PETIT RAPPORT **GIT**

LO43 Group TP1A ZHU Yuxin

Résumé

Testez et indiquez le rôle de les commandes ci-dessous en apprenant *"Learn Git Branching"* et en utilisant l'outil *"Visualizing Git"* et *le terminal OS*.

Git config
Git init
Git status
Git merge
Git diff
Git blame

Git config

Git config permet d'obtenir et de définir des variables de configuration, lesquelles contrôlent les différents aspects de l'apparence et du fonctionnement de Git. Vous pouvez *configurer l'emplacement de stockage des fichiers, configurer votre nom d'utilisateur et votre mot de passe, configurer l'éditeur, configurer l'outil de comparaison et vérifier la configuration*, etc. à l'aide des commandes suivantes.

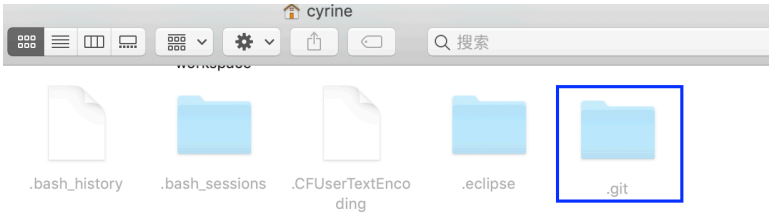
```
zhuyuxindeMacBook-Pro-2:~ cyrine$ git config
usage: git config [<options>]

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_regex]
    --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, command lin
e)
    --default <value>     with --get, use default value when missing entry
```

(La liste des options de git config)

Git init

Git init peut créer un nouveau dépôt vide ou inclure des projets existants dans la gestion des versions.

```
[zhuyuxindeMacBook-Pro-2:~ cyrine$ git init
 Initialized empty Git repository in /Users/cyrine/.git/
```



(*Créer automatiquement un répertoire .git (la valeur par défaut est masquée)*)

Git status

Git status peut afficher le statut des fichiers d'espace de travail et de bloc-notes actuels. L'état du fichier est décrit comme suit: **(1) État non suivi** (Untracked), qui se trouve dans la zone de travail mais n'est pas inclus dans le fichier de gestion Git, ne participe pas au contrôle de version et peut être utilisé pour gérer les fichiers non suivis à l'aide de la commande *git add* **(2) Situé dans la zone intermédiaire à soumettre** (Staged) **(3) Statut modifié**: les fichiers inclus dans le suivi seront à l'état modifié après avoir été modifiés dans l'espace de travail.

```
[zhuyuxindeMacBook-Pro-2:~ cyrine$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .CFUserTextEncoding
        .DS_Store
        .HelloWorld.java.swp
        .Helloworld.java.swo
```

(*Afficher le statut des fichiers d'espace de travail et de bloc-notes actuels*)

La commande **git add** place le fichier non suivi dans la trace et le valide dans la zone de stockage intermédiaire.

Git merge

Git merge arrange une fusion entre deux branches (la branche courante et une autre branche spécifiée en paramètre.)

```
➜ LO43-A2019 git:(zhuyuxin0627) git merge new1
Updating b7049a2..d8db1f3
Fast-forward
 test1.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 test1.txt
```

(*Ci-dessus, le résultat de la commande git merge*)

Git diff

Git diff permet de montrer la différence entre un cache écrit et un changement de cache modifié mais pas encore écrit (voir les résultats de l'exécution de *Git status*).

Modifications non mises en cache: **git diff**

Afficher les modifications mises en cache: **git diff –cached**

Afficher toutes les modifications mises en cache et non mises en cache: **git diff HEAD**

Afficher le résumé au lieu de entier diff: **git diff –stat**

```
➜ LO43-A2019 git:(zhuyuxin0627) ✗ git add petit_rapport_GIT.pages
➜ LO43-A2019 git:(zhuyuxin0627) ✗ git commit -m "GIT rapport"
[zhuyuxin0627 2edeecf] GIT rapport
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100755 petit rapport GIT.pages
~
```

```
➜ LO43-A2019 git:(zhuyuxin0627) ✗ git diff HEAD
```

```
diff --git a/petit_rapport_GIT.pages b/petit_rapport_GIT.pages
index c8f7e8f..8d2bb29 100755
Binary files a/petit_rapport_GIT.pages and b/petit_rapport_GIT.pages differ
~
```

(*Ci-dessus, le résultat de la commande git diff HEAD*)

Git blame

Git blame permet d'afficher la validation la plus récente dans le fichier qui a modifié chaque ligne. Si vous trouvez une méthode défectueuse dans votre code, vous pouvez utiliser Git Blame pour annoter le fichier afin de voir qui modifie chaque ligne de la méthode.

```
zhuyuxindeMacBook-Pro-2:~ cyrine$ 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
    --color-lines         color redundant metadata from previous line differently
    --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-list
    --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
```

(La liste des options de git blame)

```
➜ LO43-A2019 git:(zhuyuxin0627) ✗ git blame HelloWorld.cpp
```

```
b7049a20 (Yuxin Zhu 2019-09-25 15:38:22 +0200 1) #include <iostream>
b7049a20 (Yuxin Zhu 2019-09-25 15:38:22 +0200 2)
b7049a20 (Yuxin Zhu 2019-09-25 15:38:22 +0200 3) using namespace std;
b7049a20 (Yuxin Zhu 2019-09-25 15:38:22 +0200 4)
b7049a20 (Yuxin Zhu 2019-09-25 15:38:22 +0200 5) int main(){
b7049a20 (Yuxin Zhu 2019-09-25 15:38:22 +0200 6)        cout<<"Hello WorldW - hello world!";
b7049a20 (Yuxin Zhu 2019-09-25 15:38:22 +0200 7)        return 0;
b7049a20 (Yuxin Zhu 2019-09-25 15:38:22 +0200 8) }
(END)
```

(Ci-dessus, le résultat de la commande git blame)

La figure suivante utilise **l'option -L <n, m>**pour limiter la plage de sortie des lignes n à m, ou **-e** pour afficher les adresses email à la place des pseudos.