Code du Document : **ITS201902PGV**

Plan de Gestion des Versions

Historique des Modifications

Version	Date	Author	Description
1.0	08-02-2019	Eloundou Célestin P. Tema Gildas Fomena Landry	Version initiale

Table des Matières

Introduction

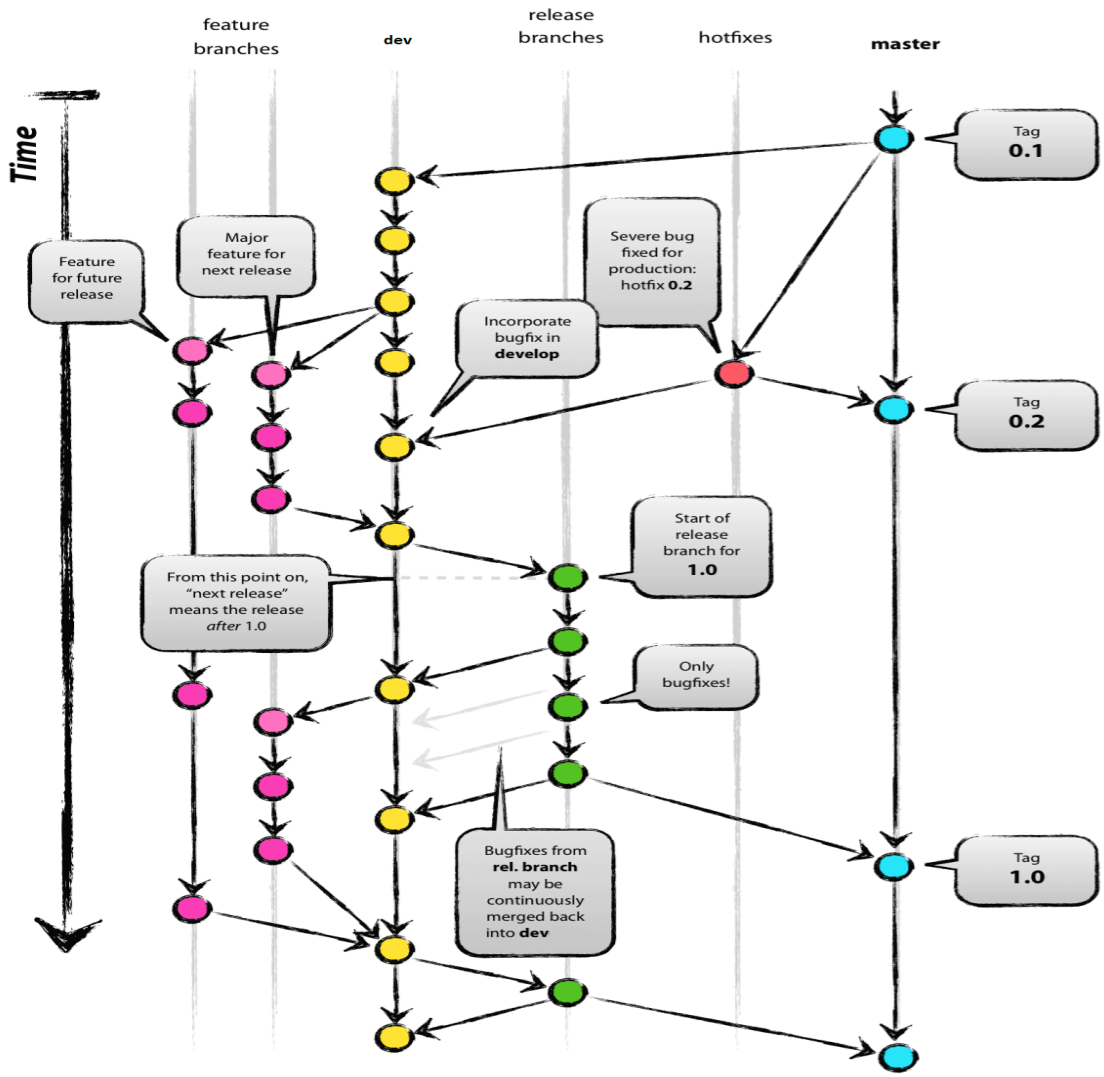
La gestion des versions d'un logiciel est un processus de planification et contrôle des sources d'un logiciel développé à travers divers étapes et environnements. Dans ce document, nous présentons l'approche à suivre pour la gestion des versions des codes sources des applications développés au sein de ITS.

Cible du document

Ce document est destiné à toute personne impliquée dans les phases de développement.

Etape à suivre

Pour la gestion des versions, nous utiliserons Git (cf annexe 1) couplé à Gitflow (cf annexe 2). Le schéma ci-après résume l'approche de gestion de branche proposé par gitflow.



