Gitの使い方

‘14

紅狼

‘14

[1. Gitとは？ 2](#_Toc383451029)

[1.1. Gitの利便性 2](#_Toc383451030)

[1.2. 状態を保存するリポジトリ 2](#_Toc383451031)

[1.3. 変更を記録するコミット 3](#_Toc383451032)

[1.4. ワークツリーとインデックス 3](#_Toc383451033)

[2. Gitのインストール(Windows), 初期設定 4](#_Toc383451034)

[2.1. Gitのダウンロード(Nextの回数等はver.により変わる可能性あり(2014/03/20現在:1.9.1)) 4](#_Toc383451035)

[2.2. Gitの初期設定 5](#_Toc383451041)

[3. [初級]Gitの使い方 6](#_Toc383451044)

[3.1. clone(git clone [URL] [ディレクトリ]) 6](#_Toc383451045)

[3.2. pull(git pull) 6](#_Toc383451046)

[4. Gitの使い方 6](#_Toc383451047)

[4.1. リモートリポジトリの作り方 6](#_Toc383451048)

[4.2. ローカルリポジトリの作成→リモートリポジトリの登録 7](#_Toc383451049)

[5. GitHubを使って練習してみよう 8](#_Toc383451050)

[5.1. アカウント登録 8](#_Toc383451051)

[5.2. リポジトリをつくろう 8](#_Toc383451052)

[5.3. クローンしよう 9](#_Toc383451053)

[5.4. commitしよう 9](#_Toc383451054)

[5.5. pushしよう 10](#_Toc383451055)

[6. より便利にするために 10](#_Toc383451056)

[6.1. Gitの出力に色をつける 10](#_Toc383451057)

[6.2. diff(差分を表示する) 10](#_Toc383451058)

[6.3. log(commitのlogを表示する) 10](#_Toc383451059)

[6.4. branch 10](#_Toc383451060)

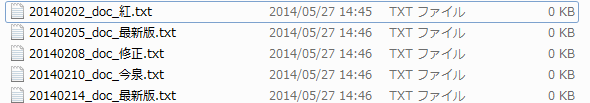
# Gitとは？

Gitは, 一言で言うと分散型バージョン管理システムといいます. 離散型バージョン管理システムってなに？ Gitを使うとどう便利なの？どうやってるの？ということをこの項では説明していきます.

## Gitの利便性

***～Gitがない世界～***

プログラムのコードを書いていくと, “このプログラム, 前は動いてたのにいろいろ追加してる間に動かなくなった！”ということが多々あります. (あって欲しくないのですが) では, 節目節目で動くようなファイルを残しておけば, 前の動く時のファイルに戻してそこから編集をすることができます. ではファイルを残してみましょう.



こうなりました. 非常にカオスです ファイル名を間違えると何が最新版なのかよくわかりません. どのような編集をしたのかもよくわかりません. また, 共同で開発をしていると, “自分が編集した部分を間違って他の人が上書きしてしまって気づかぬうちに自分のファイルが変わっていた.“ということもありえます. 非常に恐ろしいです.

***～分散型バージョン管理システムGitがある世界～***

* Gitでは, 他人が自分のファイルを編集するときに警告がでます.
* Gitでは, 誰が, どのような編集をしたかをオフラインで(分散型)確認することができます.
* Gitでは, 簡単に最新バージョンを取得, いつ編集したかもわかります(バージョン管理システム)

