
Table of Contents

README	1.1
0. はじめに	1.2
1. Git/GitHubとは	1.3
2. GitHubの登録	1.4
3. Gitのインストール	1.5
4. Gitの基本操作	1.6
5. GitHubの基本操作	1.7
6. さいごに	1.8

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](#)又は[Pull Request](#)までお願いします。

環境構築

```
$ git clone https://github.com/after-school-study-group/git-study.git  
$ cd git-study  
$ gitbook serve
```

ビルド

```
$ make build
```

はじめに

アプリケーションの大規模化・複雑化が進むにつれ、アプリケーションの品質を左右するバージョン管理システムは開発する上で「当たり前のもの」となりつつある。

ここではそんなバージョン管理システムの中でもデファクトスタンダードとして君臨しているGitについて、その基礎の基礎を学んでいく。

それに加え、Gitのホスティングサービスの中でも最もポピュラーであるGitHubの取り扱いについても学ぶ。

本資料は、GitとGitHubという2つの技術をより実践的な視点からできる限り平易に解説しており、資料内の概念を理解しコマンドを使いこなせるようになるだけでも十分に実際の開発で役立つような内容となっている。

それでは、Dive into Git!!!!

著者について

かずきち(飛田一貴)



大阪情報コンピュータ専門学校 総合情報メディア学科 ネットワークセキュリティ専攻 3年。
二郎系ラーメンが好き。

たかし(高橋秀明)



大阪情報コンピュータ専門学校 総合情報メディア学科 ネットワークセキュリティ専攻 3年。
CSSアニメーションとフロントエンドが好き。
メロンをこよなく愛する。

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



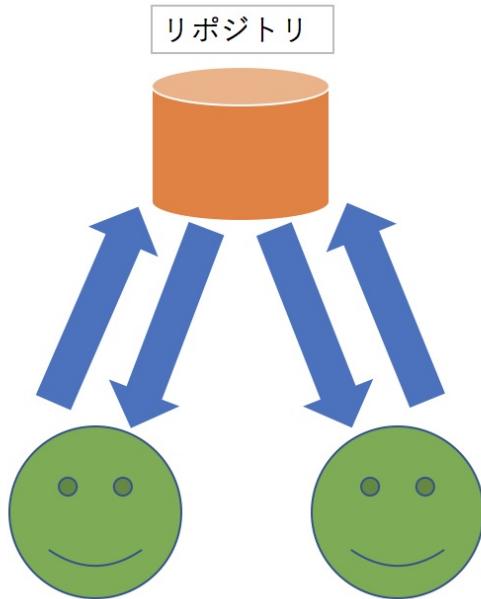
Gitなどのバージョン管理システムを使わずに開発すると、主に以下のような問題が発生する。

- 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

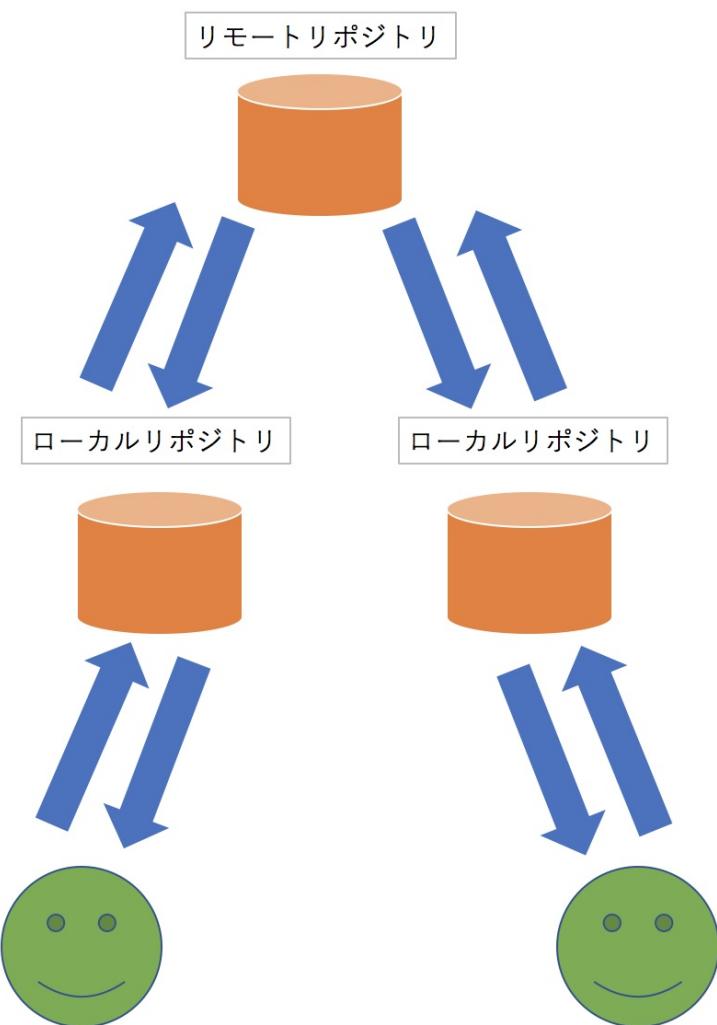
これらの問題を解決するGitは個人開発でも十分に役立つが、チーム開発でその真価を発揮するため、是非とも2年後期から始まる「システム開発演習」では導入していきたい。

