
Table of Contents

Introduction	1.1
1. Git/GitHubとは	1.2
2. GitHubの登録	1.3
3. Gitのインストール	1.4
4. Gitの基本操作	1.5
5. GitHubの基本操作	1.6
6. さいごに	1.7

Git/GitHub勉強会

ほかご勉強会主催のGit/GitHub勉強会で使用する資料です。
資料の修正・改善等は[本リポジトリのissue](#)までお願いします。

環境構築

```
$ git clone https://github.com/tyokinuhata/git-study.git  
$ cd git-study  
$ gitbook serve
```

ビルド

```
$ make build
```

Gitとは



Gitはソースコードをリポジトリと呼ばれる1つのまとまりとして変更履歴を記録・追跡できる分散型バージョン管理システムである。

元々はLinuxカーネルのソースコードを管理するためにリナス・トーバルズによって開発された。

Gitを使わずに開発するとどうなるか



プロジェクト



プロジェクト_new



プロジェクト(最新版)



プロジェクト2

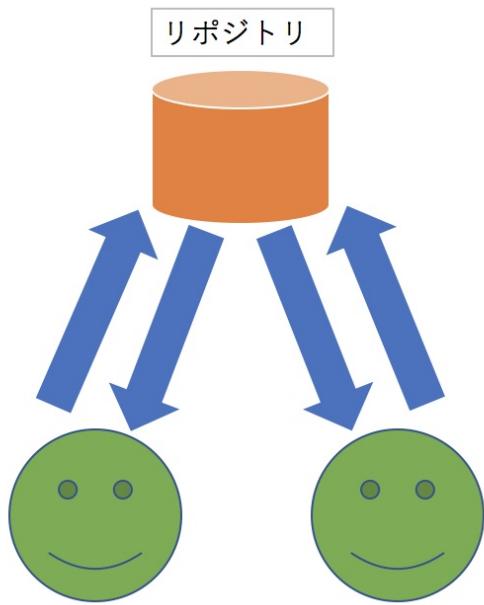


プロジェクト2のコピー

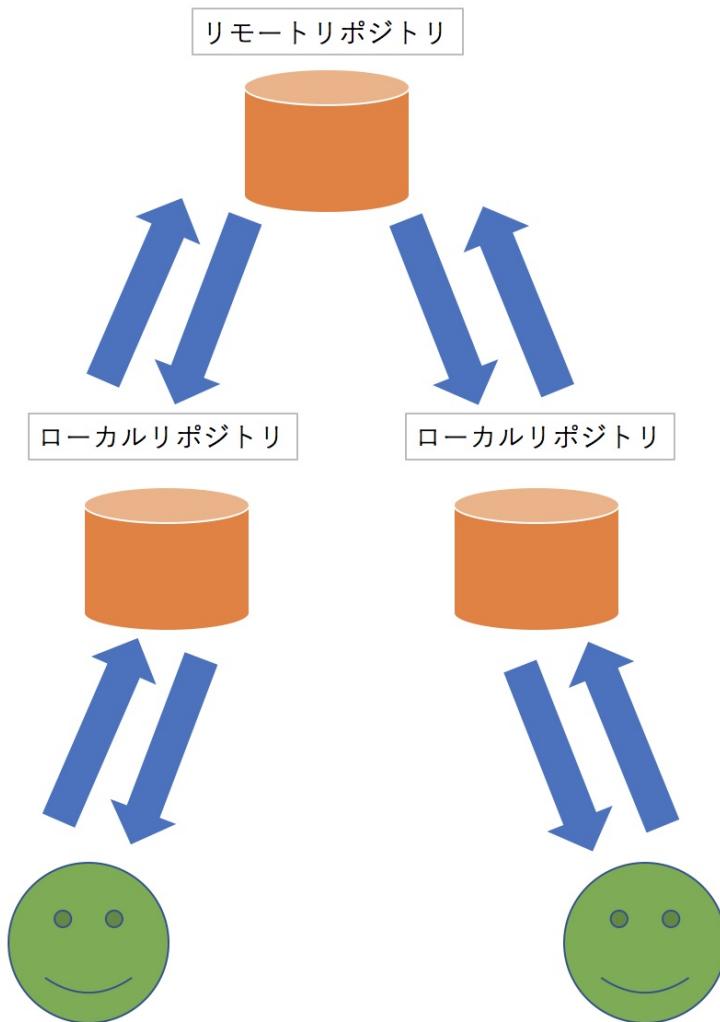
- 様々なバージョンのプロジェクトが散乱し、どれが最新版か分からぬ
- コードの変更箇所を容易に元に戻せない
- 誰がどこのコードを変更したのか特定できない
 - > このような問題をGitは解決してくれる
 - > 特にチーム開発で真価を発揮するため、2年後期から始まる「システム開発演習」では是非導入していきたい

Gitが生まれる以前の話

Gitが生まれる以前はCVSやSubversionを使ってバージョン管理がされていたが、これらのツールは集中型バージョン管理システムのため、リポジトリが置かれたサーバに接続できない環境では最新のソースコードを取得やファイル編集の反映ができないなど、様々な問題が発生した。



