セキュリティ・キャンプ 全国大会 2023 ジュニア開発ゼミ

Git実践基礎

2023-08-07

高山 尚樹 (Naoki Takayama)

事前学習資料

バージョン管理の基本概念

事前学習資料

- ## ファイルを間違えて編集してしまう

  皆さんは意図せずファイルを編集してしまったことはありますか？

  ファイルを間違えて別のファイルの内容に上書きしてしまったり、ファイルそのものを削除してしまったり、人間は大なり小なりミスをします。

バージョン管理の基本概念

事前学習資料

● 復旧するための方法

では、ファイルを元の (編集する前の) 状態に戻す方法として何が考えられるでしょうか。次のページに進む前に一度考えてみましょう。

事前課題１

あなたが考えるファイルを元の状態に戻す方法について説明してください。

バージョン管理の基本概念

事前学習資料

- ## バックアップ

  ファイルを元の状態に戻せる、最も手軽な方法はバックアップです。
  そのファイルのコピーを事前に作成して、どこかに保存しておけば、必要に応じて編集前のファイルに戻すことができる。
  しかしバックアップには幾つか欠点があります。

バージョン管理の基本概念

事前学習資料

- **定期的にファイルを手動でコピー・保存する必要がある**

  頻繁にファイルをコピーし、それを間違えて編集する心配のない安全な場所に保存する作業が必要になります。

- **情報の管理が面倒**

  バックアップの取得日時などを正確に記録しておく必要があります。

バージョン管理の基本概念

事前学習資料

- ## バージョン管理システム

  半自動的にファイルの変更履歴を作成し保存するシステム。変更内容だけでなく、変更日時などの情報も併せて保存される。

  → 本講義の主題であるGitもバージョン管理システム

変更履歴: 5W1H

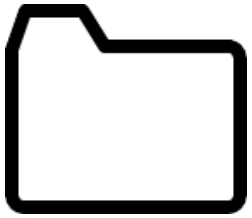
事前学習資料

- **When, Where, Who, What, Why, How (5W1H)**

  皆さんは学校などで “5W1H” という言葉を聞いたことはありますか？
  これは When (いつ) , Where (どこで) , Who (誰が) , What (何を) , Why (なぜ) , How (どうやって) という六つの英単語の頭文字をもとにした用語です。
  バージョン管理システムが管理する変更履歴にはファイルの変更に関する5W1Hへの答えが蓄積されています。

変更履歴: 5W1H

事前学習資料



リポジトリ

- 2023年5月3日にファイルAの1~10行目を太郎さんが新機能の追加のために編集した
- 2023年7月3日にファイルBの23~30行目を花子さんがバグの修正のために編集した
- 2023年7月9日にファイルCの41~92行目を次郎さんが新機能の追加のために編集した

・・・

用語解説: リポジトリとコミット

事前学習資料

- ## リポジトリ

  Git (バージョン管理システム) が変更履歴を保存する場所。

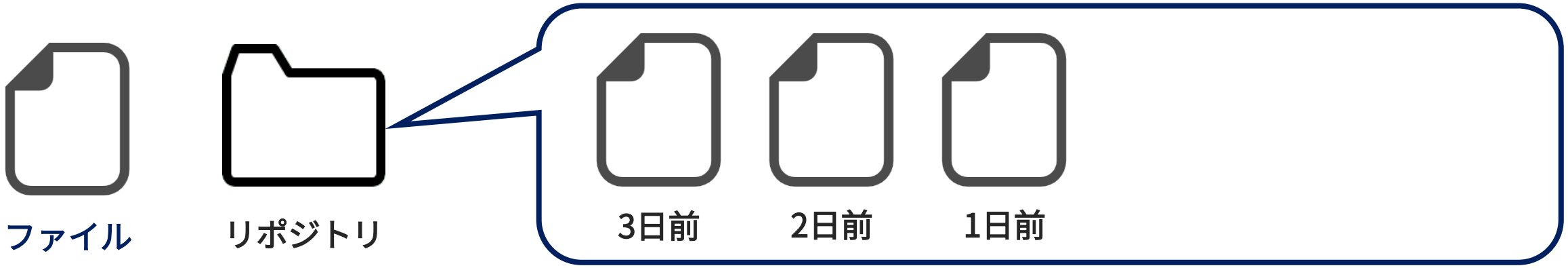
- ## コミット

  ファイルの変更内容、変更日時などをリポジトリに記録する (変更履歴を追加 / 更新する) 操作。

図解: リポジトリとコミット

事前学習資料

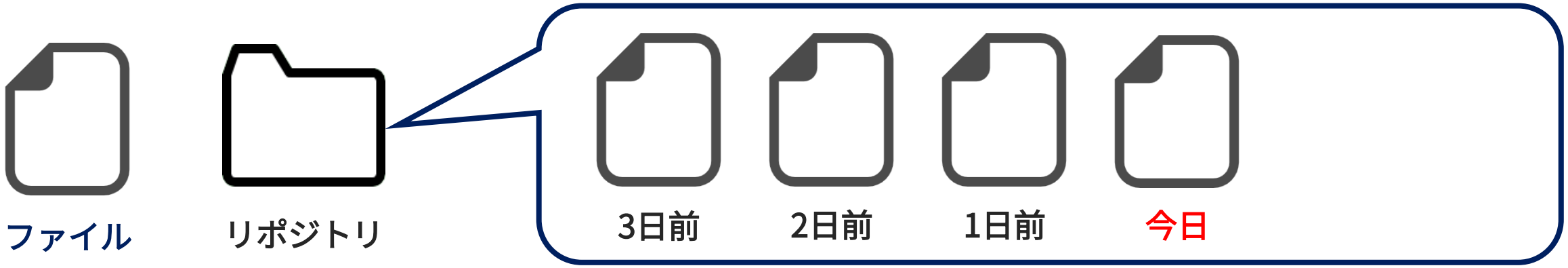
- バージョン管理システムでファイルを管理している



図解: リポジトリとコミット

事前学習資料

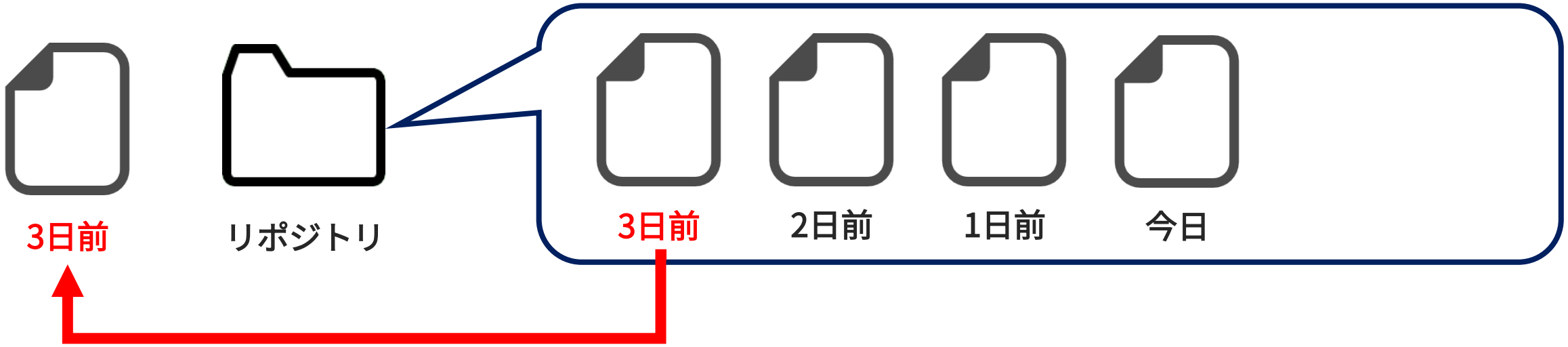
- コミットすると、リポジトリに記録される



図解: リポジトリとコミット

事前学習資料

- いつでも以前のコミットの時点のファイルに戻せる



リポジトリの種類

事前学習資料

- **ローカルリポジトリ**

  手元のコンピュータ上の (あなた専用の) リポジトリ。

- **リモートリポジトリ**

  サーバー上の (複数人で使うための) リポジトリ。

用語解説: プッシュとプル

事前学習資料

- **プッシュ**

  ローカルリポジトリに記録された情報をリモートリポジトリに記録する操作。

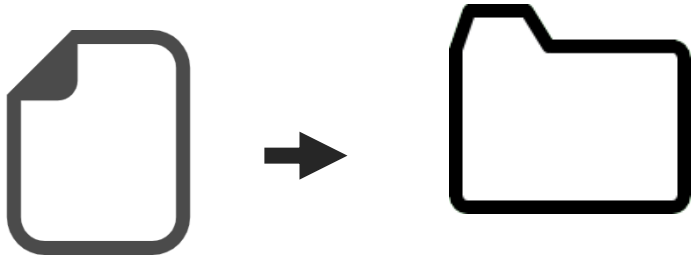
- **プル**

  リモートリポジトリに (他のユーザーによって) 記録された情報をローカルリポジトリに記録する操作。

図解: リモートリポジトリ

事前学習資料

- ローカルリポジトリにコミットして変更内容を記録する

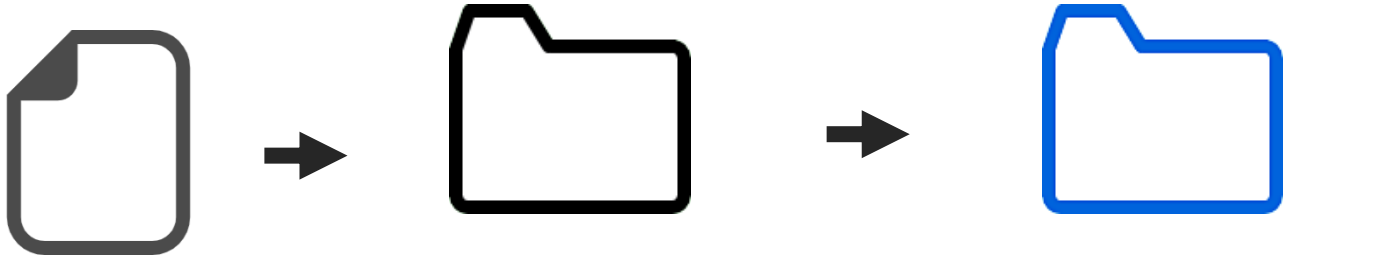


ローカルリポジトリ

図解: リモートリポジトリ

事前学習資料

- 記録された情報をリモートリポジトリにプッシュする



図解: リモートリポジトリ

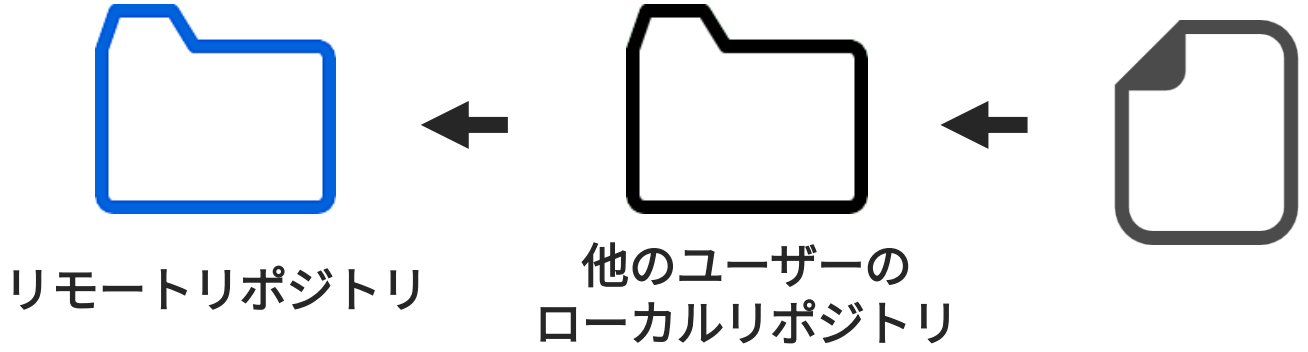
事前学習資料

- 他のユーザーが自身のローカルリポジトリにプルして変更内容 (編集されたファイル) を取得する

図解: リモートリポジトリ

事前学習資料

- 他のユーザーがファイルを編集して自身のローカルリポジトリにコミットし、リモートリポジトリにプッシュする



図解: リモートリポジトリ

事前学習資料

- ローカルリポジトリにプルして他のユーザーの変更内容 (変更されたファイル) を取得する

図解: リモートリポジトリ

事前学習資料

- これを繰り返すことで**チーム開発**が可能になる

分散型バージョン管理システム

事前学習資料

- ## 分散型バージョン管理システム

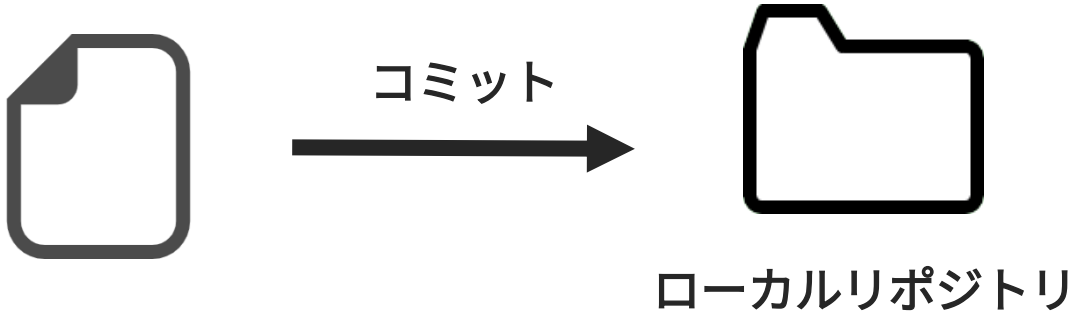
  ローカルリポジトリとリモートリポジトリの2種類のリポジトリを運用できるバージョン管理システム。

