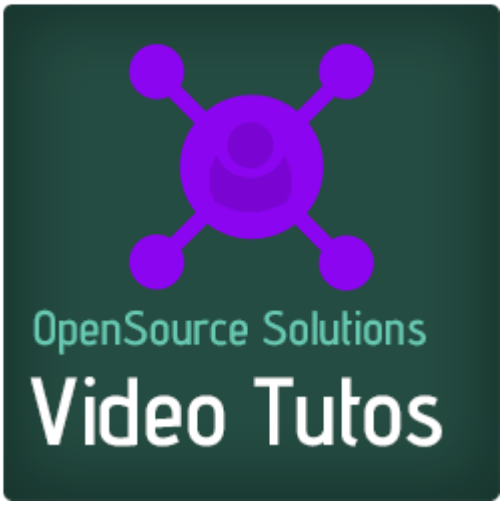
Tiki et Git, fork et merge request

Tiki et Git, fork et merge request

 Bernard Sfez - 2020-06-12 21:10



Dans ce tutoriel Tiki Express, nous allons voir ensemble :

- Comment créer votre projet sur GitLab
- Une explication rapide sur la sécurité et GitLab
- Comment mettre en miroir (mirror) le dépôt Tiki dans votre projet
- Une vue d'ensemble du workflow
- Comment installer un Tiki en local
- Comment créer votre branche
- Comment faire un commit de votre code
- Comment le pousser (push) dans votre fork
- Comment sélectionner votre branche par défaut
- Comment créer une demande de fusion (merge request) sur le dépôt Tiki

Bonjour Internet, je suis Bernard Sfez, spécialiste Tiki, et dans ce tutoriel nous allons voir ensemble comment travailler avec Tiki et Git simplement, en utilisant votre fork et en créant une merge request.
Ce tutoriel s'adresse aux développeurs, débutants ou expérimentés, aux designers, aux traducteurs, ou à toute personne souhaitant contribuer au projet Tiki.
Accrochez-vous, ce sera un tutoriel rapide.

Et tout d'abord, j'espère que vous êtes tous en sécurité et que tout va bien pour vous et votre famille.

Ensuite, je dois m'excuser de ne pas avoir pu publier plus tôt ce tutoriel Tiki Express.
Comme vous l'avez vu dans la pré-intro, j'ai essayé plusieurs fois de le démarrer, mais tellement de choses se sont mises en travers de mon chemin que je ne pouvais rien laisser de côté pour le terminer, et c'est comme ça.

J’ai également dû changer le sujet de la vidéo comme annoncé dans le tutoriel précédent, pour une version plus courte, mais pas d’inquiétude : il y en aura bientôt une sur le plug-in custom search dans Tiki.

Maintenant, quelques nouvelles rapides.

Tiki 21 est sorti et c’est une version formidable. Consultez les détails sur https://doc.tiki.org/tiki21. Et vous l’avez deviné : le lien sera dans la description ci-dessous.

Un Tikifest arrive et cette année ce sera une réunion virtuelle. Je n’ai pas besoin de vous dire que tout ce qui implique des déplacements est très compliqué, mais Tiki est un outil parfait, une application parfaite pour la collaboration en ligne et le travail à distance.
La communauté Tiki a donc organisé et invité à participer et contribuer à notre Tikifest virtuel 2020. Consultez les détails — encore une fois, lien dans la description, je ne le redirai pas.
https://tiki.org/TikiFestVirtual2020

Enfin, et ce n’est pas le moindre, suite aux demandes, souhaits et discussions sur la manière dont je peux optimiser l’aide que je peux apporter à propos de Tiki, j’ai décidé d’organiser une discussion ouverte hebdomadaire.
C’est une opportunité de partage d’informations non scénarisée, et si vous voulez recevoir mes notifications de meet-up, vous devrez aller sur mon site et
https://www.bsfez.com/Welcome#contact|vous abonner à la newsletter], pas d’autre moyen.

