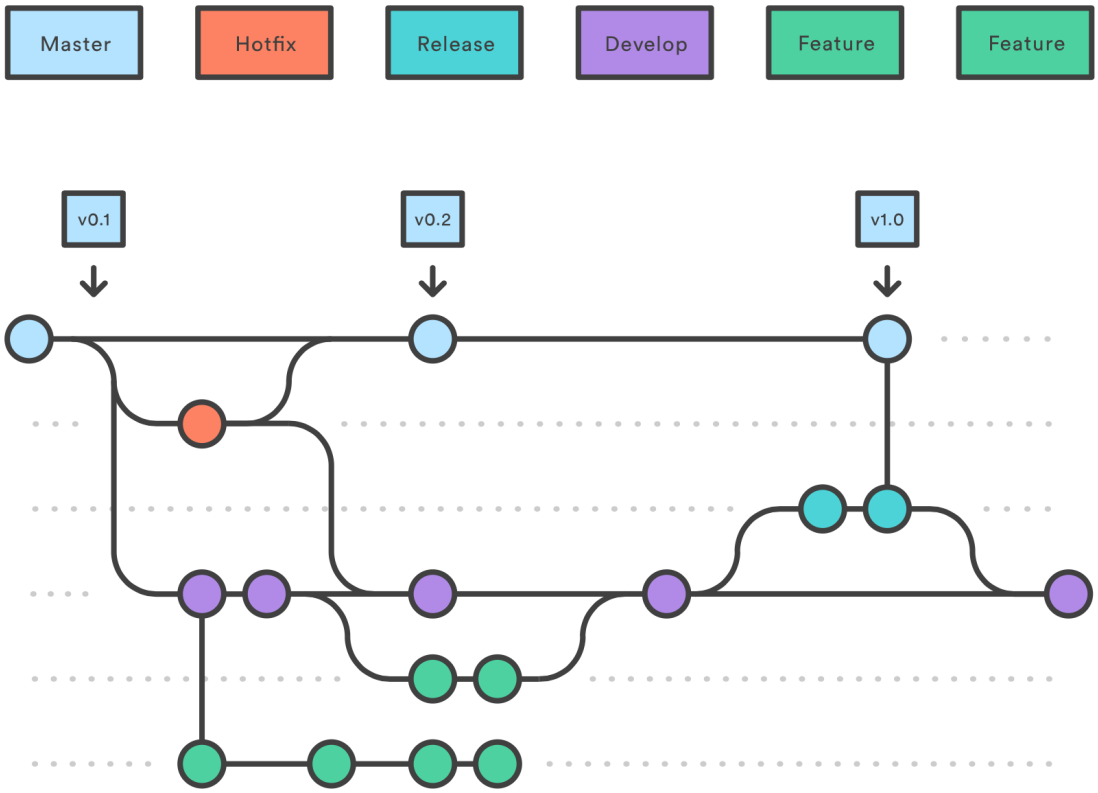
GitFlow / Git

Le workflow Gitflow définit un modèle de création de branche strict conçu autour de la livraison de projet. Les différentes branches ont des rôles très spécifiques, et le workflow définit comment et quand elles doivent interagir.

Les rôles des branches sont les suivants :

- Master : stocke l'historique officiel des version, contient le code déployé en production
- Develop : sert de branche d'intégration pour les fonctionnalités et est générée à la création du projet
- Feature : branche prévue pour réaliser une fonctionnalité. Elle est forkée depuis develop puis mergée quand la fonctionnalité est terminée. Pour relier la fonctionnalité a sa demande, le numéro de Jir@ peut être ajoutée dans le nom de la branche.
- Release : branche servant à la préparation d'une nouvelle version. Elle permet de différencier la future version des éventuelles fonctionnalités destinées a une version ultérieure. Une fois la version prête (fix, documentation,..) elle est mergée dans Master et Develop.
- HotFix : branche dédiée à une correction d'incident de production. Une fois la correction appliquée sur le Master, elle doit également être appliquée sur la branche Develop



Tutoriel détaillé du fonctionnement de gitflow : https://www.atlassian.com/fr/git/tutorials/comparing-workflows/gitflow-workflow

Commande git	Détails

git config	Cette commande permet de paramétrer les configurations de git comme par exemple pour paramétrer le nom et l'adresse mail dans git :
	
	`git config --global user.name "Hello World"`
	`git config --global user.email "hello.world@lcl.fr"`

git clone	Permet de cloner un répertoire distant dans un répertoire en local. `git clone git@scm.saas.cagip.group.gca:lcl/wip/templates/framework-lcl/WAR_TOMCAT.git`

git add	Permet d'ajouter le contenu indiqué à l'index.
	