図解: 他のバージョン管理システム

事前学習資料

- ## ローカル型バージョン管理システム

  ローカルリポジトリのみ運用できるバージョン管理システム。

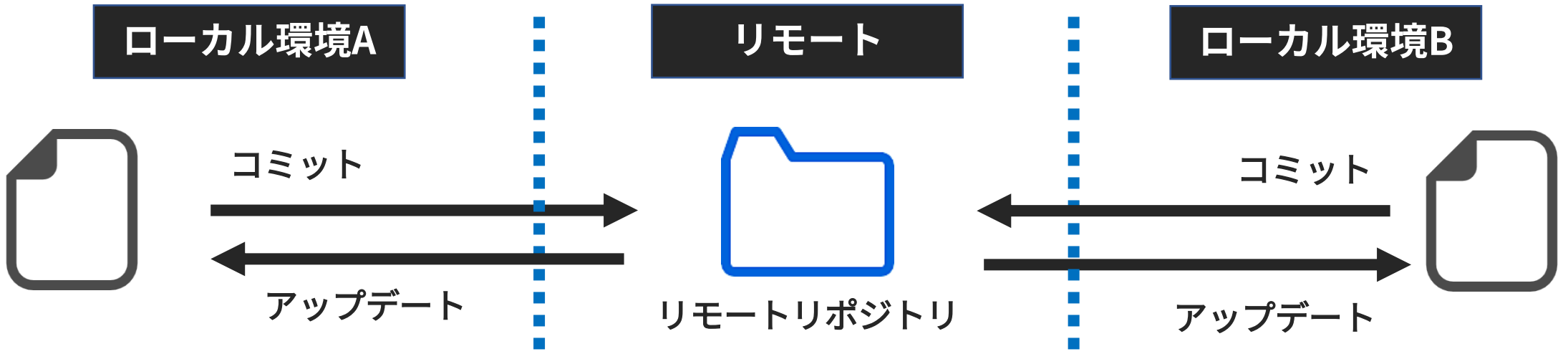


図解: 他のバージョン管理システム

事前学習資料

- 集中型バージョン管理システム

  リモートリポジトリのみ運用できるバージョン管理システム。



各バージョン管理システムの例

事前学習資料

- **ローカル型バージョン管理システム**

  Revision Control System (RCS) , Source Code Control System (SCCS) など

- **集中型バージョン管理システム**

  Concurrent Versions System (CVS) , Subversion など

- **分散型バージョン管理システム**

  Git , Mercurial など

他のバージョン管理システム

事前学習資料

事前課題２

分散型バージョン管理システムが他のバージョン管理システムと比較して優れていると思う点を答えてください。

事前課題

事前学習資料

事前課題１ (p.6)

あなたが考えるファイルを元の状態に戻す方法について説明してください。

事前課題２ (p.22)

分散型バージョン管理システムが他のバージョン管理システムと比較して優れていると思う点を答えてください。

事前課題の提出先・締切

事前学習資料

- ## 提出先

  以下のGoogle Formにお名前・メールアドレスとともに提出してください。

  https://forms.gle/YzFdXeJG1PLFMxY88

- ## 提出締切

  提出締切は2023年7月31日 正午とします。締切以降の提出は受け付けません。

イントロダクション

はじめに

- **講義中の質問について**

  何かわからないこと / 困ったことがあれば積極的に質問してください。

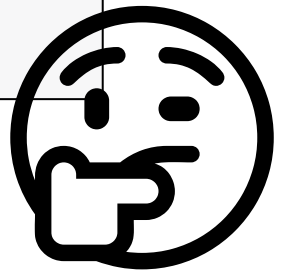
- **講義資料の取り扱いについて**

  講義資料や演習で使用するデータ等を第三者に譲渡・共有しないでください。

Gitとは (by ChatGPT)

Gitとは何ですか？

Gitは、分散型バージョン管理システムです。コンピュータプログラムのソースコードの変更履歴を管理することができます。Gitは、複数の人々が同時に同じプロジェクトに取り組む場合に、プロジェクトの変更の衝突を解決するために非常に便利です。Gitは、GitHub、GitLab、Bitbucketなどの多くのホスティングサイトで使用されています。Gitは、オープンソースであり、無料で利用できます。



本講義で目指すところ

✅ Gitとは何なのか大まかに説明できるようになる

✅ Gitを用いたチーム開発を体験する

❌ Gitについて完全に理解する

❌ プログラミングについて学習する

Git基礎

Gitの歴史と成り立ち

Git基礎

- **Linuxカーネルのソースコードを管理するために開発された**

  Gitとは2005年にLinus Torvaldsによって開発されたバージョン管理システムです。様々な操作を高速で行えること、開発者フレンドリーな様々な機能が実装されていることから、バージョン管理システムにおける今日のデファクトスタンダードとなっています。

＊ デファクトスタンダード: 事実上の標準

活用事例

Git基礎

- **およそ90%の開発者がGitを使用している**

  Stack Overflowが今年1月に公開した記事では、およそ90%のソフトウェア開発者がGitを日常的に使用しているというサーベイの結果が報告されています。

https://stackoverflow.blog/2023/01/09/beyond-git-the-other-version-control-systems-developers-use/

ステージング

Git基礎

- **コミットする準備ができていることを示す操作**

  編集したファイルをすべてコミットに含めてしまうと不便です。

  (一時ファイルや編集途中のファイルなどの情報も記録されてしまうため。)

  Gitではステージングと呼ばれる操作をファイルに対して実行することで、初めてファイルがコミットの対象として認識されます。

ブランチ

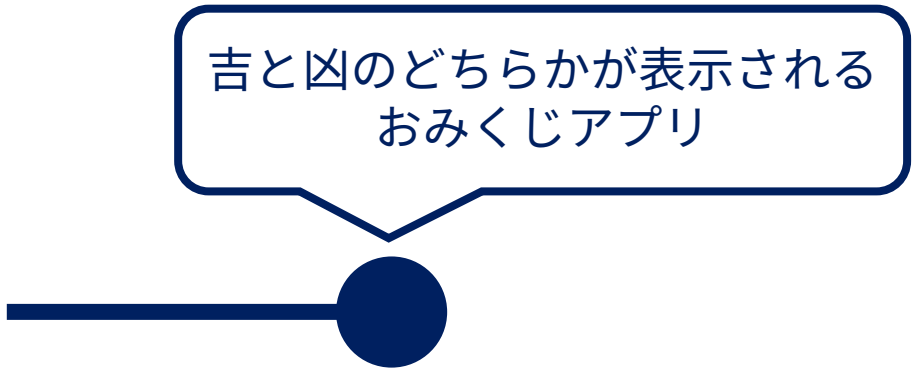
Git基礎

- **Gitの代表的な機能の1つ「ブランチ」**

  元となるデータから分かれた新しいデータのこと。新しいデータを編集しても元のデータには影響を与えない。また、マージすることで、新しいデータで編集した内容を元のデータに取り込むこともできる。

