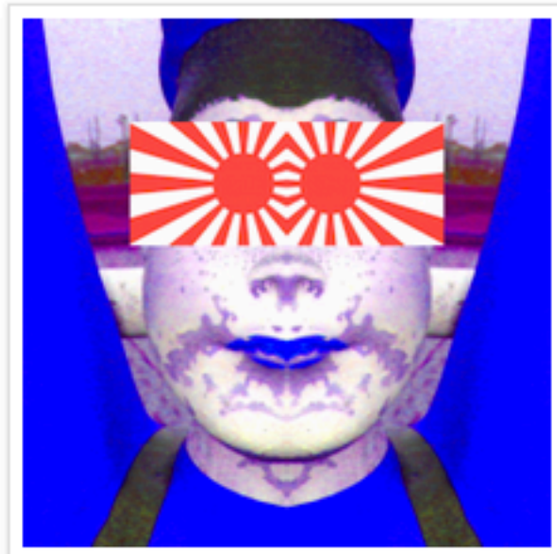
Git ライフを
{ちょっと,もっと,超}
快適にする
知られざるコマンドたち

@uasi

uasi <3


Color palette
Writing Tools
PROJECT: APPLICATION.AP
ARCHITECT: XCODE

&



Tomoki Aonuma
uasi

<> Objective-C, Perl, Ruby
Tsukuba University
Tokyo, Japan
uasi@uasi.jp
http://uasi.hatenablog.com/
Joined on Aug 28, 2008

58 followers **124** starred **7** following

- GitHub 5年生
- ずっと使ってる
- 一人で使ってる

ずっと一人で使ってる

```
$ git blame Document.h
ca55733e (Tomoki Aonuma ...
ad2d179e (Tomoki Aonuma ...
95074544 (Tomoki Aonuma ...
c7bc39f9 (Tomoki Aonuma ...
95074544 (Tomoki Aonuma ...
ca55733e (Tomoki Aonuma ...
^02e6c80 (Tomoki Aonuma ...
a3a5d708 (Tomoki Aonuma ...
fc79def1 (Tomoki Aonuma ...
```

チームのことは

分からない…

一人で便利に
使う方法
教えます

知られざる度 ☆☆☆

役立ち度 ★★★

```
git help
```

git help <cmd>

- 困ったらHELP
- よく使うコマンドほどよく変わる
- git help -a ←コマンド一覧
- git help -g ←ガイド一覧

git help <cmd>

- git help config
- git help glossary
  - 用語集
- git help revisions
  - HEAD^とかdev..masterとか

知られざる度 ★☆☆

役立ち度 ★★☆

`git status -sb`

```
git status -sb
```

```
git status \
  --short  \
  --branch
```

git status

```
$ git status
# On branch master
# Your branch is ahead of 'origin/mast
#   (use "git push" to publish your lo
#
# Untracked files:
#   (use "git add <file>..." to includ
#
#       hoge
nothing added to commit but untracked
files present (use "git add" to track)
```

git status -sb

```
$ git status -sb
## master...origin/master [ahead 1]
?? hoge
```

知られざる度 ★☆☆

役立ち度 ★★☆

```
ls .git/hooks
```

```
git help hooks
```

`ls .git/hooks`
`git help hooks`

- 色々なタイミングで走るフック
- コミット前後
- リベース前
- チェックアウト後...

