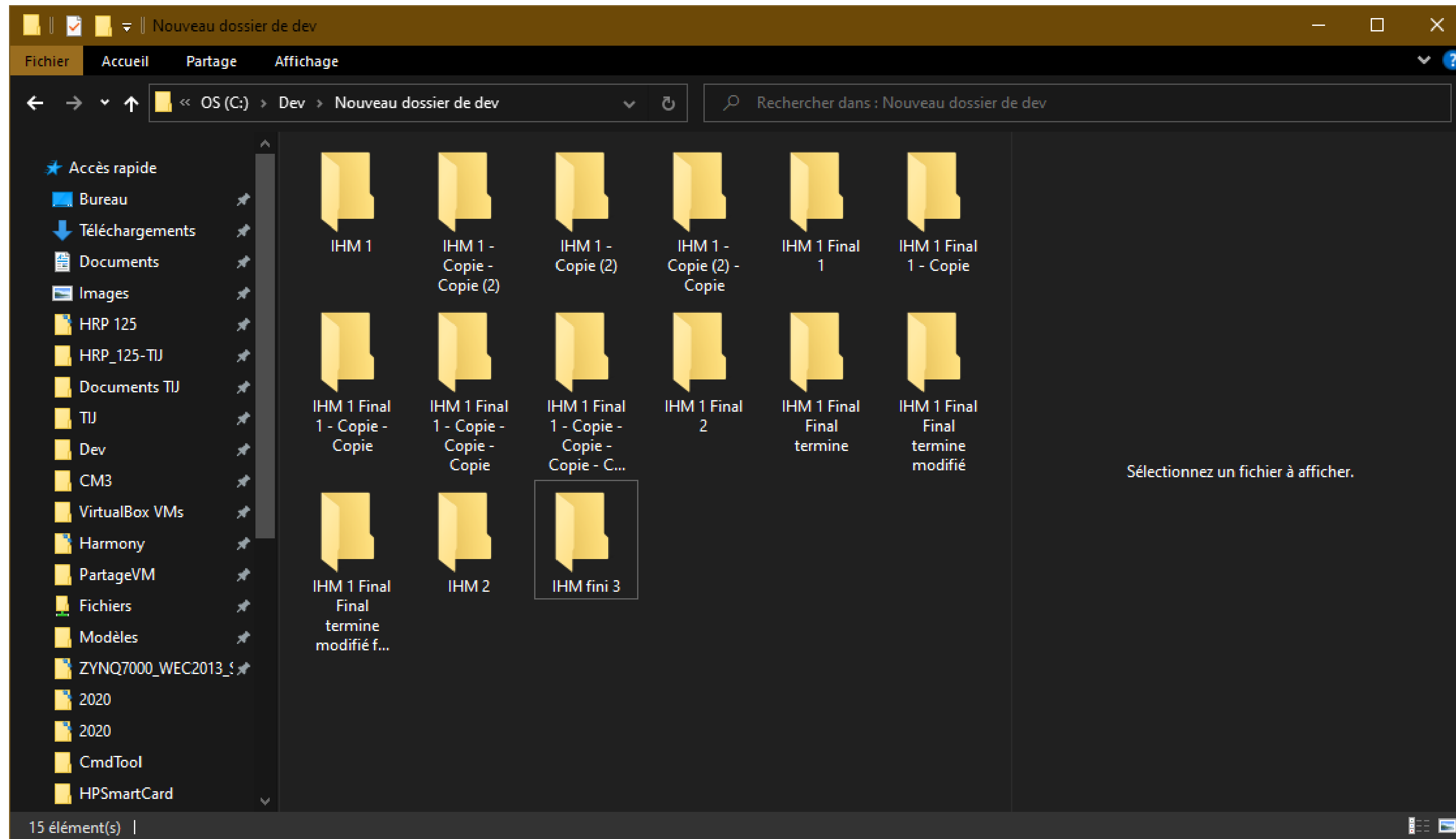


GIT / SVN

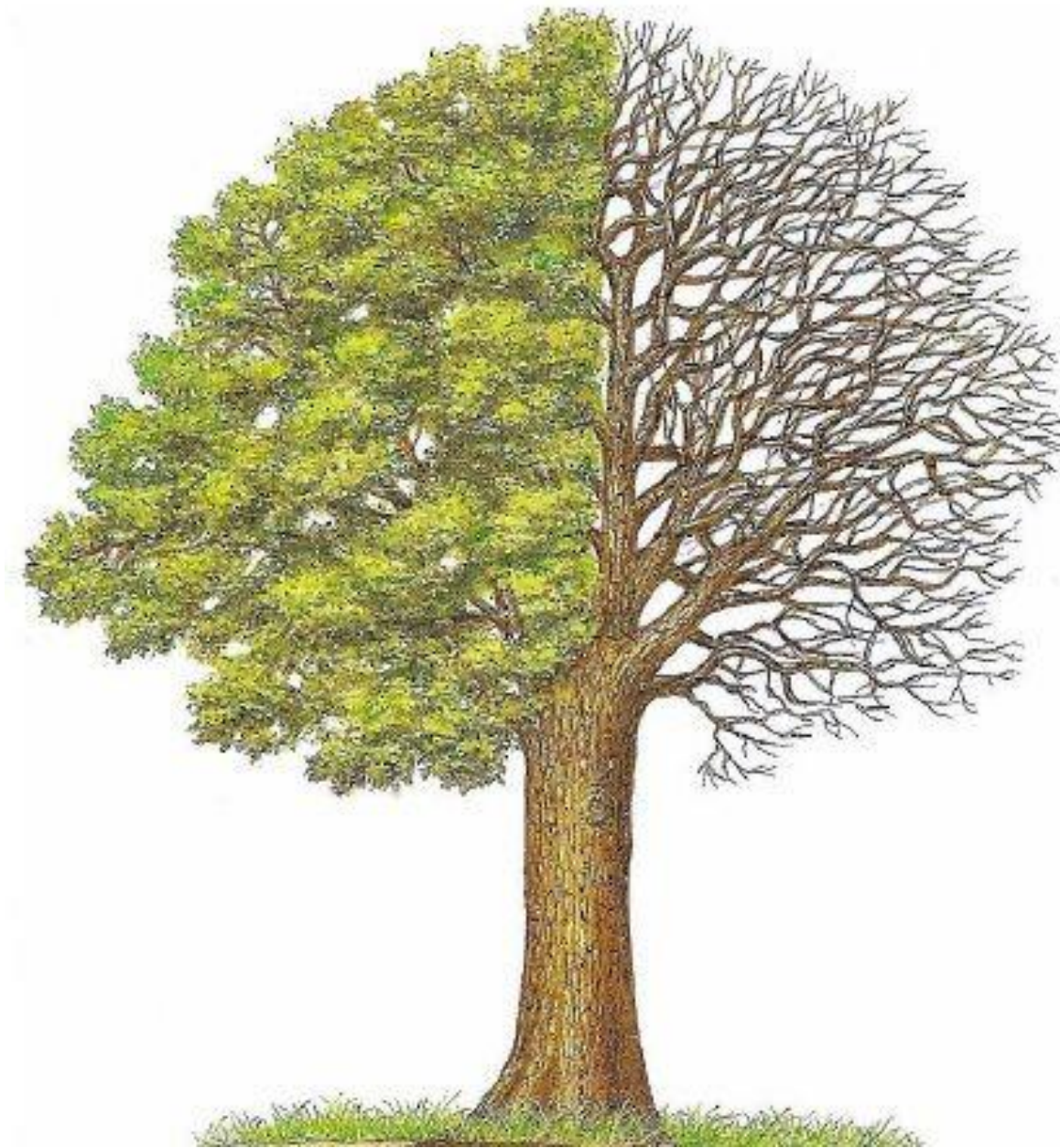
Le match

- Le terrain
- Les joueurs
- Le match
- Le debrief

Echauffement



Echauffement



Terme anglais	Traduction
Tree	Arbre
Trunk	Tronc
Branch	Branche
Tag	Etiquette
Release	Version
Merge	Fusion
Merge Request	Demande de fusion
Commit	Soumettre
Push	Pousser/Envoyer
Checkout	Télécharger les fichiers
Pull	Récupérer
Repository (Repo)	Dépôt (base de donnée)
Diff (Change view)	Evaluation des modifications entre 2 commit/versions

Le terrain



Fonction avec bug.cpp



Fonction.cpp

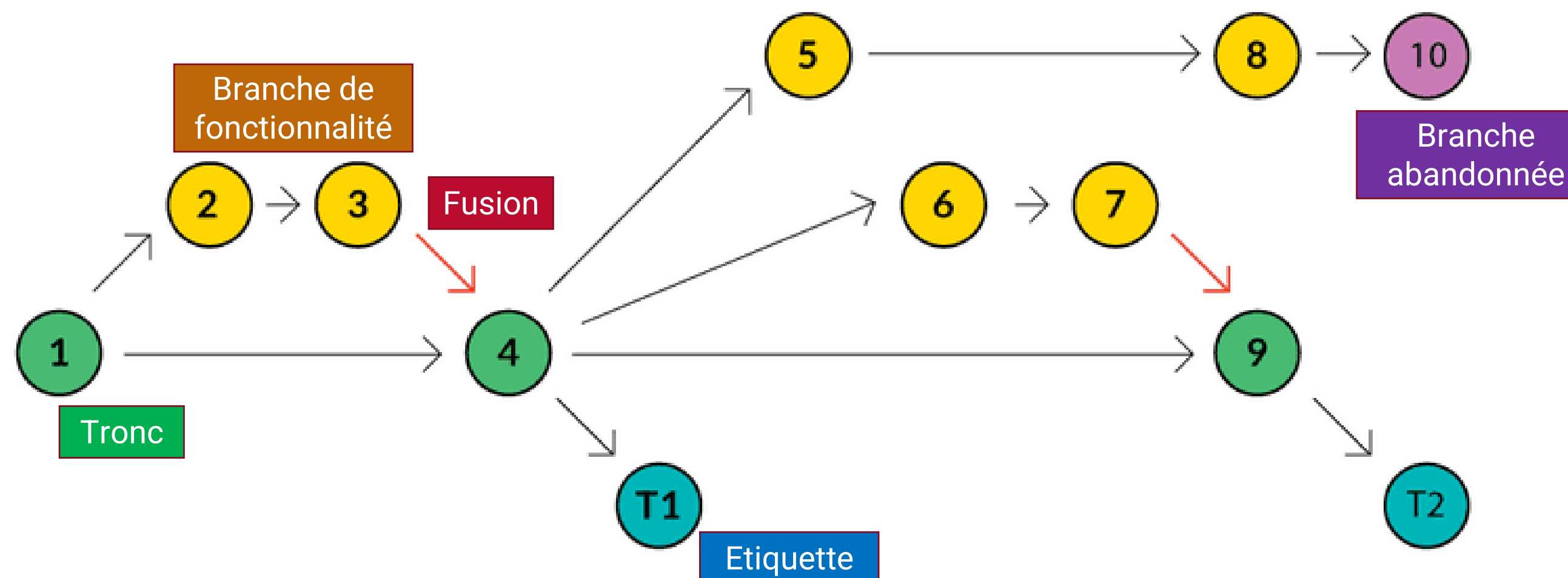


Le dossier du code



Envoi à Mr Client

What is "version control"?



Le code d'un projet, d'une app ou d'un composant logiciel est généralement organisé en une **structure de dossiers** ou « arborescence de fichiers ». Un **développeur** de l'équipe peut travailler sur une nouvelle **fonctionnalité**, tandis qu'un autre corrige un bug sans rapport en changeant le code. Chaque **développeur** peut apporter ses **changements** dans plusieurs parties de l'arborescence de fichiers.

Le **contrôle de version** aide les équipes à résoudre ce genre de problèmes, en **suivant** chaque **changement individuel** de chaque contributeur et en aidant à **prévenir les conflits** entre les tâches concomitantes. Les changements apportés à une partie du logiciel peuvent être incompatibles avec ceux apportés par un autre développeur travaillant en même temps.