ブランチの例: おみくじアプリ

Git基礎

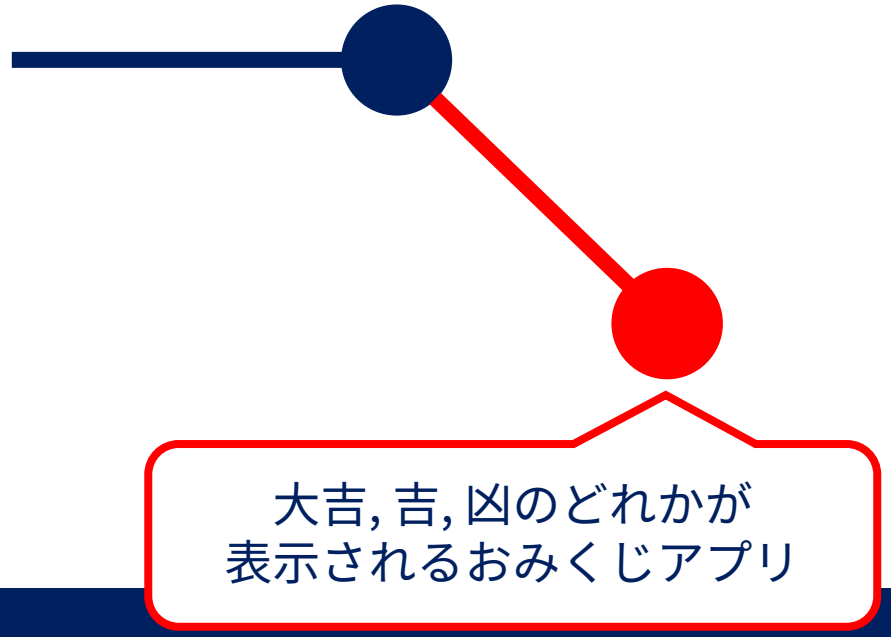


ブランチの例: おみくじアプリ

Git基礎

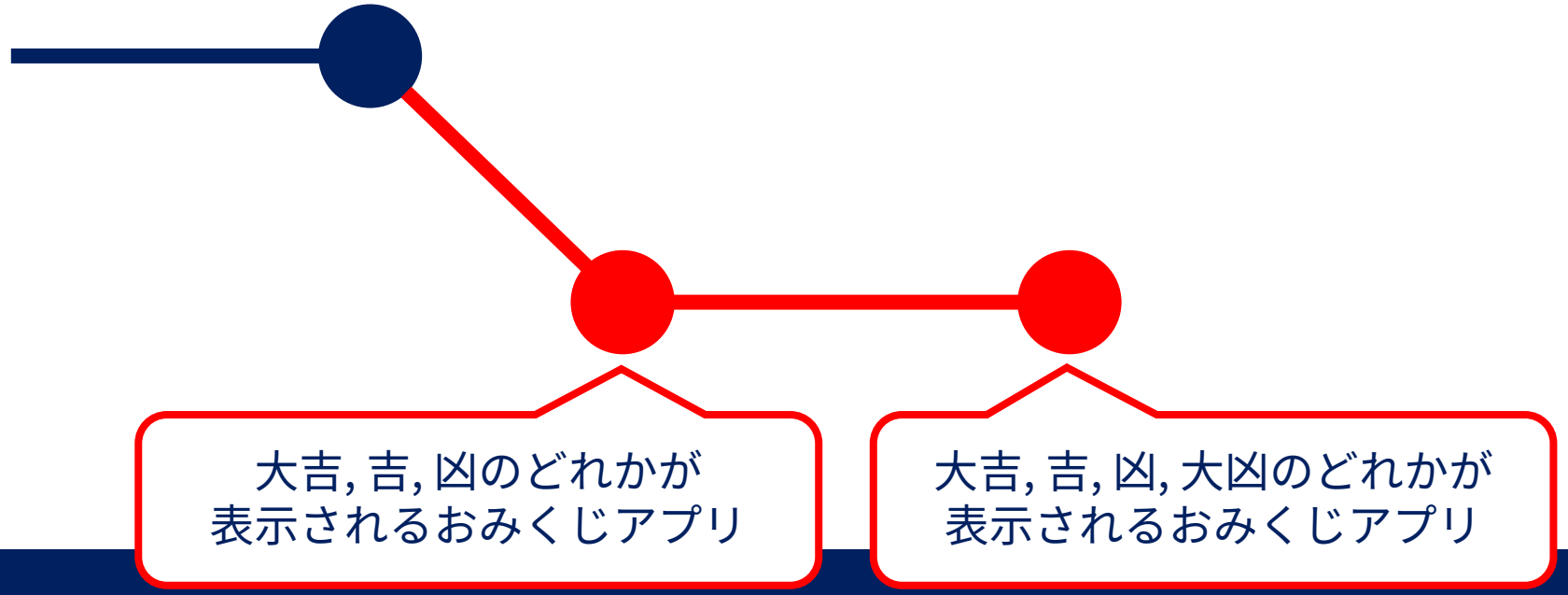
ブランチの例: おみくじアプリ

Git基礎



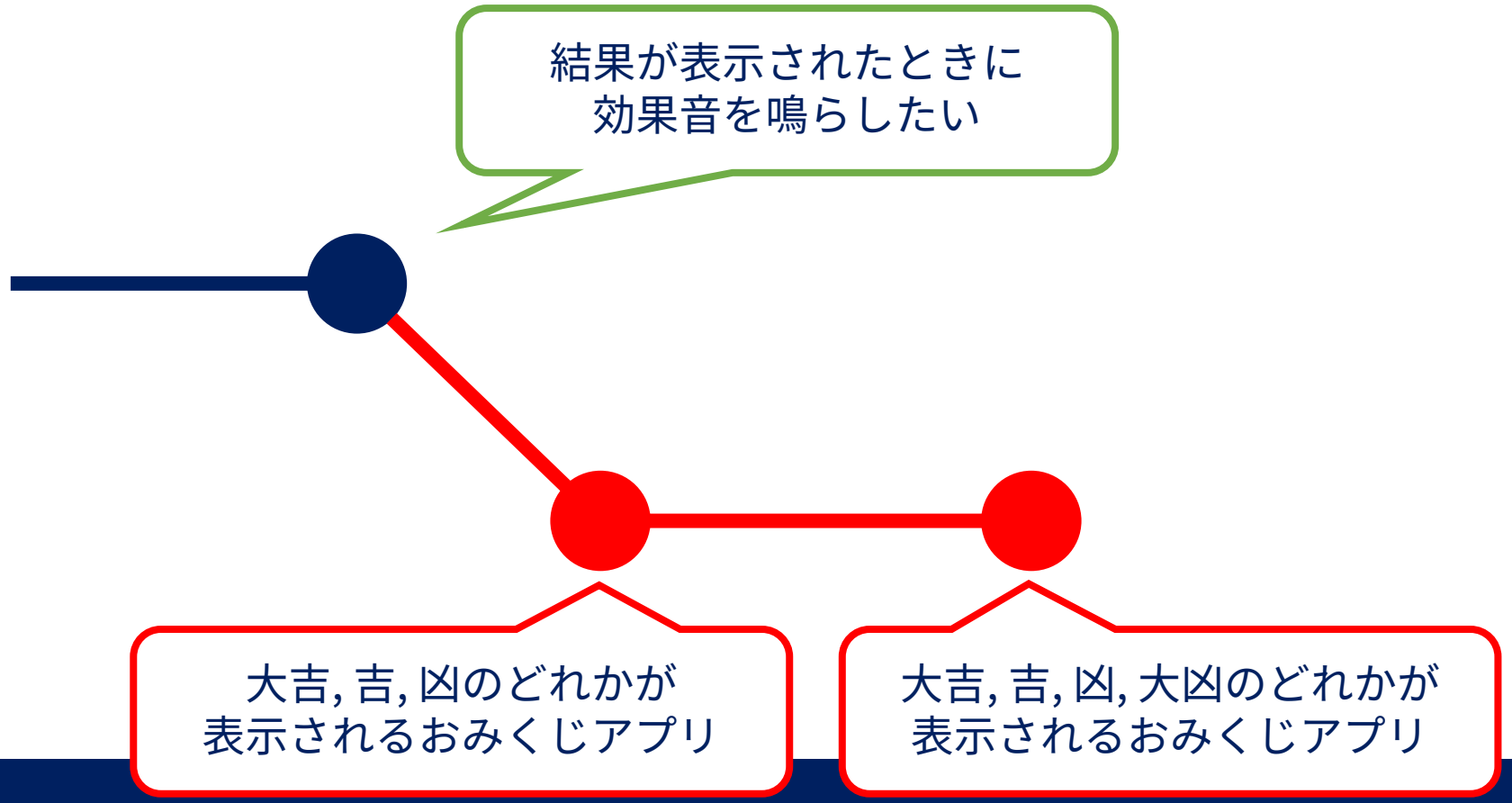
ブランチの例: おみくじアプリ

Git基礎

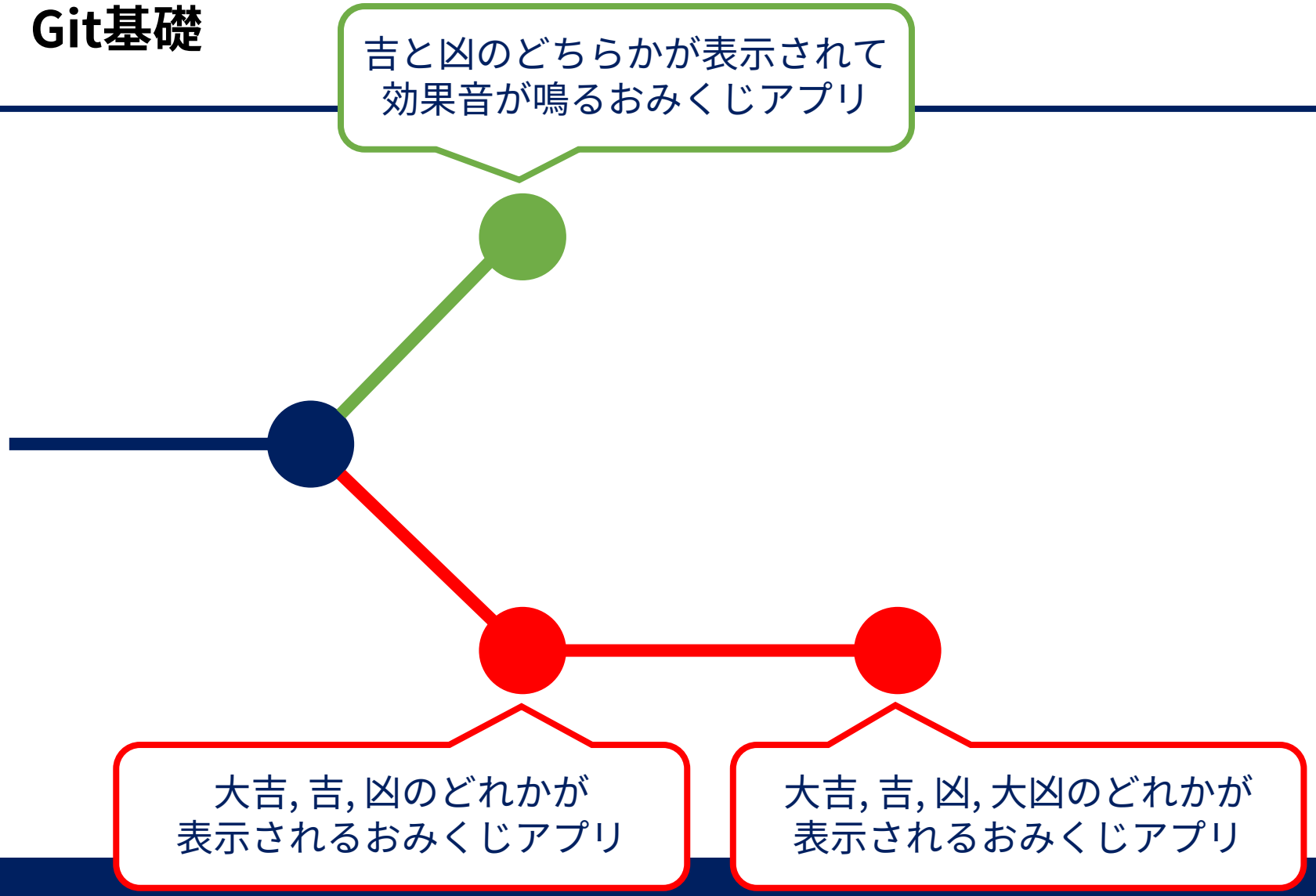


ブランチの例: おみくじアプリ

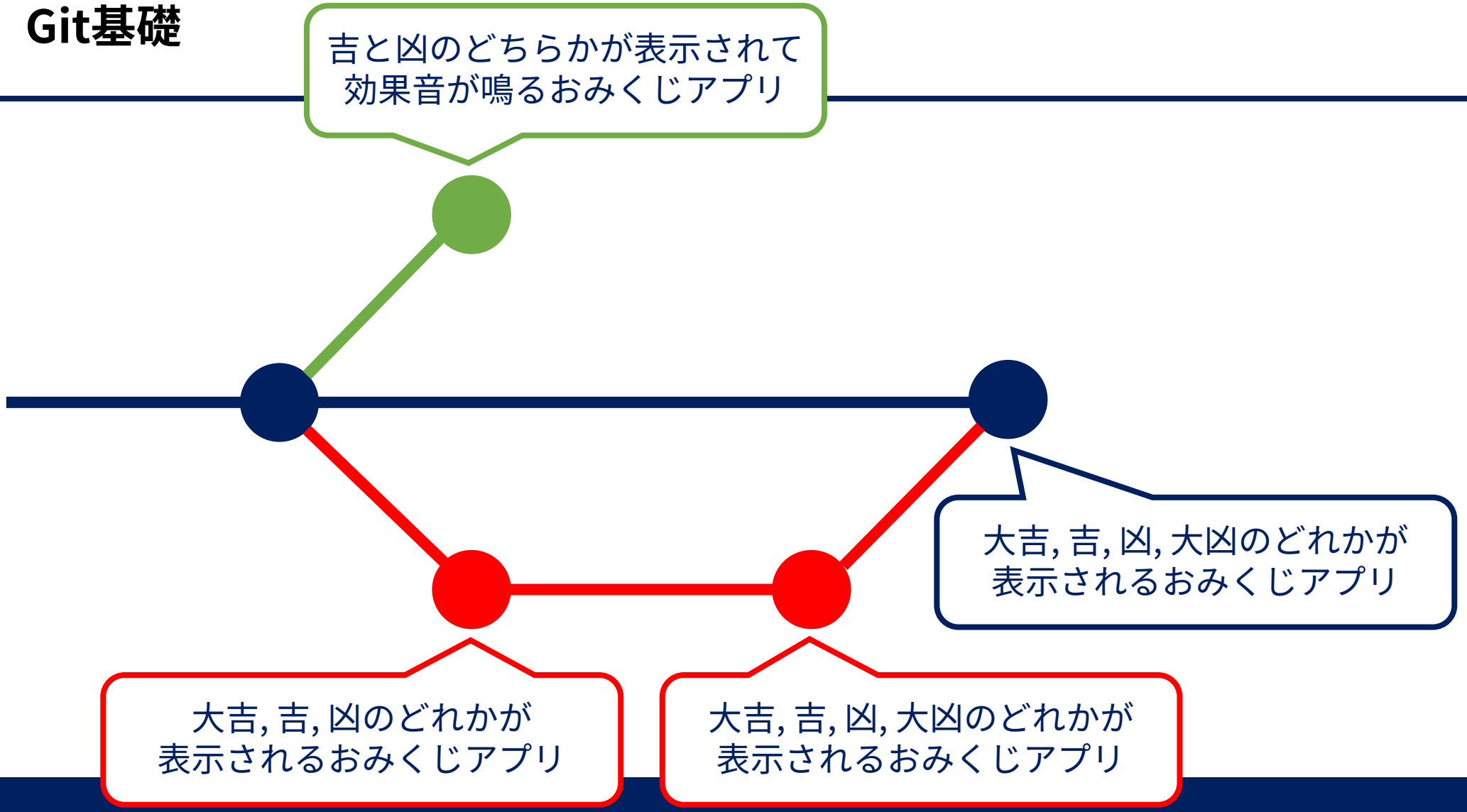
Git基礎



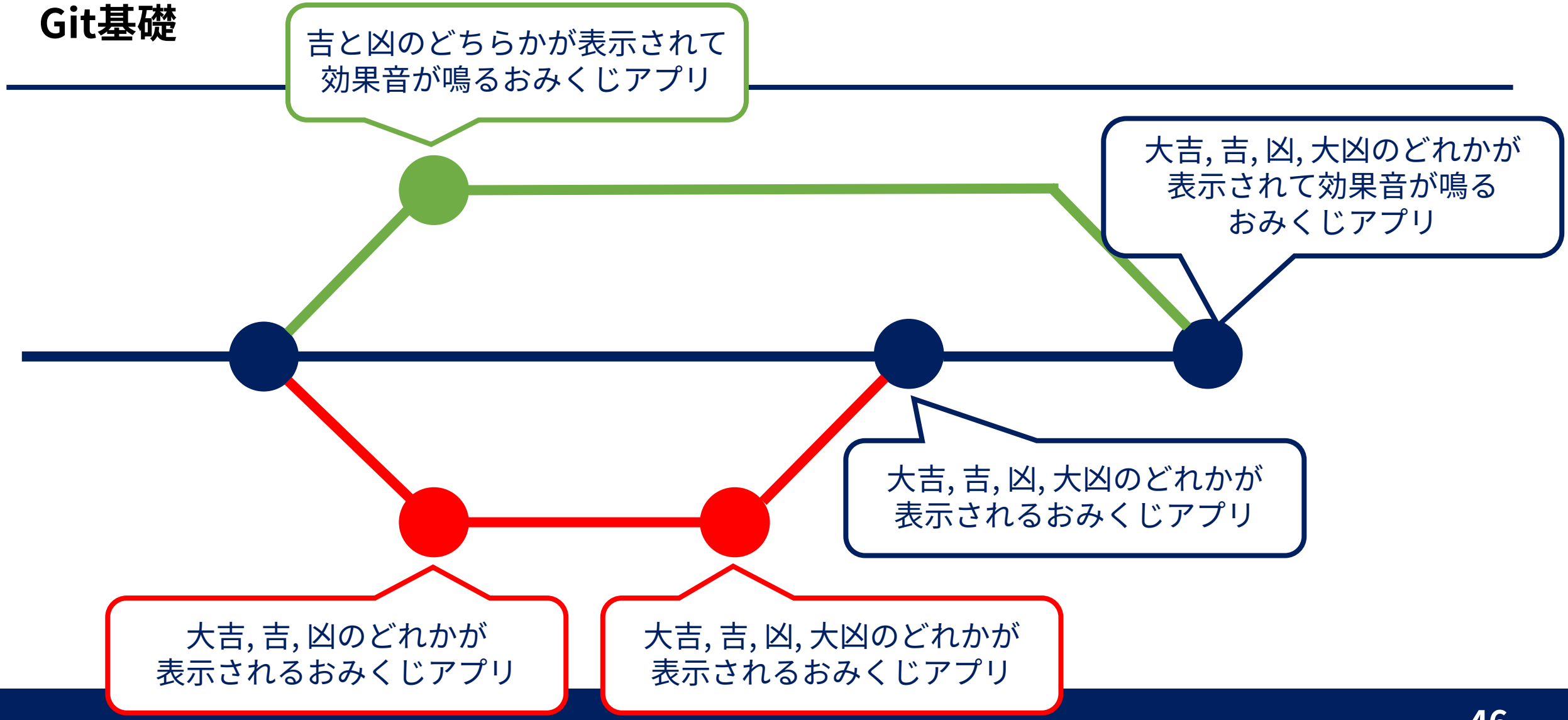
ブランチの例: おみくじアプリ



ブランチの例: おみくじアプリ

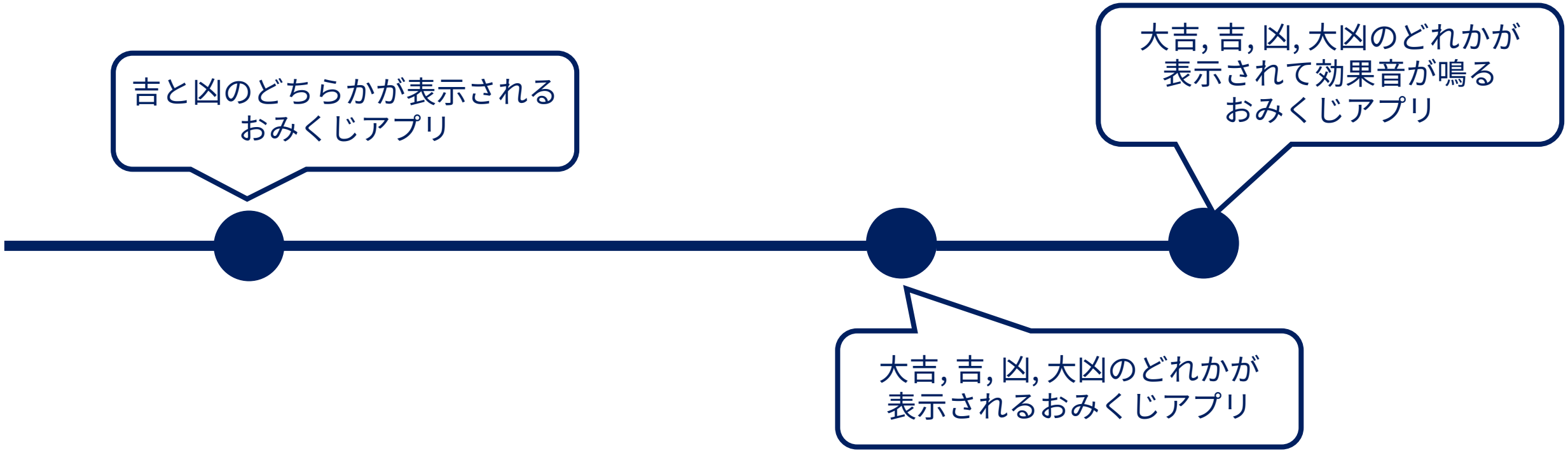


ブランチの例: おみくじアプリ



ブランチの例: おみくじアプリ

Git基礎



マージ

Git基礎

- **元となったブランチに編集内容を取り込む操作**

  マージとは、そのブランチに記録された編集内容を、元となったブランチに取り込む操作のことです。便宜上、今後は元となったブランチをmainブランチと呼びます。

コラム: Gitの弱点

- **Gitはバイナリファイルの管理が得意ではない**

  Gitはソフトウェアのソースコードを管理するために開発されたシステムなので、バイナリファイル（画像や動画など）の管理はあまり得意ではありません。

Git演習

セットアップ

Git演習

Gitを利用するにはユーザー名およびメールアドレスを設定する必要があります。

以下のコマンドを参考に設定を行ってください。

```
git config --global user.name "[ユーザー名]"
git config --global user.email [メールアドレス]
```

※　[ユーザー名] および [メールアドレス] は各自で入力してください

ローカルリポジトリを作成する

Git演習

バージョン管理するディレクトリ内で以下のコマンドを実行することで、ローカルリポジトリを作成 (初期化) することができる。

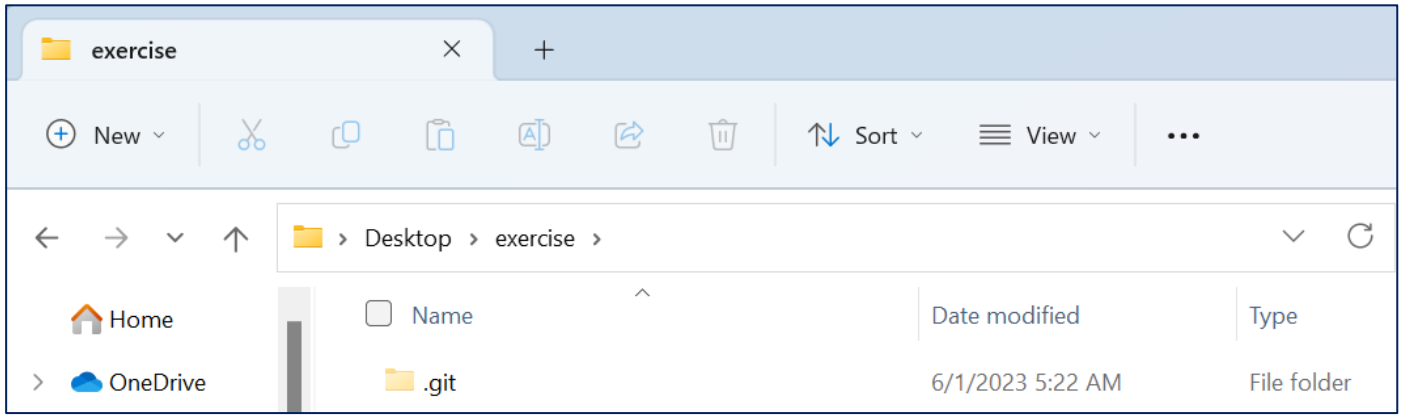
```
git init
```

コマンド実行後にエクスプローラ等でディレクトリ内を確認すると、”.git” というディレクトリがGitによって作成されていることがわかる。

演習: ローカルリポジトリを作成する

Git演習

デスクトップ上に "exercise" というディレクトリ (フォルダ) を作成し、その中でローカルリポジトリを作成してください。



