



簡単！ JupyterNotebook+Ansibleを使った 作業手順書自動生成

2019/7/25

ビッグロース株式会社

前野洋史

手順書の作成、楽にしたいですね！

- 定型手順書は自動生成しちゃいましょう
 - ❓ 手順書の定型化→自動生成！

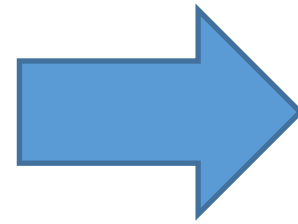
- 理想の手順書自動生成システム

- ❓ 簡単

- 学習難易度が低い
 - 業務の空いた時間で作れる

- ❓ シンプル

- 運用チームも使いやすい



JupyterNotebook

+

Ansible

が良さそう！

Jupyter Notebookとは

- WEBブラウザ上で動作する対話型プログラム実行ツール

- ❑ Python/Ansible等のコードを**ブラウザで実行可能**
- ❑ 実行結果は**ファイルとして保存可能**
- ❑ 出力したファイルを**別環境で動かすことも可能**

PythonSample

```
In [26]: #python
print("Hello!! Janoger!!")

Hello!! Janoger!!
```

AnsibleSample

```
In [27]: #play
name: test job
hosts: localhost
connection: local
gather_facts: false
vars_files:
- var
```

```
In [28]: #vars var
janog44:
location: Kobe
```

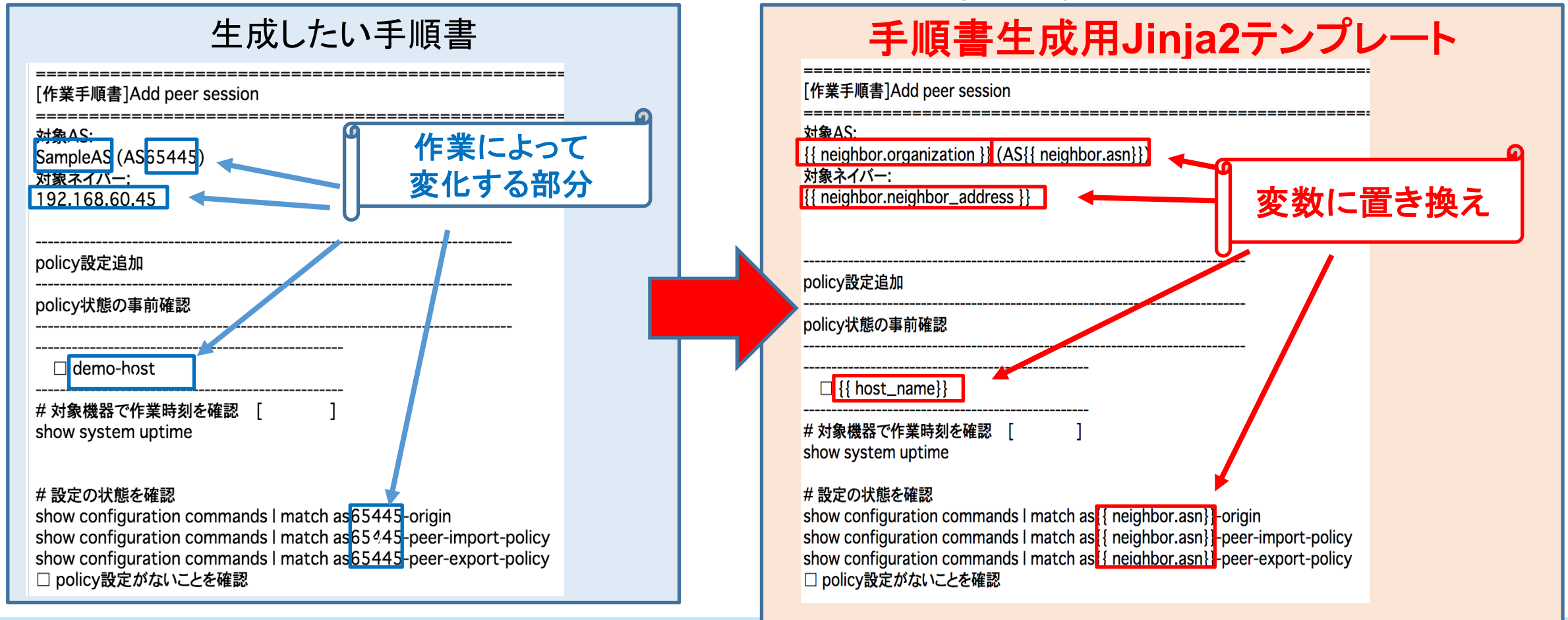
```
In [29]: #task
name: display janog44.location
debug: msg={{ janog44.location }}
```

```
TASK [display janog44.location] *****
ok: [localhost] => {
  "msg": "Kobe"
}
```

