Frédéric paquette

Francis lemaire

Introduction A GitHub

**Qu'est-ce que GitHub**

Premièrement qu’est-ce que un outil de gestion de version. Cet outil permet de stocker toutes les modifications en partant de la création du répertoire au produit final. Il permet ainsi de revenir à une version antérieure facilement.

Github est un site d’hébergement pour la création de logiciel qui utilise le gestionnaire de version git

**Les principaux Avantages de GitHub**

Les avantages d’utiliser un gestionnaire t’elle que GitHub sont nombreux. Premièrement la gestion du projet est énormément simplifiée. Le transport et le transfert des parties se fait rapidement et simplement, car tout le monde travail sur le même projet (code).

Les mises à jour son simplifié par son utilisation il suffit d’implémenter une nouvelle fonctionnalité ou Déboguer une partie du projet et ensuite l’envoyer sur github et tout le monde y ont accès immédiatement. Par ces même faits le suivie du projet peut être beaucoup plus vigoureux simple et efficace.

**Les Principaux Désavantages de GitHub**

L’apprentissage de GitHub peut sembler ardu pour certain. Les conflits Au début son un casse-tête et l’apprentissage de leur gestion prend un certain temps. Si ces mêmes conflits ne sont pas bien gérer lorsqu’ils sont d’une certaine ampleur la perte de donnée est probable et ont doit prévenir cela.

**Pourquoi le Shell et non l’interface dans certain cas**

Certaine commande du shell ne sont pas offerte avec l’interface ou n’est pas encore implanter donc pour certain cas nous devons passer par le shell par exemple lors de certain cas d’erreur ou que tu veux revenir a un commit antérieur (le rollback) aucune option n’est offerte avec l’interface donc tu dois le faire avec le shell ou lorsque une synchronisation ne veux pas être effectuer tu peux la forcer a l’aide du shell ce qui est impossible avec l’interface. Donc, certaine commande du shell son indispensable et doivent être connu pour pouvoir fonctionner à 100% avec GitHub.

**Les principales commandes**

Git config

initialise les configurations du répertoire

git init

initialise le répertoire initiale du projet

git clone

copie un répertoire localement

git add

ajout des fichier dans l’index pour les enregistrement(commit)

git commit

enregistre les changements

git push

envoie les modification au repertoire

git pull

merge le repertoire a ton repertoire local

git status

Indique l’état actuel de ton répertoire

Personnellement je crois qu’avec ces commandes vous pouvez vous débrouillez facilement pour tout faire avec vos projets.

Pour plus d’information  ou plus de commandes: https://www.siteground.com/tutorials/git/commands.htm

**L’interface en Bref**

L’interface Facilite l’apprentissage par sa simplicité elle permet aussi de gérer des conflits très rapidement et rend les commits et la synchronisation beaucoup plus rapide et simple.

L’interface te permet de copier (cloner) un projet git sur ton ordinateur elle te permet de créer un nouveau projet git de supprimer un projet git cloner sur ton ordinateur mais pas du site(pour sela tu dois aller sur le site) . Avant toute synchronistion avec le site tu dois avoir enregistrer tout changement sur le projet (commit) a moins que tu ne veux plus des changements effectuer tu peux revenir à l’état du dernier commit simplement avec la fonctionnalité discard all change.

**Comment Gérer les Erreurs**

Unecause de conflit difficile à gérer sont les windows forms

Les raisons le Designer.cs est perçu comme un fichier texte et lorsque 2 personnes tente de changer une partie de l’interface il y aura conflit car GitHub n’est pas capable de gérer le fichier correctement. Donc pour éviter ce genre de conflit jamais 2 personnes ne doivent travailler sur la même interface en même temps.

**Comment Gérer les Merges Conflics**

Il y a plusieurs manières de gérer les merges conflics

Avec l’interface windows de github

l’interface de github permet de gérer les conflits lors de Merge elle permet de voir les différences entre les deux fichiers les parties qui cause problèmes.

Premièrement pour voir le conflits il suffit de double cliquer sur le fichier concerné pour le faire afficher il y aura une entete qui delimite la zone de conflit vous pouvez éditer carder les changements que vous voulez les supprimer et ensuite enregistrer(commit) et ensuite le synchronisé.