".git" ディレクトリが存在することを確認しましょう。見つけられない場合はエクスプローラの設定を確認してください。

ステージング

Git演習

以下のコマンドを実行することで指定したファイルをステージングできる。

```
git add [ファイル名]
```

ステージングできているかどうかは以下のコマンドを実行することで確認できる。

```
git status
```

ステージング

Git演習

```
> git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   [ステージングしたファイル]
```

コミット

Git演習

以下のコマンドを実行することでステージングされた変更をコミットできる。

```
git commit -m "[コミットメッセージ]"
```

コミットメッセージには、そのコミットで行った変更に関する簡単な説明を書きましょう。特にルールはありませんが、一般的に良いとされるコミットメッセージについて、時間があれば講義の終盤で紹介します。

演習: ステージング・コミット

Git演習

リポジトリ内に適当な内容のファイルを１つ作成し、そのファイルに対する変更を３回（もしくは４回以上）コミットしてください。

コミットログ

Git演習

コミットログではこれまでのコミットに関する様々な情報を確認できる。

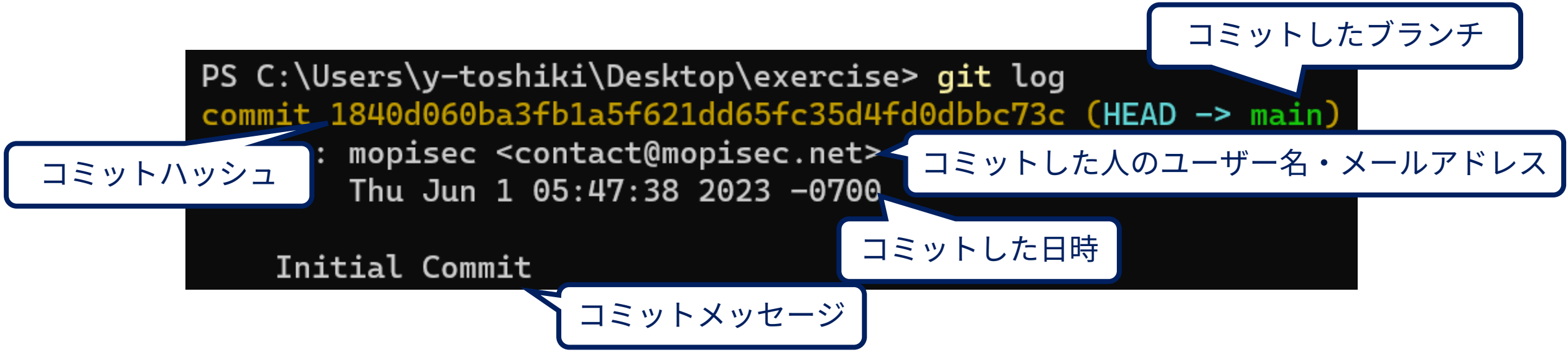
```
PS C:\Users\y-toshiki\Desktop\exercise> git log
commit 1840d060ba3fb1a5f621dd65fc35d4fd0dbbc73c (HEAD -> main)
Author: mopisec <contact@mopisec.net>
Date:   Thu Jun 1 05:47:38 2023 -0700

    Initial Commit
```

※　コミットログは "git log" コマンドで確認することができる。

コミットログ

Git演習

コミットログではこれまでのコミットに関する様々な情報を確認できる。



※　コミットログは "git log" コマンドで確認することができる。

演習: コミットログ

Git演習

コミットログを見て、正しくコミットできているか確認しましょう。

コミットした回数分のログのエントリーが表示されているはずです。

コミット間の差分

Git演習

以下のコマンドを実行することで、コミット間の差分（変更された内容）を表示することができる。

```
git diff [コミットハッシュA] [コミットハッシュB]
```

コミットハッシュの代わりにHEADのようなポインタを指定することも可能。
コミットハッシュの後に `:ファイル名` を書き足すことで、特定のファイルに関する情報のみ表示することもできる。

