Working in several branches in git

1) Stash les modifications: Lorsqu'on travaille sur un ticket T1 et on veut changer vers le T2 sans faire le commit on peut utiliser `git stash` pour mettre de coté ces modifications temporairement

```
git stash push -m "in progre ss"
```

2) Switcher de branche : On peut changer de branche pour passer au T2 avec la commande git checkout

```
git checkout nom_br anc he_T2
```

3) Appliquer les modifications stashées si nécessaire: Si on a besoin de revenir sur le ticket T1 et de continuer là où on était arrêté, on peut listerles stash avec `git sta sh list` et réappliquer celui q'on veut avec `git stash pop` ou `git stash apply`.

```
Git checkout nom_br anc he_T1
```

```
Git stash list
```

```
Git stash pop stash@{0}
```

Git restore all modifications

Pour restaurer les modifications dans tous les fichiers (c'est-à-dire revenir à l'état du dernier commit pour tous les fichiers et dossiers sous ce répertoire. Le point après git restore spécifie le répertoire courant, qui inclut tous les fichiers et dossiers de ce répertoire. `Git restore .`

2) Si vous avez également également ajouté des fichier à la zone de staging (avec git add) et que vous souhaitez les retirer de la zone de staging et annuler leurs modifications, vous devez utiliser l'option --staged en plus de restaurer les modifications dans le répertoire de travail : dans ce cas on va avoir deux commandes, la première retire les fichiers de la zone de staging (sans modifier leur contenu dans votre répertoire de travail), et la deuxième commande restaure le contenu de ces fichiers à leur dernier état commité.

```
Git restore --staged .
```

```
Git restore .
```

Modify Remote branch

Si vous avez créé la branche en local avec un autre nom que celui de la branche distante (feature-table-bug), cela ne causera pas de problème pour votre travail local ou pour le merge avec master. Cependant, cela peut introduire des différences lors de la synchronisation avec le dépôt distant, notamment si vous souhaitez pousser vos modifications sur la branche distante originale (feature-table-bug). Pour gérer cela, voici ce que vous pouvez faire :

Pousser vos modifications sur la branche distante originale Si votre objectif est de pousser les modifications de votre branche locale (avec un nom différent) sur la branche distante feature-table-bug, vous pouvez spécifier explicitement le nom de la branche distante lors du push. Cela poussera les modifications de votre branche locale (peu importe son nom) vers la branche distante feature-table-bug.

```
git push origin branch e_l ocale
```

```
: featur e-t abl e-bug
```

By **Nouha_Thabet**
cheatography.com/nouha-thabet/

Not published yet.
Last updated 31st March, 2024.