Pour plus d’information : https://help.github.com/articles/resolving-merge-conflicts/

Avec visual studio

Dans l’onglet Team explorer il y a plusieurs option lorsqu’un conflit survient l’option voir les différences apparait cette option ouvre une fenêtre avec les deux fichiers ou les différences qui cause conflit son surligner en rouge alors tu peux choisir ce que tu veux garder enlever tu peux aussi choisir de tout garder. Il y a aussi une option rapide ou tu peux choisir le fichier à Distance qui est la version sur github ou Choisir la version locale qui est ta version du fichier. Ensuite un commit est nécessaire.

Avec le shell

Utiliser la commande git status elle vous listera dans la console les informations du conflits ensuite vous faite les changements souhaité encore une fois les changement que vous voulez etc ensuite il suffit d’utiliser la commande git add le fichier et ensuite enregistrer(commit) et synchroniser

Pour plus d’information : https://help.github.com/articles/resolving-a-merge-conflict-from-the-command-line/

**Les GitIgnore**

Les gitignore sont très utiles il permettre lorsque un commit est effectuer d’ignorer certain type de fichier en fonction du type de langage exemple les .exe ou les .suo qui sont des fichiers que git ne peut gérer et que si il se retrouve dans les commit cause beaucoup d’erreur .lorsque tu créée un répertoire met Des GitIgnore automatiquement de base. Mais il est primordial de choisir les bons pour le projet que tu es en train d’effectuer par exemple il y a des gitignores spécialement pour un projet visual studio.

**Le RollBack**

Le Rollback consiste a revenir à un commit antérieur. Lorsque plusieurs erreur survienne ou que pendant une synchronisation des données son perdu alors le rollback est très utile pour ne pas perde des données. Il doit etre absolument effectuer dans le shell . Premièrement tu dois utiliser la comment git reset avec l’option –hard cette option est nécessaire pour écraser la dernier version sur le git sinon tu as en main la dernière version mais le répertoire sur git ne l’est pas. La syntaxe de la commande est : git reset –hard <l’adresse du commit> l’adresse du commit se trouve sur le site internet on peut facilement la copier coller.

**Le Force Push**

Le force push est une technique pour forcer une synchronisation avec la version local qui autrement ne peut être effectué il suffit d’effectuer la commande git push origin master –force.

Il faut être prudent avec cette commande car elle peut corrompre la branch master si vous lui envoyer n’importe quoi. Donc, utiliser cette commande avec précaution.

**New Branch**

Premièrement Qu’est-ce-que une branch? Une branch est une copie de la branch master elle permet de travailler en parallèle sans compromettre la branch master.

Elle sert premièrement à isoler une partie de code incomplet ou «brisée » si tu veux régler un problème sans compromette le projet au complet ou simplement tester de nouvelles choses.

Premièrement la création de la branch

Avec le shell : git branch <Lenom> la commande crée une nouvelle branch

Suite à cette commande tu ne peux pas encore enregistrer (commits) tu dois la sélectionner a l’aide de la commande git : checkout –b <LeNomDeLabranhExistante>

Pour supprimer une branch il suffit d’utiliser la commande : git branch –d <lenom> cette commande est la commande sécuritaire qui ne te laisse pas supprimer si tu n’as pas merge avant

La commande non sécuritaire qui supprime sans aucune vérification : git branch –D <lenom>

L’interface te permet aussi de créer rapidement une nouvelle branch et elle permet quelques outils de gestion.

Pour plus d’informations : http://git-scm.com/docs/git-branch

**Les tags**

Les tags permettre de marquer certain points dans l’histoire du projet exemple les release ou des Commits important par exemple l’implantation de la boucle de jeu ou d’une partie très importante du projet.

La commande git tag <lenom>

La gestion des tags se fait un peu comme les branch

Pour plus d’informations : http://git-scm.com/docs/git-tag

**Tutorial**

Plusieurs tutoriel son offert sur internet

Openclassroom en offre un bon et complet

http://openclassrooms.com/courses/gerer-son-code-avec-git-et-github