Python & DevOps

jib.li

Chakib Benziane @sp4ke

- CTO & CoFondateur sur Jib.li
- Partisan de du logiciel libre depuis 10 ans
- Developpeur Python depuis prés de 6 ans
- Ex étudiant Epitech
- Freelance

Au sommaire

1. Environnement & Stack

2. Déploiement agile avec uWSGI

Nos besoins:

- Application Web
- Intégration aux réseaux sociaux
- Développement Agile
- Communauté et Packages disponibles

Nos besoins:

- Application Web
- Intégration aux réseaux sociaux
- Développement Agile
- Communauté et Packages disponibles

= Django + MongoDB + Github

Nos besoins:

- Application Web
- Intégration aux réseaux sociaux
- Développement Agile
- Communauté et Packages disponibles

= Django + MongoDB + Github

gevent-socketio, zmq, celery, AWS boto ...

Environnement local : les incontournables

- Virtualenv
- PIP

Environnement local : les incontournables

- Virtualenv
- PIP
- Mais les autres ont RVM !

Environnement local : les incontournables

- Virtualenv
- PIP
- Mais les autres ont RVM !
- Pythonbrew: utahta/pythonbrew.git

- Pythonbrew: utahta/pythonbrew.git
  - Compiler des pythons indépendants du systéme

```
$ pythonbrew install 2.7.3
$ pythonbrew use 2.7.3
$ pythonbrew list && which python
# pythonbrew pythons
  Python-2.7.3 (*)
/home/spike/.pythonbrew/pythons/Python-2.7.3/bin/python
```

- Pythonbrew: utahta/pythonbrew.git
  - Créer et activer facilement les virtualenv

```
$ pythonbrew venv create jibli
$ pythonbrew venv use jibli && which python && which pip

# Using `jibli` environment
# To leave an environment, simply run `deactivate`
/home/spike/.pythonbrew/venvs/Python-2.7.3/jibli/bin/python
/home/spike/.pythonbrew/venvs/Python-2.7.3/jibli/bin/pip
```

Setup de l'environnement:

```
git clone jibli/project && cd project
pythonbrew create venv jibli && pythonbrew activate jibli
pip install -r dependencies.txt
```

Setup de l'environnement:

```
git clone jibli/project && cd project
pythonbrew create venv jibli && pythonbrew activate jibli
pip install -r dependencies.txt --download-cache=CACHE
```

pip et dependencies.txt:

pip et dependencies.txt:

```
package-foo
package-bar==4.2

git+https://github.com/user/repo
```

pip et dependencies.txt:

```
package-foo
package-bar==4.2

git+https://github.com/user/repo
git+https://github.com/user/repo#egg=mon-package
```

pip et dependencies.txt:

```
package-foo
package-bar==4.2

git+https://github.com/user/repo
git+https://github.com/user/repo#egg=mon-package
git+https://github.com/user/repo@branch
```

pip et dependencies.txt:

```
package-foo
package-bar==4.2

git+https://github.com/user/repo
git+https://github.com/user/repo#egg=mon-package
git+https://github.com/user/repo@branch
```

- Mettre à jour les dépendences

```
pip freeze > dependencies.txt
```

MongoDB

- NoSQL, Schemaless, Orientée documents
- Données sous format BSON (JSON Binaire)
- Avantage: Python Dict -> JSON
- Excellente API Python pymongo

MongoDB

- Console MongoDB Javascript

```
$ mongo jibli
MongoDB shell version: 2.0.6
connecting to: jibli
> db.users.find( {'profil.age': 10} );
```

MongoDB

- Console MongoDB Javascript

```
$ mongo jibli
MongoDB shell version: 2.0.6
connecting to: jibli
> db.users.find( {'profil.age': 10} );
```

- Equivalent pymongo

```
u = pymongo.Connection(host='localhost', port=27017)['jibli']['users']
u.find( {'profil.age': 10} )
```

Développement en local

- Git branch feature
- Unit test
- Implémentation
- Test sur serveur local (./manage.py runserver)
- Commit et merge dans la branche master

