知っておくと小回りがきく Ansibleの使い方

BBIX株式会社　丸野 達矢

自己紹介

名前：丸野 達矢 (MARUNO Tatsuya / tmaruno)

趣味：ゲーム、バイク、キャンプ、スキー、写真

普段の主な業務：
各メガクラウドとの直接接続における色々

出身/住まい：熊本 / 今は栃木県の北部に在住

【背景と動機】サービス運用の現実

メガクラウドへの直接接続サービスをいくつか展開
提供方法が似てる2つのサービス

① OCX（Open Connectivity eXchange）

- 2022年5月30日よりスタートしたクラウドサービス
- Cloud Connection 機能

→ネットワーク機器への設定は、システム基盤からの全自動運用

② マルチクラウド接続サービス

▪弊社IX利用者向けのオプションサービス

→ネットワーク機器への設定は、”手運用”

→→→手運用部分、まだまだ最適化が進められてない

【背景と動機】Ansibleを使って設定作業自動化

作業の負荷軽減したい

→一旦”Ansible”を使ったネットワーク設定作業自動化

今回Ansibleを使う理由

- 機器へのアクセスが簡単（ SSH接続ができれば基本 OK、エージェントレス）
- すでに利用できる環境が存在してた
- 初期構築用で使われた簡易ツールが存在してた（ただ運用時にそのまま使える状態ではなかった）
- 筆者自身がある程度の知見があった（ただネットワーク機器で　Ansibleを使うのは今回が初めて）
- 今すぐ欲しい
- 一時的な運用改善ツールに留めたい、作り込まない（　OCX側システムから管理することも検討してるため）

Ansibleで設定自動化実施

そのとき得た「知っておくと小回りがきくAnsibleの使い方」を共有します

「知っておくと小回りがきくAnsibleの使い方」

【その1】ansible-playbook実行時、-e (--extra-vars) +“@”を使う

【その2】YAML記述に中括弧 ”{ }” と カンマ ”,” を使う

【その1】ansible-playbook実行時、-e (--extra-vars) +“@”を使う

-e (--extra-vars) +“@”のオプションを使うことで、
ansible-playbook実行時に変数ファイルを指定できる

メリット:

- コンフィグを「使い捨て」前提で運用できる（別でコンフィグバックアップ環境が整ってる前提）
  →ロールバックの実装＆運用建て付けがとっても楽になる

デメリット:

- @以降のパス指定時に、プロンプト上の補完機能が効かない
- 変数ファイルがコンフィグバックアップも兼ねるような Ansible運用には向かない（マルチベンダー想定でコンフィグを一元管理するとか）

※参考資料:

- てくなべ(tekunabe):[Ansible] ansible-playbook コマンドの -e (--extra-vars) オプションは変数ファイル名も指定できる

【その1】ansible-playbook実行時、-e (--extra-vars) +“@”を使う

```
group_vars
 └aws
    └config.yml ( “{{ vlans }}” を定義 )
 └azure
    └config.yml
 └googlecloud
    └config.yml
hosts_vars
 └SW1.yml ( “{{ aws_interface }}” とかを定義 )
 └SW2.yml
 └SW3.yml
 └SW4.yml
roles
 └show—※showまわりのrole実装
 └vlan_add—※vlanまわりのrole実装
hosts
config_aws_vlan.yml
config_azure_vlan.yml
config_googlecloud_vlan.yml
```

■hosts

```
[all:vars]
ansible_ssh_user=xxxxx
ansible_ssh_pass=yyyyy
ansible_become_method=enable
ansible_become_pass=zzzzz

[aws]
SW1 ansible_host=192.168.1.1
SW2 ansible_host=192.168.1.2

[azure]
SW3 ansible_host=192.168.1.3
SW4 ansible_host=192.168.1.4

[googlecloud]
SW1 ansible_host=192.168.1.1
SW2 ansible_host=192.168.1.2
```

■config_aws_vlan.yml

```
- name: Add vlan for AWS
  hosts: aws
  gather_facts: no

  vars:
    - interface: "{{ aws_interface }}"
    - vlans: "{{ vlans }}"

  roles:
    - show
    - vlan_add
    - show
```