JupyterNotebook+Ansibleを使った手順書生成(1/3)

- 手順書生成用Jinja2テンプレートファイルの作成

※Jinja2=Pythonのテンプレートエンジン



JupyterNotebook+Ansibleを使った手順書生成(2/3)

- JupyterNotebookファイルの作成

設定パラメータの定義

```
#inventory
[vyos-44-self]
192.168.60.44

[vyos-44-peer]
192.168.60.45

[localhost]
localhost

[vyos:vars]
ansible_user=vyos
ansible_ssh_pass=vyos
ansible_network_os=vyos
```

Ansibleの Inventory情報

```
#ansible.cfg
[persistent_connection]
command_timeout = 360
connect_timeout = 360
connect_retry_timeout = 360

[defaults]
host_key_checking=False
```

Ansibleの Ansible.cfgの情報

```
#vars var
neighbor:
organization: SampleAS
asn: 65445
neighbor_address: 192.168.60.45
host_name: demo-host
```

Ansibleの varsの情報

[作業手順書生成]Add peer session

```
#play
name: generate operation-file job
hosts: localhost
connection: local
gather_facts: false
vars_files:
- var
```

Ansibleの playの情報

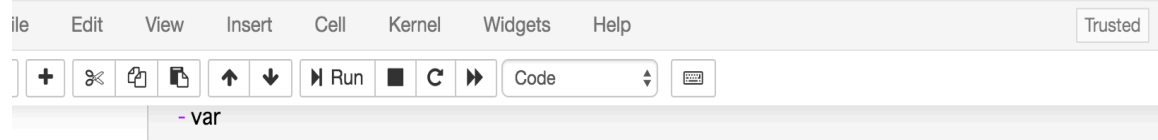
```
#task
name: generate operation-file
template:
src: /home/notebook/notebooks/JANOG44/vyos-operation-template-not-for.j2
dest: /home/notebook/notebooks/JANOG44/not-for.txt
```

Ansibleの taskの情報

手順書生成用jinjaテンプレートから
ファイルを生成するPlaybookを書くだけ😊

JupyterNotebook+Ansibleを使った手順書生成(3/3)

- 作成したJupyterNotebookファイルを実行



```
In [19]: #task
name: generate operation-file
template:
src: /home/notebook/notebooks/JANOG44/vyos-operation-template-not-for.j2
dest: /home/notebook/notebooks/JANOG44/not-for.txt

TASK [generate operation-file] *****
changed: [localhost] => {
  "checksum": "a77c842e779ba3cb1cbc2510f4c5bad40159192e",
  "dest": "/home/notebook/notebooks/JANOG44/not-for.txt",
  "diff": [],
  "gid": 1000,
  "group": "notebook",
  "md5sum": "b1de80e5b74da2f8bb8deb29c64953dc",
  "mode": "0644",
  "owner": "notebook",
  "size": 992,
  "src": "/home/notebook/.ansible/tmp/ansible-tmp-1563847997.57-103160491850366/source",
  "state": "file",
  "uid": 1000
}
```

ブラウザ画面上部の

▶ Run

ボタンを押していく

設定パラメータの読み込みや
Ansibleの実行が順番に実施される

実行結果もブラウザに表示

```
File Edit View Language

1 =====
2 [作業手順書]Add peer session
3 =====
4 対象AS:
5 SampleAS (AS65445)
6 対象ネイバー:
7 192.168.60.45
8
9 -----
10 policy設定追加
11
12 policy状態の事前確認
13 -----
14 ☐ demo-host
15 -----
16
17 # 対象機器で作業時刻を確認 [      ]
18 show system uptime
19
20
21 # 設定の状態を確認
22 show configuration commands | match as65445-origin
23 show configuration commands | match as65445-peer-import-policy
24 show configuration commands | match as65445-peer-export-policy
25 ☐ policy設定がないことを確認
26
```

手順書完成！

JupyterNotebook+Ansibleを使った手順書生成(まとめ)

手順書生成用 Jinja2テンプレートファイル を作成

```
jupyter operation-template.j2 ✓ 15時間前
File Edit View Language

1 =====
2 [作業手順書]Add peer session
3 =====
4 対象AS:
5 {% for neighbor in data.neighbors %}
6 {{ neighbor.organization }} (AS{{ neighbor.asn}})
7 {% endfor %}
8 対象ネイバー:
9 {% for neighbor in data.neighbors %}
10 {{ neighbor.neighbor_address }}
11 {% endfor %}
12
13 -----
14 policy設定追加
15 -----
16 policy状態の事前確認
17 -----
18 ☐ {{ data.host_name }}
19 -----
20
21 # 対象機器で作業時刻を確認 [      ]
22 {% if data.host_os == 'vyos' %}
23 show system uptime
24 {% endif %}
25
```

