Git チュートリアル

[Git]

・この本からのメモ

リポジトリの作成

・subversion などたいていの VCS はリポジトリと作業ツリーは別の場所にあるが、Git では、どちらも .git ディレクトリに存在する
・Git でリポジトリを作るには、まずプロジェクトコードを格納したい場所を決める (例として、mysite とする)
・mysite ディレクトリを作成し、その中で git init とする

```
$ mkdir mysite
$ cd mysite
$ git init
Initialized empty Git repository in /home/piroto/git_test/mysite/.git/
```

・これでリポジトリが作成された
・リポジトリのあらゆるメタデータを格納する .git ディレクトリ、mysite が リポジトリからチェックアウトしたソースの作業ツリーとなる

変更を加える

ファイル (index.html) の追加

index.html

```
<html>
<body>
<h1>Hello World.</h1>
<body>
</html>
```

リポジトリへの追加

・２段階の手順

■ git add

・リポジトリへの追加を指示
・１つ以上のファイル名を指定

```
$ git add index.html
```

■ git commit

・コミットの実行
・コミットは、リポジトリに格納される１つ１つの履歴
・コミットでは、ユーザー名、メールアドレス、コミットメッセージが格納される

```
$ git commit -m "add in hello world HTML"
[master (root-commit) 6b4f2e7] add in hello world HTML
```

```
 1 files changed, 5 insertions(+), 0 deletions(-)
 create mode 100644 index.html
```

■ git log

・コミットの確認
・先頭行はコミット名、SHA-1 のハッシュ値によりユニークに識別

```
$ git log
commit 6b4f2e70a4e1950083000148ec8b625be8a86eee
Author: YAGI Hiroto <piroto@ryujyu.typea.info>
Date:   Mon Dec 6 22:10:38 2010 +0900

    add in hello world HTML
```

プロジェクト作業の開始

index.html の編集

・<head> と <title> を追加

```
<html>
<head>
<title>Hello World in Git</title>
</head>
<body>
<h1>Hello World.</h1>
</body>
</html>
```

git status

・作業ツリーを Git がどう認識しているか確認
・Changed but not updated (変更されているが更新されていない) に分類されているが、コミットするには、変更をステージ (stage) する必要がある
・ステージするとコミットの準備を整えたことになる

```
$ git status
# On branch master
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   index.html
#
no changes added to commit (use "git add" and/or "git commit -a")
```

Git ではプロジェクトコードが保持される場所が３つある

1. 作業ツリー
2. インデックス (ステージングエリア)
3. リポジトリ

ステージングエリアは、作業ツリーとリポジトリの間にあるバッファ。リポジトリにコミットしたい変更だけをアレンジ。

変更をステージ (再度 git add) する

・Change to be commitd (コミットされる変更) に状態が変わる

```
$ git add index.html
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   index.html
#
```

コミット

■ git commit

・-m パラメータは複数追加でき、それぞれ段落になる

```
$ git commit -m "add <head> and <title> to index" \
> -m "This allows for a more semantic document."
[master 372e445] add <head> and <title> to index
1 files changed, 3 insertions(+), 0 deletions(-)
```

■ git log

・git log に数値を渡すことで、任意のコミットを表示できる

```
$ git log -1
commit 372e445146791569f4e2b68d04d730c58e6b2ec9
Author: YAGI Hiroto <piroto@ryujyu.typea.info>
Date:   Mon Dec 6 22:48:06 2010 +0900

    add <head> and <title> to index

    This allows for a more semantic document.
```

ブランチ

2 つの便利な使いかた

1. プロジェクトの別バージョンを別ブランチにする
2. 特別な機能を使うための目的別ブランチ

使いかた

■ブランチを作る git branch

```
git branch [ 作りたいブランチの名前 ] [ 分岐元にしたいブランチの名前 ]
```

例

```
 $ git branch RB_1.0 master
*master は Git でのデフォルトブランチ名
```

・RB は Release Branch の略

これで、リリース準備が整ったブランチが切り離せたので、影響をあたえずに変更できる

■ HTML をさらに編集

```
    :
<body>
<h1>Hello World.</h1>
<ul>
  <li><a href="bio.html">Biografy</a></li>
</u>
</body>
    :
```

