



Tokyo Tech

GitHub & Pull Request

2020年10月30日（金）
システム開発プロジェクト応用第一

東京工業大学
特任助教 内田公太

- 実際のシステム開発プロジェクトの現場で使われている現代的な開発ツールや手法を学ぶ
 - 正しいツールや手法の選択はソフトウェア開発を効率的に、そして楽しいものにする

到達目標：

- 現代的な開発ツールの基本的な使い方と適する用途が分かる

- 情報収集
- GDB
- Git
- バグトラッキング
- GitHub & Pull Request
- ユニットテスト
- 継続的インテグレーション
- デプロイと冪等性
- コミュニケーション

自己紹介

- 内田公太
- Twitter @uchan_nos
- 週3日：サイボウズ・ラボ株式会社
週2日：東工大の特任助教
- osdev-jpコアメンバー
- 『30日でできる! OS自作入門』の校正担当
- 『自作エミュレータで学ぶ
x86アーキテクチャ』の著者



スタイル：

- 少し講義して演習，の繰り返し

成績評価：

- 現代の開発技術・手法の理解度を評価する
- 各トピックを受講者自身のソフトウェア開発プロジェクトに適用し，レポートおよびリポジトリを提出する
- レポートおよびリポジトリの充実度で成績を決定する

- 色々な要素がある
 - トピックに対する回答
 - ドキュメント
 - コミットメッセージ
 - プルリクのやり取り
 - Etc.
-
- 総合的に判断して評価します

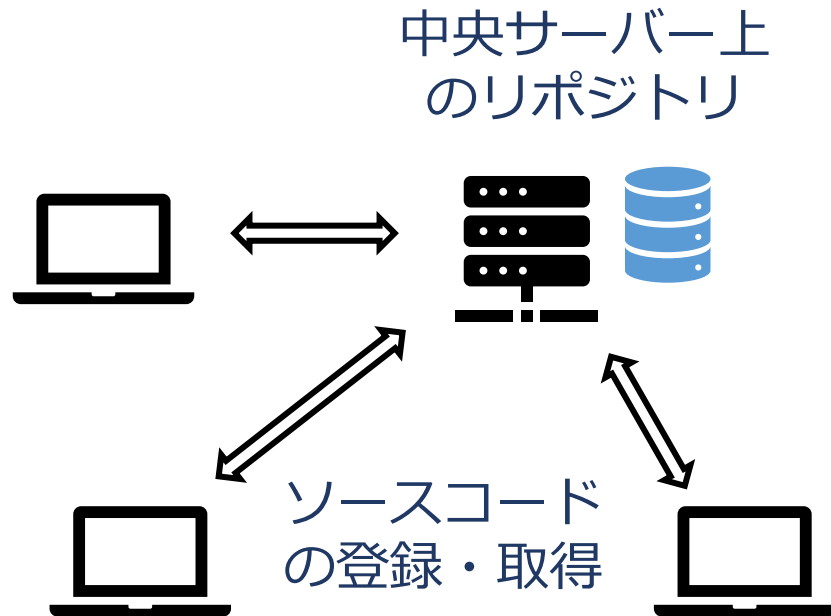
- 課題を含めたリポジトリとレポートを作成し, 提出
- 初回 (10/2) 説明したので詳しい話はしないつもり
 - 改めて聞きたい方がいたらお知らせください



