Cycle de développement Oxygen Cheat Sheet

by [deleted] via cheatography.com/122775/cs/22951/

1 - Création d'un workspace

Convention de nommage du workspace	Exemple
[feat|fix|doc|patch]--{description}-{JIRA}	feat-add-new-feature-OXY-0000
	fix-bug-of-at-creation-time-OXY-0000

Pour chaque tâche à traiter un workspace doit obligatoirement être crée.

feat : Nouvelle fonctionnalité
fix : Correction d'une anomalie
doc : Amélioration de la documentation
patch : Patch appliqué sur une branche de release*

La description doit permettre de donner une vision globale du développement qui sera effectué

* Cas particulier

4 - Tests unitaires

Développement terminé ? Ajustez vos test unitaires.
Pour une feat, ajoutez de nouveaux tests unitaires et vérifier que votre couverture de tests dépasse 80%.
Pour se faire retournez en phase **2 - Tests unitaires**

7 - Remise à niveau depuis development

Commande	git commit --amend

Le merge depuis development ne se fait que dans 2 cas :
- Bitbucket détecte un conflit
- la compilation de la feat échoue suite à un BC

Une fois le merge commité, il convient de revenir à la Phase **4 - Tests unitaires** .

2 - Tests unitaires

Pour toute tâche de type fix (anomalie constaté sur le produit) portant sur du code JAVA (service, util, action élémentaire, process, ...), la première chose à faire avant le développement d'un fix est la création d'un test unitaire qui reproduit le cas énoncé dans le jira.
Ce test unitaire doit être en erreur à l'exécution et valide le fait que l'erreur est bien reproductible.

5 - Tests runtimes

Ces tests sont manuels et ne nécessite pas de code. Ils servent à valider que le développement réagit bien avec l'ensemble des composants déployés sur le système.
Dans le cas ou des anomalies seraient identifié, il faudra retourner à la phase **2 - Tests unitaires** afin de corriger l'anomalie constatée.

8 - Prise en charge des commentaires

Toutes modifications de code suggérées par le commentaire d'un relecteur doit faire l'objet d'un commit respectant le nommage de la phase **3 - Développement**
Si le commentaire demande une modification du code, revenir à la phase **4 - Tests unitaires**

Tips

Référence	https://github.com/ineat/refcards/blob/master/git/FR.md

poussez la branche de feat le plus tardivement possible, si possible uniquement lors de la mise en PR. Cela permet de modifier facilement son historique git (amend de commit, rebase interactif, ...)

3 - Développement

Nommage des commits	{JIRA}: {desc1}
	{desc2}
	{...}
	{descn}

Les messages de commit doivent décrire se qui est fait dans le commit et non ce qui doit être fait pour la tâche.

En cours de développement, si la compilation échoue, il convient de mettre à jour la branche du repository concerné avec développement en suivant la procédure en Phase 7.

Une fois le développement terminé, retour en **2 -Tests unitaires**

6 - Mise en Pull Request

Commande	workspace pr

Modifier la pr repository principal pour inclure une description complète des modifications apportées.

Modifier la pr de tout repository contenant des modifications justifiant une explication.

9 - Merge sur la branche de développement

Convention de nommage du commit	[feat|fix|doc|patch](-{modules}): {Title}
	{Description}
	{JIRA}

Ce message doit être positionné lors de l'ouverture de la pop-up de merge par Bitbucket

By **[deleted]**
cheatography.com/deleted-122775/

Published 22nd May, 2020.
Last updated 22nd May, 2020.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com