

Déploiement d'une application Rails avec Capistrano

par [Frédéric Bollon](#)

Date de publication : 13 Décembre 2006

Dernière mise à jour : 13 Décembre 2006

Capistrano est un outils de déploiement d'application web qui permet d'automatiser la mise en production d'une nouvelle version . Nous allons voir ici un exemple d'utilisation de cet outil.

- I - Introduction et pré-requis
- II - Versionnement du projet
- III - Capistrano
- IV - Déploiement de l'application
- V - Conclusion

I - Introduction et pré-requis

L'exemple ci-dessous est adapté à [mon hébergement chez Dreamhost](#), à vous de le modifier selon vos besoins.

[Capistrano](#) est un outils de déploiement d'application web qui permet d'automatiser la mise en production d'une nouvelle version.

Les possibilités sont énormes, voir [le site officiel](#) pour tous les détails, dans cet article je vous vais vous décrire un exemple d'utilisation de Capistrano pour déployer chez votre hébergeur votre application Rails développée en local sur le serveur WEBrick (serveur de développement inclus dans rails).

Il y a un pré-requis, vous devez versionner votre projet avec [Subversion](#) car le déploiement va se faire du serveur [Subversion](#) vers le serveur web, l'hébergeur Dreamhost inclu Subversion dans son offre de base mais vous pouvez très bien héberger vous même le serveur Subversion sur votre propre machine.

Dans un premier temps, vous devez versionner correctement votre projet car se sont les fichiers qui se trouvent dans votre "subversion repository" qui seront envoyés sur votre serveur web.

II - Versionnement du projet

Import initial du projet sur le serveur Subversion :

```
svn import monprojet/ http://svn.mondomaine.com/monprojet -m
  "Import initial"
```

Checkout du projet pour créer votre copie de travail en local :

```
svn checkout http://svn.mondomaine.com/monprojet monprojet
```

On se place à la racine de l'application :

```
cd monprojet
```

On supprime les fichiers de log dans Subversion pour ne pas les déployer sur le serveur de production et on les ignore :

```
svn remove log/*
svn commit -m 'suppression des fichiers de log de subversion'
svn propset svn:ignore "*.log" log/
svn update log/
svn commit -m 'on ignore les fichiers du répertoire /log/ qui se terminent par .log'
```

On ignore le contenu du répertoire /tmp/ :

```
svn remove tmp/*
svn propset svn:ignore "*" tmp/
svn update tmp/
svn commit -m "ignore contenu de tmp/ "
```

Pour les fichiers qui doivent être différents en local et sur le serveur, par exemple le fichier de configuration de la librairie `auth_generator`, je versionne une version de production et une version de développement (`auth_generator.yml.pr` et `auth_generator.yml.dev`) nous verrons pourquoi dans la configuration de Capistrano.

Je fais également la même chose avec le `.htaccess` du répertoire `/public` (la ligne `-> "RewriteRule ^(.*)$ dispatch.fcgi [QSA,L]"` sur le serveur diffère de la version de développement `"RewriteRule ^(.*)$ dispatch.cgi [QSA,L]"`) :

```
svn remove auth_generator.yml
svn propset svn:ignore auth_generator.yml
svn commit -m "ignore auth_generator.yml"
svn remove .htaccess
svn propset svn:ignore .htaccess
svn commit -m "ignore .htaccess"
```

III - Capistrano

Installation de Capistrano :

```
gem install capistrano
```

Attention ! Sous linux, il faut être root pour pouvoir lancer cette commande.

Création des fichiers Capistrano :

```
cap -A
```

Modification du fichier config/deploy.rb ([voir mon exemple](#)).

Création de la structure de répertoire sur votre hébergement :

```
rake remote:exec ACTION=setup
```

Nous allons également modifier les lignes suivantes dans le fichier config/deploy.rb :

```
task :restart, :roles => :app do
  run "ruby #{current_path}/script/process/reaper -dispatcher=dispatch.fcgi"
end
```

En :

```
task :restart, :roles => :app do
  run "ruby #{current_path}/script/process/reaper -a graceful -dispatcher=dispatch.fcgi"
end
```

Encore une petite précision, de temps en temps j'exécute la commande "rake remote:cleanup" ce qui a pour effet de supprimer toutes les anciennes versions des déploiements précédents en ne conservant que les 5 plus récentes.

IV - Déploiement de l'application

```
rake deploy
```

C'est tout, si il n'y a pas d'erreur dans le fichier `deploy.rb` la nouvelle version de votre application est en ligne, ça peut paraître un peu fastidieux en début de projet mais tellement simple par la suite que l'on ne peut plus s'en passer.

Si vous faites une erreur lors d'une nouvelle version il existe la commande "*rake rollback*" qui vous permettra de revenir à la version précédente.

V - Conclusion

Cet article est un exemple rapide d'utilisation, je vous conseille de vous documenter d'avantage sur le [site officiel](#) pour bien comprendre le fonctionnement.

Si vous avez des questions, n'hésitez pas à aller les poser directement à l'auteur sur [son blog](#).