05/10/2019

RAPPORT SUR GIT

LO43

TAO ZHEHAN

CATALOGUE

1.GIT CONFIG-Obtention et définition d'options de référentiel ou globales

1.1SYNOPSIS

il y a un tableau pour exprimer des options :

```
16431@LAPTOP-69G7B2LE MINGW64 ~
$ git config
usage: git config [<options>]

Config file location
    --global              use global config file
    --system              use system config file
    --local               use repository config file
    --worktree            use per-worktree config file
    -f, --file <file>     use given config file
    --blob <blob-id>      read config from given blob object

Action
    --get                 get value: name [value-regex]
    --get-all             get all values: key [value-regex]
    --get-regexp          get values for regexp: name-regex [value-regex]
    --get-urlmatch        get value specific for the URL: section[.var] URL
    --replace-all         replace all matching variables: name value [value_rege
x]
    --add                 add a new variable: name value
    --unset               remove a variable: name [value-regex]
    --unset-all           remove all matches: name [value-regex]
    --rename-section      rename section: old-name new-name
    --remove-section      remove a section: name
    -l, --list            list all
    -e, --edit            open an editor
    --get-color           find the color configured: slot [default]
    --get-colorbool       find the color setting: slot [stdout-is-tty]

Type
    -t, --type <>         value is given this type
    --bool                value is "true" or "false"
    --int                 value is decimal number
    --bool-or-int         value is --bool or --int
    --path                value is a path (file or directory name)
    --expiry-date         value is an expiry date

Other
    -z, --null            terminate values with NUL byte
    --name-only           show variable names only
    --includes            respect include directives on lookup
    --show-origin         show origin of config (file, standard input, blob, com
mand line)
    --default <value>     with --get, use default value when missing entry
```

Git contient un outil appelé git config pour vous permettre de voir et modifier les variables de configuration qui contrôlent tous les aspects de l'apparence et du comportement de Git. Ces variables peuvent être stockées dans trois endroits différents :

- Fichier /etc/gitconfig : Contient les valeurs pour tous les utilisateurs et tous les dépôts du système. Si vous passez l'option --system à git config, il lit et écrit ce fichier spécifiquement.
- Fichier ~/.gitconfig : Spécifique à votre utilisateur. Vous pouvez forcer Git à lire et écrire ce fichier en passant l'option --global.
- Fichier config dans le répertoire Git (c'est-à-dire .git/config) du dépôt en cours d'utilisation : spécifique au seul dépôt en cours.

1.2EXEMPLE CONCERT

Configurer les informations de l'utilisateur pour tous les dépôts locaux

git config --global user.name "[nom]"

Définit le nom que vous voulez associer à toutes vos opérations de commit

$ git config --global user.email "[adresse email]"

Définit l'email que vous voulez associer à toutes vos opérations de commit

$ git config --global color.ui auto

Active la colorisation de la sortie en ligne de commande

```
16431@LAPTOP-69G7B2LE MINGW64 ~
$ git config --global user.name "TAO ZHEHAN"

16431@LAPTOP-69G7B2LE MINGW64 ~
$ git config --global user.email roantzh@gmail.com

16431@LAPTOP-69G7B2LE MINGW64 ~
$ git config --list
core.symlinks=false
core.autocrlf=true
core.fscache=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
help.format=html
rebase.autosquash=true
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
http.sslbackend=openssl
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge --skip -- %f
filter.lfs.process=git-lfs filter-process --skip
filter.lfs.required=true
credential.helper=manager
user.name=TAO ZHEHAN
user.email=roantzh@gmail.com
```

2.GIT INIT-Crée un dépôt Git vide ou réinitialise un dépôt existant

2.1SYNOPSIS

git init [-q | --quiet] [--bare] [--template=<template_directory>]
[--separate-git-dir <git dir>]
[--shared[=<permissions>]] [directory]

2.2DESCRIPTION

Cette commande crée un dépôt Git vide - essentiellement un répertoire .git avec des sous-répertoires pour les fichiers objects, refs/heads, refs/tags et les fichiers de modèle. Un fichier HEAD initial qui fait référence à la tête de la branche principale (master) est également créé.

