CAPISTRANO

Capistrano est un outil open source pour exécuter des scripts sur plusieurs serveurs. Son utilisation principale est le déploiement d'applications Web. Il automatise le processus de création d'une nouvelle version d'une application disponible sur un ou plusieurs serveurs Web, y compris le soutien des tâches telles que la modification des bases de données.

Bien que Capistrano soit d'origine conçu pour déployer des applications Rails, il convient parfaitement pour tout type d'applications, que ce soit du PHP, du nodeJS ou bien d'autres.

En quelques mots, Capistrano vous permettra d'automatiser vos déploiements de façon sécurisée en faisant un snapshot de la version actuelle avant d'éxécuter les tâches de déploiement. De cette manière, vous pourrez facilement revenir en arrière en cas de problème avec votre nouvelle version.

1. Installation

- Pour obtenir Capistrano, il vous faut Ruby, à installer de préfèrence avec rvm :

```
curl -sSL https://get.rvm.io | bash -s stable --ruby
```

Ceci mettra en place l'installeur `rvm`, la dernière version stable de Ruby ainsi que son populaire gestionnaire de paquets RubyGems. Plus d'informations sur le site officiel.

- Installons maintenant Capistrano avec RubyGems :

```
gem install capistrano
```

Ceci fait, vous devriez avoir la dernière version de Capistrano (v3.11 actuellement), vous pouvez le vérifier avec cette commande :

```
cap --version
```

2. Initialisation

Maintenant que Capistrano est installé sur votre machine, il vous faut l'initialiser dans votre projet.

Nous prendrons comme base un petit projet PHP utilisant MySQL, le tout sur un dépot GIT. Vous pouvez retrouver le code source de ce projet ici : https://github.com/nexylan/capistrano-php-sandbox.

Admettons que notre projet est composé pour le moment de 3 fichiers :

- `config/config-sample.inc.php` ? Fichier modèle de configuration
- `config/config.php` ? A créer à partir du fichier modèle (Ignoré du dépot git)
- `web/index.php` ? Fichier principal qui inclut le fichier de configuration avant l'éxecution

Pour initialiser Capistrano, dans un terminal, allez dans le dossier de votre projet puis tapez la commande suivante :

```
cap install
```

Plusieurs dossiers et fichiers sont créés :

- **Capfile** ? Fichier de configuration de Capistrano. Principalement utilisé pour inclure des modules supplémentaires.
- **config/deploy.rb** ? Votre fichier de configuration principal de déploiement
- **config/deploy/[production|staging].rb** ? Fichiers secondaires de configuration, appelés selon le type de déploiement choisi.
- **lib/capistrano/tasks** ? Vous pouvez ici définir des tâches supplémentaires qui seront utilisées pour le déploiement.

Pour faire simple, supprimons le fichier `staging.rb` et concentrons nous sur `deploy.rb` et `production.rb` que nous allons épurer.

Voici une configuration basique du fichier deploy.rb et ses explications :

```
set :application, 'capistrano-php-sandbox' # Le nom de votre projet à titre informatifset :repo_url, 'git@github.com:Nexylan/capistrano-php-sandbox.git' # L'adresse (avec protocole) où retrouver le code source de votre projetset :deploy_to, '/var/www/php-sandbox.cap/html' # Le dossier distant où déployer votre projet
set :scm, :git # Le gestionnaire de version à utiliser, ici GIT
set :log_level, Logger::INFO # Niveau d'information
set :linked_files, %w{config/config.inc.php} # Les fichiers partagés# set :linked_dirs, %w{bin log tmp/pids tmp/cache tmp/sockets vendor/bundle public/system} # Les dossiers partagés
set :keep_releases, 5 # Le nombre maximum de releases à sauvegarder pour un eventuel retour arrière```
Les fichiers et dossiers partagés sont les éléments propre au serveur de votre site et qui doivent être conservés lors du déploiement d'une nouvelle version. Ils seront placés dans le dossier shared, c'est le cas de notre fichier config/config.php
Pour le fichier production.rb, nous allons simplement indiquer le serveur à utiliser ainsi que les options de connexion:
```

```
server 'php-sandbox.cap', user: 'cap', roles: %w{web app db} # Nom du serveur, l'utilisateur de connection et les roles associés

set :ssh_options, { # Ici les options sont présentes à titre d'exemple, le port 22 étant le port par défault.
    port: 22
}
```

Vous pouvez bien sûr définir plusieurs serveurs avec des rôles particuliers ainsi que surcharger les options du fichier parent `deploy.rb`.

Vous pouvez voir et reprendre nos fichiers sur notre dépot Github.

---

3. Préparation du serveur de production

Avant de déployer, nous allons préparer le serveur de production.