Gitはこれを解決するために分散型バージョン管理という方式を取り、それらの不便な点を解決した。
現在では、Gitはエンジニア必須の能力と言っても過言ではない。



GitHubとは

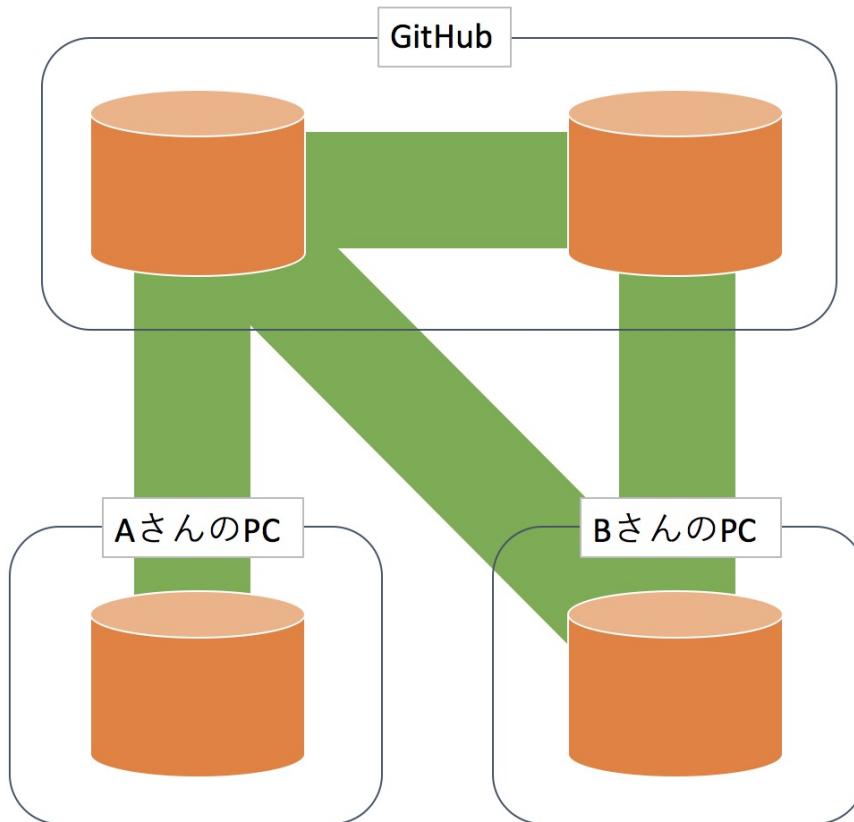


GitHubはGitのリポジトリをホスティングするためのサービスである。

つまり、自分がローカル環境で作ったGitリポジトリを簡単にインターネット上で公開したり、他人と共有したりできるサービスである。

同様のサービスにGitLabやBitBucketなどがある。

Git/GitHubのイメージ



Git/GitHubでは主に、

- ローカルリポジトリから自分のリモートリポジトリに対する操作(pushなど)
- ローカルリポジトリから相手のリモートリポジトリに対する操作(pushなど)
- 相手のリモートリポジトリからローカルリポジトリに対する操作(pullなど)
- 自分のリモートリポジトリから相手のリモートリポジトリに対する操作(Pull Requestなど)
- 相手のリモートリポジトリから自分のリモートリポジトリに対する操作(forkなど)

によって成り立っている。

ここで、ローカルリポジトリとは自分のPC内に作成したGitリポジトリのことを指し、リモートリポジトリとは(今回の場合)GitHub上に作成したGitリポジトリのことを指す。

