継続的インテグレーション

2020年11月13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

本講義の概要

- 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

講義で扱うトピック

- 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 →

講義の進め方

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

レポート，リポジトリの充実度

- 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

課題提出の方法

- 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

継続的インテグレーション

インテグレーション

- 「複数の異なる要素を組み合わせて一つにしたり、一体として機能するよう調整すること。」
  - IT用語辞典 e-Wordsより

- プログラム部品（関数，クラス，モジュール，コマンド，サブシステム，…）を組み合わせて1つのシステムを組む
- SI: System Integrator
  - サブシステムを集め，全体として機能するよう調整して納入する業態の企業

継続的インテグレーション

- CI: Continuous Integration
- 変更をコミットするたびに自動化されたビルドとテストを実行すること
- ソフトウェアを常に動作可能に保つ戦略
  - バグにいち早く気づく
  - 影響が小さいうちに直す

- 自動化されたビルドとテストのみ実施可能
  - 手動手順を極力少なくするのがCIの効果を高めるコツ

CIを支援するツール

- Jenkins
  - CIツールの元祖
  - 自前サーバーにインストールして使う
- TravisCI/CircleCI
  - GitHubなどで手軽に使える
  - 小規模な使用なら無料
  - 設定例が豊富
- GitHub Actions
  - GitHub公式のビルド自動化システム
  - Issue/PRなどとの連携性が高い

CircleCI

- https://circleci.com
- テスト，ビルド，デプロイをカバー
- Dockerイメージを用いた柔軟なプロセス設定
  - 対象の言語・ライブラリ用のDockerイメージを指定可能
  - その分，設定ファイルは複雑化…
- 小さいプロジェクトなら無料で使える
  - 無料枠
  - 1vCPUと2GBメモリで500分/週
  - 2vCPUと4GBメモリで250分/週

Docker

- 有名なコンテナエンジン
  - コンテナ：Linuxカーネルの各種の隔離機能を用いて，他プロセスから隔離された実行環境を提供する技術
- コンテナエンジン=コンテナイメージの中でプロセスを動作させる
- コンテナイメージ=ファイルシステムを固めたもの
- 1つのLinux上に複数の仮想Linux環境を構築
  - VMとは異なる技術
  - VMよりオーバーヘッドが小さく，仮想化要素が少ない
- Docker入門 https://speakerdeck.com/cybozuinsideout/2019-10-docker

CircleCIとDockerイメージ

- CircleCIでは任意のDockerイメージを指定可能
  - DockerHubなどで配布されているイメージが使える
- 各種Dockerイメージ
  - cimg/ruby:2.6.5
    CircleCIが提供するRuby用イメージ
  - quay.io/cybozu/ubuntu:20.04
    最小構成のUbuntu
  - quay.io/cybozu/ubuntu-dev:20.04
    build-essentialパッケージを追加
  - quay.io/cybozu/golang:1.15-focal
    ubuntu-devにGolangパッケージを追加

ドメイン名が無いものは
DockerHubのイメージ

演習1

CircleCIを使っているOSSを2つ探し,
CircleCIの設定を読み,
何をしているか調査せよ

- CircleCIの設定は.circleci/config.ymlにある
- どのDockerイメージを使い
- どんなコマンドを実行しているだろうか

- 制限時間10分

演習2

自分のOSSにCircleCIを導入し,
Push時にハローワールドを表示させよ

- CircleCIのログに”hello, world”と表示させる
- CircleCI 2.0のGetting Started
  - https://circleci.com/docs/ja/2.0/getting-started/

