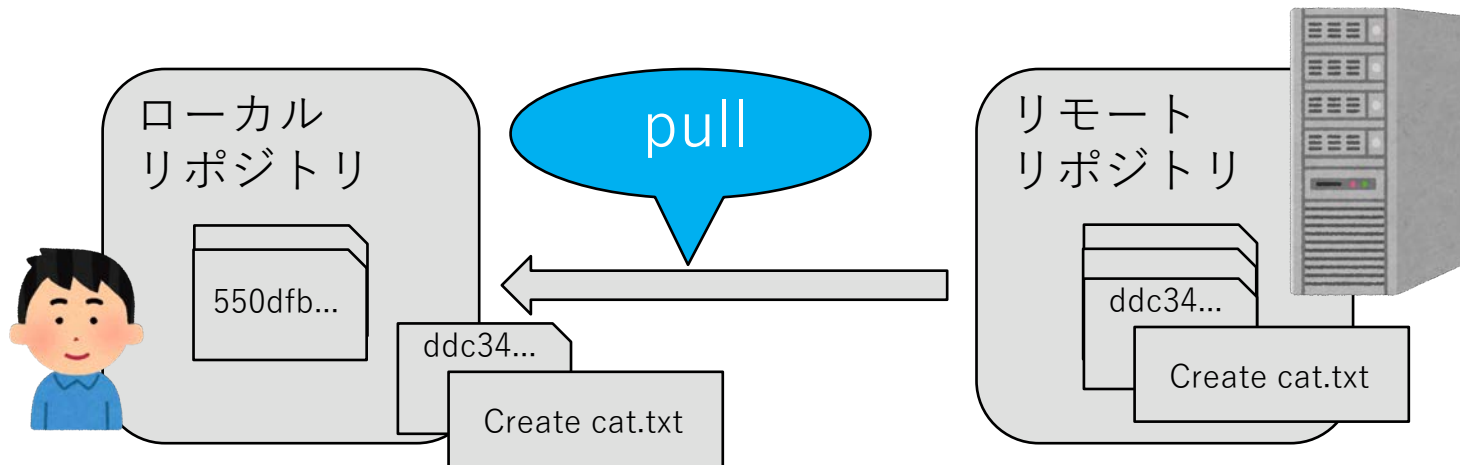
```
git pull <エイリアス> <ブランチ>
```

<エイリアス>のリモートリポジトリの内容でローカルの<ブランチ>を更新する。

## ローカルリポジトリを更新

```
$ git pull origin main
```

- リモートリポジトリの開発がローカルリポジトリよりも進んでいるとき、リモートリポジトリの内容でローカルリポジトリを更新する。





## リモートリポジトリからのclone

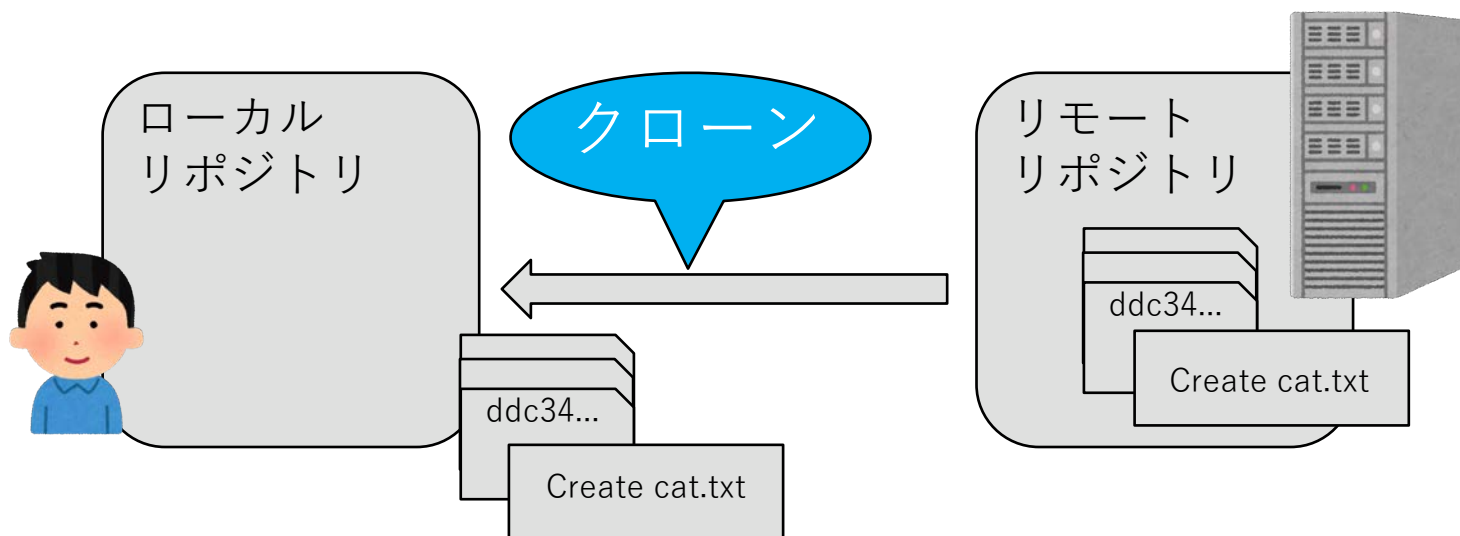
```
git clone <URL>
```

<URL>のリモートリポジトリの内容をローカルにコピーする。

## リモートリポジトリからのclone

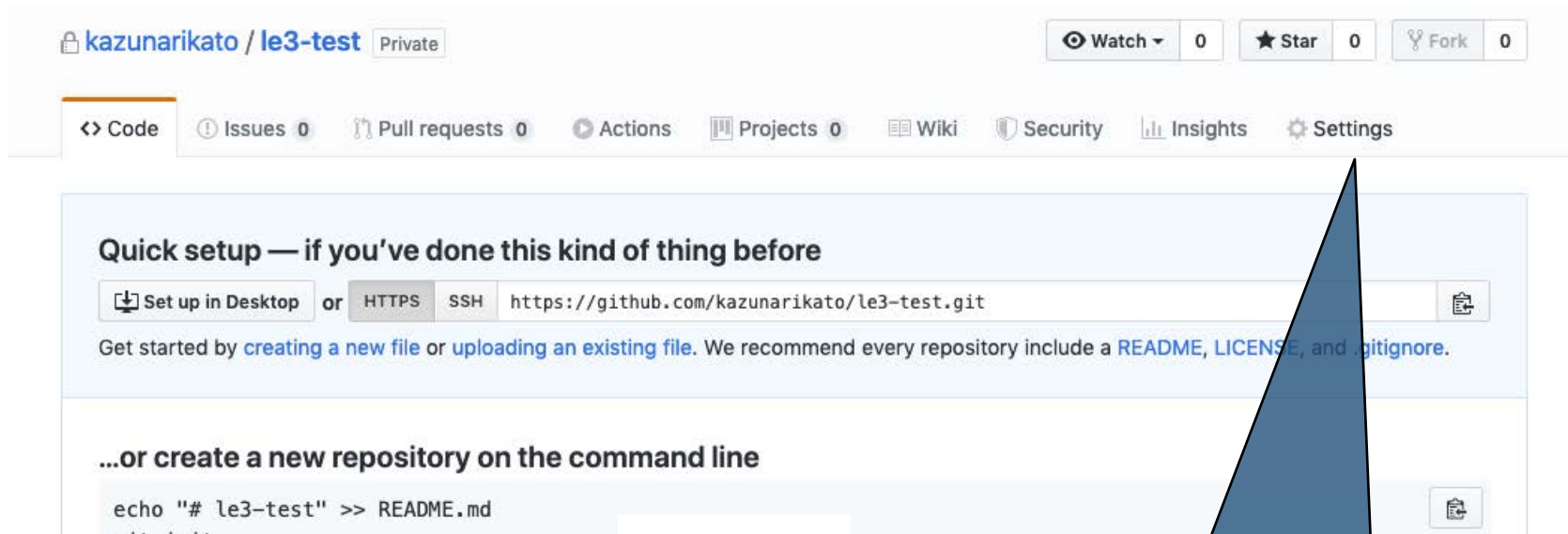
```
$ git clone <URL>
```

- <URL>で指定したリモートリポジトリをcloneする。



