

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)

事前學習資料

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)

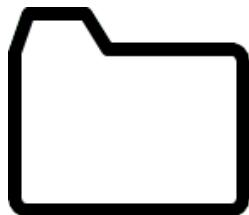
皆さんは学校などで“5W1H”という言葉を聞いたことはありますか？

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

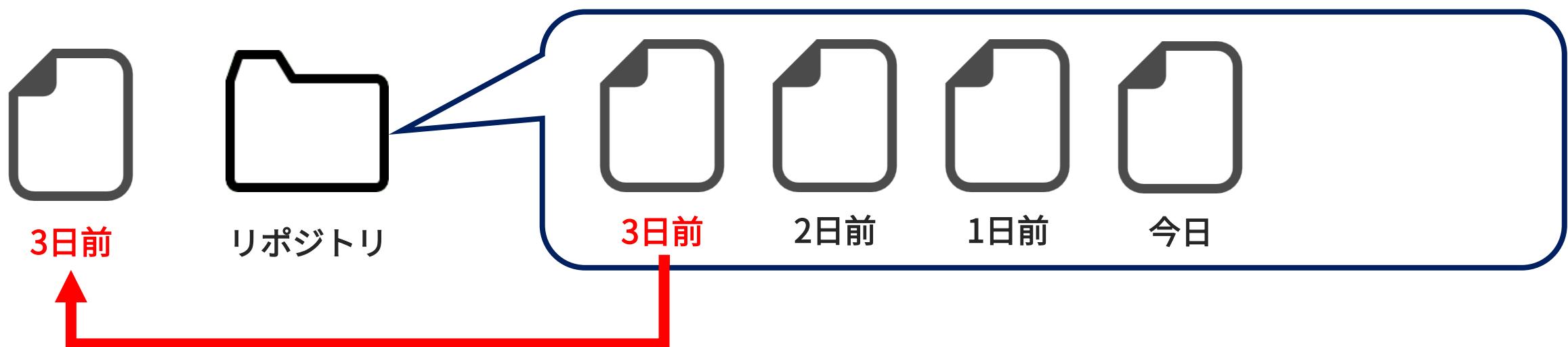
- コミットすると、リポジトリに記録される



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。

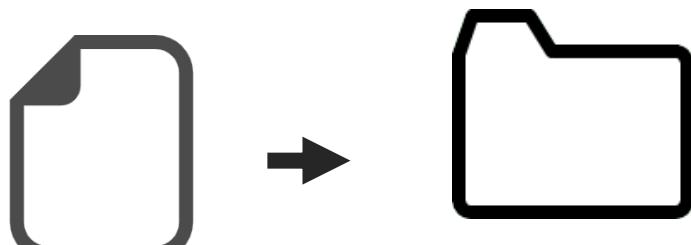
- **プル**

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

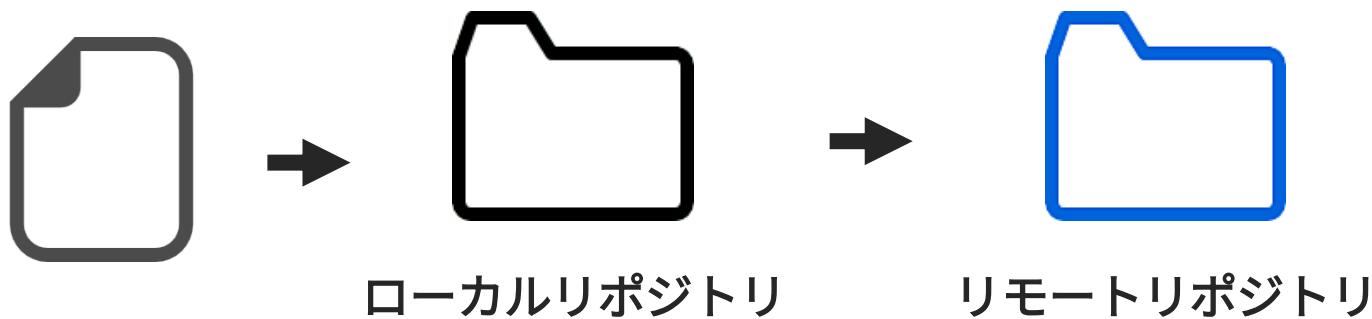


ローカルリポジトリ

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

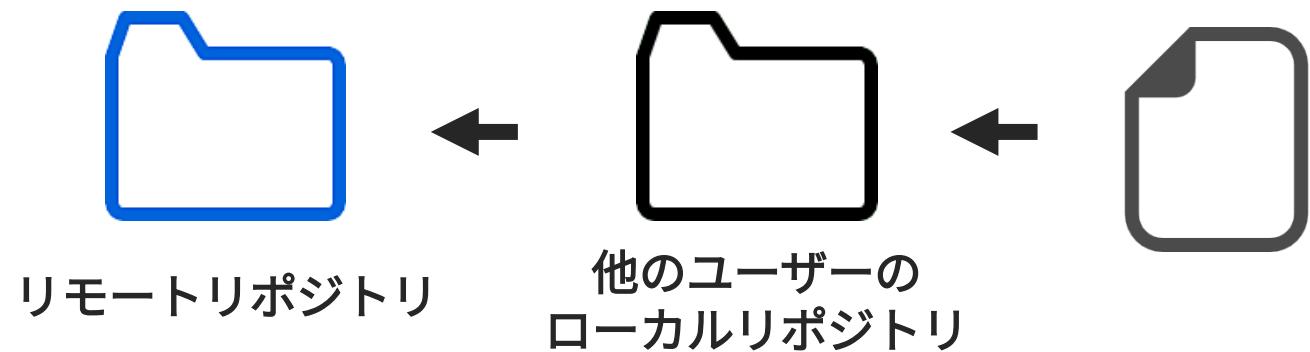
- 他のユーザーが自身のローカルリポジトリにプルして変更内容(編集されたファイル)を取得する