基本的にローカルリポジトリで各自が作業を行い、その結果をリモートリポジトリに反映する。

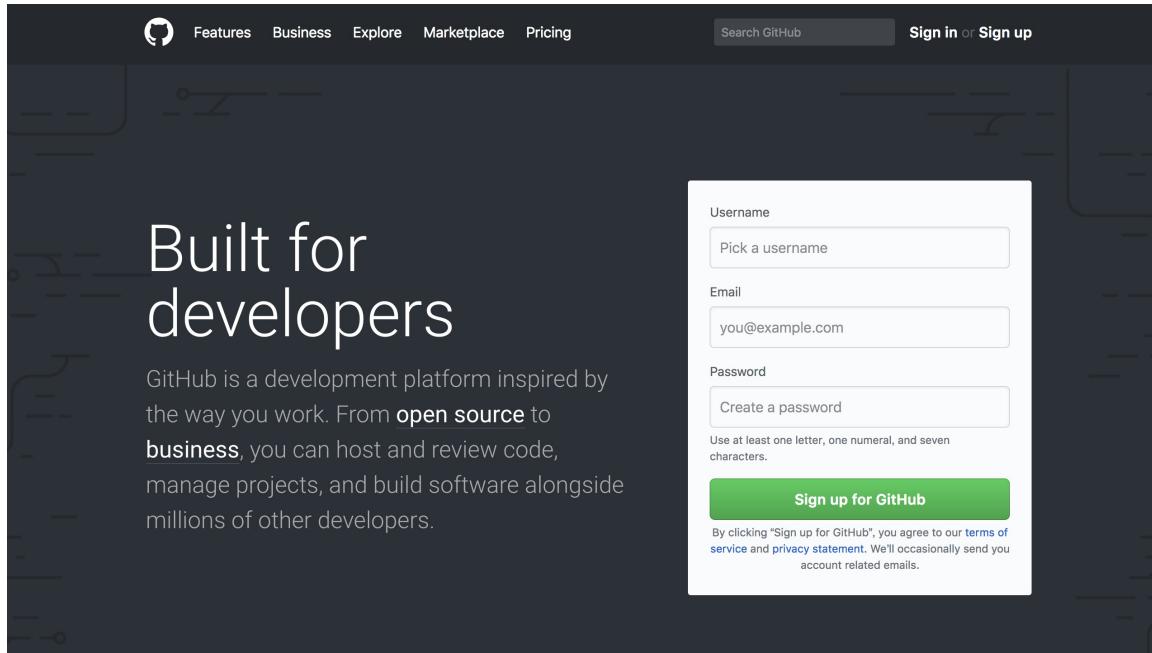
その反映された変更点を見たり、取得するためにリモートリポジトリは存在する。

これから、それらの操作について順を追って見ていこう。

GitHubの登録

それでは、GitHubのアカウントを作成する手順を説明する。

まずはGitHubにアクセスし、すでにアカウントを持っている人は右上からサインイン、持っていない人はテキストボックスに何も入力せず、"Sign up for GitHub"をクリックする。



ユーザ名、メールアドレス、パスワードを入力し、"Create an account"をクリックする。

Join GitHub

The best way to design, build, and ship software.

 Step 1:
Create personal account

 Step 2:
Choose your plan

 Step 3:
Tailor your experience

Create your personal account

Username

This will be your username. You can add the name of your organization later.

Email address

We'll occasionally send updates about your account to this inbox. We'll never share your email address with anyone.

Password

Use at least one lowercase letter, one numeral, and seven characters.

By clicking "Create an account" below, you agree to our [terms of service](#) and [privacy statement](#). We'll occasionally send you account related emails.

Create an account

次に, "Unlimited public repositories for free."(フリープラン)にチェックを入れ, "Continue"をクリックする。ちなみに, プライベートリポジトリを無制限に作成できる学生プランも用意されているので, 学生の間に[GitHub Education](#)で申請を行っておくと良い。
プライベートリポジトリとは, 限られたメンバーのみが閲覧可能なリポジトリのことで, 外部に公開される通常のリポジトリはパブリックリポジトリ(または単にリポジトリ)と呼ばれる。

Welcome to GitHub

You've taken your first step into a larger world, @ITCtaro.

 **Completed**
Set up a personal account

 **Step 2:**
Choose your plan

 **Step 3:**
Tailor your experience

Choose your personal plan

- Unlimited public repositories for free.
- Unlimited private repositories for \$7/month. (view in JPY)**

Don't worry, you can cancel or upgrade at any time.

Help me set up an organization next

Organizations are separate from personal accounts and are best suited for businesses who need to manage permissions for many employees.

[Learn more about organizations](#)

Send me updates on GitHub news, offers, and events

Unsubscribe anytime in your email preferences. [Learn more](#)

Continue

Both plans include:

-  Collaborative code review
-  Issue tracking
-  Open source community
-  Unlimited public repositories
-  Join any organization

アカウントの作成に成功すると以下のようなページが表示される。

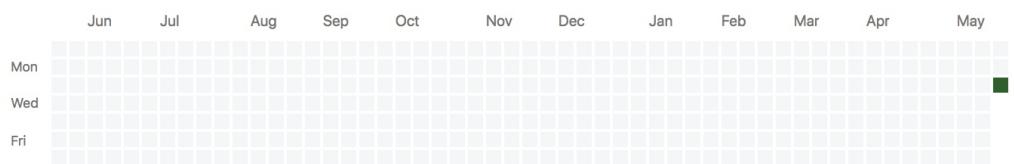
Overview Repositories 0 Stars 0 Followers 0 Following 0

Popular repositories

You don't have any public repositories yet.

1 contribution in the last year

Contribution settings ▾



Learn how we count contributions.

Less More

This is your **contribution graph**. Your first is for joining GitHub and you'll earn more as you make [additional contributions](#). More contributions means a darker green square for that day. Over time, your chart might start looking [something like this](#).

We have a quick guide that will show you how to create your first repository and earn more green squares!

[Read the Hello World guide](#)

Gitのインストール

Macでインストールする場合

<https://git-scm.com/download/mac> からダウンロードし, インストールする.

macOS用のパッケージ管理システムのHomebrewが入っている場合なら,

```
$ brew install git
```

でインストールすることが可能.

Windowsでインストールする場合

<https://git-scm.com/download/win> からダウンロードし, インストールする.

Linux/UNIXでインストールする場合

Debian系ならapt, Red Hat系ならyumでインストールする.

詳しくは <https://git-scm.com/download/linux> を参照.

Gitがインストールされているかを確認

macOSならターミナル, WindowsならGit Bash, Linux/UNIXなら端末を開き, `git --version` と入力する.

```
$ git --version
git version 2.15.1 (Apple Git-101)
```

のようにGitのバージョン情報が表示されればOK.