Déploiement agile

- Beaucoup de features nécessitent un environnement similaire à la prod:
  - Authentification OAuth et Résaux sociaux
  - Taches asynchrones Celery (notifications, crons ...)
  - Notifications push
  - Difficile de reproduire tous l'écosystème d'une application web en local

- Scénario souhaité
  - Je commence une nouvelle feature

```
$ git checkout -b feature
```

- Scénario souhaité
  - Je commence une nouvelle feature

```
$ git checkout -b feature
```

  - J'implémente puis push en dev

```
$ fab push
```

- Scénario souhaité
  - Je commence une nouvelle feature

```
$ git checkout -b feature
```

  - J'implémente puis push en dev

```
$ fab push
```

  - Ma branche est UP sur feature.dev.com

- Scénario souhaité
  - Je commence une nouvelle feature

    ```
    $ git checkout -b feature
    ```
  - J'implémente puis push en dev

    ```
    $ fab push
    ```
  - Ma branche est UP sur feature.dev.com
  - Commander l'appli depuis mon shell local mais avec un environnement de prod (ie. restart, upgrade, ipython, mongo shell ...)

- Scénario souhaité
  - Je commence une nouvelle feature

    ```
    $ git checkout -b feature
    ```
  - J'implémente puis push en dev

    ```
    $ fab push
    ```
  - Ma branche est UP sur feature.dev.com
  - Commander l'appli depuis mon shell local mais avec un environnement de prod (ie. restart, upgrade, ipython, mongo shell ...)
  - Une fois satisfait je merge en master et déploie en prod

Solution

- Nginx
- Github
- Fabric

et

- uWSGI

uWSGI

- Permet de construire une stack de developpement
- Hébergement de clusters d'applications

uWSGI

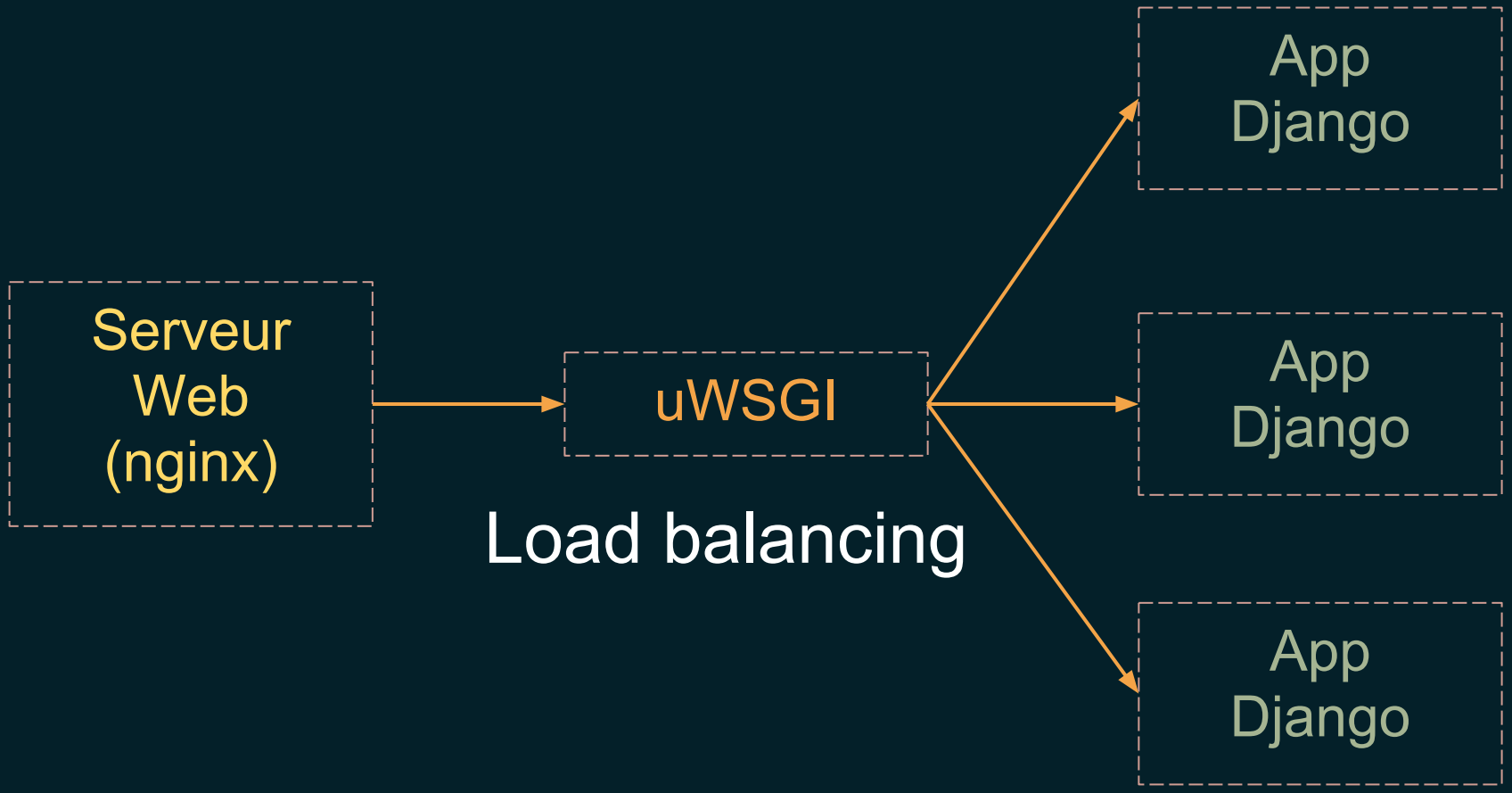
- Comment ?

