Introduction à Git :

Un logiciel de gestion de Versions Décentralisé (DVCS)

Isabelle BLASQUEZ
@iblasquez

2022

Qui sait ce qu’est un gestionnaire de version ?

Qui utilise un gestionnaire de version ?

Qui utilise git ?

Qui a un compte github/gitlab/bitbucket et l’utilise ?

Isabelle BLASQUEZ

@iblasquez

Enseignement : Génie Logiciel

Recherche : Développement logiciel agile

ICSTUG #IUTAgile

CodeWeek.EU

#Software Craftsmanship

MUSEOMIX LIMOUSIN

Pourquoi un gestionnaire de version ? … et pourquoi décentralisé ?

Quelle est la dernière version ?
Quel est l'ordre des versions ?

Extrait : http://liris.cnrs.fr/~pchampin/enseignement/intro-git/

Gestionnaires de Versions à la rescousse !

Un logiciel de gestion de versions *(ou* ***VCS*** *:* ***V****ersion* ***C****ontrol* ***S****ystem) est un logiciel qui permet de stocker un ensemble de fichiers en conservant la chronologie de toutes les modifications qui ont été effectuées dessus. Il permet notamment de retrouver les différentes versions d'un lot de fichiers connexes* ***(extrait Wikipedia)***

Bon exemple

Avantages de la gestion de versions

Faciliter la <u>sauvegarde</u> et le <u>travail collaboratif</u> en :

→ disposant d'un **historique** sur les changements apportés

→ permettant facilement le **retour en arrière**

→ **Partageant** les changements

(documenter (messages de commit), fusionner, récupérer, visualiser les modifications apportées,…)

Note: Le répertoire `.git` est un répertoire caché, qui contient tout l'historique des fichiers.

Extrait : http://liris.cnrs.fr/~pchampin/enseignement/intro-git/

Historique des Gestionnaires de Version

✓ ***Gestion de versions centralisé*** *(**un seul dépôt** **fait référence**)*

*(1986 : le premier : **C**oncurrent **V**ersioning **S**ystem)*

*(2000 : **SVN** lancé pour remplacer CVS,*
a suscité un réel intérêt pour les VCS,
le plus populaire avant l'arrivée des DVCS…)

✓ ***Gestion de versions décentralisé*** *(**D**istributed **V**ersion **C**ontrôl **S**ystem)*

(2005 : développé et optimisé pour le noyau Linux par Linus Thorval
*⇨ **le plus populaire** …)*

(2005)

(2008)

Plusieurs dépôts pour un même projet
⇒ Travail en local possible **sans accès réseau!**

Pour un historique : https://www.smashingmagazine.com/2008/09/the-top-7-open-source-version-control-systems
A lire : http://blogs.atlassian.fr/2014/06/17/git-vs-mercurial-pourquoi-git/

Gestion Centralisée vs Gestion Décentralisée (pour la collaboration : copie de travail vs dépôt)

Joel Spolsky (Stack Overflow, Trello) décrit ***la gestion de version décentralisée comme « probablement la plus grande avancée dans les technologies de développement logiciel dans ces années »***

A lire : https://www.joelonsoftware.com/2010/03/17/distributed-version-control-is-here-to-stay-baby

Git ne fait aucune distinction
entre la copie de travail
et le dépôt centralisé.
Ce sont des **dépôts Git à part entière.**

Extrait: https://fr.atlassian.com/git/tutorials/setting-up-a-repository

Git : le plus populaire des gestionnaires de version

REBELLABS by ZEROTURNAROUND

All rights reserved. 2016 © ZeroTurnaround Inc.

Extrait : http://pages.zeroturnaround.com/RebelLabs-Developer-Productivity-Report-2016.html

Isabelle BLASQUEZ

Pourquoi préférer un DVCS ...

Avantages de la gestion décentralisée :

→ **Ne pas être dépendant d'une seule machine** comme point de défaillance

→ Travailler **sans être connecté** au gestionnaire de version

→ **Pas de permissions particulières** pour participer à un projet

(les droits de commit/soumission peuvent être donnés après avoir démontré son travail et non pas avant)

→ **Opérations plus rapides** pour la plupart car réalisées en local (sans accès réseau)

→ **Travail privé** (réalisation de brouillons sans devoir publier ses modifications et gêner les autres contributeurs)

→ ... Avec **un dépôt de référence contenant les versions livrées d'un projet**.

Inconvénients :

→ **cloner un dépôt est plus long** que récupérer une version car tout l'historique est copié

(ce qui est toutefois un avantage par la suite)

→ il n'y a **pas de système de lock**

(ce qui peut poser des problèmes pour des données binaires qui ne se fusionnent pas).

Extrait : https://fr.wikipedia.org/wiki/Gestion_de_versions

Git : les premiers pas …

Isabelle BLASQUEZ

Installation

✓ ***Installation:***
https://git-scm.com/downloads

✓ ***Utilisation de git :***

→ ***dans un terminal en ligne de commande***
(`git --help` : la commande à retenir !)

```
$ git --help
usage: git [--version] [--exec-path[=<path>]] [--html-path] [--man-path] [--info
-path]
           [-p|--paginate|--no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           [-c name=value] [--help]
           <command> [<args>]

The most commonly used git commands are:
   add        Add file contents to the index
   bisect     Find by binary search the change that introduced a bug
   branch     List, create, or delete branches
   checkout   Checkout a branch or paths to the working tree
   clone      Clone a repository into a new directory
   commit     Record changes to the repository
   diff       Show changes between commits, commit and working tree, etc
   fetch      Download objects and refs from another repository
   grep       Print lines matching a pattern
   init       Create an empty git repository or reinitialize an existing one
   log        Show commit logs
   merge      Join two or more development histories together
   mv         Move or rename a file, a directory, or a symlink
   pull       Fetch from and merge with another repository or a local branch
   push       Update remote refs along with associated objects
   rebase     Forward-port local commits to the updated upstream head
   reset      Reset current HEAD to the specified state
   rm         Remove files from the working tree and from the index
   show       Show various types of objects
   status     Show the working tree status
```

→ ***via un outil graphique***

… https://www.gitkraken.com/ *et bien d'autres sur* https://git-scm.com/downloads/guis *et* https://git.wiki.kernel.org/index.php/InterfacesFrontendsAndTools#Graphical_Interfaces

Les commandes Git

Commandes de porcelaine

*(porcelain commands **haut niveau**)*

commandes utilisées au quotidien

Commandes de plomberie

*(plumbing commands **bas niveau**)*

commandes qui manipulent les informations au cœur de Git

Nuages de mots réalisés avec à partir de http://www.wordclouds.com/
avec des données issues de http://stackoverflow.com/questions/39847781/which-are-the-plumbing-and-porcelain-commands

Isabelle BLASQUEZ

Git Command Explorer : https://gitexplorer.com

An alternative manual for #git, based on tasks and usage rather than the git syntax

Git Command Explorer

Find the right commands you need without digging through the web.

I want to:

clone

existing repo into a new directory

Usage

```
git clone <repo-url> <directory>
```

Note

```
The repo is cloned into the specified directory

Replace "directory" with the directory you want
```

Git Command Explorer

Find the right commands you need without digging through the web.

I want to:

delete/remove

files from old commits

Usage

```
git filter-branch --index-filter
'git rm --cached --ignore-unmatch path/to/mylarge_file'
--tag-name-filter cat -- --all

filter-branch keeps backups too, so the size of the repo won't
decrease immediately unless you expire the reflogs and garbage
collect:

rm -Rf .git/refs/original       # careful
git gc --aggressive --prune=now # danger
```

Note

```
Like the rebasing option described before, filter-branch is rewriting
operation. If you have published history, you'll have to --force push
the new refs.
```

Isabelle BLASQUEZ

Configuration (à la première utilisation)

```
git config --global user.name "Prénom Nom"
git config --global user.email email@domaine.extension
```

Pas de modification anonyme avec Git !!!