Gitの初期設定

Gitにユーザネームとメールアドレスを設定する(初回のみ).

```
$ git config --global user.name "Taro Tanaka"
$ git config --global user.email "taro@example.com"
```

また、必須ではないがログやステータスを表示した際にハイライトされる設定もしておく。

```
$ git config --global color.ui true
```

Gitリポジトリの作成

```
$ mkdir hello-git
$ cd hello-git
$ git init
```

`mkdir` や `cd` はLinuxコマンドなどと呼ばれるもので、Linuxコマンドはコマンドラインでファイルやディレクトリの操作を行うために用意されている。

コマンド名	説明
<code>mkdir</code>	Make Directoryの略。引数で指定した名前のディレクトリを作成する。
<code>cd</code>	Change Directoryの略。引数で指定した名前のディレクトリに移動する。

`git init` コマンドを叩くと、カレントディレクトリ以下に `.git` という隠しディレクトリが生成される。これにより、カレントディレクトリ以下にあるファイルはGitに管理され、Gitリポジトリという1つのまとまりとして扱われる。

また、「Gitに管理される」というのは、Gitの各種コマンドが使える状態であるということでもある。

大まかな流れ



Gitの管理下に置かれた実際に作業をしているディレクトリのことをワークツリーと呼ぶ。

ワークツリーでファイルの作成や編集、削除を行った後、変更を記録したいファイルをステージングエリア(インデックスとも呼ぶ)にステージングする作業を行う。

その後、ステージングされたファイルをコミットする作業を行い、これで初めてGitリポジトリに変更履歴として残る。

ステージングする



ステージングするにはまずなにかしらファイルを作成する必要があるため,

```
$ touch hello.txt
```

としてファイルを作成する.

`touch` コマンドは引数で指定した名前のファイル名を作成するコマンドである.

次にステージングを行う.

```
$ git add .
```

`git add` コマンドでワークツリーにある変更されたファイルをステージングエリアにステージングすることができる.

ここで, `.` (ドット)というのは, ワークツリーにあるファイルの変更を全てステージングエリアにステージングするという意味である.

ここでは `.` (ドット)を使って全てのファイルをステージングしているが, ファイル名を指定して特定のファイルの変更のみステージングさせることもできる.

ちなみに, 作成, 編集, 削除などを行ったファイルの状態を確認するには, `git status` コマンドを使うと良い.

```
$ git status
On branch master

No commits yet

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

  hello.txt

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

ステージングされているファイルは'No commits yet'の下に, ステージングされていないファイルは'Untracked files'の下に表示される.

コミットする



ステージングしたファイルはコミットと呼ばれる作業を行い、Gitの変更履歴に記録する必要がある。

コミットするには、`git commit` コマンドを使う。

また、このときに `-m` オプションを付けることによってコミットに対するメッセージを残すことができる。

```
$ git commit -m "Initial commit"
[master (root-commit) 23bbb67] Initial commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 hello.txt
```

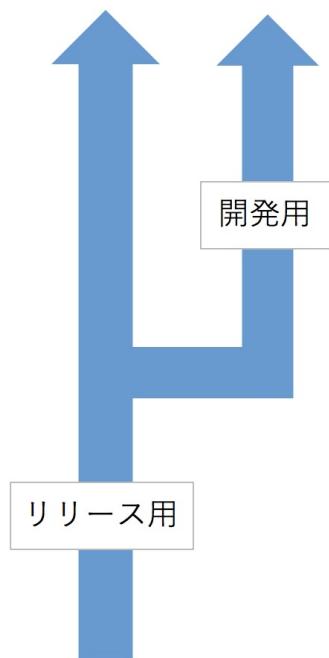
コミットによってGitに記録された変更履歴は、`git log` コマンドで確認できる。

```
$ git log
commit 23bbb672088c1eeaeab1a4c48e81256812edf02e7 (HEAD -> master)
Author: Kazuki Tobita <kazukiti201@gmail.com>
Date:   Tue May 22 18:53:08 2018 +0900

  Initial commit
  ...
```

閉じるときは `:q` と入力する。

ブランチとは



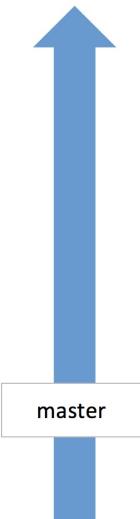
開発用とリリース用を分けたい場合など、同じアプリケーションでも状況に応じて切り分けたい場合がある。

その問題を解決するのがGitにおけるブランチという概念である。

まずは現在存在するブランチを確認する。

ブランチの一覧を表示するには `git branch` コマンドを使う。

```
$ git branch
* master
```

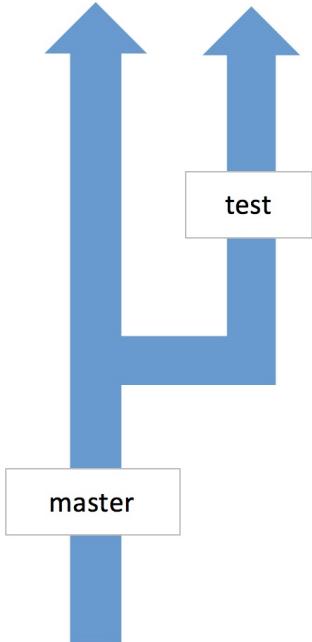


現在自分がいるブランチには左側に * (アスタリスク)が表示される。
また, デフォルトでは master ブランチしか無いため, 必要に応じてブランチを作る必要がある.

次に, ブランチを生成する.
今回は test という名前のブランチを生成してみる.

