

TUTORIEL D'UTILISATION DES BRANCHES AVEC GIT

DESCRIPTION DU THÈME

Propriétés	Description
Intitulé long	Tutoriel d'utilisation des branches avec GIT
Formation(s) concernée(s)	<input type="checkbox"/> Classes de première Sciences et technologies du management et de la gestion (STMG) <input type="checkbox"/> Terminale STMG Système d'information de gestion (SIG) <input checked="" type="checkbox"/> BTS Services Informatiques aux Organisations
Matière(s)	<input type="checkbox"/> Sciences de gestion <input type="checkbox"/> SIG <input checked="" type="checkbox"/> Bloc 1 – Support et mise à disposition de services informatiques <input type="checkbox"/> Bloc 2 SISR – Administration des systèmes et des réseaux <input type="checkbox"/> Bloc 3 SLAM – Cybersécurité des services informatiques
Présentation	Ce document apporte la compréhension et l'utilisation des commandes GIT
Savoirs	Gestion de version
Compétences	Pour certains types de ressources : labo, exolab
Transversalité	SLAM/SISR
Prérequis	
Outils	git
Mots-clés	git, tutoriel, utilisation, commandes
Durée	Indicative et non obligatoire
Auteur·e·s	David ROUMANET
Version	v 13
Date de publication	15 Septembre 2020

SOMMAIRE

A Gestion des fonctionnalités avec git.....	4
1 Objectifs.....	4
2 Compréhension du fonctionnement.....	4
2.1 Fonctionnement.....	4
2.2 Schéma de flux vers un serveur distant.....	4
B Mise en pratique.....	6
1 Première partie Anna.....	7
1.1 Création de la branche Anna-UI.....	7
1.2 Ajout d'un fichier de style.....	7
1.3 Envoi vers le serveur.....	8
2 Première partie Bob.....	9
2.1 Créer la branche Bob-bugJS.....	9
2.2 Corriger le script.....	9
2.3 Pousser la modification sur le serveur.....	9
2.4 Comparaison des branches.....	10
3 Seconde partie Anna.....	11
3.1 Édition et publication du fichier index.html.....	11
3.2 Fusion avec la branche master.....	11
4 Seconde partie Bob.....	11
4.1 Édition et publication du fichier index.html.....	11
4.2 Fusion avec la branche master.....	12

A GESTION DES FONCTIONNALITÉS AVEC GIT

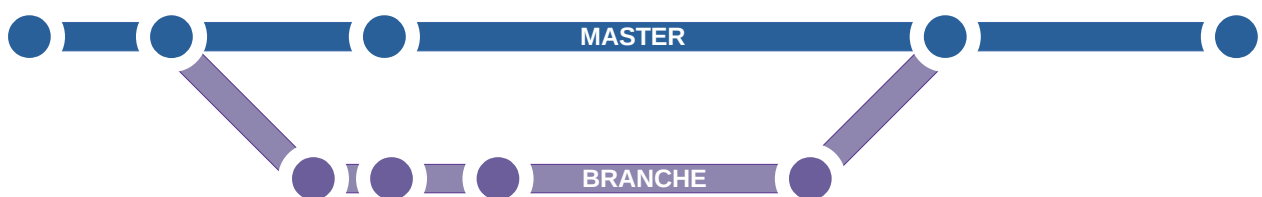
1 OBJECTIFS

L'utilisation simple de GIT étant acquise, nous allons étendre nos compétences par l'utilisation des notions de branches et de fusion, ainsi que la résolution de conflits (lors GIT ne peut arbitrer la fusion de modifications).

2 COMPRÉHENSION DU FONCTIONNEMENT

Il existe deux cas de figures où les branches sont utiles :

- Lors du développement d'une évolution d'une application ou d'un site web.
- Lors de la correction d'un bogue sur une fonctionnalité.