Gitが生まれる以前の話

Gitが生まれる以前は**CVS**や**Subversion**をはじめとするバージョン管理システムが使用されていたが、これらのツールはいずれも「集中型バージョン管理」という方式を取っていたため、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生じた。



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、それらの欠点を克服した。



GitHubとは



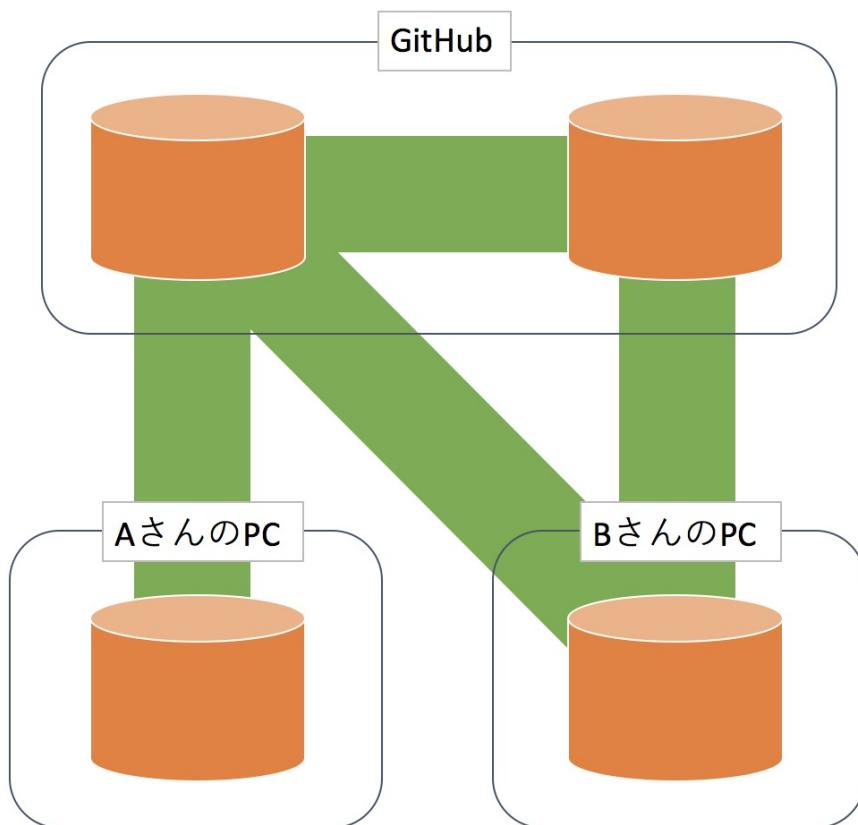
GitHubはGitのリポジトリをホスティングするためのサービスである。

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。

これらのサービスが存在するおかげで、我々は自分でGitサーバを作ることなしに手軽にGitリポジトリを公開することができる。

Git/GitHubのおおまかなイメージ



上図のそれぞれのリポジトリは、

ローカルリポジトリ ... AさんのPC内にあるGitリポジトリ、BさんのPC内にあるGitリポジトリ
リモートリポジトリ ... AさんのGitHub上にあるGitリポジトリ、BさんのGitHub上にあるGitリポジトリ
の二種類に分類することができる。