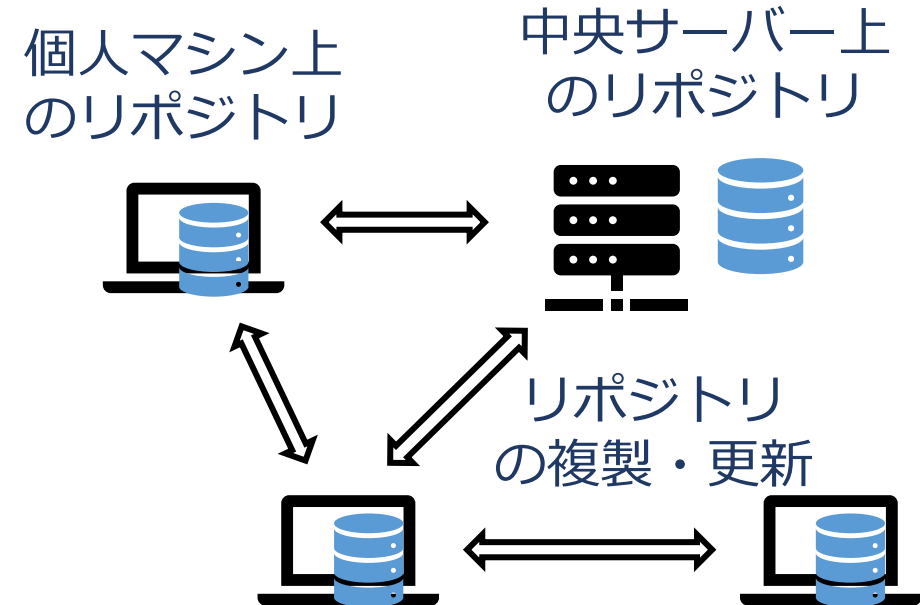
Tokyo Tech

GitHub & Pull Request

Gitは分散型のバージョン管理



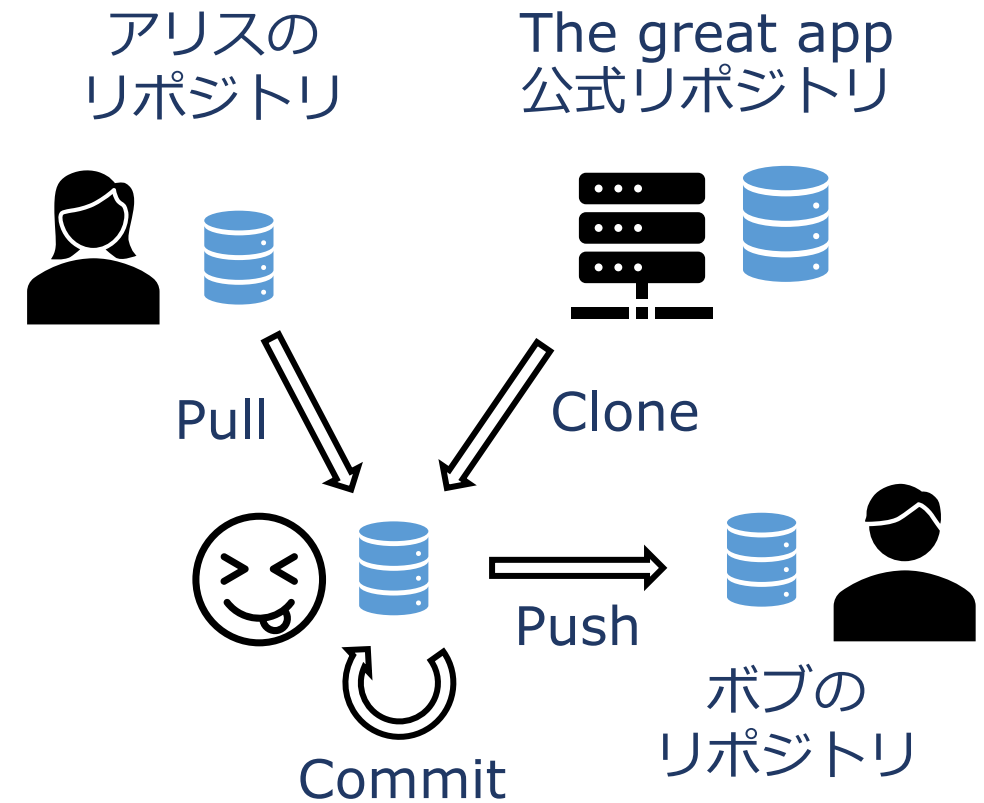
中央集中型
バージョン管理



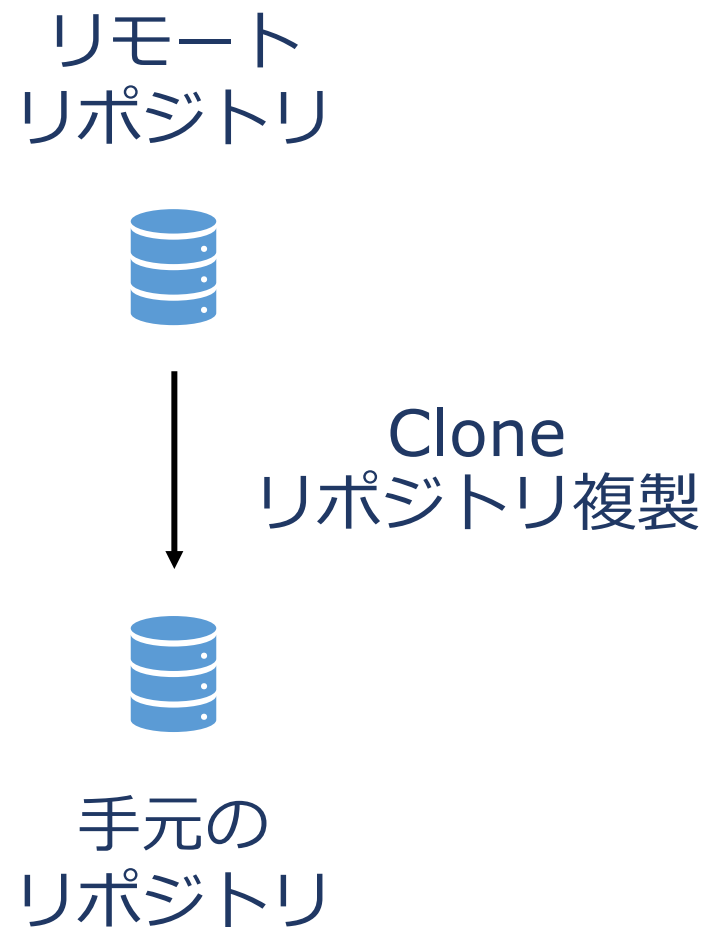
分散型
バージョン管理

Gitを用いた開発の流れ

- 開発対象となる“The great app”のリポジトリを複製
- 開発し，変更をcommit
- アリスが便利なAPIを追加したらしい！使いたいのので取り込もう！
- 素晴らしい機能が完成した！ボブが欲しがっているので送ってあげよう！