Le dépôt au cœur de Git (ou repository)

Créer un nouveau dépôt :

```
git init
```
(dans le répertoire courant)

```
git init <directory>
```

Clôner un dépôt existant :

```
git clone <repo>
```
(dans le répertoire courant)

```
git clone <repo> <directory>
```

Dépôt : *Un dépôt est un dossier qui porte le nom* ***.git****.*

C'est là que Git va stocker l'historique du projet c-a-d toutes les informations en rapport avec votre projet comme la liste des commits effectués, la liste des branches, etc…

(En savoir plus sur le contenu réeel de ce dépôt :
https://git-scm.com/book/fr/v2/Les-tripes-de-Git-Plomberie-et-porcelaine#_git_internals)

Enregistrer des changements

(`git add`, `git commit`, `.gitignore`)

Un répertoire avec son fichier …
… prêt à être versionné

```
git init
```

Ajout d'un fichier dans l'Index (zone en attente de commit)

```
git add test.txt
```

espace de travail
(working directory)

test.txt

Bonjour tout le monde !

add

Index
(staging index)

test.txt 306c18dea…

zone de staging :
pour regrouper les fichiers
du prochain commit
(=> plusieurs fichiers possibles dans un même commit)

status

Liste, entre autres, les fichiers stagés (dans l'index en attente d'être commité), non stagés

```
$ git status
On branch master

Initial commit

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

        new file:   test.txt
```

ls-files

Liste les fichiers dans l'index et leur hash associé (signature en SHA-1) associé

```
$ git ls-files --stage
100644 306c18deac1991cddc759c58a9098f3cb2eb602f 0       test.txt
```

Commiter (le contenu de l'Index) dans le dépôt local

committer :
enregistrer un ensemble de modifications

`git commit` **+ Message obligatoire qui va être demandé**

`git commit -m "message"`

Le commit vide l'index du fichier stagé

Un hash (SHA-1) propre au commit

Index
(staging index)

Dépôt local
(local repository)

test.txt
+ autres infos

c1eb479

commit

```
$ git commit -m "Premier Commit"
[master (root-commit) c1eb479] Premier Commit
 1 file changed, 1 insertion(+)
 create mode 100644 test.txt
```

log

Liste des commits (dans le dépôt local)

```
$ git log
commit c1eb479524b7352604b2cf156d73d757b735f175
Author: Isabelle BLASQUEZ <isabelle.bl[redacted]@[redacted]>
Date:   Fri Feb 17 09:16:52 2017 +0100

    Premier Commit
```

show

Détail d'un commit (du dépôt local)

```
$ git show
commit c1eb479524b7352604b2cf156d73d757b735f175
Author: Isabelle BLASQUEZ <isabelle.b[redacted]@[redacted]>
Date:   Fri Feb 17 09:16:52 2017 +0100

    Premier Commit

diff --git a/test.txt b/test.txt
new file mode 100644
index 0000000..306c18d
--- /dev/null
+++ b/test.txt
@@ -0,0 +1 @@
+Bonjour tout le monde !
\ No newline at end of file
```

A propos du commit

Pour identifier une version, git s'appuie sur l'objet de type commit.

Un commit est l'unité de sauvegarder de git.
C'est une image du répertoire de travail à l'instant t
représentée à l'aide de **trees** et de **blobs**

Chaque objet est identifié
40 caractères hexadécimaux
(**somme de contrôle SHA-1** de son contenu)

Chaque commit est répertorié avec son **auteur**, sa **date**, son **message** et son **SHA-1**
ainsi que des **pointeurs vers un(des) commit(s) parent(s).**

A propos du dépôt Git

Un dépôt Git peut être vu comme une collection d'objets liés entre eux.

L'historique d'un dépôt Git est l'ensemble des versions du répertoire de travail (succession de commits)

La tête (HEAD) de l'historique est un pointeur vers le dernier commit (utilisé comme prochain commit parent)

En résumé : les espaces de travail

Git utilise **3 espaces différents** pour manipuler les données.

espace de travail
(working directory)

Dossier du projet tel qu'il est stocké sur le disque.

Les fichiers dans cette zone peuvent être connus ou non de git

add

Index
(staging index)

Espace de Transit

Données en préparation pour le prochain commit (Fichiers à intégrer au prochain commit)

commit

Dépôt local
(local repository)

contient tous les commits c-a-d contient l'historique du projet

*La **tête de l'historique** (**HEAD**) pointe vers le commit le plus récent de la branche courante*

status

Recherche les fichiers stagés (en attente d'être commités) ou non

log *Liste des commits*

show *Détail d'un commit*

Isabelle BLASQUEZ

Enregistrer de nouvelles modifications dans un nouveau commit (en une seule ligne)

```
git commit -am "message"
```

Tous les fichiers *trackés*
c-a-d
ceux qui ont déjà été
ajoutés à l'index une fois

espace de travail
(working directory)

Demo

- test.txt
- .git

test.txt

Bonjour tout le monde !
J'utilise git !

Dépôt local
(local repository)

HEAD

c1eb479 — 9085634

```
$ git commit -am "ajout git"
[master 9085634] ajout git
 1 file changed, 2 insertions(+), 1 deletion(-)
```

```
$ git log
commit 908563408714f417561a443e4766821c30b25ec6
Author: Isabelle BLASQUEZ <isabelle.b[redacted]>
Date:   Fri Feb 17 11:10:45 2017 +0100

    ajout git

commit c1eb479524b7352604b2cf156d73d757b735f175
Author: Isabelle BLASQUEZ <isabelle.b[redacted]>
Date:   Fri Feb 17 09:16:52 2017 +0100

    Premier Commit
```

log

```
$ git show
commit 908563408714f417561a443e4766821c30b25ec6
Author: Isabelle BLASQUEZ <isabelle.bl[redacted]1@[redacted]>
Date:   Fri Feb 17 11:10:45 2017 +0100

    ajout git

diff --git a/test.txt b/test.txt
index 306c18d..9e77514 100644
--- a/test.txt
+++ b/test.txt
@@ -1 +1,2 @@
-Bonjour tout le monde !
\ No newline at end of file
+Bonjour tout le monde !
+J'utilise git !
\ No newline at end of file
```

show

Ignorer des fichiers dans l'historique (`.gitignore`)

Certains **fichiers n'ont pas besoin d'être versionnés,** et ne doivent **pas être *trackés***

(fichiers de log, fichiers résultants d'une compilation, fichiers de configuration, …)

… à spécifier dans un fichier **`.gitgnore`** à la racine du dépôt.