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

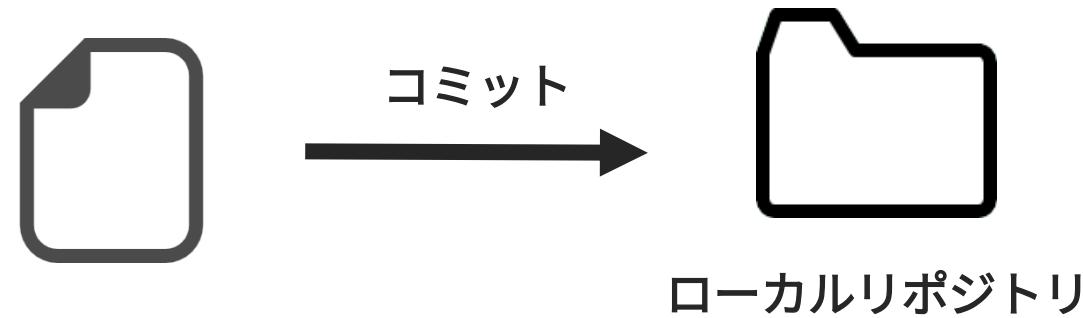
ローカルリポジトリとリモートリポジトリの2種類のリポジトリを運用できるバージョン管理システム。

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。

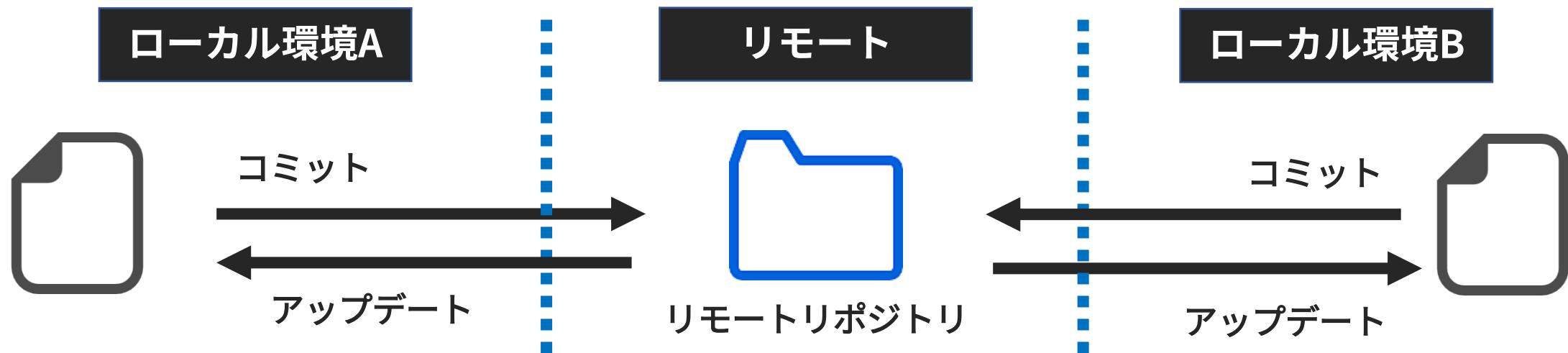


図解: 他のバージョン管理システム

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。

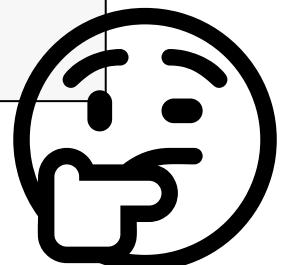
Gitとは (by ChatGPT)



