Utilisation de GIT.

Crée un dépôt

Création d'un répertoire
Dans ce répertoire, faire un >>**git init**

Cloner un dépôt existant

>>**git clone http://github.com/symphony/symphony.git**
Le protocole peut être git:// et ssh://

Opérations de base

>>**git status** ou **git st**
Affiche toutes les fichiers modifiés

>>**git diff**
Affiche les modifications de tous les fichiers

>>**git diff src/symphony/compionents/Yam.yl**
Affiche les modifications sur le fichier Yam.yl

Effectuer un commit des changements

- faire **git add nomfichier1 nomfichier2** pour ajouter les fichiers à la liste de ceux devant faire l'objet d'un commit, puis faire un git commit. Si vous faites un git status après un git add, vous les verrez alors en vert ;
- faire **git commit -a** pour « commiter » tous les fichiers qui étaient listés dans git status dans les colonnes « Changes to be committed » et « Changed but not updated » (qu'ils soient en vert ou en rouge) ;
- faire **git commit nomfichier1 nomfichier2 -m 'mon commentaire'** pour indiquer lors du commit quels fichiers précis doivent être « commités ».

Un commit avec git est Local

Vérifions les logs

Git log
On utilise les touches Page up – Page down et Q pour quitter

Git commit – amend → Pour changer le commentaire du commit précédent
Git reset HEAD^ → Pour effacer le dernier commit. (Attention au chapeau)

Seul le commit est retiré de Git (du répertoire .git), vos fichiers, eux, restent modifiés

Pour effacer le dernier commit et les changements effectués dans les fichiers, il faut un reset hard.
Git reset –hard HEAD^ . (Attention au chapeau)

Pour restaurer un fichier avant commit
Git checkout nomfichier

Pour annuler un **git add nomfichier → git reset HEAD – fichier a supprimé**

Travail avec le serveur

Télécharge les sources à partir du serveur
Git pull
Effectue une fusion des souces avec la version locale.

Pour visualiser les logs local
Git log –p

Pousser les commits locaux avec les tags vers le serveur
Git push --tags
Conseille de faire un pull avant de faire un pull. Permet d'éviter les conflits.

Pour annuler un commit publié
Git revert id

Travailler avec les branches

Lister les branches

Git branch

Créer une branche
Git branch mabranche

Pour changer de branche
Git checkout mabranche

Fusionner les changements
Git checkout master
Git merge mabranche

Supprimer une branche
Git branch –d mabranche

Liste des fichiers en cours de modification
Git status
Permet aussi de visualiser

>>Pour changer de branche sans perdre de modification et sans commit

Fige les fichiers modifiés et met de coté.
Git stash

Restaurer les fichiers modifiés l'état des fichiers avant le git stash
Git stash apply

Les branches Partagées

Lister toutes les branches du serveur
Git branch –r

On peut faire une copie d'une branche serveur
Git branch –track branchelocale origin/brancheserveur

Ajout une branche sur le serveur
Git push origin origin :refs/heads/nom_nouvelle_branche

Voir les autres commandes de suppressions de branch sur le serveur.

Tagger une version

Pour créer un tag
Git tag nomtag idcommit

Pour supprimer un tag
Git tag –d nomtag

Recherche dans les fichiers source

Recherche un mot dans les fichiers
Git grep « toto »

Recherche un mot dans les fichiers, pour connaître les numéros de ligne
Git grep -n « toto »

Demander a git d'ignorer des fichiers

Créer un fichier .gitignore

```
project.xml
dossier/temp.txt
*.tmp
cache/*
```

Aucun de ces fichiers n'apparaîtra dans **git status**, même s'il est modifié. Il ne paraîtra donc pas dans les commits. Utilisez cela sur les fichiers temporaires par exemple, qui n'ont aucune raison d'apparaître dans Git.

Il est possible d'utiliser une étoile (*) comme joker. Dans l'exemple précédent, tous les fichiers ayant l'extension « .tmp » seront ignorés, de même que tous les fichiers du dossier cache.