*(non trackés par git, ils ne seront pas détectés comme ayant subi un changement (**`modified`**) et potentiellement commitable)*

introduit une ligne de commentaires

.gitignore

```
# Eclipse
.classpath
.project
.settings/

# Maven
target/
```

(Exemple pour un projet Maven sous Eclipse)

Quelques liens à consulter pour personnaliser son **`.gitignore`** :

https://git-scm.com/docs/gitignore#_pattern_format , https://github.com/github/gitignore
http://gary-rowe.com/agilestack/2012/10/12/a-gitignore-file-for-intellij-and-eclipse-with-maven , https://fr.atlassian.com/git/tutorials/gitignore

Consulter l’historique

(`git status`, `git log`, `git diff`)

espace de travail
(working directory)

```
git status
```

Liste les fichiers stagés, non stagés et non trackés

Dépôt local
(local repository)

```
git log
```

Affiche l'ensemble de l'historique (tous les commits)

```
git log -n <limite>
```

Affiche les derniers n commits où n=<limite>

`git log-n-3` *pour les 3 derniers commits*

```
git log -oneline
```

Affichage de l'historique où chaque commit est affiché sur une seule ligne

```
git log –author="nom auteur"
```

Affichage uniquement les commits filtrés avec l'auteur spécifié

```
git log <fichier>
```

Affichage uniquement des commits Contenant le fichier spécifié

```
…
```

Visualiser les différences (`git diff`)

```
git diff commit1..commit 2
```

différences entre deux commits

(Exemple : modifications introduites par le dernier commit)

```
$ git diff c1eb479..HEAD
diff --git a/test.txt b/test.txt
index 306c18d..9e77514 100644
--- a/test.txt
+++ b/test.txt
@@ -1 +1,2 @@
-Bonjour tout le monde !
\ No newline at end of file
+Bonjour tout le monde !
+J'utilise git !
\ No newline at end of file
```

```
git diff
```

différences entre espace de travail et index

```
git diff --cached
```

différences entre l'index et le HEAD

```
git diff HEAD
```

différences entre espace de travail et le HEAD

```
…
```

Consulter un changement c-a-d se positionner sur un ancien commit de l'historique

(`git checkout`)

Se positionner sur un commit donné (`git diff`)

```
git checkout commitparSonSHA
```

Consulter (afficher) une ancienne version, sans toucher à votre espace de travail

Le checkout d'un commit transforme l'espace de travail pour afficher un ancien état de votre projet sans modifier son état actuel.

L'historique n'est pas modifié…

… ainsi pour retrouver l'état actuel de l'espace de travail, il suffit de …

```
git checkout master
```

Revenir au commit le plus récent de la branche principale

Possible aussi pour un fichier en particulier :

```
git checkout commitparSonSHA <fichier>
```

… suivi de …

```
git checkout HEAD <fichier>
```

Annuler un changement

(`git revert`, `git reset`)

J'ai fait mon commit un peu trop vite… est-ce que je peux l'annuler ?

```
git revert commitparSonSHA
```

Revenir à un état antérieur en créant un nouveau nouveau commit *(défait en créant un nouveau commit, mais ne revient pas en arrière à proprement parlé…)*

```
git reset HEAD
```

Annuler les changements dans l'index
(remet l'index dans le même état que la version la plus récente du dépôt)

```
git reset --hard HEAD
```

Annuler les changements de l'espace de travail et de l'index
*(Attention, **commande irréversible** !!!)*

```
git commit --amend -m "Votre nouveau message"
```

Modifier le message du commit *(possible uniquement si commit pas non pushé sur dépôt distant (origin))*

D'après https://openclassrooms.com/courses/gerer-son-code-avec-git-et-github/se-positionner-sur-un-commit-donne
Et https://fr.atlassian.com/git/tutorials/undoing-change

Créer des Branches

(`git branch, git checkout`)

Isabelle BLASQUEZ

A propos des branches

La branche par défaut est `master`

Une branche correspond à une version en parallèle de celle en cours de développement.

Une branche peut servir à développer de nouvelles fonctionnalités, à corriger des bugs sans pour autant intégrer ces modifications à la version principale du logiciel

Créer une branche et en faire la branche courante

```
git branch une-branche
```

Créer une branche
Un nouveau pointeur (une-branche) est pour l'instant simplement créé sur le commit courant.
L'historique ne change pas

Nom de branches : ne pas commencer par un tiret, ni deux points consécutifs, ne pas terminer par un slash

```
git checkout une-branche
```

Se positionner sur la branche fraichement créée *pour qu'elle devienne la branche courante (celle qui recevra le prochain commit)*

Ou en une seule ligne de commande :

```
git checkout -b une-branche
```

En pratique …

1. Créer une nouvelle branche *anglais* et se positionner dessus

```
git checkout -b anglais
```

2. Travailler dans l'espace de travail

test.txt

```
Bonjour tout le monde !
J'utilise git !
```

test_EN.txt

```
Hello world !
I'm using git !
```

3. Commiter
(sur *anglais* puisqu'on est positionné sur cette branche !)

Si nouveau fichier, il doit être ajouté à l'index !

```
git add test_EN.txt
```

```
git commit -am "message en anglais"
```

Que se passe-t-il lorsqu'on change de branche ?

→ **Le répertoire de travail est modifié**
→ **L'index reste intact**
→ **La référence HEAD est mise à jour sur le commit le plus récent de la branche choisie (nouvelle branche courante)**

Liste des branches existantes

```
git branch
```

```
$ git branch
* anglais
  master
```

*** indique la branche courante**
(c-a-d la branche qui contiendra le commit)

<u>Remarque</u> : Le prompt de la console (invite de commande) indique la branche courante :

```
demo (anglais)
```

: si **`anglais`** est la branche courante

ou

```
demo (master)
```

: si **`master`** est la branche courante

Créer (tirer) une autre branche …

La branche est tirée depuis la branche courante …

```
git branch espagnol
```

```
git checkout master
git branch espagnol
```

Fusionner des branches

(`git merge`)

Images :
https://twitter.com/girlie_mac/status/906691511496011778
https://twitter.com/tut_tuuut/status/1091976679684300801

Isabelle BLASQUEZ

Fusionner des branches

Intégrer les modifications faites sur une branche dans une autre branche.
(Récupération et intégration des différences d'une branche dans une autre)

Fusionner une branche : Principe général (fusion avec commit)

Si on est sur `master`
et
qu'on souhaite fusionner
ma-branche dans master :

```
git merge ma-branche
```

→ **git recherche le dernier commit en commun (commit « ancêtre commun »)**

→ **git crée un nouveau commit sur master qui contiendra les modifications apportées par ma-branche (commit de fusion)**

Grâce au commit de fusion, les deux branches gardent leur identité dans le graphe de l'historique

Fusionner une branche : Cas particulier de l'avance rapide (`Fast-Forward`)

Si pas de modification sur le « commit ancêtre » depuis la création de la branche…

```
git merge ma-branche
```

La fusion `Fast-Forward` a simplement pour effet de déplacer le sommet de la branche cible

⇒ pas de commit de fusion, simple avance de pointeur : gain de temps … mais …

Avec une avance rapide, l'historique restera linéaire (perte d'indentité)

<u>Remarque :</u> possibilité de forcer la création d'un commit de fusion, même dans cette situation, grâce à l'ajout de l'option **--no-ff** : ***no fast forward***)

Fusion Fast-Forward : Exemple

git merge anglais

```
$ git merge anglais
Updating 9085634..9f68c02
Fast-forward
 test_EN.txt | 2 ++
 1 file changed, 2 insertions(+)
 create mode 100644 test_EN.txt
```

```
$ git log
commit 9f68c02c75df9b88d138ba7559f34991f0b1b264
Author: Isabelle BLASQUEZ <isabelle.[redacted]@[redacted]>
Date:   Tue Mar 7 16:18:47 2017 +0100

    message en anglais

commit 908563408714f417561a443e4766821c30b25ec6
Author: Isabelle BLASQUEZ <isabelle.[redacted]@[redacted]>
Date:   Fri Feb 17 11:10:45 2017 +0100

    ajout git

commit c1eb479524b7352604b2cf156d73d757b735f175
Author: Isabelle BLASQUEZ <isabelle.[redacted]@[redacted]>
Date:   Fri Feb 17 09:16:52 2017 +0100

    Premier Commit
```

git log confirme la présence d'uniquement 3 commits

Remarque : La branche anglais existe toujours, possibilité de s'y positionner dessus avec **checkout**

Fusion classique : Exemple

test.txt

```
Bonjour tout le monde !
J'utilise git !
```

test_EN.txt

```
Hello world !
I'm using git !
git is a distributed version control system.
```

test_EN.txt

```
Hello world !
I'm using git !
```

test.txt

```
Bonjour tout le monde !
J'utilise git !
git est un logiciel de gestion de versions décentralisé.
```

test.txt

```
Bonjour tout le monde !
J'utilise git !
git est un logiciel de gestion de versions décentralisé.
```

test_EN.txt

```
Hello world !
I'm using git !
git is a distributed version control system.
```

```
$ git merge anglais -m "fusion définition git en anglais"
Merge made by the 'recursive' strategy.
 test_EN.txt | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
```

Comme la fusion créée un **nouveau commit**,
Si vous ne spécifiez pas de **message**,
il vous sera automatiquement demandé !