これらのリポジトリは、互いに様々な操作を通じてやり取りを行う。

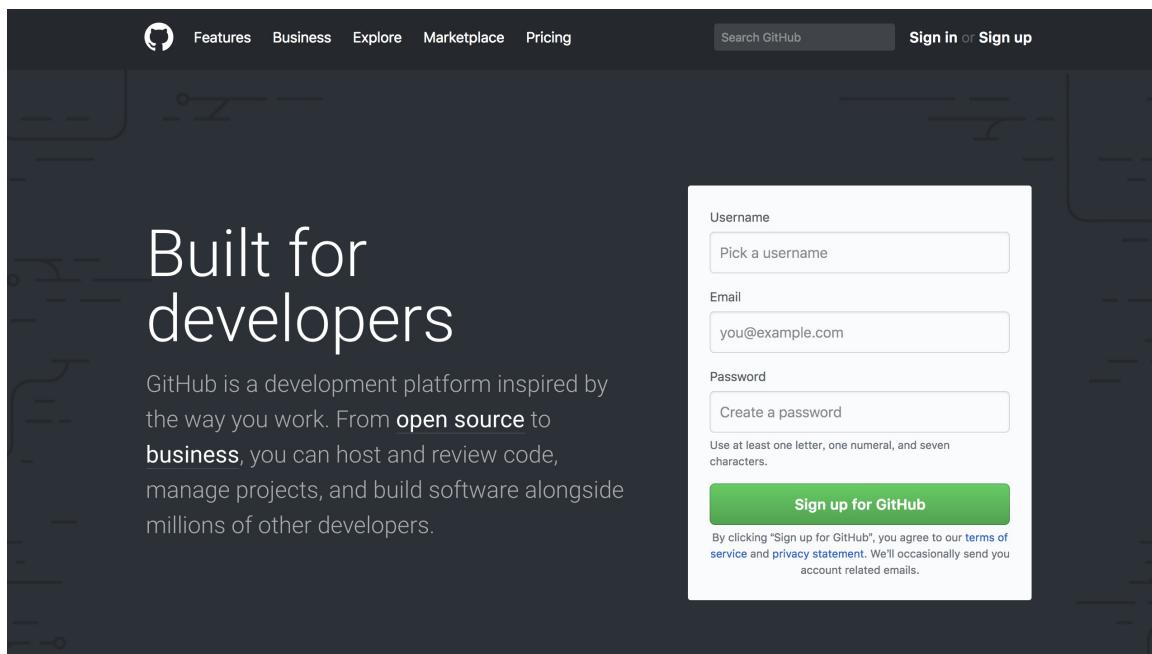
例えば、リモートリポジトリでのファイルの変更をローカルリポジトリに取り込んだり、ローカルリポジトリでのファイルの変更をリモートリポジトリへ反映するなどである。

これらのリポジトリ間のやり取りに関しては4章から順に解説する。

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、"Verify account"でアカウントが正しいことを検証し、"Create an account"をクリックする。

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

Step 1: Set up your account	Step 2: Choose your subscription	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 *

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

You'll love GitHub

- Unlimited public repositories
- Unlimited private repositories
- ✓ Limitless collaboration
- ✓ Frictionless development
- ✓ Open source community

Verify account

このパズルを解いて、実在の人物であることを証明してください。

検証開始

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, "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、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're a few steps away from building better software, @aska0720.

 Completed Set up your account	 Step 2: Choose your subscription	 Step 3: Personalize your experience
--	---	--

Choose your subscription

With tools developers love and the world's largest open source community, there's no wrong choice.

 Free The basics of GitHub for every developer	 Pro Pro tools for developers with advanced requirements
\$0 per month	\$7 per month (view in JPY)

Includes:

- ∞ Unlimited public and private repositories
- ✓ 3 collaborators for private repositories
- ✓ Issues and bug tracking
- ✓ Project management

Are you a **student**? Get access to the best developer tools for free with the [GitHub Student Developer Pack](#).

