FORMATION

GIT

27/10/2023

Glodie TSHIMINI
contact@tshimini.fr

2iTECH Academy by M2i

PLAN

PLAN DU COURS

Plan

I. Git, GitHub et GitLab
II. Commandes de bases
III. Commandes avancées
IV. Framework Git

Annexe

- Convention nommage commits Angular
- Aller plus loin avec les commandes Unix
- Terminologie

2iTECH
Academy by M2i

INSTALLATIONS

Environnement de travail

À installer ou créer uniquement si vous n'avez pas encore les éléments ci-dessous

1. Installez Git
   - Laisser les paramètres par défaut
   - Vous pouvez modifier l'éditeur
2. Installez Visual Studio Code
3. Créez un compte sur GitHub

I. GIT, GITHUB ET GITLAB

Qu’est-ce que *Git* ?

- Système(logiciel) de gestion de versions
- Permet
  - Plusieurs versions du projet
  - Plusieurs collaborateurs travaillent sur le même projet
  - Historique du projet
- Repose sur
  - Snapshots (sauvegarde de l’état du dépôt à un moment précis)
  - Commits
  - Dépôts
  - Local
  - Distant
- Système décentralisé
- Plusieurs dépôts locaux liés à un dépôt distant

Qu’est-ce que *GitHub* ?

- Hébergeur en ligne des dépôts *git*
- Crée en 2008
- Racheté par Microsoft en 2018

- Travail collaboratif
  - Entreprises
  - Écoles
  - Open Source

- Plusieurs offres
- Gratuites (offre étudiante)
- Payantes

Qu'est-ce que *GitLab* ?

- Concurrent de *GitHub*
- Crée en 2011
- Propose plus de fonctionnalités
  - Possible d'héberger ses dépôts distants sur son propre serveur privé.
  - Définition des rôles des utilisateurs (documentation permissions)
    - Guest
    - Developer
    - Maintener
  - CI/CD
  - Etc.