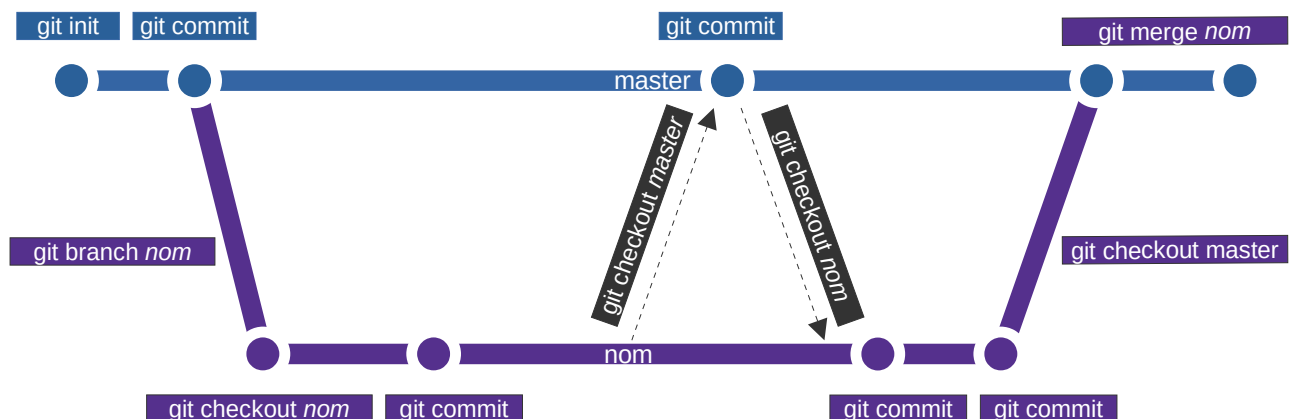
2.1 Fonctionnement

La branche principale (elle s'appelle presque toujours "master" mais ce n'est qu'une convention) contient les versions officielles en production d'une application ou d'un site web. Les évolutions principales sont celles que l'utilisateur final pourra accéder : cependant, il n'est pas imaginable qu'une fonction sur cette branche soit instable ou boguée pendant le développement.

En utilisant une branche, GIT permet de limiter l'impact du développement d'un patch ou la mise à jour d'une fonction, par la création de branches. Les développeurs peuvent désormais travailler sur une partie du code, sans modifier le code principal, jusqu'à ce que son propre code puisse être intégré dans la branche principale (le master).

2.2 Schéma de flux vers un serveur distant

La création, le passage d'une branche à une autre et enfin la fusion d'une branche respectent le flux suivant :



1. Création d'une branche, en utilisant la commande **git branch** avec comme paramètre, le nom de la branche.

2. La commande **git checkout** permet de se déplacer vers une branche.
3. Pour fermer une branche, il faut retourner sur la branche finale utiliser la commande **git merge** suivi de la branche vers laquelle on souhaite effectuer la fusion.

Ce flux est relativement simple.



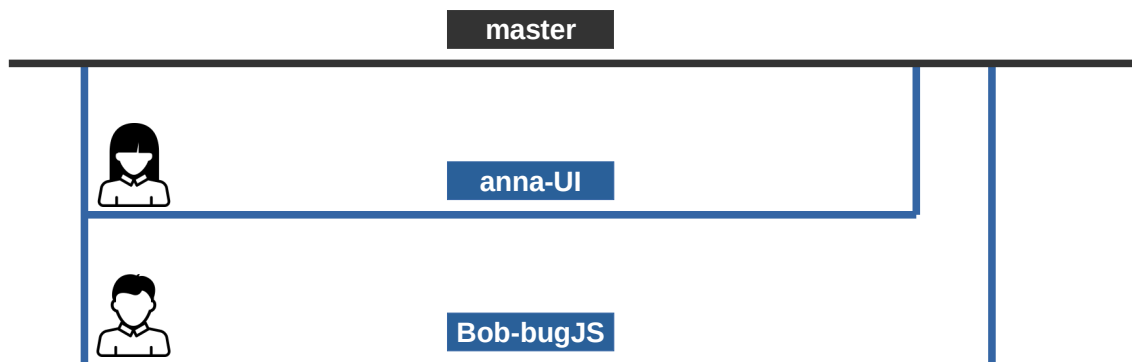
Git checkout ne permet que de changer de branche : en aucun cas, les fichiers modifiés d'une branche ne sont copiés/fusionnés/intégrés vers la branche de destination.



Comme les branches évoluent séparément, il peut y avoir des fichiers communs modifiés sur chacune d'elle. La difficulté est alors de réussir la fusion des modifications.

B MISE EN PRATIQUE

Anna et Bob vont maintenant travailler sur des branches, afin d'éviter d'éditer le même fichier en même temps : cela évitera que Bob ne supprime une partie du travail d'Anna.



Nous tenterons de montrer que la fusion de branche se fait facilement (GIT essaye en général d'intégrer seul, les modifications).