Les Joueurs SVN



Fonction avec bug.cpp



Le dossier du code



Serveur



Récupère le contenu de la branche release



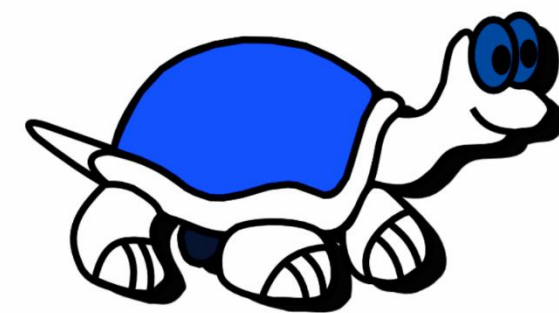
Envoi à Mr Client



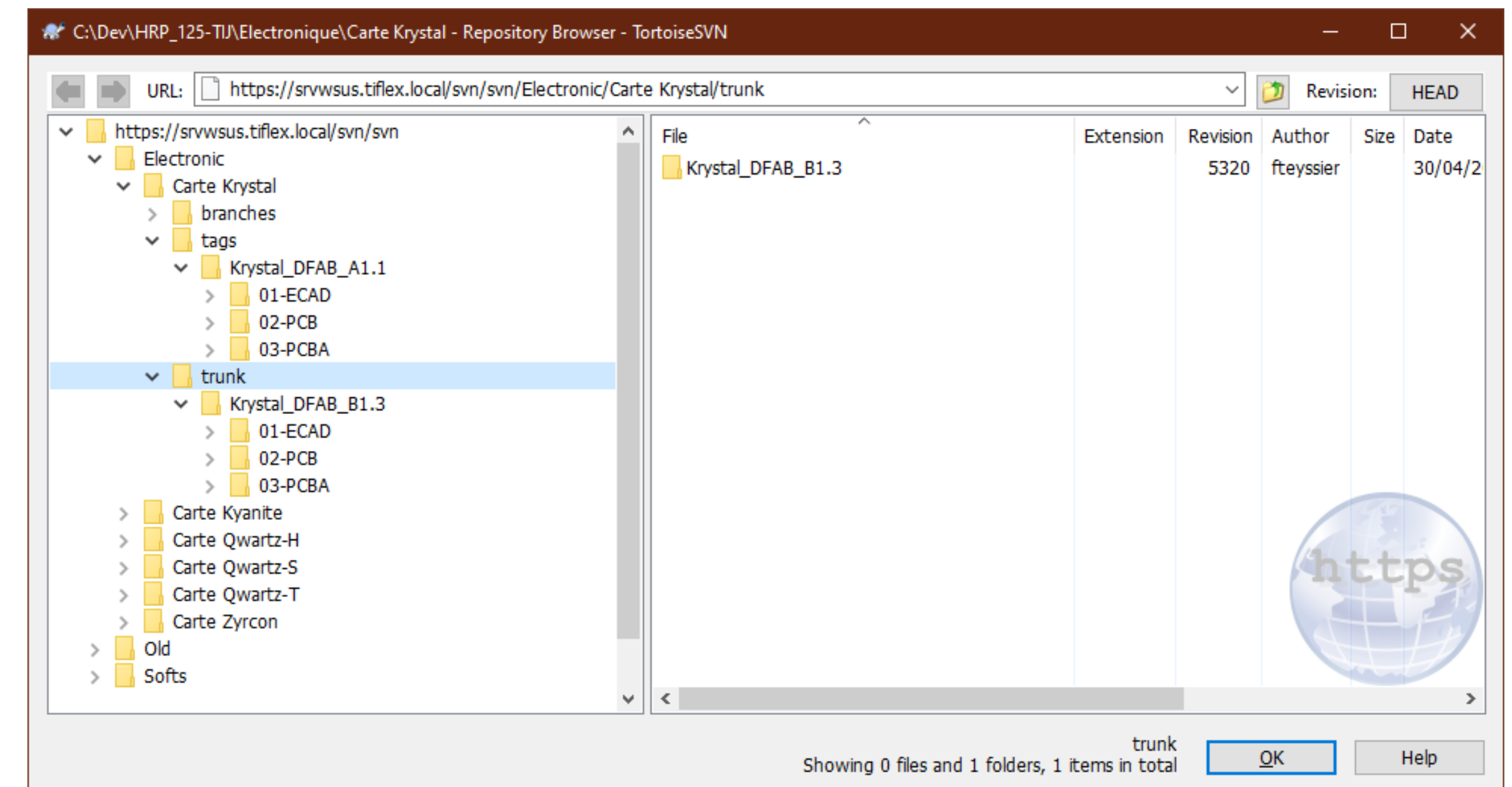
Fonction.cpp



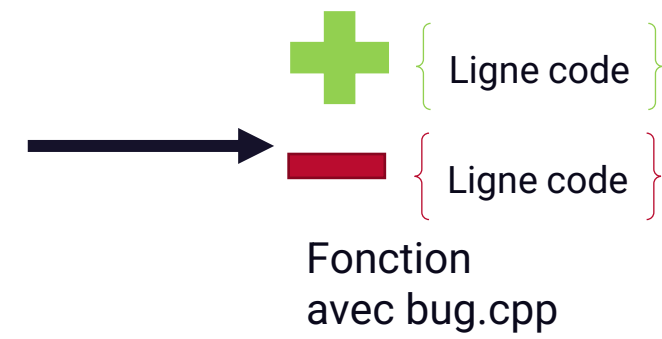
Le dossier du code



TortoiseSVN



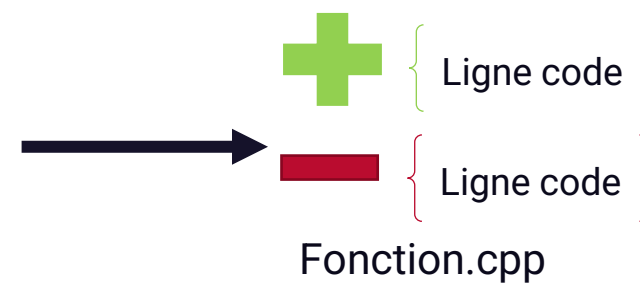
Les Joueurs GIT



Récupère la dernière release



Envoi à Mr Client



Description	Date	Auteur	Validation
test ok+ doc	21 juin 2021 03:05	Franck <teyssier.f>	ba3981b
ajout build	8 juin 2021 12:50	Franck <teyssier.fr>	c5a899b
Modif readme	8 juin 2021 11:13	Franck <teyssier.fr>	69ffa9d
Merge branch 'master' of https://gitlab.com/dedounet/dotnet-core-example	8 juin 2021 11:08	Franck <teyssier.fr>	aaa1053
augmentation des features	8 juin 2021 11:07	Franck <teyssier.fr>	01db42e
Update README.md	8 juin 2021 10:58	dedounet <teyssie>	14acf5a
Initial commit	8 juin 2021 09:39	Franck <teyssier.fr>	18af680

```
Validation : c5a899b704cad05e45970ad3af8a486e670ccfb0 [c5a899b]
Parents : 69ffa9d26c
Auteur : Franck <teyssier.franck@gmail.com>
Date : mardi 8 juin 2021 12:50:44
Auteur : Franck

ajout build

.gitlab-ci.yml
README.md
MyProject/Program.cs
```