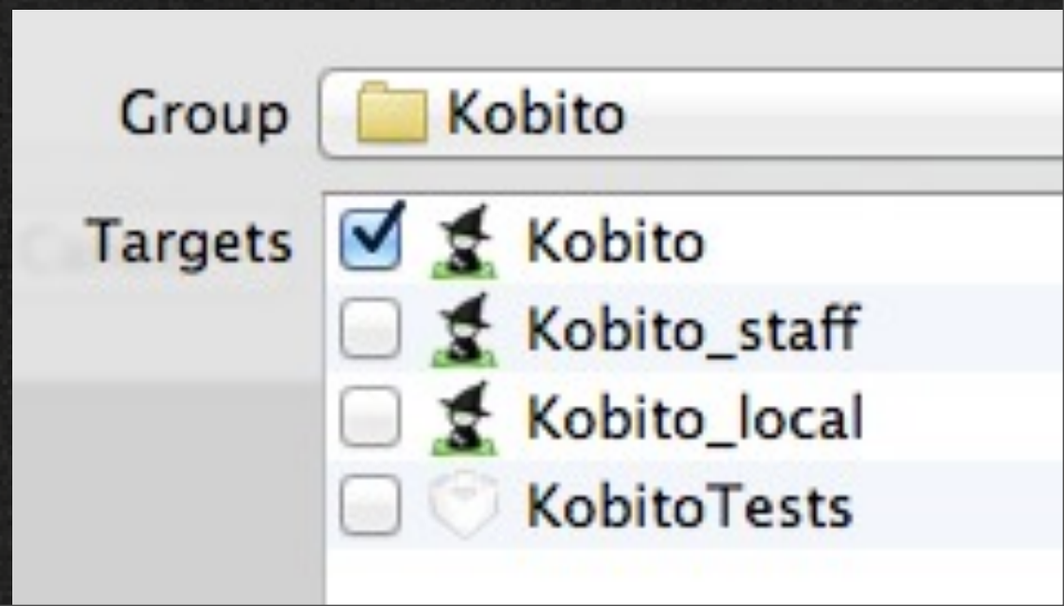
ls .git/hooks
git help hooks

- Kobito の開発では
  pre-commit フックで
  ソースファイルがターゲットに
  含まれているかチェック

- gem ‘xcodeproj’

ls .git/hooks
git help hooks

- Kobito の開発では
  pre-commit フックで
  ソースファイルがターゲットに
  含まれているかチェック
- gem ‘xcodeproj’



知られざる度 ★★☆

役立ち度 ★★☆

```
git diff \
--patience
```

git diff --patience

- （ちょっと）賢い diff
- 差分が読みやすくなる（かも）

例

編集前

```
void foo() {
    printf("foo\n");
}
```

編集後

```
void foo() {
    printf("foo\n");
    printf("foo\n");
}

void bar() {
    printf("bar\n");
}
```

git diff

```
$ git diff a.c b.c
...
@@ -1,4 +1,9 @@
 void foo() {
     printf("foo\n");
+    printf("foo\n");
+
+}

+void bar() {
+    printf("bar\n");
 }
```

git diff --patience

```
$ git diff --patience a.c b.c
...
@@ -1,4 +1,9 @@
 void foo() {
     printf("foo\n");
+    printf("foo\n");

 }
+
+void bar() {
+    printf("bar\n");
+}
```

git diff --histogram

- （もうちょっと）賢い diff
- 差分が読みやすくなる（よね？）

git config \
diff.algorithm

- git config \
  diff.algorithm \
  (patience|histogram)
- ふつうの diff を使うときは

```
git diff \
  --diff-algorithm=default
```

git config \ diff.algorithm

- git log -U や git format-patch にも影響
- 実はそれらのコマンドにも --patience などを指定できる

知られざる度 ★★★

役立ち度 ★★☆

git diff
git apply
(outside worktree)

git diff

- Git リポジトリの外でも使える
- ふつうの diff より便利
  - git diff --binary
  - git diff --color
  - git diff --word-diff

git diff

- Git リポジトリの中で
  任意の2ファイルの差分を取れる
- git diff --no-index

git apply

- Git リポジトリの外でも使える
- git diff のパッチを
  当てるときに

- git diff old new >patch
  git apply patch
  # old が 消える！

  #(∴old+変更+改名==new)

知られざる度 ★★★

役立ち度 ★☆☆

pushInstead0f

pushInstead0f なし

```
# HTTPS URL でクローン
$ git clone https://github.com/foo/bar

# ...
# 書き込み権限をもらった！

$ git push origin master
Username for 'https://github.com':
# 面倒！
```

pushInstead0f

```
$ cat ~/.gitconfig
...
[url "git@github.com:"]
  pushinsteadof = "git://github.com/"
  pushinsteadof = "https://github.com/"
```

pushInstead0f あり

```
# HTTPS URL でクローン
$ git clone https://github.com/foo/bar

# ...
# 書き込み権限をもらった！


$ git push origin master
Total 0 (delta 0), reused 0 (delta 0)
To git@github.com:foo/bar
...
```

知られざる度 ★★★

役立ち度 ★★☆

```
git new-workdir
```

git new-workdir

- リポジトリの作業ディレクトリを新たに作る

- `git new-workdir \`
  `<repository> \`
  `<new_workdir_path> \`
  `[<branch>]`

git new-workdir

- contrib ディレクトリにある
- /usr/local/share/git-core/contrib/workdir あたり

知られざる度 ★★★

役立ち度 ★★★

```
git rebase -i \
--autosquash
```

--autosquash

- コミットメッセージが `fixup!` か `squash!` で始まるコミットを自動的に `fixup/squash`
- `fixup!` の後に対象のコミットのメッセージを指定（数文字でOK）

--autosquash

```
$ git log oneline
a99e5d7 fixup! X
d2cb153 fixup! A
c85fc65 Add Foo
```

```
$ git rebase -i --autosquash c85fc65^
pick c85fc65 Add Foo
fixup d2cb153 fixup! A
pick a99e5d7 fixup! X
```

rebase.autosquash

```
$ git config [--global] \
  rebase.autosquash true
```

いつでも autosquash

知らないコマンドは

ありましたか？

Git にはまだまだ
コマンドやオプションが
追加されています
(リリースノートをチェック！)

面白いコマンドを
見つけたら
Qiita でシェア！
(宣伝です)

Enjoy!