Support pédagogique Ansible - 5

Sommaire

- Mise à jour par roulement
- Lancer les actions en parallèle
- Débogage avec Ansible
- Ansible Galaxy
- Gestion du Callback d’affichage
- Optimisation

Mise à jour par roulement (rolling update)

> Il est parfois utile de ne pas exécuter une tâche sur tous les serveurs à la fois mais de travailler par séquence, pour éviter de rendre indisponible tous les serveurs en même temps (pour une grappe de serveurs Web par exemple).

- Utilisation de l'option `serial` pour limiter le nombre d'occurrences à 3

```
- name: "MediaWiki apache configuration"
  hosts: apache
  # 3 hosts at a time
  serial: 3
  tags: "apache"
  gather_facts: no
  roles:
    - role: "mediawiki/configuration"
```

- Un autre exemple avec une combinaison qui permet de faire l'action sur le premier serveur puis sur tous les autres.

```
- name: "MediaWiki apache configuration"
  hosts: apache
  # one, all hosts
  serial: [ "1", "100%" ]
  tags: "apache"
  gather_facts: no
  roles:
    - role: "mediawiki/configuration"
```

Lancer les actions en parallèle

> Par défaut les actions sont effectuées en série les unes à la suite des autres, rendant dépendant le temps d'exécution au serveur le plus lent.
> Il est possible de modifier cela en passant la `strategy` à `free`

```
# by default, ansible will use the 'linear' strategy but you may want to try
#strategy = free
```

Par contre, le gain de temps à l'exécution se traduit par une complexification des sorties d'exécution (callback).

Débogage avec Ansible

> Le mode `debug` permet d'interagir directement sur la sortie du `playbook` (callback) pour debuguer

- Active le mode `debug`

```
$ export ANSIBLE_STRATEGY=debug
```

- Un exemple de `callback` avec le mode `debug`

```
$ cat simple-error.yml
```

```
- hosts: localhost
  tasks:
    - shell: ls /does/not/exist
    - shell: echo {{undefined}}
```

```
$ ansible-playbook simple-error.yml
```

```
PLAY [localhost] ********************

TASK [Gathering Facts] ******************
ok: [localhost]

TASK [shell] ************************************
fatal: [localhost]: FAILED! => {"changed": true,
"cmd": "ls /does/not/exist",
"delta": "0:00:00.002550", "end": "2018-11-07 18:11:38.897731",
"msg": "non-zero return code", "rc": 2, "start": "2018-11-07 18:11:38.895181", "
"stderr_lines": ["ls: impossible d'accéder à '/does/not/exist': Aucun fichier ou

[localhost] TASK: command (debug)>
```

```
 [localhost] TASK: command (debug)> p task.args
{[...]
'_ansible_version': '2.5.1',
 u'_raw_params': u'ls /does/not/exist',
 '_uses_shell': True,
 'warn': True}
```

- Modification de l'argument dans le Debug

```
[localhost] TASK: command (debug)> task.args['_raw_params'] = "ls /dev/null"
```

- Relance le playbook

```
[localhost] TASK: command (debug)> redo
changed: [localhost]

TASK [shell] ****************************
fatal: [localhost]: FAILED! => {"msg": "The task includes an option with an
undefined variable. The error was: 'undefined' is undefined\n\nThe error appears
have been in '/home/alex/.ansible/dawan/simple-error.yml': line 4, column 7, but
may\nbe elsewhere in the file depending on the exact syntax problem.\n\nThe
offending line appears to be:\n\n    - shell: ls /does/not/exist\n
- shell: echo {{undefined}}\n      ^ here\nWe could be wrong, but this one looks
- like it might be an issue with\nmissing quotes.  Always quote template express
-  brackets when they\nstart a value. For instance:\n\n    with_items:\n
-  - {{ foo }}\n\nShould be written as:\n\n    with_items:\n      - \"{{ foo }}\
```

- Modification de la variable dans le Debug

```
[localhost] TASK: shell (debug)> task.vars['undefined']='coucou'
```