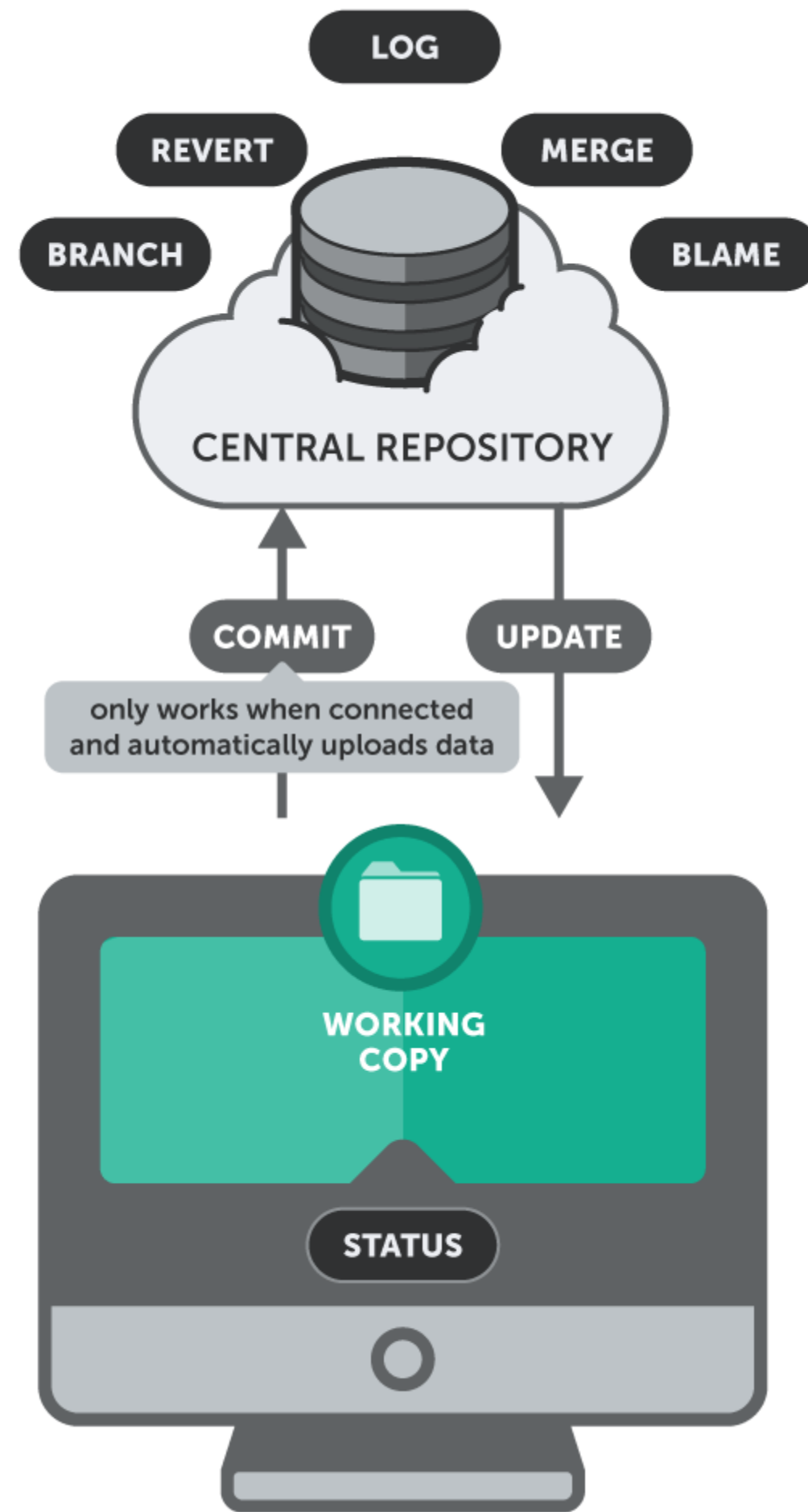
```
17 17  ...
18 18  ...
19 19  ...
20 20  ...
21 21  +--- build
22 22  +--- release
23 23  ...
24 24  ...
25 25  ...
26 26  +--- master
27 27  +--- release
28 28  ...
29 29  ...
```



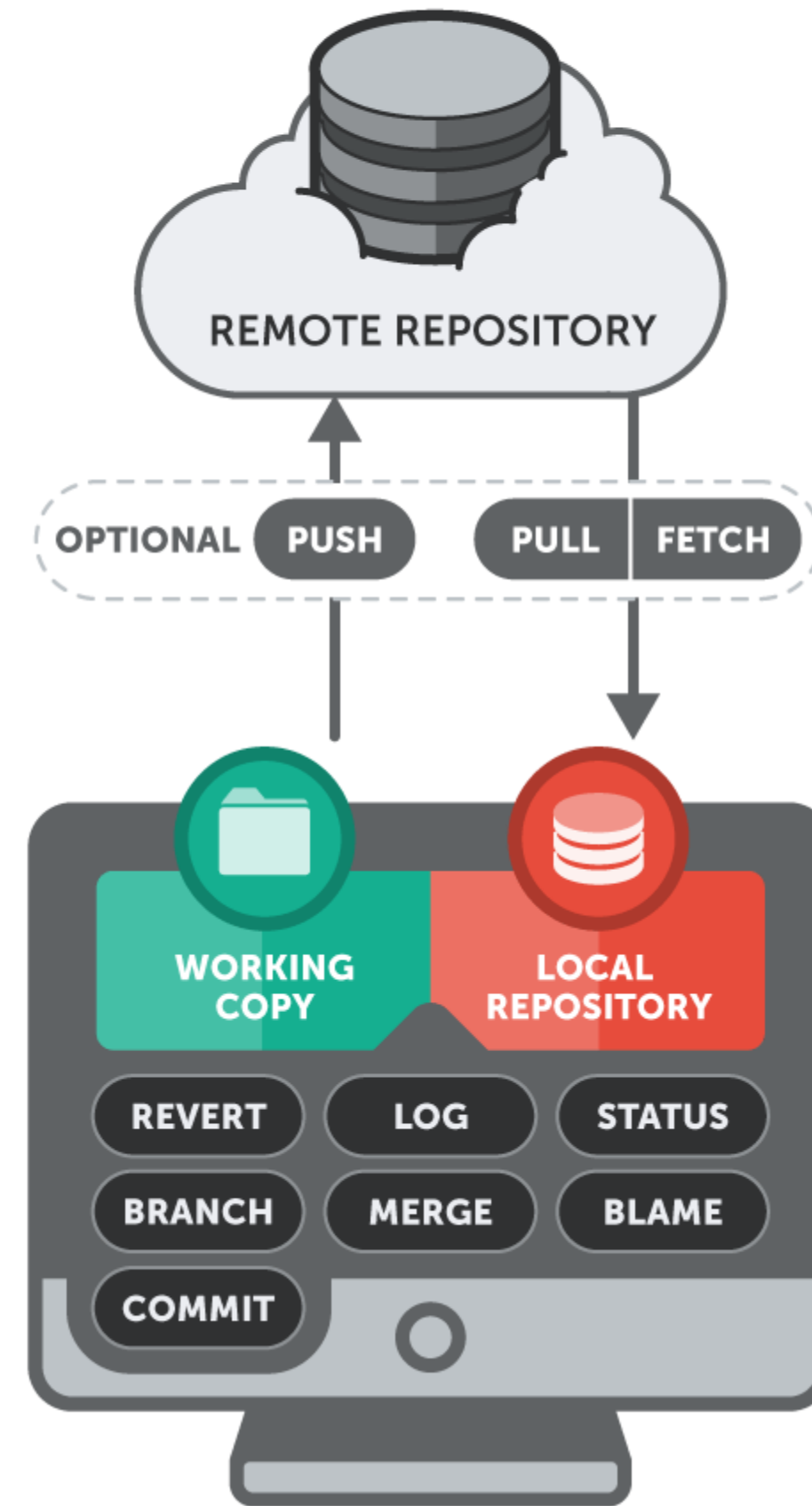
Coup d'envoi !

