L1 HLIN102 - Du binaire au Web
TD/TP 8 - partie 1
Introduction au travail collaboratif à distance

Ce TP a pour objectif de vous initier à l'utilisation d'outils permettant à plusieurs utilisateurs de travailler ensemble sur un même projet (donc de collaborer). Cela vous sera très utile lors de la réalisation de projets ou de TP à plusieurs.

1 Framasoft

Framasoft est un service de stockage et de partage de documents entre plusieurs personnes, qui sont amenées à les utiliser et à les modifier (simultanément) à distance. Ces documents peuvent être des fichiers texte, des tableurs, des présentations, des images, etc. Cette section a pour objectif la découverte des principales fonctionnalités de Framasoft pour le traitement de texte et les tableurs.

1.1 Pour commencer

Regroupez-vous par groupes de deux ou trois. Connectez vous sur `http://framasoft.net/`.

Durant ce TP, nous utiliserons des services dit "publics", c'est-à-dire ne nécessitant pas d'identification : la seule possession de l'url permet d'éditer le document. Si par la suite vous voulez utiliser framasoft de manière plus confidentielle, il est possible de s'identifier et de ne partager vos documents qu'avec les personnes que vous avez choisies.

1.2 Traitement de texte

1. L'un des membres du groupe va créer un document texte (Libres Services -> Éditer -> framapad -> pad publics -> Créer un pad).

2. Envoyer par mail le lien vers votre nouveau document à vos camarades.

3. Travailler en groupe sur le même document partagé et observez ce qu'il se passe.

4. Lorsque vous travaillez ensemble sur un même document, il est possible de discuter en utilisant l'application de chat. Tester cette dernière. Quelle est son utilité ?

5. Framasoft garde un historique des modifications (dans l'ordre chronologique) effectuées par les collaborateurs. Pour voir cet historique, aller dans "Timeslider" (l'horloge en haut à droite).

6. Exporter votre travail en format texte puis en format pdf.

1.3 Tableur

1. De la même manière que pour le traitement de texte, l'un des membres du groupe va créer un document de type tableur, dit "framacalc", et envoyer par mail le lien aux autres membres du groupe.

2. Travailler en groupe sur le document partagé, en particulier, tester des écritures simultanées dans une même cellule et observer ce qui se passe.

3. Sélectionner plusieurs cellules, puis aller dans "format". Tester les différentes options proposées. Que fait chacune d'elle ?

4. Voir l'historique des modifications et observer leur évolution.

2 GIT

GIT est un système qui permet le développement collaboratif de logiciels (organisés sous la forme de projets). Il s'agit d'un logiciel qui permet de conserver l'historique des modifications des fichiers dans un projet. Il permet aussi à plusieurs personne de travailler en parallèle sur un même projet et chacun de son coté sans systématiquement saboter le travail des autres. Il existe beaucoup d'autres gestionnaires de versions tels que Subversion (alias SVN), CVS, Mercurial, Bazaar, Darcs, Monotone, ...

Git est un gestionnaire de version décentralisé. Comprendre par là que lorsque vous téléchargez un projet git sur un serveur, vous téléchargez aussi tout l'historique qu'il contient. Vous obtenez ainsi sur votre machine toutes

les informations du projet qui sont stockées sur le serveur et vous pouvez ainsi vous-même devenir un serveur qui fournit ce même projet.

2.1 Configuration en local de git

Avant de commencer à utiliser git, il faut le configurer. Cela permet que votre nom soit associé à chaque modification que vous avez apportée à votre projet collaboratif.

```
git config --global user.name "Prenom Nom"
git config --global user.email "prenom.nom@etu.umontpellier.fr"
```

2.2 Connexion au gitlab

Par souci de simplicité, nous utiliserons principalement git de manière centralisée. Pour cela il nous faut un serveur. Ça tombe bien, l'université nous en fournit un.

Connectez-vous sur le gitlab de l'UM: `https://gitlab.info-ufr.univ-montp2.fr/users/sign_in`. On rappelle que votre identifiant est de la forme "prenom.nom@etu.umontpellier.fr".