演習: コミット間の差分

Git演習

HEADと最初のコミットの差分を表示してみましょう。

ブランチの作成・切替

Git演習

以下のコマンドを実行することでブランチを作成することができる。

```
git branch [ブランチ名]
```

以下のコマンドを実行することで指定したブランチに切り替えることができる。

```
git switch [ブランチ名]
```

演習: ブランチの作成・切替

Git演習

ブランチを作成して切り替えた後、新しいファイルを作成して、再度3回以上コミットしてください。その後、mainブランチに戻って（切り替えて）ください。

マージ

Git演習

以下のコマンドを実行することでブランチをマージすることができる。

```
git merge [マージする (取り込む) ブランチ名]
```

ブランチの削除

Git演習

マージ後、不要となったブランチは削除してもよい。

以下のコマンドを実行することでブランチを削除することができる。

```
git branch -d [削除するブランチ名]
```

演習: マージ

Git演習

以前の演習で作成したブランチをmainブランチにマージしてください。使用したブランチはもう使わないので、削除しておきましょう。
作業を終えた後、コミットログでマージ元のブランチでのコミットが表示されることを確認しましょう。

クローン

Git演習

以下のコマンドを実行することで、リモートリポジトリをクローンできる。

```
git clone [クローン元のリモートリポジトリのパス]
```

(例)　https://example.com/example.git をクローンするコマンド:

```
git clone https://example.com/example.git
```

演習: クローン

Git基礎

親ディレクトリ（..）に移動してから、以下のパスのリモートリポジトリをクローンしてみましょう。

```
https://camp.mopisec.net/mopisec/clone-exercise.git
```

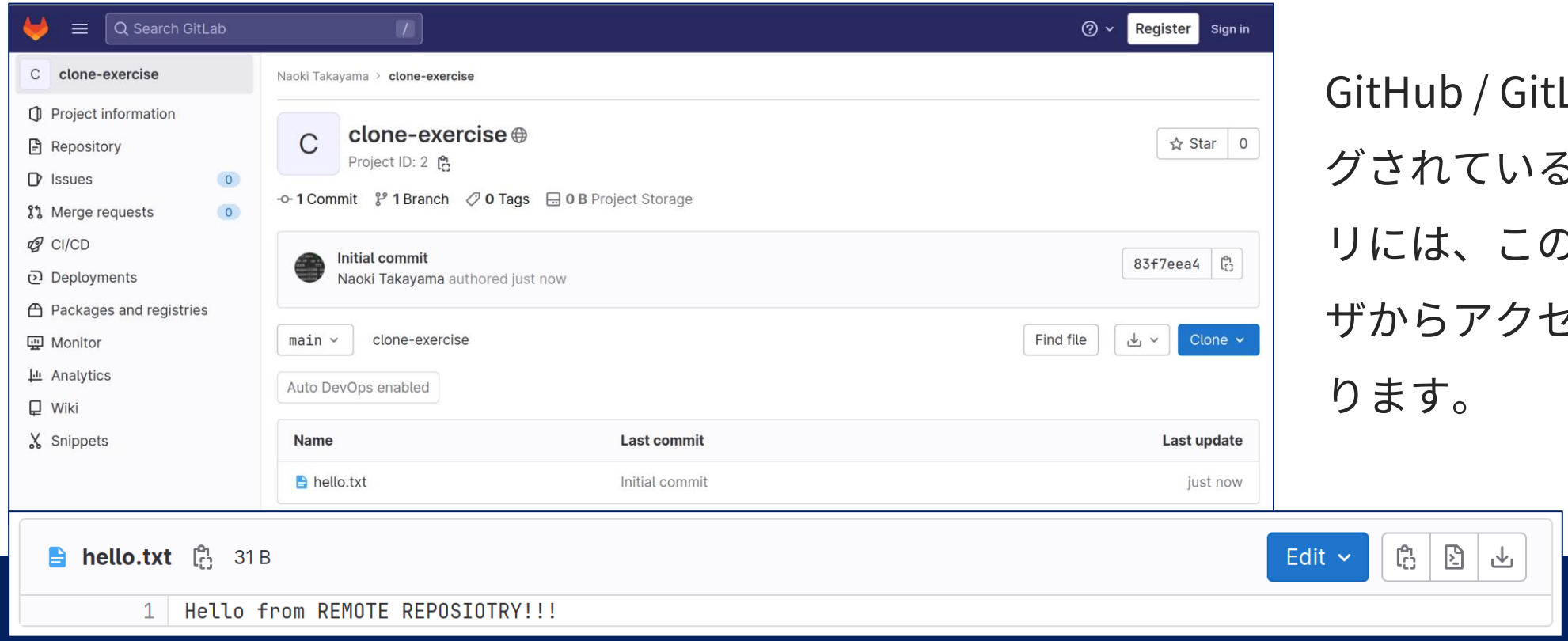
クローンしたリポジトリの中に "hello.txt" というファイルがあるはずです。

何が書かれているか確認してください。

演習: クローン

Git基礎

Webブラウザでリモートリポジトリにアクセスしてみましょう。



GitHub / GitLab等でホスティングされているリモートリポジトリには、このようにWebブラウザからアクセスできる場合があります。

演習で使用するGitLab環境

Git演習

- **GitLab**

  Gitのリモートリポジトリをホスティングするためのプラットフォーム。同系統のプラットフォームとしてGitHub等がある。

- **演習で使用するGitLab環境**

  https://camp.mopisec.net （キャンプ中は利用可能）

コラム: GitHub

Git演習

- GitHub

  世界最大のGitのプラットフォーム。2023年1月時点で1億人以上のユーザーを有しており、3億7300万個のリモートリポジトリをホスティングしている。

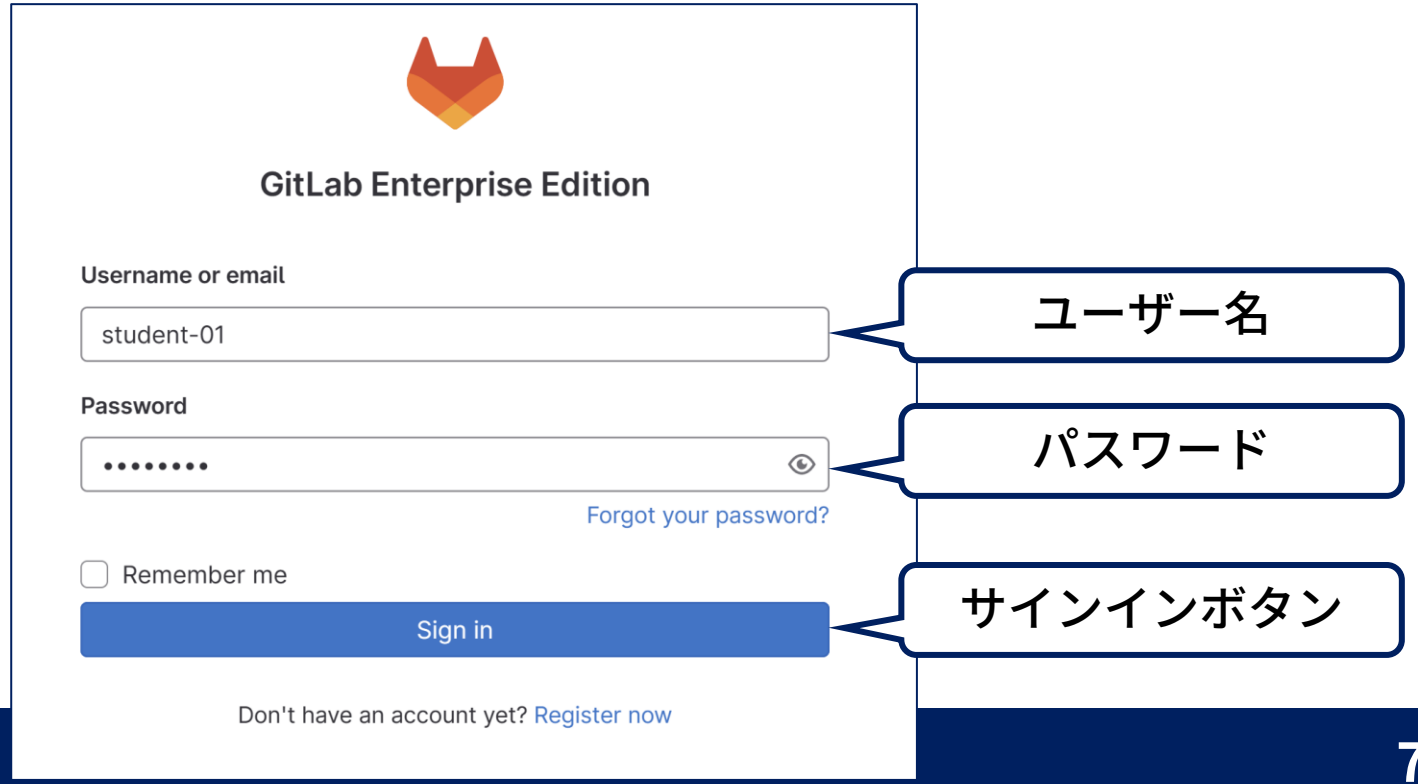
  ※ 13歳以上でないと使用することができません。

  参考: https://docs.github.com/ja/site-policy/github-terms/github-terms-of-service

演習で使用するGitLab環境

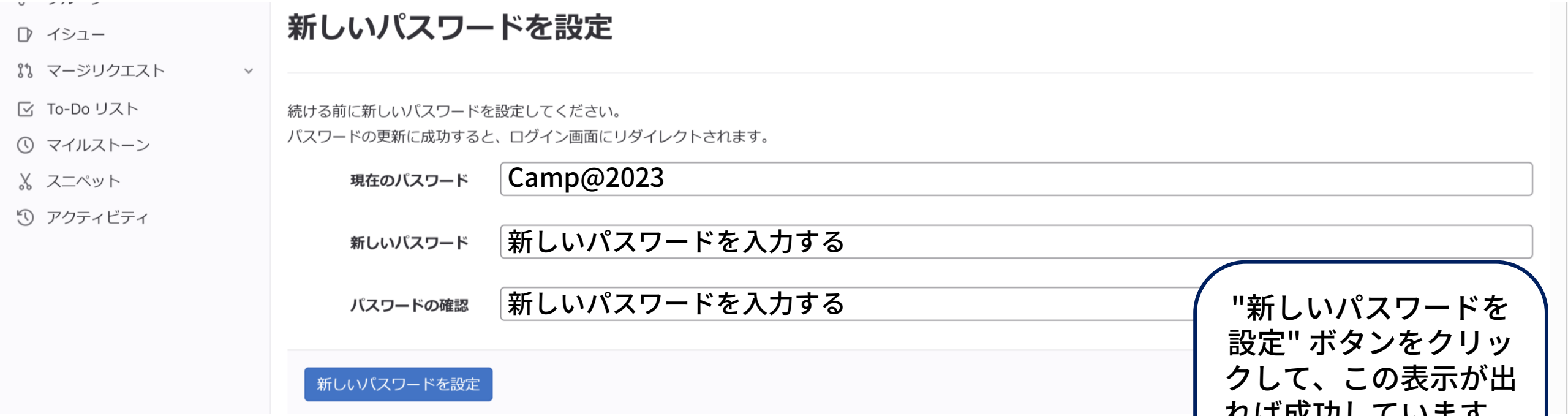
Git演習

- アカウント情報（初期パスワードは Camp@2023 です）
  - student-01（安部さん）
  - student-02（木津さん）
  - student-03（村山さん）
  - student-04（永井さん）
  - student-05（迫園さん）