Le match 1^{ère} mi-temps

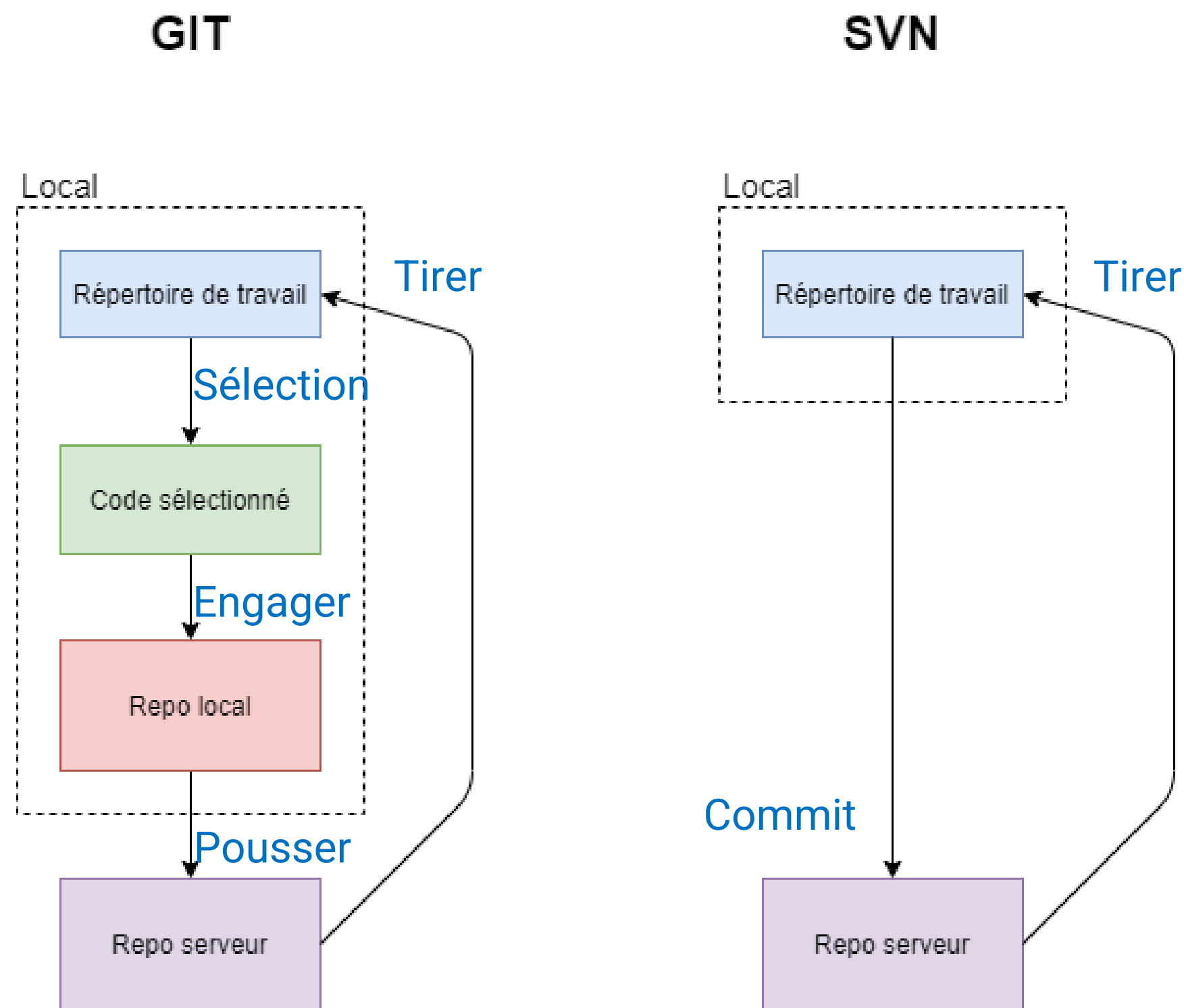
SUBVERSION



GIT



Le match 2^{ème} mi-temps



	SVN	GIT
Versionning	Centralisé	Décentralisé
Dépôt	Un dépôt central dans lequel les copies de travail sont créées	Des copies de dépôt, présentes localement, dans lesquelles il est possible de travailler
Droit d'accès	Basé sur le chemin	Pour le répertoire complet
Suivi des modifications	Enregistre des données	Enregistre des données
Journal de modifications des données	Complet seulement dans le dépôt. Les copies de travail ne contiennent que la version la plus récente.	Le dépôt et les copies de travail contiennent l'historique complet.
Connectivité au réseau	Pour tous les accès	Nécessaire seulement pour réaliser une synchronisation
Vitesse de transfert	Plus rapide pour des fichiers binaires volumineux	Très rapide pour des fichiers textes standards, moins pour des binaires très lourds
Granularité	Sauvegarde par arborescence du dossier Gestion des dossiers vides	Sélection des lignes de codes unitaires ou par fichier Pas de gestion des répertoires vides
Exclusions	Par sélection contextuelle	Par fichier .gitignore

Changement de joueur GITLAB

The screenshot shows the GitLab interface for the 'dotnet-core-example' repository. The left sidebar contains navigation options like Project information, Repository, Files, Commits, Branches, Tags, Contributors, Graph, Compare, Issues, Merge requests, CI/CD, Security & Compliance, Deployments, Monitor, Infrastructure, Packages & Registries, Analytics, Wiki, Snippets, and Settings. The main content area displays the repository path 'dedounet > dotnet-core-example > Repository' and a commit history table. Below the table, the README.md file is open, showing a project description and build status.