Pour que Capistrano puisse fonctionner correctement, nous devons au préalable créer le dossier `share` et y placer tous les fichiers et dossiers partagés. En l'occurence pour nous, le fichier `config/config.inc.php`.

```
cd /var/www/php-sandbox.cap/html
mkdir -p shared
mkdir -p shared/config
touch shared/config/config.inc.php
# ? Cela va créer un fichier vide, nous pourrons le remplir plus tard.
```

Gardez en tête de toujours créer les fichiers et dossiers partagés supplémentaires sur votre serveur avant de déployer !

N'oubliez pas de prévenir l'équipe de Nexylan de votre utilisation à venir de Capistrano afin que nous puissions changer le

chemin de destination du serveur web vers le lien "current", qui sera le chemin par défaut de votre site web.

4. Déploiement de votre projet

Le grand moment que vous attendez tous, le déploiement ! :)

Pour cela, rien de plus simple, tapez la commande suivante sur votre terminal :

```
cap production deploy
```

Le mot production correspondant au fichier de configuration défini.

Après le déploiement, Capistrano vous à créé les dossiers et fichiers suivants :

```
cap@php-sandbox.cap:/var/www/php-sandbox.cap/html$ ls -l
total 16lrwxrwxrwx 1 cap cap   53 May  6 16:36 current -> /var/www/php-sandbox.cap/html/releases/20140506143556 # La version actuellement utilisée
drwxrwxr-x 3 cap cap 4096 May  6 16:36 releases # Les différentes versionsdrwxrwxr-x 7 cap cap 4096 May  6 16:35 repo # Le dossier contenant le code source servant pour les mises à jour-rw-rw-r-- 1 cap cap   74 May  6 16:36 revisions.log # Un fichier listant les différents déploiements effectués avec dates et auteurs
drwxrwxr-x 3 cap cap 4096 May  6 16:35 shared # Le dossier contenant les fichiers partagés
```

Un soucis lors du déploiement ? Aucun problème ! Utilisez cette commande :

```
cap production deploy:rollback
```

Et vous revenez à la version précédente.

5. Cas particuliers

Ici seront énoncés les différents cas d'utilisation et astuces pour Capistrano.

Les migrations MySQL

Notre exemple de déploiement ne permet pas de gérer les changements au niveau des bases de données.

Pour le gérer, il faut utiliser les migrations. Nous prendrons MySQL comme exemple.

Le concept des migrations SQL est de définir une liste de requêtes dans des fichiers permettant la mise en place de la base de données. Lors de changements, de nouveaux fichiers de migration s'ajouteront aux dossiers. C'est fichiers sont ensuite exécutés par Capistrano selon la tâche que vous aurez défini.

Les frameworks les plus récents proposent des outils de migrations qui leurs sont propres, pouvant être utilisés par Capistrano:

- **Symfony** : DoctrineMigrationsBundle
- **Rails** : Active Record Migrations

La liste n'est pas exhaustive, si vous en connaissez d'autres, n'hésitez pas à nous le préciser ! ;)

Pour les projets ne possédant pas de tel système, il va falloir en créer un. Vous trouverez ici une piste pour vous aider à le mettre en place. Nous vous conseillons fortement de créer un système "stand-alone", hermétique de votre projet actuel afin de pouvoir le réutiliser facilement dans les projets futurs sans devoir tout réécrire.

Les extensions Capistrano

De nombreuses extensions Capistrano existent, permettant de faciliter la configuration en évitant de réécrire des tâches déjà existantes pour vos outils. Vous pouvez, en nous contactant, proposer des extensions supplémentaires pour les ajouter à cette liste.

Gestionnaires de paquets

- **capistrano/bundler** : Support pour le gestionnaire de dépendances bundler pour Ruby.
- **capistrano/composer** : Support pour le gestionnaire de dépendances composer pour PHP.
- **capistrano/bower** : Support pour le gestionnaire de dépendances bower pour les librairies JS et CSS.
- **capistrano/npm** : Support pour le gestionnaire de dépendances npm pour NodeJS. Frameworks
- **capistrano/symfony** : Support pour le framework Symfony2.
- **capistrano/rails** : Support pour le framework Rails.
- **capistrano/laravel** : Support pour le framwork Laravel.

Outils et divers

- **capistrano/maintenance** : Permet de gérer facilement les mises en maintenance de vos sites avant le déploiement.
- **capistrano/file-permissions** : Permet de changer les droits de certains fichiers et dossiers nécessaires au déploiement.

Revision #3
Created Fri, Nov 29, 2019 10:03 PM by Admin
Updated Tue, Jan 14, 2020 4:48 PM by cmiran