新しいパスワードを設定する

Git演習

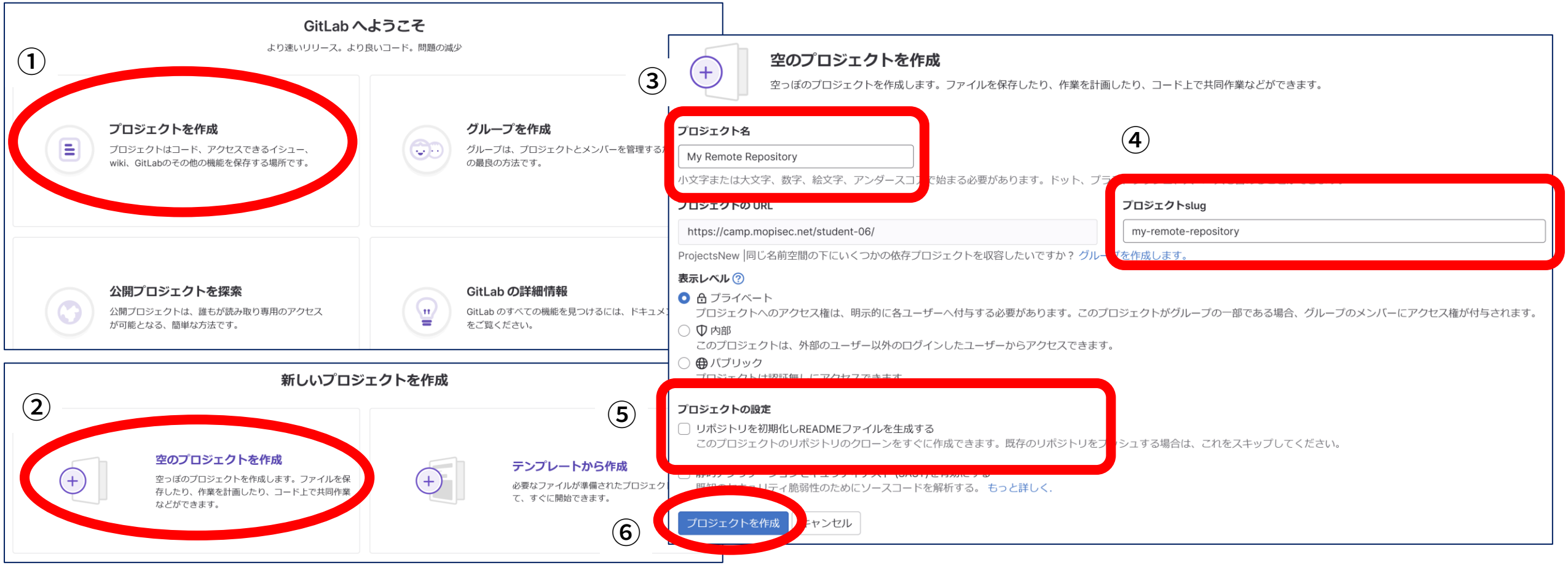


パスワードが正常に変更されました

"新しいパスワードを設定" ボタンをクリックして、この表示が出れば成功しています。再度 **新しいパスワードを使って** サインインしてください。

演習: リモートリポジトリを作成する

Git演習



演習: リモートリポジトリを作成する

Git演習

演習: GitLabアカウントにSSH鍵を追加する

Git演習

ターミナル上でSSH鍵ペアを作成します。

```
ssh-keygen
type  .ssh¥id_rsa.pub
```

上記の二つのコマンドを実行すると ".ssh" ディレクトリ内に秘密鍵 "id_rsa" ファイルと公開鍵 "id_rsa.pub" が保存され、公開鍵の内容が出力されます。

秘密鍵はあなただけの秘密の情報です。取り扱いには注意してください。

演習: GitLabアカウントにSSH鍵を追加する

Git演習

```
> ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (C:¥Users¥[snip]/.ssh/id_rsa):[ENTER]
Created directory 'C:¥¥Users¥¥[snip]/.ssh'.
Enter passphrase (empty for no passphrase):[ENTER]
Enter same passphrase again:[ENTER]
Your identification has been saved in C:¥Users¥[snip]/.ssh/id_rsa
[snip]
> type .ssh¥id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQA[snip]
```

演習: GitLabアカウントにSSH鍵を追加する

Git演習

演習: GitLabアカウントにSSH鍵を追加する

Git演習

演習: GitLabアカウントにSSH鍵を追加する

Git演習

```
ssh git@camp.mopisec.net
```

```
> ssh git@camp.mopisec.net
PTY allocation request failed on channel 0
Welcome to GitLab, @[snip]!
Connection to camp.mopisec.net closed.
```

リモートリポジトリを登録する

Git演習

以下のコマンドを実行することでローカルリポジトリにリモートリポジトリのパスを登録することができる。

```
git remote add [リモートリポジトリの名前] [〃のパス]
```

リモートリポジトリの名前は自由に設定可能ですが、多くの場合デフォルトの名前である "origin" が使われる。

プッシュ

Git演習

以下のコマンドを実行することで、ローカルリポジトリに蓄積されたコミットをリモートリポジトリに対してプッシュすることができる。

```
git push［リモートリポジトリの名前］［プッシュするブランチ］
```

プッシュやプル等の操作を実行するにはリモートリポジトリのパスをローカルリポジトリに登録しておく必要があります。

演習: プッシュ

Git演習

"exercise" ローカルリポジトリからGitLab上で作成したリモートリポジトリに対してmainブランチをプッシュしてみましょう。

ヒント

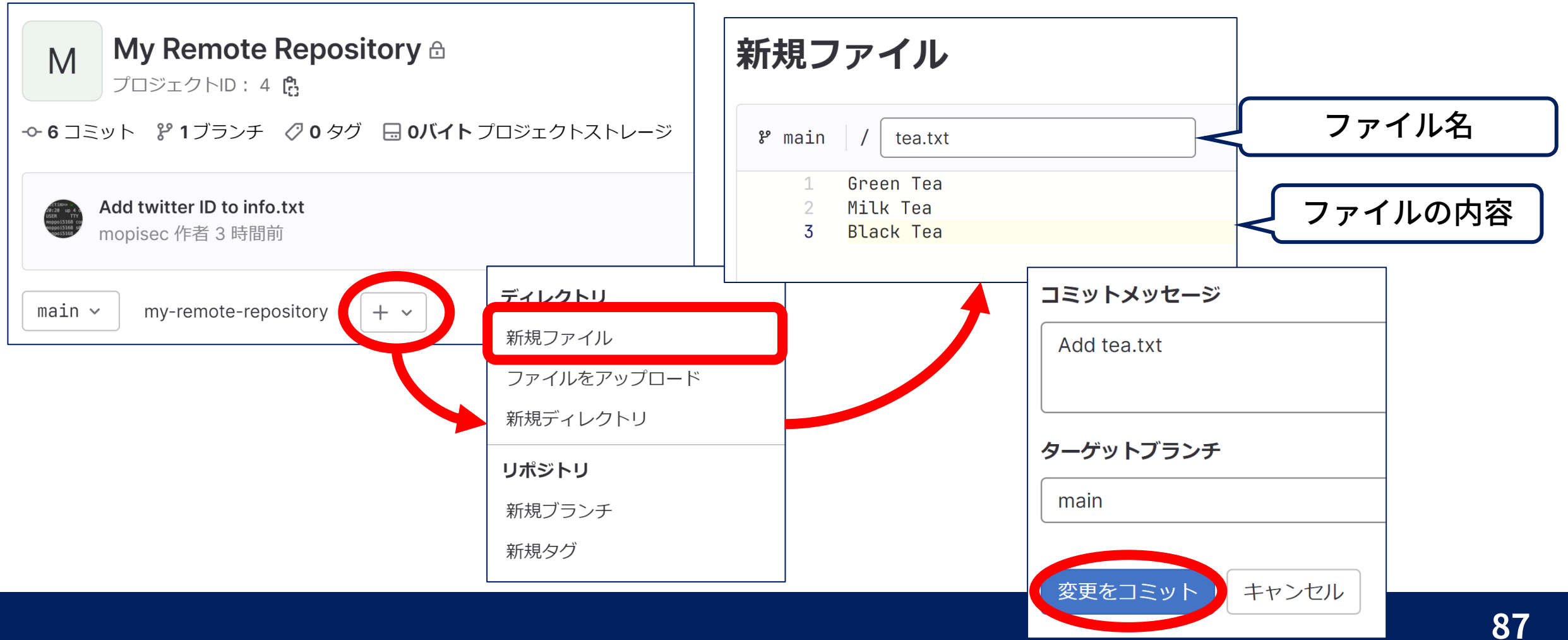
GitLabのリモートリポジトリのページからリモートリポジトリのパスを確認することができる。青い "クローン" ボタンをクリックして "SSHでクローン" の下に表示されているパスを登録してください。
※ HTTPSクローンは使用しないでください

演習: プッシュ

Git演習

演習: GitLab上でファイルを作成/編集する

Git演習



プル

Git演習

以下のコマンドを実行することで、リモートリポジトリに蓄積されたコミットをローカルリポジトリに対してプルすることができる。

```
git pull [リモートリポジトリの名前] [プルするブランチ]
```

演習: プル

Git演習

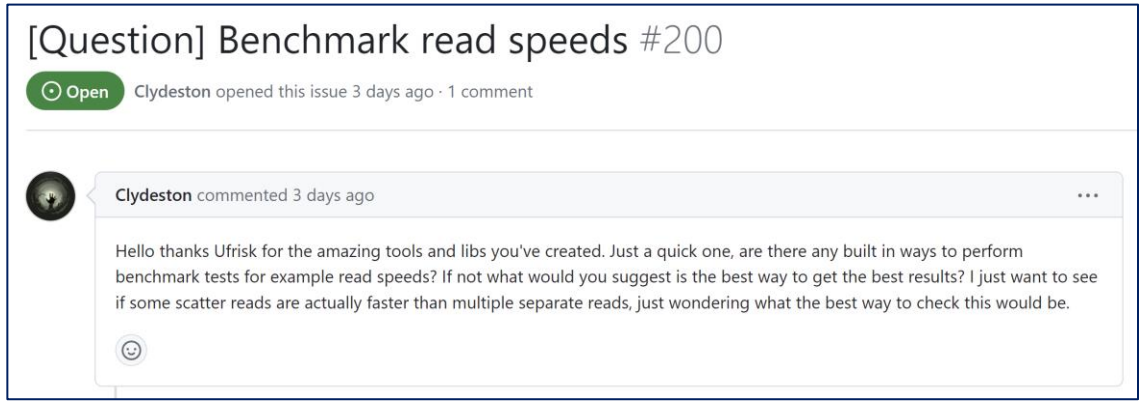
GitLab上で作成したリモートリポジトリから"exercise" ローカルリポジトリに対してmainブランチをプルしてみましょう。

Gitホスティングサービスの機能

イシュー

Gitホスティングサービスの機能

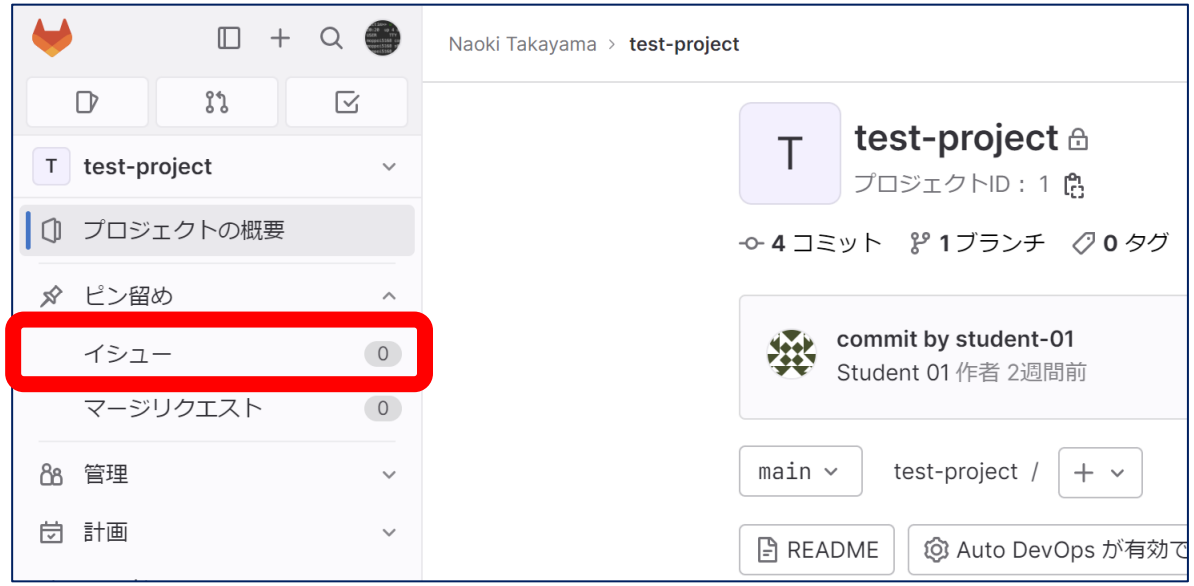
開発者間（もしくは開発者・ユーザー間）で特定の話題についてスレッド形式で議論するための機能。バグの報告、機能追加の要望、ユーザーからの質問、やるべきタスクなどが記載されることが多い。