Fichiers spéciaux de GIT

README.md

Documentation du projet

.gitkeep

Conserver un dossier vide

.gitignore

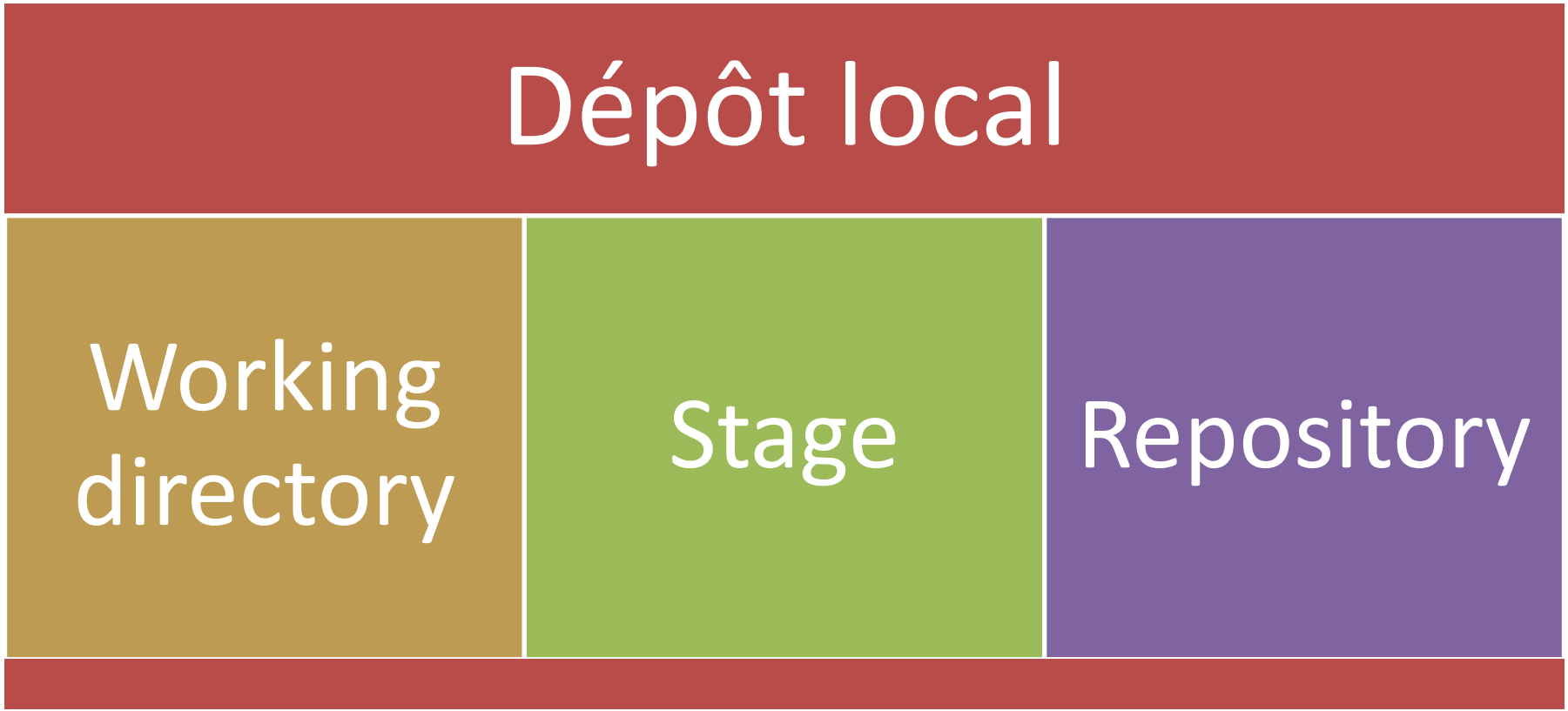
Liste fichiers et dossiers à ignorer

Fichier .gitignore

.gitignore

```
# Cache, temp and personal files

/.htaccess
*.log
npm-debug.log.*
.sass-cache/
/cache/*

#/img/*
/log/*
/upload/*
docs/phpdoc-sf/
#composer.lock
tests/Selenium/errorShots/
tests/Selenium/errorDumps/


admin134ntao8l/autoupgrade/*
admin134ntao8l/backups/*
admin134ntao8l/import/*

admin134ntao8l/import/*
!admin134ntao8l/import/.htaccess
!admin134ntao8l/import/index.php

themes/*/cache/*
```

3 sections de travail du dépôt de travail



II. COMMANDES DE BASE

Aide sur une commande git

git {cmd} --help

- *{cmd}* à remplacer par une des commandes git que nous verrons tout au long de ce cours.
- La documentation complète sur la commande s'ouvrira via une page web ou une autre sortie selon la configuration de votre logiciel git.

Configurer et initialiser un dépôt git

Configuration minimale de git

- Au minimum
  - Nom et prénom
  - Email

```
Glodie@Glodie MINGW64 ~
$ git config --global user.name 'Glodie'

Glodie@Glodie MINGW64 ~
$ git config --global user.email 'contact@tshimini.fr'
```

- Autres
  - Editeur
  - Couleurs
  - Format de l'aide

- À faire une seule fois sur votre ordinateur.
  - l'option global permet de vous identifier sur tous vos dépôts git de votre machine avec les informations fournies.

Créer un dépôt local

```
Glodie@Glodie MINGW64 /d/demo
$ git init
Initialized empty Git repository in D:/demo/.git/

Glodie@Glodie MINGW64 /d/demo (master)
$ ls -lah
total 44K
drwxr-xr-x 1 Glodie 197121 0 avr.  19 19:04 ./
drwxr-xr-x 1 Glodie 197121 0 avr.  19 19:04 ../
drwxr-xr-x 1 Glodie 197121 0 avr.  19 19:04 .git/
```

- Initialise un dépôt git
- Crée un dossier caché .git

- Une fois le dépôt initialisé sur un répertoire de travail à l'aide de cette commande, vous n'avez plus besoin d'utiliser cette commande sur le projet en cours, hormis si vous avez supprimé le dossier caché .git

- **Soyez vigilant à l'emplacement du dossier où vous allez initialiser votre dépôt.**