```
$ git branch test
```

これにより, masterブランチからtestブランチが新たに分岐された.



```
$ git branch  
* master  
  test
```

上の実行結果から分かるように、現在自分がいるブランチが `master` になっている。

そのため、`test` ブランチで作業するには `master` ブランチから `test` ブランチに切り替える必要がある。

ブランチの切り替えには `git checkout` コマンドを使う。

```
$ git checkout test  
Switched to branch 'test'
```

この状態で再びブランチを確認してみると、

```
$ git branch  
  master  
* test
```

のように、`test` の左側に * が付いていることから、ブランチが切り替わったことが見て取れる。

リポジトリの作成

GitHub上でマイページの"Repositories"タブに移動し, "New"をクリックする。

The screenshot shows the GitHub user profile interface. At the top, there are tabs for Overview, Repositories (0), Stars (0), Followers (0), and Following (0). Below the tabs is a search bar labeled "Search repositories..." and a button labeled "Type: All ▾". A prominent green button labeled "New" is located on the right. The main content area displays a message: "ITCtaro doesn't have any public repositories yet."

"Repository name"に任意の名前を入力し(今回の場合は"hello-git"), "Create repository"をクリックする。

Create a new repository

A repository contains all the files for your project, including the revision history.

The screenshot shows the "Create a new repository" form. It includes fields for Owner (set to "takashi0602") and Repository name ("hello-git"). Below these, a note says "Great repository names are short and memorable. Need inspiration? How about scaling-octo-waffle." There is a "Description (optional)" field with an empty input box. Under "Visibility", the "Public" option is selected, with a note: "Anyone can see this repository. You choose who can commit." The "Private" option is also shown with its note: "You choose who can see and commit to this repository." A checkbox for "Initialize this repository with a README" is checked, with a note: "This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository." At the bottom, there are dropdowns for ".gitignore" (None) and "Add a license" (None), and a large green "Create repository" button.

以下のような画面が出たら成功である。

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

Set up in Desktop or HTTPS <https://github.com/takashi0602/hello-git.git>

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

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

```
echo "# hello-git" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/takashi0602/hello-git.git
git push -u origin master
```

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

```
git remote add origin https://github.com/takashi0602/hello-git.git
git push -u origin master
```

...or import code from another repository

ローカルリポジトリに追加

まずはGitHubにて、リモートリポジトリのURLをコピーする。

リポジトリの画面の右側にあるアイコンを押し、表示しているURLをコピーする。

You've done this kind of thing before

HTTPS SSH <https://github.com/takashi0602/hello-git.git>

次に、ローカルリポジトリにリモートリポジトリを追加する。

これにより、後述するpushやpullなどの作業を簡単に行えるようになる。

ターミナルで下記のコマンドを実行すると、ローカルリポジトリに対してリモートリポジトリが `origin` という名前で登録される。

ちなみに、`origin` 以外の名前を付けても良いが、慣習的に `origin` と付けられることが多い。

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

ローカルリポジトリに登録されているリモートリポジトリの名前やそれに対応するURLを確認する場合は、

```
$ git remote -v
origin https://github.com/takashi0602/hello-git.git (fetch)
origin https://github.com/takashi0602/hello-git.git (push)
```

とコマンドを入力し、実行する。

実行結果が上記のようになれば成功である。

pushする

pushと呼ばれる操作を行うことで、ローカルリポジトリの変更内容をリモートリポジトリに反映させることができる。
 先程作成した `hello.txt` はコミットまでされているので、次はこれをpushする。
 pushする際は `git push` の後に続けてリモートリポジトリの名前(今回は `origin`)、push先ブランチ名(今回は `master`)を入力する。

