GITコマンド一覧

基本コマンド

git clone <URL>
　　<URL>が示すリモートリポジトリを複製する。
git add <パス>
　　作業ツリーの<パス>以下のファイルをインデックスにコピーする。
git commit
　　インデックスの変更点を一括してカレントブランチに適用する。
git status
　　作業ツリーとインデックスとカレントブランチの状態を表示する。
git status -s
　　作業ツリーとインデックスの状態を簡潔に表示する。
git status -uall
　　サブディレクトリの中のファイルも表示する。
git diff<パス>
　　作業ツリーとインデックス間における<パス>の変更箇所を表示する。
git diff –cached <パス>
　　インデックスとカレントブランチ間における<パス>の変更箇所を表示する。
git log
　　コミット履歴を一覧表示する。
git log –oneline
　　コミット履歴を簡潔に一覧表示する。

取消系コマンド

git checkout HEAD <パス>
　　作業ツリーの<パス>以下のファイルを元に戻す（編集内容の破棄）

そのディレクトリ全部の編集を取り消す。

```
$ git checkout .
```

そのファイルだけ編集を取り消す。

```
$ git checkout hoge.html
```

git reset HEAD <パス>

インデックスの<パス>以下のファイルを元に戻す（git addコマンドのアンドゥ）

git reset –soft HEAD^

直前のコミットを取り消す（git commitコマンドのアンドゥ）

git commit –amend

直前のコミットをやり直す

git reset のモードについて

–soft

作業ブランチとインデックスは変化しない。

–mixed

インデックスがリセットされる。

–hard

作業ブランチもインデックスもリセットされる。

（他に –merge、–keepがある）

git reset –hard <特定のコミットのハッシュ値>

特定のコミットまで戻す

（例）$ git reset –hard 2694e7dd0c3036fbdef4df42c07e7042278010fa

ローカルブランチの管理コマンド

git branch

ブランチの一覧を表示する。

git branch <ブランチ名>

<ブランチ名>という名前のブランチを作る。

git branch -m <古い名前> <新しい名前>

ブランチの名前を<古い名前>から<新しい名前>に変更する。

git branch -d <>

<ブランチ名>という名前のブランチを削除する。

git checkout <>

<ブランチ名>をカレントブランチにする。

git checkout -b <>

新規<ブランチ名>を作り、それをカレントブランチにする。

ローカルのブランチとリモートのブランチの同期

リモート追跡ブランチ

git branch -r

リモート追跡ブランチの一覧を表示する。リモート追跡ブランチとは、リモートリポ

ジトリにあるブランチの読み取り専用のコピー。ローカルに作成される。

ローカルリポジトリとリモートリポジトリの間の同期

git fetch origin

リモートoriginのすべてのリモート追跡ブランチが最新の状態にアップデートされる。

```
$ git fetch origin
$ git status
```

1) ローカルが進んでいる場合

ローカルブランチがリモート追跡ブランチよりも進んでいる場合、以下のような表示になる。

```
# On branch master
# Your branch is ahead fo 'origin/master' by 3 commits.
#
nothing to commit (working directory clean)
```

==>「あなたのブランチはorigin/masterよりも3コミット分だけ進んで(ahead)います。」

この場合はpushできる。

```
$ git push origin master
```

2) リモートが進んでいる場合

もし、リモート追跡ブランチの方が進んでいる場合は、以下のような表示。

```
# On branch master
# Your branch is behind 'origin/master' by 2 commits, and can be fast-forwarded.
#
nothing to commit (working directory clean)
```

==>「あなたのブランチはorigin/masterよりも2コミット分だけ遅れて(behind)いて、かつ早送り(fast-foward)可能です。」

この場合は、merge。

```
git merge origin/master
```

※git pull は本物のマージが始まるかもしれないので、初心者はやらないほうがいいみたい。

3) ローカルとリモートが分岐している場合

ローカルブランチとリモート追跡ブランチが分岐(diverge)している場合、以下のような表示になる。

```
# On branch master
# Your branch and 'origin/master' have diverged,
# and have 3 and 4 different commit(s) each, respectively.
#
nothing to commit (working directory clean)
```