1 PREMIÈRE PARTIE ANNA

1.1 Création de la branche Anna-UI

Conformément à ce qu'ils ont décidé (bonne pratique), Anna va créer sa branche.

Son premier réflexe est de... synchroniser son projet !

```
git pull
```

Effectivement, il y avait les modifications de Bob, sur la branche principale (par exemple, 2 lignes ajoutées et une ligne supprimée dans le fichier index.html).

```
From https://github.com/droumanet/Anna_Test
* branch      master      -> FETCH_HEAD
Merge made by the 'recursive' strategy.
index.html | 3 ++-
index.js   | 19 ++++++
2 files changed, 21 insertions(+), 1 deletion(-)
```

Elle peut maintenant créer sa branche à l'aide des deux commandes :

```
git branch Anna-UI
git checkout Anna-UI
```

1.2 Ajout d'un fichier de style

Anna veut modifier l'esthétique de la page, en ajoutant un fichier CSS dans son répertoire de travail.

Dans le répertoire Anna_Test, elle va créer le fichier (style.css) :

style.css

```
/* Code By Webdevtrick ( https://webdevtrick.com ) */
body {
    background: #212121;
}
.clock {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translateX(-50%) translateY(-50%);
    color: #ff4747;
    font-size: 60px;
    font-family: 'Righteous', cursive;
    letter-spacing: 7px;
}
```

Ce nouveau fichier doit être ajouté à la base locale GIT, puis versionné

```
git add style.css
git commit -am "mise en place du style par Anna"
```

```
[Anna-UI dbce3a3] Ajout du style par Anna
1 file changed, 1 insertion(+)
```


1.3 Envoi vers le serveur




Anna synchronise avec le serveur :

```
git push origin Anna-UI
```


```
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 4 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (8/8), 1024 bytes | 512.00 KiB/s, done.
Total 8 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 2 local objects.
remote:
remote: Create a pull request for 'Anna-UI' on GitHub by visiting:
remote:   https://github.com/droumanet/Anna_Test/pull/new/Anna-UI
remote:
To https://github.com/droumanet/Anna_Test.git
* [new branch]      Anna-UI -> Anna-UI
```





Anna vérifie sur le serveur que sa branche a bien été créée :

 Anna-UI had recent pushes 2 minutes ago

 Anna-UI ▾  3 branches  0 tags

This branch is 3 commits ahead of master.

 droumanet Ajout du style par Anna

 index.html	Ajout du script par Bob
 index.js	Ajout du script par Bob
 readme.md	modification des fichiers
 style.css	Ajout du style par Anna

2 PREMIÈRE PARTIE BOB

Bob découvre que son script ne fonctionne pas : en recopiant un script existant et en le modifiant, il a laissé une erreur.

2.1 Créer la branche Bob-bugJS

Bob peut utiliser les deux commandes qu'Anna a utilisé, mais il connaît un raccourci (grâce à l'option -b de la sous-commande checkout) :

```
git checkout -b Bob-bugJS
```

```
Cloning into 'Anna_Test'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 238 bytes | 2.00 KiB/s, done.
```

2.2 Corriger le script

Si vous n'avez pas trouvé l'erreur sur le script de l'exploration précédente, il faut corriger cette ligne (qui contient une référence à une variable 'session' non-définie) :

```
var time = h + ":" + m + ":" + s;
```

Enregistrez la modification (testez l'affichage de la page correcte).

Enfin, enregistrez la modification de version avec la bonne commande *git commit* (consultez l'exploration précédente si vous avez oublié).

```
[Bob-bugJS 1aec3df] Correction code JS (suppression session)
1 file changed, 1 insertion(+), 1 deletion(-)
```

2.3 Pousser la modification sur le serveur

Bob a validé son script, il envoie les modifications sur le serveur avec la commande suivante :

```
git push origin Bob-bugJS
```

```
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 364 bytes | 182.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'Bob-bugJS' on GitHub by visiting:
remote:   https://github.com/droumanet/Anna_Test/pull/new/Bob-bugJS
remote:
To https://github.com/droumanet/Anna_Test
* [new branch]      Bob-bugJS -> Bob-bugJS
```











Il constate que la branche est bien créée et que sur les 3 fichiers, un seul avait un écart : tout c'est bien passé. N'oubliez pas que GIT utilise une base de données (les changements sont des écarts entre un état de départ et l'état actuel).



Aviez-vous pensé à faire un git pull pour Bob avant changements ? Cela doit devenir un réflexe !