- リポジトリを複製
 - コミットオブジェクト, ブランチ, タグなどすべてが複製される
- リモートリポジトリとして**origin**が設定される
- リモートと言いつつ, ローカルマシン内での複製も可能



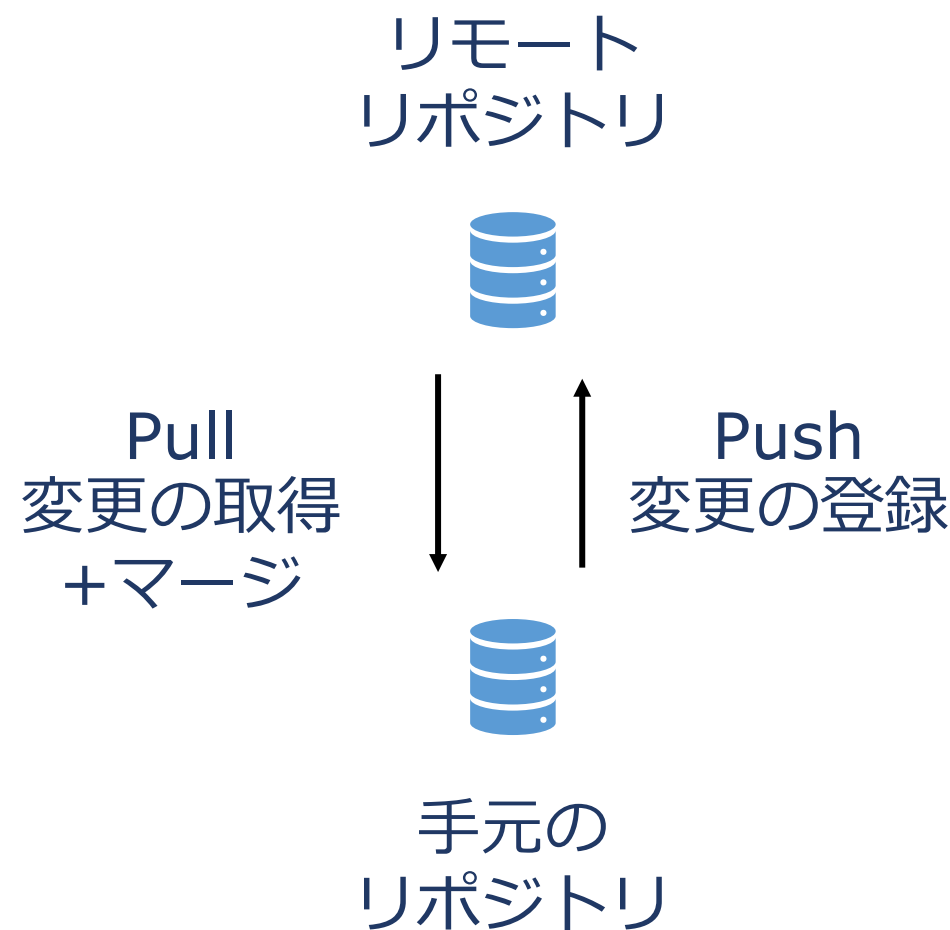
- 追跡されているリモートリポジトリを表示・管理する
- リモートリポジトリ=自分の外のリポジトリ
 - ローカルマシンにある別のリポジトリも含む
- `git remote -v`
 - 追跡されているリモートリポジトリを一覧する
- `git remote add alice john@alice.com:great-app.git`
 - `alice.com`上の`great-app`リポジトリを`alice`という名前で追加
 - 使える通信プロトコルはLocal, HTTP, SSH, Git
 - <https://git-scm.com/book/en/v2/Git-on-the-Server-The-Protocols>

ローカルマシン内のGitリポジトリを
別のディレクトリに複製し
originを確認せよ

- `cd /path/to/somewhere`
- `git clone /path/to/myproj cloned`
- `cd cloned`
- `git remote -v`
- 制限時間10分

Push & Pull

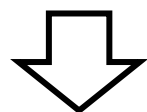
- Pushはリモートに変更を送る
 - Pushを禁止しているリポジトリもある
- Fetchはリモートから変更を取ってくる
- Pull=fetch+merge
 - git fetch
 - git merge FETCH_HEAD