- Relance le playbook

```
[localhost] TASK: shell (debug)> r
changed: [localhost]

PLAY RECAP *********************************************
localhost                  : ok=1    changed=2    unreachable=0    failed=0
```

> Plusieurs actions sont possibles: consulter les informations (lettre `p`), modifier les éléments, ajouter des variables (`vars`) ou relancer la tache (`redo`)

Ansible Galaxy

> Galaxy est la plaque tournante pour trouver, réutiliser et partager du contenu Ansible
> https://galaxy.ansible.com/

- Recherche d'un `role` pour installer `Apache` sur `Centos`

```
ansible-galaxy search apache,centos
```

```
Found 939 roles matching your search:

 Name                                                  Description
 ----                                                  -----------
 olibob.apache                                         Installs apache on Centos 7
 mstantoncook.apache                                   Apache 2.x for RedHat/.[...]
 RobertoSolis.apache                                   Apache 2.x for RedHat.[...]
 morten.apache                                         Apache 2.x for RedHat/[...]
 sguter90.apache                                       Apache 2.x for RedHat/[...].
 augustohp.apache                                      Apache 2.x for RedHat/[...]
 Hotelsnl.apache                                       Apache 2.x for [...]
```

- Information sur le `rôle`

```
$ ansible-galaxy info viniciusfs.apache
```

```
Role: viniciusfs.apache
        description: Installs and configures Apache Server
        in CentOS/RHEL systems.
        active: True
[...]
        created: 2017-03-11T23:09:59.548Z
        download_count: 17
        forks_count: 0
[...]
        min_ansible_version: 1.9
        modified: 2018-06-29T20:21:18.492Z
[...]
## Role Variables
* `apache_`:
    - Description: Enable service at boot time
    - Values: `True | False`
    - Default: `True`
## Example Playbook
    - hosts: servers
      roles:
        - { role: viniciusfs.apache }
```

Téléchargement d'un `role`

```
$ ansible-galaxy install viniciusfs.apache -p roles
```

```
- downloading role 'apache', owned by viniciusfs
- downloading role from https://github.com/viniciusfs
/ansible-role-apache/archive/master.tar.gz
- extracting viniciusfs.apache to /home/alex/.ansible/dawan
/galaxy/roles/viniciusfs.apache
- viniciusfs.apache (master) was installed successfully
```

```
.
└── roles
    └── viniciusfs.apache
        ├── defaults
        │   └── main.yml
        ├── handlers
        │   └── main.yml
        ├── meta
        │   └── main.yml
        ├── molecule.yml
        ├── README.md
        ├── tasks
        │   └── main.yml
        ├── templates
        │   ├── httpd.conf.tpl
        │   └── vhost.conf.tpl
        └── tests
            ├── playbook.yml
            └── test_default.py
```

Utilisation d'un fichier de prérequis

- Il faut créer un fichier YAML `requis.yml` dans lequel, on indique les sources des `rôles` à télécharger

```
- src: viasite-ansible.apache
- src: pinkeen.apache
```

```
$ ansible-galaxy install -r requis.yml -p roles
```

```
- downloading role 'apache', owned by viasite-ansible
- downloading role from https://github.com/viasite-ansible
- /ansible-role-apache/archive/v2.1.1.1.tar.gz
- extracting viasite-ansible.apache to /home/alex/.ansible
- /dawan/galaxy/roles/viasite-ansible.apache
- viasite-ansible.apache (v2.1.1.1) was installed successfully
- downloading role 'apache', owned by pinkeen
- downloading role from https://github.com/pinkeen/ansible-role-apache
- /archive/master.tar.gz
- extracting pinkeen.apache to /home/alex/.ansible/dawan/galaxy/roles
- /pinkeen.apache
- pinkeen.apache (master) was installed successfully
```

Suppression d’un rôle

```
$ ansible-galaxy remove viasite-ansible.apache -p roles
```

```
- successfully removed viasite-ansible.apache
```

Les dépendances peuvent être définies dans le `main.yml` `meta` d'un rôle.

