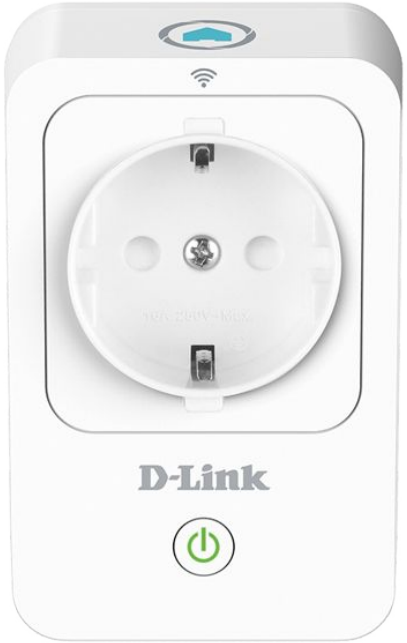
Python + Ansible = ♥

Gonéri Le Bouder
Ansible Montéal
mai 2017

GET YOUR
HOME
TALKING



Un peu de contexte

Mon D-Link DSP w215 smart plug

- Une catastrophe en terme de sécurité
- Se connecte au Wifi et attend des ordres
- Attend les ordres depuis Internet
- Peut se contrôler depuis le réseau local

Un peu de contexte

Mon D-Link DSP w215 smart plug

- Une catastrophe en terme de sécurité
- Se connecte au Wifi et attend des ordres
- Attend les ordres depuis Internet
- **Peut se contrôler depuis le réseau local**

Un module Python existe pour le manipuler (pyW215)

```
from pyW215.pyW215 import SmartPlug

sp = SmartPlug('192.168.1.110', '******')

# Allumer l'interrupteur et l’étendre
sp.state = 'ON'
sp.state = 'OFF'
```

Et si j’utilisais Ansible pour allumer ma lampe?

Les différents types de modules/plugins (1/3)

- Callback Plugins
  - Traite les retours d'exécution
- Connection Plugins
  - SSH
- Lookup Plugins
  - Par example: with_items

Les différents types de modules/plugins (2/3)

- Vars Plugins
  - Pour ajouter des variables supplémentaires
- Filter Plugins
  - Filtre pour Jinja2, par exemple: `{{ myvar | ipv4 }}`
- Modules
  - Par exemple: `command: cat /etc/motd`
- Action plugins
  - Par exemple: `template: src=/mytemplates/foo.j2 dest=/etc/file.conf`

Les différents types de modules/plugins (3/3)

- ~~Vars Plugins~~
  - ~~Pour ajouter des variables supplémentaires~~
- ~~Filter Plugins~~
  - ~~Filtre pour Jinja2, par exemple: {{ myvar | ipv4 }}~~
- **Modules**
  - **Par exemple: command: cat /etc/motd**
- ~~Action plugins~~
  - ~~Par exemple: template: src=/mytemplates/foo.j2 dest=/etc/file.conf~~

Un module Ansible (1/5)

Ma lib est écrite pour Python 3.

Ne pas oublier que le module peut être lancé sur une machine distante:

- Réduire les dépendances
- Éviter d’imposer un virtual-env
- Un seul script sera copié (pas de sous module)
- On utilise la sortie standard pour communiquer

```
#!/usr/bin/python3
from ansible.module_utils.basic import *
```

Un module Ansible (2/5)

```
if __name__ == '__main__':
    global module
    module = AnsibleModule(
        argument_spec={
            'host': { 'required': True, 'type': 'str' },
            'password': { 'required': True, 'type': 'str' },
            'state': { 'required': True, 'type': 'str' }
        },
        supports_check_mode=False      ⇐ Ne PAS lancer en “mode à froid” (dry mode)
    )
```

Un module Ansible (3/5)

```
if module.params['state'] in ('present', 'on', 'ON'):
    expectation = 'ON'
elif module.params['state'] in ('absent', 'off', 'OFF'):
    expectation = 'OFF'
else:
    module.fail_json(msg="Invalid parameter for state")

try:
    from pyW215.pyW215 import SmartPlug
except ImportError:
    module.fail_json(msg="The module depends on pyW215")
```

Un module Ansible (4/5)

```
if module.params['state'] in ('present', 'on', 'ON'):
    expectation = 'ON'
elif module.params['state'] in ('absent', 'off', 'OFF'):
    expectation = 'OFF'
else:
    module.fail_json(msg="Invalid parameter for state")

try:
    from pyW215.pyW215 import SmartPlug
except ImportError:
    module.fail_json(msg="The module depends on pyW215")
```

⇐ ne PAS utiliser print() !

Un module Ansible (5/5)

```
sp = SmartPlug(module.params['host'], module.params['password'])
changed = expectation != sp.state
sp.state = expectation
module.exit_json(changed=changed)
```

Les chemins

Enregistrer le fichier dans : ./plugins/library/smartplug.py

Et créer un fichier local ansible.cfg:

[defaults]

library = ./plugins/library

Exécution du nouveau module

A ce stade je peux déjà lancer mon module:

```
ansible localhost -m smartplug -a 'host=somewhere password=secret state=on'
```

Mon playbook pour allumer la lumière

```yaml
---
- hosts: localhost
  tasks:
    - smartplug:
        host: '192.168.1.120'
        password: '851004'
        state: 'on'
```

ansible-playbook -vvv set_light.yml

Et si il fait sombre ?!

