Process pour apporter des modifications

1. Je reproduis sur ma dev : dev-julien / dev-perrine
2. J'effectue les développements
3. Je teste sur ma dev : dev-julien / dev-perrine
4. Je vérifie les modifications effectuées (commande A/B)
5. J'ajoute mes modifications à mon GIT en local (commande C)
6. Je supprime les fichiers de mon GIT local (commande D)
7. Je commit en indiquant un message clair (commande E)
8. Je pousse mes modifications sur le serveur GIT distant (commande F)

Si besoin de mettre en PREPROD

9. Je liste les tags exitants (commande G)
10. Je créé un tag dans la continuité, en respectant les règles définies (commande H)
11. Je pousse le tag sur le serveur GIT distant (commande I)
12. Je demande une évolution en PREPROD à l'EXPLOIT (voir informations nécessaires)

Si besoin de mettre en PROD

13. Je demande une évolution en PROD à l'EXPLOIT (voir informations nécessaires)

Si besoin d'effectuer un retour en arrière sur PROD

14. Je cherche la dernière version stable sur GITLAB
15. Je demande une évolution en PROD à l'EXPLOIT (voir informations nécessaires)

Liste des commandes importantes

#	Description	Commande
A	Voir les fichiers en attente	`git status`
B	Voir les modifications en attente	`git diff <name_of_file>`
C	Ajouter les fichiers modifiés sur le GIT local	`git add .`
D	Supprimer les fichier supprimés du GIT local	`git rm <name_of_file>`
E	Commiter les fichiers	`git commit -m '<message>'`
F	Envoyer les commits en attente sur le GIT distant	`git push origin <my_dev>`
G	Voir les tags existants	`git tag`
H	Créer un tag	`git tag -a <tag> -m '<message>'`
I	Envoyer un tag sur le GIT distant	`git push origin <tag>`

Règles sur la nommage des tags

Le tag (nom donné par GIT) sert à fixer définitivement une version de l'application.

- Le tag doit toujours commencer par « v »
- Le tag est ensuite composé de 2 à 9 digits indiquant la version (exemple : v1.2.5)
    - Avec 1 étant la version majeure actuelle (en cas de refonte complète)
    - Avec 2 étant la version mineure (en cas de nouvelle fonctionnalité)
    - Avec 5 étant la version corrective (en cas de correctif sur la version précédente)

Informations nécessaires pour l'EXPLOIT

Dans tout les cas, lors du passage d'une nouvelle version sur un autre environnement, il faut leur communiquer :

- L'environnement de destination (PREPROD ou PROD)
- Le numéro de la version (tag créé)
- S'il y a (ou non) un changement du fichier parameters.yml

Au besoin il faut :

- Communiquer les changements effectués sur parameters.yml (ajout de ligne, modification suppression)
- Demander la relance des démons par les scripts de service
- Préciser l'urgence de la demande

Relancer les démons, pourquoi et quand ?

Les démons en PREPROD et en PROD tournent constamment. S'il y a une modification apportée, le démon lancé, continuera d'utiliser l'ancienne version. Il faut donc le relancer.

Si la commande de l'un des deux démon est modifiée ou si un fichier appelé par le démon est modifié, alors il est nécessaire de relancer le démon. En cas de doute, mieux vaut le relancer.