Includes:

- ∞ Unlimited public and private repositories
- ∞ Unlimited collaborators
- ✓ Issues and bug tracking
- ✓ Project management
- ✓ [Advanced tools and insights](#)

[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

次に、アンケートに答えることができるが、"skip this step"をクリックするとアンケートをスキップすることも可能である。

Welcome to GitHub

You'll find endless opportunities to learn, code, and create, @aska0720.

 Completed Set up a personal account	 Step 2: Choose your subscription	 Step 3: Tailor your experience
--	---	---

What is your level of programming experience?

- None—I don't program at all
- New to programming
- Somewhat experienced
- Very experienced

What do you plan to use GitHub for? (Select up to 3)

- Learning to code
- Learning Git and GitHub
- Host a project (repository)
- Creating a website with GitHub Pages
- Collaborating with my team
- Finding a project to contribute to
- School work / School-related project
- The GitHub API
- I don't know yet
- Other (please specify)

What are you interested in?

e.g. tutorials, android, ruby, web-development, machine-learning, open-source

[Submit](#) [skip this step](#)

最後に、登録したメールアドレスにGitHubからメールが届くので、メールを開いて"Verify email address"をクリックする。

Almost done, @████████! To complete your GitHub sign up, we just need to verify your email address:

[Verify email address](#)

Once verified, you can start using all of GitHub's features to explore, build, and share projects.

Button not working? Paste the following link into your browser: ██████████

You're receiving this email because you recently created a new GitHub account or added a new email address. If this wasn't you, please ignore this email.

[Email preferences](#) · [Terms](#) · [Privacy](#) · [Sign into GitHub](#)

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。

The screenshot shows a GitHub user profile for the account 'aska0720'. The profile picture is a yellow pixelated version of the Tux logo. The 'Overview' tab is selected, showing 0 repositories, 0 projects, 0 stars, 0 followers, and 0 contributions. Below the tabs, it says 'Popular repositories' and 'You don't have any public repositories yet.' Under the repository section, there is a 'Contribution graph' for the last year, which shows one dark green square for May 1st. A note below the graph explains what the squares represent and provides a link to a guide. At the bottom, there is a button to '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

MacにはGitが標準搭載されているため、特にインストール作業を行う必要が無いが、もしインストールされていなかったり最新版に更新したい場合は以下の手順に従う。

Macに後からGitをインストールする方法は主に2つある。

1つ目は <https://git-scm.com/download/mac> からダウンロードし、インストールするという方法である。

2つ目はmacOS用のパッケージマネージャであるHomebrewがインストールされている場合にしかできないが、ターミナル上で

```
$ brew install git
```

とコマンドを入力してインストールするという方法である。

Homebrewのインストール方法に関しては https://brew.sh/index_ja を参照。

Homebrewは他にも便利なアプリケーションをインストールできるため、暇な時間にインストールしておくと良い。

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正常にインストールができている。

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
```

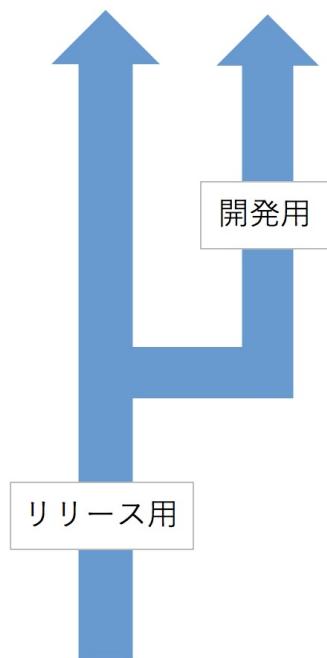
コミットによってGitに記録された変更履歴は、`git log` コマンドで確認できる。

```
$ git log
commit 23bbb672088c1eeaeab1a4c48e81256812edf02e7 (HEAD -> master)
Author: Aska Yukawa <aska0720@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



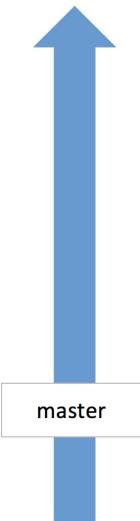
開発用とリリース用を分けたい場合や試験的な機能を作つてみたい場合など、同じアプリケーションでも状況に応じて切り分けたい場合がある。

