Fabric, un outil pour déployer ses projets

Thomas Gratier

Des manières de déployer

A la mano

- Répétitif
- Risqué
- Source d'erreurs
- Non traçable

Approche fichiers

- FTP / SCP / Rsync
- Gestionnaires de version + hooks

Approche services

- Services online de déploiement
- PaaS
- Intégration continue

Approche "deploy as code"

- Déploiement SSH via ligne de commande
- Logiciels de déploiement automatique, centralisés ou non.

Et Fabric dans tout ça?

De l'usage

- Simple développeur avec quelques connaissances Sysadmin
- Besoin de versionner
- Besoin d'automatiser
- Temps limité pour monter en compétences
- Basé sur Python et connaissance de Python

Un choix par opposition

- Ne pas dépendre de services tiers
- Ne pas apprendre de DSL
- Bash puissant mais pas "cross platform", de plus bas niveau que Python

Enfin un peu de technique

Installation

- Installer Python 2.x, setuptools
- Dépendances Paramiko et PyCrypto (pour SSH)
- Lancer

```
pip install fabric
```

Jouer

- Créer un fichier fabfile.py
- Insérer le contenu ci-dessous

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from fabric.api import *

# Utilisateur pour la connexion distante

env.user = 'utilisateurapplication'

# Urls des serveurs de déploiement

env.hosts = ['server1.exemple.com', 'server2.exemple.

def deploy():
    ...
```

- Exécuter avec

```
fab server1.exemple.com
```

Un aperçu plus complet

Les commandes de base

- Exécuter des commandes locales (**local**)
- Exécuter des commandes distantes (**run** et **sudo**)
- Envoyer ou récupérer des fichiers distants (**get**, **put**)
- Gérer des environnements, comme la gestion des chemins système (**context_managers**)

Les contributions

- Manipuler des fichiers (**append**, **exists**, **contains**, **sed**, **comment**)
- Gérer Rsync, envoyer un projet, utiliser des templates

Passer au niveau supérieur

Fabtools, le facilitateur

Un outil fournissant des helpers de plus haut niveau pour Fabric.

Installer avec

```
pip install fabtools
```

Dans le fabfile.py

```
from fabric.api import *
from fabtools import require
import fabtools

deploy():
    require.deb.packages([
        'imagemagick',
        'libxml2-dev',
    ])
```

Gérer les logiciels de l'OS et le versionning

- Installation de paquets (**require.deb**, **require.rpm**)
- Installation de serveur et configuration (**require.apache**, **require.nginx**, **require.tomcat**)
- Bases de données (**fabtools.mysql**, **fabtools.postgres**)
- Versionning (**fabtools.git**, **fabtools.mercurial**)

Gérer des éléments système

- Services et processus (**fabtools.service**, **fabtools.systemd**, **fabtools.cron**)
- Groupes et utilisateurs (**fabtools.group**, **fabtools.user**)
- Informations réseaux (**fabtools.network**)
- Vagrant (**fabtools.vagrant**)

Allez plus loin

Sites officiels

- Site de Fabric http://www.fabfile.org
- Site de Fabtools http://fabtools.readthedocs.org

Bonus

- Article de Chris Coyier sur un éventail des solutions pour le déploiement http://css-tricks.com/deployment/
- Présentation de Ronan Amicel, le développeur principal de Fabtools
  http://fr.slideshare.net/ronan.amicel/je-configure-mes-serveurs-avec-fabric-et-fabtools