



セキュリティ & プログラミングキャンプ 2008 「gitの使い方」

ミラクル・リナックス株式会社
よしおかひろたか

Contents

- はじめに
- git入門
 - 分散バージョン管理システム
 - CVS/Subversionとの比較
- gitユーザのワークフロー
- git管理者のワークフロー
- 様々な話題

はじめに

- git
 - ギットと読む
 - LinusがLinuxのカーネル開発のために開発
 - 浜野さんがメンテしている
 - 背景
 - 2002年BitKeeper(パッチ単位の管理)
CVSはファイル単位の管理なのでカーネル開発に適していない。
 - 2005年4月BitKeeperとの決別

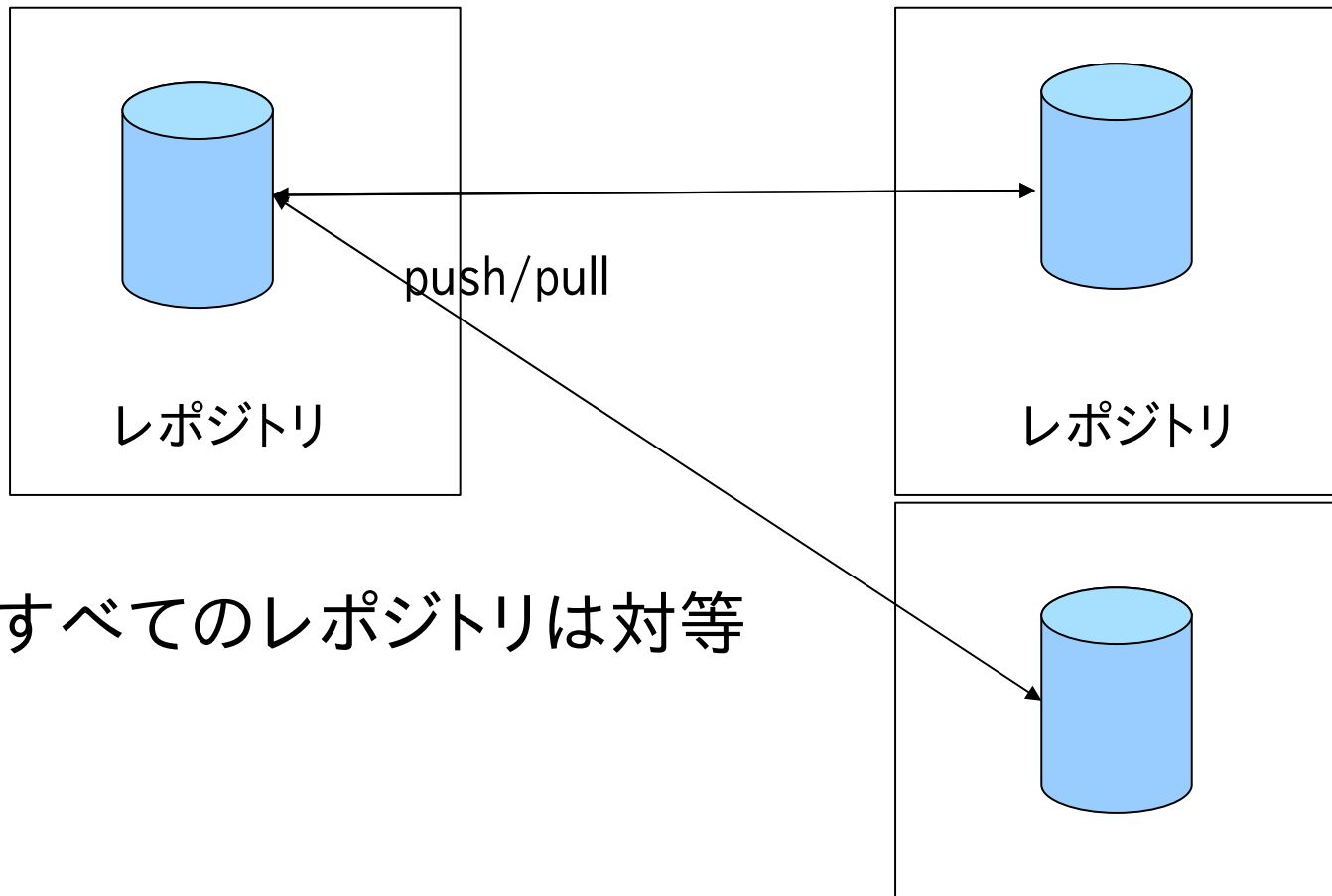
gitの特徴

- ブランチが高速で簡単
- オフライン作業ができ、ローカル・コミットを実行できる
- コミットはファイル単位ではなくアトミックに行える
- 完全なプロジェクト履歴を持つリポジトリが含まれる
- 対等なりポジトリ

git入門

- 分散バージョン管理システム
- CVS/Subversionとの比較

分散バージョン管理システム



分散バージョン管理システム

- リモートにあるリポジトリを元にしてローカルに自分用リポジトリを作成(clone)
- ローカルに自由に変更(commit)
- ローカルの変更をリモートに反映(push)
- リモートの変更をローカルに反映(pull)
- それぞれのリポジトリはどちらが親、子という関係ではなく対等

