

Ansibleハンズオンセミナー (Network編) 補足資料

レッドハット株式会社
テクニカルセールス本部
西日本ソリューションアーキテクト部

田中 耕輔 <ktanaka@redhat.com>

2019.9.11 / Rev 2.0.1

演習 1~4

Ansible Engine

Ansible Engineの演習 ~ もくじ

Exercise 1 - Exploring the lab environment

~ ハンズオン環境をチェックしましょう

Exercise 2 - Execute your first network automation playbook

~ プレイブックを書いてみよう

Exercise 3 - Use Ansible facts on network devices

~ ファクトを使ってみよう

Exercise 4 - Use Jinja to template network configurations

~ ネットワーク設定をテンプレートで記述しよう

Exercise. 1 (1/2) - 環境のチェック

step1. ~/networking-workshop/ ディレクトリがありますか？

```
$ cd networking-workshop
```

step2. ansibleのバージョンや設定を確認

```
$ ansible --version
ansible 2.8.1
  config file = /home/student1/.ansible.cfg
  ...
```

step3. ansibleの設定ファイル(ユーザ独自)

```
$ cat ~/.ansible.cfg
...
inventory = XXXXX
...
```

Exercise. 1 (2/2)

step4. インベントリファイルの内容を見る

```
[all:vars]
```

```
...
```

```
[group:vars]
```

```
...
```

```
[group:children]
```

```
...
```

```
[group]
```

```
client-1
```

```
client-2
```

step5. インベントリ中の各種変数(ホスト変数/グループ変数)

- ansible_host
- private_ip
- ansible_user=ec2-user
- ansible_network_os=ios
- ansible_connection=network_cli

Exercise. 2 (1/3) - プレイブックを書く

step1. あらかじめ用意されているプレイブックを確認 (playbook.yml)

```
$ cat playbook.yml
---
- name: snmp ro/rw string configuration
  hosts: cisco
  gather_facts: no

  tasks:

    - name: ensure that the desired snmp strings are present
      ios_config:
        commands:
          - snmp-server community ansible-public RO
          - snmp-server community ansible-private RW
```

step3. プレイブックを実行してみる

```
$ ansible-playbook playbook.html
```

Exercise. 2 (2/3)

step4. 設定が反映されているかどうかの確認

```
$ ssh rtr1
rtr1#show run | i snmp
snmp-server community ansible-public RO
snmp-server community ansible-private RW
```

step5. "冪等性" (idempotency)の確認

```
$ ansible-playbook playbook.html
...
PLAY RECAP *****
rtr1 : ok=1  changed=0  unreachable=0  failed=0  skipped=0  rescued=0...
```

step6. playbook.html のカスタマイズ(設定項目の追加)

```
ios_config:
  commands:
    - snmp-server community ansible-public RO
    - snmp-server community ansible-private RW
    - snmp-server community ansible-test RO
```

Exercise. 2 (3/3)

step7. dry-run (--check)と、コマンド実行時の詳細出力 (-v, -vv, -vvv)

```
$ ansible-playbook playbook.html --check --verbose
...
PLAY RECAP *****
rtr1 : ok=1  changed=1  unreachable=0  failed=0  skipped=0  rescued=0...
```

step8. dry-runの確認 (設定が変更されていないことを確認)

```
$ ssh rtr1
rtr1#show run | i snmp
snmp-server community ansible-public RO
snmp-server community ansible-private RW
```

step9 & 10. playbook.html の実行と、設定変更の確認

```
$ ansible-playbook playbook.html
$ ssh rtr1
rtr1#show run | i snmp
snmp-server community ansible-public RO
snmp-server community ansible-private RW
snmp-server community ansible-test RO
```


Exercise. 3 (1/3) - ファクトの利用

step1. モジュールのマニュアル確認 (debug, ios_facts)

```
$ ansible-doc debug  
$ ansible-doc ios_facts
```

→ 収集するfactの範囲を指定する方法を見つけてください*

step2&3. ファクトを収集するプレイブック(facts.yml)の作製

```
---  
- name: gather information from routers  
  hosts: cisco  
  gather_facts: no  
  
  tasks:  
    - name: gather router facts  
      ios_facts:
```

step4. プレイブックを実行してみる

```
$ ansible-playbook facts.html
```

Exercise. 3 (2/3)

step5. ios_factsが収集したファクトを画面で確認 (-v)

```
$ ansible-playbook facts.html -v
...
TASK [gather router facts]
*****
ok: [rtr1] => changed=false
  ansible_facts:
    ansible_net_all_ipv4_addresses:
      - 192.168.35.101
    ...
    ansible_net_iostype: IOS-XE
    ansible_net_memfree_mb: 1853993
    ...
    ansible_net_system: ios
    ansible_net_version: 16.09.02
    discovered_interpreter_python: /usr/bin/python
  ...
PLAY RECAP *****
rtr1      : ok=1  changed=0  unreachable=0  failed=0  skipped=0  ...
```

