TP : Gérer son site web avec Git

- Objet : Gérer son site web avec Git
- Niveau requis :
  avisé
- Commentaires : *Vous gérez un ou plusieurs sites web sur votre serveur. Vous voulez l'administrer avec git pour faciliter les mises-à-jour/patch/etc.*
- À savoir : Utilisation de git
- Suivi :
  - Création par captnfab 06/03/2014
  - Testé par <...> le <...> Fix Me!
- Commentaires sur le forum : ici [1)]

Nota :

Contributeurs, les Fix Me! sont là pour vous aider, supprimez-les une fois le problème corrigé ou le champ rempli !

Introduction

Vous gérez vos projets ou vos documents avec git, et vous vous demandez comment faire pour gérer vos sites web de même?

Une des fonctionnalités de Git que nous allons fortement utiliser est celle de «*branches*». Nous allons mettre en place un système de *hooks*[2)], pour faciliter l'intégration de notre flot de travail.

Nous allons considérer le schéma suivant, assez classique :

- Une machine « **serveur** »
  - sur laquelle est installé un serveur **ssh** permettant à l'utilisateur *toto* de se connecter
  - sur laquelle est installé un serveur web publiant les données présentent dans `/srv/monsiteweb/www` (dossier inscriptible par *toto*)
  - ainsi que les données présentent dans `/srv/monsiteweb/www-dev` (dossier inscriptible par *toto*)
- Une machine « **client** »
  - sur laquelle *toto* développe son site web.

Installation

Installer **git** sur le **serveur** et le **client** :

```
apt-get install git
```

Mise en place

Initialisation du dépôt

Création d'un dépôt nu sur le serveur

Sur le **serveur** :

```
mkdir -p /srv/monsiteweb/www.git
```

```
mkdir www.git
```

```
git init --bare www.git
```

Configuration des hooks sur le serveur

Sur le **serveur** :

```
vim /srv/monsiteweb/www.git/hooks/post-update
```

Le fichier doit contenir le code suivant :

/srv/monsiteweb/www.git/hooks/post-update

```
#!/bin/sh
for i in $*
do
  b=$(basename $i)
  if [ "$b" = "prod" ]
  then
    # Remplacer le chemin par celui utilisé par le serveur web pour
    # la version en production
    GIT_WORK_TREE=/srv/monsiteweb/www/ git checkout -f $b
  fi
  if [ "$b" = "dev" ]
  then
    # Remplacer le chemin par celui utilisé par le serveur web pour
    # la version en développement
    GIT_WORK_TREE=/srv/monsiteweb/www-dev/ git checkout -f $b
  fi
done
```

Le rendre exécutable :

```
chmod +x /srv/monsiteweb/www.git/hooks/post-update
```

Si vous utilisez un moteur de blog statique comme pelican ou acrylamid, vous pouvez lancer la recompilation du blog depuis ce fichier, juste après le checkout.

Clonage du dépôt sur le client

Sur le **client** :

```
mkdir -p ~/projets/mon-site/
```

```
cd ~/projets/mon-site/
```

```
git clone toto@serveur:/srv/monsiteweb/www.git
```

Création des branches

Ces manipulations sont à faire sur **client**:

La branche upstream

Cette branche est facultative et contient les sources du site web telles que fournies par son développeur.

```
cd ~/projets/mon-site/www
```

- Rajouter les-dites sources dans le dossier.

Par exemple:

```
wget
'http://download.dokuwiki.org/src/dokuwiki/dokuwiki-oldstable.tg
z' -O/tmp/dokuwiki-oldstable.tgz
tar xf /tmp/dokuwiki-oldstable.tgz --strip-components 1
```

- Indiquer à git de les prendre en compte :

```
git add .
```

```
git commit -m 'Version initiale'
```

- Renommer la branche en *upstream* :

```
git branch -m master upstream
```

- Envoyer la branche sur le dépôt distant :

```
git push -u origin upstream
```

La branche dev

Cette branche est facultative et contient les sources du site web de développement. Un tel site est pratique pour mettre le code en test avant de le passer en production.

- Créer la branche *dev* basée sur la branche *upstream* :

```
git checkout upstream
```

```
git branch dev
```

```
git checkout dev
```

- Appliquer les modifications voulues.

Par exemple :

```
rm README VERSION COPYING
```

- Valider les changements et envoyer le résultat sur le dépôt distant :

```
git commit -a -m 'Préparation à la mise en test'
```

```
git push -u origin dev
```

Le hook de la branche dev devrait alors s'activer automatiquement et placer les sources dans le dossier de dev sur le serveur.

La branche prod

Cette branche correspond à la version en cours d'utilisation sur le serveur.

- Créer la branche *prod* basée sur la branche *dev* :

```
git checkout dev
```

```
git branch prod
```

```
git checkout prod
```

- La branche de test étant satisfaisante (sinon on ne la passerait pas en production), on se contente de la publier.

```
git push -u origin prod
```

Le hook de la branche prod devrait alors s'activer automatiquement et placer les sources dans le dossier de prod sur le serveur.

Opérations courantes

Comment intégrer les opérations de maintenance courantes avec la structure git définie.

Ajout d'une fonctionnalité

Ou correction d'une fonctionnalité indésirable...

- Récupérer des versions à jour du dépôt :

```
git checkout dev
```

```
git pull
```

- Éditer la branche dev
- Valider vos modifications :

```
git commit -a -m 'bug corrigé'
```

- Soumettre vos modifications

```
git push
```

- Lorsque le résultat est convenable, mettre à niveau la branche master

```
git checkout master
```

```
git merge dev -m 'intégration de la correction de bug'
```

- Serrer les fesses et soumettre le résultat :

```
git push
```

Mise à jour depuis upstream

- Récupérer les nouvelles sources dans la branche *upstream*.

```
git checkout upstream
```

Par exemple :

```
git checkout upstream
```

```
rm -r *
```

```
wget 'http://download.dokuwiki.org/src/dokuwiki/dokuwiki-stable.tgz' -O/tmp/dokuwiki-stable.tgz
```

```
tar xf /tmp/dokuwiki-stable.tgz --strip-components 1
```

```
git add --all .
```

```
git commit -a -m 'nouvelle version upstream: Binky (2013-12-08)'
```

- Soumettre le résultat :

```
git push
```

- Importer les changements dans la branche de dev :

```
git checkout dev
```

```
git rebase upstream
```

Corriger les erreurs éventuelles de fusion. Par exemple :

```
git rm VERSION
```

```
git rebase --continue
```

- Puis soumettre la nouvelle version :

```
git push --force
```

- Fusionner la branche dev opérationnelle depuis la branche prod :

```
git checkout prod
```

```
git merge dev -m 'intégration de la nouvelle version upstream'
```

- Serrer les fesses et soumettre le résultat :

```
git push
```

Références

- http://www.git-scm.com/book/fr
- http://toroid.org/ams/git-website-howto (en)
- https://www.kernel.org/pub/software/scm/git/docs/githooks.html (en)

[1]

N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !

[2]

crochets: scripts exécutés automatiquement lors de certaines opérations effectuées sur un dépôt git