■変更をコミット

・ -a パラメータを付けて Git が変更があったと認識している全てをコミット

```
git commit -a
```

コメントの入力を促されるので、入力してエディタを終了させる

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Committer: root <root@ryujyu.typea.info>
#
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   index.html
#
```

■ブランチを切り替える git checkout

・リリースバージョンのブランチに切り替えて修正を施してみる

ブランチの切り替え

```
$ git checkout RB_1.0
Switched to branch 'RB_1.0'
```

■リリースバージョンの HTML を修正

・ meta タグを追加

```
<head>
<title>Hello World in Git</title>
<meta name="description" content="hello world in Git" />
</head>
```

■コミット

```
$ git commit -a
```

・ エディタにより、メッセージの保存を行う

リリース

タグ

■タグを打つ

・タグを打つことで、リポジトリの特定の時点に目印を付けて参照しやすくする。

```
git [ タグ名 ] [ タグを打つポイント ]
$ git tag 1.0 RB_1.0
```

■タグの一覧

・リポジトリにおけるタグの一覧

```
$ git tag
1.0
```

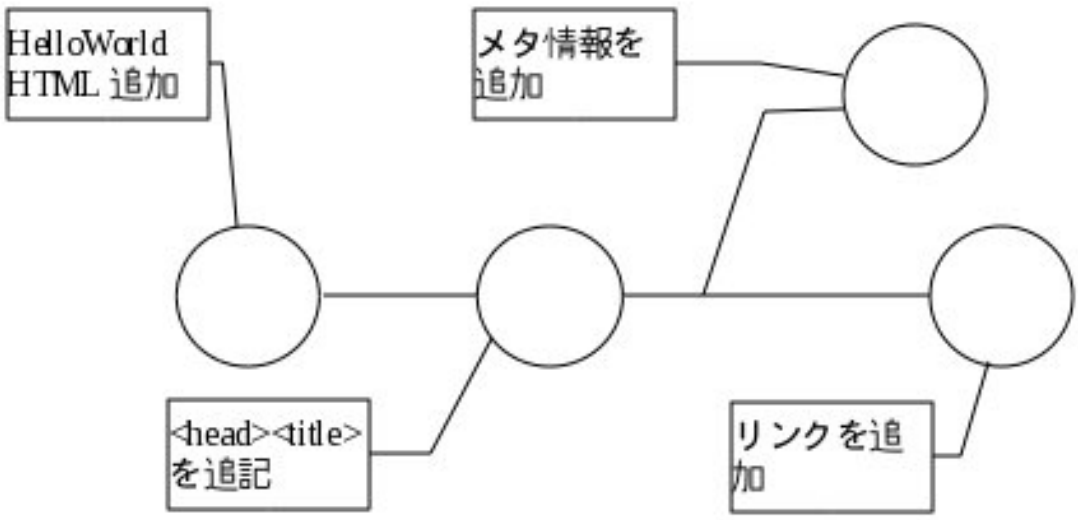
リベース

・以下のためには、git rebase コマンドを利用する

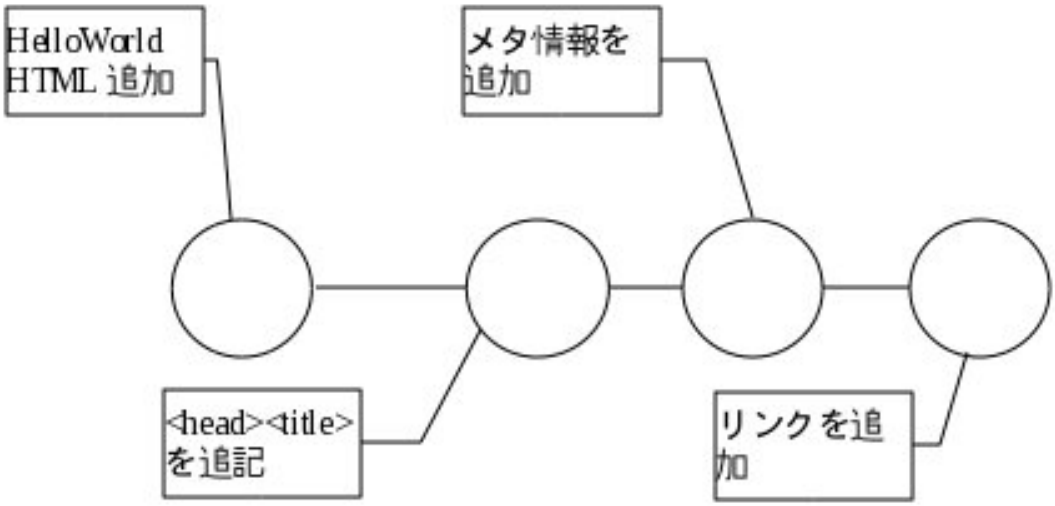
現時点で、ブランチが 2 つあり、それぞれが関知していないコミットを持っている。RB_1.0 (リリース 1.0) に変更がなされたことを、master ブランチ (リリース 2.0 向けの作業) にも知らせる必要がある。

・ブランチから変更を取り出してきて、別のブランチの先頭で再生するのがリベース (rebase)

この状態を



こうする



まずは、ブランチを切り替えて master ブランチに戻る

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

rebase コマンドの実行