2.4 Comparaison des branches

Anna et Bob travaillent sur des branches différentes. La configuration actuelle est la suivante :

master	 index.html	Ajout du script par Bob
	 index.js	Ajout du script par Bob
	 readme.md	modification des fichiers
Anna-UI	 index.html	Ajout du script par Bob
	 index.js	Ajout du script par Bob
	 readme.md	modification des fichiers
	 style.css	Ajout du style par Anna
Bob-bugJS	 index.html	Ajout du script par Bob
	 index.js	Correction code JS (suppression session)
	 readme.md	modification des fichiers

Anna doit opérer une dernière modification sur le fichier index.html (ajouter l'emplacement du style CSS).

Bob va également modifier le fichier index.html, pour remettre le titre qu'il avait effacé... comment GIT va réagir ?

3 SECONDE PARTIE ANNA

Il n'est pas nécessaire de faire un git pull sur la branche d'Anna-UI, personne n'a travaillé dessus, mais cela reste un réflexe à avoir.

3.1 Édition et publication du fichier index.html

Ajoutez la ligne suivante dans la partie <head> du fichier.

Index.html

```
<link rel="stylesheet" href="./style.css">
```

Sauvegardez, commitez et poussez votre modification.

3.2 Fusion avec la branche master

Maintenant, Anna est satisfaite de son travail (*ok, pour cette exploration, c'est un peu léger... mais souvenez-vous que ce qui nous intéresse maintenant, c'est de voir comment toutes les modifications vont ré-intégrer la branche master*) et fusionne son travail sur la branche principale.

Elle doit retourner sur la branche principale, puis effectuer la fusion avec la commande git merge.

```
git checkout master  
git merge Anna-UI
```

```
Updating d9bfa96..01b4f49  
Fast-forward  
 index.html | 1 +  
 style.css  | 1 +  
 2 files changed, 2 insertions(+)
```

La branche master reçoit les corrections de la branche Anna-UI. Les modifications sont locales (d'où l'importance de faire un git pull avant de travailler), il faut maintenant synchroniser le serveur :

```
git push origin master
```

Le serveur est à jour (vérifiez sur le vôtre).

4 SECONDE PARTIE BOB

4.1 Édition et publication du fichier index.html

Bob va simplement remplacer la ligne de titre manquante dans la section <body> du fichier :

index.html

```
<h1>Projet Anna et Bob</h1>
```

Si Bob a un doute sur la branche, sur laquelle il travaille, il peut utiliser la commande :

```
git branch
```

```
* Bob-bugJS  
master
```

Un petit commit et un petit push pour valider la branche sur le serveur, vous savez quoi faire...

4.2 Fusion avec la branche master

L'heure délicate de la fusion a sonnée : Bob ne sait pas qu'Anna a modifiée le fichier index.html.

```
git checkout master
```

Surprise, GIT constate l'oubli de synchronisation de Bob !

```
Switched to branch 'master'  
Your branch is behind 'origin/master' by 4 commits, and can be fast-forwarded.  
(use "git pull" to update your local branch)
```

En suivant les instructions données par GIT (utiliser la commande git pull), Bob obtient localement les changements présents sur le serveur.

```
Updating cf5c339..01b4f49  
Fast-forward  
index.html | 1 +  
style.css | 14 ++++++  
2 files changed, 15 insertions(+)  
create mode 100644 style.css
```

Le fichier style.css est maintenant présent dans l'espace de travail de Bob. La fusion va s'effectuer localement...

```
git merge Bob-bugJS
```

Résultat :





```
Auto-merging index.html  
Merge made by the 'recursive' strategy.  
index.html | 1 +  
index.js | 2 +-  
2 files changed, 2 insertions(+), 1 deletion(-)
```

Voici le fichier index.html sur le disque de Bob :

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
  <meta charset="UTF-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <title>Document</title>  
  <link rel="stylesheet" href="./style.css">  
</head>  
<body>  
  <h1>Projet Anna et Bob</h1>  
  <div id="DigitalCLOCK" class="clock" onload="showTime()"></div>  
  <script src="index.js"></script>  
</body>  
</html>
```

Bob teste le projet et finit par envoyer les modifications de la branche master locale, sur le serveur :

```
git push origin master
```

 index.html	Merge branch 'Bob-bugJS' into master
 index.js	Correction code JS (suppression session)
 readme.md	modification des fichiers
 style.css	Ajout du style par Anna