CVS/Subversionとの比較

- CVS/Subversionは集中バージョン管理システム
 - リポジトリは一ヶ所
 - 書き込み権限は一部のユーザー
 - ワークフローがコミッターと開発者で違う

gitユーザのワークフロー

- リモートリポジトリのコピー(clone)
 - \$ git clone
git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux-2.6.git
- ファイルの編集、テスト、...
- ファイルの追加
 - \$ git add new-file
- コミット(ローカルレポジトリ)
 - \$ git commit -a
コメントの記述など

gitユーザのワークフロー

- リモートレポジトリのマージ
 - \$ git pull
変更点などを取り込む
- リモートレポジトリへのコミット
 - \$ git push

gitのユースケース

- 書き込み権限がない場合。
 - 自分のローカルレポジトリを公開してメンテナ(コミッター)にpullしてもらう。
 - メール等でパッチを送る。(よくあるケース)
- 書き込み権限がある場合。
 - 自分でpushする。
 - 大きな変更でも履歴が残っているので問題を追跡しやすい。

git管理者のワークフロー

- リポジトリの作成
 - \$ git init (リポジトリの初期化)
 - \$ git add (ファイルの追加)
- gitツリーの公開
 - ssh
 - http
 - git
 - rsync

ブランチ

- HEAD:ブランチ上の最新のコミット
- HEAD^:HEADの親
- HEAD^^:HEADの親の親

リグレッションを見つける

- git bisect

---v1 ---v2---...---v1000

\$git bisect start

\$git bisect good v1(OKだったバージョン)

\$git bisect bad v1000(NGのバージョン)

\$git branch

一時的に'bisect'ブランチへ移動。テストして
OKかNGかを確認する

\$ git bisect bad (NGの場合)

\$ git bisect good (OKの場合)

上記を繰り返す。gitが問題を発生させたコミットを指摘してくれる。

高度なブランチ管理

- ---Z---0---X---...---0---A---0---
Y*---...---B*---D*
ZからBの履歴をAの先頭にrebaseした場合、1
行の履歴を得られる。
ZとD*の間をgit-bisectするとY*が問題だと指
摘できる。

gitの内部構造

- オブジェクト
 - blob:ファイルの中身
 - tree:ディレクトリ構造
 - commit:コミット
 - tag:タグ
 - オブジェクトはすべてzlibで圧縮されSHA1ハッシュを持つ。
 - ハッシュ値:40桁16進

gitをgitを利用して理解する

- Read the git source code using by git!
- ソースコードのコピー

```
$ time git clone git://git.kernel.org/pub/scm/git/git.git;date
Initialized empty Git repository in /proj/git/.git/
remote: Counting objects: 72046, done.
remote: Compressing objects: 100% (21922/21922), done.
remote: Total 72046 (delta 51731), reused 68367 (delta 48756)
Receiving objects: 100% (72046/72046), 15.46 MiB | 803 KiB/s, done.
Resolving deltas: 100% (51731/51731), done.
```

```
real 0m59.271s
user 0m30.590s
sys 0m1.224s
```

```
2008年 5月 16日 金曜日 09:00:07 JST
```

gitをgitを利用して理解する

- 最初のコミットにアクセス
\$ git checkout e83c5163

```
$ git checkout e83c5163
```

Note: moving to "e83c5163" which isn't a local branch

If you want to create a new branch from this checkout, you may do so (now or later) by using -b with the checkout command again. Example:

```
git checkout -b <new_branch_name>
```

HEAD is now at e83c516... Initial revision of "git", the information manager from hell

```
hyoshiok@hyoshiok-laptop:~/proj/git$ ls
```

```
Makefile cache.h commit-tree.c read-cache.c show-diff.c write-tree.c
```

```
README cat-file.c init-db.c read-tree.c update-cache.c
```

```
hyoshiok@hyoshiok-laptop:~/proj/git$ wc *
```

```
40  99  957 Makefile
```

```
168 1415 8392 README
```

```
93  350 2484 cache.h
```

```
23  66  503 cat-file.c
```

```
172 576 4103 commit-tree.c
```

```
51  186 1198 init-db.c
```

```
259 820 5681 read-cache.c
```

```
43  133  986 read-tree.c
```

```
81  245 2034 show-diff.c
```

```
248 767 5395 update-cache.c
```

```
66  223 1441 write-tree.c
```

```
1244 4880 33174 合計
```

gitをgitを利用して理解する

- 最初のコミットは1000行程度のプログラムであった。
- READMEを読む
- Makefileを読む
- makeしてみる (libssl-debパッケージをインストールしておく)

様々な話題

- gitk (git GUI)
 - tcl, tk(wish) で実装
- github
 - git向けのホスティングサービス

gitを利用する主なOSSプロジェクト

- Linux
- git
- X
- samba