==>「あなたのブランチとorigin/masterは分岐(diverge)しており、それぞれ3個および4個の異なるコミットを持っています。」

5

「Ruby on Rails 環境構築ガイド」p.126より抜粋

参考図書 「Ruby on Rails 環境構築ガイド」黒田努・著 インプレス・ジャパン 2013.3.21 初版第1刷

リモート追跡ブランチのクリーンアップ

git branch -r とすると、リモートで削除されたブランチ名がローカルに残っていることがある。その場合は、以下でクリーンアップできる。

```
$ git fetch --prune
```

もしくは

```
$ git remote prune origin
```

ローカルをリモートに強制的に合わせる

```
$ git fetch origin
$ git reset --hard origin/master
```

枝分かれブランチの変更をmasterに取り込む

```
$ git branch
```

まず、現在のブランチを確認する。

現在、*versin2* ブランチにいるとして、

```
$ git checkout master
$ git merge version2 --no-ff
```

これで、*version2* ブランチの内容が *master* に取り込まれる。

ブランチの状況を表示する

以下のコマンドを入力する。（エイリアスの設定）

```
$ git config --global alias.graph "log --graph --date-order --all --pretty=format:'%h
%Cred%d %Cgreen%ad %Cblue%cn %Creset%s' --date=short"
```

これで、以下のコマンドで、ブランチの状況（コミットグラフ）を表示できる。

```
$ git graph
```

(注) 2021.05.26 追加

以下のようにすると、ブランチのログが見える

```
$ git log --graph --oneline
```

リモートに存在するブランチを削除する

ローカルのブランチ version2 を削除。

```
$ git branch -d version2
```

リモートのブランチを確認

```
$ git branch -r
  origin/HEAD -> origin/master
  origin/master
  origin/version2
```

リモートのブランチを削除

```
$ git push --delete origin version2
```

リモートに存在するブランチをローカルに取り込む

ここでは、以下のような状況を想定している。

ローカル

```
$ git branch
* master
```

リモート

```
$ git branch -r
origin/HEAD -> origin/master
origin/master
origin/ver2
```

コマンド

```
$ git checkout -b ローカルでのブランチ名 チェックアウトするブランチ名
```

まあ、基本、両者のブランチ名は同じでいいだろう。

リモートのブランチを取り込み、そのブランチでチェックアウトするコマンド

```
$ git checkout -b ver2 origin/ver2
$ git branch
  master
* ver2
```

gitignore

すでにトラックされているファイル・ディレクトリをイグノアするには、.gitignore をつくってから、

```
$ git rm -r --cached .（ピリオド）
$ git add .
$ git commit -m "gitignoreを作り直した"
$ git push -u origin main
```

などとする必要がある。

このサイトが参考になる。

- gitignoreまとめ

git push -u origin master の -u オプションについて

ここの記述がわかりやすい。

- git push -u オプションで"上流ブランチ"を設定

  結論としては、-u オプションをつけた方が安心かも。

git status で文字化けする

以下のコマンドを実行。

```
$ git config --global core.quotepath false
```

(参考) git statusで表示する日本語のファイル名が文字化けする場合の対応方法

git でユーザー名とパスワードを毎回聞かれないようにする

以下のコマンドを実行すると、最初だけユーザー名パスワードを きかれるが、次からは聞かれなくなる。

```
$ git config credential.helper store
```

~~あるいは、初回に git clone するときに、以下のようにする。~~ (このやり方はできなくなった。2021.09.06追記)

```
$ git clone https://<username>:<pasword>@github.com/SeiichiN/php-lesson.git
```

(参考) git パスワードを毎回聞かれる問題をHTTPSでも解決

カテゴリー: Git, memo, コマンド

タグ: command, Git, コマンド

カウント: 37

Apache cmd command

DateTimeFormatter Docker

emacs GHC Git Haskell HomeBrew
install java jsp JupyterNotebook linux
LocalDate mac MacOS mamp
mariadb Monterey mysql
oracle php php7.4 phpmyadmin
Python Python3 remote servlet sql
sqlplus stop tomcat ubuntu
w3m windows wordpress wps-office
xampp zip オラクル パス 文字化け
日本語化