手順書生成用 JupyterNotebookファイル を作成&実行

```
#vars var
data:
neighbors:
- organization: SampleAS
  asn: 65445
  neighbor_address: 192.168.60.45
  ip_ver: ipv4
  host_name: demo-host
  host_os: vyos
```

設定パラメータを定義

[作業手順書生成]Add peer session

```
#play
name: generate operation-file job
hosts: localhost
connection: local
gather_facts: false
vars_files:
- var
```

設定パラメータと
テンプレートファイルから
手順書を生成する処理を記述

```
#task
name: generate operation-file
template:
src: /home/notebook/notebooks/JANOG44/operation-template.j2
dest: /home/notebook/notebooks/JANOG44/operation-file.txt
```

手順書完成！

```
jupyter operation-file.txt ✓ 1時間前
File Edit View Language

1 =====
2 [作業手順書]Add peer session
3 =====
4 対象AS:
5 SampleAS (AS65445)
6 対象ネイバー:
7 192.168.60.45
8
9 -----
10 policy設定追加
11 -----
12 policy状態の事前確認
13 -----
14 ☐ demo-host
15 -----
16
17 # 対象機器で作業時刻を確認 [      ]
18 show system uptime
19
20
21 # 設定の状態を確認
22 show configuration commands | match as65445-origin
23 show configuration commands | match as65445-peer-import-policy
24 show configuration commands | match as65445-peer-export-policy
25 ☐ policy設定がないことを確認
26
```

JupyterNotebookを使うメリット

- 全てWebブラウザ上で完結可能
 - ❑ 専用Webアプリの開発/メンテ不要
 - ❑ CLIに不慣れな人でも手順書を作りやすい
- 作成した手順書の信用性向上
 - ❑ Notebookの実行結果からテンプレの利用を確認できる
 - ❑ レビューはパラメータの確認のみでよい
- 手順書の実行基盤としても使える
 - ❑ 設定自動投入/状態取得
 - ❑ 実行結果がそのまま作業ログに

```
#task
name: execute "set bgp simple config"
vyos_config:
lines:
- set protocols bgp 65444 neighbor 192.168.60.45 remote-as 65444
register: result
```

```
TASK [execute "set bgp simple config"] *****
ok: [192.168.60.44] => {
  "commands": [],
  "filtered": []
}
```

```
#task
name: show task result
debug: var=result
```

```
TASK [show task result] *****
ok: [192.168.60.44] => {
  "result": {
    "changed": false,
    "commands": [],
    "failed": false,
    "filtered": []
  }
}
```

設定投入例

```
#task
name: execute "show ip bgp summary"
vyos_command:
commands:
- show ip bgp summary
register: result
```

```
TASK [execute "show ip bgp summary"] *****
```

```
#task
name: show task result
debug: var=result["stdout_lines"].0
```

```
TASK [show task result] *****
ok: [192.168.60.44] => {
  "result[\"stdout_lines\"]: [
    "BGP router identifier 192.168.60.44, local AS number 65444",
    "IPv4 Unicast - max multipaths: ebgp 1 ibgp 1",
    "RIB entries 3, using 288 bytes of memory",
    "Peers 1, using 4560 bytes of memory",
    "",
    "Neighbor      V  AS MsgRcvd MsgSent  TblVer  InQ OutQ Up/Down S
    "192.168.60.45  4 65445    0      0    0  0 0 never Active "
    ""
  ],
  "Total number of neighbors 1"
}
```

状態取得例

まとめ

- JupyterNotebookは簡単/シンプルで便利
- 今後は定型手順書増加→Notebook化
 - ☐ CheckBox等も組み合わせてよりよいものに
- 千里の道も一歩から
 - ☐ 小さな改善を積み重ねて大きな業務改善へ

```
: #python
import ipywidgets as widgets
from IPython.display import display

def disp_multiple_checkbox(items):
    check_item_list=[]

    for item in items:
        check_item_list.append(widgets.Checkbox(value=False,description=item))

    for checkbox_item in check_item_list:
        display(checkbox_item)

items = [
    u"対象ルータが正しいこと",
    u"asnが正しいこと",
    u"neighbor_addressが正しいこと"
]
disp_multiple_checkbox(items)
```

チェックボックス
サンプル

- ☐ 対象ルータが正しいこと
- ☐ asnが正しいこと
- ☐ neighbor_addressが正しいこと

**JupyterNotebookの活用例、
それ以外でも社内の業務改善自動化例などあれば
お話しいただけると幸いです！！**