Et si il fait sombre ?!

Oui ? Et si il y a des nuages ?

Et si il fait sombre ?!

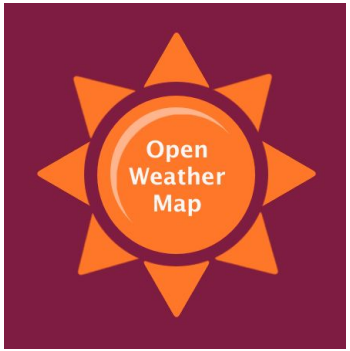
Oui ? Et si il y a des nuages ?

Et si le soleil est couché ?

Comment indiquer la météo à Ansible ? (1/3)

Openweathermap offre une API REST/JSON, par exemple:

http://api.openweathermap.org/data/2.5/weather?q=Montréal,CA&appid=mon_secret&units=metric



Comment indiquer la météo à Ansible ? (2/3)


```
{
  "base": "stations",
  "clouds": {
    "all": 5
  },
  "cod": 200,
  "coord": {
    "lat": 45.5,
    "lon": -73.68
  },
  "dt": 1491703200,
  "id": 6077246,
  "main": {
    "humidity": 55,
    "pressure": 1012,
    "temp": 3.33,
    "temp_max": 4,
    "temp_min": 2
  },
```


<= Et voilà la couverture nuageuse en %

Comment indiquer la météo à Ansible ? (3/3)


```
"name": "Montréal",
"sys": {
  "country": "CA",
  "id": 3829,
  "message": 0.0397,
  "sunrise": 1491733166,
  "sunset": 1491780822,
  "type": 1
},
"visibility": 24140,
"weather": [
  {
    "description": "clear sky",
    "icon": "02n",
    "id": 800,
    "main": "Clear"
  }
],
"wind": {
  "deg": 230,
  "speed": 2.1
```


<= Je sais quand il fait nuit !

Facile, Il faut faire un curl et parser du JSON

Ansible n’est pas vraiment fait pour ça … enfin si avec le filtre to_json mais bon

… Python serait tellement plus chouette ! ☺

Les différents types de modules/plugins (3/3)

- ~~Vars Plugins~~
  - ~~Pour ajouter des variables supplémentaires~~
- ~~Filter Plugins~~
  - ~~Filtre pour Jinja2, par exemple: {{ myvar | ipv4 }}~~
- ~~Modules~~
  - ~~Par exemple: command: cat /etc/motd~~
- Action plugins
  - Par exemple: template: src=/mytemplates/foo.j2 dest=/etc/file.conf

Un module Action

- Cette fois on veut un traitement local
- L’API est un peu différente
- Pas d'exécution par une machine distante
- Évalué par le process Python du Ansible papa

Un module Action (1/2)

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from __future__ import (absolute_import, division, print_function)
__metaclass__ = type

from ansible.plugins.action import ActionBase
from ansible.utils.vars import merge_hash
try:
    from __main__ import display
except ImportError:
    from ansible.utils.display import Display
    display = Display()
import os
import requests
```

Un module Action (2/2)

```
class ActionModule(ActionBase):
    def run(self, tmp=None, task_vars=None):
        if task_vars is None:
            task_vars = dict()


        results = super(ActionModule, self).run(tmp, task_vars)
        location = self._task.args.get('location', None)
        units = self._task.args.get('units', 'metric')
```

Un module Action (2/2)

```
display.vv("Looking for weather via 'openweathermap': location=%s" % location)

r = requests.get(
    'http://api.openweathermap.org/data/2.5/weather',
    params={
        'q': location,
        'appid': os.environ['OPENWEATHERMAP_ID'],
        'units': units})
```

Un module Action (2/2)

```
results = merge_hash(
    results,
    r.json())
return results
```

- Ansible utilise la valeur retournée ici (dictionnaire Python)
- On peut donc polluer la sortie standard avec des print() partout
- mais on ne le fait pas pour ne pas contourner les callbacks :-)

J’enregistre mon Action

Dans le fichier plugins/action/weather.py

J’édite mon ansible.cfg pour ajouter ce nouveau chemin:

```
[defaults]
action_plugins = ./plugins/action
library = ./plugins/library
```

Plus qu’a !

```yaml
---
- hosts: localhost
  tasks:
    - name: Récupère la météo de Montréal
      weather:
        location: "Montréal,CA"
      register: meteo
    - set_fact:
        plug_state: 'on'
        when: (meteo.clouds.all | int > 75)
              or
              (meteo.sys.sunrise < ansible_date_time.epoch | int)
    - set_fact:
        plug_state: 'off'
        when: ansible_date_time.hour | int < 21
    - smartplug:
        host: '192.168.1.120'
        password: 'mon secret'
        state: '{{ plug_state }}'
```

Quelques idées pour continuer

Déclencher le playbook automatiquement, par exemple:

- Toutes les 20 minutes
- Avec un unit et un timer pour systemd

Quelques idées pour continuer

Une notification à la fin, par exemple avec

- Le module “mail”
- Ou telegram.