## 状態を保存するリポジトリ

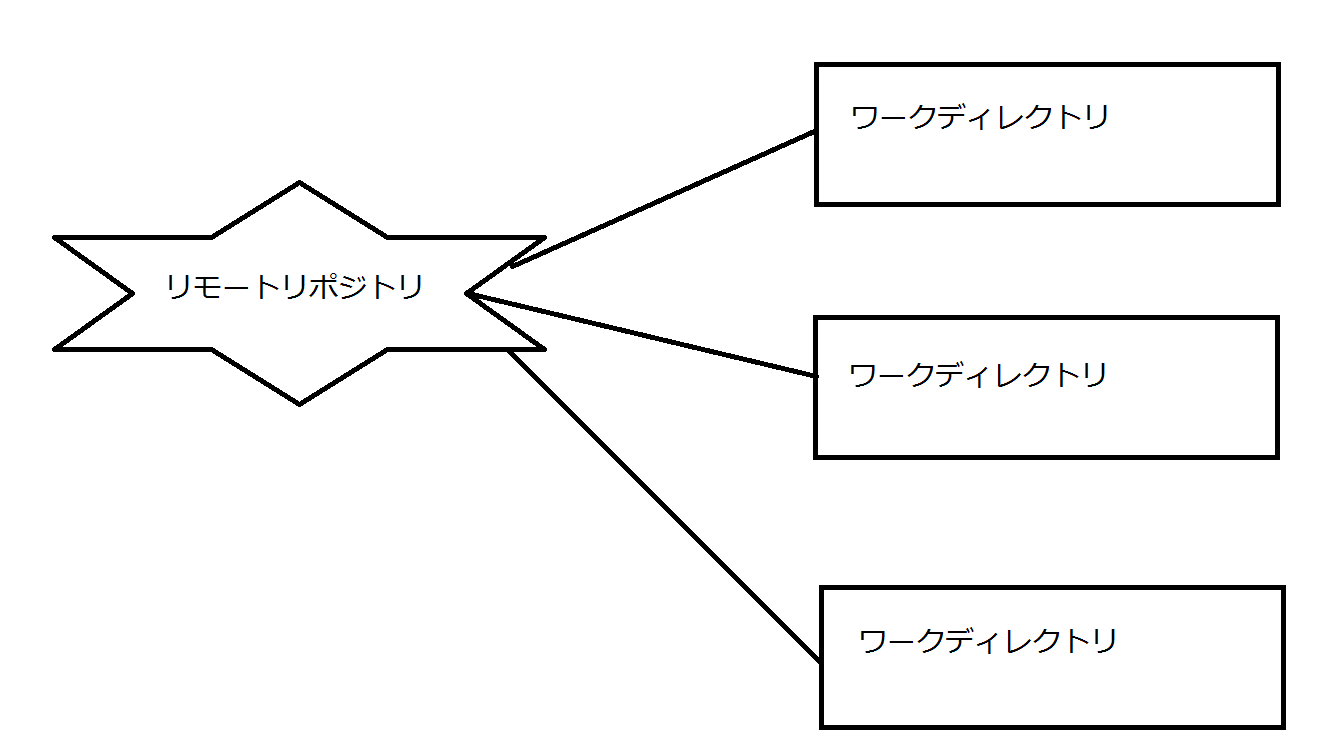
リポジトリとは, ファイルやディレクトリの状態を保存する場所です. (どんな変更がされたかなどはここに入っています. ) Gitのリポジトリには, 以下の2つがあります

* リモートリポジトリ

サーバーを使って複数人で共有する

* ローカルリポジトリ

ユーザーが使用するために個人のPCにおいておくリポジトリ(操作するのはこちらです)



リポジトリを2つに分けることによって, 中途半端な状態でサーバーにアップロードせず, ちゃんと動作する状態を常にサーバーに残すことができます. こうすることで, プログラムをサーバーから引き出しても常に動く最新版のプログラムを取得することができます.

## 変更を記録するコミット

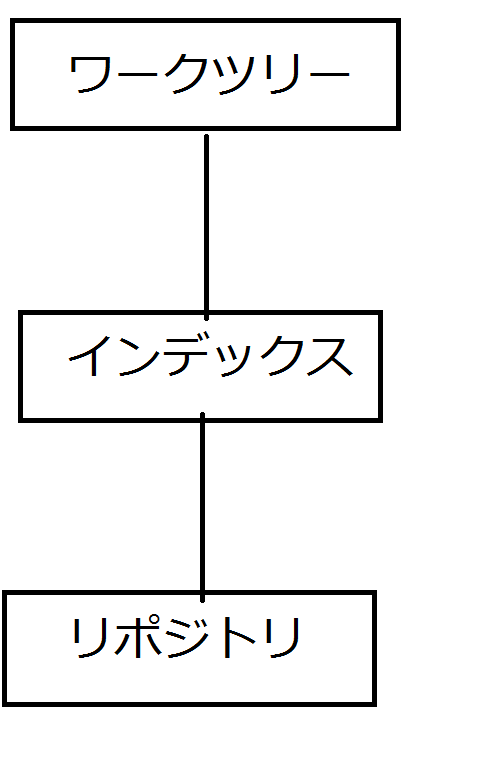
ファイルやディレクトリの追加・変更をリポジトリに記録することをコミットといいます.