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

吉と凶のどちらかが表示される
おみくじアプリ



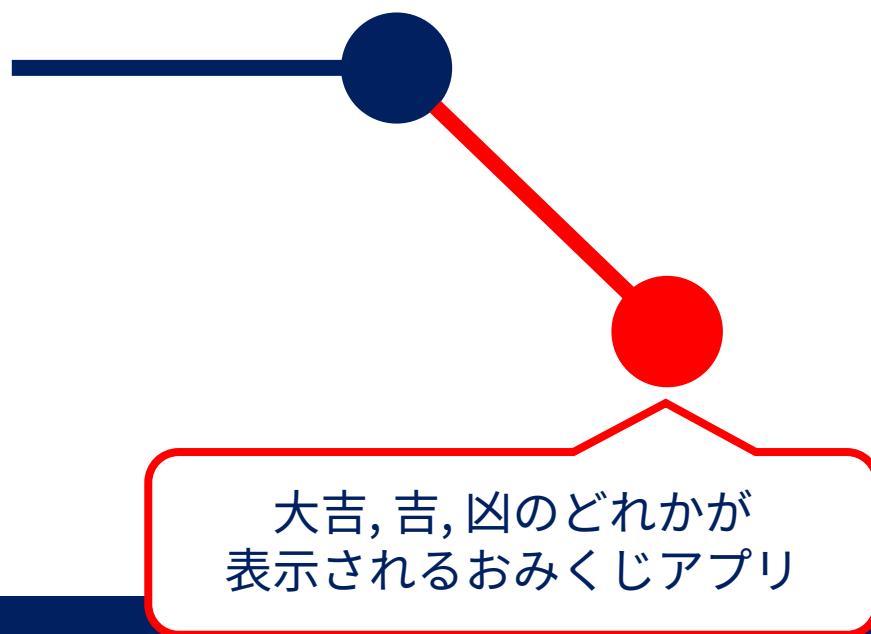
ブランチの例: おみくじアプリ

Git基礎



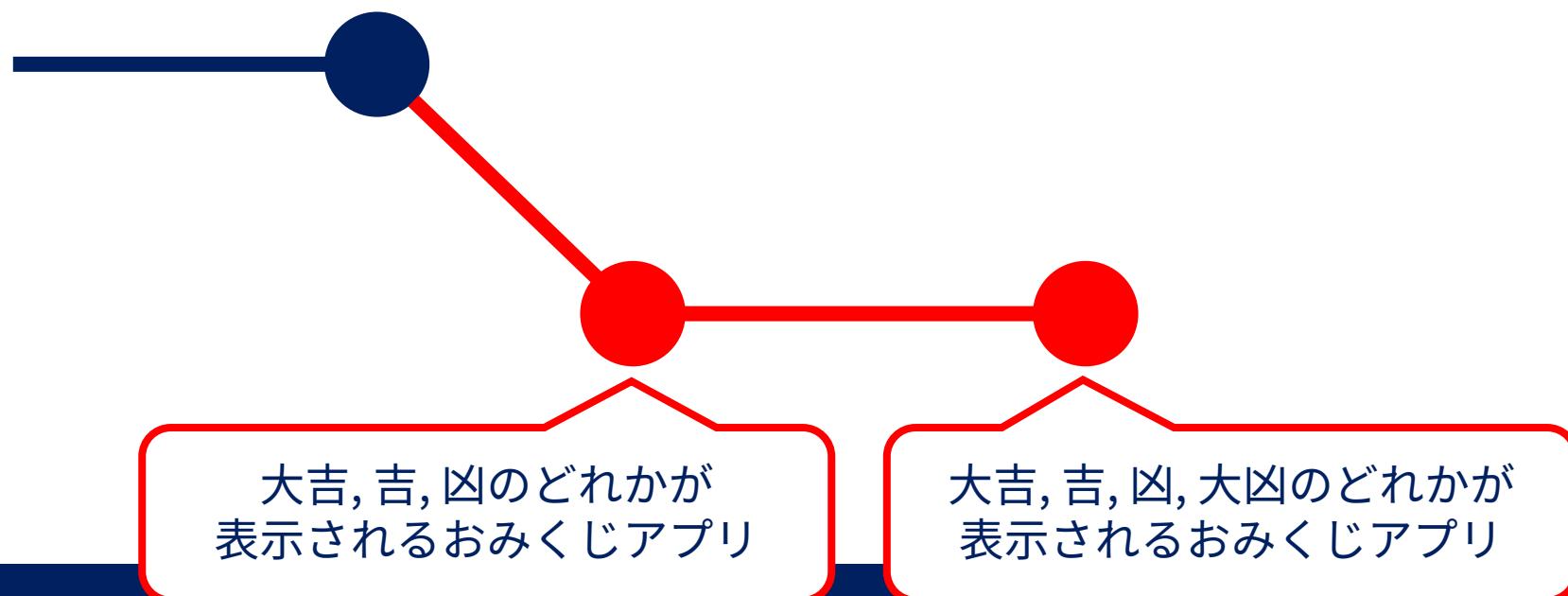
ブランチの例: おみくじアプリ

Git基礎



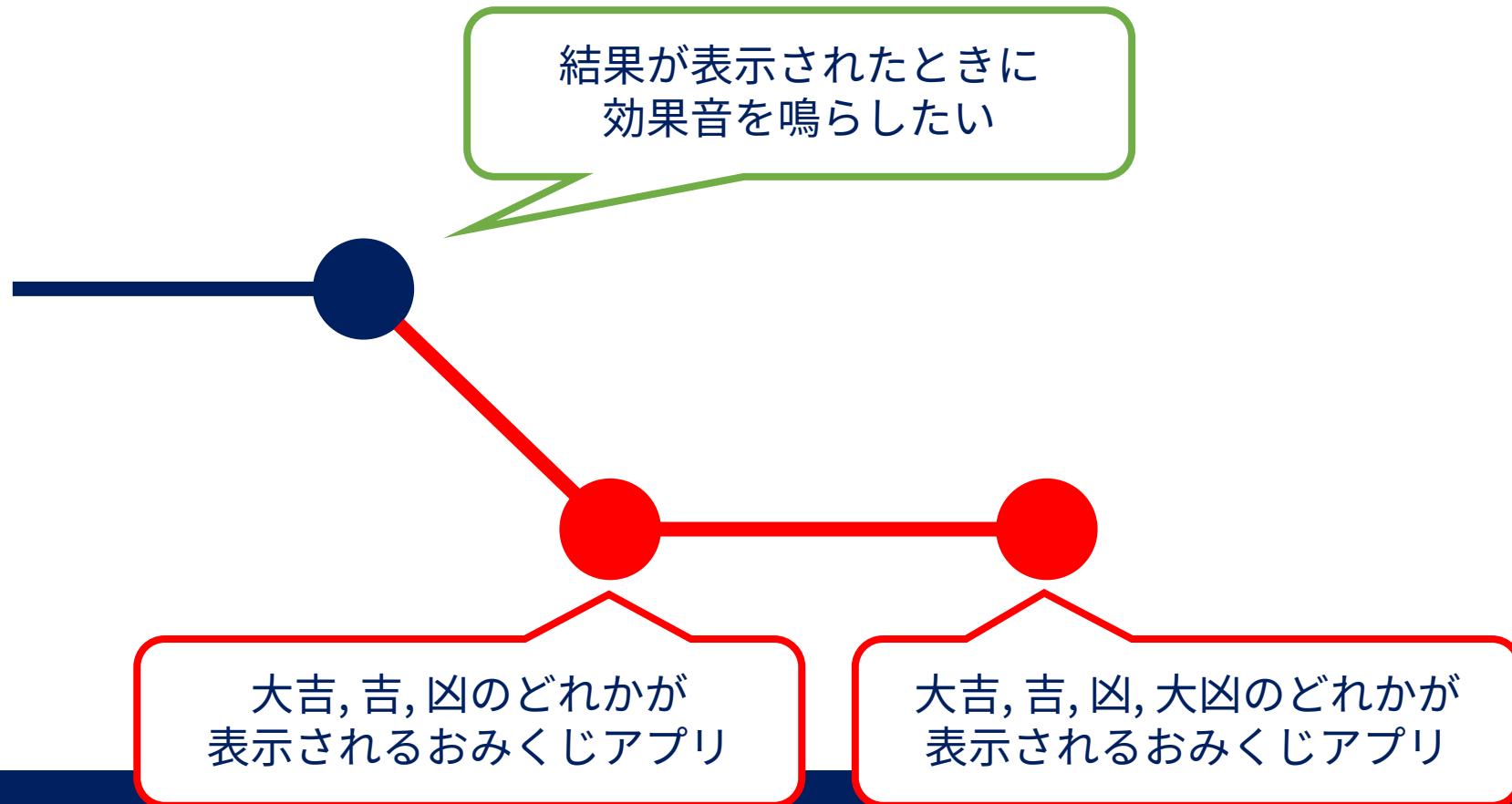
ブランチの例: おみくじアプリ

Git基礎



ブランチの例: おみくじアプリ

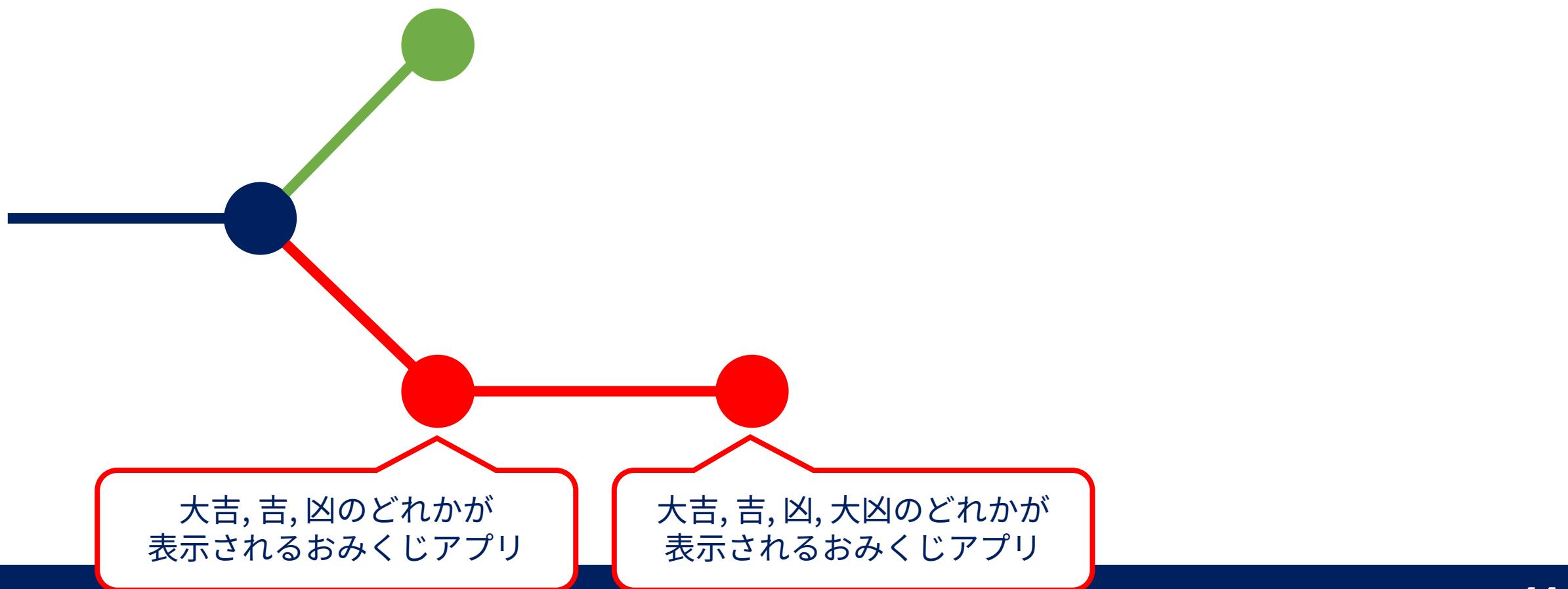
Git基礎



