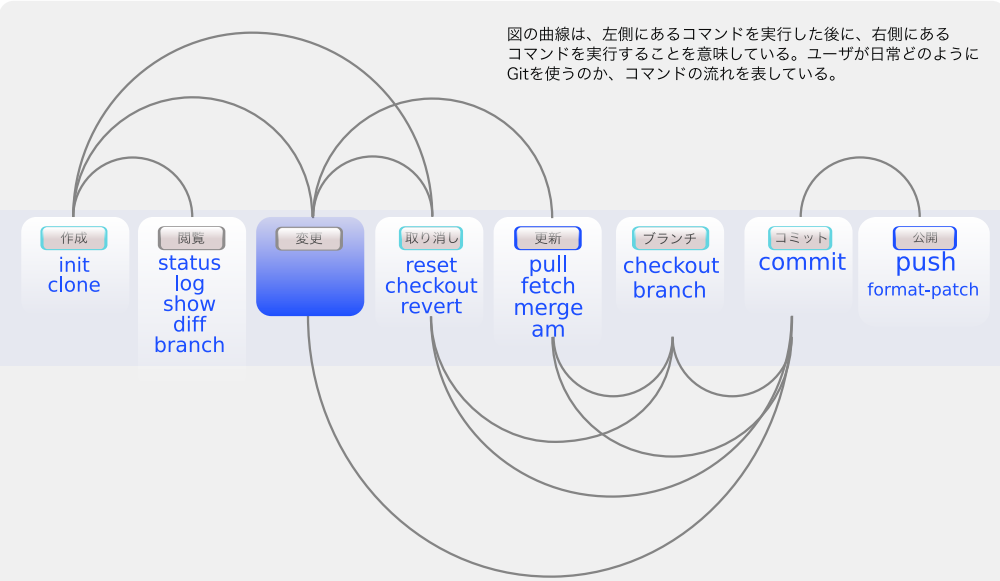
Git Cheat Sheet

http://git.or.cz/

覚えておこう: git command --help

Gitの全体設定は $HOME/.gitconfig に記述する(git config --help)

コマンドの流れ



作成

既存のデータから作成する

```
cd ~/projects/myproject
git init
git add .
```

既存のリポジトリから作成する

```
git clone ~/existing/repo ~/new/repo
git clone git://host.org/project.git
git clone ssh://you@host.org/proj.git
```

表示

作業ディレクトリで変更したファイルを表示する

```
git status
```

追跡対象のファイルに対する変更を表示する

```
git diff
```

$id1と$id2の差分を表示する

```
git diff $id1  $id2
```

変更履歴を表示する

```
git log
```

ファイルの変更履歴を差分付きで表示する

```
git log -p $file $dir/ec/tory/
```

ファイルのどこを誰がいつ変更したか表示する

```
git blame $file
```

$idのコミット内容を表示する

```
git show $id
```

特定$idのファイル$fileを表示

```
git show $id:$file
```

ローカルブランチをすべて表示する

```
git branch
```

("*"は現在のブランチを示す)

Cheat Sheetの表記法

$id : コミットID、ブランチ名、タグ名のいずれか
$file : 任意のファイル名
$branch : 任意のブランチ名

概念

Gitの基礎

master : デフォルトの開発ブランチ
origin : デフォルトの上流リポジトリ
HEAD : 現在のブランチ
HEAD^ : HEADの親
HEAD~4 : HEADの祖父母の祖父母

取り消し

最後にコミットした状態に戻す

```
git reset --hard
```

⚠ ハードリセットは取り消せない

最新のコミットを取り消す

```
git revert HEAD
```

新しいコミットが作られる

指定したコミットを取り消す

```
git revert $id
```

新しいコミットが作られる

最新のコミットを修正する

```
git commit -a --amend
```

(壊れたファイルを編集した後)

ファイル$fileのバージョン$idをチェックアウトする

```
git checkout $id $file
```

ブランチ

ブランチ$idに切り替える

```
git checkout $id
```

ブランチ$branch1を$branch2にマージする

```
git checkout $branch2
git merge $branch1
```

HEADから$branchという名前のブランチを作成する

```
git branch $branch
```

ブランチ$otherからブランチ$new_branchを作成し、$new_branchに切り替える

```
git checkout -b $new_branch $other
```

ブランチ$branchを削除する

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

更新

originから最新の変更を取得する

```
git fetch
```

(マージはしない)

originから最新の変更を取得する

```
git pull
```

(取得した後、マージする)

他人から届いたパッチを当てる

```
git am -3 patch.mbox
```

(競合発生時は、git am --resolved で解消する
git am --resolved)

公開

ローカルの変更をすべてコミットする

```
git commit -a
```

他の開発者向けにパッチを用意する

```
git format-patch origin
```

変更をoriginに反映する

```
git push
```

バージョンやマイルストーンのタグを付ける

```
git tag v1.0
```

役に立つコマンド

リグレッションを見つける

```
git bisect start          (開始)
git bisect good $id       ($idは最後に動作していたバージョン)
git bisect bad $id        ($idは動作しないバージョン)

git bisect bad/good       (badかgoodか、マーク)
git bisect visualize      (gitkを起動して、bisectとマーク)
git bisect reset          (完了したとき)
```

エラーチェックをして、リポジトリをきれいにする

```
git fsck
git gc --prune
```

作業ディレクトリでfoo()を検索する

```
git grep "foo()"
```

マージの競合を解消

マージの競合を確認する

```
git diff          (競合のある差分すべて)
git diff --base  $file (ベースファイルに対して)
git diff --ours  $file (あなたの変更に対して)
git diff --theirs $file (他人の変更に対して)
```

競合しているパッチを捨てる

```
git reset --hard
git rebase --skip
```

競合を解消してマージする

```
git add $conflicting_file (競合を解消した全ファイルに)
git rebase --continue
```


Zack Rusin
Based on the work of:
Sébastien Pierre
Xprima Corp.

日本語版
Takashi Sasai
(http://www.textdrop.net/)