コミットを実行すると, 前回コミットした時の差分を記録したリビジョン(コミット)と呼ばれるものが作られます. また, コミットをするさいにメモ書きを残すことができます(変更内容の要約, 理由など)

## ワークツリーとインデックス

Gitで, 実際にユーザーが操作する部分をワークツリーと言います. リポジトリとワークツリーの間に, コミットするために準備をするインデックスと呼ばれる部分があります. このインデックスがあることにより, ディレクトリの中の一部のファイルのみをアップロードしたり, メモ書きなど, 必要のないファイルのコミットを防ぐことができます.

注）インデックスに登録されていないファイルはコミットできません



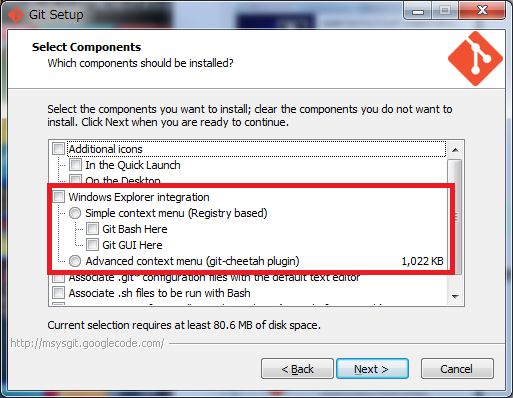
# Gitのインストール(Windows), 初期設定

## Gitのダウンロード(Nextの回数等はver.により変わる可能性あり(2014/03/20現在:1.9.1))

### <http://git-scm.com/> の Download for Windows からダウンロードする.

### ダウンロードした.exeファイルを実行し, Nextを押していきます下記の画像の部分に注意してください.

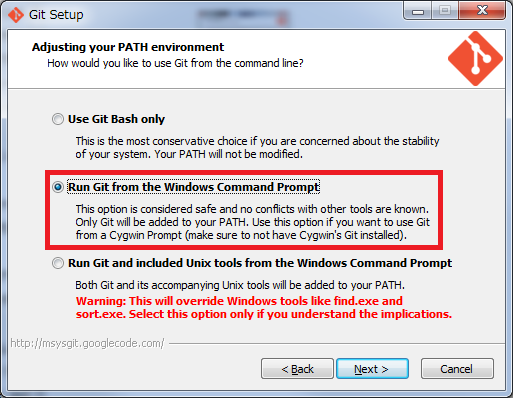
### 下図では, 好きな様にチェックを付けてもらって構いません. 赤い四角のところをつけておくと便利かもしれません(この本(?)では扱いません)



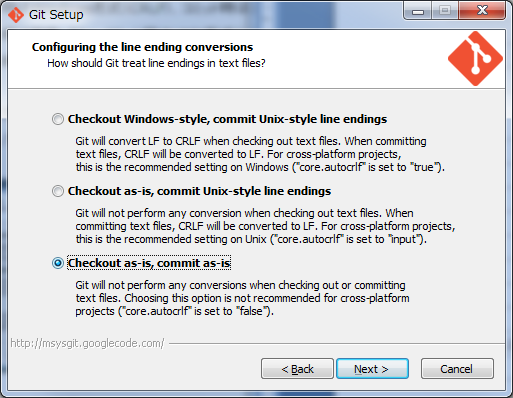
### 下図では, 赤い四角のところにチェックを付けるのがオススメです. 上から簡単に説明すると

* + - GitBashというツールのみでGitを使う
    - Windowsのコマンドプロンプトでも使えるようにする
    - WindowsのコマンドプロンプトにUnixのコマンドを使えるようにする

となっています.



### 最後に, 下図でも好きなところのチェックを入れてもらって構いません. よくわからない場合は一番上がオススメです(ちなみに自分は下図のようになってました)



## Gitの初期設定

### インストールしてでてきたフォルダの中にGitBashを実行します.

### デフォルトのユーザー名とメールアドレス名を登録します. 自分の場合

ユーザー名がkurenaif

メールアドレスがfujiwara\_koh@zeus.eonet.ne.jp

なので

$ git config --global user.name kurenaif

$ git config --global user.email fujiwara\_koh@zeus.eonet.ne.jp

となります. (わかりづらいですがハイフンは2個です)

$git config --global --list

でちゃんと入力できたかを確認するといいです.