Par défaut nous aurons toujours deux branches : **dev** et **master**

- La branche **dev** est utilisée pour l'intégration de toute nouvelle fonctionnalité prête pour les tests,
- La branche **master** est utilisé pour l'intégration de toute nouvelle version homologué de l'application. Une fois qu'une nouvelle version de dev est prête pour l'homologation, la branche **release** est créé pour contenir les changements fait sur une période (un sprint). Après homologation de la branche release, elle est mergé dans la branche master puis tagué avec un numéro de version suivant la numérotation des versions décrite plus bas par le chef d'équipe, ou toute autre personne ayant formellement reçu le privilège de le faire. La nouvelle version de master peut être déployé en production. **NB** uniquement les versions de master peuvent être déployé en production. Une fois la branche release mergé à master, elle est supprimée.

En plus des branches précédemment citées, nous aurons les branches suivantes :

- **hotfix :** crée à partir de la branche master chaque fois qu'un nouveau bug est rencontré en production. Le bug est corrigé sur cette branche puis mergé dans la branche master après un « **test rapide** » puis de nouveau tagué. La branche hotfix est par la suite mergé à la branche dev puis supprimée.
- Tout développement d'une nouvelle fonctionnalité se fait sur une branche portant un identifiant de la fonctionnalité à développer, et créé à partir de la dernière version de la branche **dev.** Une fois le développement de la fonctionnalité terminée, la branche est mergé avec dev pour résoudre les éventuels conflits puis on se place dans dev pour merger dev avec la branche de la nouvelle fonctionnalité développée. La branche dev est par la suite pushé sur le serveur pour rendre les modifications accessibles par les autres développeurs intervenant sur le projet.

Rôles et responsabilités

A l'initialisation du projet, l'**administrateur** crée 2 branches par défaut sur le serveur central ; les branches **dev** et **master**. Seul l'administrateur pourra faire des changements sur la branche master. Chaque développeur fait ses développements à partir de la branche **dev.** Une fois le développement d'une nouvelle fonctionnalité terminée, le développeur met ses modifications dans la branche dev puis il push sur le serveur. A la fin de chaque sprint, une version livrable de l'application doit être disponible. Un des développeurs travaillant sur le projet crée donc une branche **release.** Et la rend disponible sur le serveur sous un délai de 24h. Une fois qu'il crée la branche, il informe l'administrateur par mail et met en copie les managers. L'administrateur va déployer la branche release sur le serveur de pré-production. Une fois déployée sur le serveur de pré-production, un mail est fait à l'équipe d'homologation et aux managers pour faire l'homologation dans un délai de 3 jours et faire un rapport envoyé par mail à l'équipe de développement puis aux managers. Si aucune erreur majeure n'est rencontrée, alors un mail est fait aux managers par l'équipe d'homologation pour autoriser le déploiement à l'administrateur. L'administrateur à 24h pour mergé la branche release dans la branche master et déploie la branche master en production. Si l'équipe d'homologation rencontre des erreurs majeurs pendant les tests, alors les développeurs vont immédiatement prendre la branche release fixer les bugs majeurs puis signaler par mail l'administrateur qui va reprendre sa procédure.

Numérotation des versions

La numérotation des versions se fera ainsi qu'il suit :

- Nous utiliserons un système de numérotation sur 4 positions. Ex : 2.4.3.12
- Tout changement issue de la résolution d'un bug entraine une incrémentation du chiffre après le 3ième point. Ex : la version après 2.4.3.12 est 2.4.3.13
- Tout changement issue d'un ajout de fonctionnalités à la fin d'un sprint entraine une version mineure ie une incrémentation du chiffre après le 2ième point et la disparition de ce qui vient après. Ex : après la version 2.4.3.13, la version mineur suivante est la version 2.4.4
- Après 6 itération de sprint, une version majeure est produit. Toute version majeure entraine une incrémentation du chiffre après le 1ier point et la disparition de la suite. Ex : la prochaine version majeure après 2.4.4.1 est 2.5.
- Tout changement important dans l'application entraine une incrémentation du premier chiffre et la disparition du reste. Ex : la prochaine version après la 2.4.4.1 est la version 3

Génération du Changelog

Un fichier nommé **chanlog.md** (ou un équivalent) devra exister à la racine de chaque projet. Ce fichier contiendra une nouvelle entrée à la fin de chaque nouvelle version afin de documenter tout ce qui a été ajouté dans cette nouvelle version tel que la résolution d'un bug, l'ajout de nouvelles fonctionnalités ou tout autre information pertinente sous forme de notes. Ceci permettra de savoir à tout moment ce qui a été ajouté sur une version.

Les entrées de ce fichier contiennent : la date, le numéro de version, et les modifications apportées. Les entrées les plus récentes doivent être au-dessus du fichier. Ci à après la nomenclature pouvant être utilisée pour les entrées :

<version x.y.z> <date>
<developer's name in charge of release> | <developer's email>
Features:<ticket/issue number>: <issue summary> (<link to issue tracker>)
…
Bugs fixed:<ticket/issue number>: <ticket/issue summary> (<link to issue tracker>)
…
Enhancements:<ticket/issue number>: <ticket/issue summary> (<link to issue tracker>)
…

Conclusion

La gestion des versions est un point important du cycle de développement logiciel et devrait être appliquée avec rigueur afin de gagner en temps et en efficacité dans les développements et les livraisons.