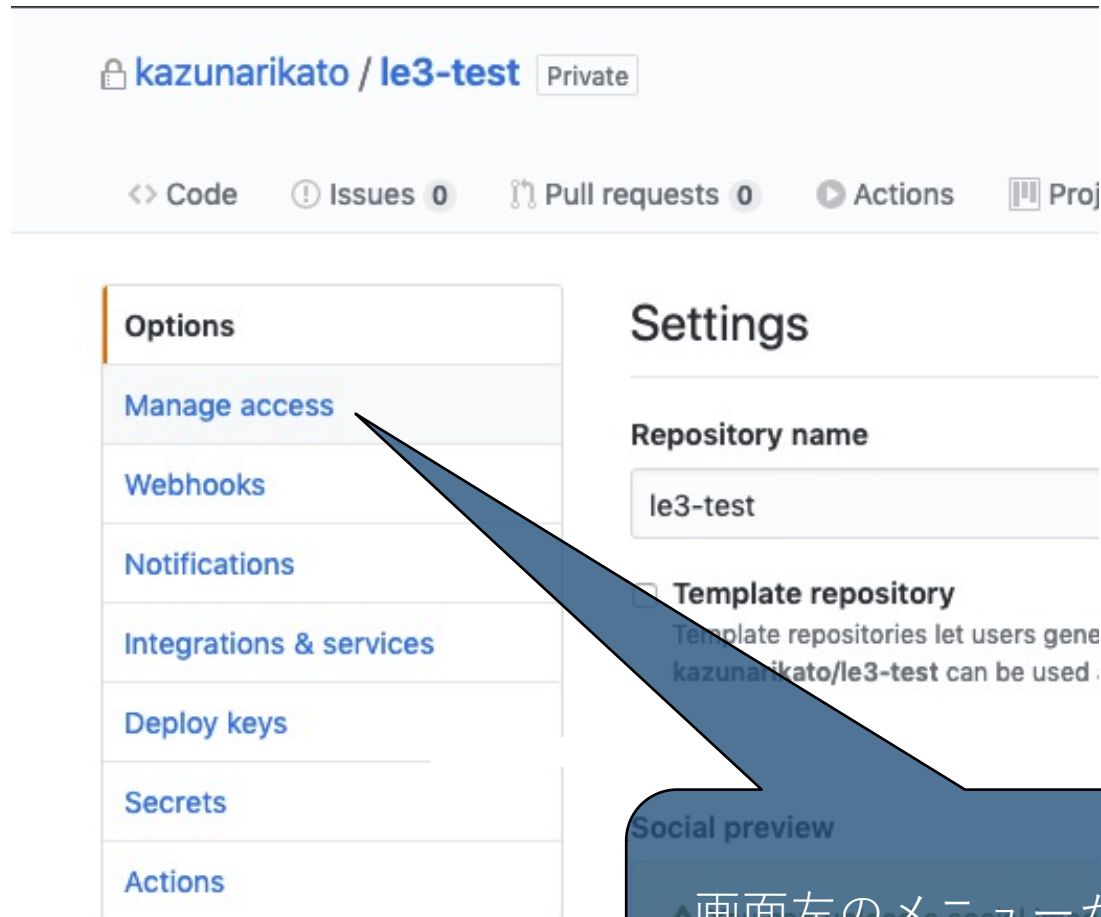
# GitHubの使い方

## collaboratorの登録



リポジトリのページで  
“Setting”をクリック

## collaboratorの登録



画面左のメニューから”Manage access”をクリック

## collaboratorの登録

### Manage access



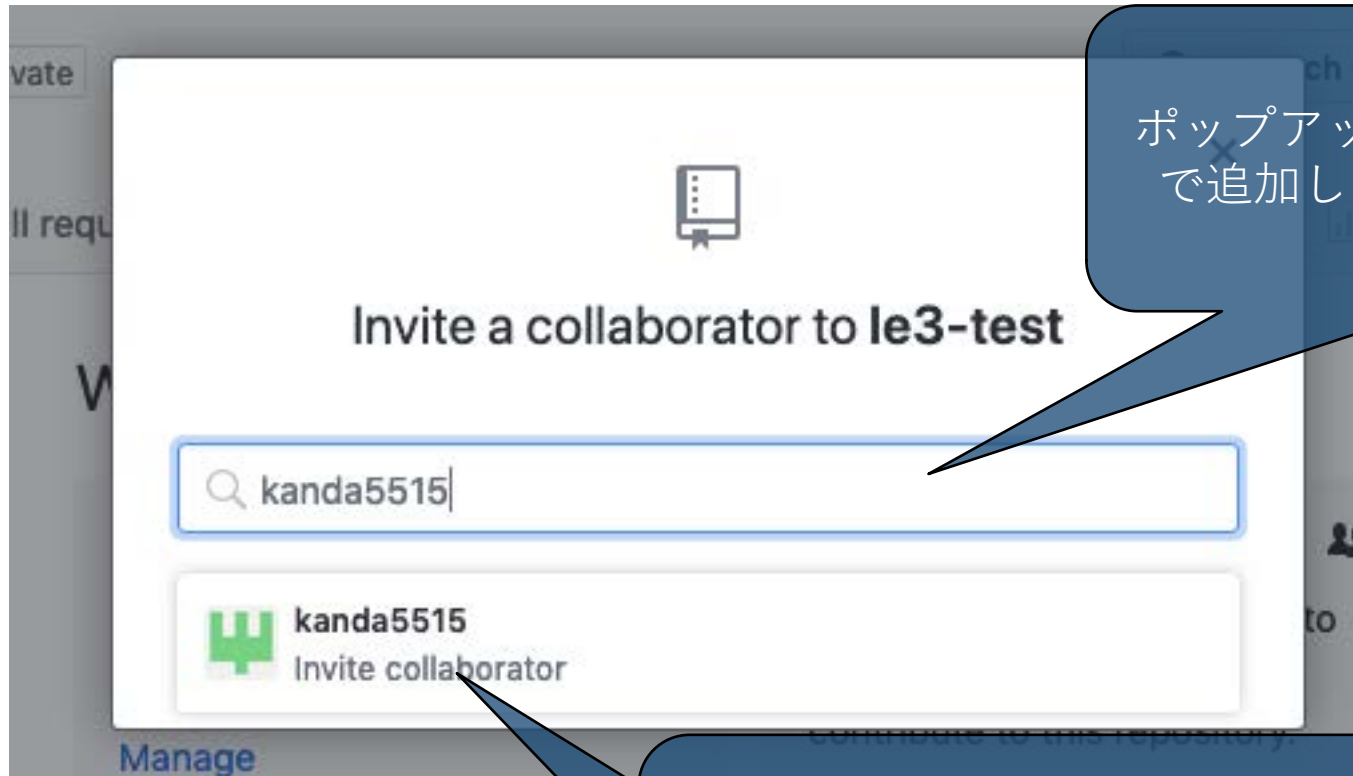
**You haven't invited any collaborators yet**

If you're using GitHub Free, you can add unlimited collaborators on **public** repositories, and up to three collaborators on private repositories owned by your **personal** account. [Learn more](#)