2iTECH Academy by M2i

EXERCICE

2iTECH
Academy by M2i

LIAISONS DES DÉPÔTS

Ajouter un dépôt distant git

- Synchroniser avec un dépôt distant (*GitHub*, *GitLab*, etc.)

git remote add origin {URL}

- Remplacer *{URL}* par l'*URL* du dépôt distant
- *Origin* est un alias de l'*URL*

Cloner à un dépôt distant git

- Cloner : Créer une copie d’un projet distant (depuis un dépôt distant) en local

git clone {URL}

- Remplacer {URL} par l’URL du dépôt distant

Récapitulatif de git en local

- Soit je fais seulement un *git clone* ou soit je fais *git init* puis un *git remote add* , mais jamais les 3 ensembles.
- **Après un *git clone*, il est inutile de faire un *git init* et/ou un *git remote add*.**

- Remplacer {URL} par l'URL de votre dépôt distant

Pas de dépôt local	Existence dépôt local et dépôt distant vierge	Pas de dépôt local et existence d'un dépôt distant
git init	*git remote add origin {URL}*	*git clone {URL}*

2iTECH Academy by M2i

Les branches

Gérer les branches d’un dépôt distant

Créer	Se déplacer sur une branche	Créer et se (dé)placer directement sur la nouvelle branche
`git branch main`	`git checkout main`	`git checkout -b feature/user`

Gérer les branches d'un dépôt distant

- Dans la liste de branches , * indique la branche courante

Lister toutes les branches	Renommer une branche	Supprimer une branche
`git branch`	`git branch -m old_name new_name`	`git branch -d feature/user`

```
Glodie@Glodie MINGW64 /e/formations/coderbase/poei-java-salesforce/1-git (main)
$ git branch
  feature/ex1-3
  feature/ex4
  feature/ex5
  feature/ex6
  feature/pratical-work/conflicts/merge/dev2/dev1
  feature/pratical-work/dev1
  feature/pratical-work/dev2
  feature/pratical-work/fix/conflicts/dev2/dev1
  feature/pratical-work/owner
* main
```

- Anglais
- Branche principale
    - nommée par défaut ***main*** (anciennement *master*)
    - Protégé
- Ne jamais travailler directement sur la branche main ou master
- Supprimer les branches inutiles après avoir effectué la fusion

2iTECH Academy by M2i

EXERCICE

2iTECH Academy by M2i

Versionner son code

Ajouter fichiers/dossiers dans l'index

- **Attention** avec l'option *--all*, soyez vigilant sur les fichiers qui seront ajoutés dans l'index après l'exécution de cette commande.

Individuel	Tous (modifiés, supprimés et nouveaux)
`git add fichier.txt dossier/`	`git add --all`

Commiter

En saisissant directement le message du commit sur ligne	À l'aide de l'éditeur configuré pour git (permet de saisir sur plusieurs lignes)
git commit -m "first commit"	***git commit***

- Avec l'éditeur
- Pour passer en mode insertion ***Echap + i***
- Insérer votre message de *commit*
- Pour sortir et enregistrer votre message
  - ***Echap***
  - ***:x Entrez***

Commiter sans modifier l'historique

Sans modifier le message du dernier commit	En modifiant le message du dernier commit
`git commit --amend --no-edit`	`git commit --amend`

- *--amend* permet de modifier le message du dernier commit
- *--amend --no-edit* va affecter les fichiers dans l'index au précédent commit

Bonnes pratiques pour le commit

- Anglais
- Pas de caractères spéciaux
- Motif des modifications

git commit -m ‘refactor: update contact form, add GPDR requirements’

