|  |  |  |  |
| --- | --- | --- | --- |
| **Séquences** | **Tâches** | **Commandes** | **Observation/ Commentaires** |
| Commandes de base | -Vérification | git --version |  |
| -Configuration | git config --global user.name ‘’…’’  git config --global user.email ‘’…@...’’ |  |
| -Copier depuis un serveur | git clone [https://<username>@bitbucket.org/<username>/xxx.git](https://%3cusername%3e@bitbucket.org/%3cusername%3e/xxx.git) git clone [https://github.com/<username>/xxx.git](https://github.com/%3cusername%3e/xxx.git) |  |
| -Initialisation | git init |  |
| -Indiquer le dépôt distant | git remote add origin [https://github.com/<username>/xxx.git](https://github.com/%3cusername%3e/xxx.git) |  |
| -Ajouter des fichiers au versionnage  (indexer nouveaux fichiers dans le ‘Staging Area’) | git add Carre.java | Pour un seul fichier |
| git add . | Pour tous les fichiers |
| -Enregistrement des modifications dans le dépôt local | git commit Carre.java | Du Workspace au Local Repo |
| git commit -m’’premier commit’’ |
| git commit -a |
| -Envoyer les modifications vers le dépôt distant | git push |  |
| -Récupérer le dépôt distant et l’appliquer sur l’espace de travail | git pull |  |
| -Récupérer le dépôt distant et l’appliquer sur le dépôt local sans affecter le Workspace | git fetch |  |
| -Renommer le fichier | git mv carre.java Carre.java |  |
| -Supprimer un fichier | git rm Carre.java |  |
| Métadonnées | -Modifs actuelles dans le Workspace (ajouté, modifié, supprimé) | git status |  |
| -Comparer entre deux commit | git diff |  |
| git diff sha1-1 sha1-2 |  |
| -Historique de commit | git log |  |
| -Afficher qui a modifié un fichier | git blame Carre.java |  |
| Annuler les modifications | Retirer un fichier indexé du ‘’staging area’’ (inverse de git add) | git reset Carre.java |  |
| Annuler les modifications apportées sur l’espace de travail (Workspace) | git checkout Carre.java | Mettre le contenu du staging area |
| git checkout sha1 Carre.java | Mettre le contenu du commit sha1 |
| Changer la HEAD (pointeur actuel) | git reset sha1 --soft | écrase le dépôt local seulement |
| git reset sha1 --mixed | N’écrase pas l’espace de travail |
| git reset sha1 --hard | écrase tout |
| Annuler le commit en laissant l’historique | git revert sha1 |  |
| Corriger le dernier commit avec les modifications actuelles (en cas d’oubli de fichiers) | git commit --amend |  |
| Quelques schémas |  |  |
| Les Branches | Introduction + présentation |  |  |
| Lister les branches existantes | git branch |  |
| Créer une nouvelle branche | git branch creer\_patient |  |
| git branch creer\_patient develop |  |
| git branch creer\_patient sha1 |  |
| Changer de branche (pointer la HEAD vers le commit le plus récent) | git checkout creer\_patient |  |
| git checkout -b creer patient | Créer et se positionner sur une branche |
| git checkout master | Se positionner sur la branche principale |
| git checkout sha1 | Se positionner sur un commit particulier |
| Supprimer une branche | git branch -d creer\_patient |  |
| Fusionner deux branches  étape 1  étape 2 |  |  |
| git checkout develop |  |
| git merge creer\_patient |  |
| La fusion peut ne pas s’exécuter à cause des conflits. Dans ce cas on peut :  -soit annuler la fusion  -soit résoudre les conflits en éditant les fichiers puis finaliser la fusion avec : |  |  |
|  |  |
|  |  |
| git merge --abort |  |
| git commit |
| Affichage des conflits | git status |  |
| Fusion avec une stratégie de résolution de conflits | git merge -s ours |  |
| git merge -s theirs |  |
| Intégrer toutes les modifications d’une branche sur une autre branche | git rebase creer\_patient |  |
| Remarques : | 1) les stratégies [git merge -s ours] et [git merge -s theirs] pas très recommandables, dans l’un ou l’autre cas ça écrase le travail de l’un ou l’autre |  |
| 2) généralement on utilise le ‘merge’ et non le ‘rebase’ |  |