Après la fusion : git log (tous les commits)

```
$ git log
commit 8186d8f3c68841eef8e63dcc7a04e195aa09346c
Merge: a51cb7e fa9ed92
Author: Isabelle BLASQUEZ <isabelle.[redacted]@[redacted]>
Date:   Thu Mar 9 11:45:00 2017 +0100

    fusion définition git en anglais

commit a51cb7ed0335be223a252e4d21630701b8e25c2a
Author: Isabelle BLASQUEZ <isabelle.[redacted]@[redacted]>
Date:   Thu Mar 9 11:43:20 2017 +0100

    ajout définition git en français

commit fa9ed9295e405f55ccb815051708e06d9ca7bdb7
Author: Isabelle BLASQUEZ <isabelle.[redacted]@[redacted]>
Date:   Thu Mar 9 11:42:00 2017 +0100

    ajout définition git en anglais

commit 9f68c02c75df9b88d138ba7559f34991f0b1b264
Author: Isabelle BLASQUEZ <isabelle.[redacted]@[redacted]>
Date:   Tue Mar 7 16:18:47 2017 +0100

    message en anglais

commit 908563408714f417561a443e4766821c30b25ec6
Author: Isabelle BLASQUEZ <isabelle.[redacted]@[redacted]>
Date:   Fri Feb 17 11:10:45 2017 +0100

    ajout git

commit c1eb479524b7352604b2cf156d73d757b735f175
Author: Isabelle BLASQUEZ <isabelle.[redacted]@[redacted]>
Date:   Fri Feb 17 09:16:52 2017 +0100

    Premier Commit
```

Tagger un commit

(`git tag`)

A propos des tags

Un tag est un alias (nom)
dont le rôle est de pointer vers un commit…

… pour éviter l'utilisation des hash SHA-1,
peu explicites et difficiles à retenir

Les tags sont utilisés pour marquer des numéros de version
sur les commits

Exemple : x.y.z (numérotation selon SemVer (*Sem*antic *Ver*sion)):
x : version majeure (modifications dans le fonctionnement de l'application :
incompatibilité avec une version précédente)
y : version mineure (ajout des fonctionnalités:
compatibilité avec l'ancien système)
z : patch (correction de bugs sans ajout de fonctionnalité)

A propos de SemVer (*Sem*antic *Ver*sion)

Tout savoir sur
la gestion sémantique des versions …

sur : http://semver.org/

2.0.0 1.0.0

Gestion sémantique de version 2.0.0

En bref

Étant donné un numéro de version MAJEUR.MINEUR.CORRECTIF, il faut incrémenter :

1. le numéro de version MAJEUR quand il y a des changements non rétrocompatibles,
2. le numéro de version MINEUR quand il y a des changements rétrocompatibles,
3. le numéro de version de CORRECTIF quand il y a des corrections d'anomalies rétrocompatibles

Des libellés supplémentaires peuvent être ajoutés pour les versions de pré-livraison et pour des méta-données de construction sous forme d'extension du format MAJEURE.MINEURE.CORRECTIF.

Introduction

Dans le monde de la gestion des logiciels, il existe un endroit redouté appelé "l'enfer des dépendances" (de l'anglais "dependency hell"). Plus votre système se développe et plus vous intégrez de composants dans votre logiciel, plus vous êtes susceptible de vous trouver un jour dans cette abîme de désespoir.

Dans les systèmes comportant de nombreuses dépendances, publier une nouvelle version d'un composant peut vite devenir un cauchemar. Si les règles de dépendance sont trop strictes, vous risquez de verrouiller vos versions (incapacité de mettre à jour un composant sans avoir à publier une nouvelle version de chaque composant qui en dépend). Si les règles de dépendances sont trop lâches, vous allez inévitablement subir la promiscuité de version (supposer une compatibilité avec plus de futures versions que raisonnable). L'enfer des dépendances est l'endroit où vous vous trouvez lorsque vous êtes bloqué dans une version et/ou qu'une incompatibilité de version vous empêche d'avancer sans risque dans votre projet.

Comme solution à ce problème, je propose un ensemble de règles et d'exigences simples qui dictent la façon dont les numéros de version sont attribués et incrémentés. Ces règles sont basées mais pas nécessairement limitées à des pratiques très répandues aussi bien dans les domaines du logiciel privé que du logiciel libre.

Les commande git autour des tags …

Créer un tag sur le commit courant : `git tag nom-du-tag`

Exemple :

```
git tag v1.0.0
```

Par la suite, il sera possible de revenir sur le commit via `git checkout v1.0.0`

Liste des tags

```
git tag --list
```

Détail d'un tag

```
git show nom-du-tag
```

Supprimer un tag

```
git tag -d nom-du-tag
```

Créer un tag à partir de son hash : `git tag nom-du-tag hash`

Créer un tag annoté `git tag -a nom-du-tag` + message

```
git tag -am nom-du-tag " message "
```

Gérer un conflit

Isabelle BLASQUEZ

…Mais parfois, git peut ne pas savoir pas comment réaliser la fusion

conflit

test.txt

```
Bonjour tout le monde !
git a été créé par Linus Torvalds en 2005.
J'utilise git !
git est un logiciel de gestion de versions décentralisé.
```

test_EN.txt

```
Hello world !
git was created by Linus Torvalds in 2005.
I'm using git !
git is a distributed version control system
```

Et maintenant, si on essayait de fusionner les modifications de `espagnol` dans `master` :

```
$ git merge espagnol -m "fusion message en espagnol"
Auto-merging test.txt
CONFLICT (content): Merge conflict in test.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Une petit correction d'erreur au passage…

test.txt

```
Bonjour le monde !
J'utilise git !
git est un logiciel de gestion de versions décentralisé.
```

test_EN.txt

```
Hello world !
I'm using git !
git is a distributed version control system.
```

test_ES.txt

```
Hola Mundo !
git fue creado por Linus Torvalds en 2005.
Estoy usando git !
git es un software de control de versiones.
```

… Et il y a un Conflit à résoudre !