```
$ git push origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 244 bytes | 244.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/takashi0602/hello-git.git
  9678fb4..52e8275  master -> master
```

上記のような実行結果が表示されれば成功である。

GitHub上でリポジトリを確認すると、`hello.txt`が反映されていることが分かる。

No description, website, or topics provided.

Add topics

2 commits 1 branch 0 releases 1 contributor

Branch: master ▾ New pull request Create new file Upload files Find file Clone or download ▾

takashi0602 Initial commit Latest commit f87330a 16 seconds ago

hello.txt Initial commit 16 seconds ago

Add a README

pullする

pullと呼ばれる操作を行うことで、リモートリポジトリでの変更内容をローカルリポジトリに反映させることができる。

基本的にGitHub上で作業することは無いが、今回は擬似的にリモートリポジトリに変更があった状況を作り出すため、GitHub上で直接 `hello.txt` の内容を変更する。

`hello.txt` を開き、画面右側にある鉛筆のアイコンを選択し、ファイルを編集する。

takashi0602 / hello-git Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Branch: master ▾ hello-git / hello.txt Find file Copy path

takashi0602 add hello.txt 52e8275 6 days ago

1 contributor

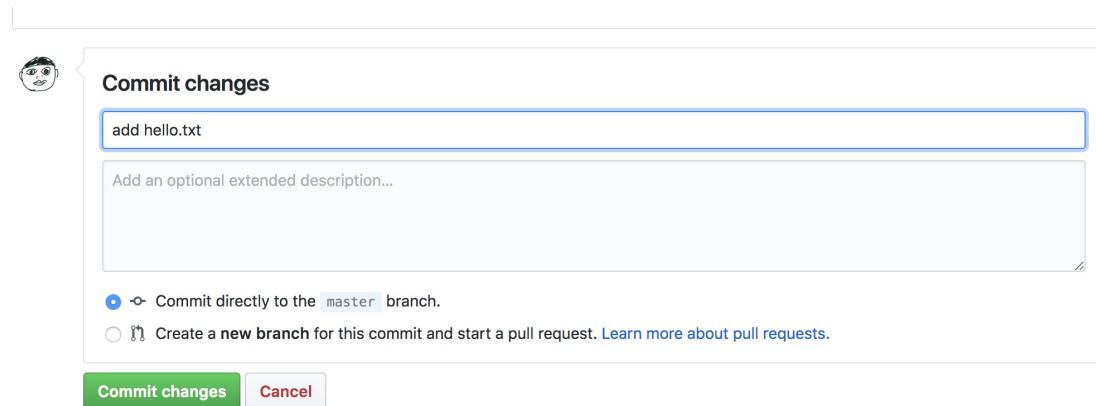
0 lines (0 sloc) | 0 Bytes Raw Blame History

今回は `hello, github!!!` と入力する。

The screenshot shows a GitHub repository page for 'takashi0602 / hello-git'. The 'Code' tab is selected, showing a file named 'hello.txt' with the content '1 hello, github!!!'. The repository has 0 issues, 0 pull requests, 0 projects, 0 wiki pages, and 0 insights.

画面上でコミットを作成する。

"Commit changes"ボタンをクリックすると変更が確定される。



変更された内容をpullしてローカルリポジトリにも反映させる。

```
$ git pull origin master
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/takashi0602/hello-git
 * branch      master    -> FETCH_HEAD
   52e8275..4acdde3  master    -> origin/master
Updating 52e8275..4acdde3
Fast-forward
 hello.txt | 1 +
 1 file changed, 1 insertion(+)
```

forkする

誰かがGitHub上に公開しているリポジトリを自分のGitHub上に複製することをforkと呼ぶ。

今回はhello-githubをforkする。

右上にある"Fork"ボタンをクリックする。



forkするユーザを選択する。

Where should we fork this repository?

- @tyokinuhata
- @AfterSchool...
- @chatbox-inc
- @ITCreate

Can't find what you're looking for?

You don't have permission to fork to these organizations:

- geekinc
- tam-bourine

tyokinuhata / hello-github
forked from takashi0602/hello-github

Code Pull requests Projects Wiki Insights Settings

Fork 1

Forking takashi0602/hello-github

It should only take a few seconds.

Refresh



forkが完了すると、リポジトリ名の下部に"forked from takashi0602/hello-github"とfork元リポジトリが記載される。

No description, website, or topics provided.

Add topics

7 commits 1 branch 0 releases 2 contributors

Branch: master New pull request

This branch is even with takashi0602:master.

takashi0602 add hello.txt Latest commit 6c4a25b an hour ago

hello.txt add hello.txt an hour ago

Help people interested in this repository understand your project by adding a README.

Add a README

cloneする

誰かがGitHub(など、インターネット上)に公開したリポジトリを自分のPCに複製(ダウンロード)することをGitではcloneと呼ぶ。

cloneするときは"Clone or Download"をクリックし、URLをコピーする。

No description, website, or topics provided.

3 commits 1 branch 0 releases 2 contributors

Branch: master New pull request

takashi0602 Merge pull request #2 from tyokinuhata/master

pasta.txt パスタの削除

Clone with HTTPS Use SSH
Use Git or checkout with SVN using the web URL.
https://github.com/takashi0602/hello-github

Open in Desktop Download ZIP

また、cloneする場所はhello-git内ではないので、`cd` コマンドを使用して一つ上の階層に移動する。

```
$ cd ..
```

移動できたら、ターミナルやGit Bashで以下のコマンドを入力する。

```
$ git clone https://github.com/ユーザ名/hello-github.git
```

これでcloneが完了した。

Pull Requestの作成

Pull Requestは自分のローカルリポジトリでの変更をコミットし、リモートリポジトリにpush後、その変更をfork元のリポジトリに取り込んでもらいたい時に使う機能である。

プルリクやPRと省略されて呼ばれることが多い。

まずは先程cloneしたリポジトリに移動する。