Interface WSGI

mais aussi: FastCGI,CGI, PHP, Rack, ...

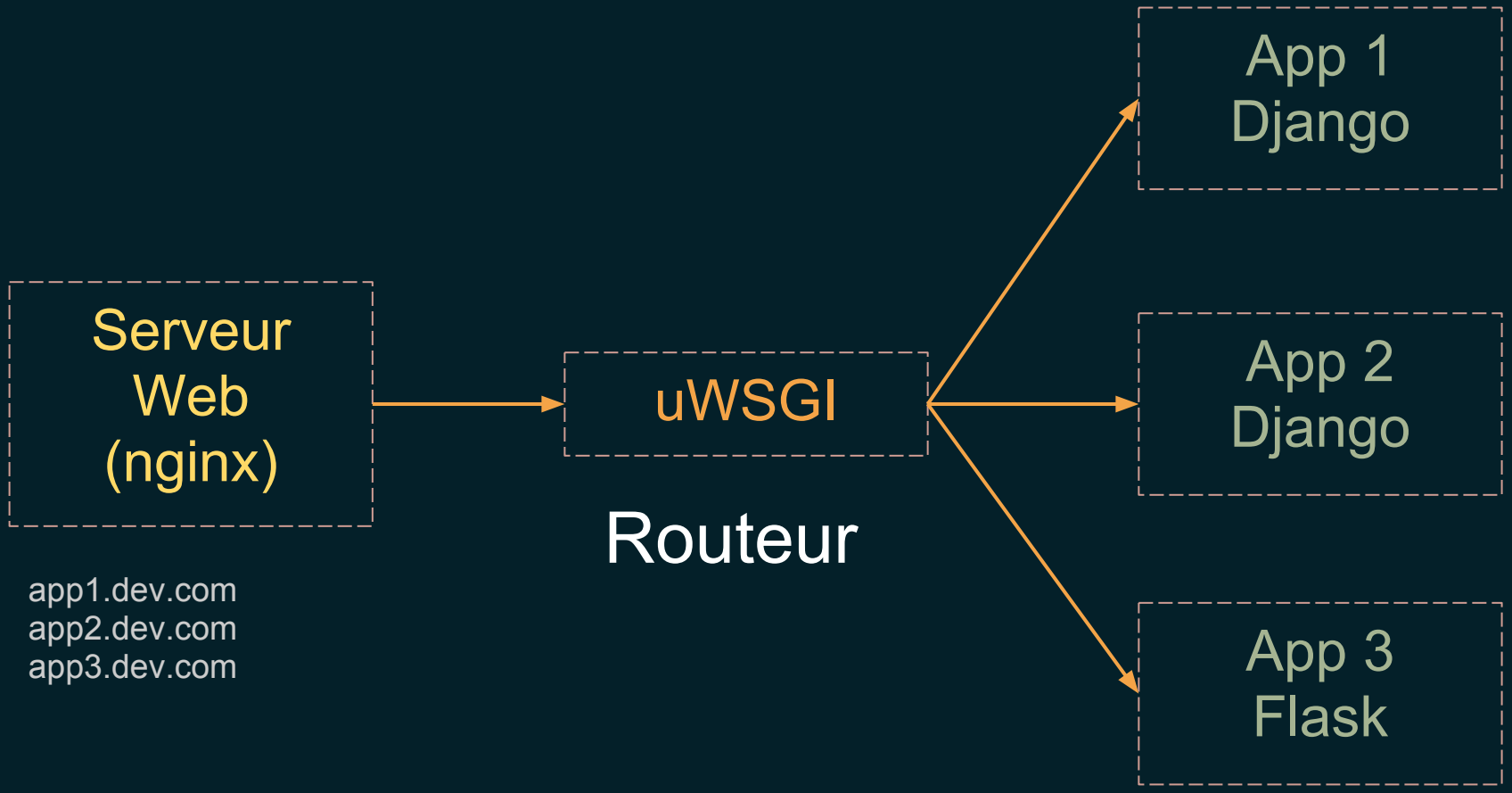
uWSGI

- Comment ?



uWSGI

- Comment ?



- Comment ?

uWSGI
App 1
Django
App 2
Django
App 3
Flask
App n
Serveur
Web
(nginx)
uWSGI
Emperor
déploiement massif
multi app
app1.dev.com
app2.dev.com
app3.dev.com
*.dev.com

uWSGI Emperor

- Manipuler des applications (Vassals) dynamiquement en écoutant certains évenements système

Par default:

- Scan des répertoires à la recherche de fichiers de configuration (.ini, .xml, .yml, .json ...)
- Plugins dir:// et glob:// pour monitorer les fichiers de conf
- D'autres plugins sont disponibles (mongodb, ampq, ldap ...)

Plugin glob://

```
uwsgi --emperor "/opt/apps/*/uwsgi.ini"
```

Plugin glob://

```
uwsgi --emperor "/opt/apps/*/uwsgi.ini"
```

Faire attention à l'interpréteur Bash !

Plugin glob://

```
uwsgi --emperor "/opt/apps/*/uwsgi.ini"
```

Exemple:

- Nouveau fichier "/opt/apps/appn/uwsgi.ini"
  - Spawn d'un vassal
- Fichier modfié
  - Redémarrer le vassal
- Fichier supprimé
  - Tuer le vassal
- Si l'emperor meurt
  - Tous les vassals meurent avec

- Créer un fichier de conf pour chaque app déployée ?

/opt/apps/app1/uwsgi.ini
/opt/apps/app2/uwsgi.ini
/opt/apps/appn/uwsgi.ini

- Créer un fichier de conf pour chaque app déployée ?