【その1】ansible-playbook実行時、-e (--extra-vars) +“@”を使う

```
group_vars
 └─aws
     └─config.yml ( “{{ vlans }}” を定義 )
 └─azure
     └─config.yml
 └─googlecloud
     └─config.yml
hosts_vars
 └─SW1.yml ( “{{ a
```

■プレイブック実行

ansible-playbook -i hosts config_aws_vlan.yml

・group_varsの紐付けが事前にされてるので、
group_vars内パラメータの鮮度を常に保つ必要があり、注意が必要

・既存のコンフィグも含めた管理方法だと、
新規投入と既存コンフィグの違いをつけるのが大変
＝ロールバック処理が大変

【その1】ansible-playbook実行時、-e (--extra-vars) +“@”を使う

```
#### group_vars の廃止
#### ”tmp_vars”という新しいディレクトリを作成

tmp_vars
 └──aws_config_yyyymmdd.yml
```

```
hosts_vars
 └─SW1.yml ( “{{ a
```

■プレイブック実行

```
ansible-playbook
 -i hosts -e @tmp_vars/aws_config_yyyymmdd.yml config_aws_vlan.yml
```

・都度ファイル指定しないと実行されないので、考え方がシンプルになる
・あくまでその時設定したいコンフィグに絞った定義、とすることでロールバック実装も簡単にできる

【その2】YAML記述に中括弧 ”{ }” と カンマ ”,” を使う

中括弧 ”{ }” と カンマ ”,” を使うことで、
インデントを使わずに階層を表現できる

（JSONっぽくかける）

メリット：

- 同じフォーマットで連続する表現をするときに、行数を減らせるので、コンフィグなどの可読性が向上する

デメリット：

- インデントがなくなるので、YAMLの階層自体の可読性は下がる、乱用注意

参考資料：

- 「YAMLの本来の使い方」を仕様から読み取ってみる
- YAML Ain’t Markup Language (YAML™) version 1.2

【その2】YAML記述に中括弧 ”{ }” と カンマ ”,” を使う

before

```
vlan_params:
  - outer_vlan_id: 2001
    internal_vlan_id: 3701
    internal_vlan_name: "AWS"
    mbps: "1000"
  - outer_vlan_id: 2002
    internal_vlan_id: 3702
    internal_vlan_name: "AWS"
    mbps: "1000"
  - outer_vlan_id: 2003
    internal_vlan_id: 3703
    internal_vlan_name: "AWS"
    mbps: "1000"
```

```
- outer_vlan_id: 2004
  internal_vlan_id: 3704
  internal_vlan_name: "AWS"
  mbps: "1000"
- outer_vlan_id: 2005
  internal_vlan_id: 3705
  internal_vlan_name: "AWS"
  mbps: "1000"
- outer_vlan_id: 2006
  internal_vlan_id: 3706
  internal_vlan_name: "AWS"
  mbps: "1000"
```

【その2】YAML記述に中括弧 ”{ }” と カンマ ”,” を使う

after

```
vlan_params:
  - { outer_vlan_id: 2001,  internal_vlan_id: 3701,  internal_vlan_name: "AWS",  mbps: "1000" }
  - { outer_vlan_id: 2002,  internal_vlan_id: 3702,  internal_vlan_name: "AWS",  mbps: "1000" }
  - { outer_vlan_id: 2003,  internal_vlan_id: 3703,  internal_vlan_name: "AWS",  mbps: "1000" }
  - { outer_vlan_id: 2004,  internal_vlan_id: 3704,  internal_vlan_name: "AWS",  mbps: "1000" }
  - { outer_vlan_id: 2005,  internal_vlan_id: 3705,  internal_vlan_name: "AWS",  mbps: "1000" }
  - { outer_vlan_id: 2006,  internal_vlan_id: 3706,  internal_vlan_name: "AWS",  mbps: "1000" }
```

まとめ

【その1】ansible-playbook実行時、-e (--extra-vars) +“@”を使う
【その2】YAML記述に中括弧 ”{ }” と カンマ ”,” を使う

→→→知っておくと便利かも

やっぱりAnsibleお手軽便利！！

No Peering, No Internet.