```
$ git merge espagnol -m "fusion message en espagnol"
Auto-merging test.txt
CONFLICT (content): Merge conflict in test.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Git met la fusion en pause (le promt indique `(master|MERGING)`)
Et demande à l'utilisateur de résoudre le conflit manuellement

Pour voir les fichiers en conflit : `git status`

```
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")

Changes to be committed:

        new file:   test_ES.txt

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both modified:   test.txt
```

Fichier qui présente un conflit :
`test.txt`

2 solutions :

→ abandonner la fusion `git merge --abort`

→ ou **résoudre manuellement le conflit**

Pour résoudre manuellement le conflit, ouvrir le fichier qui présente le conflit …

<<<<<<< branche-de-base
début du conflit avec le contenu de la branche de base (**HEAD** qui correspond à master)

=======
séparation du contenu des deux branches

>>>>>>> branche-a-fusionner
fin du conflit avec ci-dessus le contenu en conflit de la branche à fusionner (ici branche **espagnol**)

test.txt

```
<<<<<<< HEAD
Bonjour tout le monde !
git a été créé par Linus Torvalds en 2005.
=======
Bonjour le monde !
>>>>>>> espagnol
J'utilise git !
git est un logiciel de gestion de versions décentralisé.
```

Résoudre le conflit et finaliser la fusion …

1. Corriger manuellement le conflit de manière à obtenir le fichier souhaité

test.txt

```
Bonjour le monde !
git a été créé par Linus Torvalds en 2005.
J'utilise git !
git est un logiciel de gestion de versions décentralisé.
```

2. Avertir git que le conflit a été résolu…
(`git add`)

```
$ git add test.txt
```

…et vérifier que git a bien pris en compte que le conflit a été résolu
(`git status`)

```
$ git status
On branch master
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:

        modified:   test.txt
        new file:   test_ES.txt
```

3. Terminer la fusion par un commit
(`git commit` *avec un message par défaut prédéfini ou préciser le message*)

```
$ git commit -m "fusion message en espagnol"
[master f304b2d] fusion message en espagnol
```

… et le prompt redevient `(master)`

Supprimer une branche

Supprimer une branche

```
git branch –d branch-a-supprimer
```

Cette commande supprime réellement le pointeur de la branche.
A utiliser avec précaution, uniquement si :

→ la branche a été fusionnée dans master et
aucune autre modification n'est prévue dans cette branche

→ les modifications de la branche ne sont plus à l'ordre du jour et elles ne seront jamais intégrées.

Rebaser

une branche dans une autre

Intégrer les modifications d'une branche dans une autre : Merge vs Rebase

```
git checkout feature
git merge master
```

```
git checkout feature
git rebase master
```

merge : intégrer plusieurs commits cohérents entre eux dans un **commit unique** représentant tous les commits mergés

rebase : réécriture de l'historique pour intégrer **tous les commits** d'une branche dans une autre (rendre l'historique linéaire)

⇒ *permet de rejouer une suite de commits à partir d'un autre point de l'histoire*

Extrait : https://fr.atlassian.com/git/tutorials/merging-vs-rebasing

Merge vs Rebase … avec des chats

Image :
https://twitter.com/girlie_mac/status/906691511496011778

A propos de `merge` et `rebase`

Il ne faut pas rebaser des commits envoyés sur un dépôt distant c-a-d
Ne pas rebaser un historique public
(un `rebase` revient à transformer des commits comme s'ils étaient supprimés pour être recréés)

En savoir plus sur `merge` et `rebase` :

→ **Bien utiliser git merge et rebase**
https://delicious-insights.com/fr/articles/bien-utiliser-git-merge-et-rebase/

→ **Comparaison entre un merge et un rebase**
https://fr.atlassian.com/git/tutorials/merging-vs-rebasing

→ **Git++ : Passez au niveau supérieur de la gestion de version**
https://www.youtube.com/watch?v=m0_C2cfM9IM
https://www.youtube.com/watch?v=rt-9mPaYtKo
Transparents : http://webadeo.github.io/git-simpler-better-faster-stronger/#1.0

→ **Exercices interactifs pour apprendre le branching sous Git**
http://learngitbranching.js.org/

Extrait : https://fr.atlassian.com/git/tutorials/merging-vs-rebasing

Dépôt distant (ou remote)

Dépôt distant

Un dépôt git peut être lié à d'autres dépôts git dits ***distants*** (ou ***remote repository***) avec lesquels il pourra partager des commits.

Un dépôt distant a un emplacement qui peut être :
→ un répertoire (sur un disque local ou partagé),
→ ou une URL (par exemple https://**github.com**/iblasquez/tuto_git).

Un *remote* peut être utilisé comme **répertoire central**, auquel tous les développeurs se synchronisent.

git fournit à chaque développeur sa propre copie du dépôt, avec son propre historique local et sa structure de branche.

Images extraites de https://fr.atlassian.com/git/tutorials/setting-up-a-repository et https://fr.atlassian.com/git/tutorials/syncing et https://fr.atlassian.com/git/tutorials/comparing-workflows

Cloner un dépôt distant

```
git clone emplacement-du-depot-à-cloner
```

Le clone peut se faire selon plusieurs protocoles : directement en local, via HTTPS, SSH ou git

Exemple : cloner un depôt distant sur Github (`https`)

Que se passe-t-il lors d'un `clone` ?

```
$ git clone https://github.com/iblasquez/geekgit.git
Cloning into 'geekgit'...
remote: Counting objects: 9, done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 9 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (9/9), done.
Checking connectivity... done.
```

→ un nouveau **dépôt local** **`geekgit`** est créé.

→ ce dépôt est lié au dépôt distant connu sous le nom **`origin`**

(c-a-d **`origin`** est un raccourci qui pointe vers l'URL du dépôt distant : https://github.com/iblasquez/geekgit.git)

```
$ git log
commit 897bb18c454879da7215ea8d7f905e4d067bae1a
Author: Isabelle BLASQUEZ <isabelle[redacted]@g[redacted]
Date:   Fri Mar 10 14:19:04 2017 +0100

    Ajout lien poesie.md

commit 0425e967093beb97763acfec269525bdb08778d5
Author: Isabelle BLASQUEZ <isabelle[redacted]@g[redacted]
Date:   Fri Mar 10 14:17:08 2017 +0100

    Ajout poésie : Source Code Poetry Challenge

commit fc8cd200702f6052bd81a474e234910171e94f86
Author: Isabelle BLASQUEZ <isabelle[redacted]@g[redacted]
Date:   Fri Mar 10 13:36:29 2017 +0100

    Initial commit
```

→ les commits de l'origine sont récupérés dans ce dépôt

Travailler en local ...

peinture.md

```
# Peinture & Développement logiciel


## Classic Programmer Paintings


Connaissez-vous le site **Classic Programmer Paintings**  ?

[Classic Programmer Paintings is a hilarious resource with classic paintings featured with
modern captions from the programming world.](http://mamchenkov.net/wordpress/2016/08/08/classic-
programmer-paintings)*


Quelques exemples :

<img src="http://68.media.tumblr.com/498dad1313019732b052d78d2738aeac/
tumblr_o5od680RFt1ugyavxo1_1280.jpg" width="600">

[*Programmer finds 1395 conflicts after `git merge develop master`, three days before deadline*
Gustav Courbet
1844-1845
Oil paint
(Collaboration form Mario)](http://classicprogrammerpaintings.com/post/143064709101/programmer-
finds-1395-conflicts-after-git-merge)
```

un commit ...

```
Isabelle@xps-Isa MINGW64 /c/demoRemote/geekgit (master)
$ git add peinture.md

Isabelle@xps-Isa MINGW64 /c/demoRemote/geekgit (master)
$ git commit -am "ajout page peinture avec classic programmer paintings"
[master 51b755d] ajout page peinture avec classic programmer paintings
 1 file changed, 71 insertions(+)
 create mode 100644 peinture.md
```

Ajouter un fichier

README.md

```
# Hello les geeks !

Pour commencer un peu de poésie par [ici](poesie.md)

Et quelques peintures par [là](peinture.md)
```

un autre commit ...

```
Isabelle@xps-Isa MINGW64 /c/demoRemote/geekgit (master)
$ git commit -am "maj README : lien page peinture"
[master 501f13b] maj README : lien page peinture
 1 file changed, 2 insertions(+)
```

Modifier un fichier

Synchroniser les dépôts : publier ses commits (push)

POUSSER

```
git push <depot-distant> <branche-locale>
```

Envoyer les modifications de la branche `master` du dépôt local sur le dépôt distant (`origin`)

```
git push origin master
```

```
$ git push origin master
Counting objects: 6, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 2.01 KiB | 0 bytes/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To https://github.com/iblasquez/geekgit.git
   897bb18..501f13b  master -> master
```

Les identifiants Github peuvent être demandés avant de pousser ;-)

Remarques : Pousser toutes les branches `git push --all`

Intéressant à lire : http://blog.xebia.fr/2017/01/17/git-depots-distants-sans-acces-reseau/

Isabelle BLASQUEZ

Chez mon client du jour 😀 #BlagueDeDev ?