2.3 Configuration de la clef ssh

Afin d'utiliser le gitlab correctement, il est nécessaire de lui renseigner une clef ssh. Si vous avez déjà une clef ssh sur votre ordinateur, la commande `cat ~/.ssh/id_rsa.pub` devrait retourner votre clef publique. Si elle retourne "Aucun fichier ou dossier de ce type", effectuez
`ssh-keygen -t rsa -C "prenom.nom@etu.umontpellier.fr"` puis à nouveau `cat ~/.ssh/id_rsa.pub`. Copiez votre clef commençant par "ssh-rsa".

Dans le gitlab, rendez-vous dans "profile settings" (l'engrenage en haut à droite), puis "ssh-keys" (dans le menu à gauche), puis "add ssh-key". Copiez votre clef ssh dans la boîte "key", puis valider avec "add key".

Noter que la manoeuvre que vous venez de faire vous permet d'utiliser le gitlab depuis l'ensemble des machines de la fac. Si vous voulez vous en servir depuis une autre machine, comme votre machine personnelle, il faudra ajouter la clef ssh de la dite machine. Pour la marche à suivre sous Windows et Mac, vous pouvez consulter la page d'aide `https://gitlab.info-ufr.univ-montp2.fr/help/ssh/README`

Voilà, toute la configuration est faite ! On va pouvoir commencer.

2.4 Git, un gestionnaire de version décentralisé

Faisons un test. Aller dans votre répertoire "HLIN102" et créer un dossier "TP8". Ouvrir un terminal dans ce dossier et effectuer la commande suivante :

```
git clone git@gitlab.info-ufr.univ-montp2.fr:p00000011491/Salade\_de\_fruit.git
```

Par cette commande, vous venez de copier le projet "Salade_de_fruit". Aller dans le dossier Salade_de_fruit. Le projet est constitué d'un simple fichier texte. Ouvrir un terminal dans ce dossier et exécuter la commande `git log` . Constater que le projet a eu au total 4 modifications. Pour chaque modification, on vous donne un identifiant, le nom de la personne ayant fait la modification, son adresse e-mail, la date de la modification, ainsi qu'un commentaire laissé par l'auteur. Si vous vous rendez sur la page du projet (`https://gitlab.info-ufr.univ-montp2.fr/p00000011491/Salade_de_fruit/commits/master`), le serveur vous donne exactement les mêmes informations.

2.5 Création du projet

Depuis le gitlab, créez un nouveau projet. On va lui donner le nom "liste". Il est important de garder le paramètre privé du projet. Votre projet est désormais créé sur le gitlab. Notez dans un coin l'adresse du projet, celle commençant par "git@gitlab". Dans la suite on y fera référence par `<adresse_projet>`.

Vous êtes bien content(e) avec un projet sur un serveur au loin, mais ça serait mieux si on pouvait l'avoir sur la machine locale. Ouvrez un terminal dans votre dossier TP8. Téléchargez le projet en effectuant la commande suivante :

```
git clone <adresse_projet>
```

Le dossier de votre projet est apparu, rendez-vous dedans. Pour le moment il n'y a rien dedans.

Créez un fichier "course" que vous remplissez de trois articles de course à acheter puis sauvegardez. Pour le moment votre fichier course n'a été modifié que dans votre répertoire de travail. Dans cet état, la modification n'a pas encore été prise en compte par git. Votre fichier est dans l'état "unstage". Effectuez un `git status` .

Vous allez ensuite sélectionner le ou les fichiers dont vous voulez que git prenne en compte la modification. Pour cela, dans le répertoire de travail, exécutez la commande `git add <nom_fichier>` pour chaque fichier à

rajouter. Ici, un seul. Votre fichier est désormais dans l'état "stage". C'est-à-dire que git a noté quels fichiers il va devoir prendre en compte la prochaine fois que vous lui demanderez d'enregistrer un changement. Effectuez à nouveau un `git status`. Notez le changement entre l'état unstage et l'état stage.