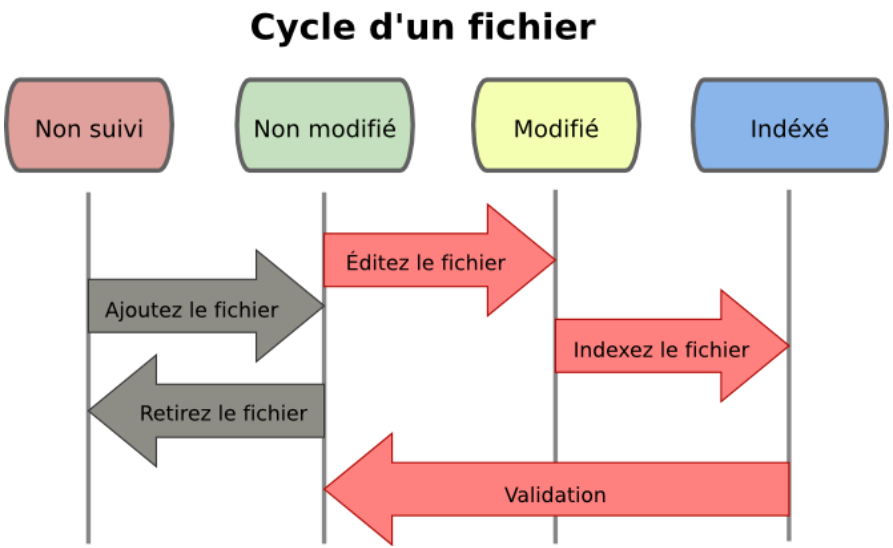
- Il existe des conventions de nommage comme par exemple celle d’Angular

2iTECH Academy by M2i

EXERCICE

État de l’historique du dépôt

Source image developpez.com



- *Untracked*
  - Non versionné
- *Unmodified*
  - Versionné
- *Modified*
  - Versionné et modifié en attente d'être indexé
- *Staged*
  - Prise en compte lors d'une nouvelle version

État du dépôt

```
Glodie@Glodie MINGW64 /e/formations/coderbase/cda_2itech/0-interns/1-intro_git (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        0-guide_installation/
        1-exercices/
        2-evaluation/
        README.md

nothing added to commit but untracked files present (use "git add" to track)
```

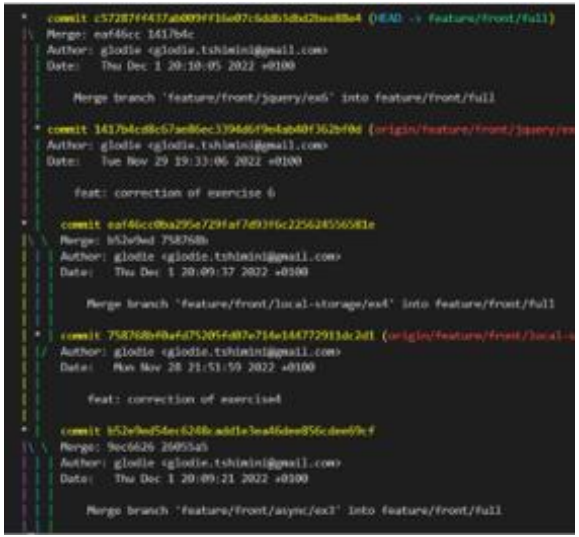
git status

- Fichiers et/ou dossiers
- Modifiés
- Supprimés
- Nouveaux (à ajouter dans l'index)
- Ajoutés (présents) dans l'index

Historique des commits

Détaillé	Sur une ligne	Graphique
git log	***git log --oneline***	***git log --graph***

```
commit b161417b432e4515f0e3b27e4e757bc35bd84739 (HEAD
Author: Glodie <contact@tshimini.fr>
Date:   Wed Dec 7 15:50:04 2022 +0100

    feat: exercise 7 bis interns proposals

commit 01c0bd16fa4178c3fca78fccd500b3dd677c8531
Author: Glodie <contact@tshimini.fr>
Date:   Wed Dec 7 14:28:47 2022 +0100

    feat: correction of exercise 7 both AlgoBox & JS

commit b3f23ab3fdb9c473b30eaa4a64831de0f995e572
Author: Glodie <contact@tshimini.fr>
Date:   Wed Dec 7 11:52:13 2022 +0100

    feat: basics JS

commit 43c1a6c7694248f644ff31dabdcdb515774b3568
Author: Glodie <contact@tshimini.fr>
Date:   Wed Dec 7 09:04:02 2022 +0100
```