複製元
リポジトリ
(myproj)

```
$ git log refs/  
refs/heads/br-a      refs/heads/br-b  
refs/heads/master    refs/tags/tag-a
```

Tabを2回で補完



Clone

複製先
リポジトリ
(cloned)

```
$ git log refs/  
refs/heads/master      refs/remotes/origin/br-b  
refs/remotes/origin/HEAD refs/remotes/origin/master  
refs/remotes/origin/br-a refs/tags/tag-a
```

- ブランチやタグの正式名称はrefs/...
- リモートブランチの略記 : origin/br-a

- `git fetch <repo> <refspec>`
- `refs/remotes/...`を更新 + `FETCH_HEAD`を設定
- Pull=fetch+merge
 - `git fetch`
 - `git merge FETCH_HEAD`

```
$ cat .git/FETCH_HEAD
2c43fc8c947ab6b660eccdef7e4c44177fcaa48a
/home/uchan/workspace/./myproj
1925598567a8adef11b65c04d02249735f9629c3
'br-a' of /home/uchan/workspace/./myproj
E526eb9858aad877f1131f57659b0fdf11ace71b
'br-b' of /home/uchan/workspace/./myproj
```

branch 'master' of

not-for-merge branch

not-for-merge branch

Assume the following history exists and the current branch is "`master`":

```
      A---B---C master on origin
      /
D---E---F---G master
      ^
  origin/master in your repository
```

Pull前はorigin/masterが
古い状態になっている

Then "`git pull`" will fetch and replay the changes from the remote `master` branch since it diverged from the local `master` (i.e., `E`) until its current commit (`C`) on top of `master` and record the result in a new commit along with the names of the two parent commits and a log message from the user describing the changes.

Replay the
changes

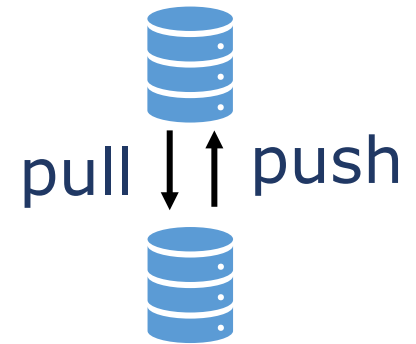
```
      A---B---C origin/master
      /       \
D---E---F---G---H master
```

Record the
result

ローカルマシン内で以下を実験せよ
リモートリポジトリにcommitしてpull
複製したリポジトリにcommitしてpush

- `git pull alice main`
 - aliceリポジトリのmainブランチをpull
- `git push alice dev`
 - devブランチをaliceリポジトリへpush
- 制限時間15分

リモート
リポジトリ

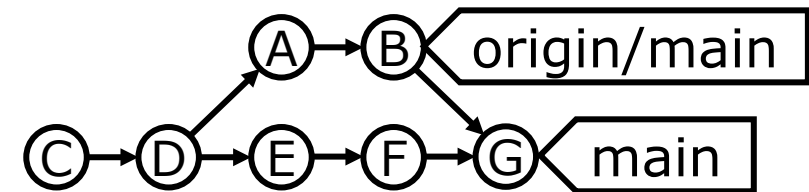


複製した
リポジトリ

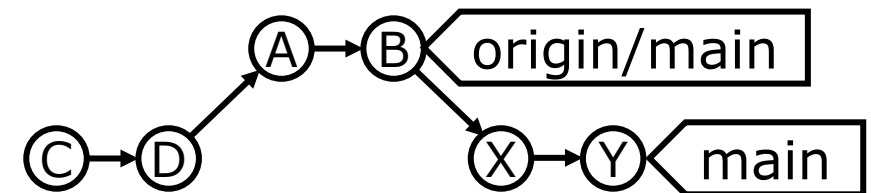
- Pull : リモートブランチを持ってきて**マージする**
 - リモートリポジトリに対するRead権限だけあればよい
 - 手元で変更が加えられていても良い
- Push : リモートへブランチを送信する
 - リモートリポジトリに対するWrite権限が必要
 - リモートで変更が加えられていると失敗する
“! [rejected] dev -> dev (non-fast-forward)”
- ある変更を開発者間で伝達する : Pullが基本
 - Pull Requestは存在するが, Push Requestは無い

Pullの戦略は2つ

- デフォルトの戦略：マージ
- Fetchにより取得した更新を、現在のブランチにマージ
- 2つ目の戦略：リベース
 - `git pull --rebase`
- Fetchにより取得した更新を、リベースにより適用
- 歴史が直線的になる