ufrisk/MemProcFS - GitHub のIssueより

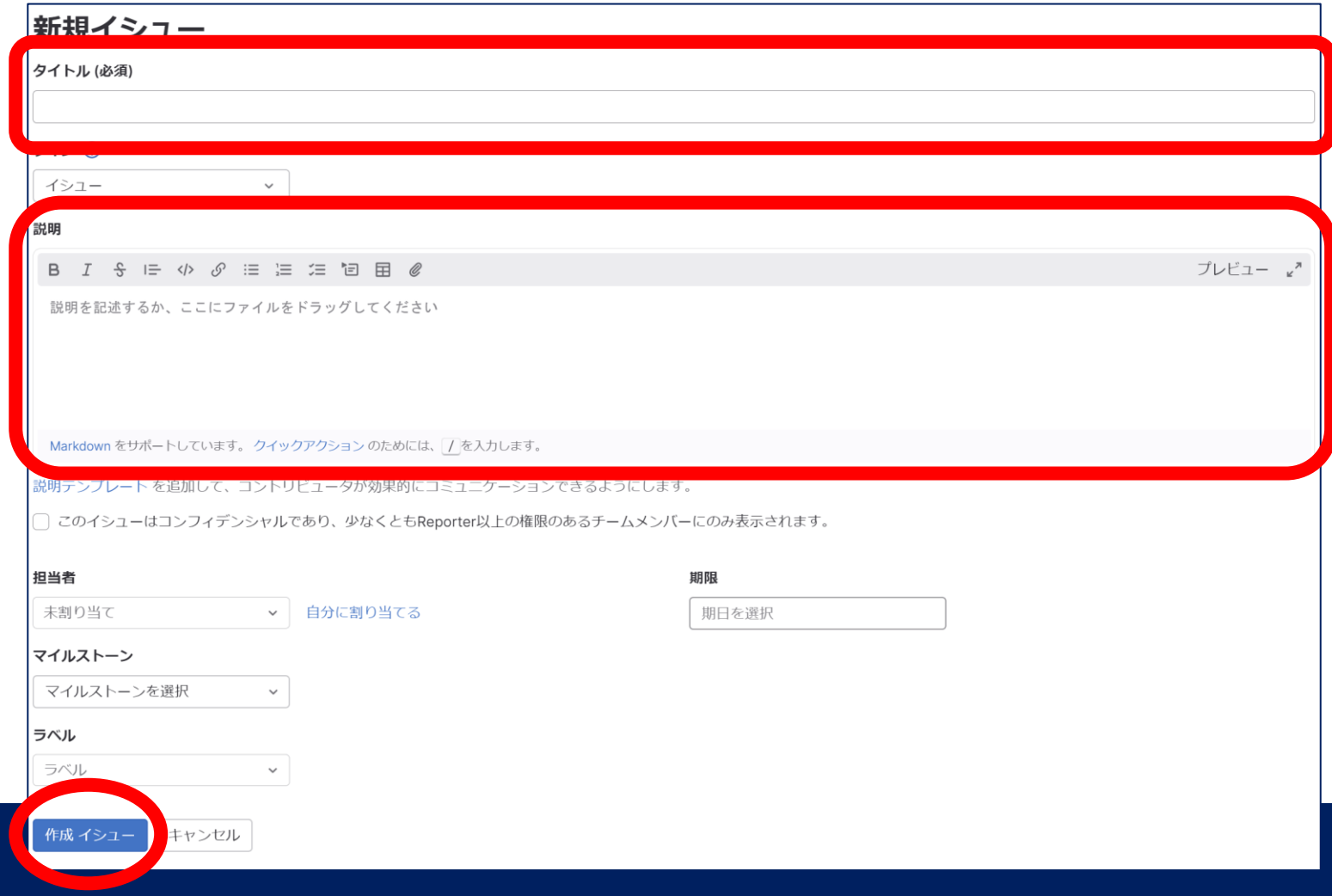
GitLabでのイシューの作成方法

Gitホスティングサービスの機能



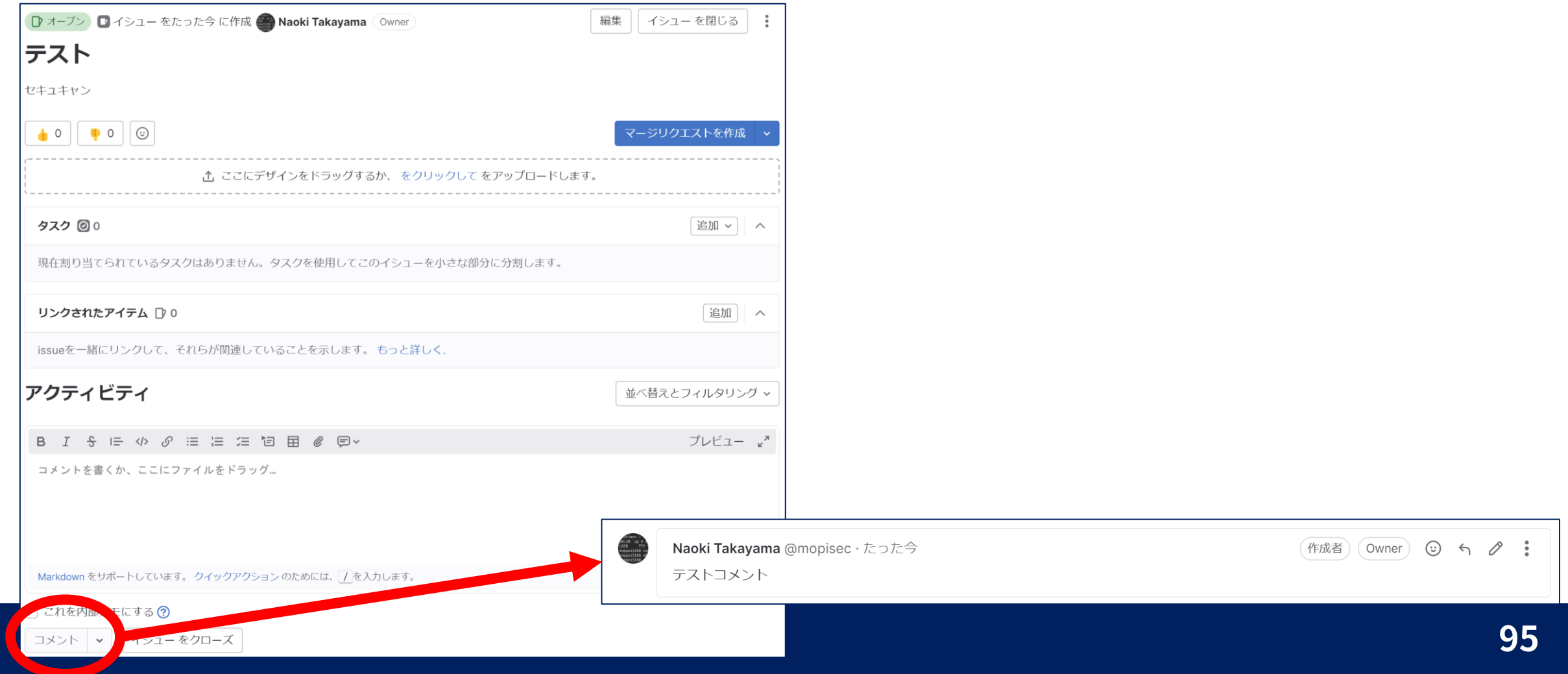
GitLabでのイシューの作成方法

Gitホスティングサービスの機能



GitLabでのイシューの作成方法

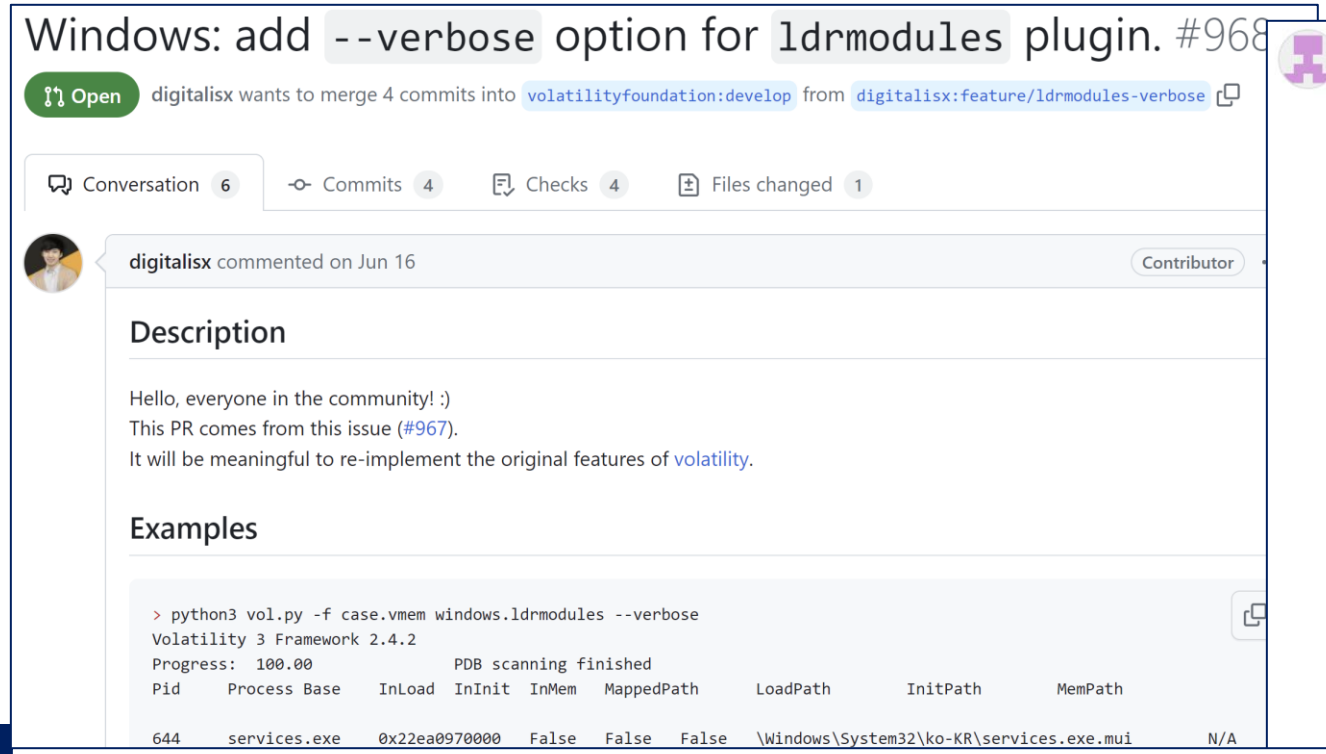
Gitホスティングサービスの機能

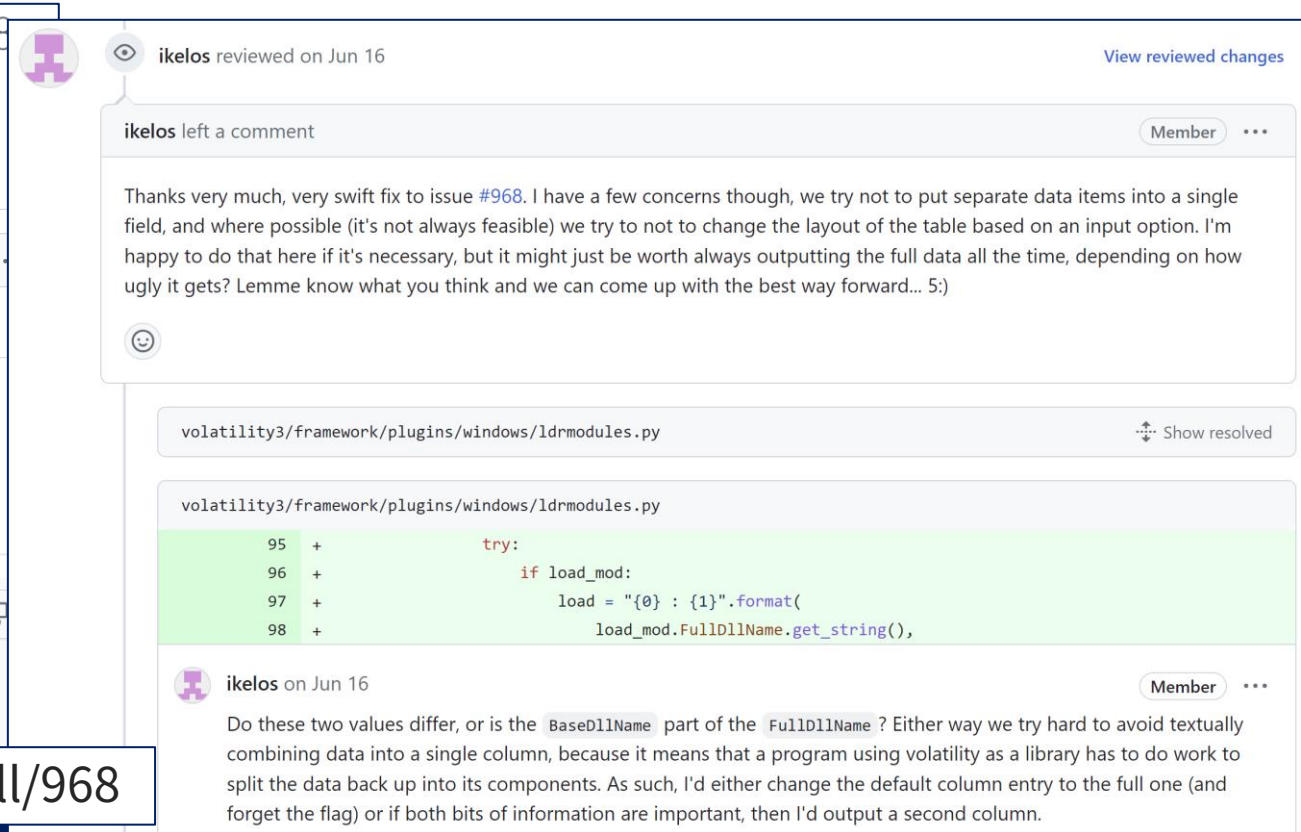


マージリクエスト（プルリクエスト）

Gitホスティングサービスの機能

開発者にブランチのレビューおよびマージを依頼するための機能。

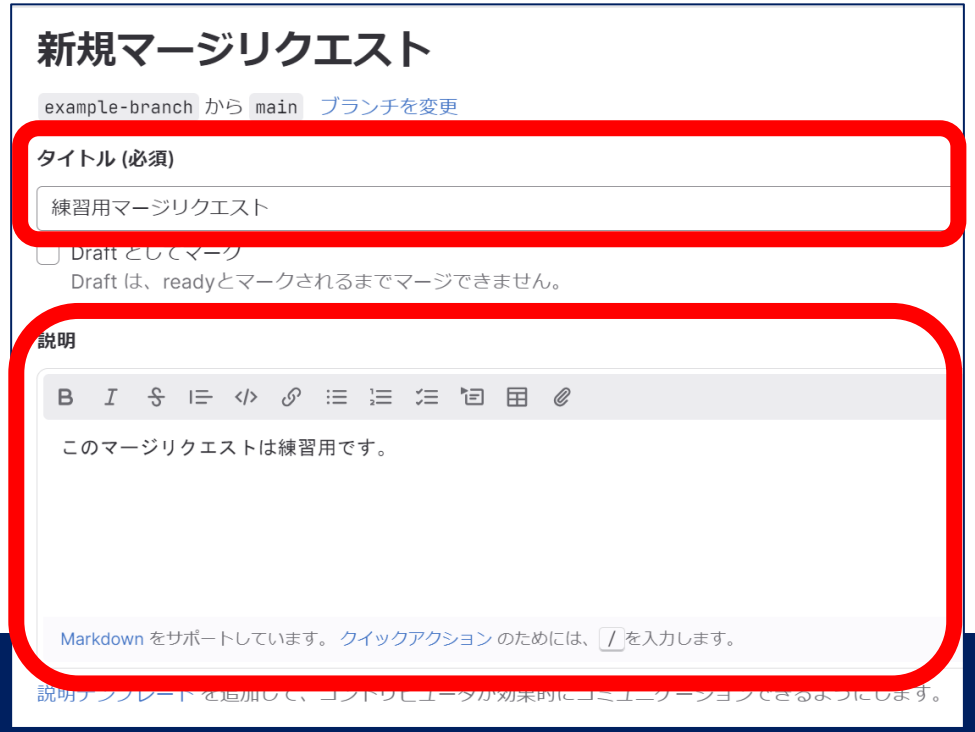




https://github.com/volatilityfoundation/volatility3/pull/968

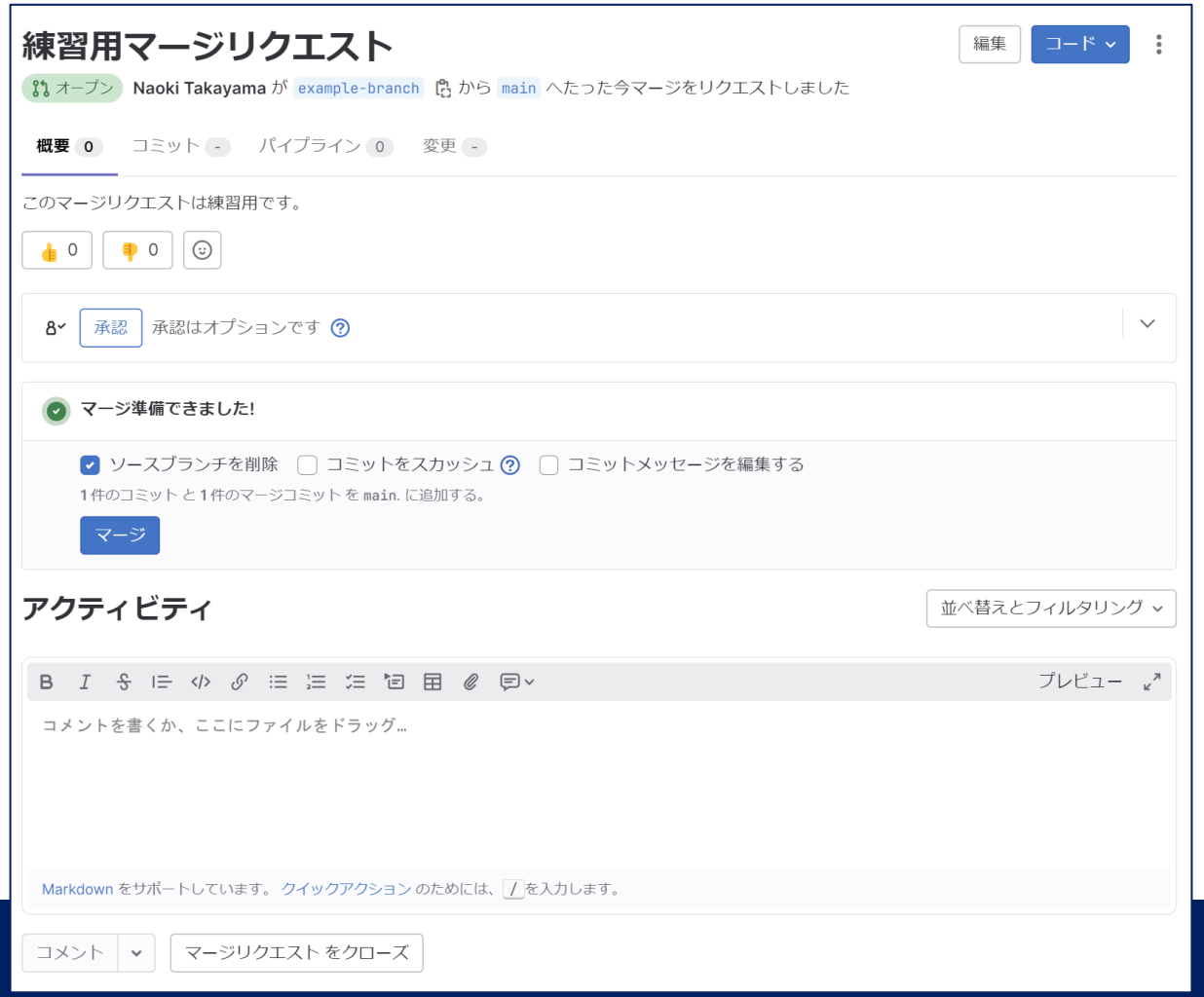
GitLabでのマージリクエストの作成方法

Gitホスティングサービスの機能



GitLabでのマージリクエストの作成方法

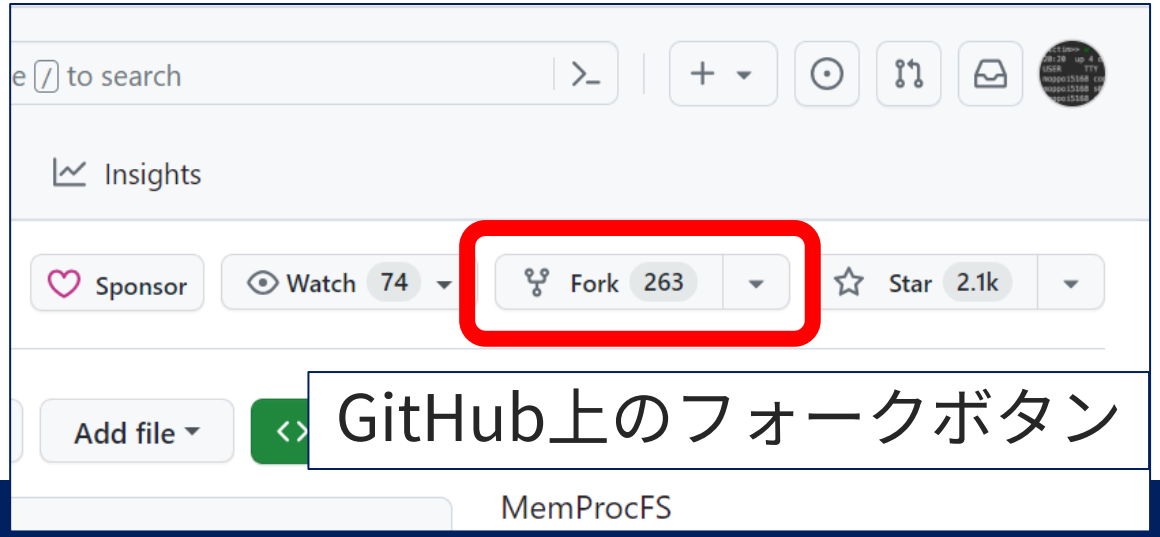
Gitホスティングサービスの機能



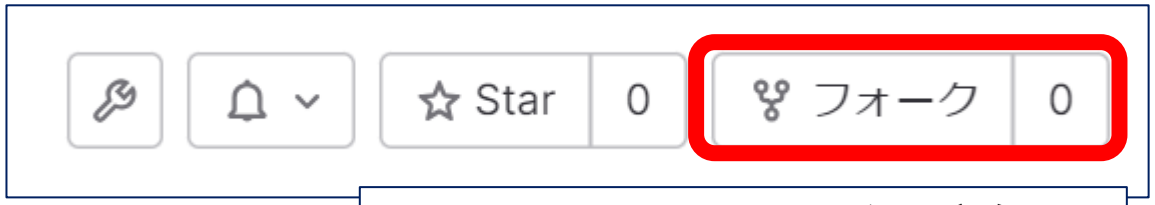
フォーク

Gitホスティングサービスの機能

リモートリポジトリをコピーする操作のこと。これにより自身が書き込み権限を持たないリポジトリであってもプッシュし、マージリクエストを作成することができる。



GitHub上のフォークボタン



GitLabのフォークボタン