- 制限時間15分

CircleCIの設定例

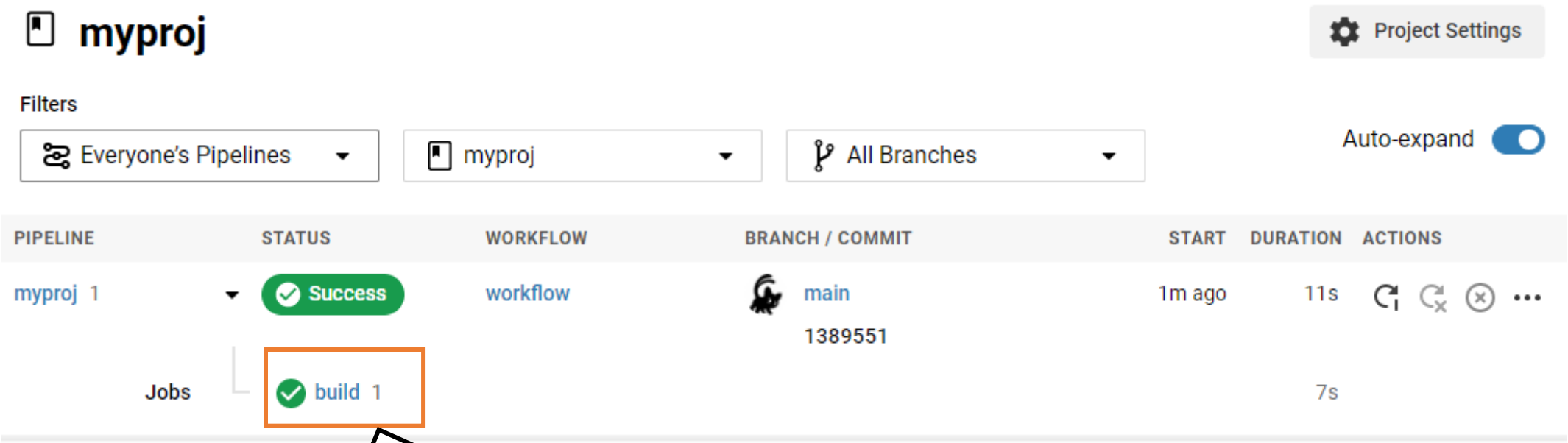
.circleci/config.yml

```
version: 2
jobs:
  build:
    docker:
      - image: quay.io/cybozu/ubuntu-dev:20.04
    steps:
      - checkout:
      - run: echo "Hello, world!"
```

- uchan-nos/myprojの設定例

- https://github.com/uchan-nos/myproj/blob/138955103b6b00282421239a50cc87ce5bce6ddd/.circleci/config.yml

CircleCIのログを見る



ここをクリックすると
ビルドログを見られる

Tokyo Tech

build Success

Rerun

Duration / Finished	Queued	Executor	Branch	Commit	Author
7s / 1m ago	0s	Docker Medium	main		Kota UCHIDA

STEPS TESTS ARTIFACTS

1 / 4 parallel runs

- Spin up environment 6s
- Preparing environment variables 0s
- Checkout code 0s
- echo "Hello, world!" 0s

```
#!/bin/bash -eo pipefail
echo "Hello, world!"

Hello, world!
CircleCI received exit code 0
```

echoコマンドが
実行されている！

演習3

CircleCIを用いて,
自分のOSSのテストを自動で実施せよ

- CircleCIの設定にテストを実行するステップを加える
- 失敗するテストを追加したらどうなるだろう？

- 制限時間30分

UIのテスト

- UI：ボタンやテキストボックス等
- 関数やAPIのテストのように自動化しにくい
  - ボタンのクリックやキーの押下などの手動操作が必要
- プログラムからUIの操作ができれば自動化できそう
  - →Headlessブラウザ
  - →Selenium

- デザインが崩れる，操作感が悪いなど，人の感性が必要なテストは自動化が困難

Headlessブラウザ

- UIを持たないブラウザ
  - 例：Google ChromeのHeadlessモード
  - https://developers.google.com/web/updates/2017/04/headless-chrome?hl=ja
- コマンドラインから操作できる
- UIが不要な目的に最適
  - 実際のWebページに対してテストする
  - WebページをPDFとして印刷する
- 実行方法
  - $ chrome --headless
  - $ firefox -headless

Selenium

- Webブラウザをプログラムから操作するためのツール・ライブラリ
  - 主にテスト目的で開発されているが，それ以外にも応用可
- HTML要素の選択
  - `driver.find_element(By.ID, "cheese")`
- キーボード入力
  - `driver.find_element(By.NAME, "name").send_keys("Charles")`
- クリック
  - `driver.find_element(By.CSS_SELECTOR, "input[type='submit']").click()`

ビルドアーティファクト

- ビルドに伴い生成されるファイル群のこと
  - ビルド済みバイナリや設定ファイル，データファイルなど
  - APTパッケージやtarballもアーティファクトの一種
- アーティファクトをリリースしておくと，ユーザー自身がビルドする必要がなくなるので便利
- CIにより自動ビルドされるタイミングで，アーティファクトを生成，自動リリースしたい