```
$ git rebase RB_1.0
Current branch master is up to date.
```

ブランチの削除

・リポジトリは、上記の下図の状態になったので、リリースブランチを削除する
・先ほど作ったタグが、RB_1.0 と同じコミットを指しているため、失うものはない
・git branch -d で削除

```
$ git branch -d RB_1.0
Deleted branch RB_1.0 (was d0068a8).
```

内容を確認

```
$ cat index.html
<html>
<head>
<title>Hello World in Git</title>
<meta name="description" content="hello world in Git" />
</head>
<body>
<h1>Hello World.</h1>
<ul>
  <li><a href="bio.html">Blografy</a></li>
</u>
</body>
</html>
```

タグからブランチを作る

・リリースブランチ (RB_1.0) を削除してしまったが、タグを指定してブランチを作成できる
・タグ 1.0 から、 RB_1.0.1 をブランチ

```
$ git branch RB_1.0.1 1.0
```

アーカイブ

・アーカイブを作成するために、git archive コマンドが用意されている

■ tar ボールを作る

```
$ git archive --format=tar --prefix=mysite-1.0/ 1.0 | gzip > mysite-1.0.tar.gz
$ ls
index.html  mysite-1.0.tar.gz
```

上記のパラメータの説明

パラメータ	内容
--format=tar	tar を出力することを伝えている
--prefix=mysite-1.0/	全体を mysite-1.0/ ディレクトリ以下に置くように指定している
1.0	使用するタグを指定している

■ zip ファイルを作る