Si la variable d'environnement $GIT_DIR est définie, alors elle spécifie un chemin à utiliser au lieu de ./.git comme base du dépôt.

2.3EXEMPLE CONCERT

```
16431@LAPTOP-69G7B2LE MINGW64 ~ (master)
$ git init mabranch
Initialized empty Git repository in C:/Users/16431/mabranch/.git/
```

3.GIT STATUS-Montrer le statut de l'arbre de travail

3.1SYNOPSIS

git status [<options>…] [--] [<spécificateur de chemin>…]

3.2DESCRIPTION

Montre les chemins qui ont des différences entre le fichier d'index et le commit HEAD actuel, les chemins qui ont des différences entre l'arbre de travail et de fichier d'index, et les chemins dans l'arbre de travail qui ne sont pas suivis par Git.

3.3EXEMPLE CONCERT

-Vérifier l'état des fichiers

L'outil principal pour déterminer quels fichiers sont dans quel état est la commande git status.

$ git status

Liste tous les nouveaux fichiers et les fichiers modifiés à commiter

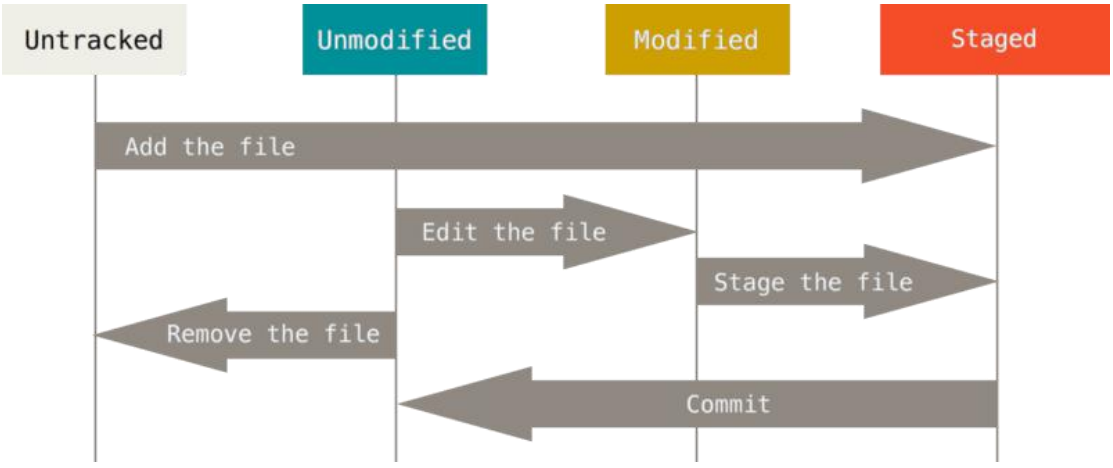
```
16431@LAPTOP-69G7B2LE MINGW64 ~/fichier/test (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   readme.txt
```

Statut court

```
16431@LAPTOP-69G7B2LE MINGW64 ~/fichier/test (master)
$ git status -s
A  readme.txt
```



4.GIT MERGE-Combine dans la branche courante l'historique de la branche spécifiée

4.1SYNOPSIS

git merge [-n] [--stat] [--no-commit] [--squash] [--[no-]edit]
[-s <strategy>] [-X <strategy-option>] [-S[<keyid>]]
[--[no-]allow-unrelated-histories]
[--[no-]rerere-autoupdate] [-m <msg>] [-F <file>]
[<commit>…]git merge (--continue | --abort | --quit)

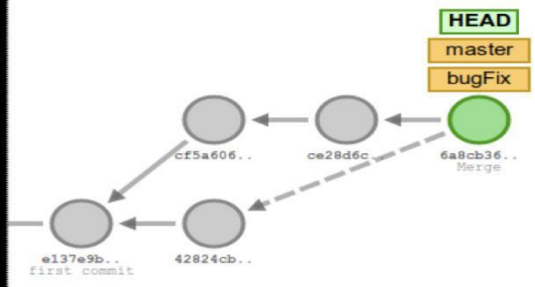
4.2DESCRIPTION

Ce command permet de combiner ensemble les contenus de deux branches différentes.