演習4

CircleCIを用いてアーティファクトを
自動でアップロードせよ

- ビルド アーティファクトの保存 - CircleCI
  - https://circleci.com/docs/ja/2.0/artifacts/

- 制限時間20分

レポート提出のトピック名

- 11/13の前半はTOPIC=ut
- 11/13の後半はTOPIC=ci

課題提出

- 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）

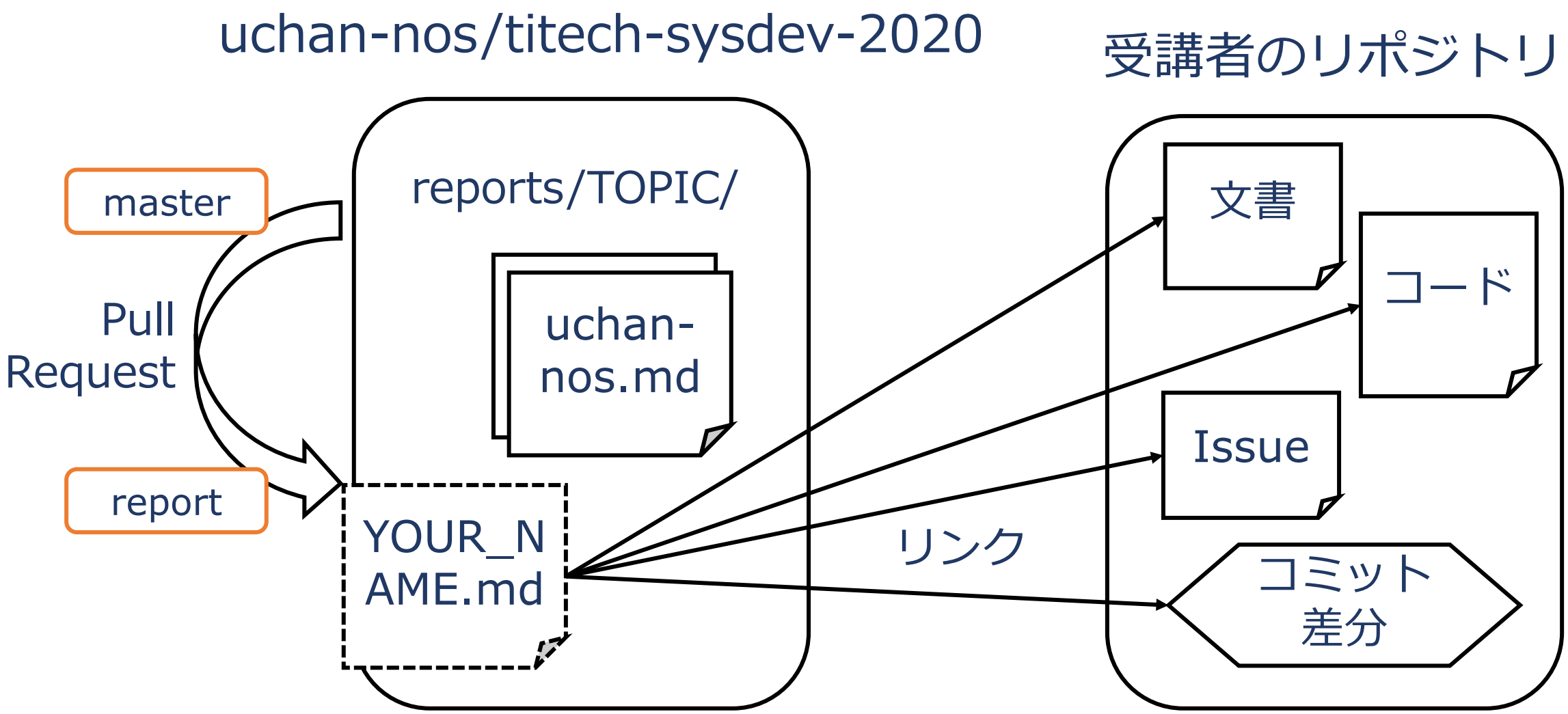
レポートの送り方

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

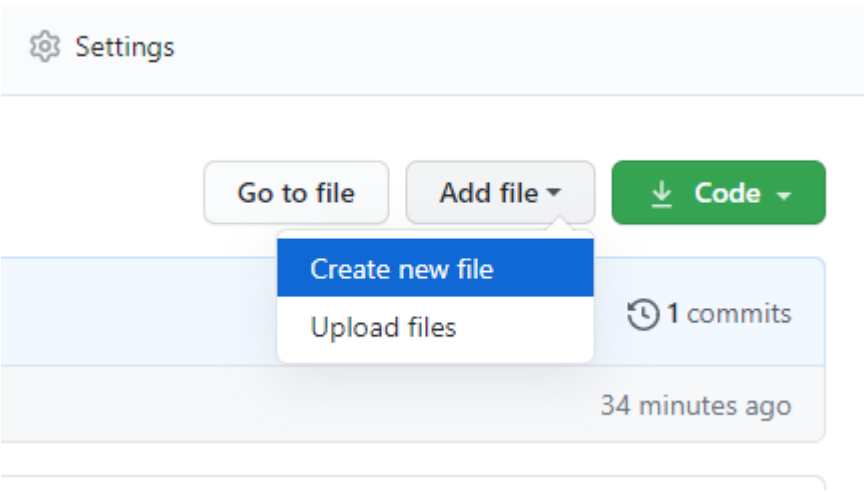
レポート