**Invite a collaborator**

“Invite a collaborator”ボタンをクリック

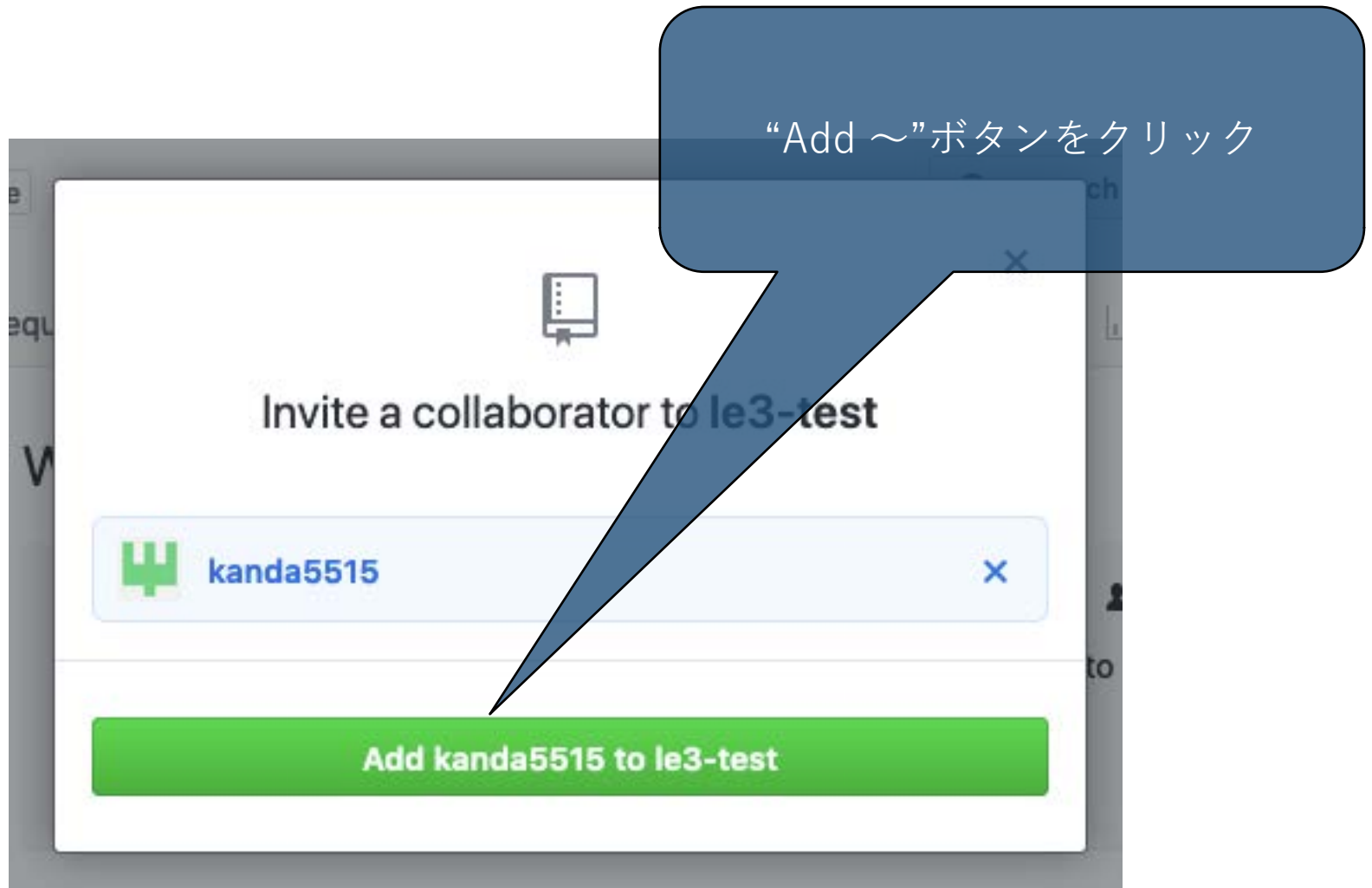
## collaboratorの登録



ポップアップウィンドウの検索窓  
で追加したいユーザー名を検索

検索されたユーザーをクリック

## collaboratorの登録



## collaboratorの登録

ユーザーが登録されていれば成功

### Manage access

Invite a collaborator

☐ Select all

Type ▾

Find a collaborator...

☐



kanda5515

Awaiting kanda5515's response

Pending Invite



Previous

Next



## コミットにtagをつける

```
git tag -a <タグ名> -m <コメント>
```

最新のコミットに対して、tagをつける。

一般的にタグ名はセマンティックバージョンングに準じてつけるが、**本実験**では課題ごとに指定されたタグ名をつける。

-a の後にタグ名、-m の後にコメントを記載

過去のコミットにタグをつけたい場合は、<コメント>の後に、コミットIDを書く。

## tagの確認

```
git tag
```

つけたタグの一覧を確認できる。

```
git show <タグ名>
```

指定したタグ名のコミットを確認できる。

## tagのpush

```
git push <エイリアス> --tag
```

git pushに--tagオプションをつけると、タグの情報がリモートリポジトリに反映される。

普通にpushしただけでは、リモートリポジトリにtagは反映されない。（逆にこのコマンドだけではブランチへのpushができない）。

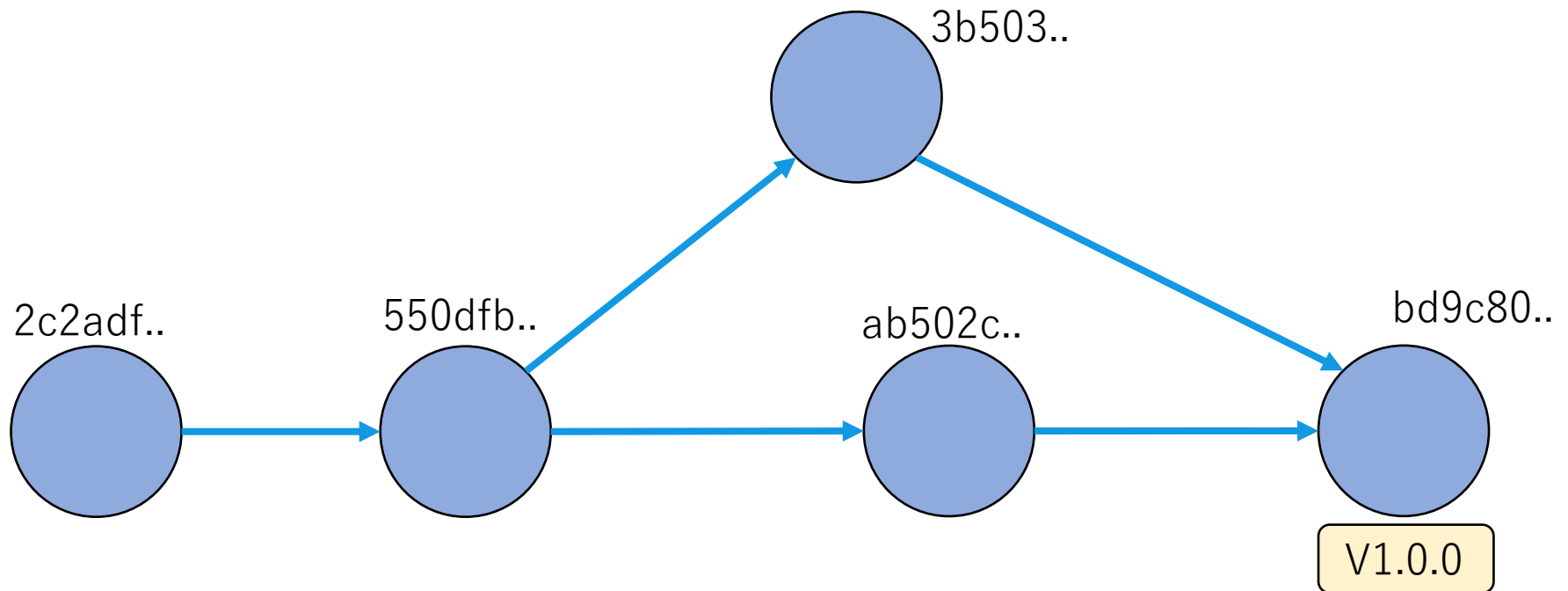
必ず“git push <エイリアス> <ブランチ>”でブランチへのpushも完了しておくこと。

# GitHubの使い方

## コミットにtagをつける

```
git tag -a v1.0.0 -m 'add v1.0.0'  
git push origin --tag
```

\*必ず “git push <エイリアス> <ブランチ>” でブランチへのpushも完了しておくこと。



# GitHubの使い方

## releaseの作成

The screenshot shows the GitHub interface for a repository named 'smnimo/le3-test'. The repository is private and has 1 branch (main) and 1 tag. The commit history shows two commits: 'Initial commit' (10 minutes ago) and 'second commit' (7 minutes ago). The 'second commit' is selected, showing the file 'test.txt'. The 'About' section on the right indicates no description, website, or topics are provided. The 'Releases' section shows 1 tag and a link to 'Create a new release'. The 'Packages' section shows no packages published. A blue callout box points to the 'Create a new release' link with the text: 'リポジトリのページで”release”内の“Create a new release”をクリック'.

smnimo / le3-test Private

Unwatch 1 Star 0 Fork 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 1 tag

Go to file Add file Code

smnimo second commit e3cb35e 7 minutes ago 2 commits

File	Commit	Time
README.md	Initial commit	10 minutes ago
test.txt	second commit	7 minutes ago

README.md

le3-test

About

No description, website, or topics provided.