Gはマージコミット



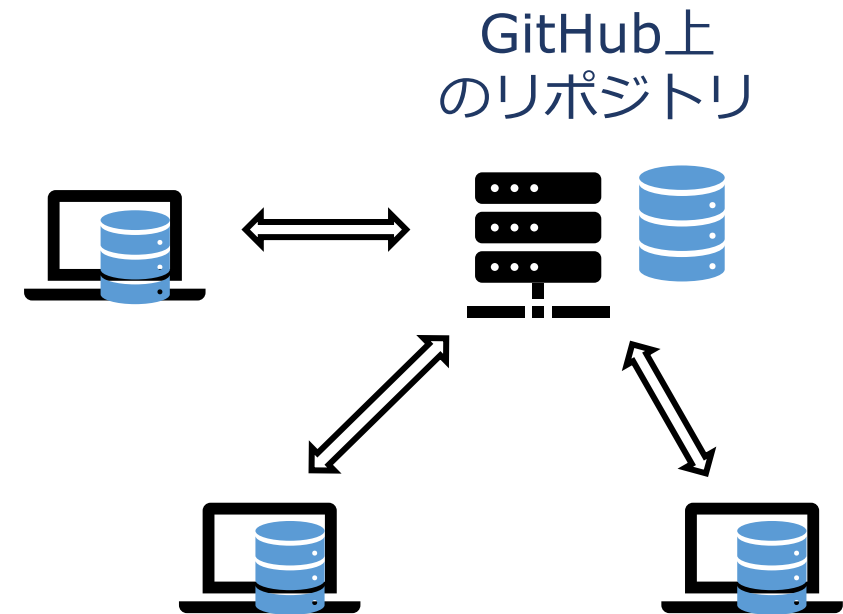
$X=E'$, $Y=F'$

複製したリポジトリ間で
Pullの2つの戦略を実験せよ

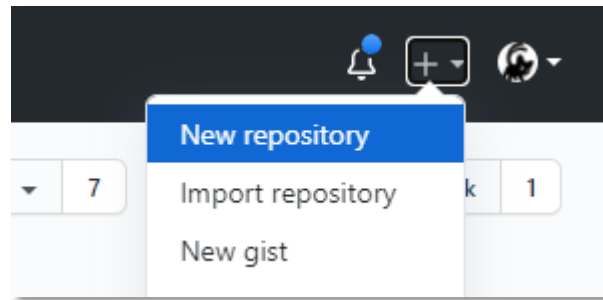
- 片方のリポジトリにだけ変更を加えた場合
- 両方のリポジトリを変更した場合
- 競合が発生しないように実験するのが楽
 - 別ファイルを編集すれば安全
- 制限時間15分

- 今日の主題
- Gitを核とした, 開発プロジェクトのためのサイト
 - Gitリポジトリ
 - リポジトリのフォーク (clone+a)
 - Pull Request
 - バグトラッキングシステム (Issues)
 - Wiki
 - ビルド自動化支援
 - プロジェクト管理
- 2018年にMicrosoftがGitHub社を買収

- GitHubにより中央集中型と同様の管理が可能
- 各個人は手元にGitHub上のリポジトリを複製
 - 更新の登録・取得は手元のリポジトリが対象
 - 手元のリポジトリをGitHub上のリポジトリへ反映
- 互いのマシン同士が疎通しなくても大丈夫



リポジトリをGitHubにアップロード



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

```
git remote add origin git@github.com:uchan-nos/myproj.git
git branch -M main
git push -u origin main
```



Create a new repository

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

Repository template

Start your repository with a template repository's contents.

No template ▾

Owner *





uchan-nos ▾

Repository name *

myproj ✓

Great repository names are myproj is available. ble. Need inspiration? How about special-dollop?

- ☒  **Public**
Anyone on the internet can see this repository
- ☐  **Private**
You choose who can see and commit to this repository

Initialize this repository with:

Skip this step if you're importing an existing repository

- ☐ **Add a README file**
This is where you can write a long description for your repository
- ☐ **Add .gitignore**
Choose which files not to track from a list of templates
- ☐ **Choose a license**
A license tells others what they can and can't do with your code

すべてチェック
をはずす

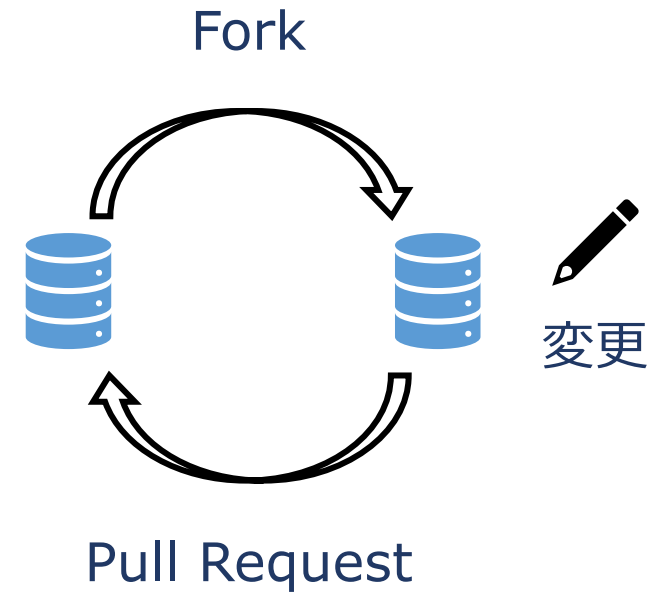
GitHubで空のリポジトリを作り
手元で構築したリポジトリをpushせよ

- GitHubに表示される3行のコマンドの意味を理解しよう
 - `git remote add origin git@...`
 - `git branch -M main`
 - `git push -u origin main`
- 制限時間10分