> meta/main.yml

```
dependencies:
  - src: geerlingguy.ansible
  - src: git+https://github.com/geerlingguy/ansible-role-composer.git
```

Gestion du Callback d’affichage

> L'affichage des résultats du `playbook` peut être modifié ou personnalisé à l'aide des variables de `CALLBACK`

- Ajout de deux plugins alternatifs

```
$ export ANSIBLE_CALLBACK_WHITELIST=profile_tasks,timer
```

- Relance la configuration de `Apache` sur le serveur `apache2`

```
$ ansible-playbook -i inventaire install-apache.yml \
                   -b -K -e php_install=yes -l apache2
```

> La liste des plugins : `$ ansible-doc -t callback -l`

```
PLAY [Installation apache] ******************************

TASK [Gathering Facts] *********************************
Friday 29 June 2018  23:12:18 +0200 (0:00:00.032)
ok: [apache2]

TASK [apache : apache installation] ************************************
Friday 29 June 2018  23:12:21 +0200 (0:00:02.397)
ok: [apache2]
[...]
PLAY RECAP *************************************
apache2                    : ok=7    changed=2    unreachable=0    failed=0

Friday 29 June 2018  23:13:11 +0200 (0:00:02.412)       0:00:53.055
===============================================================================
apache : install php70 packages ------------------ 41.56s
apache : apache restart --------------------------- 2.41s
Gathering Facts ----------------------------------- 2.40s
apache : remi repo activation --------------------- 2.27s
apache : apache service activation ---------------- 1.56s
apache : apache installation ---------------------- 1.48s
apache : epel activation --------------------------1.35s
Playbook run took 0 days, 0 hours, 0 minutes, 53 seconds
```

Optimisation

Réduire le nombre de connexions SSH

> En ajoutant plus de détails à l'exécution d'un `playbook`, on constate que chaque action déclenche une connexion `SSH`

```
$ ansible -i inventaire all -m ping -l apache1  -vvv
```

```
<apache1> ESTABLISH SSH CONNECTION FOR USER: None
<apache1> SSH: EXEC ssh -C -o ControlMaster=auto -o ControlPersist=60s -o KbdInteractiveAuthenticati
<apache1> (0, '/home/alex\n', '')
<apache1> ESTABLISH SSH CONNECTION FOR USER: None
<apache1> SSH: EXEC ssh -C -o ControlMaster=auto -o ControlPersist=60s -o KbdInteractiveAuthenticati
<apache1> (0, 'ansible-tmp-1541613419.01-122179076156310=/home/alex/.ansible/tmp/ansible-tmp-1541613
<apache1> PUT /home/alex/.ansible/tmp/ansible-local-4999AL__oK/tmpEgvw8f TO /home/alex/.ansible/tmp/
<apache1> SSH: EXEC sftp -b - -C -o ControlMaster=auto -o ControlPersist=60s -o KbdInteractiveAuthen
<apache1> (0, 'sftp> put /home/alex/.ansible/tmp/ansible-local-4999AL__oK/tmpEgvw8f /home/alex/.ansi
<apache1> ESTABLISH SSH CONNECTION FOR USER: None
<apache1> SSH: EXEC ssh -C -o ControlMaster=auto -o ControlPersist=60s -o KbdInteractiveAuthenticati
<apache1> (0, '', '')
<apache1> ESTABLISH SSH CONNECTION FOR USER: None
<apache1> SSH: EXEC ssh -C -o ControlMaster=auto -o ControlPersist=60s -o KbdInteractiveAuthenticati
<apache1> (0, '\r\n{"invocation": {"module_args": {"data": "pong"}}, "ping": "pong"}\r\n', 'Shared c
<apache1> ESTABLISH SSH CONNECTION FOR USER: None
<apache1> SSH: EXEC ssh -C -o ControlMaster=auto -o ControlPersist=60s -o KbdInteractiveAuthenticati
<apache1> (0, '', '')
apache1 | SUCCESS => {
    "changed": false,
    "invocation": {
        "module_args": {
            "data": "pong"
        }
    },
    "ping": "pong"
}
```

Pipelining

Quand Ansible exécute une tache :

- Ansible génère un script Python basé sur le module appelé.
- Il copie le script python sur le serveur
- Il exécute le script

Avec le `pipelining = True` , Ansible exécute le script Python en le connectant à la session SSH au lieu de le copier.