ブランチの例: おみくじアプリ

Git基礎

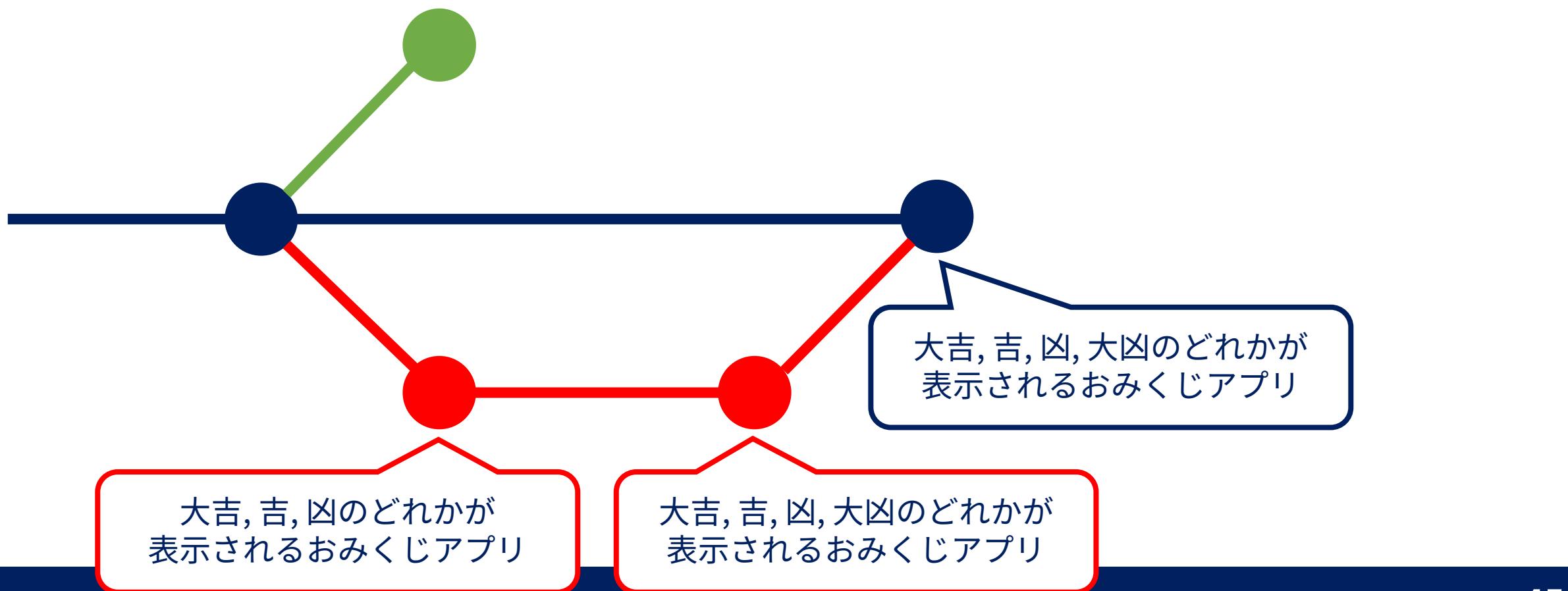
吉と凶のどちらかが表示されて効果音が鳴るおみくじアプリ



ブランチの例: おみくじアプリ

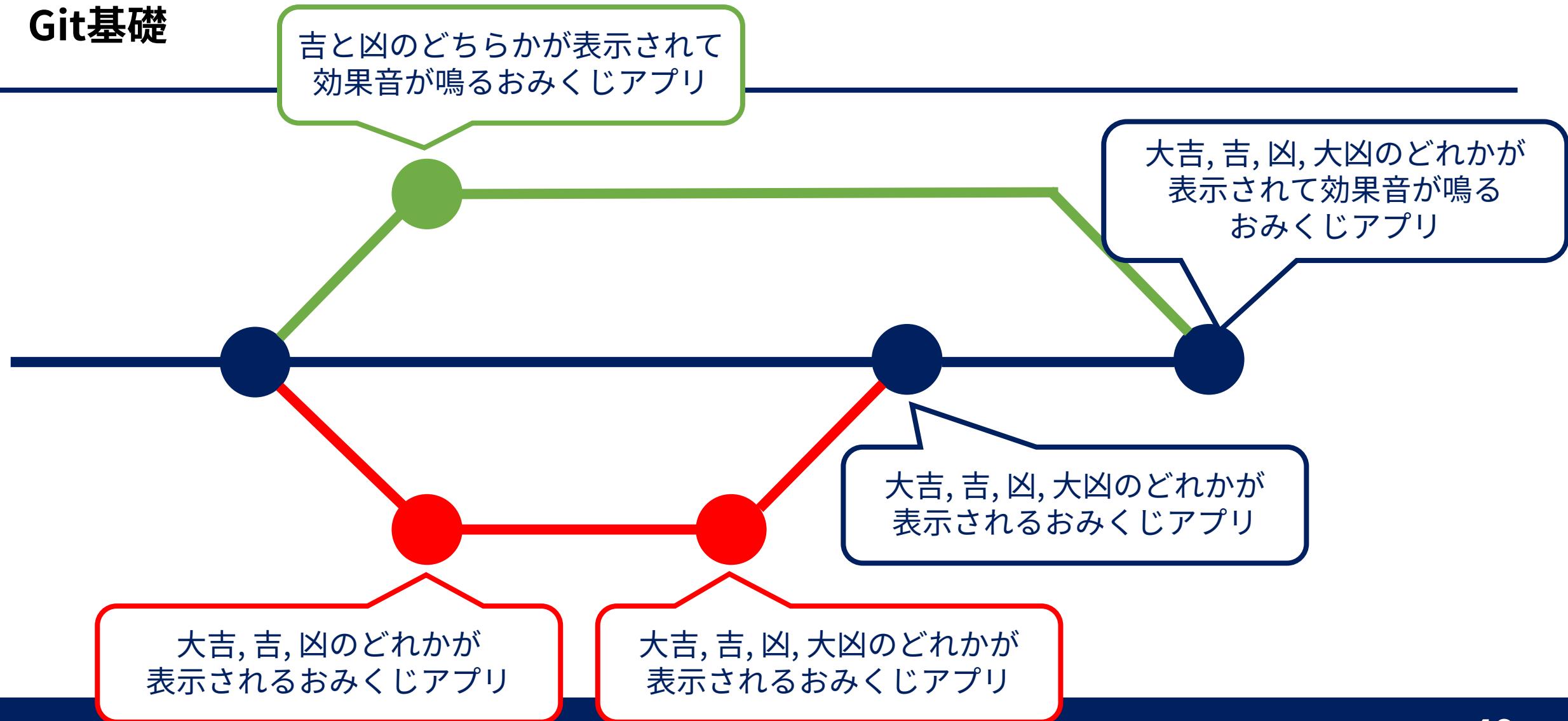
Git基礎

吉と凶のどちらかが表示されて効果音が鳴るおみくじアプリ



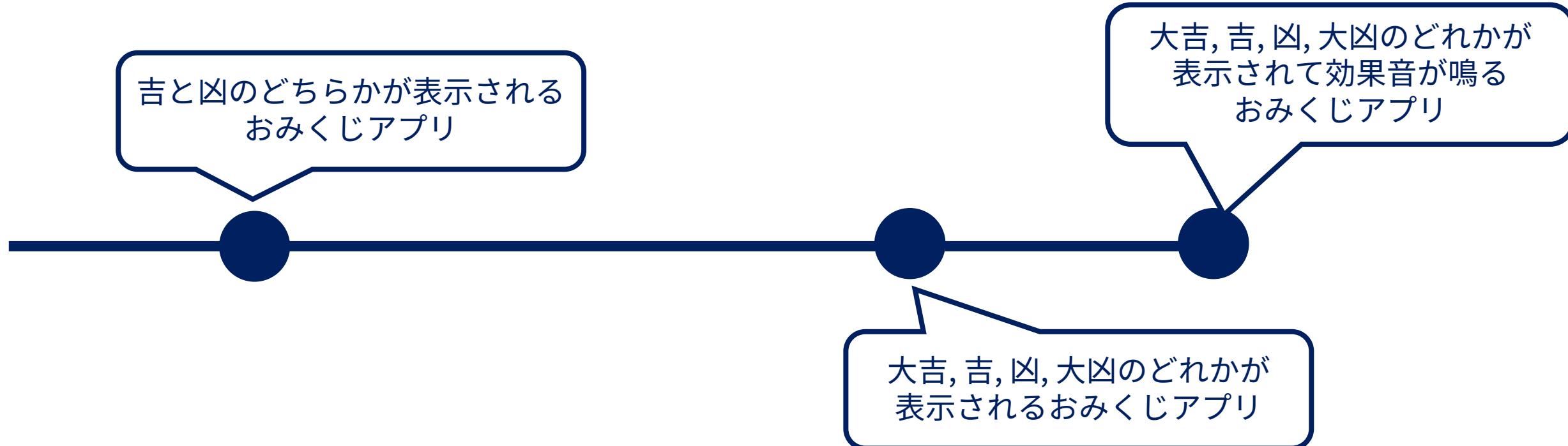
ブランチの例: おみくじアプリ

Git基礎



ブランチの例: おみくじアプリ

Git基礎



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基礎