	`"git add ."  ou "git add -A" (ajoute toutes les fichiers modifiés)`
	`"git add <chemin>/<fichier> (ajoute le fichier indiqué)`

git commit	Permet d'enregistrer dans le dépôt les fichiers qui ont été ajouté à l'index par la commande précédente. L'utilisation de l'option "-m" à la suite de la commande permet l'ajout d'un commentaire lié à votre commit. `git commit -m "add unitTest"`
git rebase	Cette commande permet de changer la base de la branche. Dans le schéma d'exemple, au début, on a crée une branche feature à partir de la branche master. Mais au fur et à mesure du temps, des commits et mise à jour ont été implémentés dans le master. Et l'on souhaite que la branche feature reste à jour quant aux modifications faites dans la branche master, on utilise alors la commande rebase qui met à jour la base de features mais en y ajoutant les commits fait dans le master, donnant l'illusion que la branche a été créer à partir d'un commit différents. blocked URLgit rebase master (permet de mettre à jour la base dans la branche feature)

git checkout	Permet de basculer sur une autre branche git, en mettant à jour les fichier de l'index , elle permet aussi de créer de nouvelle branche en ajoutant l'option "-b".
	
	`git checkout -b <nouvelleBranche> (permet de créer une nouvelle branche à partir de celle actuelle)`
	`git checkout <nomBranche> (bascule sur la branche indiqué)`

git push	Permet d'exporter les modifications contenu dans des commits sur le répertoire local vers le répertoire distant. Le push est susceptible d'écraser les changements. Vous devez donc prendre des précautions lorsque vous l'exécutez. `git push <remote> <branch>`
git merge	Fusionne deux ou plusieurs historiques de développement ensemble (souvent des branches) Cette fusion peut entrainer des conflits qui devront être réglés avant de pouvoir finaliser le merge. git merge --abort (annule le processus de merge après un merge conflictueux. L'état initial sera tenté d'être restauré, mais si des modifications non commitées étaient présentes, cet état initial peut ne pas être atteint) git merge --continue (continue le processus ce merge après un merge conflictueux et une correction de ces conflits

git diff	La commande diff prend deux ensembles de données qu'elle compare pour indiquer les différences. Cette commande peut aussi bien comparer des fichiers que des branches d'un répertoire.
	
	`git diff --git a/fileA b/fileB (compare des fichiers)`
	`git diff branche1 branche2 (compare des branches)`

git status	Permet d'afficher l'état du répertoire de travail en indiquant les modifications qui ont été effectués et si elle ont été ajoutés ou non à un future commit avec la commande add. Les modifications sont triés selon 3 catégories : "changes to be commited" ce qui veut dire qu'elles ont déjà été ajoutés et attendent d'être commit, "changes not staged for commit" ce sont les modifications qui n'ont pas été ajouté et qui ne seront pas commit, et enfin "Untracked files" qui sont les modification qui sont ignorés.
git log	Affiche les journaux de validation
git show	Affiche des objets (commits, étiquettes, arbres,..)
git pull	Rapatrie un dépôt ou une branche distante dans la branche actuelle
git remote	Gestion des dépôts suivis git remote (liste les dépôts distants) git remote add (ajoute un dépôt distant) git remote remove (retire un dépôt distant)
git branch	Gestion des branches git branch (liste les branches locales) git branch <MaNouvelleBranche> (crée une nouvelle branche, option -d pour supprimer la branche)
git tag	Gestion des tags, par exemple de version. Il est préférable d'utiliser des tags annotés qui permettent de profiter des métadonnées associées à ceux-ci (message de tag, créateur, date de création..) git tag (voir la liste des tags) git tag <nomDeMonTag> (crée un tag) git tag -a <nomDeMonTag> [-m "my version 1.4"] (crée un tag annoté accompagné d'un message de tag) git tag -a <nomDeMonTag><empreinteSHACommit> (applique un tag a un commit)
git rm	Supprime des fichiers de l'arbre de travail

git stash	Remettre au propre un repo "sale" en enregistrant les modifications réalisées git stash (remise immédiatement les modifications et rétablit l'état du commit HEAD) git stash push [-m "Stash de ma super branche"] (remise les modifications mais permet l'ajout d'options comme le nommage, la précision des fichiers concernés, etc) git stash list (Liste des stash précédemment réalisés) git stash show (voir les détails d'un stash précis) git stash apply (appliquer un stash)
git fetch	Récupérer les objets et références depuis un ou plusieurs dépôts distants git fetch (récupère les branches et leur contenu de origin) git fetch --all (récupère le contenu de tous les dépôts distants)
git grep	Permet d'effectuer une recherche simple dans l'arborescence ou dans le répertoire de travail. L'ajout de l'option "-n" permet d'indiquer les numéros des lignes correspondantes. "–count " permet de compter le nombre d'occurrence. `git grep -n <Recherche>`
git help	A l'aide

Git Commit Template

- Afin d'uniformiser le contenu du message de commit, il est nécessaire d'utiliser le fichier template suivant (le stocker sur P:\ ou dans le répertoire de votre profil) :

- Il faut configurer votre client git pour utiliser ce template avec la commande :

```
git config --global commit.template [CHEMIN_VERS_FICHIER_TEMPLATE]\gitcommittemplate.txt
```