4.3EXEMPLE CONCERT

```
16431@LAPTOP-69G7B2LE MINGW64 /c/git/LO43-A2019 (taozhehan)
$ git merge new1
Updating 7d37bde..88a71f2
Fast-forward
 test/ntest.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 test/ntest.txt
```

```
$ git checkout master
$ git merge bugFix
You have performed a fast-forward
merge.
```



5.GIT DIFF- Affiche les modifications entre les commits, un commit et l'arbre de travail, etc

5.1SYNOPSIS

git diff [<options>] [<commit>] [--] [<path>...] git diff [<options>] --cached [<commit>] [--] [<path>...] git diff [<options>] <commit> <commit> [--] [<path>...] git diff [<options>] <blob> <blob> git diff [<options>] --no-index [--] <path> <path>

5.2DESCRIPTION

Affiche les modifications entre l'arbre de travail et l'index ou un arbre, les modifications entre l'index et un arbre, les modifications entre deux arbres, les modifications entre deux objets blobs ou les modifications entre deux fichiers sur disque.

5.3EXEMPLE CONCERT

6.GIT BLAME

6.1SYNOPSIS

git blame [-c] [-b] [-l] [--root] [-t] [-f] [-n] [-s] [-e] [-p] [-w] [--incremental] [-L <range>] [-S <revs-file>] [-M] [-C] [-C] [-C] [--since=<date>] [--ignore-rev <rev>] [--ignore-revs-file <file>] [--progress] [--abbrev=<n>] [<rev> | --contents <file> | --reverse <rev>..<rev>] [--] <file>

6.2DESCRIPTION

Montrer quelle révision et quel auteur a modifié en dernier chaque ligne d'un fichier

6.3EXEMPLE CONCERT

```
16431@LAPTOP-69G7B2LE MINGW64 /c/git/LO43-A2019 (taozhehan)
$ git blame
usage: git blame [<options>] [<rev-opts>] [<rev>] [--] <file>

    <rev-opts> are documented in git-rev-list(1)

    --incremental         Show blame entries as we find them, incrementally
    -b                    Show blank SHA-1 for boundary commits (Default: off)
    --root                Do not treat root commits as boundaries (Default: off)
    --show-stats          Show work cost statistics
    --progress            Force progress reporting
    --score-debug         Show output score for blame entries
    -f, --show-name       Show original filename (Default: auto)
    -n, --show-number     Show original linenumber (Default: off)
    -p, --porcelain       Show in a format designed for machine consumption
    --line-porcelain      Show porcelain format with per-line commit information
    -c                    Use the same output mode as git-annotate (Default: off
)
    -t                    Show raw timestamp (Default: off)
    -l                    Show long commit SHA1 (Default: off)
    -s                    Suppress author name and timestamp (Default: off)
    -e, --show-email      Show author email instead of name (Default: off)
    -w                    Ignore whitespace differences
    --ignore-rev <rev>    Ignore <rev> when blaming
    --ignore-revs-file <file>
                          Ignore revisions from <file>
    --color-lines         color redundant metadata from previous line differentl
y
    --color-by-age        color lines by age
    --indent-heuristic    Use an experimental heuristic to improve diffs
    --minimal             Spend extra cycles to find better match
    -S <file>             Use revisions from <file> instead of calling git-rev-l
ist
    --contents <file>     Use <file>'s contents as the final image
    -C[<score>]           Find line copies within and across files
    -M[<score>]           Find line movements within and across files
    -L <n,m>              Process only line range n,m, counting from 1
    --abbrev[=<n>]        use <n> digits to display SHA-1s
```

```
16431@LAPTOP-69G7B2LE MINGW64 /c/git/LO43-A2019/test (taozhehan)
$ git blame ntest.txt
88a71f2d (taozhehan 2019-10-06 14:59:40 +0200 1) creating a new branch
```