Et maintenant, on commence.

Tiki est une application open source et existe depuis 2004 avec un modèle original qui nous a maintenus au sommet des CMS, du Wiki Web, des plateformes collaboratives et des générateurs d’applications web.
Tiki est gratuit et tout le monde peut rejoindre la communauté et commencer à contribuer immédiatement, à condition de suivre nos trois règles (https://dev.tiki.org/3-Rules) :
*Respecter l’environnement,
*Commiter tôt, commiter souvent,
*Rendre optionnel,

Pour soutenir Tiki, nous utilisons Git et un dépôt Tiki sur Gitlab.
La communauté Tiki a décidé d’utiliser Gitlab pour développer le projet Tiki, ce qui signifie qu’il stocke et distribue aussi les fichiers du projet dans un dépôt. Pour pouvoir suivre correctement ce travail, vous devez avoir un compte Gitlab ou en créer un sur Gitlab.com.
(https://dev.tiki.org/Git-workflow-For-Collaborators)

Une explication rapide sur la sécurité et Gitlab

Ouvrons une discussion sur la sécurité sur Gitlab.

Une chose importante concernant votre compte et votre projet est de les garder en sécurité et, la plupart du temps, privés. Cela se fait via les paramètres de sécurité de vos comptes. Gitlab a des préoccupations sérieuses et justifiées concernant la sécurité et les mots de passe. Je vous suggère d’enregistrer votre clé SSH en suivant la documentation Gitlab et d’activer l’authentification à double facteur. Je vous recommande vivement de prendre le temps de lire leur

manifeste sur les pratiques de sécurité. (https://about.Gitlab.com/handbook/security/)

Comment mettre en miroir le dépôt Tiki dans votre projet

Une fois votre compte créé, créez un projet afin de pouvoir créer votre fork Tiki et le mettre en miroir avec le dépôt Tiki original. De cette façon, votre projet sera constamment mis à jour à partir du dépôt Tiki et vous travaillerez toujours sur des fichiers à jour. Sur Gitlab, allez dans setting repository et dépliez "mirroring repositories". Depuis ici, copiez/collez l'URL git du projet Tiki. (https://Gitlab.com/tikiwiki/tiki.git)

Activez "overwrite divergent branches" et sauvegardez.

Vue d'ensemble du workflow

Vous avez peut-être déjà quelques inquiétudes.
Et si vous modifiez des fichiers, ne seront-ils pas écrasés ?
Avec le nouveau code arrivant depuis Tiki ? Seront-ils fusionnés ?
Mes modifications sont-elles renvoyées vers le dépôt Tiki ? Etc.

Même s'il y a plus d'une raison de travailler avec Git, je vais expliquer ici notre modèle communautaire et, en fait, la raison pour laquelle vous travaillez sur votre fork et créez votre branche afin de soumettre votre merge request.

En bref :

- Vous créez une branche dans votre fork pour modifier et checkout votre code
- Vous la poussez vers votre branche dans votre fork (votre dépôt)
- Vous soumettez une merge request au projet Tiki afin que d'autres puissent relire votre code, poser des questions… #Merge your requests

Sur Gitlab, je vais sur la page de mon projet. Comme vous pouvez le voir, par défaut la première version de Tiki est sélectionnée et elle sera utilisée pour la comparaison. Pour éviter toute confusion, je vous suggère de définir votre branche de travail par défaut sur celle que vous utilisez. Par habitude et par bonne pratique, je sélectionne toujours la branche par défaut vers laquelle j'envoie une merge request.
Allez dans "Settings", "Repository" et ouvrez la section "Default branch".
Sélectionnez la branche que vous voulez et sauvegardez.
C'est fait.

J'ai un repo, j'ai mon fork de Tiki et je vais cloner ma première branche depuis mon fork distant sur mon ordinateur local. Au passage, j'ai déjà fait une https://youtu.be/2vCBhfzSe7Q |vidéo beaucoup plus dense sur Git et Tiki où vous pouvez trouver plus de détails] sur le nettoyage de l'historique (striping history), la sélection de branche, etc.
Vous pouvez la regarder ou faire des recherches sur le web pour aller plus loin.