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

※ [ユーザーネーム] 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。

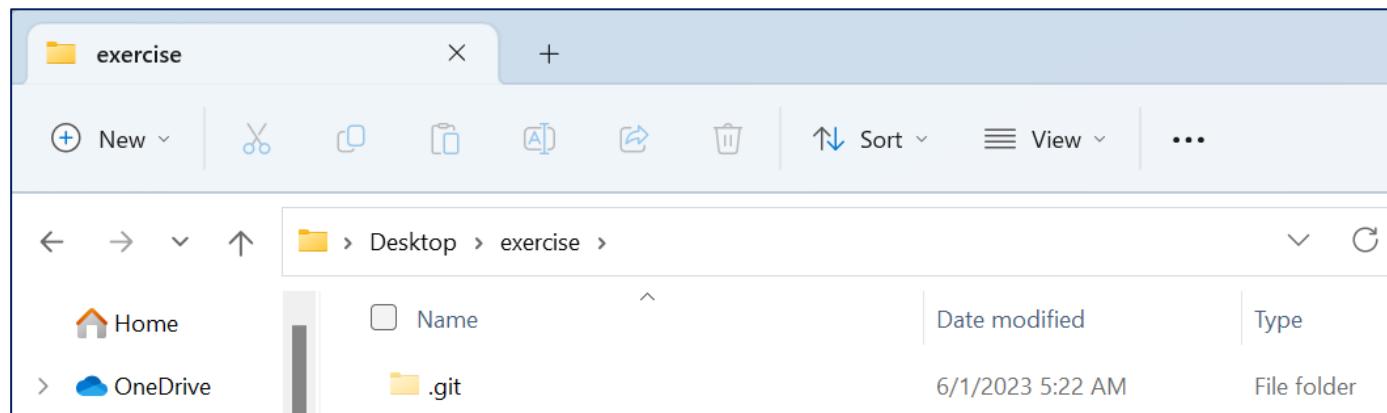
```
git init
```

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。

```
PS C:\Users\y-toshiki\Desktop\exercise> git log  
commit 1840d060ba3fb1a5f621dd65fc35d4fd0dbbc73c (HEAD -> main)
```