Différences et traçabilité

- Remplacer *{id1}* et *{id2}* par les identifiants des commits

Différences entre 2 commits	Différences détaillés avec la précision des auteurs des modifications
`git diff {id1} {id2}`	`git blame index.html`

2iTECH Academy by M2i

EXERCICE

2iTECH
Academy by M2i

Synchroniser un dépôt local avec le distant

Envoyer vos modifications vers un dépôt distant

- Pusher (pousser) du dépôt local vers le dépôt distant GitHub

```
git push origin main
```

Récupérer les mises à jour sans fusionner

- Récupérer en local les MAJ du dépôt distant GitHub sans fusionner

```
git fetch
```

Fusionner 2 branches avec un merge

Fusionner 2 branches	Annuler la fusion
git merge feature/newsletter	*git merge --abort*

1. Se placer sur la branche de réception
2. Exécuter la commande git merge
3. Cette commande peut créer un commit de fusion automatique

Fusionner 2 branches avec un pull

1. Se placer sur la branche de réception
2. Exécuter la commande *git pull*

PULL = FETCH + MERGE

Gestion des conflits

- Quand ?
  - Fusion des branches
- Pourquoi ?
  - Modification du même fichier aux mêmes endroits dans les 2 branches
- Comment résoudre le problème ?
  - Choisir la version à conserver avec ces collaborateurs
    - Version de la branche 1
    - Version de la branche 2
    - Les 2 versions
    - Une résultante
- Commiter la résolution du conflit

Gestion des conflits (source image Atlasian)

```
$ cat merge.txt
<<<<<<< HEAD
this is some content to mess with
content to append
=======
totally different content to merge later
>>>>>>> new_branch_to_merge_later
```

Démonstration

2iTECH Academy by M2i

EXERCICE

Remisage

- Mettre les modifications en cours de côté

```
git stash
```

- Voir la liste des remisages

```
git stash list
```

- Récupérer les modifications mises de côté et les supprimer de liste de remisage

```
git stash pop
```

- Récupérer les modifications mises de côté et les garder dans la liste remisage

```
git stash apply
```

Les étiquettes sur les commit (tags)

- A partir de l'identifiant d'un commit

```
git tag v1.0.0 f3089fe5
```

- Du dernier commit

```
git tag v1.0.1
```

- Annoter avec un message

```
git tag v2.0.0 -m "Message"
```

- Liste des tags

```
git tag –list
```

- Détails d'un tag

```
git show v1.0.0
```

III. COMMANDES AVANCÉES

Revenir en arrière avec *git checkout*

- Sans modifier l'historique (en tant que simple observateur)

État du dépôt tel qu'il était à partir d'un commit	Fichier tel qu'il était à partir d'un commit	Fichier tel qu'il est au niveau du HEAD (pointeur)
git checkout {id}	*git checkout {id} file.txt*	*git checkout file.txt*

- Pour quitter le mode spectateur, utilisez la commande

git checkout {name_of_current_branch}

- Remplacer {id} par l'identifiant de votre commit

Revenir en arrière avec git reset

	git reset {id} --soft	*git reset {id} --mixed*	*git reset {id} --hard*
Supprime les commits après l'id indiqué	Oui	Oui	Oui
Conserve les modifications effectuées	Oui	Oui	Non
Les fichiers dans l'index	Oui	Non	Non

- Modifie l'historique
- **À utiliser avant une publication** (*push*)
- Après avoir effectué un *push*, vous ne devez pas utiliser la commande *reset* pour modifier votre historique, c'est trop tard.
- Attention avec l'option ***--hard***, toutes les modifications survenues après l'***id*** du ***commit*** indiqué seront perdues