Exercise. 3 (3/3)

step6&8. 収集したファクトの一部を debug モジュールで出力するよう、
facts.yml ヘタスクを追記して実行

```
---
- name: gather information from routers
  hosts: cisco
  gather_facts: no

tasks:
  - name: gather router facts
    ios_facts:

  - name: display version
    debug:
      msg: "The IOS version is: {{ ansible_net_version }}"

  - name: display serial number
    debug:
      msg: "The serial number is:{{ ansible_net_serialnum }}"
```

(参考)

Fact modules return structured data

```
rtr1#show version
Cisco IOS XE Software, Version 16.09.02
Cisco IOS Software [Fuji], Virtual XE Software (X86_64_LINUX_IOSD-UNIVERSALK9-M), Version 16.9.2, RELEASE SOFTWARE (fc4)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2018 by Cisco Systems, Inc.
Compiled Mon 05-Nov-18 19:26 by mcpre
..
.
<rest of output removed for brevity>
```

```
[student1@ansible ~]$ ansible -m ios_facts rtr1
.<abbreviated output>>
```

```
.
  "ansible_net_iostype": "IOS-XE",
  "ansible_net_memfree_mb": 1853921,
  "ansible_net_memtotal_mb": 2180495,
  "ansible_net_model": "CSR1000V",
  "ansible_net_neighbors": {},
  "ansible_net_python_version": "2.7.5",
  "ansible_net_serialnum": "964A1H0D1RM",
  "ansible_net_system": "ios",
  "ansible_net_version": "16.09.02",
```

Exercise. 4 (1/2) - テンプレートの利用

step1. 変数の定義(~/networking-workshop/group_vars/all.yml)
→ nodes[hostname].Loopback100

```
nodes:
  rtr1:
    Loopback100: "192.168.100.1"
  rtr2:
    Loopback100: "192.168.100.2"
```

step2&3. テンプレートファイルの作製 (template.j2)

```
{% for interface,ip in nodes[inventory_hostname].items() %}
interface {{interface}}
  ip address {{ip}} 255.255.255.255
{% endfor %}
```

Exercise. 4 (2/2)

step4. 機器設定を行うプレイブックの作製 (config.yml)

→ cli_config モジュールを利用 (機種依存の無い設定モジュール)

```
---  
- name: configure network devices  
  hosts: rtr1,rtr2  
  gather_facts: false  
  tasks:  
    - name: configure device with config  
      cli_config:  
        config: "{{ lookup('template', 'template.j2') }}"
```

lookupプラグインをして、外部からデータを取込可能
その際に"template"を指定して、Jinja2 のテンプレート処理を実行

step5&6. プレイブックの実行と、結果の確認

```
$ ansible-playbook config.yml  
...  
rtr1 : ok=1  changed=1  unreachable=0  failed=0  skipped=0  
rtr2 : ok=1  changed=1  unreachable=0  failed=0  
$ ssh rtr1  
rtr1#show ip int br | include Loopback100  
Loopback100      192.168.100.1 YES manual up
```

* gather_subset: {all,hardware, config, interfaces}

(参考)

Network Automation config modules

cli_config (agnostic)

ios_config:

nxos_config:

