# HowDoWeDoGit

- /!\ **On ne fait JAMAIS git push -f sur la branche main.** /!\

Travailler dans une branche dont le nom suit le convention de nommage **wip/XXXXX-mini-description**, où XXXXX est le numéro de ticket où le développement est suivi.

```
git checkout -b wip/XXXXX-mini-description
```

Une fois les commit faits dans la branche, pousser pour permette à Jenkins d'exécuter les tests :

```
git push origin wip/XXXXX-mini-description
```

Une fois la PR créée pour la branche, les tests seront exécutés par Jenkins et un mail vous sera envoyé en cas d'erreur.

- séparer les patchs de traduction des patchs de code
- pareil, ne pas mêler des modifications de style (genre réindentation ou autre) dans un patch ajoutant une fonctionnalité/corrigeant un bug.

- tagguer pour que ça puisse être déployé en recette :
  - git tag -a vX.Y
  - git push origin vX.Y
  - aller dans jenkins forcer le build
    - https://git.entrouvert.org/entrouvert/misc-fred/src/branch/main/bin/tag est un script qui emballe tout ça

- Gérer des release de branche stable, → HowDoWeDoHotfixes

- Sur la forme des messages de commit, http://tbaggery.com/2008/04/19/a-note-about-git-commit-messages.html
  - Modulo la première ligne il y est écrit "max 50 caractères", ça peut être 80.  Et pas nécessairement de majuscule au début.
  - Et inclure une référence au ticket, façon (#12345) par exemple : doc: explain how we are supposed to commit (#12345)
    - Le message du commit doit être en anglais, au présent.

```
Describe your changes in imperative mood, e.g. "make xyzzy do frotz"
instead of "[This patch] makes xyzzy do frotz" or "[I] changed xyzzy
to do frotz", as if you are giving orders to the codebase to change
its behaviour.
 -- http://git.kernel.org/cgit/git/git.git/tree/Documentation/SubmittingPatches
```

## Gitea

On utilise gitea, installé ici : https://git.entrouvert.org/

Bugs et limitations connues :

- texte tronqué à l'emploi de touche morte, Agate a mis en place une option; c'est de l'optin, il faut lancer localStorage.setItem("useTextarea", 'true') au moins une fois dans sa console navigateur pour que ça soit affiché, ça permet de continuer à utiliser l'éditeur plus complet si on le souhaite. (https://dev.entrouvert.org/issues/73988#note-4).

## Conseils & astuces git

- dans ~/.gitconfig:

```
[color]
    branch = auto
    diff = auto
    interactive = auto
    status = auto
[pull]
    rebase = true
```

- git log
- git show (avec la coloration syntaxique activée plus haut, ça permet de voir les espaces qui trainent, par exemple)
- git add -p

- travail dans une branche, git rebase main avant de merger
  - git cherry-pick aussi, peut-être

- travail dans main, git rebase -i origin/main, taper les commits qu'on veut en premier
  - git push origin hash:main
  - ou second git rebase -i, effacer les commits en trop, noter le hash
    - git push; git merge hash

- retrouver l'état d'une branche avant un rebase, un merge, un git --amend: git reflog , une fois retrouvé le bon commit on y retourne de façon non destructive par git reset <commit> ou destructive (on perd les différences) par git reset --hard <commit>

- revenir sur le le dernier commit pour le découper (git reset HEAD^)

- ajouter les modifications en cours au dernier commit (git commit -av --amend)

- réorganiser manuellement les n (ici 5) derniers commits (git rebase -i HEAD~~~~~, ou git rebase -i HEAD~5)

(?) http://blogs.gnome.org/bastian/2015/02/12/a-guide-to-git-in-gnome-for-the-simple-minded/

- pour obtenir 8 lignes de contexte dans les patchs, ajout à ~/.gitconfig:

```
[diff]
  context = 8
```

- Pour merger des fichiers .po sans se faire chier: https://gist.github.com/mezis/1605647

- Certains dépôts présentent des commits annotés (voir git-notes). Ces notes ne sont pas tirées par défaut. Pour ce faire il suffit d'ajouter la référence au remote correspondant, dans le fichier de configuration git du dépôt. Par exemple, dans l'entrée [remote "origin"], y ajouter :

```
    fetch = +refs/notes/*:refs/notes/*
```

## Porcelaine git pour notre workflow (avant gitea)

J'ai créé ça :

> https://pypi.org/project/git-redmine/

C'est sympatoche:

- pip install --user git-redmine
- poser ça dans ~/.config/git/config:

```
[redmine]
url·=·https://dev.entrouvert.org/
key·=·<apikey à récupérer sur redmine sur la page de votre utilisateur>
```

- git redmine project set <redmine-project-slug> : associe le dépôt git à un projet
- git redmine issue new : ouvre un $EDITOR pour décrire un nouveau ticket, obtient un numéro et crée la branche wip/XXXX-description pour commencer à bosser
- git redmine issue take XXXX : prendre un ticket existant pour bosser
- git redmine rebase : fais un pull sur main puis vérifie que la branche courante rebase sur main
- git redmine issue submit : fait un rebase comme juste avant, puis si ça passe poser les patchs dans le ticket en passant le ticket en Solution Proposée (ça demande si on veut changer le status et s'affecter le ticket)
- git redmine merge-and-push : pull un main récent, vérifie qu'un rebase de la branche est possible, puis merge la branche, supprime une éventuelle branche distante si créée pour jenkins, pousse sur origin/main puis demande s'il faut supprimer la branche locale