/opt/apps/app1/uwsgi.ini
/opt/apps/app2/uwsgi.ini
/opt/apps/appn/uwsgi.ini

- ln -s
  - Fichier de conf template

/opt/apps/template

ln -s /opt/apps/template /opt/apps/app1/app1.ini

Fichier de conf template (App Django)

```
[uwsgi]
djangoproject = %d/app/
home = %d/virt
pythonpath = %d/
env = DJANGO_SETTINGS_MODULE=app.settings
chdir = %(djangoproject)
module = uwsgi_app
socket = /tmp/sockets/%n.socket
master = true
processes = 1
idle = 300
subscribe-to = 127.0.0.1:9999:%n.dev.com
logto = %d/log/uwsgi.log
```

Fichier de conf template

```
[uwsgi]
djangoproject = %d/app/
```

- Déclaration d'une variable djangoproject
- Utilisation des variables magiques :
  - %d - Chemin absolu vers le répertoire contenant le fichier de configuration
  - %n - Nom du fichier de conf sans l'extension

Fichier de conf template

```
[uwsgi]
djangoproject = %d/app/
home = %d/virt
pythonpath = %d/
env = DJANGO_SETTINGS_MODULE=app.settings
```

- On peut définir un environnement virtuel

Fichier de conf template

```
[uwsgi]
djangoproject = %d/app/
home = %d/virt
pythonpath = %d/
env = DJANGO_SETTINGS_MODULE=app.settings
```

- On peut définir un environnement virtuel
- Les chemins de recherches python

Fichier de conf template

```
[uwsgi]
djangoproject = %d/app/
home = %d/virt
pythonpath = %d/
env = DJANGO_SETTINGS_MODULE=app.settings
```

- On peut définir un environnement virtuel
- Les chemins de recherches python
- Ajouter des variables d'environnement

Fichier de conf template (App Django)

```
[uwsgi]
chdir = %(djangoproject)
module = uwsgi_app
```

- Le module qui sera executé pour lancer l'application wsgi
  - django.core.handlers.wsgi:WSGIHandler()

- L'endroit idéal pour executer des scripts et préparer l'environnement avant le lancement de l'application

A ce stade on peut déployer une feature

- Git push origin feature
- Clone la feature dans /opt/apps/feature
- Préparer la structure des répertoires
- Creer le venv et installer les dépendances
- Faire un lien vers le template uwsgi
- uWSGI emperor lance l'application feature

Il nous manque:

- créer des sous domaines dynamique et parler avec le serveur web

FastRouter

- Proxy/LoadBalanceur/Routeur
- Parle le protocole uWSGI
- Possiblités de configuration illimitées
- Cache clés/valeurs

FastRouter

- Proxy/LoadBalanceur/Routeur
- Parle le protocole uWSGI
- Possiblités de configuration illimitées

Pour faire court:

```
uwsgi --fastrouter  /tmp/fastrouter.socket  \ --
fastrouter-subscription-server 127.0.0.1:9999
```

Toujours privilégier les sockets unix au tcp localhost !

Nginx

```
server {
        listen              80;
        server_name         dev.com *.dev.com;
        location / {
                include /etc/nginx/uwsgi_params;
                uwsgi_param UWSGI_FASTROUTER_KEY $host;
                uwsgi_pass unix: /tmp/fastrouter.socket;
        }
}
```

Fichier de conf template (App Django)