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
```

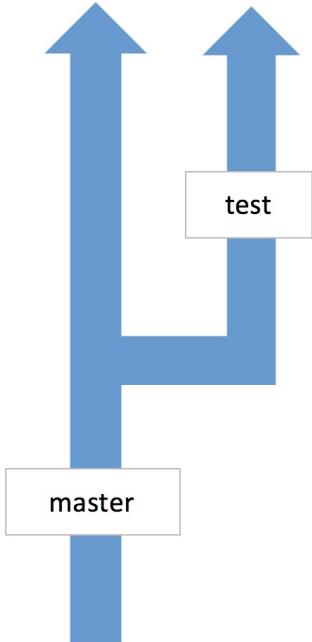


現在自分がいるブランチには左側に * (アスタリスク)が表示される。
また、デフォルトでは master ブランチしか無いため、必要に応じてブランチを作る必要がある。

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 for 'asko0720'. On the left is a large yellow pixelated placeholder image. Below it are profile details: 'Set status' button, 'Edit profile' button, and a note '(Joined 7 minutes ago)'. At the top right is a 'ProTip!' message: 'Updating your profile with your name, location, and a profile picture helps other GitHub users get to know you.' Below the tip are tabs: Overview, **Repositories 0**, Projects 0, Stars 0, Followers 0, Following 0. Underneath are search fields 'Find a repository...', 'Type: All ▾', 'Language: All ▾', and a green 'New' button.

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

Owner	Repository name *
asko0720 ▾	/ hello-git

Great repository names are short and memorable. Need inspiration? How about [ubiquitous-train](#)?

Description (optional)

Public
Anyone can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None ▾ | Add a license: None ▾

Create repository

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 SSH <https://github.com/aska0720/hello-git.git>

Get started by [creating a new file](#) or [uploading an existing file](#). 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/aska0720/hello-git.git
git push -u origin master
```

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

```
git remote add origin https://github.com/aska0720/hello-git.git
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)

ProTip! Use the URL for this page when adding GitHub as a remote.

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

or [HTTPS](#) [SSH](#) <https://github.com/aska0720/hello-git.git>

[a new file](#) or [uploading an existing file](#). We recommend every repository include a `README`, `LICENSE`, and `.gitignore`.

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/<ユーザ名>/hello-git.git (fetch)
origin https://github.com/<ユーザ名>/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/aska0720/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.

Manage topics

1 commit 1 branch 0 releases 1 contributor

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

aska0720 Initial commit Latest commit 4a698b8 6 minutes ago

hello.txt Initial commit 6 minutes 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。

aska0720 / 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

aska0720 Initial commit 4a698b8 8 minutes ago

1 contributor

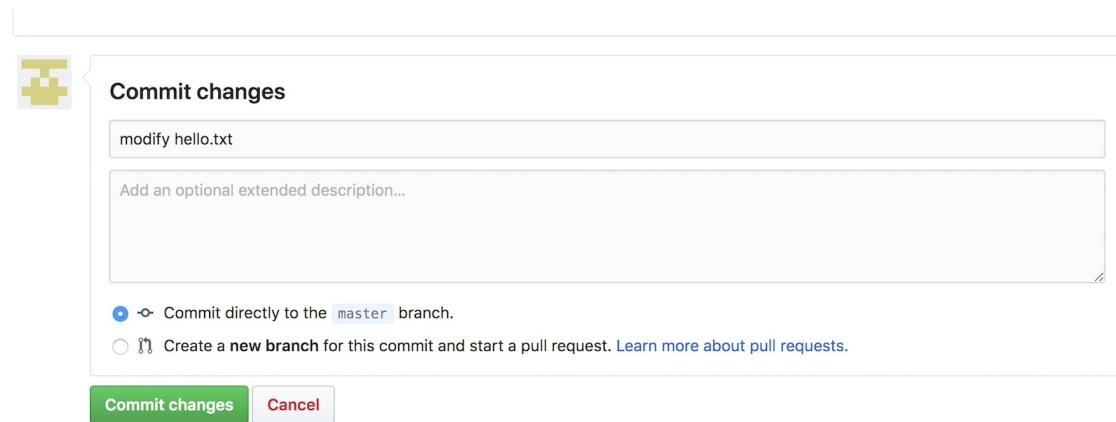
0 lines (0 sloc) 0 Bytes Raw Blame History

今回は `hello, github!!!` と入力する。

The screenshot shows a GitHub repository page for 'aska0720/hello-git'. In the 'Code' tab, there is a file named 'hello.txt' containing the text '1 hello, github!!!'. The interface includes standard GitHub navigation buttons like Watch, Star, Fork, Issues, Pull requests, Projects, Wiki, Insights, and Settings.

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。

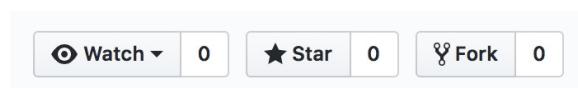
```
$ git pull origin master
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/aska0720/hello-git
 * branch      master    -> FETCH_HEAD
   52e8275..4acdde3  master    -> origin/master
Updating 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のアカウントに複製することをforkと呼ぶ。