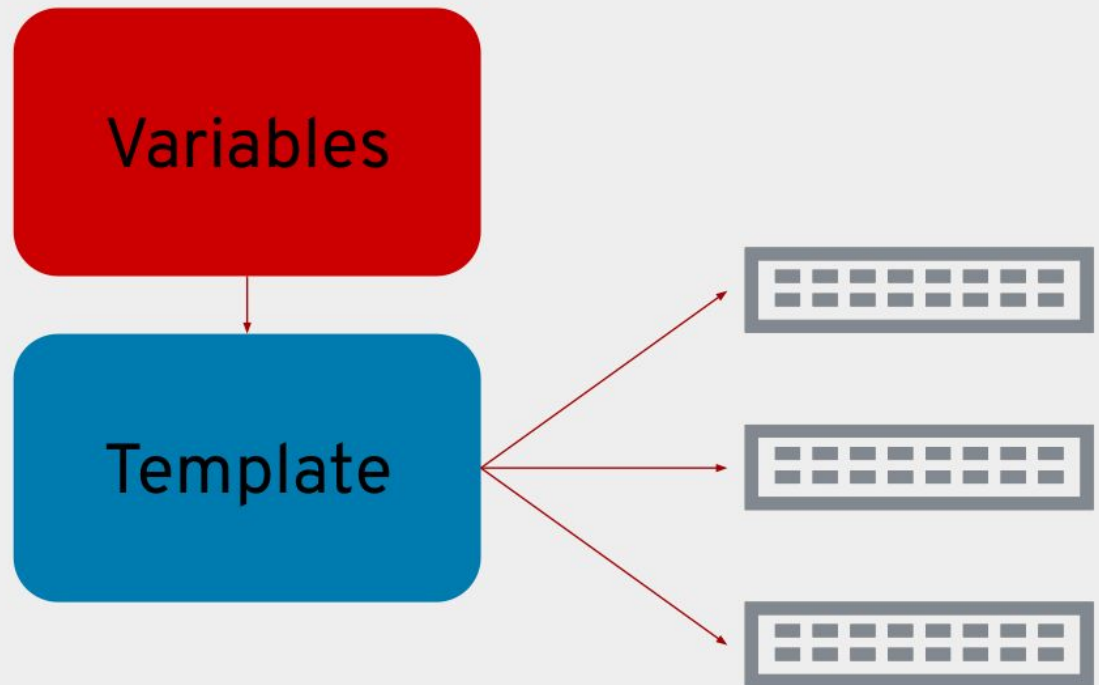
iosxr_config:

eos_config

.

.

*os_config:



(参考)

Jinja2

- Ansible has native integration with the Jinja2 templating engine
- Render data models into device configurations
- Render device output into dynamic documentation

Jinja2 enables the user to manipulate variables, apply conditional logic and extend programmability for network automation.



(参考)

Jinja2 Templating Example (1/2)

Variables

```
ntp_server: 192.168.0.250
name_server: 192.168.0.251
```

Jinja2 Template

```
!
ntp server {{ntp_server}}
!
ip name-server {{name_server}}
!
```

Generated Network Configuration

rtr1

```
!
ip name-server 192.168.0.251
!
ntp server 192.168.0.250
!
```

rtrX

```
!
ip name-server 192.168.0.251
!
ntp server 192.168.0.250
!
```

(参考)

Jinja2 Templating Example (2/2)

Variables

```
nodes:
  rtr1:
    Loopback100: "192.168.100.1"
  rtr2:
    Loopback100: "192.168.100.2"
  rtr3:
    Loopback100: "192.168.100.3"
  rtr4:
    Loopback100: "192.168.100.4"
```

Jinja2 Template

```
{% for interface,ip in nodes[inventory_hostname].items()
%}
interface {{interface}}
    ip address {{ip}} 255.255.255.255
{% endfor %}
```

Generated Network Configuration

rtr1

```
interface Loopback100
  ip address 192.168.100.1
!
```

rtr2

```
interface Loopback100
  ip address 192.168.100.2
!
```

rtrX

```
interface Loopback100
  ip address X
!
```

演習 5~9

Ansible Tower

※Engineの演習が終わってない方も
ご一緒にお願いします

Ansible Towerの演習 - もくじ

Exercise 5 - Explore the Ansible Tower environment

~ Ansible Towerの設定を見てみよう

Exercise 6 - Create an Ansible Tower job template

~ ジョブテンプレートを作る

Exercise 7 - Create an Ansible Tower Survey

~ サーベイ(SURVEY)機能を使ってみる

Exercise 8 - Using the Role Based Access Control (RBAC)

~ ユーザの権限管理

Exercise 9 - Create an Ansible Tower Workflow

~ ワークフロー

Exercise. 5 - Towerの環境確認

step1. Ansible Towerへのログインと、バージョンの確認

→ <https://studentX.NNNN.rhdemo.io>

username:	admin
password:	ansible



step2-4. 設定済の各種オブジェクトの内容を確認

- インベントリー (Inventories)
 - Workshop Inventory: グループ、ホスト、グループ変数、ホスト変数...
- プロジェクト (Projects)
 - Workshop Project: github リポジトリへアクセスしてみる
- 認証情報 (Credentials)
 - Workshop Credential の設定値 → secretなものはencrypt済

Exercise. 6 (1/2)- ジョブテンプレートの作製

step1. (ジョブ)テンプレートをつくる:

+ ジョブテンプレート

- 名前 Backup network configurations
- ジョブタイプ 実行
- インベントリ Workshop Inventory
- プロジェクト Workshop Project
- PLAYBOOK network_backup.yml
- 認証情報 Workshop Credential

