Lab5 _ Centraliser Avec Ansible

Gérer la sortie standard d'Ansible

On peut agir sur la sortie standard, soit en rajoutant de l'information, soit en supprimant certains paragraphes de la sortie standard du playbook. Nous avons ici un fichier playbook avec des variables, des « tasks » une task « file » pour créer un fichier, mais conditionnée sur l'existence d'une variable qui n'est pas définie.

On peut exécuter des commandes avec la commande « shell », par exemple ici, usr/bin/uptime. Et si on souhaite récupérer le résultat dans une variable, le module « register » permet de stocker le résultat de la commande précédente dans une variable. Ici, register stocke le résultat de « uptime » dans la variable « results ».

Un peu plus loin, dans un module « debug », on va pouvoir afficher la variable results. Attention ici, on affiche la variable results, le module debug est validé que si la « verbosity » est au niveau 2.

On va exécuter tout ça. On va exécuter la commande ansible-playbook sur notre playbook de manière standard puis avec un niveau de verbosité 2. Interpréter les résultats ?

$ ansible-playbook play9.yml
$ ansible-playbook play9.yml -vv

Modifier le fichier config de Ansible et rajouter display_skipped_hosts et la passer à « no », la valeur par défaut étant « yes ».

On va exécuter à nouveau notre dernière commande, Interpréter les résultats ?

```yaml
---
- hosts: web
  vars:
    hostname: control
  tasks:
  - file:
      dest: '{{ha_config}}'
      state: '{{file_state}}'
    when: ha_config is defined

  - shell: /usr/bin/uptime
    register: result

  - name: "Resultat de uptime"
    debug:
      msg: "{{result}}"
      verbosity: 2
```

Gérer le callback de la sortie

Ansible se base sur la notion de « call back » pour tout ce qui est sortie standard. On va voir comment modifier ce mécanisme.

```
---
- hosts: haproxy
  gather_facts: no
  vars:
    ha_config: /home/ansible/hproxy.cfg
  tasks:
  - file:
      dest: '{{ha_config}}'
      state: touch
    when: ha_config is defined

  - shell: /usr/bin/uptime
    register: result

  - name: "Resultat de uptime"
    debug:
      msg: "{{result}}"

```

Maintenant, on va rajouter dans le fichier de config, une variable qui est très intéressante et qui permet de récupérer sur chaque tâche, le temps d'exécution. Cette variable s'appelle « callback_whitelist » et on va utiliser ce qu'on appelle du « profiling », on va utiliser du profiling de tasks.

Ajouter cette ligne dans le fichier config de Ansible

callback_whitelist = profile_tasks

Je sauvegarde, j'exécute mon playbook.

Ceci est un format standard de sortie, on peut le modifier en utilisant, même principe, des variables de callback, cette fois-ci, pour la sortie : stdout_callback et on va utiliser une version minimale de la sortie standard.

Ajouter cette ligne dans le fichier config de Ansible

stdout_callback = minimal

Exécuter le playbook.

Vous avez une version un peu plus réduite de la sortie standard. Changer la valeur minimal par one line »

Pour désactiver la sortie standard :

stdout_callback = minimal

Centraliser les résultats d'exécution

Lorsque vous avez énormément de playbooks qui sont lancés sur plusieurs machines par plusieurs équipes de plusieurs personnes, il faut pouvoir loguer l'activité d'Ansible. Là encore, Ansible propose plusieurs solutions, des solutions basées sur Syslog ou du Logstash, ou autres solutions. Vous pouvez aussi créer votre propre callback de logs. On va commencer par utiliser la solution Syslog pour l'ensemble de l'activité Ansible. Pour cela, il faut éditer le fichier de configuration. Rajouter une callback pour syslog.

Ajouter dans le fichier de config :

Callback_whitelist = syslog_json

[Callback_syslog_json]

Syslog_server = localhost

Tester cette configuration avec un playbook qui permet de créer un fichier.

Puis afficher les messages du log :

sudo tail /var/log/messages, on voit qu'il y a eu ici en fait un message qui provient du module ansible-file qui en fait invoque la création du fichier avec la méthode touch.

Cette fois-ci, on va utiliser non pas syslog au niveau de l'ensemble du module ou des modules Ansible, mais uniquement sur une action particulière. On va utiliser ici le module syslogger qui permet d'envoyer un message au niveau syslog avec une certaine « priority » et une certaine « facility ». Ici, on a choisi d'utiliser un script playbook qu'on avait déjà utilisé auparavant.

```
- name: Syslog Simple Usage
  syslogger:
    msg: "This is my name : {{inventory_hostname}}"
    priority: "debug"
    facility: "daemon"
    log_pid: true
```