今回はrecipeをforkする。

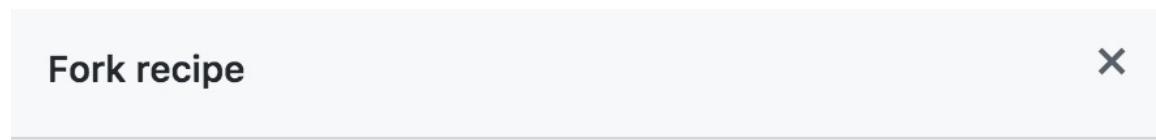
右上にある"Fork"ボタンをクリックする。



そして, forkするユーザを選択する.

ちなみに, Organizations(組織)に所属している場合は以下の選択画面が表示される.

その場合, ユーザを選択するとforkが開始される.



Where should we fork recipe?



asko0720 / recipe
forked from yuka0719/recipe

Watch 0 Star 0 Fork 1

Code Pull requests 0 Projects 0 Wiki Insights Settings

Forking yuka0719/recipe

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 yuka0719/recipe"とfork元リポジトリが記載される.

No description, website, or topics provided.

Manage topics

1 commit 1 branch 0 releases 1 contributor

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

This branch is even with yuka0719:master.

yuka0719 Initial commit Latest commit 70fc91a 2 minutes ago

README.md Initial commit 2 minutes ago

README.md

recipe

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内に複製(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.

Manage topics

1 commit 1 branch 0 releases 1 contributor

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

This branch is even with yuka0719:master.

yuka0719 Initial commit

README.md Initial commit

README.md

recipe

Clone with HTTPS Use SSH
Use Git or checkout with SVN using the web URL.
<https://github.com/aska0720/recipe.git>