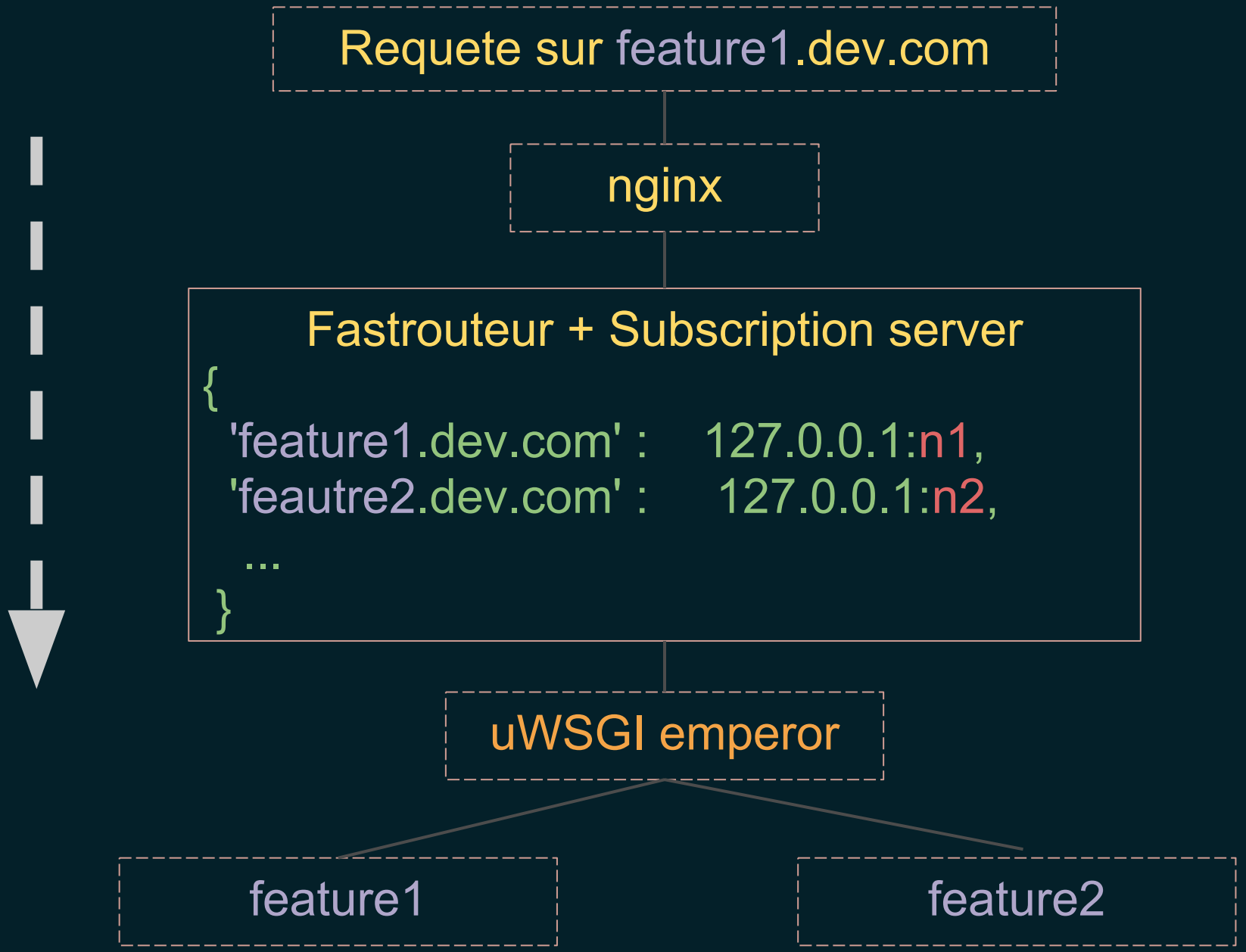
```
[uwsgi]
...
socket = 127.0.0.1:0
subscribe-to = 127.0.0.1:9999:%n.dev.com
```

Fichier de conf template (App Django)

```
[uwsgi]
...
socket = 127.0.0.1:0
subscribe-to = 127.0.0.1:9999:%n.dev.com
```

/opt/apps/feature1/feature1.ini

2. Déploiement agile avec uWSGI



HTOP

On deploie feature1

```
├─ supervisord
│   ├─ /uwsgi --fastrouter ... --emperor /opt/apps/*/*.ini
│   │   ├─ /usr/local/bin/uwsgi   #fastrouter
│   │   └─ /usr/local/bin/uwsgi   #master
│   │     ├─uwsgi --ini /opt/apps/feature1/feature1.ini
```

HTOP

On deploie feature2

```
├─ supervisord
│   ├─ /uwsgi --fastrouter ... --emperor /opt/apps/*/*.ini
│   │   ├─ /usr/local/bin/uwsgi   #fastrouter
│   │   └─ /usr/local/bin/uwsgi   #master
│   │     ├─uwsgi --ini /opt/apps/feature1/feature1.ini
│   │     └─uwsgi --ini /opt/apps/feature2/frature2.ini
```

Fin

twitter: @sp4ke
email: spike@jib.li
slides: sp4ke.com/pythondevops