# [初級]Gitの使い方

この項では”アップロード側”ではなく, “ダウンロード側”の説明をします

## clone(git clone [URL] [ディレクトリ])

cloneコマンドは既存のリポジトリをコピーしてきます. では, ネット上にあるリポジトリをcloneしてみましょう. (これにはGitHubを使用していますが, 今は特に気にしなくていいです. )

$ git clone https://github.com/kurenaif/HowToGit

lsコマンドでディレクトリが増えたか見てみましょう. (Unixコマンドつかえます)

$ ls

それがコピーしてきたディレクトリです.

## pull(git pull)

pullコマンドでは, .git/configに設定されたリモートリポジトリの変更を取得(し, 現在のブランチにマージします. ブランチ, マージは後で説明します)例を挙げるのが難しいので, 複数人でpush(後述)したりpullしたりしてみて試してみましょう.

$ git pull

cloneはリポジトリを持ってきて生成するのに対し, pushは更新する. この違いは抑えておきましょう

# Gitの使い方

こちらではGitの作り方, 書く人が見る項目です.

## リモートリポジトリの作り方

$ mkdir -p [リポジトリ名].git (ディレクトリを作ります)

$ cd [リポジトリ名].git (ディレクトリの移動をします)

$ git --bare init (git用に初期化します)

いろいろあると思いますが, とりあえず次へ進んでみてください.

## ローカルリポジトリの作成→リモートリポジトリの登録

$ mkdir [リポジトリ名] (.gitは不要です)

$ cd [リポジトリ名]

$ git init

$ echo “[文字列]” > readme

$ git add readme (add:インデックスにファイルを登録します(1.4参照) )

$ git commit –m “[メモ書きの内容]” (commit:コミット(1.3参照)します)

$ git remote add origin [URL] (URLにoriginというニックネームを付ける)

$ git push origin master (origin(上のURL)にmasterブランチをアップロードする)

一気に書いてしまってすみません. 最後のmasterブランチと言うのは気にしないでください. 後で説明します. よく分からなかったらとりあえずmasterブランチでやっときゃあ大丈夫です. コマンドの説明は上記で十分じゃないかと思います. 一つ言いたいことがあるので書いておきます.

こまめにcommit 慎重にpush (commitはゲームで言うクイックセーブみたいなものです. 残せば残すほどその地点に復帰しやすくはなりますが, 過剰なcommitは探すのが大変になるのでほどほどに・・・. pushはみんなが見るリモートリポジトリにアップロードします. みんながダウンロードすることを頭においてpushしましょう. もちろんバグがあるものや, 編集途中のものをpushしたらみんな困ります. あとcommitしたのを戻すのは簡単ですが, pushしたのを戻すのはめんどうです・・・(経験談) )

ファイルをコミットしたいときはaddした後, commitすればいいです.

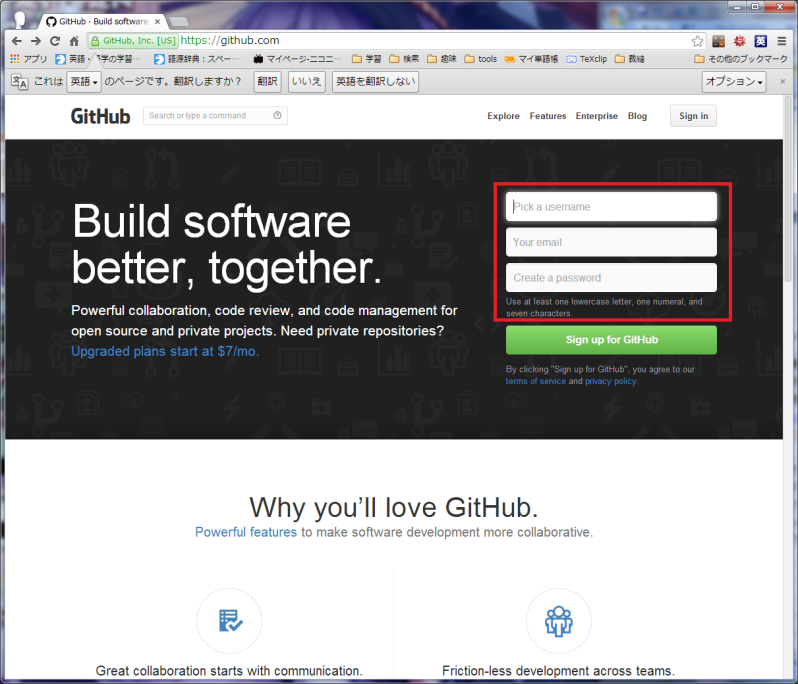
必要があればpushすればリモートディレクトリに反映されます. （後でやるのでここでは説明は省略します 上のやつとほぼ一緒ですし.