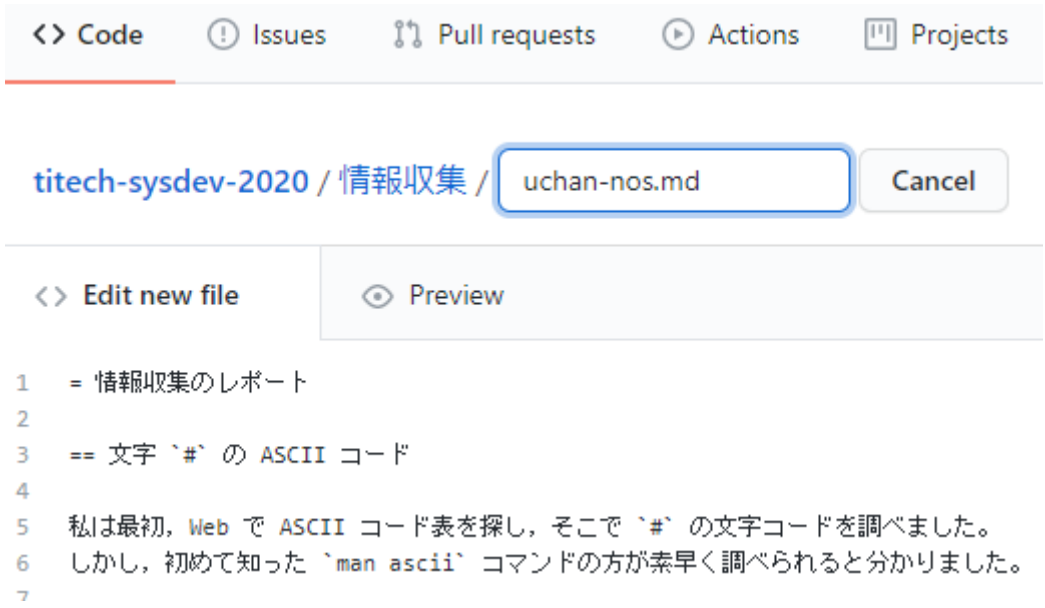
- reports/TOPIC/YOUR_NAME.md
- このファイルに課題への回答を記載する
- 必要なら以下のものを含める
  - Issueへのリンク
  - コミット差分へのリンク
    https://github.com/HOGE/REPO/compare/COMMIT1...COMMIT2
  - その他
- 要するに，成績評価に必要な情報をYOUR_NAME.md自体に記載するか，そこから辿れるようにする

レポートの構成


uchan-nos/titech-sysdev-2020
受講者のリポジトリ
master
Pull Request
report
reports/TOPIC/
uchan-nos.md
YOUR_NAME.md
リンク
文書
コード
Issue
コミット差分