→ 保存

step2. (ジョブ)テンプレートの実行:



step3. ジョブ詳細画面の確認

- 「詳細」ペイン (左)
- 「標準出力」ペイン (右) → 出力の展開
- 標準出力ペインのクリック → 構造化された(JSON)形式の出力表示

```
1 {  
2   "_ansible_no_log": false,  
3   "job_template": "RESTORE NETWORK CONFIG",  
4   "changed": true,  
5   "state": "present",  
6   "invocation": {  
7     "module_args": {  
8       "ask_diff_mode": false,  
9       "module_name": "network_cli",  
10      "password_prompt": "Password: ",  
11      "username": "admin",  
12      "transport": "ssh",  
13      "use_persistent_connections": true,  
14      "validate_certs": false  
15    }  
16  }
```

Exercise. 6 (2/2)

step4. ジョブ画面の確認


- メニューから「ジョブ」をクリック
- 「Backup network configurations」ジョブをクリック→ジョブ詳細画面

step5. バックアップファイルが作製されたことを確認

```
$ ls /backup
2019-09-11-15-30
$ cat /backup/2019-09-11-15-30/rtr1
Current configuration : 5625 bytes
...!
version 16.9
service tcp-keepalives-in
...
service password-encryption
!
! [[REST OF OUTPUT REMOVED FOR BREVITY]]
!
```

→ 他の機器についても確認してみましょう

Exercise. 7 (1/3) - サーベイ(Survey)の使用

step1. (ジョブ)テンプレートをつくる:  ジョブテンプレート

- 名前 Network-Banner
- ジョブタイプ 実行
- インベントリ Workshop Inventory
- プロジェクト Workshop Project
- PLAYBOOK network_banner.yml
- 認証情報 Workshop Credential

→ 保存

step2. プレイブックの中身を確認

- [network_banner.yml](#)
- ロール: banner の内容を確認
 - [roles/banner/tasks/main.yml](#)
 - ↑で include されるファイルを確認 → [roles/banner/tasks/](#)

Exercise. 7 (2/3)

step3. Survey(サーベイ)作製:「テンプレート」→"Network-banner"

SURVEY の編集

- プロンプト
- 説明
- 回答の変数名
- 回答タイプ
- 「必須」をチェック

Please enter the banner text

Please type into the text field the desired banner

net_banner

textarea

→ + ADD

もう一つ入力項目を追加:

- プロンプト
- 説明
- 回答の変数名
- 回答タイプ
- 複数の選択オプション

Please enter the banner type

Please choose an option

net_type

複数の選択(単一の選択肢)

[login

motd

login

- DEFAULT ANSWER
- 「必須」をチェック

→ + ADD

→

保存

Exercise. 7 (3/3)

step4. (ジョブ)テンプレートの実行:

- bannerメッセージの入力
- login / motd いずれかを選択

→ 次へ

- 変数の値をプレビュー

→ 起動



```
\      ^  ^
 \      (oo) \_____
      (__)          A ) \/\
                || ----w ||
                ||         ||
```

step5. bannerの変更を確認

```
$ ssh rtr4
```

```
Warning: Permanently added 'rtr4,35.182.247.114' (ECDSA) to the list
```

```
\      ^  ^
 \      (oo) \_____
      (__)          A ) \/\
                || ----w ||
                ||         ||
```

```
Last login: Sun Sep  8 15:31:43 2019 from 34.203.199.203
```

```
localhost>
```

Exercise. 8 (1/3) - ユーザー権限管理(RBAC)

用語の理解:

- 組織 ≡「テナント」(名前空間を分ける最上位の単位)
- チーム 組織内に存在し、ユーザーが所属するグループのこと
- ユーザー 利用者。一つ以上のチームに属することができる
- ロール ユーザーごとに設定する、組織内での役割のこと。
詳細パーミッション設定の集合(組込済のローラー覧は[ココ](#))

step1. ハンズオン環境で定義済の組織を確認する

- →「組織」
 - REDHAT COMPUTE ORGANIZATION
 - REDHAT NETWORK ORGANIZATION

step2. 「REDHAT NETWORK ORGANIZATION」をクリック

- → ユーザーボタンを押して、この組織に所属するユーザーを確認
 - admin, bbelcher, network-admin, network-operator
それぞれのロールは？

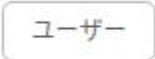
ユーザー

Exercise. 8 (2/3)


step3. メニューから「チーム」をクリック

- Compute T1
- Compute T2
- Netadmin
- Netops

step4. Netopsチームの確認

- Step3で表示された画面で「Netops」をクリック
- 「ユーザー」ボタンをクリック 
 - Netadmin ... RED HAT NETWORK ORGANIZATIONの管理権限を持つ
 - network-operator ... 単なるNetopsチームメンバ(「メンバー」ロール)

step5. network-adminとしてのTowerログイン

- adminユーザーのログアウト→ 
- network-admin としてログイン (password=ansible)
- 「組織」メニューをクリック
 - 何が違うか？他にも admin でログインしていた際と違うところは？