Readme

Releases

1 tags

Create a new release

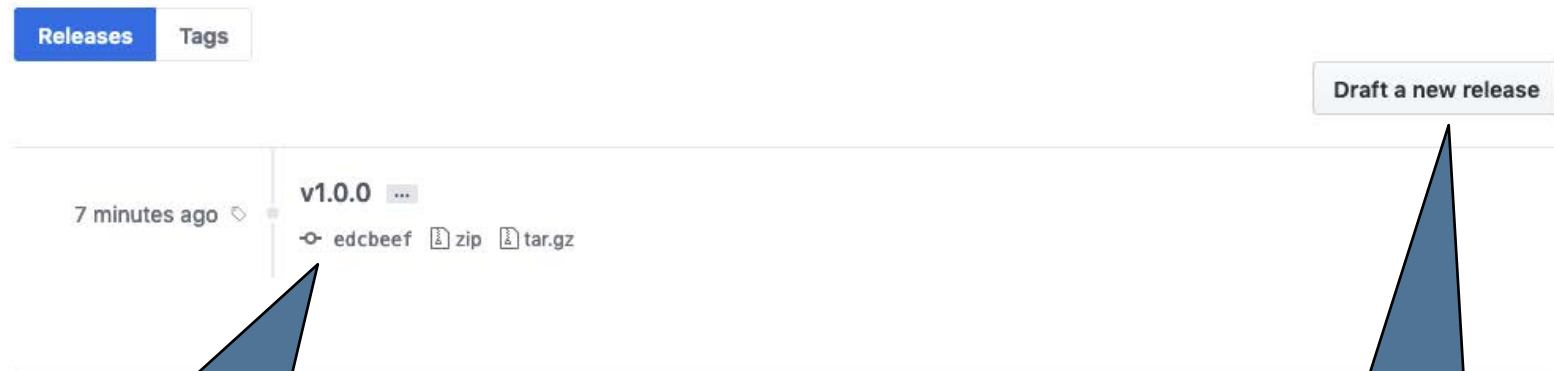
Packages

No packages published

Publish your first package

リポジトリのページで”release”内の  
“Create a new release”をクリック

## Releaseの作成



作成したタグが表示されている。

“Draft a new release”  
ボタンをクリック

## Releaseの作成

The screenshot shows the GitHub 'Create a new release' page. It includes tabs for 'Releases' and 'Tags'. A 'Tag version' dropdown is set to '@ 1.2 Target: master'. Below it is a 'Release title' input field. There are 'Write' and 'Preview' tabs, with 'Write' selected. A large text area is labeled 'Describe this release'. Below the text area is a section for attaching files, with instructions to 'Attach files by dragging & dropping, selecting or pasting them.' and a dashed box for 'Attach binaries by dropping them here or selecting them.' At the bottom, there is a checkbox for 'This is a pre-release' and two buttons: 'Publish release' (green) and 'Save draft' (grey).

Releases Tags

Tag version @ 1.2 Target: master  
Choose an existing tag, or create a new tag on publish

Release title

Write Preview

Describe this release

Attach files by dragging & dropping, selecting or pasting them.

↓ Attach binaries by dropping them here or selecting them.

☐ This is a pre-release  
We'll point out that this release is identified as non-production ready.

Publish release Save draft

Releaseのバージョン番号を入力。  
バージョンはGit tagでつけたタグ  
名から選択。タグ名は課題ごとに  
指定されている。

リリースのタイトルを入力

リリースの説明を入力

“Publish release”をクリック

# Releaseの作成

## Releaseの作成

The screenshot shows a GitHub release page for version v1.0.0. The page includes a sidebar with 'Releases' and 'Tags' tabs, a 'Latest release' badge, and a 'Compare' button. The main content area displays the version number 'v1.0.0', the user 'kazunarikato', and the description 'the first version'. Below this, there is a section for 'Assets' with two items: 'Source code (zip)' and 'Source code (tar.gz)'. Two blue callout boxes are overlaid on the image: one pointing to the release title and description, and another pointing to the assets list.

Releases Tags

Edit release Delete

Latest release

v1.0.0

edcbeef

Compare

v1.0.0

kazunarikato released this now

the first version

Assets 2

Source code (zip)

Source code (tar.gz)

Releaseが作成される

ファイルがダウンロードできる形で提供される



## ⑤GitHubに公開鍵を登録

## SSH用の公開鍵の登録

SSHで使うRSAキーの公開鍵をGitHub登録することで、ターミナルでGitHubにあるリモートリポジトリをclone/pushできるようになる。

詳しくは

<https://docs.github.com/ja/authentication/connecting-to-github-with-ssh> を参考にする。