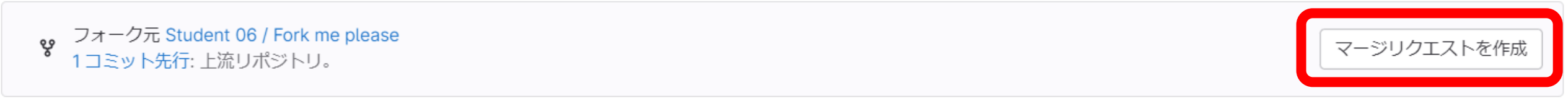
フォーク

Gitホスティングサービスの機能

フォーク → マージリクエスト

Gitホスティングサービスの機能

フォークしたリポジトリにプッシュすると、以下のような表示が出てマージリクエストを作成することができる。



その後、先ほどと同様の手順でマージリクエストを作成することができる。

演習: イシュー

Gitホスティングサービスの機能

以下のGitLabリポジトリ上でイシューを作成してみましょう。練習用なので、内容は適当で構いません。

```
https://camp.mopisec.net/mopisec/gitlab-exercise
```

また、自分以外の人が作成したイシューにコメントを書いてみましょう。

演習: マージリクエスト

Gitホスティングサービスの機能

以下のGitLabリポジトリ上でマージリクエストを作成してみましょう。練習用なので、内容は適当で構いません。

```
https://camp.mopisec.net/mopisec/merge-exercise
```

演習: フォーク → マージリクエスト

Gitホスティングサービスの機能

以下のGitLabリポジトリをフォークしてみましょう。

```
https://camp.mopisec.net/mopisec/gitlab-exercise
```

その後、フォークしたリポジトリにプッシュして、マージリクエストを作成してください。

クロージング

さいごに

クロージング

- **Gitの知識はあらゆる場面で役に立つ**

  開発はもちろん、開発に関わらない場面でも、ドキュメントやブログの管理、情報を集約する場としてGitは世界中で活用されています。
  Gitの知識は必ずこれから先、あなたの武器になるので、何かを開発する際には是非Gitを使ってソースコードをバージョン管理してみてください。