Comment installer votre Tiki local

Je vais donc créer un clone léger de la branche 21.x avec un historique minimal depuis mon repo en local. Sur mon local, en utilisant mon fork, cela ressemblera à ceci :

```
git clone --depth=1 --branch=21.x [https://Gitlab.com/mycloneurl.git
```

](https://Gitlab.com/mycloneurl.git{CODE})
Je vais dans mon terminal et je la saisis.
Maintenant, dans la vraie vie, je devrais terminer l'installation de Tiki, tester tout ce que je fais et m'assurer que mon environnement local fonctionne bien avant de commiter quoi que ce soit.
Mais ici c'est une démo et je vais sauter ces étapes — vous, vous ne faites pas ça à la maison.
Avant de commiter, vérifiez que tout fonctionne correctement et que cela ne casse rien d'autre dans Tiki.

Comment créer votre branche

Nous allons maintenant créer une nouvelle branche temporaire qui sera utilisée pour commiter un petit ajout simple à la traduction française de Tiki.

Sur mon terminal local, je vérifie la branche existante.

```
git branch -a
```

Je crée généralement une branche par correctif afin de pouvoir soumettre et contrôler mes commits et merge requests. Créons une branche "french-translation".

```
git checkout -b french-translation
```

Je vérifie à nouveau mes branches locales et je peux voir que ma nouvelle branche a été créée et que c'est celle sur laquelle je travaille.

Comment commiter votre code

Une autre bonne pratique consiste à faire un rebase, à vérifier que vous utilisez la dernière version du code avant de commencer à travailler, afin de ne pas avoir de conflits ou de fusions à gérer lors de l'opération de commit.

```
git pull —rebase
```

Je modifie maintenant rapidement le fichier de traduction, je choisis une chaîne à traduire et je

sauvegarde.
Je suis prêt à commiter, mais avant cela je vous rappelle qu'il est important de revérifier votre Tiki d'abord. Tout aussi important, vous devriez consulter régulièrement les informations "How and where to commit" sur dev.tiki.org.

Pour commiter mon fichier la première fois, je dois ajouter ce fichier afin de préparer le contenu (staged) pour mon prochain commit.

```
git add lang/fr/language.php
```

Puis je crée mon commit et j'ajoute le message de commit comme il devrait apparaître.

```
git commit -m "[TRA] Multilingual: French translation"
```

J'ai ajouté ici le tag approprié, la fonctionnalité concernée et la description du commit.

Comment le pousser dans votre fork

Créer un commit est une opération locale et tant que vous ne le poussez pas, cela n'affectera pas le dépôt distant ni le dépôt Tiki. Il est possible de vérifier les commits en attente avec cette commande, et ce que nous voyons ici est conforme à ce qui était attendu.

```
git log --branches --not —remotes
```

Pour pousser ma branche modifiée dans mon fork, je fais :

```
git push -u origin french-translation
```

J'obtiens un message sympa qui montre que cela a bien été fait. Maintenant, je veux que mon commit soit fusionné dans le dépôt Tiki.
Sur Gitlab, je vais sur la page de mon projet, puis dans le menu Gitlab je clique sur "Repositories", "Branches". Là, je peux voir la nouvelle branche que j'ai créée avec le numéro de commit et l'état des changements entre versions. Cette comparaison est faite en utilisant la branche par défaut que nous avons définie.
Les chiffres devraient être très faibles ici ; si vous avez des chiffres plus élevés, cela peut être le signe que vous ne faites pas la bonne chose et vous devriez revérifier.