ここではざっくり過程を見せる。

# SSH用の公開鍵の登録

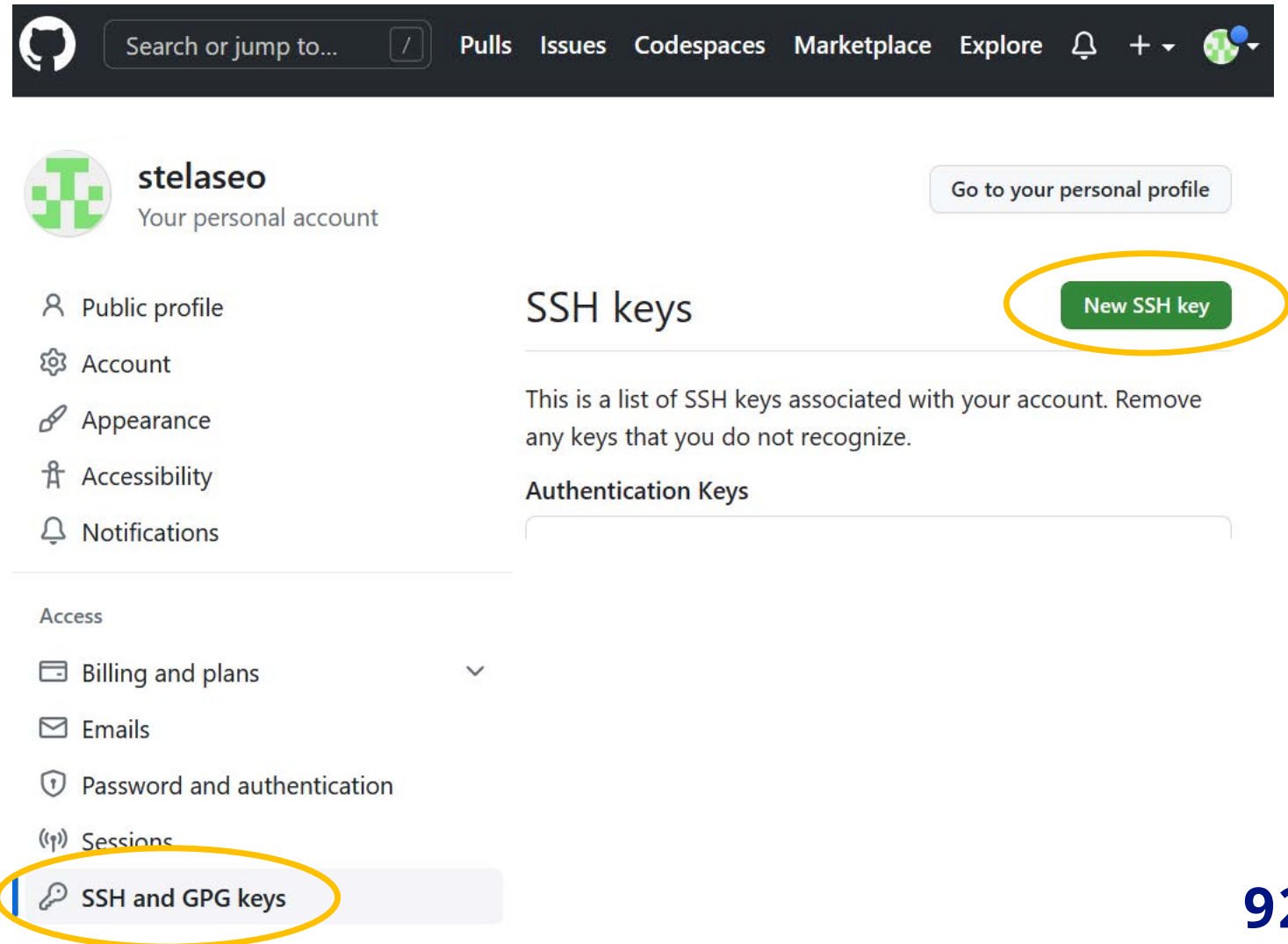
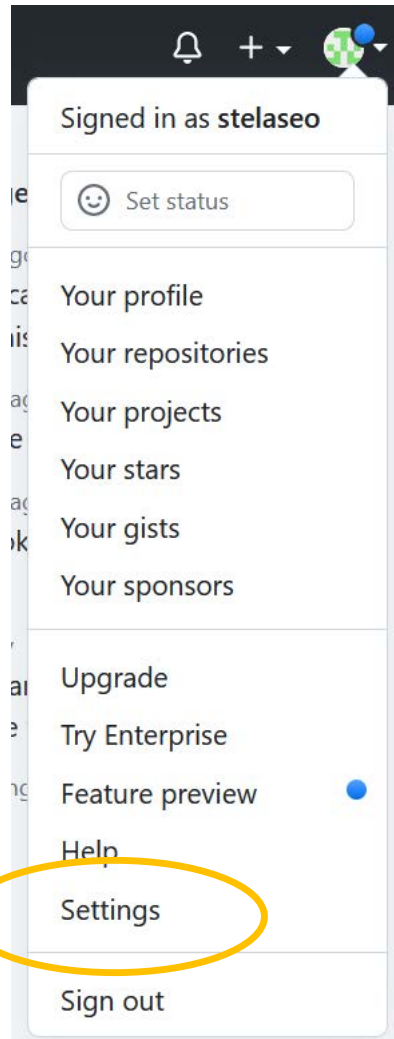
## ターミナルで「ssh-keygen」ツールを使ってキーを作成

```
stela@jupiter:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/stela/.ssh/id_rsa):
Created directory '/home/stela/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/stela/.ssh/id_rsa
Your public key has been saved in /home/stela/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:EtwevEOyzifh82x7F4+/A6MIwX4Fr0NTgDXtdl5Ztnc stela@jupiter
The key's randomart image is:
+---[RSA 3072]-----+
|      o+o          |
|    ..o..o    o    |
|  .+ ==      +.    |
|    o*oo= . o.E    |
|  .+oS= o .  o    |
|  +oo+.  =        |
|    *o.o . *      |
|      *o o o o    |
|      .+o . .oo   |
+-----[SHA256]-----+
stela@jupiter:~$
```

# SSH用の公開鍵の登録

## 公開鍵をGitHubに登録する

Settings → SSH and GPG keys → New SSH Key



# SSH用の公開鍵の登録

## 公開鍵をGitHubに登録する

```
stela@jupiter:~$ cat ~/.ssh/id_rsa.pub  
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCoA9Wqej0cpRrdWWI58cJoD6EXy  
d0m70k5nQPX36FGougS1xj2h2dRMvZw24D7hgXJ8enwWZ5FJSpJ1Kf2jE5VrVX4J2  
OzLNPEfVgVtaq3kM5UNguFzVKyYhithMyuKZiWDuAVinMSm4xk9Am1RglSHVVHNGG  
P/B03L5vSpuE/AzwLApJ3s0GE16wb/pxtEya7/vU2R1fKugsiT2KqXsi2qB4zUr  
Cf/+EJSIKUh3Z+/0F+UeThP...  
n4CAF54ok9ZpBkDg...  
eLtroouPUsEGZXM...  
9dYxUXqIGAqA7c4...  
IM/YAs8RGZAlmSP...
```

id\_rsa.pubの中身を  
コピー & ペースト

SSH key / Add new

Title

Key type

Authentication Key

Key

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

Add SSH key

## ターミナルでgit repositoryをclone

```
stela@jupiter:~$ git clone git@github.com:kuis-isle3hw/2023-intro-stelaseo.git
Cloning into '2023-intro-stelaseo'...
Enter passphrase for key '/home/stela/.ssh/id_rsa':
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 1 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
stela@jupiter:~$
```

## ⑥GitHub Classroomの利用

## GitHub Classroom

- **GitHub Classroom は GitHub を講義で利用するためのプラットフォーム**
  - － 個人用，グループ用の課題の作成が可能
  - － 各学生の個人GitHub アカウントで作成したリポジトリとリンクさせることで課題用のリポジトリを一括管理
  - － 本実験ではこのGitHub Classroom を利用し，  
課題は各リポジトリでRelease を作成することで提出



## GitHub Classroomの導入

- 招待リンクはPandA内のお知らせを参照
  - ー リポジトリは2種
    - 導入課題（個人課題）：2023-Intro
    - プロセッサ課題（グループ課題）：2023-Simple
- まずGitHub Classroom と GitHub の連携について authorizationを求められるので承認
  - ー GitHub アカウントを持っていない人はここで作成を

## 導入課題

- 導入課題は各学生 1 人に対して 1 つのリポジトリ



Join the classroom:

kuis-isle3hw-classroom-2021

To join the GitHub Classroom for this course, please select yourself from the list below to associate your GitHub account with your school's identifier (i.e., your name, ID, or email).

Can't find your name? Skip to the next step

Identifiers	
1029296773	>
1029299434	>
1029302951	>
1029304955	>
1029307671	>
1029309236	>
1029310012	>
1029310040	>

Identifiersの中から  
自分の学生番号を探し出してクリック  
→ Accept でリポジトリが作成される  
2023-intro-XXX (Github アカウント名)

