- Organisation de l’équipe Agile
- Cycle de vie d’un projet
- Gestionnaire de source
- Git
- Pull Request & revues de code
- Intégration continue
- Gestion de la dette

Organisation de l’équipe Agile

Une équipe
- travaillant sur le même projet
- développant en même temps différentes user stories lié à différentes features
- travaillant parfois sur les mêmes fichiers

L’équipe doit donc pouvoir récupérer le travail de chacun en permanence.
L’équipe doit pouvoir avoir connaissance des développements effectués par chacun des membres de l’ équipe.

Cycle de vie du projet

Un sprint représente un cycle pendant lequel l’équipe travaille sur le développement de fonctionnalité (et correctifs ou tâches techniques).
A la fin d’un sprint, on doit être en mesure de figer le projet pour le déployer pour les utilisateurs (mise en production) mais un nouveau sprint démarre et les développements continus.

Le gestionnaire de code source (Gestionnaire de versions)

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.

Il permet de faciliter la collaboration des différents membre de l'équipe sur un projet.

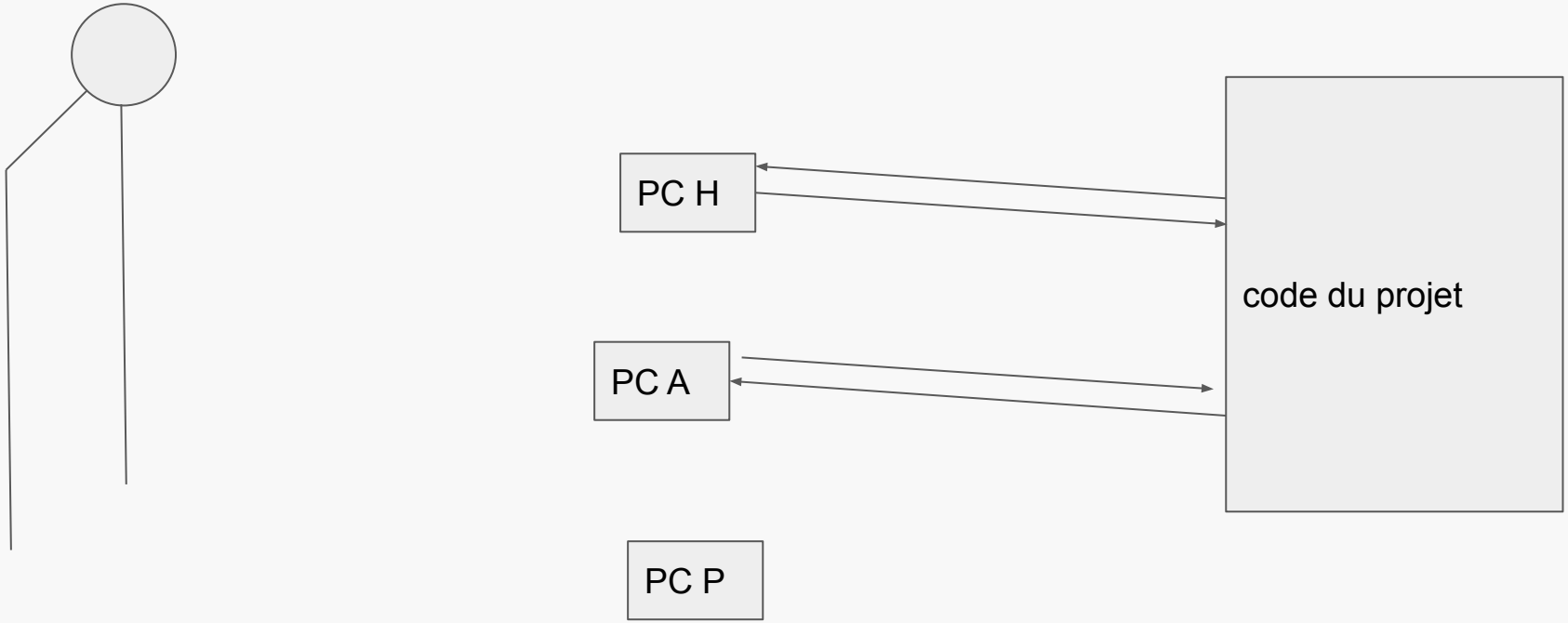
Il permet notamment d'avoir

- un historique des modifications du code
- la possibilité de gérer des versions

L'avant Git…

SVN (Subversion) est un logiciel de gestion de versions. Il s'appuie sur le principe du dépôt centralisé et unique. SVN fonctionne donc sur le mode client-serveur, avec

- un serveur informatique centralisé et unique où se trouvent les fichiers constituant la référence (le « dépôt » ou « référentiel », ou « repository » en anglais).
- des postes clients sur lesquels se trouvent les fichiers recopiés depuis le serveur, éventuellement modifiés localement depuis