レポートの送り方 1/2

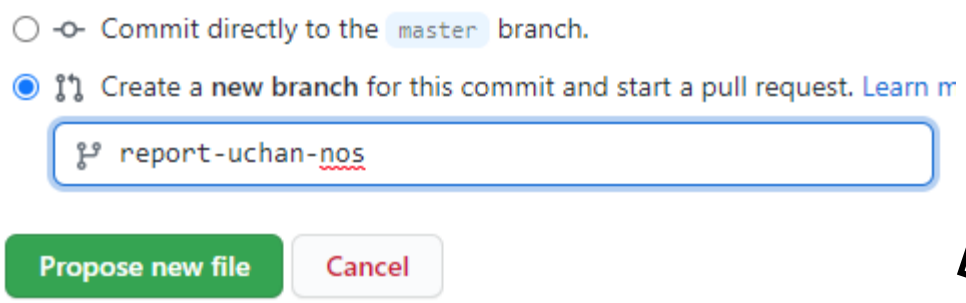




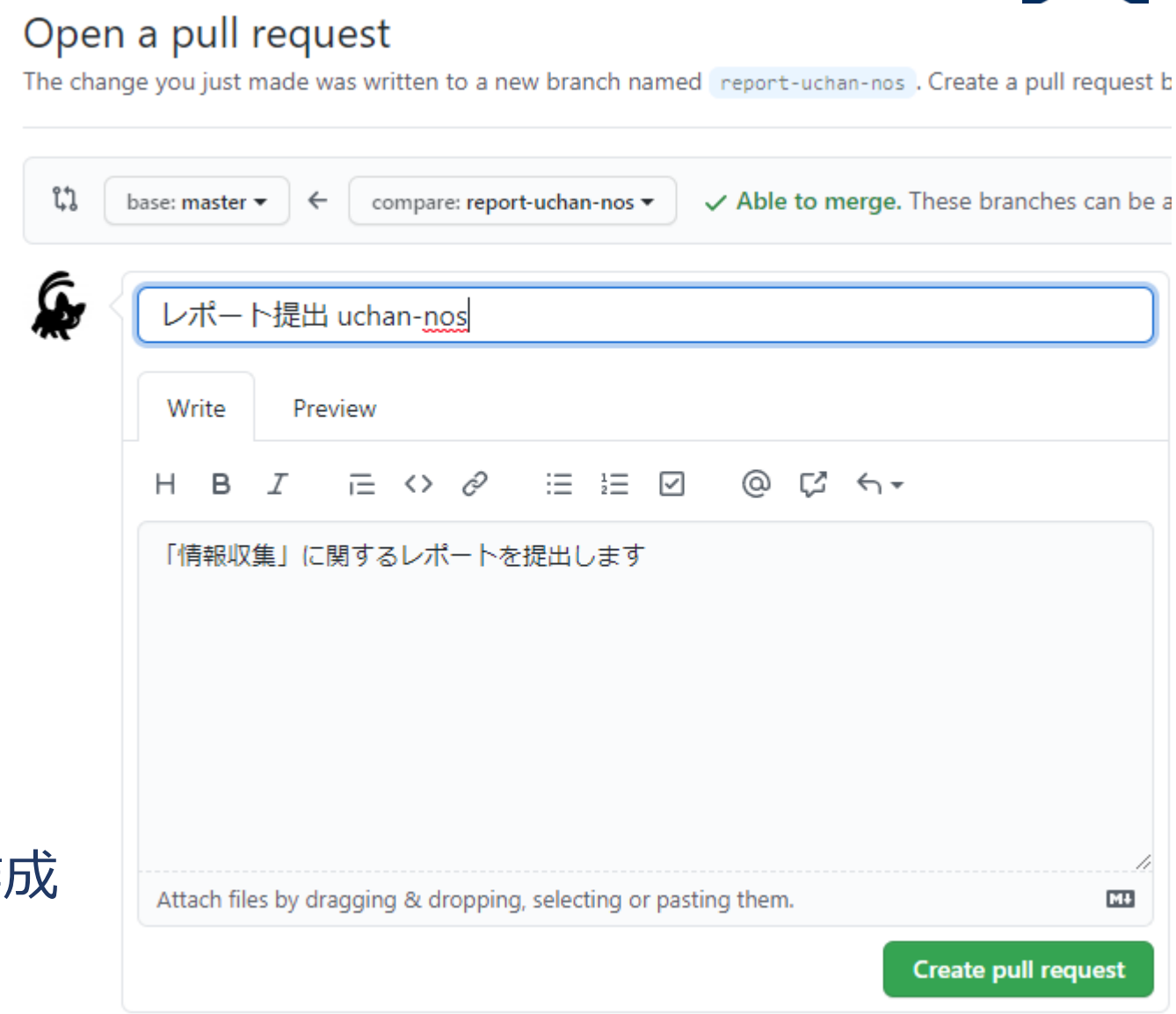
ファイルを新規作成

YOUR_NAME.mdの内容を記述

レポートの送り方 2/2

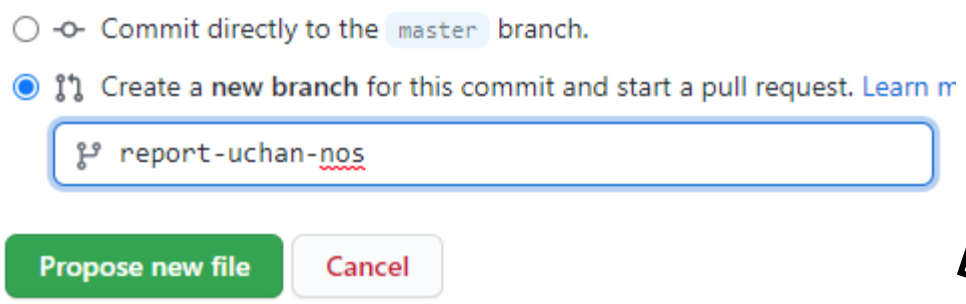


新規ブランチにコミット