コミットハッシュ

コミットしたブランチ

```
: mopisec <contact@mopisec.net>  
Thu Jun 1 05:47:38 2023 -0700
```

コミットした人のユーザー名・メールアドレス

Initial Commit

コミットした日時

コミットメッセージ

※ 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。

The screenshot shows a GitLab project page for 'clone-exercise'. The left sidebar contains links for Project information, Repository, Issues, Merge requests, CI/CD, Deployments, Packages and registries, Monitor, Analytics, Wiki, and Snippets. The main content area displays the project details: 'clone-exercise' (Project ID: 2), 1 Commit, 1 Branch, 0 Tags, and 0 B Project Storage. A commit titled 'Initial commit' by Naoki Takayama is shown, authored just now. Below the commit is a file list: 'main' (clone-exercise) and 'Auto DevOps enabled'. A table lists files: 'hello.txt' with 'Initial commit' and 'Last update' at 'just now'. At the bottom, there's a preview of 'hello.txt' showing the content 'Hello from REMOTE REPOSITORY!!!'.

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

The screenshot shows a password change form titled "新しいパスワードを設定". The left sidebar contains navigation links: イssue, マージリクエスト, To-Do リスト, マイルストーン, スニペット, and アクティビティ. The main form has three input fields: "現在のパスワード" (Current Password) containing "Camp@2023", "新しいパスワード" (New Password), and "パスワードの確認" (Confirm Password). A blue button at the bottom left says "新しいパスワードを設定" (Set New Password). A message at the bottom left says "① パスワードが正常に変更されました" (Password was changed normally). A small "x" icon is in the top right corner.

"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

The screenshot shows the 'Create New Project' process on GitLab. The steps are numbered 1 through 6.

- Step 1:** 'GitLabへようこそ' (Welcome to GitLab). A red circle highlights the 'プロジェクトを作成' (Create Project) button.
- Step 2:** '新しいプロジェクトを作成' (Create New Project). A red circle highlights the '空のプロジェクトを作成' (Create Empty Project) button.
- Step 3:** '空のプロジェクトを作成' (Create Empty Project). The 'プロジェクト名' (Project Name) field contains 'My Remote Repository'. A red box highlights this field.
- Step 4:** '空のプロジェクトを作成' (Create Empty Project). The 'プロジェクトslug' (Project Slug) field contains 'my-remote-repository'. A red box highlights this field.
- Step 5:** 'プロジェクトの設定' (Project Settings). A red box highlights the checkbox for 'リポジトリを初期化し README ファイルを生成する' (Initialize repository and generate README file).
- Step 6:** 'プロジェクトを作成' (Create Project). A red circle highlights the 'プロジェクトを作成' (Create Project) button.

演習: リモートリポジトリを作成する

Git演習

The screenshot shows a software interface for managing a Git repository. On the left, there's a sidebar with various project management and code-related tabs like 'Pin', 'Issues' (0), 'Merge Request' (0), 'Management', 'Planning', 'Code', and 'Build'. The main area is titled 'Student 06 > My Remote Repository'. It displays a message: '新しいナビゲーションへようこそ' (New navigation welcome) with a note about enabling or disabling it. A warning message says: 'SSH鍵をプロフィールに追加しない限り、SSH経由でリポジトリにプルしたりプッシュしたりできません' (If you do not add your SSH key to your profile, you will not be able to pull or push via SSH). Below this, a success message states: "'My Remote Repository' プロジェクトは正常に作成されました" (The 'My Remote Repository' project was created successfully). At the bottom, there's a summary card for 'My Remote Repository' with a lock icon, 'プロジェクトID : 3', and a star count of '0'.

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 AAAAB3NzaC1yc2EAAAQAB[snip]
```

演習: GitLabアカウントにSSH鍵を追加する

Git演習

Student 06 > My Remote Repository

新しいナビゲーションへようこそ

次のいくつかのリリースでは、アバターでいつでも新しいナビゲーションをオンまたはオフにできます。Read more about the [changes](#), the [vision](#), and the [design](#).

詳細 フィードバックを提供

SSH鍵を追加 今後表示しない

'My Remote Repository' プロジェクトは正常に作成されました。

M My Remote Repository ロック プロジェクトID : 3

Star 0

The screenshot shows the GitLab interface for a user named 'Student 06'. The main navigation bar includes icons for dashboard, search, and profile. The left sidebar lists sections like 'My Remote Repository', 'プロジェクトの概要', 'ピン留め', 'イシュー', 'マージリクエスト', '管理', '計画', 'コード', and 'ビルド'. The 'My Remote Repository' section is selected. A prominent message box in the center contains a warning about SSH key addition and a blue button labeled 'SSH 鍵を追加' which is circled in red. Below the message box, a success message states that the project was created successfully. At the bottom right, there are 'Star' and '0' buttons.

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] [リモートリポジトリのパス]
```

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

M My Remote Repository

プロジェクトID: 4

- 6 コミット 1 ブランチ 0 タグ 0 バイト プロジェクトストレージ

Add twitter ID to info.txt 5ab6d438

mopisec 作者 3 時間前

main my-remote-repository / + ファイルを検索 編集 クローン

Auto DevOps が有効です README を追加 ライセンスの追加 変更履歴を追加 CONTRIBUTING を追加 Kubernetes クラスターを追加

Wiki を追加する 連携設定

名前	最新コミット	最終更新
hello.txt	Add information about my older brother.	4 時間前
info.txt	Add twitter ID to info.txt	3 時間前

演習: GitLab上でファイルを作成/編集する

Git演習

The screenshot illustrates the process of creating a new file in a GitLab repository.