- 本流（メインストリーム）から派生する機能
- フォークする理由は様々
 - 本流のメンテが止まってしまった
 - 開発方針の相違により分家する
 - 本流をベースに差分開発をしたい
- Gitのcloneに似ている
 - 違い1：フォーク先のページにフォーク元が表示される
 - 違い2：フォーク元からフォーク先のリポジトリに飛べる
 - 違い3：フォーク先からフォーク元へのPRが簡単に作れる

Pull Request (プルリク)

- 変更を取り込んでもらう (pull)
- 要請 (request)
- GitHub独自の仕組み
- Gitのブランチとマージの仕組みを利用して実現



- ビルド自動化支援の仕組み
- Pushなどのタイミングで, GitHub側で任意のコマンドを実行できる
- ビルドして, 成果物をサイトに公開するまでを自動化
→ 継続的インテグレーション, 継続的デリバリー
- 実際のリポジトリで説明
 - WebAssembly on Rust for kintone customize
<https://github.com/uchan-nos/rust-wasm-kintone>

レポート提出のトピック名

- 10/30の前半はTOPIC=bts
- 10/30の後半はTOPIC=github



Tokyo Tech

GitHub & PullRequest 後編

11/6 (金) 14:20-16:00

- 課題を含めたご自身のリポジトリとレポートを提出
- 提出先は内田のGitHubリポジトリ
 - <https://github.com/uchan-nos/titech-sysdev-2020>
 - プライベートリポジトリのためアカウント登録必須
皆さんのGitHubアカウントを教えてください
- このリポジトリに対し、レポートを送る
 - レポートには、トピックに対する回答を含める
- 提出期限は講義の1週間後の10:00 (JST)

1. 独自のブランチを作る

1. titech-sysdev-2020:master

↓ branch

titech-sysdev-2020:report-YOUR_NAME

2. 回答の概要をまとめたファイルを加える

1. titech-sysdev-2020/reports/TOPIC/YOUR_NAME.md

2. Commit & Push

3. プルリクを送る (リポジトリ内プルリク)