Open in Desktop Download ZIP

また、cloneする場所はrecipe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>/recipe.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 recipe
```

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 <好きな料理の名前>_recipe.txt (例: tkg_recipe.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 tkg_recipe.txt
$ git add .
$ git commit -m "add tkg_recipe.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。

The screenshot shows a GitHub repository comparison page for 'yuka0719 / recipe'. At the top, there are buttons for 'Code', 'Issues 0', 'Pull requests 0', 'Projects 0', 'Wiki', and 'Insights'. On the right, there are buttons for 'Watch 0', 'Star 0', 'Fork 1', and a 'Compare' dropdown set to 'base repository: yuka0719/recipe' and 'head repository: aska0720/recipe'.

The main title is 'Comparing changes' with a subtitle '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.' Below this, a message says 'Able to merge. These branches can be automatically merged.'

A green button labeled 'Create pull request' is visible. Below it, summary statistics: '1 commit', '1 file changed', '0 commit comments', and '1 contributor'. A commit log shows a single commit from 'aska0720' on May 21, 2019, titled 'create tkg_recipe.txt' with hash '5cd6351'.

Below the commit log, a message says 'Showing 1 changed file with 0 additions and 0 deletions.' with 'Unified' and 'Split' options. The file list shows 'tkg_recipe.txt' with 0 changes. A message at the bottom says 'No commit comments for this range'.

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。

The screenshot shows the GitHub interface for creating a pull request. At the top, the repository 'yuka0719 / recipe' is selected. The navigation bar includes 'Code', 'Issues 0', 'Pull requests 0', 'Projects 0', 'Wiki', and 'Insights'. In the top right, there are buttons for 'Watch', 'Star', and 'Fork'. The fork count is 1.

The main section is titled 'Open a pull request' with the sub-instruction 'Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.' Below this, a configuration bar shows 'base repository: yuka0719/recipe', 'base: master', 'head repository: aska0720/recipe', and 'compare: master'. A green checkmark indicates 'Able to merge. These branches can be automatically merged.'

The central area is a text editor for the file 'create tkg_recipe.txt'. It has tabs for 'Write' and 'Preview', and various rich text editing tools. Below the editor is a comment input field labeled 'Leave a comment' and a file attachment area labeled 'Attach files by dragging & dropping, selecting or pasting them.' A checkbox 'Allow edits from maintainers.' is checked, and a large green button labeled 'Create pull request' is visible.

Below the editor, summary statistics are shown: '1 commit', '1 file changed', '0 commit comments', and '1 contributor'. The commit details show a single commit from 'aska0720' on May 21, 2019, with the file 'create tkg_recipe.txt' and hash '5cd6351'.

The 'File' tab is selected, showing 'Showing 1 changed file with 0 additions and 0 deletions.' The file content is listed as 'No changes.' Below this, a message states 'No commit comments for this range'.

そうすると、Pull Requestが作成される。

create tkg_recipe.txt #1

Open aska0720 wants to merge 1 commit into `yuka0719:master` from `aska0720:master`

Conversation 0 Commits 1 Checks 0 Files changed 1 +0 -0

aska0720 commented just now
No description provided.

create tkg_recipe.txt 5cd6351

Add more commits by pushing to the `master` branch on `aska0720/recipe`.

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

Leave a comment

Close pull request Comment

ProTip! Add comments to specific lines under [Files changed](#).

Reviewers
No reviews

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

Notifications
Unsubscribe

You're receiving notifications because you authored the thread.

1 participant

Allow edits from maintainers. Learn more

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。

create tkg_recipe.txt #1

Open aska0720 wants to merge 1 commit into [yuka0719:master](#) from [aska0720:master](#)

Conversation 0 Commits 1 Checks 0 Files changed 1 +0 -0

aska0720 commented a minute ago First-time contributor + ...

No description provided.

create tkg_recipe.txt 5cd6351

Add more commits by pushing to the **master** branch on [aska0720/recipe](#).

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](#).

Write Preview

Leave a comment

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

Close pull request Comment

ProTip! Add `.patch` or `.diff` to the end of URLs for Git's plaintext views.

Reviewers: No reviews

Assignees: No one—assign yourself

Labels: None yet

Projects: None yet

Milestone: No milestone

Notifications: Subscribe

You're not receiving notifications from this thread.

1 participant

Lock conversation

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。

The screenshot shows a GitHub pull request page for the repository `yuka0719 / recipe`. The pull request is titled `create tkg_recipe.txt #1`. The status is `Open`, indicating that `aska0720` wants to merge 1 commit from `aska0720:master` into `yuka0719:master`. The commit `create tkg_recipe.txt` was pushed by `aska0720` a minute ago. The commit message is `No description provided.`. The commit hash is `5cd6351`.

On the right side of the pull request page, there is a sidebar with various settings and options:

- Reviewers:** No reviews.
- Assignees:** No one—assign yourself.
- Labels:** None yet.
- Projects:** None yet.
- Milestone:** No milestone.
- Notifications:** You're not receiving notifications from this thread. A `Subscribe` button is available.
- Participants:** 1 participant (represented by a yellow profile icon).
- Lock conversation:** A lock icon is present.

At the bottom of the pull request page, there is a note: `💡 ProTip! Add .patch or .diff to the end of URLs for Git's plaintext views.`

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.

The screenshot shows a GitHub pull request page for the repository `yuka0719 / recipe`. The pull request is titled `create tkg_recipe.txt #1` and has been merged. The merge message is: `yuka0719 merged 1 commit into yuka0719:master from aska0720:master just now`. The commit hash is `5cd6351`. The pull request has 0 commits, 0 checks, and 1 file changed. The conversation tab is selected, showing a comment from `aska0720` that says "No description provided." Below the conversation, there is a link to the commit `create tkg_recipe.txt`. On the right side, there are sections for Reviewers, Assignees, Labels, Projects, Milestone, and Notifications. The Notifications section indicates that the user is receiving notifications because they modified the open/close state. There are also sections for 2 participants and a lock conversation button.

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。

yuka0719 / recipe

Watch 0 Star 0 Fork 1

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Filters ▾ is:issue is:open Labels 8 Milestones 0 New issue

ⓘ 0 Open ✓ 0 Closed Author ▾ Labels ▾ Projects ▾ Milestones ▾ Assignee ▾ Sort ▾

!

There aren't any open issues.

You could search [all of GitHub](#) or try an [advanced search](#).

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 a GitHub repository page for 'yuka0719 / recipe'. The main header includes the repository name, a 'Watch' button (0), a 'Star' button (0), a 'Fork' button (1), and a 'Code' link. Below the header are links for 'Issues 0', 'Pull requests 0', 'Projects 0', 'Wiki', 'Insights', and 'Settings'. The main content area features a large text input field with placeholder text 'ラーメンのレシピを追加'. Below this are 'Write' and 'Preview' buttons, and a toolbar with icons for bold, italic, code blocks, lists, and other rich text features. A list of steps is entered into the text area:

- スープを作る
- トッピングを作る
- 麺を茹でる
- 器に盛り付ける

Below the text area is a note: 'Attach files by dragging & dropping, selecting or pasting them.' To the right of the text area are several configuration sections: 'Assignees' (None yet), 'Labels' (None yet), 'Projects' (None yet), and 'Milestone' (None yet). At the bottom left is a note about styling support, and at the bottom right is a green 'Submit new issue' button.

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

① Open yuka0719 opened this issue just now · 0 comments

yuka0719 commented just now

- スープを作る
- トッピングを作る
- 麺を茹でる
- 器に盛り付ける

Leave a comment

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

Comment

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Notifications
Unsubscribe

You're receiving notifications because you authored the thread.

1 participant

Lock conversation

Pin issue

Transfer issue (Beta)

Delete issue

.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, npmが生成する node_modules やComposerが生成する vendor のようなモジュール群, .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
```