# GitHubを使って練習してみよう

やってみないと意味がわかりません！とりあえずやってみましょう. 研究室のサーバーは研究室でしかアクセスできない可能性があるのでGitHubを借りましょう！（便利ですし）

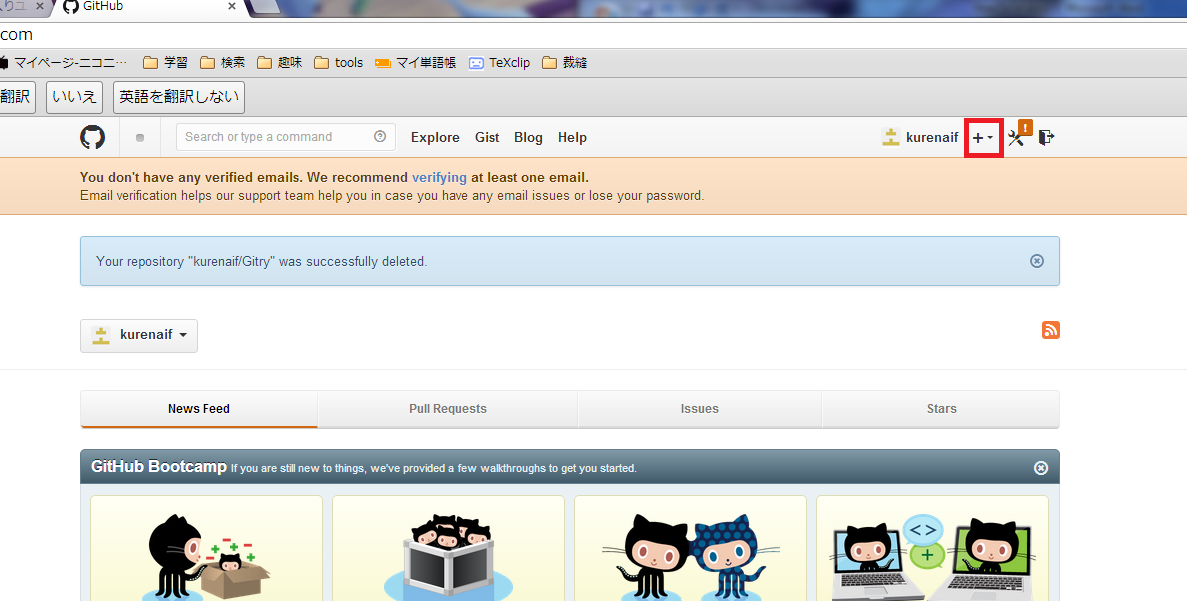
## アカウント登録

1. <https://github.com/> にアクセスする
2. あとはここに適当に入力して Sign up for GitHub をクリックすればOKです