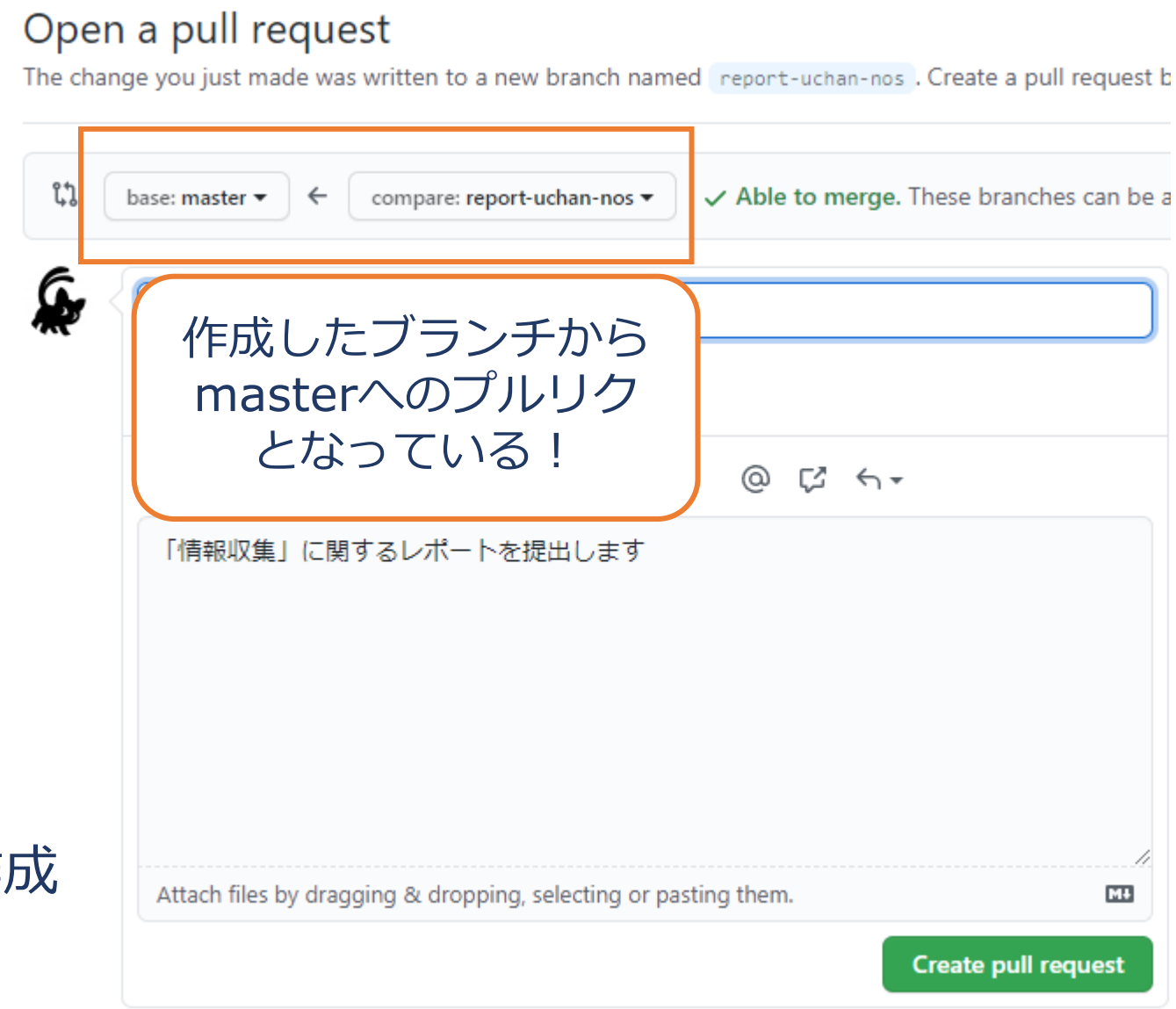


プルリクを作成

レポートの送り方 2/2



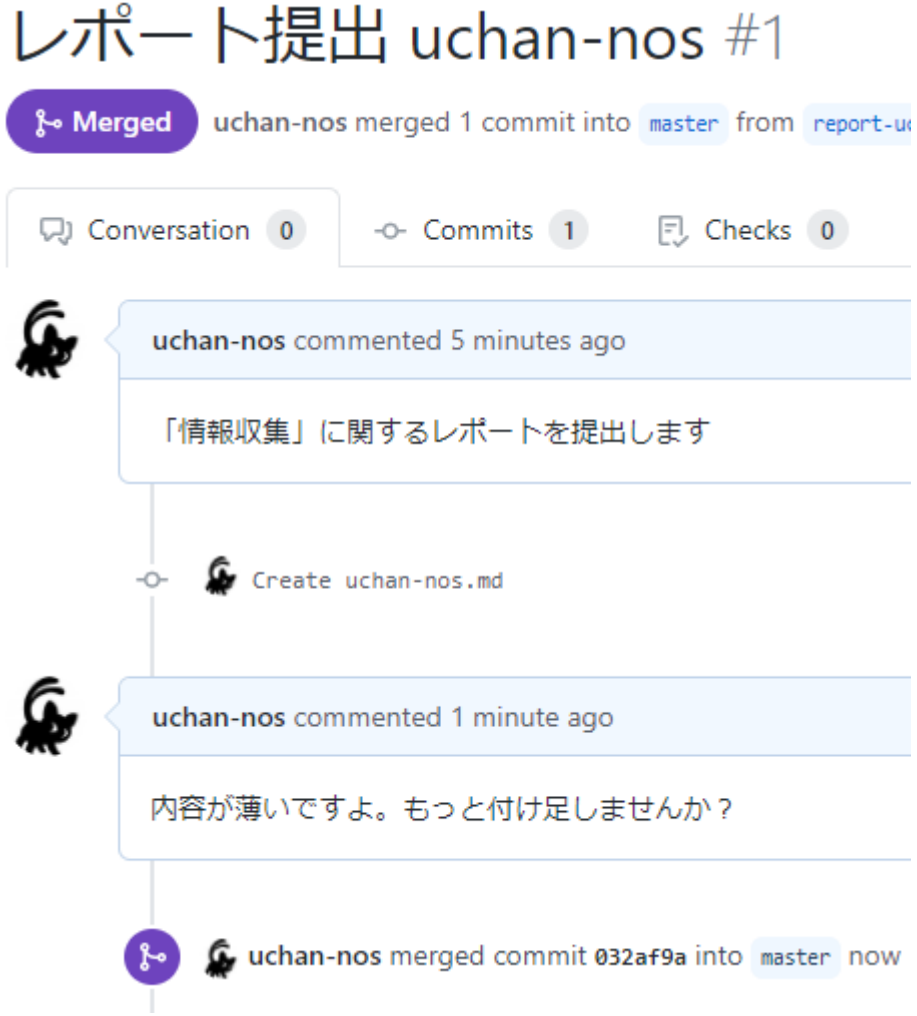
新規ブランチにコミット



作成したブランチから
masterへのプルリク
となっている！

プルリクを作成

レポートの受理