Chez Tiki, nous utilisons une pipeline CI/CD pour automatiser la revue de code et les tests. Vous devrez peut-être attendre un peu que tout cela se termine avant de pouvoir finaliser votre merge request. Si l'on regarde nos commits, on voit quelques warnings, mais même si vous pouvez vérifier à quoi ils correspondent, des warnings restent des warnings, et selon ce que vous commitez et ce que vous attendez, cela peut être parfaitement normal.

Dans notre cas, tout est bon, on peut y aller.

Comment créer une merge request sur le dépôt Tiki

Je clique donc enfin sur le bouton merge request, je vérifie les informations de branche à branche et éventuellement je les change en cliquant sur le lien "Change branches".

Je colle ou je réécris le même message que j'ai utilisé pour mon message de commit, je donne une description avec le détail de ce qui a été modifié ou corrigé, comment reproduire, etc. Tout ce qui peut aider les autres à relire rapidement la merge request. Voici un exemple de ce que je fais généralement ; pour cette merge request, une traduction, il n'y a pas vraiment grand-chose à dire ou à expliquer.

Comme vous pouvez le voir, j'ai aussi recréé sur mon projet les mêmes tags que nous utilisons dans le repo Tiki : c'est plus simple et plus agréable pour identifier la nature de ma merge request.

J'ai aussi des options pour supprimer la branche source lorsque la merge request est acceptée. Je clique sur "Submit the merge request" et maintenant je suis redirigé dans le dépôt Tiki où je peux voir ma merge request pour ma traduction française multilingue. Je clique sur le bouton "Merge" dans ce cas, parce qu'il n'y a rien d'autre que je souhaite commenter (à propos de ce commit).

Maintenant, comme vous pouvez le voir, nous avons une branche et elle est à jour avec le dernier commit.
Notez aussi que vous avez sur la gauche un menu merge request que vous pouvez toujours utiliser pour commenter et discuter, ou voir des commits en attente plus complexes qui nécessiteraient une approbation, ou si vous voulez partager et obtenir des conseils lorsque vous commitez quelque chose.

Au revoir

Plus d'excuses !
Vous pouvez contribuer sans crainte de casser quoi que ce soit, et vous pouvez rendre votre code disponible pour que d'autres le relisent et éventuellement vous aident.

J'espère que vous avez apprécié ce tutoriel, comme d'habitude. Dites-moi dans les commentaires si vous l'avez aimé. Likez et partagez cette vidéo, bien sûr.

Je n'oublierai pas de publier une vidéo sur le plug-in customSearch, et il y aura d'autres tutoriels vidéo sur les plug-ins listExecute et les trackers. Si vous ne voulez pas rater mon prochain Tiki Express Tutorial, abonnez-vous et faites ding ding sur le petit YouTube.

N'oubliez pas : je réponds, je répondrai aux questions lors de ma réunion hebdomadaire ouverte en chat, donc abonnez-vous sur bsfez.com à la newsletter et vous recevrez mes notifications de meet-up.

C'est tout pour aujourd'hui, et que la puissance de Tiki soit avec vous...

Bêtisier

Eh bien... j’avais dit que ce serait assez rapide. ?

Liens intéressants

https://doc.tiki.org/tiki21
https://dev.tiki.org/tiki21
https://dev.tiki.org/3-Ruleshttps://dev.tiki.org/Git-workflow-For-Collaborators
https://Gitlab.com
https://about.Gitlab.com/handbook/security/
https://docs.Gitlab.com/ee/ssh/
https://Gitlab.com/tikiwiki/tiki.git
https://about.Gitlab.com/handbook/security/
http://dev.tiki.org/Where-to-commit
http://dev.tiki.org/Commit-Tags
https://youtu.be/2vCBhfzSe7Q

Commandes utilisées

```
git clone --depth=1 --branch=21.x https://Gitlab.com/bsfez/yourprojecturl.git .
git branch -a
git checkout -b french-translation
git pull —rebase
git add lang/he/language.php
git commit -m "TRA Multilingual: French translation"
git push -u origin french-translation
git log --branches --not —remotes
```