```
$ cd hello-github
```

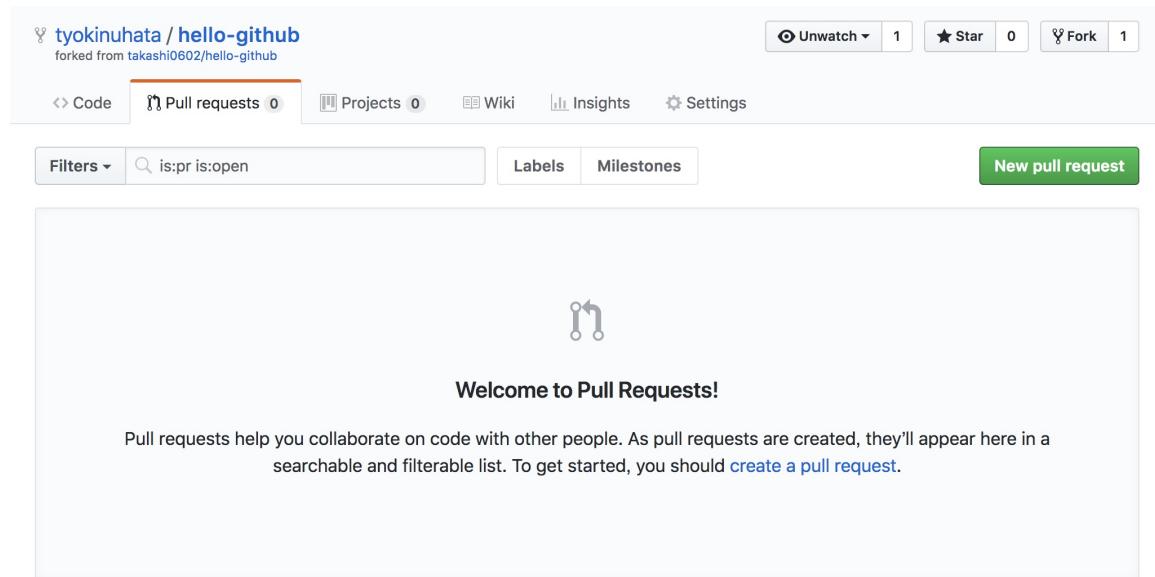
コミットするにはリポジトリに変更を加える必要があるため、今回は 学籍番号.txt (例: B7233.txt)という名前のファイルを作成する。

その後、ステージングとコミットを行い、pushする。