※自分の学生番号が見つからない  
場合はスタッフまで連絡を

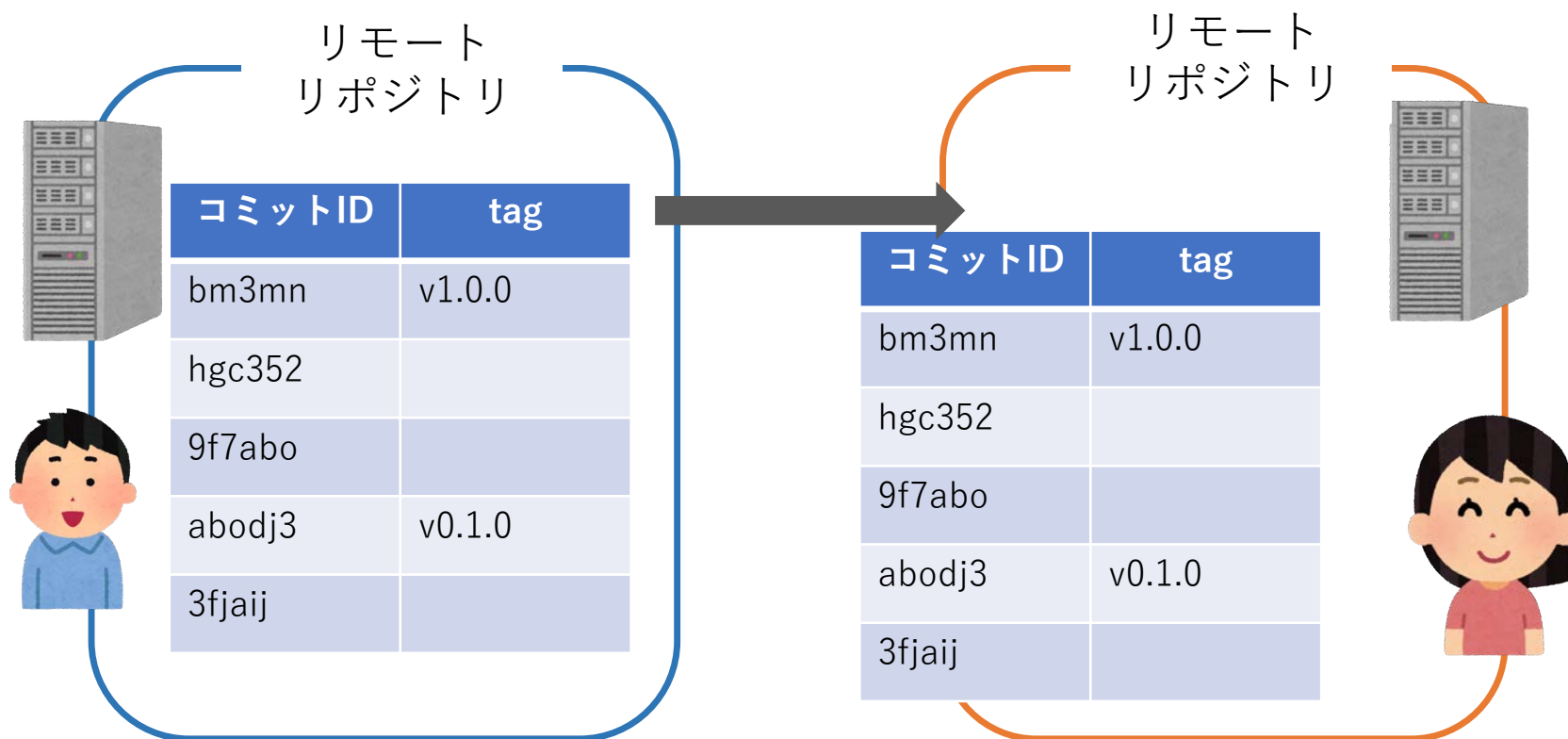
## プロセッサ課題

- プロセッサ課題は基本的に学生 2 人に 1 つのリポジトリ
  - 3 人グループもある
- グループはこちらで割り振って連絡しますので後ほどPandAを確認してください.
- 1 人の学生がチームの作成（連動してリポジトリの作成）
  - もう 1 人の学生は作成されたチームを探して参加
    - チーム名はグループ分けの番号と対応して「teamXX」（例：team01）としてください.
      - リポジトリは 2023-simple-teamXX が作成されます.