PC H
PC A
PC P
code du projet

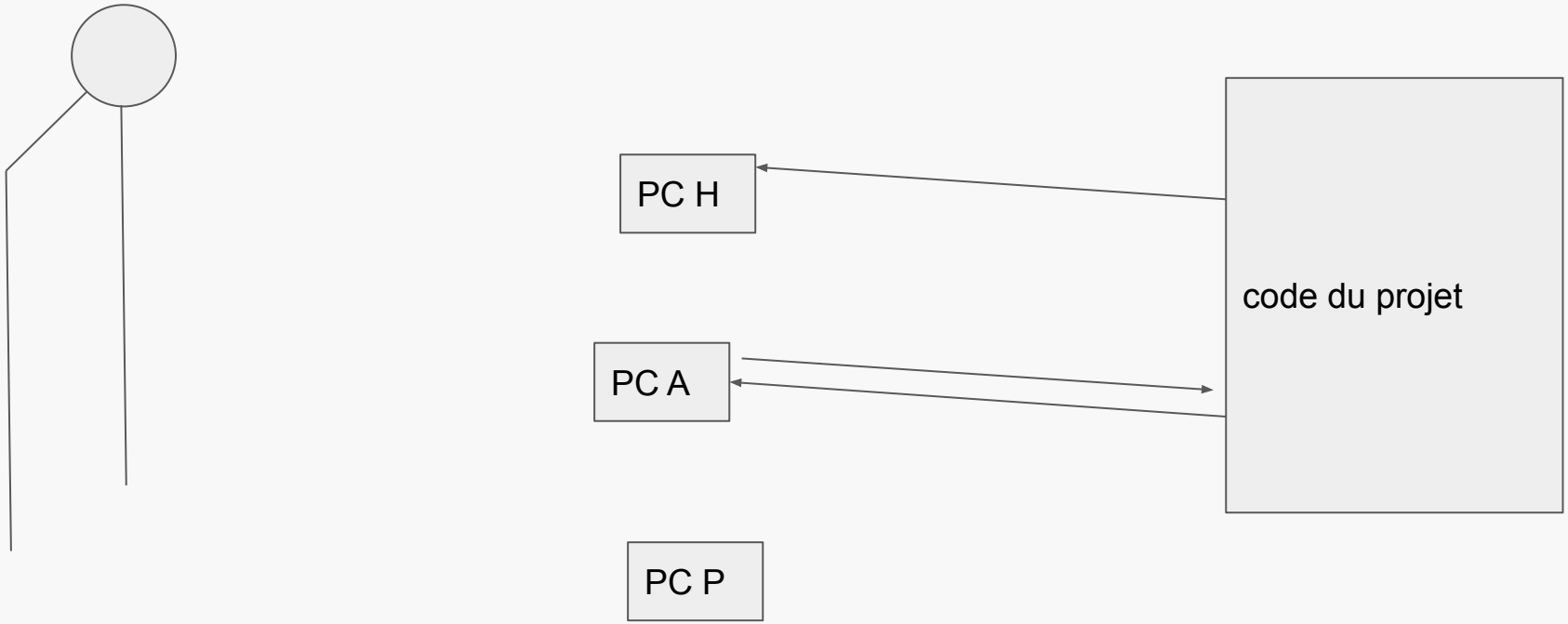
Git est un gestionnaire de version décentralisé.
Il ne repose pas sur un serveur centralisé, mais il utilise un système de connexion pair à pair. Le code informatique développé est stocké non seulement sur l'ordinateur de chaque contributeur du projet, mais il peut également l'être sur un serveur dédié.
Contrairement à SVN qui historise des versions de fichier, Git permet de gérer l'évolution d'une arborescence de modifications.

Comment ça marche ?

Le projet existe :
On récupère localement un projet (repository) distant via la commande git clone
Le projet n'existe pas encore
On crée localement le projet que l'on va déposer sur un serveur distant via la commande git push.

On travaille localement sur une branche et on effectue des commits pour organiser les modifications effectuées sur son code.
Un commit est une différence, une modification sur un ensemble de fichier.
On retrouve l'auteur et la date de modification pour chacune des différences.
Le commit est défini par un message, un auteur et une date. Enfin, pour chaque commit, une clé unique (hash) est généré ; c'est le SHA-1 du commit.

On peut ensuite à tout moment rapatrié ses commits sur le serveur distant (remote) ou récupérer localement les modifications qui y ont été apportées.