Revenir en arrière avec *git revert*

- Après avoir effectué un *push* et **sans modifier l'historique**
- La commande créée un commit de « revert » de l'annulation du commit concerné
- Elle se sert à retirer les modifications introduites par un commit (défaire)

git revert {id}

- Remplacer *{id}* par l 'identifiant du commit

2iTECH Academy by M2i

EXERCICE

IV. FRAMEWORK GIT

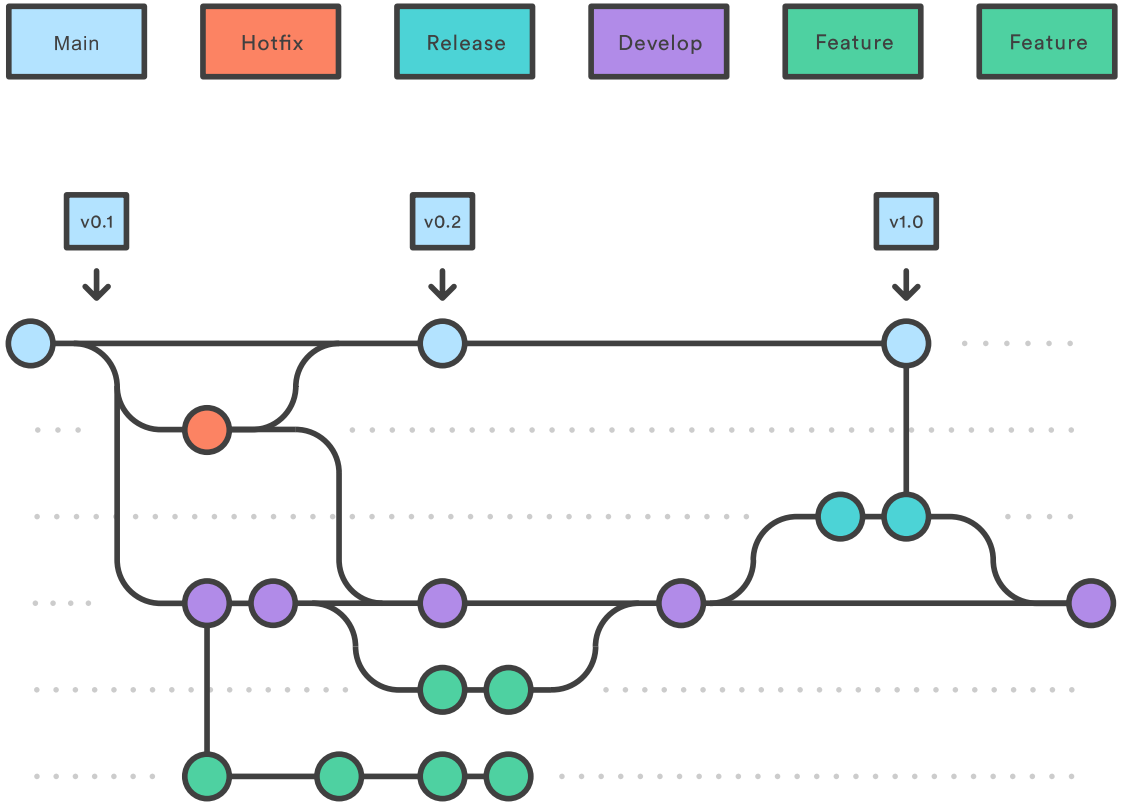
Qu’est-ce qu’un framework ?

- Cadre de travail
- Avantages
  - ✓ Gains en productivité, qualité et sécurité
  - ✓ Appliquer les bonnes pratiques
  - ✓ Applications plus robustes
- Inconvénients
  - X Apprentissage
  - X Applications plus lourdes