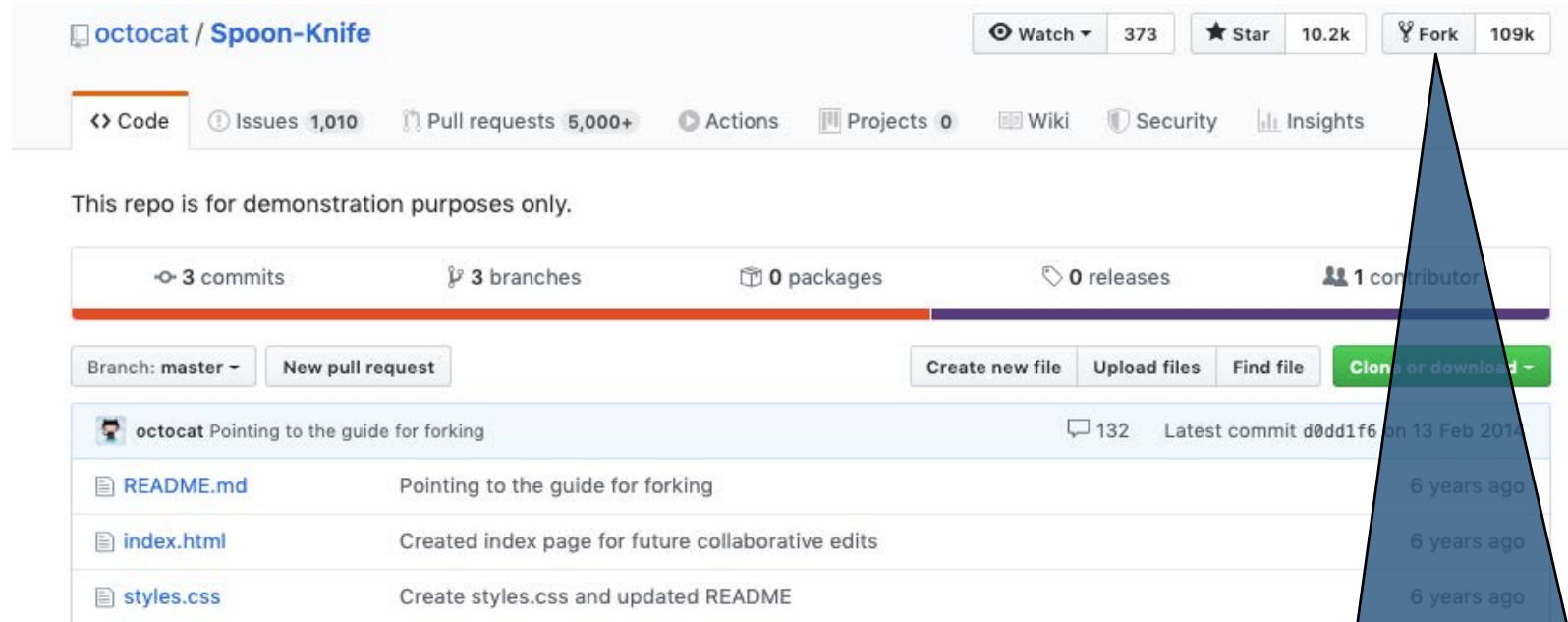
## ⑥ その他のGitHub機能

## fork

- 他の人のリモートリポジトリを自分のリモートリポジトリにコピーする。



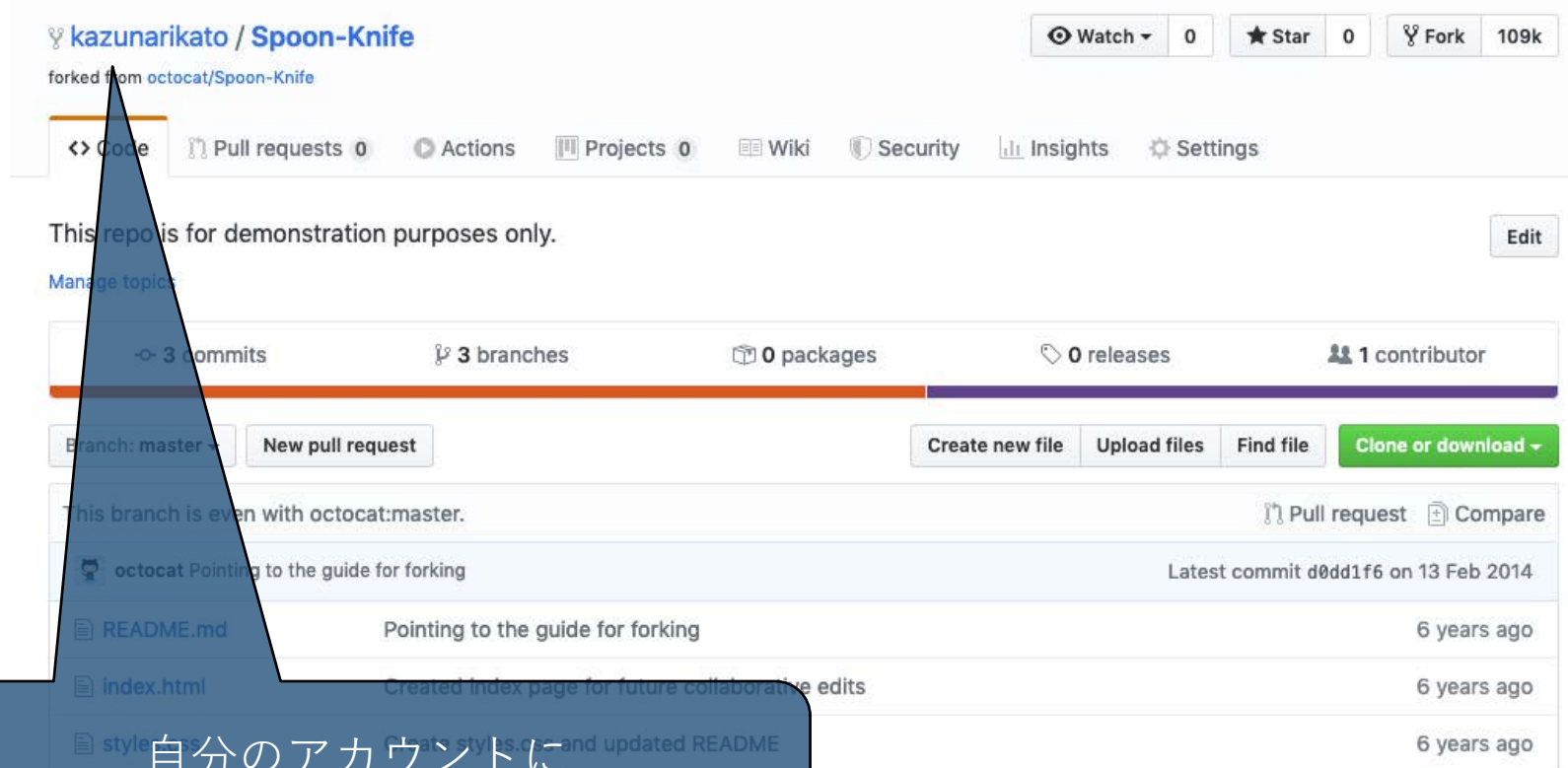
## forkの方法



コピーしたいリポジトリページの  
リポジトリ名の右側にある  
"Fork"をクリック

# その他の機能

## forkの方法



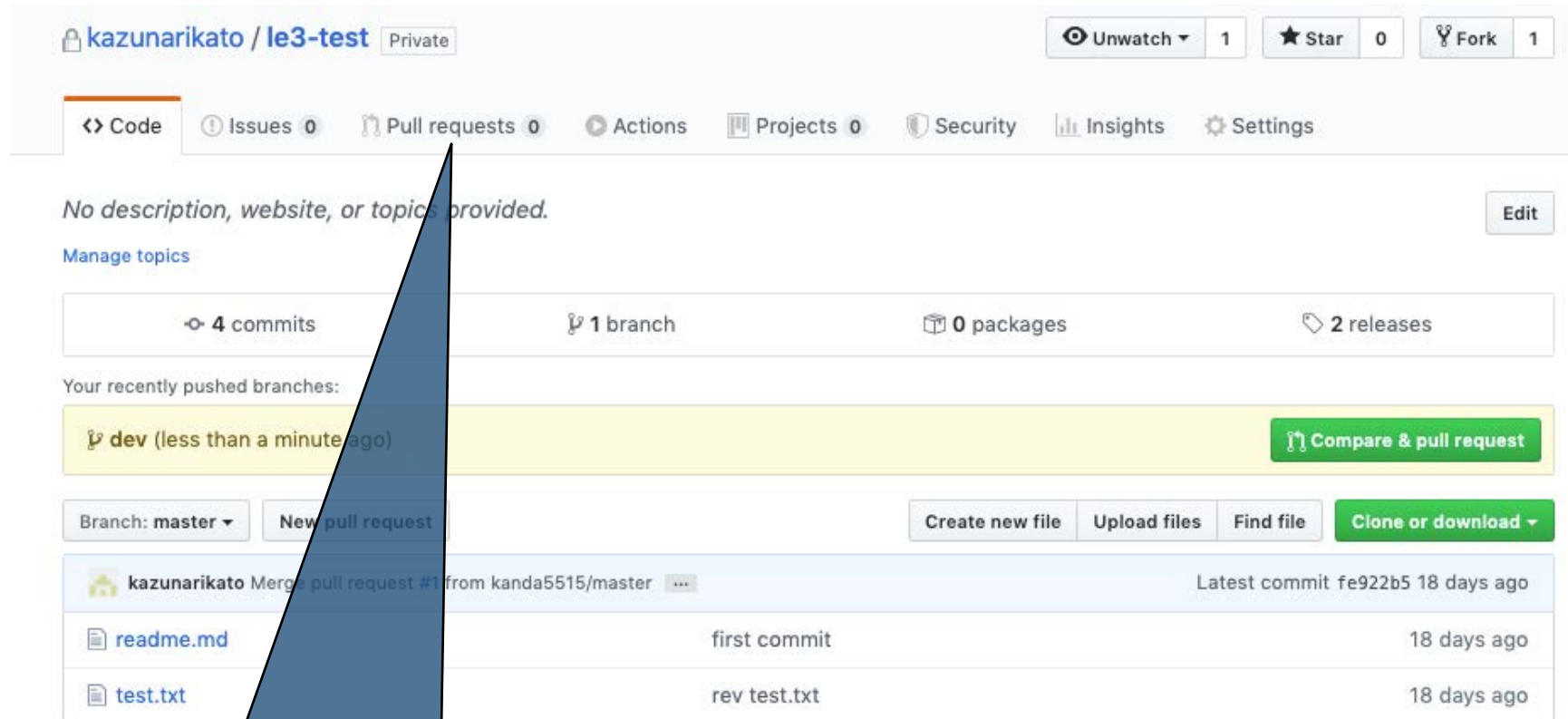
自分のアカウントに  
コピーしたリポジトリができる

### Pull request

- 新機能を追加し、ファイルをcommitした時、いきなりリモートリポジトリのmainにpushすると、そこにバグが含まれていた時に問題となる。
- Pull requestを使うと、pushする前にローカルリポジトリでの変更内容をリモートリポジトリを共有している他の人に通知することができる。
- 変更内容を確認後、リモートリポジトリ上でmergeすることで、リモートリポジトリが更新される。
- コード・レビューがしやすくなる。



## Pull requestの作成



リポジトリページの  
"Pull requests"をクリック

## Pull requestの作成

The screenshot shows the GitHub web interface for the repository 'kazunarikato / le3-test'. The 'Pull requests' tab is selected, showing 0 open pull requests. A blue callout box with a pointer indicates the 'New pull request' button. The text inside the box is: "New pull requests"ボタンをクリック.

There aren't any open pull requests.

You could search [all of GitHub](#) or try an [advanced search](#).

## Pull requestの作成

### Compare changes

Compare changes across branches, commits, tags, and more below. If you need to, you can also [compare across forks](#).

base: master ← compare: master

Choose a head ref

Find a branch

Branches Tags

✓ master default

dev

Create pull request

mergeする\_BRANCHを決める。  
“base:”にmerge先\_BRANCH  
“compare:”にmerge元\_BRANCHを指定  
する。

### Compare and review just about anything

Branches, tags, commit ranges, and time ranges. In the same repository and across forks.

EXAMPLE COMPARISONS	
dev	1 minute ago
master@{1day}...master	24 hours ago

この例では“master”←“dev”のmergeを  
リクエストするので、compare:に“dev”  
を選択する。

## Pull requestの作成

### Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

base: master ← compare: dev ✓ Able to merge. These branches can be automatically merged.

Create pull request

1 commit 1 file changed 0 commit comments 1 contributor

Commits on Apr 10, 2020

kazunarikato mod readme.md 1a786eb

Showing 1 changed file with 1 addition and 0 deletions.

Unified Split

1 readme.md

```
@@ -1,2 @@
1 1    le3-testle3-testle3-test
2  + devで編集
```

## Pull requestの作成

### Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

base: master ← compare: dev ✓ Able to merge. These branches can be automatically merged.

mod readme.md

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Create pull request

Reviewers  
No reviews

Assignees  
No one—assign yourself

Labels  
None yet

Projects  
None yet

Milestone  
No milestone

Pull requestの名前を記入する。

Pull requestの内容を記入する

“Create pull request”ボタンをクリックする。

## Pull requestの作成

The screenshot shows a GitHub Pull Request titled "mod readme.md #2". At the top, it says "kazunarikato wants to merge 1 commit into master from dev". Below this, there are tabs for Conversation (0), Commits (1), Checks (0), and Files changed (1). A comment from "kazunarikato" is visible, stating "No description provided." and showing a file change for "mod readme.md". A green box highlights the status section, which includes a warning about continuous integration not being set up, a green checkmark indicating "This branch has no conflicts with the base branch", and a green "Merge pull request" button. A blue callout box with white text "Pull requestが作成された。" (Pull request has been created.) points to the "Merge pull request" button. The right sidebar shows settings for Reviewers, Assignees, Labels, Projects, Milestone, and Linked issues. At the bottom, there is a "Write" tab with a "Preview" button and a "Comment" button.

mod readme.md #2

Open kazunarikato wants to merge 1 commit into master from dev

Conversation 0 Commits 1 Checks 0 Files changed 1 +1 -0

kazunarikato commented now

No description provided.

mod readme.md 1a786eb

Add more commits by pushing to the dev branch on kazunarikato/le3-test.