PC H
code du projet
PC A
PC P

En pratique

- Installation de git for Windows
- Principales commandes Git

- `git status` permet d'avoir un état des lieux de la branche de travail actuelle (fichiers modifiés, fichiers suivis…) ;
- `git add` ajoute de nouvelles modifications à la base des modifications suivies;
- `git commit -m "message"` permet de créer un commit à partir de modifications présentes dans la base des modifications suivies;

  message -> <type>(<scope>): <subject>

- `git init` crée un nouveau dépôt ;
- `git clone` clone un dépôt distant ;
- `git branch` liste les branches ;
- `git merge` fusionne une branche dans une autre ;
- `git rebase` déplace les commits de la branche courante devant les nouveaux commits d'une autre branche ;
- `git log` affiche la liste des commits effectués sur une branche ;
- `git push` publie les nouvelles révisions sur le *remote* ;
- `git pull` récupère les dernières modifications distantes du projet (depuis le *Remote*) et les fusionne dans la branche courante ;
- `git fetch` prendre connaissances des modifications distantes
- `git stash` stocke de côté un état non commité afin d'effectuer d'autres tâches.

Les Pull Request (PR)

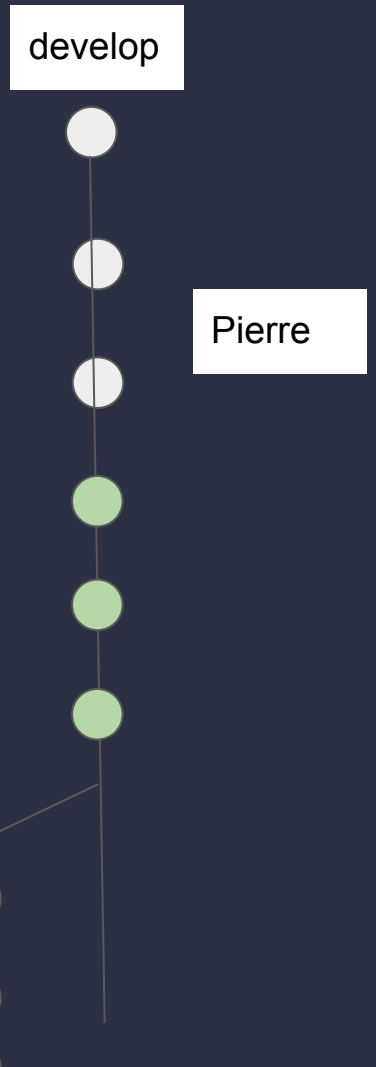
On travaille toujours sur une branche dédiée et on effectue une demande de merge dans la branche générale: une Pull Request.
La PR permet de visualiser tout code avant de le merger afin de :

- Détecter les défauts au plus tôt
- Obtenir un feedback rapide et efficace sur la qualité du code
- Détecter les écart vis-à-vis des standards de code de l'équipe
- Faire évoluer ces standards
- Renforcer la propriété collective du code et donc de diminuer les risques de spécialisations trop forte où seul l'auteur du code est capable de le modifier

Les revues de code sont des moments de partage au sein de l'équipe.

Elles peuvent avoir lieu sur certaines PR afin de:

- Faciliter l'apprentissage
- Partager l'expérience
- Améliorer la qualité des échanges



2.3.1.XXXXXXXXXXX

3.1.4.XXXXXXXXXXX

4.0.0.XXXXXXXXXXX

git clone https://github.com/foreach-academy/imm-it-web-statique.git

L’intégration continue

Il est possible de coupler au gestionnaire de version des outils permettant d’effectuer du build ou du déploiement automatique
Ces outils peuvent détecter :

- les nouvelles branches et effectuer des builds afin de valider les développement (passages des test unitaire ou d’intégrations….) afin de détecter les bug éventuel avant le merge.
- les merges sur des branches afin de relancer des déploiement automatiques.

cf. Jenkins

Il est également possible de coupler un outils d’analyse de code qui permettra de fournir des indicateurs :

- duplication de codes
- couverture de tests
- codes obsolète, code mort
- …

cf. Sonar

La gestion de la dette

La dette technique doit être résorbé tout au long de la vie du projet.

Comment la reconnaître ?

- On ne touche pas à ça
- Ah, c'est pas la fonctionnalité qui a été demandée et retirée du Backlog 50 fois ?
- Si on upgrade la version du Framework, ça va péter
- Des méthodes de 1000 lignes

Comment la suivre ?

- Noter systématiquement les points impactés dans le code pendant les revues de code
- Créer des tâches spécifiques dans un backlog technique

Comment la réduire ?

- Intégrer les tâches de dette dans le sprint planning
- BoyScout

BOYSCOUT

"Toujours laisser un endroit dans un état meilleur que celui dans lequel on l'a trouvé"

- Refactoring continu de l'application
- Sortir du périmètre de son développement
- Ne pas attendre de dégager du temps pour éponger la dette technique
- Valable pour le code mais aussi pour les tests