Name	Last commit	Last update
MyProject.Test	augmentation des features	1 week ago
MyProject	ajout build	1 week ago
.gitattributes	Initial commit	1 week ago
.gitignore	augmentation des features	1 week ago
.gitlab-ci.yml	ajout build	1 week ago
MyProject.sln	augmentation des features	1 week ago
README.md	test ok+ doc	1 hour ago

README.md

Un exemple de projet automatisé avec .Net Core

pipeline passed coverage 50.00%

Le build est réalisé quand un push est fait dans la branche "master" puis des tests sont réalisés via Junit. Les artefacts sont les résultats des tests et les fichiers compilés visibles dans les Jobs.

Le résultat de compilation est publié lors d'une release via les artefacts de compilation.

La qualité du code est évaluée via codeclimate à la demande via l'interface de pipeline

The screenshot shows the diff view of the '.gitlab-ci.yml' file in the 'dotnet-core-example' repository. The interface highlights changes between two versions of the file. The diff shows modifications to the 'stages', 'release', 'build', and 'debug' sections. The 'release' section now includes a 'stage: release' and a 'publish' job. The 'build' section includes a 'stage: build' and a 'script' block with 'dotnet build'. The 'debug' section includes a 'before_script' block with 'echo | dotnet --version'.

```
... .. @@ -17,12 +17,13 @@ image: mcr.microsoft.com/dotnet/core/sdk:2.2.301
17 17
18 18     stages:
19 19     - test
20 20     - deploy
21 21     + build
22 22     + release
23 23     release:
24 24     - stage: deploy
25 25     + stage: release
26 26     only:
27 27     - master
28 28     + release
29 29     artifacts:
30 30     paths:
31 31     - publish/
32 32     @@ -30,6 +31,11 @@ release:
33 33     # The output path is relative to the position of the csproj-file
34 34     - dotnet publish -c Release -o ../publish MyProject/MyProject.csproj
35 35
36 36     + build:
37 37     +   stage: build
38 38     +   script:
39 39     +     - dotnet build
40 40
41 41     debug:
42 42     before_script:
43 43     - 'echo | dotnet --version' # must be v 2.2.300 or later for reporter tool to work
44 44     @@ -70,7 +76,6 @@ code_quality:
45 45     --volume /var/run/docker.sock:/var/run/docker.sock
46 46     --volume /tmp/cc:/tmp/cc
47 47     codeclimate/codeclimate --url https://codeclimate.com
```