```
$ git archive --format=zip --prefix=mysite-1.0/ 1.0 > mysite-1.0.zip
$ ls
index.html  mysite-1.0.zip
```

リモートリポジトリのクローンを作る

■ git clone

・リモートリポジトリで作業を始めるためには、git clone でクローンを作成する
・クローンとは、作業を始めるリポジトリの完全なコピー

■プロトコル

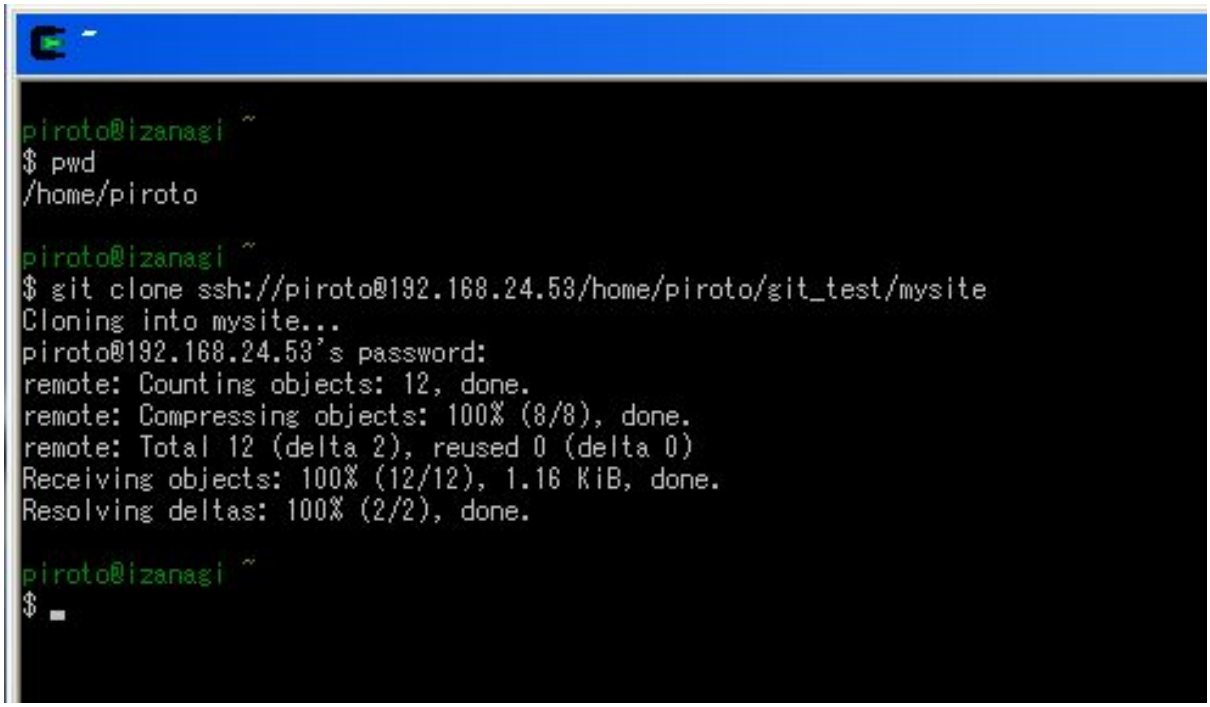
SSH

・ドメイン名と、ユーザー名を付加する以外は、ファイルシステムから直接アクセスする場合と同様

```
ssh://[user]@[server][ リポジトリのパス ]
```

・別クライアントからアクセスし、クローンを作成する例

```
git clone ssh://piroto@192.168.24.53/home/piroto/git_test/mysite
```



git

・スピード重視の独自プロトコル
・9418 ポートを使うのでファイアーウォールの設定が必要
・パスは、リポジトリだと見なすパスを Git サーバー起動時に指定するので、クライアントはリポジトリ名を指定
・多くの git:// は読み取り専用として、書き込みには、ssh:// を使うことが多い

```
git://[server][ リポジトリ ]
```

HTTP/HTTPS

・困ったときの最後のプロトコル
・効率的ではないが、厳格なファイアーウォールでも許可されていることが多い

```
http://[server][ リポジトリのパス ]
```

Subversion とのコマンド対比

subversion	git
svnadmin create	git init
svn checkout	git clone
svn update	git pull
svn add	git add
svn commit	git add、git commit
svn status	git status

svn switch <branch>	git checkout <branch>
svn merge <branch>	git merge <branch>
svn revert <file>	git checkout <file>

リモートでの作業

- http://git-scm.com/book/ja/v1/Git-%E3%81%AE%E5%9F%BA%E6%9C%AC-%E3%83%AA%E3%83%A2%E3%83%BC%E3%83%88%E3%81%A7%E3%81%AE%E4%BD%9C%E6%A5%AD
- http://naokirin.hatenablog.com/entry/20111201/1322576109

git init --bare

- bare オプションをつけた場合、bare リポジトリを作成します。
- bare リポジトリは、作業ファイルがなく、管理ファイルのみのリポジトリになります。
- コミットなどはそのリポジトリでは行われません。(というか行えない。)
- bare リポジトリは、サーバ側のリモートリポジトリとして使うリポジトリを作成する際に使用します。

```
# git init --bare hoge
```

git init --bare --shared=true

- 通常リモートリポジトリはようにします。グループで共有するためには --shared=true を指定しないといけないみたいです。

git remote add

- リモートリポジトリの追加
- 新しいリモート Git リポジトリにアクセスしやすいような名前をつけて追加する

```
git remote add [shortname] [url]
$ git remote
origin
$ git remote add pb git://github.com/paulboone/ticgit.git
$ git remote -v
origin  git://github.com/schacon/ticgit.git
pb  git://github.com/paulboone/ticgit.git
```