Fabricでお手軽サーバ管理

ssmjp 2014年10月

自己紹介

こばやし けんいち (@Niratama)

ソーシャルゲーム屋
Perlが主戦場……のはず
最近はインフラとかそっちの作業が多くて、
がっつりプログラム書いてない感じ

Fabricとは

- http://www.fabfile.org
- シンプルなデプロイ・システム管理ツール
  - とりあえず始めるのに覚えることが少ない
- Python製で設定ファイルもPythonのスクリプト
  - いざとなればPythonで書けることはなんでもできる
- SSH経由で他のサーバに乗り込んで作業する
  - 乗り込んだ先のマシンにいろいろインストールする必要はない
- MySQL Fabricとは関係ないです

位置づけ

- ChefやPuppetでがっつり管理するのと、シェルスクリプトで簡単にやっつけるのの中間あたり
- Pythonのスクリプトなので、プログラマの立場から見ると扱いやすい
- 今がシェルスクリプト管理なら移行しやすいかも

設定ファイル

- `fabfile.py`という名前でファイル(= Pythonのスクリプトファイル)を用意する
- タスク(= Pythonの関数)に処理を書く
- 複数ファイルに分割してモジュール化できるので、規模がある程度大きくなっても安心

```
from fabric.api import *

@task
def hostname():
    run('hostname')

@task
def upload_module():
  put('~/works/webapp/lib', '/home/user/webapp/')

@task
def download_logs():
  get('/home/user/webapp/log/access.log')

@task
def reboot_httpd():
  sudo('/sbin/service httpd restart')
```

Core API

- リモートのコマンドを実行、リモートとのファイル転送
- コンテキストマネージャを使った特定範囲での設定変更
- デコレータを使ったタスクの属性変更
- ログやメッセージの表示、カラー表示
- その他ユーティリティ

Contrib API

- リモートのファイル操作(テキストの追加・置換、ファイルの存在チェック)
- プロジェクトまるごとのアップロード
- Djangoとの連携

Fabric + Cuisine

- https://github.com/sebastien/cuisine
- 冪等性が欲しいときに使う追加API
- 残念ながらドキュメントがあまり整備されてないので、使うには`cuisine.py`を読む必要がある

```
from fabric.api import *
from fabric.contrib.files import *
import cuisine
from pit import Pit

cuisine.select_package('yum')

@task
def create_user():
    with settings(user='root'):
        cuisine.user_ensure('ssmjp')
        append('/etc/sudoers', 'ssmjp ALL=(ALL) ALL')
        cuisine.ssh_authorize('ssmjp', cuisine.file_local_read('~/.ssh/ssmjp.pub'))
        conf = Pit.get('ssmjp-user', { 'require': { 'password': 'Your password' } })
        cuisine.user_passwd('ssmjp', conf['password'])

@task
def install_packages():
    with settings(user='root'):
        cuisine.package_ensure('nginx')
```

Tips

SSHのconfigファイルを参照する

- 通常はSSHのconfigファイルは参照しないが、以下の設定をしておくと参照してくれる
- IdentityfileやProxyCommandも見てくれるので便利

```
env.use_ssh_config = True
```

リモートにテンポラリディレクトリ

- リモートで`mktemp -d`を実行して、`try`〜`finally`で最後に後始末する
- `run()`の実行結果に作ったディレクトリ名が戻る

```
def diff_hosts():
  tempdir = run('mktemp -d')
  try:
    tempfile = '%s/hosts' % tempdir
    put('/etc/hosts', tempfile)
    run('diff -u /etc/hosts %s' % tempfile, warn_only=True)
  finally:
    run('rm -r %s' % tempdir)
```

モジュール化

- `fabfile`というディレクトリに`__init__.py`というファイルを作って、その中でタスクを書いたファイルをimportするとモジュール化できる
- タスクの呼び出しは`fab module.task`みたいな形で呼び出せる

ファイル構成

```
.
└── fabfile/
    ├── __init__.py
    ├── foo.py
    └── math.py
```

__init__.py

```
import foo
import math
```

ドキュメンテーション文字列

- タスクにドキュメンテーション文字列を付けておくと、`fab -l`を実行したときにタスクの説明として表示される

```
@task
def calc_add(x, y):
    '''
    add two integer
    '''
    print '%d + %d = %d' % (int(x), int(y), int(x) + int(y))
```

ホスト名の管理

- コマンドラインオプションで渡したホスト名は env.hosts でアクセスできるので、 fabfile.py に埋め込むことも可能

```
env.hosts = ['server1', 'server2']
```

- env.roledefを設定しておくと、ホスト群をRoleでまとめて扱える

```
env.roledefs = {
   'web': ['web01','web02'],
   'db': ['db01']
}
```

- Pythonスクリプトなので、外部からホスト一覧を読み込むことも可能
  - PyYAMLが簡単

```
import yaml

def load_servers(filename):
    config_yaml = open(filename).read()
    config = yaml.load(config_yaml)
    roledefs = config['roles']
    all_hosts = []
    for role in roledefs:
        all_hosts.extend(roledefs[role])
    roledefs['all'] = all_hosts
    return roledefs

env.roledefs.update(load_servers('./servers.yaml'))
```

servers.yaml

```
roles:
    even:
        - conoha01.poi.jp
        - conoha03.poi.jp
    odd:
        - conoha02.poi.jp
```

- ホストリストの書き換えとかも可能
  - `host[00-03]` → `host00`,`host01`,`host02`,`host03`

```
import re

def expand_hosts(hosts):
    new_hosts = []
    for host in hosts:
        m = re.search('\[(\d+)-(\d+)\]', host)
        if m:
            pre = host[:m.start()]
            post = host[m.end():]
            prec = len(m.group(1))
            for n in range(int(m.group(1)), int(m.group(2)) + 1):
                new_hosts.append(pre + ('%%0%dd' % prec % n) + post)
        else:
            new_hosts.append(host)
    return new_hosts

env.hosts = expand_hosts(env.hosts)
```