Il existe même un plugin **`git-fire`** à découvrir **sur** **https://github.com/qw3rtman/git-fire** **(**Save Your Code in an Emergency)

Extraits de https://twitter.com/agilex/status/836123942683308032
https://openclipart.org/detail/130795/trollface
Voir aussi : https://www.reddit.com/r/ProgrammerHumor/comments/3nc531/in_case_of_fire

Affiche à imprimer depuis : http://abload.de/img/in_case_of_fireirrtb.jpg ou
https://hikaruzone.wordpress.com/2015/10/06/in-case-of-fire-1-git-commit-2-git-push-3-leave-building

Isabelle BLASQUEZ

Le dépôt distant a subi/reçu de nouvelles modifications …

iblasquez / geekgit

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki

dépôt utilisé pour le cours git

Add topics

5 commits | 1 branch

Branch: master | New pull request

iblasquez maj README : lien page peinture

README.md	maj README : lien page peinture
peinture.md	ajout page peinture avec classic programmer paintings
poesie.md	Ajout poésie : Source Code Poetry Challenge

iblasquez / geekgit

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Pulse

dépôt utilisé pour le cours git

Add topics

7 commits | 1 branch | 0

Branch: master | New pull request

iblasquez committed on GitHub Maj README : lien page BD

README.md	Maj README : lien page BD
bandeDessinee.md	ajout bd : Commit Strip
peinture.md	ajout page peinture avec classic programmer paintings
poesie.md	Ajout poésie : Source Code Poetry Challenge

Synchroniser les dépôts : récupérer des commits distants (`pull`)

TIRER

Télécharger les commits du dépôt distant de la branche courante…

```
git fetch <depot-distant>
```

… *puis* Fusionner ces commits à la branche locale (`git merge origin/master`)

```
git merge origin/<branche-locale>
```

***Ou directement* : Recevoir sur la branche courante les modifications en provenance du dépôt distant**

```
$ git pull origin
remote: Counting objects: 6, done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (6/6), done.
From https://github.com/iblasquez/geekgit
   501f13b..7d770d2  master     -> origin/master
Updating 501f13b..7d770d2
Fast-forward
 README.md        |  2 ++
 bandeDessinee.md | 26 ++++++++++++++++++++++++++
 2 files changed, 28 insertions(+)
 create mode 100644 bandeDessinee.md
```

```
git pull <depot-distant>
```

Possibilité de demander un rebase :
`git pull --rebase <depot-distant>`

Remarque : Si un conflit se présente `git merge --abort` revient à l'état avant le `pull`.

Isabelle BLASQUEZ

Les workflows
(modèles de développement)

A lire : *La puissance des workflows git* : https://medium.com/@OVHUXLabs/la-puissance-des-workflows-git-12e195cafe44#.rbkmhjcc2

Git Flow : un workflow à base de branches (1/2)

Les branches principale

→ **master** sur **origin**
(doit rester stable : prêt pour être déployé en production)

→ **develop** (appelée aussi branche d'*intégration*)
(reflète les derniers changements livrés pour la prochaine version)

Les branches de support (à durée de vie limitée)

→ les branches pour les **fonctionnalités** (features)

→ les branches pour les **versions** (releases)
(préparer les nouvelles versions de production, correction d'anomalies mineures, préparation des méta-données pour une version (numéro, date de compilation, etc.)

→ Les branches de **correctifs** (hotfixes)

Article original : http://nvie.com/posts/a-successful-git-branching-model/
Version française : https://www.occitech.fr/blog/2014/12/un-modele-de-branches-git-efficace/
Pour faciliter sa mise en place voir Git flow : http://danielkummer.github.io/git-flow-cheatsheet/

Git Flow : un workflow à base de branches (2/2)

→ Chaque développeur **pull** et **push** sur **origin**

→ Au delà de la relation centralisée push-pull, chaque développeur peut aussi **récupérer des changements d'autres équipiers** et ainsi former des **sous-équipes**…

→ …utile quand deux ou plusieurs développeurs travaillent ensemble sur une fonctionnalité, plutôt que de pousser prématurément le travail en cours sur origin.

Article original : http://nvie.com/posts/a-successful-git-branching-model/
Version française : https://www.occitech.fr/blog/2014/12/un-modele-de-branches-git-efficace/
Pour faciliter sa mise en place voir Git flow : http://danielkummer.github.io/git-flow-cheatsheet/

GitHub Flow : un workflow branché simplifié …

1 **master** et des branches **features** !!!

Un développeur ne doit pas fusionner sa branche dans master lorsqu'il pense que c'est bon, un autre doit venir faire une revue du code et confirmer la stabilité de la branche.

Extrait : https://guides.github.com/introduction/flow

Un exemple de workflow mis en place lors d'un projet tuteuré

1 fonctionnalité => 1 branche

Periodic table of devops tools

Autres outils autour de la gestion de version

Carte interactive disponible sur : https://xebialabs.com/periodic-table-of-devops-tools/

A compléter par :
https://xebialabs.com/glossary/

Isabelle BLASQUEZ

Le versioning, c'est important

mardi 12 septembre 2017

Extrait : http://www.commitstrip.com/fr/2017/09/12/versioning-is-important

Des simulateurs web pour se familiariser avec les commandes `git` de manière visuelle

http://git-school.github.io/visualizing-git

https://nic-hartley.github.io/git-gud/

Isabelle BLASQUEZ

Annexes

Pour s’entraîner et comprendre visuellement les premiers concepts de git

https://learngitbranching.js.org/

Isabelle BLASQUEZ

Un mémo interactif (1/2)

http://ndpsoftware.com/git-cheatsheet.html

GIT CHEATSHEET

AN INTERACTION FROM NDP SOFTWARE

en fr 简 es de

previous version»
escape a git mess, step-by-step»
discover character entities at &what»

REMISE | ESPACE DE TRAVAIL | INDEX | DÉPÔT LOCAL | DÉPÔT DISTANT

status
diff
diff <commit ou branche>
add <fichier(s) ou dossier(s)>
add -u
rm <fichier(s)>
mv <fichier(s)>
commit -a [-m 'message']
checkout <fichier(s) ou dossier(s)>
reset --hard
checkout <branche>
checkout -b <branche>
merge <commit ou branche>
rebase <source>
cherry-pick <commit>
revert <commit>
clone <dépôt_distant>
pull <dépôt_distant> <référence>
reset --hard <dépôt_distant> <branche>
clean
stash save ['message']
stash apply [état]
stash pop

G+1 +2314

(c) Andrew Peterson 2009-2016 All Rights Reserved.

