Contents

Généralités

- Communication est la clef
  Règles des 5 w (who = à qui on parle, why = pourquoi on le fait et comment, what = savoir ce qu'on demande etc..., where, when = deadline)
  La communication est différente en fonction de la personne à qui on parle
- Chef de projet = personne la plus polyvalente, avec une bonne communication
- Bilans
  revenir sur le projet, et voir ce qu'on a fait bien et mal fait (retour sur expérience)
- SMART (pour découvrir et atteindre ses objectifs)
  - S (spécifique)
  - M (mesurable)
  - A (atteignable)
  - R (réaliste)
  - T (temporel)

- Rex
  Original goals ➔ Original plan ➔ “What went wrong ?”
  = Retour sur experience
- Conseils
  - Préparer un support pour une réunion
  - Préparer la réunion à l’avance
  - Fixer les rdv sois même
  - Lors d’une prise de décision, appliquer les 5w
  - Réunion régulière avec d’autres personnes ➔ sortie du tunel vision
  - Fixer des deadlines

Git

Généralités

- Logiciel de gestion de versions.
- Permet de stocker un ensemble de fichiers chronologiquement. Pour voir l'évolution et revenir sur des version du projet précédente

Avantages

- Gérer des versions différentes d'un projet (local/online)
- Partage de code en communauté
- Coordination du travail en équipe
- Déploiement d'application (Dépôt à distance)
- Revenir sur les versions antérieures du projet.
- Seul = Gérer son code, A plusieurs = Partager le code et travailler de manière isolée.

Inconvénients

- Apprentissage du logiciel…

Architecture décentralisée/distribuée

- Partage de sauvegardes sur le même réseau sur différents appareils
- (Git est un système décentralisé ???)

Fonctionnement interne

- Constitution hiérarchique (comme un oignon et ses couches)
- .md = markdown (langage de balisage)

Temporalité

- Les versions dans le temps t, évoluent (version t1, version t2, version t3…)

Dépôt git

- La seule différence entre répertoire classique et dépôt git, dans le dépôt il y a un dossier caché supplémentaire .git (ce dossier contient les sauvegardes des codes sources et modifications)
- NE PAS SUPPRIMER le dossier .git car il fait du dépôt sa caractéristique de dépôt, si supprimé, redevient juste une seule version du projet.

Persistence map

- Git utilise des dictionnaires pour sauvegarder. ➔ appelé « map » avec une clef et valeur
- La clef est unique, utilisation d'une fonction « SHA-1(str) » qui génère une clef unique. SHA-1 est une fonction de hachage cryptographique.

Base de donnée Clef-valeur

- Commande de bas niveau ➔ « git hash-objet » mais jamais utilisé.
- Cette commande prend les données, stock dans le dossier .git et retourne la clef de stockage.
- *Dans .git, on retrouve un dossier « objet » dans lequel se trouve un dossier nommé avec les deux premiers éléments de la clef (hachée) et dans ce dossier se trouve la clef hachée complète (c'est un fichier) et dans ce fichier se trouve les donnée du contenu lié à la clef*
- *Commande « cat-file » est la fonction inverse, qui vient chercher la valeur du fichier nommé par la clef (hachée) et retourne la valeur stockée.*

- *Le dossier dans lequel sont stocké les dossiers de données est appelé « object »*

- *Structure de donnée ➔ DAG (directed acyclic Graph), Graphiques sans boucle fermée.*

Objet blob

- Blob = binary large object, stock le contenu d'un fichier
- Le contenu est compressé par un système Zlib (illisible par l'humain)

Object tree

- Permet de sauvegarder la structure du projet, donc les dossier, noms de fichier, etc…
- Exemple, un dossier sera sauvegardé dans un fichier avec les noms des fichiers dedans, le types, etc.
- Tree veut dire que c'est un dossier.

Object commit

- Contient…
  - La racine du projet
  - Le parent, date et heure, message (commentaire), auteur, …
- Il est stocké aussi dans le dossier object et haché via SHA-1 et Zlib
- Les commits sont liés entre-eux dans le temps pour pouvoir créer une temporalité, ils sont parents. Le premier commit racine est le commit racine du projet.

Références

- Référence tag
  - Pointeur fixe, permet de marquer un commit. Par exemple pour donner la référence de version (V1.0, V1.1, …) Afin d'appeler les commit via un tag et non le nom des hash (SHA-1).
- Référence Branche
  - Pointeur qui pointe vers le dernier commit.
  - Branche Main (Master) = Branche principale (par défaut) *(dynamique)*
- Référence Head
  - Fichier texte contenant la référence d'une branche
  - Permet la navigation dans le graphe en s'attachant à une branche
  - Permet de sélectionner la branche sur laquelle on travaille.
- Commit orphelins
  - Commit qui n'est plus référencé, aucuns commit ne pointe vers lui.
  - Ils sont supprimés par le « garbage collector » après 90jours

Trois zones git

- Working directory (edition des fichiers) (wd)
- Dans le dossier .git
  - Index (zone tampon, pour préparer les commits)
  - Objects (dépot, il stock les commits)
- Indexion
  - Permet d'ajouter les fichiers modifiés à la zone git (index)
  - Via la commande "git add ."
- Commit
  - Via commande "git commit"
  - Permet d'ajouter les fichiers en index (tampon) à l'historique des commits du dossier git. Afin de sauvegarder les modifications

Commandes de bases

- git init [nom_dépot]
  Crée le dépôt .git
- Config
  - git config –system <paramètre> <valeur>
    Niveau système, liés à tous les users du système (/etc/gitconfig)
  - git config –global <paramètre> <valeur>
    Niveau précifique à chaque utilisateur (~/.gitconfig
  - git config <paramètre> <valeur>
    Niveau local, spécifique à un dépôt local (.git/config)