Si vous effectuez un `git log`, ce dernier est vide. Jusqu'à maintenant git n'a enregistré aucune modification. Il est peut-être temps que ça change ? Pour enregistrer un changement, il faut effectuer un `git commit -m ''premier commit''` . L'action de commit ne va sauvegarder que les fichiers qui sont dans l'état "stage". L'option `-m` permet de rajouter un commentaire. Ce commentaire est obligatoire et devra toujours résumer la modification que vous êtes en train de sauvegarder. Effectuez à nouveau un `git status`. Que s'est-il passé ? Effectuez un `git log`. Que constatez-vous ? Est-ce que votre git local a pris en compte la modification ? Est-ce que la modification a été prise en compte sur le gitlab ?

Pour que le serveur prenne connaissance des modifications, il faut les lui envoyer. Pour cela, effectuez `git push origin master` . Voilà, votre modification a été enregistrée sur le gitlab. Inspectez la page du projet et repérez où votre modification apparaît.

Rajoutez deux autres articles sur votre liste de course et enregistrez à nouveau ces modifications.

Pourquoi est-il intéressant de ne sauvegarder que les fichiers déjà dans l'état "stage" ? Quel est l'intérêt du commentaire ?

2.6 Travail en parallèle et concurrence

De retour dans votre dossier "TP8", vous allez créer deux dossiers "Projet1" et "Projet2". Allez dans Projet1 et clonez votre projet avec `git clone` comme en 2.5. Vérifiez que vous avez bien récupéré la dernière version de votre projet. Faites de même dans Projet2.

Maintenant à la fin de la liste de course du Projet1, rajoutez de l'"huile d'olive". Enregistrez puis exportez vos modifications sur le gitlab comme en 2.5. Dans le Projet2, rajouter non pas de l'"huile d'olive", mais du "beurre salé". Enregistrez puis exportez vos modifications. Que se passe-t-il ? Comment résoudre le problème ? Mettez à jour votre git local du Projet2 en effectuant un `git pull` . Lisez le retour de votre commande, elle annonce des conflits. Ouvrez votre fichier de course, repérez les conflits et résolvez les. Enregistrez puis exportez vos modifications. Cette fois-ci, normalement, il n'y a plus d'erreur, le fichier a bien été exporté.

Réitérez la même expérience mais cette fois-ci en rajoutant dans Projet1

des “bananes” en haut du fichier course et en rajoutant dans Projet2 des “pommes” en bas du fichier course. Que constatez-vous ?

Que se passe-t-il si vous rajoutez exactement la même chose dans les deux projets ? Que fait `git pull` lorsque vous avez modifié un fichier mais pas sauvegardé ses changements dans git quand le git est à jour ? Quand il ne l'est pas ?

Lorsque vous travaillez sur un projet git, il est vivement conseillé de récupérer la dernière version du projet, via un `git pull` à chaque fois que vous commencez une session de travail et de sauvegarder a chaque fois que vous avez fait une avancée.

2.7 Travail en groupe

L'une des personnes du groupe va créer un nouveau projet puis rajouter ses camarades au projet. Recréez chacun de ces problèmes de concurrence.

2.8 Pour aller plus loin

Le site officiel de git est `http://www.git-scm.com/`. Vous y trouverez toutes les informations concernant git.

Cherchez comment faire pour que lors de l'utilisation de `git add`, tous les fichier d'un certain type (typiquement .c, .cpp ou .ml) passe dans l'état “stage” d'un seul coup. À l'inverse comment faire pour que certains types n'y passent jamais ?

A quoi correspond le “origin” et le “master” lors de l'utilisation de `git push origin master` ?

Qu'est-ce qu'une branche d'un projet git ? Comment fusionner deux branches ?

3 Conclusion

Quelles sont les principales différences que vous constatez à l'utilisation de GIT et Framasoft ? Quels sont les avantages et inconvénients de chaque outil ?