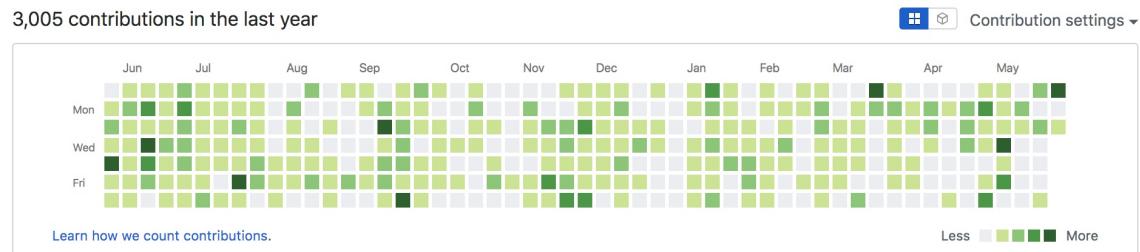
上記のようにすることで, ステージングしても hoge.txt はステージングされないことが分かる.

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

コントリビューション

GitHubでは「pushする」「PRを送る」「mergeする」「issueを立てる」などの作業を行う度にコントリビューションに反映される。

コントリビューションというのは、以下の画像のように、「何年何月何日にどれだけ稼働したか」という情報の可視化であり、これによりモチベーションが上がる(人もいる)。



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である。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 blame
- git cherry-pick
- git rebase
- git reflog
- git reset
- git revert
- git rm
- git stash
- git tag

Git/GitHubを勉強できるWeb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/GitHub関連のアプリケーションの紹介



- [SourceTree](#) ... GitをGUIで操作するためのアプリケーション。しかし、慣れてしまえばCUIの方が使い勝手が良いよう感じる。



- [GitHub Desktop](#) ... Webブラウザ上でも不自由なく GitHubを操作できるが、GitHubのデスクトップアプリケーション。



- [Sourcegraph](#) ... WebブラウザのGitHub上でソースコードを見る場合に、分かりやすく表示してくれるブラウザ拡張。



- [Isometric Contributions](#) ... GitHubのコントリビューションを立体的に表示するChrome拡張。