Left Panel: My Remote Repository

- Project ID: 4
- Commits: 6
- Branches: 1
- Tags: 0
- Project Storage: 0 bytes

A recent commit is shown:

- Add twitter ID to info.txt
- mopisec 作者 3 時間前

Middle Panel: 新規ファイル (New File)

File path: main / tea.txt

Content of tea.txt:

```
1 Green Tea
2 Milk Tea
3 Black Tea
```

Annotations:

- ファイル名 (File Name):** Points to the file path input field.
- ファイルの内容 (File Content):** Points to the content area of the file.

Bottom Panel: Commit Message

Commit message: Add tea.txt

Target Branch: main

Buttons:

- 変更をコミット (Commit Changes)**: Circled in red.
- キャンセル (Cancel)**

Side Panel: Action Buttons

- ディレクトリ (Directory):
 - 新規ファイル (New File) (highlighted with a red box)
 - ファイルをアップロード (Upload File)
 - 新規ディレクトリ (New Directory)
- リポジトリ (Repository):
 - 新規ブランチ (New Branch)
 - 新規タグ (New Tag)

A red arrow points from the '+ v' button in the main repository panel to the '新規ファイル' button in the side panel.

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。

[Question] Benchmark read speeds #200

 Open Clydeston opened this issue 3 days ago · 1 comment



Clydeston commented 3 days ago

Hello thanks Ufrisk for the amazing tools and libs you've created. Just a quick one, are there any built in ways to perform benchmark tests for example read speeds? If not what would you suggest is the best way to get the best results? I just want to see if some scatter reads are actually faster than multiple separate reads, just wondering what the best way to check this would be.



5.1 Read the data bug #147

 Open DIXIN20 opened this issue on Nov 14, 2022 · 11 comments



DIXIN20 commented on Nov 14, 2022

5.0 Everything is normal. After replacing h, lib and dll of 5.1, it is normal to read a small amount of data. The data can't be read more than once, and then it can be read again a few minutes later. When it can't be read, the device can't be connected after closing the program



1

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

Naoki Takayama > test-project

test-project

プロジェクトID : 1

4 コミット 1 ブランチ 0 タグ

commit by student-01
Student 01 作者 2週間前

main / test-project / +

README Auto DevOps が有効で

イシューを使用して、アイデア、問題解決、作業計画を共同で作業する

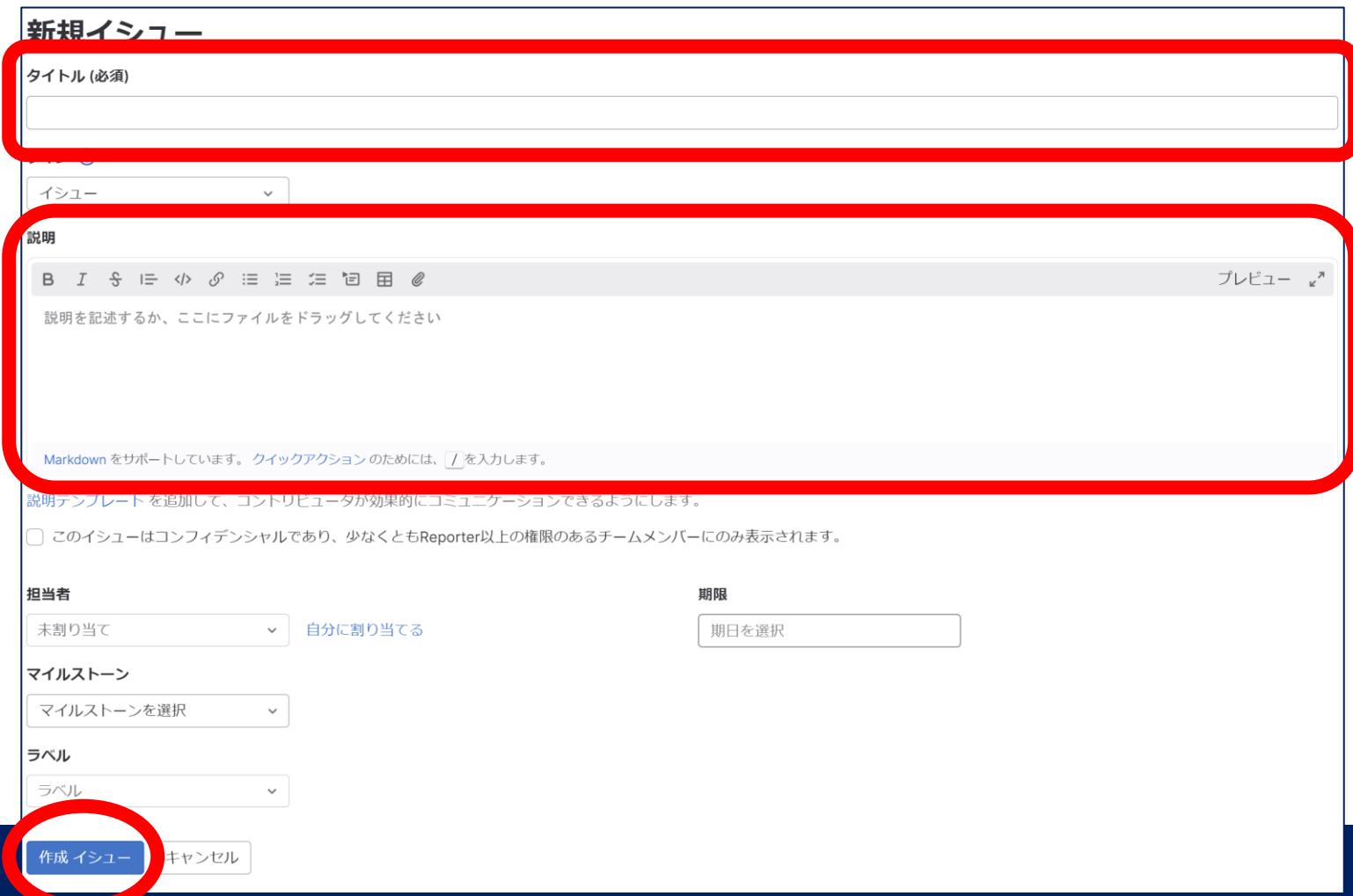
イシューについて詳しく知る

新規イシュー イシューのインポート

イシューの管理にJiraを使いますか？
Jiraの統合を有効にして、GitLab で Jira Issue を表示します。
この機能にはプレミアムプランが必要です。

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

The screenshot shows the GitLab issue creation interface. At the top, there are buttons for 'オープン' (Open) and 'イシューをたった今に作成' (Create issue now), followed by the user 'Naoki Takayama (Owner)'. Below this is a title 'テスト' (Test) and a subtitle 'セキュリティ' (Security). There are upvote, downvote, and share buttons. A 'マージリクエストを作成' (Create merge request) button is also present. The main area contains sections for 'タスク' (Tasks) and 'リンクされたアイテム' (Linked items), both with '0' entries. The 'アクティビティ' (Activity) section includes a rich text editor toolbar and a preview button. A large red arrow points from the 'コメント' (Comment) button at the bottom left to the comment input field at the bottom right, which contains the text 'Naoki Takayama @mopisec · たった今 テストコメント'. The bottom navigation bar includes buttons for 'コメント' (Comment), 'イシューをクローズ' (Close issue), and other issue-related actions.