Exercise. 8 (3/3)

step6. チームロールを理解する

- adminとしてログインし直す
- 「インベントリー」→ "Workshop Inventory" →
 - network-admin に対する管理者権限
 - network-operator が所属する**チーム**への利用権限
(インベントリーに対する権限設定の例)

パーミッション

step7. ジョブテンプレートに対する権限設定

- 「テンプレート」メニュー → "Network-Commands" →
 - network-admin ... 管理者権限を持つ
 - network-operator ... 実行のみが可能な権限

パーミッション

step8. network-operator としてのログイン (password=ansible)

- 「テンプレート」メニュー → "Network-Commands"
- admin でログインしていたときと何が違う？

step9. "Network-Commands"ジョブテンプレートの実行

(余裕があれば network-admin でログインし直してSurveyを編集してみる)

Exercise. 9 (1/2) - ワークフロー

(adminでログインし直して)

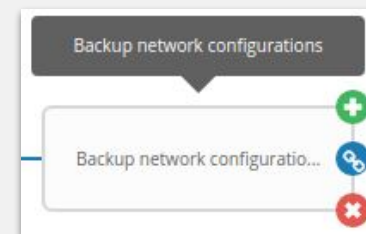
step1. (ジョブ)テンプレートをつくる:  ワークフローテンプレート

- 名前 Workshop Workflow
- 組織 Default
- インベントリー Workshop Inventory

→ 保存

step2&3. ワークフロービジュアライザ画面

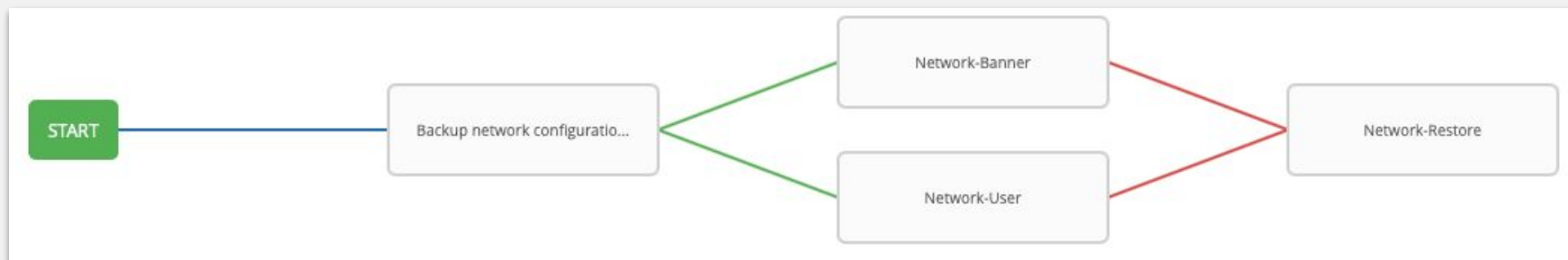
- 「開始」をクリック 
- 右側に「ノード追加」画面が表示される
 - "Backup network configurations"をクリック
 - 「選択」ボタン 
 - "Backup network configuration"ノード上へマウスポインタを移動し、「+」アイコンをクリック
 - 「ノード追加」画面にて "Network-banner" を追加



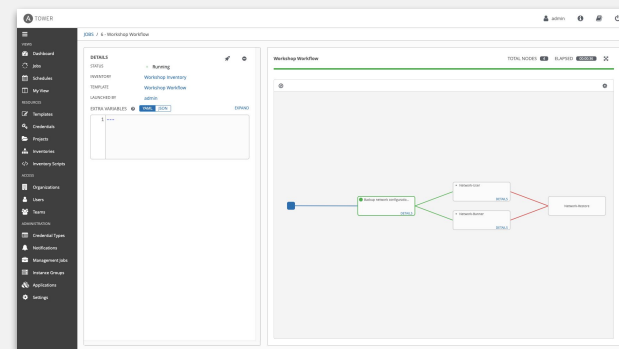
Exercise. 9 (2/2)

step4&5&6. さらにノードを追加:

- 同じ要領で、以下のようなワークフローを作製
 - FAIL時のみ遷移する場合(赤い線)、「実行」の値を"障害発生時"に
 - 他フローに合流(既存ノードへ接続)する場合は、青のリンクボタン



step7. 作製したワークフロージョブを実行



おつかれさまでした！