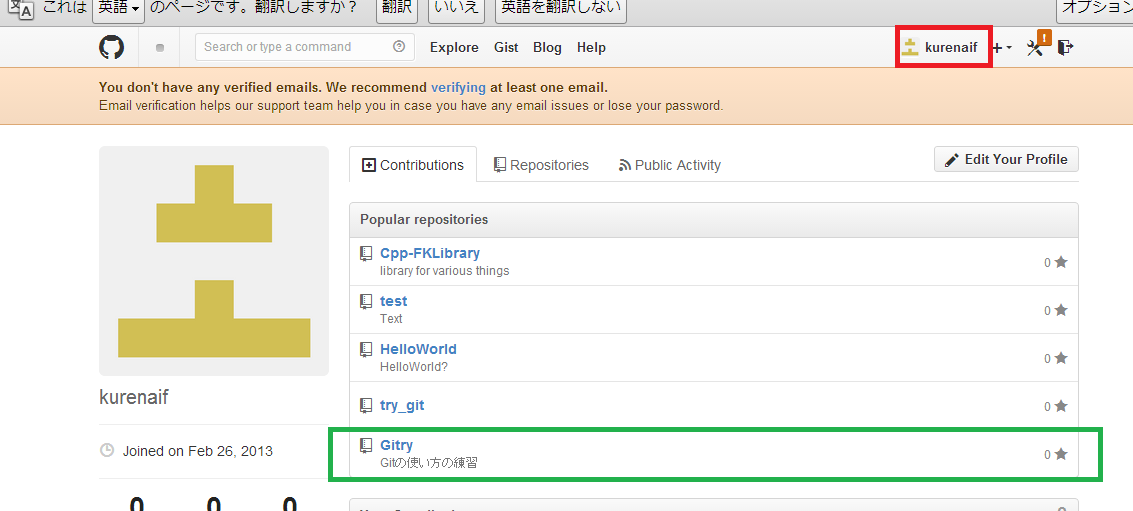
## リポジトリをつくろう

1. ログインした後,画面上部の赤い四角をクリックしてNew Repositoryをクリックしてください
2. 適当に入力した後, Create repository をクリックしてください



## クローンしよう (自分はGitryリポジトリを作りました. )

1. 赤い四角の部分をクリックした後, 下の緑の四角の部分をクリックします. (次ページヘ)



1. そのページのURLをコピーし, GitBashを起動します.
2. ワークディレクトリを作りたい場所にcdで移動し, cloneします自分の場合, デスクトップに作って後で消したいので

$ cd Desktop

$ git clone https://github.com/kurenaif/Gitry

としました. (ちなみにGitBash上でペーストは Shift+Insertです)

## commitしよう

1. 出てきたフォルダの中に適当にファイルを作ってください. (適当に中も書いといてください)

(自分の場合, Hello.txtにWorldという内容にしました)

1. GitBashにて, cdコマンドでさっきできたディレクトリに移動してください

$ cd Gitry

1. いま, あなたのワークツリーにはHello.txtがありますが, インデックス上にはなにもありません(意味がわからない時は1.4の図を見てください. ) commitするためには, インデックスに登録しないといけないので, addします.(1.4参照)

$ git add --all

git add --all とすれば, 追加したファイルや, 変更したファイルをすべてインデックスに追加できます. “4.2 ”でやったように, ファイル名を直接指定してやってもらっても構いません.

1. 続けてcommitします.

$ git commit -m “First commit”

First Commitはメモ書きの内容です. 好きなように変更してください.

## pushしよう

1. 最後に, pushします. これでリモートリポジトリに反映されます. (originというのは, GitHubのさっきのURLのことです. masterは気にしなくていいです. )

$ git push origin master

1. 先ほどのページに戻ってみると, 追加したファイル(Hello.txtが追加されていることがわかります. )

さらにファイルの変更があればまたaddしてからcommitしたらいいです. そして必要があればpush

# より便利にするために

最低限上記の事ができればアップロードダウンロードはできますが, この項目ではGitをより便利にするために書く項目ですちょくちょく更新すると思います.

## Gitの出力に色をつける

$ git config --global color.ui true

ユーザー設定です. 便利です.

## diff(差分を表示する)

冒頭で説明したどのような変化があったか. のコマンドになります. いくつか種類があるので紹介します.

$ git diff

index と working treeの差分を表示します. (working treeは, コミットする前のファイル, indexはcommitした後のファイル)

## log(commitのlogを表示する)

$ git log

## branch

私が, Gitで一番便利で, 難しく, 危険でもあると思うbranchです. 使うかどうかよくわからないのでとりあえず保留. これの内容はかなり重いので, これだけで1つ本(?)をつくろうと思います.

[参考資料]

サルでも分かるGit入門 <http://www.backlog.jp/git-guide>

Wikipedia <http://ja.wikipedia.org/wiki/Git>

git 公式ページ <http://git-scm.com/>

Gitの使い方(見出し) <http://transitive.info/article/git/>

gitリポジトリの作成方法 <http://taichino.com/memo/1587>