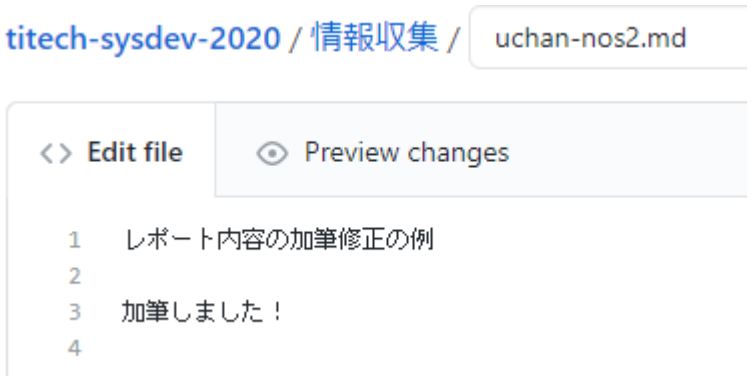
プルリクにコメントが付くことも

Merged：レポート受理済み

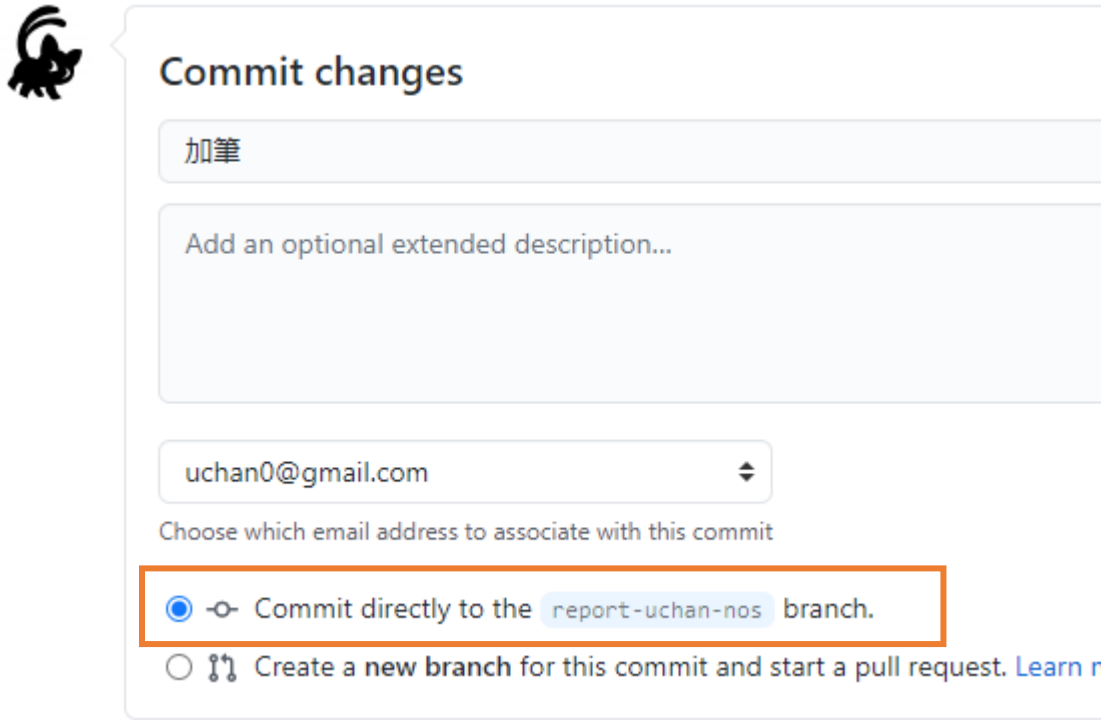
レポートの更新 1/2

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



納得いくまで加筆修正



report-Xブランチにコミット

レポートの更新 2/2

一発目のコミット

Pull Requestに
コミットが追加されていく

レポートの送り方：その他の方法

- 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

次回予告

- 次回は11/13（金）14:20から
- 「ユニットテスト」の続きと
  「継続的インテグレーション」をやります