Framework *git flow*

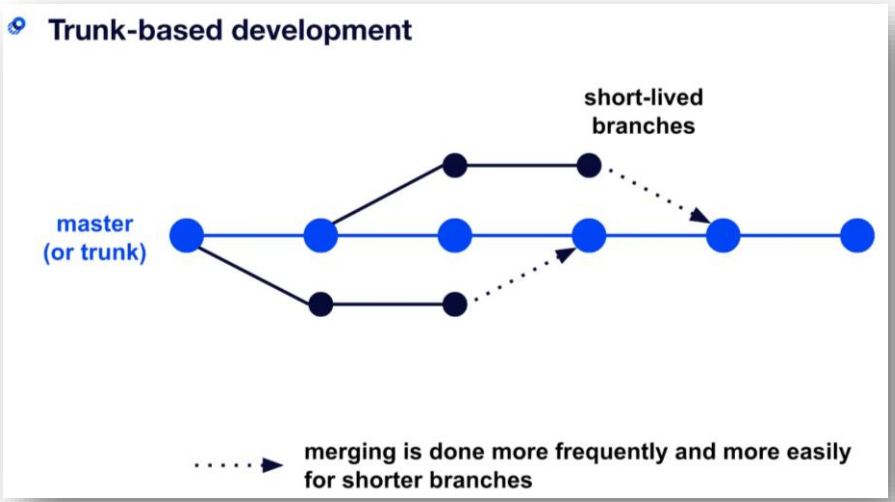
- **2 branches principales**
  - La branche ***main*** : fonctionnement habituel
  - La branche ***develop*** : sert de branche d'intégration pour toutes les **modifications. Elle est créée à partir de la main**
- Pour chaque besoin, une branche spécifique est créée à partir de la branche *develop*.
  - Pour les **fonctionnalités** (la branche des ***features)***, le nom commence par ***feature/xxx***.
  - Pour les **livraisons** : elle commence par ***release/xxx***.
    Elle est mergée dans ***main*** et ***develop***).
  - Pour la maintenance : ***hotfix/xxx*** (créée à partir de la branche *main*, elle est mergée dans la *main* et *develop*).

Gitflow résumé (source image Atlassian)



Framework *trunk-based development*

Source image optimizely



- 1 seule branche principale nommée *master* ou *main* ou *trunk*.
- C'est codé, on commit et on livre (*merge* immédiatement sur la branche principale).
- Anticiper les problèmes de fusion en amont avant la livraison finale

Framework *trunk-based development*

- Requiert des process fiables et solides au niveau de la qualité du code
- Tests automatisés
- Cycle de développement très court (tâche)
- Développement agile (*TDD*, *Code Review*)
- Développer en tenant compte de l'existant

ANNEXE

2itechacademy.com

Annexe

Convention de nommage des commits

Convention nommage commits Angular

Pour aller plus vite, commandes Unix

Commandes de base console

TERMINOLOGIE

Terminologie *git*

- ***Repository*** (dépôt) : dossier de travail contenant les fichiers et dossiers d'un projet.

- ***Init*** : initialiser un dépôt git en local.

- ***Commit*** : action d'enregistrer vos modifications dans l'historique du projet. Il est accompagné d'un commit message.

- ***Status*** : visualiser l'état du dépôt local.

Terminologie *git*

- ***Push*** : permet d'envoyer les modifications commitées du dépôt local vers le dépôt distant.
- ***Pull*** : récupérer les commits depuis le dépôt distant (ex: GitHub) en local.
- ***Clone*** : récupérer un dépôt distant en local.
- ***Checkout*** : visualiser ou revenir en arrière, également créer une branche et s'y déplacer.
- ***Log*** : visualiser l'historique des commits.

Terminologie *git*

- ***Merge*** : action de fusionner deux versions (branches) d'un projet.
- ***Fetch*** : action de récupérer les modifications du dépôt distant en local sans effectuer directement la fusion.
- ***Conflicts*** (conflits) : modification(s) effectuée(s) sur les mêmes lignes d'un fichier dans 2 branches distinctes qui doivent être fusionnées.
- ***Branch*** : version alternative du projet.
- ***Diff*** : voir les différences entre 2 commits.

Terminologie *git*

- ***Rebase*** : action de rembobiner pour modifier des *commits* précédents ou de mettre les *commits* d'une branche à la suite d'une autre *branche* (les 2 *branches* doivent avoir un ancêtre (*commit*) en commun). Pour ce dernier point, c 'est l'équivalent d'une fusion.

- ***Tag*** : attribuer un numéro de version avec un message à un *commit*.

- ***Mv*** : déplacer ou renommer un fichier.

- ***Rm*** : supprimer un fichier.

MERCI DE VOTRE ATTENTION ET PARTICIPATION

Glodie Tshimini
contact@tshimini.fr