1. titech-sysdev-2020:report-YOUR_NAME

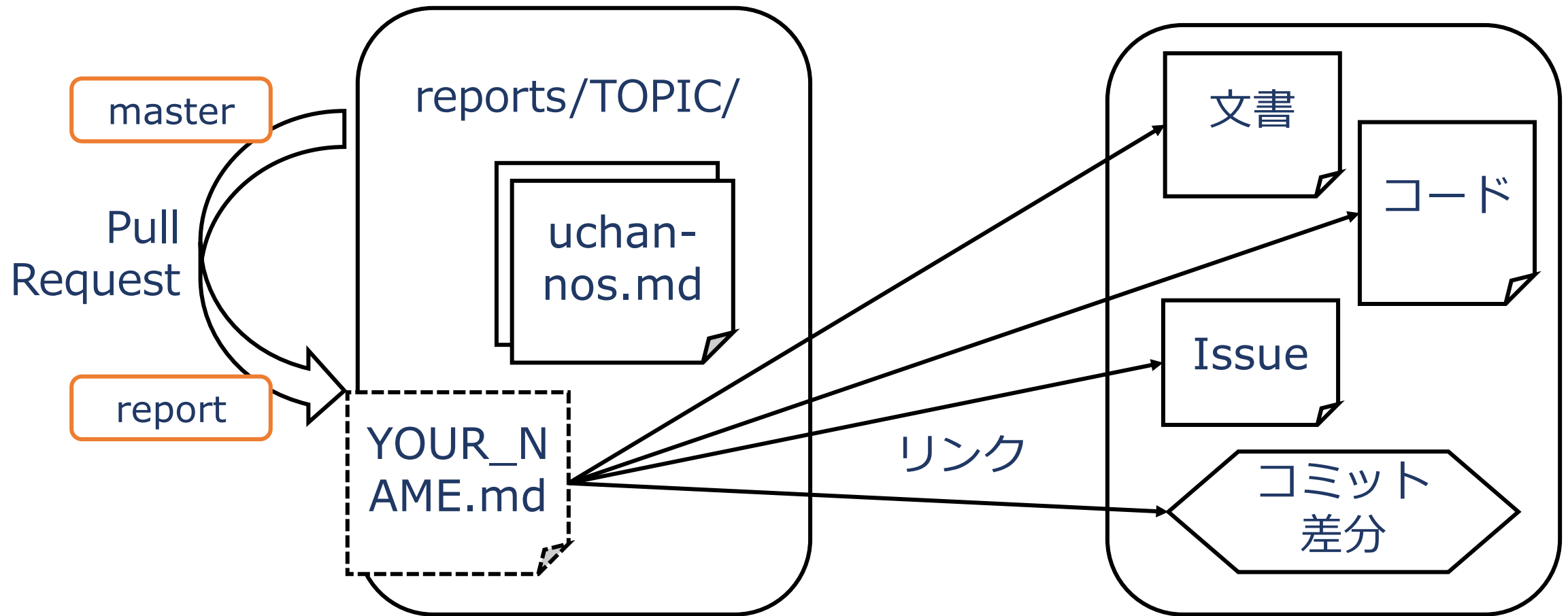
↓ pull request

titech-sysdev-2020:master

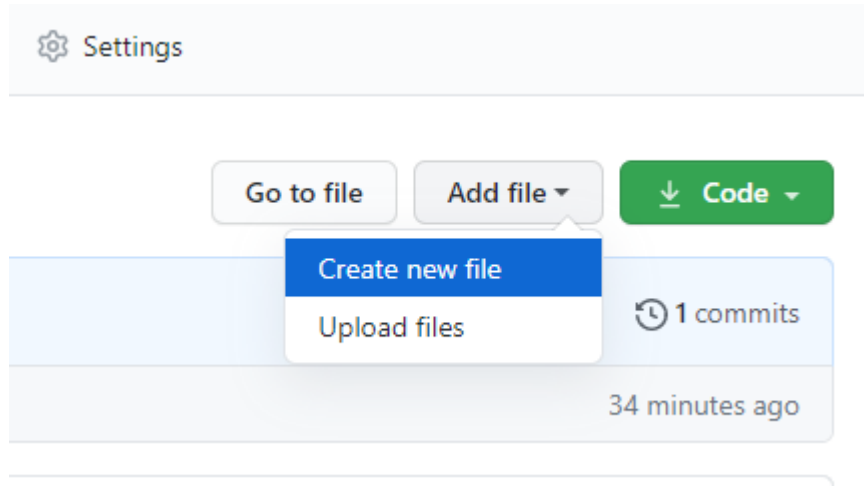
- reports/TOPIC/YOUR_NAME.md
- このファイルに課題への回答を記載する
- 必要なら以下のものを含める
 - Issueへのリンク
 - コミット差分へのリンク
[https://github.com/HOGE/REPO/
compare/COMMIT1...COMMIT2](https://github.com/HOGE/REPO/compare/COMMIT1...COMMIT2)
 - その他
- 要するに、成績評価に必要な情報をYOUR_NAME.md自体に記載するか、そこから辿れるようにする

uchan-nos/titech-sysdev-2020

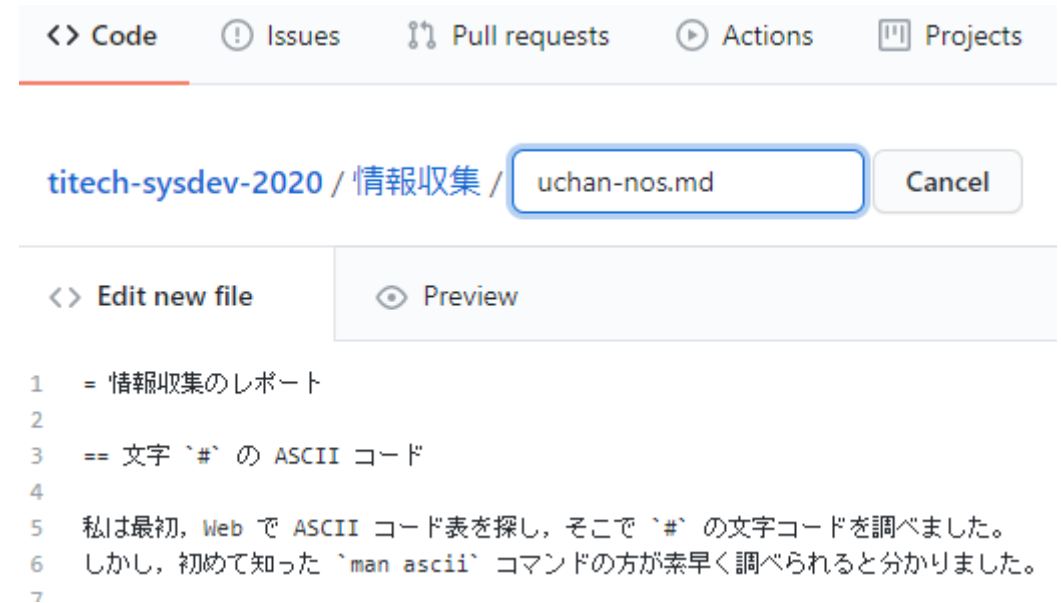
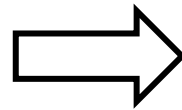
受講者のリポジトリ