Continuous integration has not been set up  
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

✓ This branch has no conflicts with the base branch  
Merging can be performed automatically.

Merge pull request You can also open this in GitHub Desktop or view command line instructions.

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Close pull request Comment

Remember, contributions to this repository should follow our GitHub Community Guidelines

Reviewers  
No reviews

Assignees  
No one—assign yourself

Labels  
None yet

Projects  
None yet

Milestone  
No milestone

Linked issues  
Successfully merging this pull request may close these issues.  
None yet

1 participant

## Pull requestの作成

### mod readme.md #2

[Edit](#)[Open](#) kazunarikato wants to merge 1 commit into `master` from `dev`[Conversation](#) 0[Commits](#) 1[Checks](#) 0[Files changed](#) 1

+1 -0

Changes from all commits ▾ File filter... ▾ Jump to... ▾ ⚙ ▾

0 / 1 files viewed ⓘ

[Review changes ▾](#)

▼ 1 ■■■■■ readme.md 📄

[<>](#) [📄](#) [Viewed](#) ...

... @@ -1,2 @@

1 1 le3-testle3-testle3-test

2 + devで編集

💡 ProTip! Use `n` and `p` to navigate between commits in a pull request.

”File changed”をクリックすると、ファイルの変更箇所を確認できる。

## Pull requestの作成

The screenshot shows a GitHub Pull Request titled "mod readme.md #2" created by "kazunarikato". The interface includes a header with "Open", "Conversation 0", "Commits 1", "Checks 0", and "Files changed 1". A comment from "kazunarikato" is visible. A merge box contains the title "Merge pull request #2 from kazunarikato/dev", the file "mod readme.md", and the email "kato.kazunari5515@gmail.com". A "Confirm merge" button is highlighted. The right sidebar shows "Reviewers", "Assignees", "Labels", "Projects", "Milestone", and "Linked issues".

mod readme.md #2

Open kazunarikato wants to merge 1 commit into master from dev

Conversation 0 Commits 1 Checks 0 Files changed 1 +1 -0

kazunarikato commented 2 minutes ago

No description provided.

mod readme.md

Add more commits by pushing to the dev branch on kazunarikato/re3-test.

Merge pull request #2 from kazunarikato/dev

mod readme.md

kato.kazunari5515@gmail.com

Choose which email address to associate with this commit

Confirm merge Cancel

Write Preview

Leave a comment

Reviewers

No reviews

Assignees

None yet

Labels

None yet

Projects

None yet

Milestone

No milestone

Linked issues

Successfully merging this pull request may close these issues.

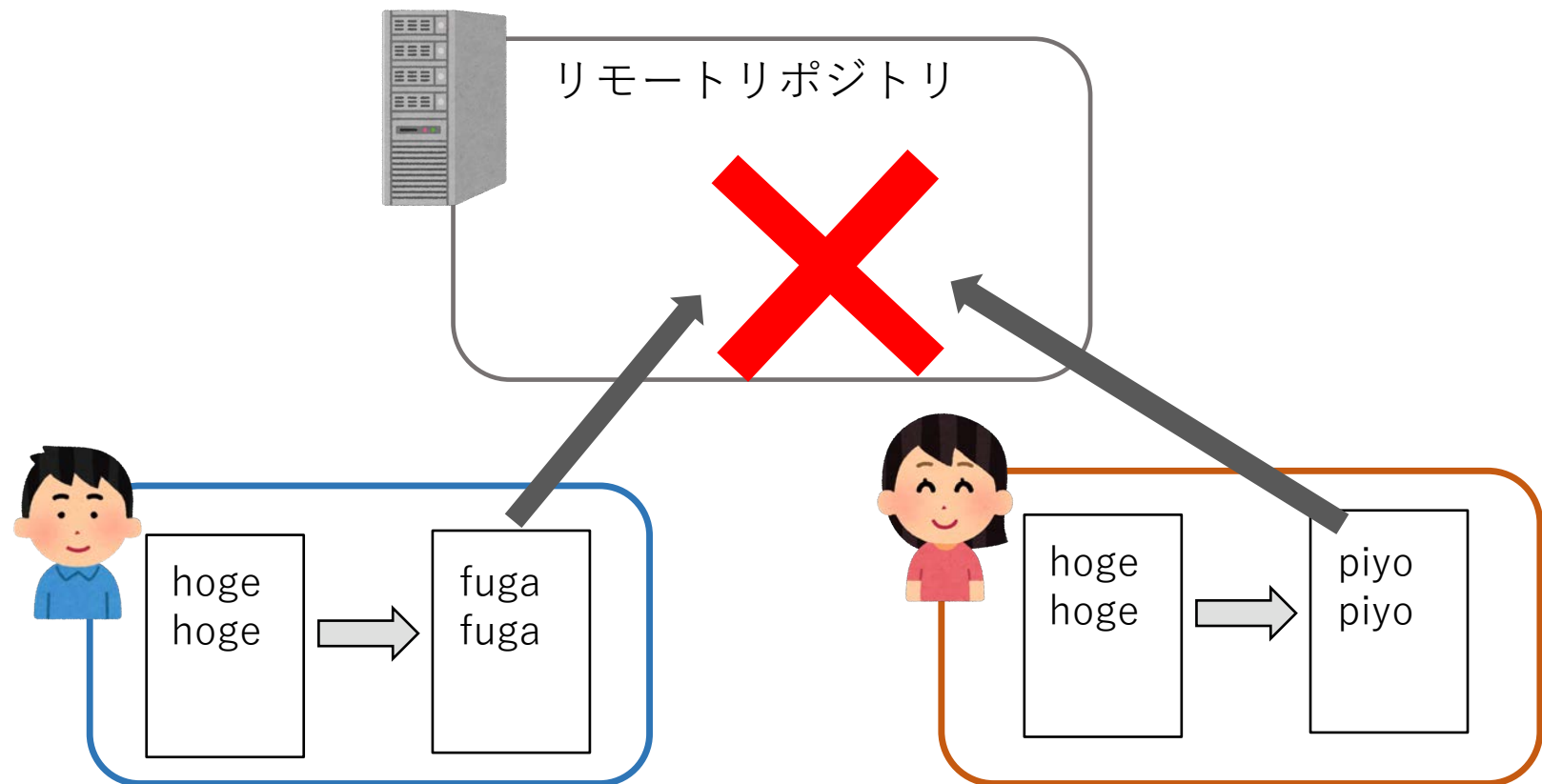
“Conversation”をクリック。

“Confirm merge”ボタンをクリックすると、mergeされる。



## コンフリクト

- 他の人が編集したファイルに重複した編集をした状態でpushやpullをするとコンフリクトが起きる。（1人で作業する場合でもmergeの時に起きる可能性あり）



## コンフリクト対策

- コンフリクトを未然に防ぐ
  - .gitignoreを設定して、ログファイルなど不要なファイルはGitの管理対象外にする。  
(<https://github.com/github/gitignore>)
  - プルリクエストを使う。
  - 二人で1つのファイルを同時に編集しないようにする。

## コンフリクト対策

- コンフリクトしてしまったら
  - git status コマンドでコンフリクトしているファイルを確認。
  - 直接ファイルを開く。編集が重複している部分が“<<<<HEAD”、“=====”、“>>>>”でハイライトされているので、不要部分を削除する。
  - 編集したファイルをステージング、コミットする。

- サルでもわかるGit入門  
<http://www.backlog.jp/git-guide/>
- Pro Git book（日本語版）  
<https://git-scm.com/book/ja/v2>
- Git Cheat Sheet  
<https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf>
- GitHubヘルプドキュメント  
<https://help.github.com/ja/github>