```
$ touch B7233.txt
$ git add .
$ git commit -m "add B7233.txt"
$ git push origin master
```

ここからが本題で、いよいよPull Requestを作成する段階に入る。

GitHubに移動し、"Pull requests"タブを選択し、"New pull request"ボタンをクリックする。



次に、"Create pull request"ボタンをクリックする。

takashi0602 / hello-github

Code Issues Pull requests Projects Wiki Insights

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 fork: takashi0602/hello-github ▾ base: master ▾ head fork: tyokinuhata/hello-github ▾ compare: master ▾

✓ Able to merge. These branches can be automatically merged.

Create pull request Discuss and review the changes in this comparison with others. ?

- 1 commit 1 file changed 0 commit comments 1 contributor

Commits on May 29, 2018

tyokinuhata バスタの削除 Verified 125bf63

Showing 1 changed file with 1 addition and 1 deletion.

Unified Split

2 pasta.txt

... ... @@ -1 +1 @@
1 -バスタを茹でました。
1 +バスタを茹でました。

すると、Pull Requestの作成画面が出るので、Pull Requestのタイトルとその説明を適当に入力し、右下にある"Create pull request"をクリックする。

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.

そうすると、pull requestが作成される。

takashi0602 / hello-github

Pull requests 1

tyokinuhata wants to merge 1 commit into `takashi0602:master` from `tyokinuhata:master`

Conversation 0 **Commits 1** **Checks 0** **Files changed 1**

tyokinuhata commented just now

No description provided.

パスタの削除

Verified 125bf63

Add more commits by pushing to the **master** branch on **tyokinuhata/hello-github**.

This branch has no conflicts with the base branch
Only those with **write access** to this repository can merge pull requests.

Write Preview

Leave a comment

Attach files by dragging & dropping, [selecting them](#), or pasting from the clipboard.

mergeする

mergeはPull Requestの内容をリモートリポジトリに反映させるために使用する機能である。

まず、"Pull Requests"タブを選択肢、mergeしたいPull Requestを選択する。
そして、"Merge pull request"ボタンをクリックする。

パスタの削除 #2

Open tyokinuhata wants to merge 1 commit into `takashi0602:master` from `unknown repository`

Conversation 0 Commits 1 Checks 0 Files changed 1

tyokinuhata commented 11 minutes ago First-time contributor +  

No description provided.

tyokinuhata パスタの削除 Verified 125bf63

 Continuous integration has not been set up
Several apps are available 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.

クリック後、適当なメッセージを入力して"Confirm merge"ボタンをクリックすると、mergeが完了する。

パスタの削除 #2

Open tyokinuhata wants to merge 1 commit into `takashi0602:master` from `unknown repository`

Conversation 0 Commits 1 Checks 0 Files changed 1

tyokinuhata commented 12 minutes ago First-time contributor +  

No description provided.

tyokinuhata パスタの削除 Verified 125bf63

 Merge pull request #2 from tyokinuhata/master
パスタの削除

Confirm merge **Cancel**

また、mergeが完了すると、Pull Requestの状態が"Open"から"Merged"に変更され、色調も緑色から紫色に変更される。

Code Issues Pull requests Projects Wiki Insights Settings

パスタの削除 #2

Merged takashi0602 merged 1 commit into takashi0602:master from unknown repository just now

Conversation 0 Commits 1 Checks 0 Files changed 1

tyokinuhata commented 12 minutes ago First-time contributor +

No description provided.

パスタの削除 Verified 125bf63 Revert

takashi0602 merged commit f54d0f5 into takashi0602:master just now

issueの作成

issueは日本語で「問題」や「課題」を指す言葉で、プロジェクトで発生したバグや新機能・改善点等の要望をissueとして一元管理することでプロジェクトの見通しを良くするための機能である。

新たにissueを作成する際は"New issue"をクリックする。

takashi0602 / hello-github Watch 0 Star 0 Fork 0

Code Issues Pull requests Projects Wiki Insights Settings

Filters is:issue is:open Labels Milestones New issue

Welcome to Issues!

Issues are used to track todos, bugs, feature requests, and more. As issues are created, they'll appear here in a searchable and filterable list. To get started, you should [create an issue](#).

ProTip! Updated in the last three days: [updated:>2018-05-24](#).

そして、issueのタイトルとその説明を記述し、"Submit new issue"をクリックすることでissueが作成される。

The screenshot shows the GitHub issue creation interface. At the top, there is a title input field containing "パスタを茹でる". Below it are two tabs: "Write" (selected) and "Preview". To the right of the tabs are various text formatting icons. The main content area contains a bulleted list: "- お湯を沸かす", "- 沸騰してから塩を入れる", and "- パスタを茹でる". At the bottom of this area, there is a placeholder: "Attach files by dragging & dropping, selecting them, or pasting from the clipboard." On the far left, there is a small user icon. On the right side, there are buttons for "Submit new issue" and "Styling with Markdown is supported".

このとき、右側にある"Assignees"からissueの担当者を割り当てる事ができたり、同じく右側にある"Labels"からissueの分類をすることもできる。

The screenshot shows a GitHub issue page for a comment on issue #1. The comment was made by "takashi0602" a minute ago. The comment content is the same as the one in the creation interface: "- お湯を沸かす", "- 沸騰してから塩を入れる", and "- パスタを茹でる". To the right of the comment, there are several settings sections: "Assignees" (No one—assign yourself), "Labels" (None yet), "Projects" (None yet), "Milestone" (No milestone), and "Notifications" (You're receiving notifications because you authored the thread). Below the comment, there is a reply form with "Write" and "Preview" tabs, a text area with "Leave a comment", and a "Comment" button. There is also a note about styling with Markdown.

.gitignore

.DS_Store や Thumbs.db といったファイルシステムによって自動生成されるファイルや、node_modules のようなモジュール、.vscode や.idea のようなエディタが生成するファイルはGitの管理下に置く必要は無い。そのようなファイルやディレクトリを.gitignoreに記述することで、Gitの監視下から除外することができる。

例えば、hoge.txtをGitの管理下から除外したい場合、まず.gitignoreファイルを作成する。

```
$ touch hoge.txt
$ touch .gitignore
```

次に、`.gitignore` に `hoge.txt` と書き込む。`echo`コマンドと `>` の詳しい説明は省略するが、これで `.gitignore` に `hoge.txt` と書き込んでいる。

```
$ echo hoge.txt > .gitignore
```

上記のようにすることで、ステージングしても `hoge.txt` はステージングされないことが分かる。

```
$ git add .
$ git status
```

さいごに

ここまでGitの主要なコマンドについて解説してきたが、正直Gitの良さが分かった人は少ないはずだ。Gitの良さは実際にアプリケーションを開発して初めて分かるものなので、開発する際は積極的にGitを導入し、少しでも多く触れてみて欲しい。今回解説したコマンドを使いこなせるようになるだけでも、開発する上でかなりの手助けになる。

その他のコマンド

ここまでいくつかのGitコマンドを解説してきたが、これらのコマンドはほんの一部に過ぎず、他にも様々なコマンドが存在する。

ここからのステップアップとしていくつかのコマンドを紹介しておく。

- git tag
- git rm
- git reset
- git revert
- git stash
- git cherry-pick
- git rebase

Git/GitHubを勉強できるサイト・書籍の紹介

初級者向け

- [Progate](#)
- [ドットインストール](#)
- [わかばちゃんと学ぶ Git使い方入門](#)

中級者向け

- [サルでもわかるGit入門](#)
- [入門git](#)

上級者向け

- [Pro Git](#)

GitのGUIアプリケーション



Sourcetreeなど、GUIで操作するためのアプリケーションも多数用意されている(筆者はCUIをオススメする)。

GitHubのDesktopアプリケーション

Webブラウザ上でも不自由なくGitHubを操作できるが、[GitHub Desktop](#)と呼ばれるGitHubのデスクトップアプリケーションも用意されている。