レポートの送り方 1/2



ファイルを新規作成



YOUR_NAME.mdの内容を記述

レポートの送り方 2/2

- ☐ Commit directly to the `master` branch.
- ☒ Create a new branch for this commit and start a pull request. [Learn more](#)

report-uchan-nos

Propose new file

Cancel

新規ブランチにコミット

プルリクを作成

Open a pull request

The change you just made was written to a new branch named `report-uchan-nos`. Create a pull request to



base: master



compare: report-uchan-nos

✓ Able to merge. These branches can be merged



レポート提出 uchan-nos

Write

Preview

H B I ≡ <> 🔗 ≡ ≡ ☑ @ ↗ ↶

「情報収集」に関するレポートを提出します

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



Create pull request

レポートの送り方 2/2

☐ Commit directly to the `master` branch.

☒ Create a new branch for this commit and start a pull request. [Learn more](#)

`report-uchan-nos`

Propose new file Cancel

新規ブランチにコミット

プルリクを作成

Open a pull request

The change you just made was written to a new branch named `report-uchan-nos`. Create a pull request to

`base: master` ← `compare: report-uchan-nos` ✓ **Able to merge.** These branches can be merged.

作成したブランチから
masterへのプルリク
となっている！

「情報収集」に関するレポートを提出します

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

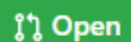
Create pull request

レポートの受理



Tokyo Tech

レポート提出 uchan-nos #1



Open

uchan-nos wants to merge 1 commit into master from re

Conversation 0

Commits 1

Checks 0



uchan-nos commented 4 minutes ago

「情報収集」に関するレポートを提出します

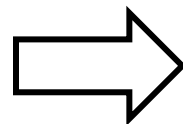


Create uchan-nos.md

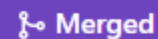


uchan-nos commented now

内容が薄いですよ。もっと付け足しませんか？



レポート提出 uchan-nos #1



Merged

uchan-nos merged 1 commit into master from report-u

Conversation 0

Commits 1

Checks 0



uchan-nos commented 5 minutes ago

「情報収集」に関するレポートを提出します



Create uchan-nos.md



uchan-nos commented 1 minute ago

内容が薄いですよ。もっと付け足しませんか？



uchan-nos merged commit 032af9a into master now

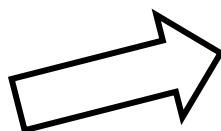
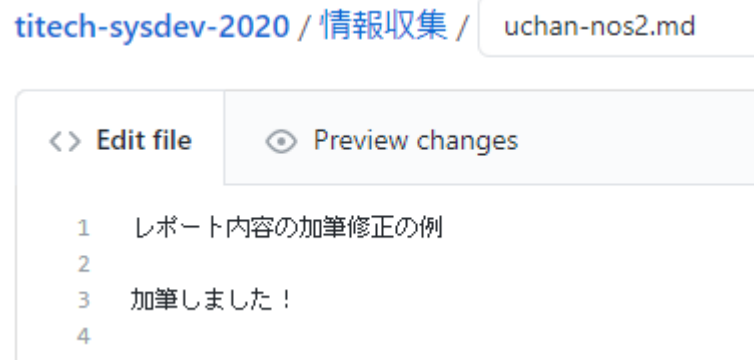
プルリクにコメントが付くことも

Merged : レポート受理済み

レポートの更新 1/2

レポートに不十分な個所があった！

まだマージされてないときの
更新方法を紹介



Commit changes

加筆

Add an optional extended description...

uchan0@gmail.com

Choose which email address to associate with this commit

☒ Commit directly to the `report-uchan-nos` branch.

☐ Create a new branch for this commit and start a pull request. [Learn more](#)

納得いくまで加筆修正

report-Xブランチにコミット

不十分な内容のレポートを作成 #4

[Open](#)uchan-nos wants to merge 2 commits into `master` from `report-uchan-nos`

Conversation 0

Commits 2

Checks 0

Files changed 1



uchan-nos commented 2 minutes ago

これは不十分なレポートなので、まだマージしないでください！



不十分な内容のレポートを作成



加筆



uchan-nos commented now

レポート完成しました。マージして大丈夫です。

一発目のコミット



Pull Requestに
コミットが追加されていく



- GitHubのWebインターフェースを使う必然性はない
- コマンドラインで作業してもよい
 - 具体的なコマンドラインは示しません
 - この講義は「情報収集」でしたね？
 - コマンドラインについて情報収集すれば、レポートをさらに充実させるネタになりますよ

- 次回は11/06（金） 14:20から
- 「GitHub & Pull Request」の続きと「ユニットテスト」をやります