Arrêts de jeu

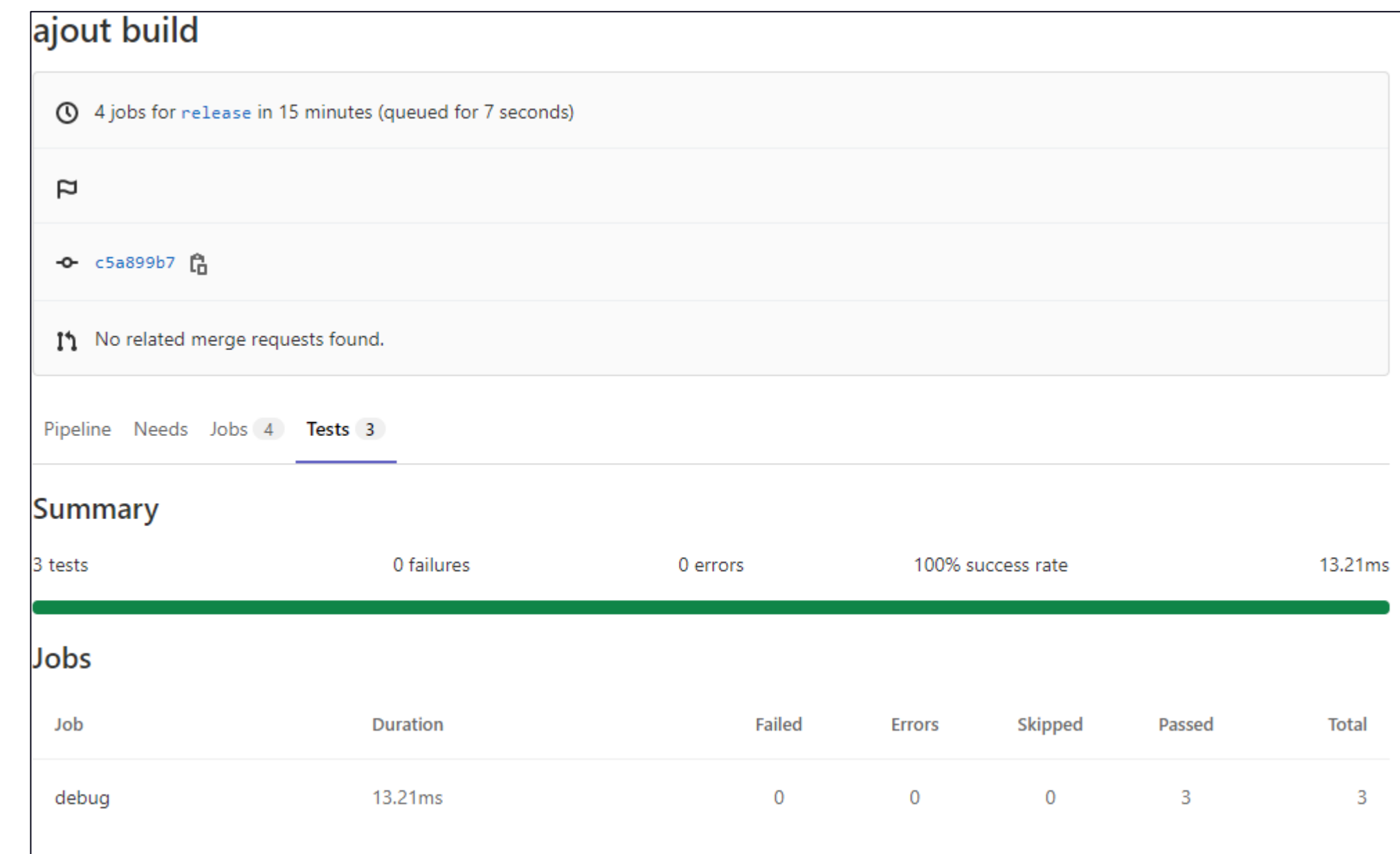
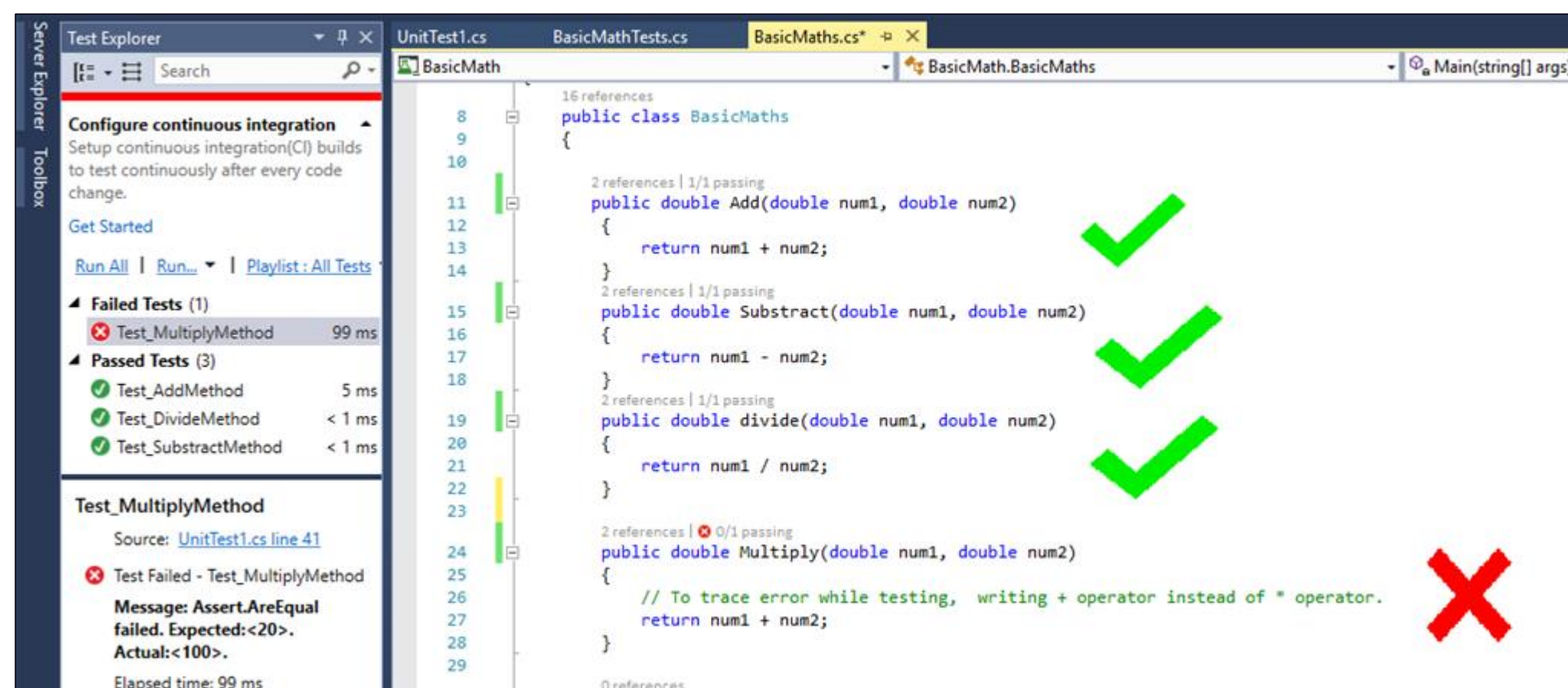
Tests unitaires et validation du code



Mise en place d'un « pipeline » (processus) en fonction des caractéristiques (eg: push sur branche release, ajout de tag, variable) d'un commit :

- Tests unitaires + couverture
- Qualité du code + rapport
- Compilation
- Mise à dispo du résultat de compilation (release)
- Documentation automatisée

Fermeture automatique de tickets correspondants



Actions choisies

Choisir les fichiers nécessaires
UNIQUEMENT

Réaliser ses « commit » sur des
fonctions simples / bug uniques

Respecter les branches

DOCUMENTER SES COMMIT

Economiser l'espace, Transférer
uniquement le nécessaire

Avoir une bonne granularité
de choix

Pour la CLARTE des explications:
Le code/fichiers montre ce qui a
changé, les messages de commit
expliquent POURQUOI

Penses bêtes

<https://guillaumebriday.fr/comment-jutilise-git-mes-astuces-et-bonnes-pratiques>