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。

The screenshot shows a GitHub pull request for issue #968. The PR title is "Windows: add --verbose option for ldrmodules plugin." It has 4 commits, 4 checks, and 1 file changed. A comment from digitalisx on Jun 16 says:

Hello, everyone in the community! :)
This PR comes from this issue (#967).
It will be meaningful to re-implement the original features of volatility.

The Examples section shows command-line output:

```
> python3 vol.py -f case.vmem windows.ldrmodules --verbose
Volatility 3 Framework 2.4.2
Progress: 100.00          PDB scanning finished
Pid      Process Base    InLoad   InInit   InMem   MappedPath   LoadPath     InitPath     MemPath
644      services.exe    0x22ea097000  False    False    False    \Windows\System32\ko-KR\services.exe.mui      N/A
```

A review by ikelos on Jun 16 includes a comment:

Thanks very much, very swift fix to issue #968. I have a few concerns though, we try not to put separate data items into a single field, and where possible (it's not always feasible) we try to not to change the layout of the table based on an input option. I'm happy to do that here if it's necessary, but it might just be worth always outputting the full data all the time, depending on how ugly it gets? Lemme know what you think and we can come up with the best way forward... 5:)

Code reviews show changes in volatility3/framework/plugins/windows/ldrmodules.py:

```
95  +         try:
96  +             if load_mod:
97  +                 load = "{0} : {1}".format(
98  +                     load_mod.FullDllName.get_string(),
```

Another comment by ikelos on Jun 16 suggests changing the default column entry to the full one or outputting a second column if both bits of information are important.

<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

The screenshot shows the GitLab interface for creating a new merge request. The top status bar indicates a push to the 'example-branch' from the 'main' branch. The main area is titled '新規マージリクエスト' (New Merge Request) and shows the following fields:

- タイトル (必須):** 練習用マージリクエスト (highlighted by a red box)
- 説明:** このマージリクエストは練習用です。 (highlighted by a red box)
- 担当者:** 未割り当て (自分に割り当てる button next to it)
- %d レビューア:** 未割り当て
- マイルストーン:** マイルストーンを選択
- ラベル:** ラベル (dropdown menu)
- マージオプション:**
 - マージリクエストが承認されたときにソースブランチを削除します。
 - マージリクエストが承認されたときにスカッシュコミットします。 (with a question mark icon)
- 作成 マージリクエスト** button (highlighted by a red box)

At the bottom, there's a note about Markdown support and a footer message about preview comments.

GitLabでのマージリクエストの作成方法

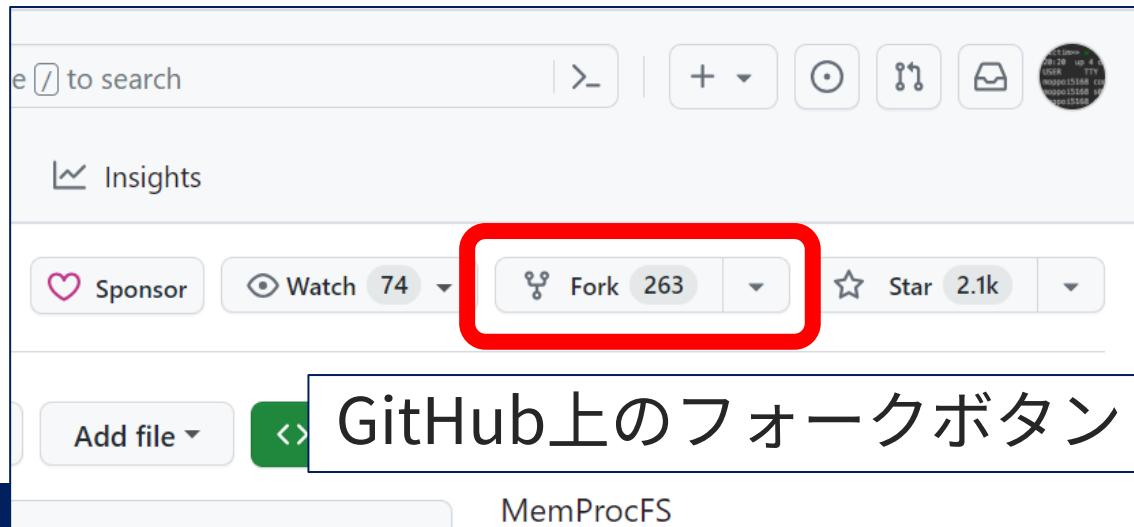
Gitホスティングサービスの機能



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

Student 06 > Fork me please > プロジェクトをフォーク



プロジェクトをフォーク

フォークとはプロジェクトのコピーのことです。リポジトリをフォークすると、元のプロジェクトに影響を与えずに変更することができます。

プロジェクト名: Fork me please
小文字または大文字、数字、絵文字、アンダースコアで始まる必要があります。ドット、プラス、ダッシュ、スペースも含めることができます。

プロジェクトの URL: https://camp.mopisec.net/ mopisec
プロジェクトslug: fork-me-please

いくつかの依存したプロジェクトを同じ名前空間で管理しますか? [グループを作成](#)

プロジェクトの説明(オプション)

表示レベル [?](#)

 非公開
プロジェクトへのアクセス権は、明示的に各ユーザーへ付与する必要があります。このプロジェクトがグループの一部である場合、グループのメンバーにアクセス権が付与されます。

 内部
このプロジェクトはログインしたユーザーからアクセスできます。

 公開
プロジェクトは認証無しにアクセスできます

プロジェクトをフォーク キャンセル

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。

fork元 Student 06 / Fork me please
1コミット先行: 上流リポジトリ。

マージリクエストを作成

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。