Commandes git autour de l'espace de travail

Un mémo interactif (2/2)

http://ndpsoftware.com/git-cheatsheet.html

Commandes git autour du dépôt distant

Reste à découvrir en ligne… commandes git autour :

→ de la remise
→ de l'index
→ du dépôt local

Aide-mémoire de commandes git

https://education.github.com/git-cheat-sheet-education.pdf

https://www.jrebel.com/blog/git-cheat-sheet

http://rogerdudler.github.io/git-guide/index.fr.html

https://docs.github.com/en

Bien nommer sous git :

→ **Guide de style git :** https://github.com/pierreroth64/git-style-guide

→ **How to use GitHub like a proper human being :** https://stories.devacademy.la/how-to-use-github-like-a-proper-human-being-1a9c895c4e13#.y5owzljfh

Time for a Change

Let's make a change to our code — first we'll... whooaooaaaaa there! We're going to use terminology here which we'll need to explain and point at a state diagram to understand fully. Check out this image:

Extrait : https://zeroturnaround.com/rebellabs/git-commands-and-best-practices-cheat-sheet/

Extrait : http://stackoverflow.com/questions/2745076/what-is-the-difference-between-git-commit-and-git-push

commit: adding changes to the local repository
push: to transfer the last commit(s) to a remote server

Git : Quelques liens …

✓ ***Git : le site de référence : https://git-scm.com***

✓ ***Documentation***

→ ***Livre Pro Git Book***
version française en ligne : https://git-scm.com/book/fr/v2

→ ***Git Wiki Homepage : https://git.wiki.kernel.org***
(à voir aussi sur ce wiki un sondage sur l'utilisation de Git en 2016: https://git.wiki.kernel.org/index.php/GitSurvey2016)

✓ ***Liste de projets sous Git :***

GitProjects

Projects that use Git for their source code management.

If your project is not listed, please add it here.

List does not currently include Git or git-related tools; see InterfacesFrontendsAndTools page for those.

Kernel-related projects that use Git for their source code management.

- Linux Kernel
  - http://kernel.org
  - http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git (gitweb)
- Linux/MIPS (port of Linux to the MIPS architecture)
  - http://www.linux-mips.org/wiki/Git (description, includes repository URLs)
  - http://www.linux-mips.org/git (gitweb)
- NFS client patches for Linux
  - http://linux-nfs.org/ (homepage)
  - http://linux-nfs.org/cgi-bin/gitweb.cgi (gitweb)
- OpenVZ (Containers virtualization for Linux)

… toute la liste à découvrir sur ***https://git.wiki.kernel.org/index.php/GitProjects***

Références sur git

Git

- Site de référence : https://git-scm.com
  Documentation sur le site de référence : https://git-scm.com/doc
- Pro Git book (guide officiel) en version française
- Listes d'outils permettant d'utiliser git en mode graphique : ici et là
- A propos de .gitgnore
  La rubrique gitignore dans la documentation de git
  A collection of useful .gitignore templates
  A .gitignore file for Intellij and Eclipse with Maven
- Bien nommer sous git
  Guide de style git
  How to use GitHub like a proper human being
- Aide mémoire de commandes git
  https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf
  https://education.github.com/git-cheat-sheet-education.pdf https://zeroturnaround.com/rebellabs/git-commands-and-best-practices-cheat-sheet/ https://www.git-tower.com/blog/git-cheat-sheet/
- A propos des branches
  Utiliser des branches Rubrique Branching in Eclipse du tutorial de vogella.com
- Merge et Rebase
  Bien utiliser Git merge et rebase
  Comparaison entre un merge et un rebase
  Resolving a merge conflict : wiki Eclipse
  Git++ : Passez au niveau supérieur de la gestion de version et une vidéo ici et là
  Exercices interactifs pour apprendre le branching sous Git
- Autres
  Reset, checkout et revert
  Tagging
- Comment créer des dépôts GIT distants sans accès réseau

Egit (Tutoriel complet)

EGit/User Guide sur le wiki d'eclipse.org EGit Tutorial sur eclipsesource.com
Git version control with Eclipse (EGit) - Tutorial sur vogella.com

Github

- Guides Github
- Aide en ligne Github : ici ou là
- Mastering Markdown

Autres

- The Designers Guide to Git
- Présentation de Git sous Wikipedia
- Git et Github

Fork & Co (contribuer à un projet existant)

- Forking Projects
- Ma méthode de travail avec Git et GitHub qui est une traduction de Documenting my git/GitHub workflow
- Chapitre 6.2 : GitHub - Contribution à un projet du Pro Git book

Exercices interactifs pour apprendre à manipuler les commandes git

- Got 15 minutes and want to learn Git? (les commandes de bases : lisez, cliquez, comprenez !)
- Learn Git Branching (apprendre le branching sous git)

Tutoriels git avec Eclipse

- Permier projet dans Eclipse avec Bitbucket (José Paumard)
- Egit : quand git s'invite dans Eclipse (o penclassrooms)
- GitHub avec Eclipse et "eGit" en 15 minutes (Laurent Guérin)
- Utiliser Git avec Eclipse (Yannick Le Goff)
- Introduction à Git (Pierre-Antoine Champin)

Des vidéos à ne pas manquer ...

- Git and GitHub for Poets (une série de courtes vidéos par Coding Train) : ici
- Git++ : Passez au niveau supérieur de la gestion de version : ici et là avec les transparents ici

Retrouvez cette page de références disponible dans le dépôt : https://github.com/iblasquez/tuto_git avec un tutoriel de découverte de git au travers d'EGit

Isabelle BLASQUEZ

Références sur les workflows …

Le modèle GitFlow (workflow de base):
→ **Article original :** http://nvie.com/posts/a-successful-git-branching-model/
→ **Version française :** https://www.occitech.fr/blog/2014/12/un-modele-de-branches-git-efficace/
→ **Pour faciliter sa mise en place voir Git flow : ** http://danielkummer.github.io/git-flow-cheatsheet

Le modèle GithubFlow (workflow simplifié)
https://guides.github.com/introduction/flow/

Quel git workflow pour mon projet ?
http://www.nicoespeon.com/fr/2013/08/quel-git-workflow-pour-mon-projet

Ma méthode de travail avec Git et GitHub
https://tech.mozfr.org/post/2016/04/16/Ma-methode-de-travail-avec-Git-et-GitHub

Intégration et déploiement en continu avec Github (ou comment github utilise git…)
https://www.youtube.com/watch?v=jCwzf9adAtE

La puissance des workflows git
https://medium.com/@OVHUXLabs/la-puissance-des-workflows-git-12e195cafe44#.rbkmhjcc2

Vidéos pour comprendre facilement git et Github

Git and GitHub for Poets

The Coding Train • 9 vidéos • 18 946 vues • Dernière modification le 1 août 2016

Tout regarder | Partager | Enregistrer

1. 1.1: Introduction - Git and GitHub for Poets
   de The Coding Train
2. 1.2: Branches - Git and GitHub for Poets
   de The Coding Train
3. 1.3: Forks and Pull Requests - Git and GitHub for Poets
   de The Coding Train
4. 1.4: GitHub Issues - Git and GitHub for Poets
   de The Coding Train
5. 1.5: Intro to the Command Line - Git and GitHub for Poets
   de The Coding Train
6. 1.6: Cloning Repo and Push/Pull - Git and GitHub for Poets
   de The Coding Train
7. 1.7: git init and git add - Git and GitHub for Poets
   de The Coding Train
8. 1.8: GitHub Pages - Git and GitHub for Poets
   de The Coding Train
9. 9.10: GitHub Pages for Hosting p5.js Sketches - p5.js Tutorial
   de The Coding Train