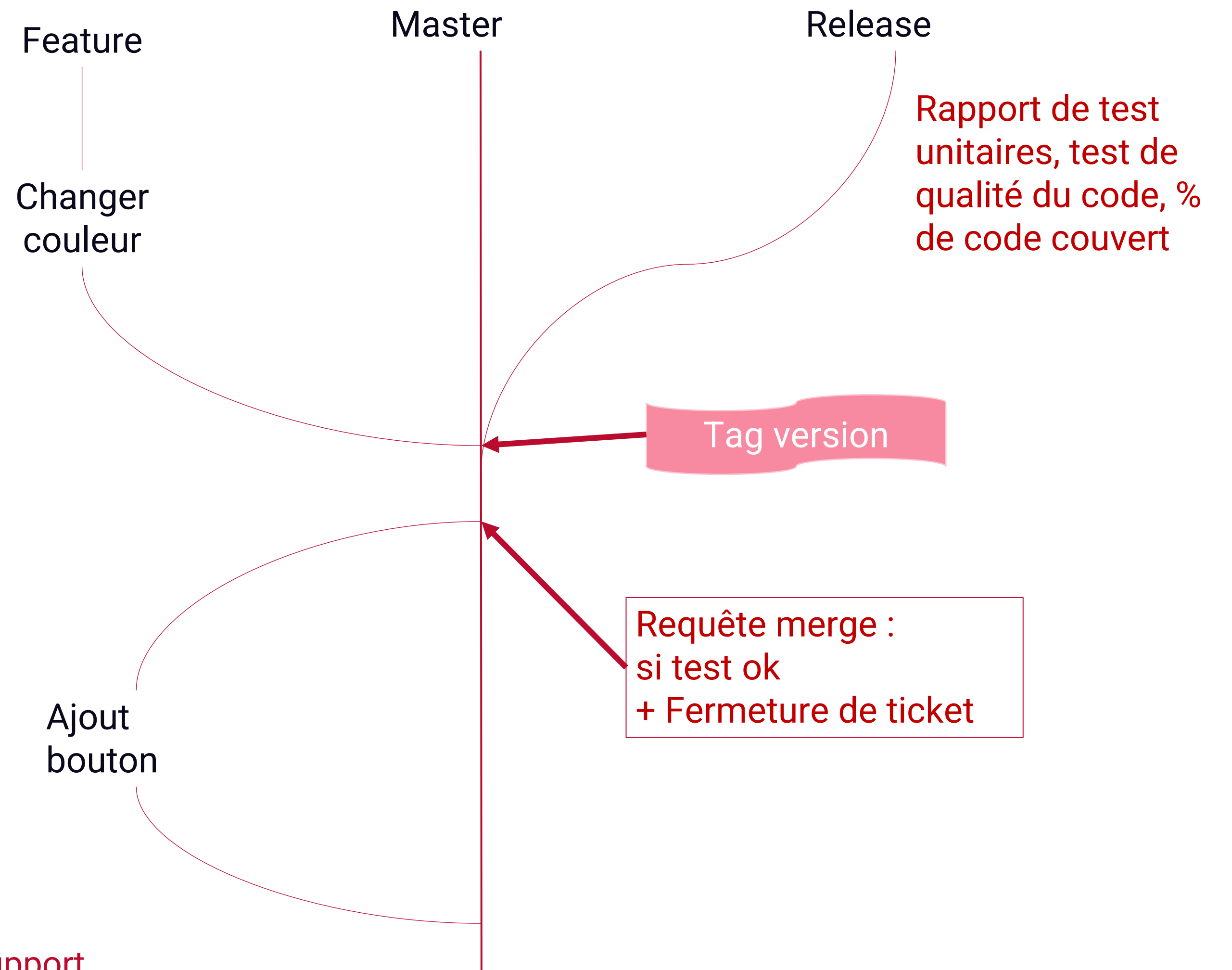
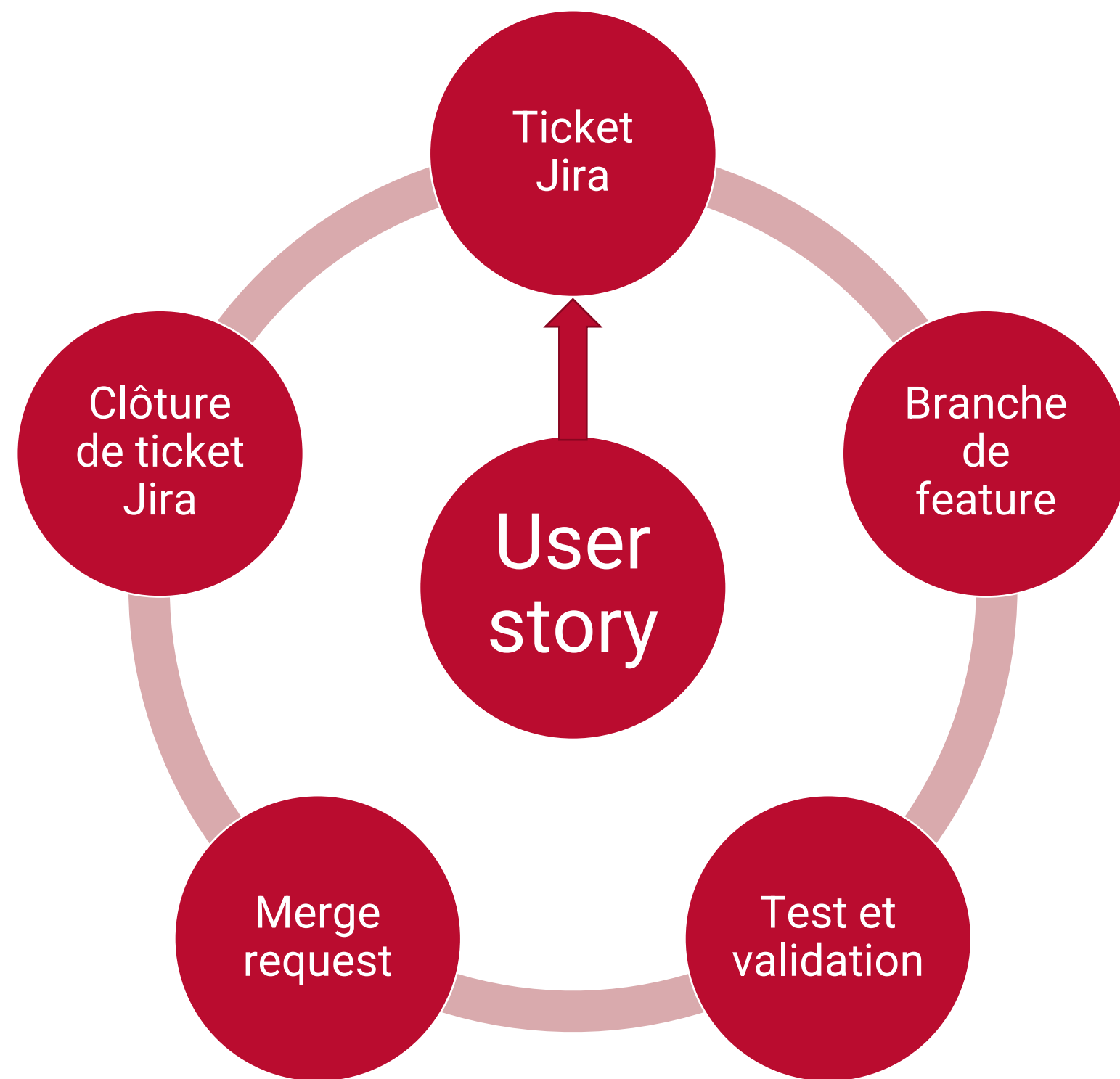
<https://git-scm.com/docs>

<https://ndpsoftware.com/git-cheatsheet.html#loc=workspace;>

Debrief

Beau jeu, merci

Processus BE



[Process issues with smart commits | Jira Software Cloud | Atlassian Support](#)