Vidéos à regarder sur :
https://www.youtube.com/playlist?list=PLRqwX-V7Uu6ZF9C0YMKuns9sLDzK6zoiV

sur la chaîne de https://www.youtube.com/user/shiffman

Isabelle BLASQUEZ

Détail du dernier commit

auteur du commit

date de commit

message de commit

commit parent

diff par rapport à la dernière sauvegarde

hash du commit

```
$ git show
commit 908563408714f417561a443e4766821c30b25ec6
Author: Isabelle BLASQUEZ <isabelle.[redacted]l@[redacted]n>
Date:   Fri Feb 17 11:10:45 2017 +0100

    ajout git

diff --git a/test.txt b/test.txt
index 306c18d..9e77514 100644
--- a/test.txt
+++ b/test.txt
@@ -1 +1,2 @@
-Bonjour tout le monde !
\ No newline at end of file
+Bonjour tout le monde !
+J'utilise git !
\ No newline at end of file
```

show

Bien nommer ses messages de commit

Guide de style git
https://github.com/pierreroth64/git-style-guide

How to use GitHub like a proper human being
https://stories.devacademy.la/how-to-use-github-like-a-proper-human-being-1a9c895c4e13#.wwf99js5w

An emoji guide for your commit messages
https://gitmoji.carloscuesta.me

Fichiers utilisés dans ce cours pour illustrer l’interaction avec un dépôt distant

(si vous voulez rejouer les exemples pour vous entraîner …)

https://github.com/iblasquez/geekgit

README.md

```
# Hello les geeks !

Pour commencer un peu de poésie par [ici](poesie.md)

Et quelques peintures par [là](peinture.md)

Des bandes dessinées [ici](bandeDessinee.md)
```

iblasquez / geekgit

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Pulse

dépôt utilisé pour le cours git

Add topics

7 commits | 1 branch | 0

Branch: master | New pull request

iblasquez committed on GitHub Maj README : lien page BD

README.md	Maj README : lien page BD
bandeDessinee.md	ajout bd : Commit Strip
peinture.md	ajout page peinture avec classic programmer paintings
poesie.md	Ajout poésie : Source Code Poetry Challenge

README.md

Hello les geeks !

Pour commencer un peu de poésie par ici

Et quelques peintures par là

Des bandes dessinées ici

```
# Amis de la poésie !

> [**Code poetry** is literature that intermixes notions of classical poetry and computer code.(extrait
Wikipedia)](https://en.wikipedia.org/wiki/Code_poetry https://en.wikipedia.org/wiki/Code_poetry)


## Source Code Poetry Challenge

Connaissez-vous le **source code poetry challenge** qui consiste à écrire un poème de manière à ce que :

➢ The poetry must properly compile in any programming language.

Quelques exemples de soumissions des années précédentes :

<img src="http://www.sourcecodepoetry.com/file/2015/05/13/179070_151126888387332_1021759542_n_1.png"
width="600">

<img src="http://www.sourcecodepoetry.com/file/2015/05/13/551939_151125001720854_1164601800_n_1.png"
width="600">

Pour en savoir plus, rendez-vous sur le site : http://www.sourcecodepoetry.com/
et jetez un petit coup d'oeil sur le livre : [./code --poetry](https://leanpub.com/code-poetry ) <img
src="https://s3.amazonaws.com/titlepages.leanpub.com/code-poetry/hero?1479665026" width="50">

--

## Autre pratique, autre challenge : autre lien à proposer ?
```

peinture.md

```
# Peinture & Développement logiciel

## Classic Programmer PaintingsConnaissez-vous le site **Classic Programmer Paintings**  ?

[Classic Programmer Paintings is a hilarious resource with classic paintings featured with modern captions from
the programming world.](http://mamchenkov.net/wordpress/2016/08/08/classic-programmer-paintings)

Quelques exemples :

<img src="http://68.media.tumblr.com/498dad1313019732b052d78d2738aeac/tumblr_o5od680RFt1ugyavxo1_1280.jpg"
width="600">
[*Programmer finds 1395 conflicts after `git merge develop master`, three days before deadline*
Gustav Courbet
1844–1845
Oil paint
(Collaboration form Mario)](http://classicprogrammerpaintings.com/post/143064709101/programmer-finds-1395-
conflicts-after-git-merge)


<img src="http://68.media.tumblr.com/0e4f740060a2688be79044735e0909e8/tumblr_o8crfolO0W1ugyavxo1_1280.jpg"
width="600">
[*The new project manager*
Francisco Goya
1787–88
Oil on canvas](http://classicprogrammerpaintings.com/post/145506655554/the-new-project-manager-francisco-goya-
178788)
```

… la suite de peinture.md

peinture.md

```
<img src="http://68.media.tumblr.com/03a9cebc4a592e0ede5403080a8e9876/tumblr_o6mglw3KWB1ugyavxo1_1280.jpg"
width="600">
[*Haskell meetup*
Edward Hopper
Oil on canvas
1942](http://classicprogrammerpaintings.com/post/143847262458/haskell-meetup-edward-hopper-oil-on-canvas)

<img src=http://68.media.tumblr.com/ab42d8e789ea72c7d43294a8308737ae/tumblr_o5ntzfxdcy1ugyavxo1_540.jpg width="600">
[*Resolving technical debt*
Accumulated technical debt is visible in the middle image. Right image after refactoring.
About 1930 Elías García Martínez, refactored by Cecilia Giménez  Fresco
Composition image CC-BY 2.0 by cea+ in Flickr https://www.flickr.com/photos/33255628@N00/7923536516  (collaboration
Tero Kinnunen)](http://classicprogrammerpaintings.com/post/143537166294/resolving-technical-debt-accumulated-
technical)

<img src="http://68.media.tumblr.com/3f0f544ac1a4409d3e1a6db6ee093a87/tumblr_o5p7zkQ7S01ugyavxo1_400.jpg" width="600">
[*Multiple inheritance in C++*
ca 1590 - Jacopo Ligozzi
Pen and brush and brown ink over pencil, brown wash, white lead
(collaboration from Jakub Konecki)](http://classicprogrammerpaintings.com/post/143486406473/multiple-inheritance-in-c-
ca-1590-jacopo)

**La suite sur le site : http://classicprogrammerpaintings.com/   et/ou sur le compte twitter
[@progpaintings](https://twitter.com/progpaintings) **

--

## Autre lien à proposer ?
```

bandeDessinee.md

```
# Bande dessinée & Développement logiciel

## Commit Strip

Connaissez-vous **Commit Strip**  ?  Des webcomics sur la vie des codeurs ...

Quelques exemples :

<img src="http://www.commitstrip.com/wp-content/uploads/2013/11/Strips-_Old-650-final1.jpg" width="600">
Extrait de : http://www.commitstrip.com/fr/2013/11/05/git-svn-ou/

<img src="https://www.commitstrip.com/wp-content/uploads/2017/03/Strip-Move-fast-break-things.jpg" width="600">
Extrait de : http://www.commitstrip.com/fr/2017/03/10/anyone-can-be-wrong-sometimes/

<img src="https://www.commitstrip.com/wp-content/uploads/2017/02/Strip-Ou-sont-les-tests-unitaires-650-
final.jpg" width="600">
Extrait de : http://www.commitstrip.com/fr/2017/02/08/where-are-the-tests/

**La suite sur le site : http://www.commitstrip.com  et/ou sur les comptes twitter
[@CommitStrip](https://twitter.com/CommitStrip) et [@CommitStrip_fr](https://twitter.com/CommitStrip_fr)**

--

## Autre lien à proposer ?
```

Git & cats

Isabelle BLASQUEZ

Extraits :

https://twitter.com/girlie_mac/status/905270297128865792
https://twitter.com/girlie_mac/status/906691511496011778
https://twitter.com/girlie_mac/status/919348196974903296

Isabelle BLASQUEZ