Mettre `pipelining = True` s'est réduire le nombre de connexions `SSH` et donc gagner du temps.

```
$ grep pipelining /etc/ansible/ansible.cfg
```

```
# Enabling pipelining reduces the number of SSH operations required to
#pipelining = False
```

Attention, Bug version 2.2 et supérieure : la section dans le fichier de configuration a changé.

Créer un ansible.cfg dans le répertoire de travail :

```
$ vi ansible.cfg
```

```
[connection]
pipelining = True
```

Il faut modifier les paramètres `SUDO` du serveur pour que le `Pipelining` fonctionne :

Pour désactiver l'option `requiretty` sur le serveur distant :

```
echo 'Defaults:alex !requiretty' > /etc/sudoers.d/disable-requiretty
```

```
ansible -i inventaire all -m ping -l apache1  -vvv
```

```
Using /home/alex/.ansible/dawan/var/ansible.cfg as config file
Parsed /home/alex/.ansible/dawan/var/inventaire inventory source with ini plugin
META: ran handlers
Using module file /usr/lib/python2.7/dist-packages/ansible/modules/system/ping.py
<apache1> ESTABLISH SSH CONNECTION FOR USER: None
<apache1> SSH: EXEC ssh -C -o ControlMaster=auto -o ControlPersist=60s -o KbdInteractiveAuthenticati
<apache1> (0, '\n{"invocation": {"module_args": {"data": "pong"}}, "ping": "pong"}\n', '')
apache1 | SUCCESS => {
    "changed": false,
    "invocation": {
        "module_args": {
            "data": "pong"
        }
    },
    "ping": "pong"
}
```

Multiplexage SSH

Lorsque le multiplexage SSH est utilisé, plusieurs sessions SSH peuvent se faire sur la même connexion TCP. Du coup, la négociation de la connexion TCP est faite uniquement la première fois (se qui économise du temps).

Comment ça marche

- A la première connexion SSH, OpenSSH démarre une connexion principale.
- OpenSSH crée un socket Unix (socket de contrôle) associé.
- A la prochaine connexion SSH, OpenSSH utilisera le socket de contrôle pour communiquer avec l'hôte au lieu d'établir une nouvelle connexion TCP.

> La connexion principale reste ouverte pendant une durée définie dans les configurations, sans activité, le socket sera fermé une fois le temps ecoulé.

Mise en oeuvre

Bien que Ansible peut gérer le multiplexage SSH (dans ansible.cfg), il est préférable de le mettre dans la configuration OpenSSH.

Fichier à modifier `ssh/config`

```
Host myserver.example.com
        ControlMaster auto
        ControlPath /tmp/%r@%h:%p
        ControlPersist 10m
```

> Vérifier le PID du processus Master de connexion

```
ssh -O check server.org
```

```
Master running (pid=12114)
```

> Contrôle le processus en question

```
ps aux | grep 12114
```

```
al 12114 0.0 0.0 49368 2492 ? Ss 12:57 0:00 ssh: /tmp/alex@server.org:22 [mux]
```

> On vérifie la présence du fichier socket utilisé par SSH

```
$ ls /tmp/alex@superalex.domont.org\:22
```

```
/tmp/alex@superalex.domont.org:22
```

Augmenter le nombre d’exécutions simultanées

> Par défaut, le traitement des serveurs est réalisé par série de 5.
> Il est possible d'augmenter cette valeur, en tenant compte des capacités disponibles sur le node manager (RAM, CPU..).

```
$ grep fork /etc/ansible/ansible.cfg
```

```
#forks          = 5
```

Fact Caching

Il est possible de désactiver par défaut les `gather_facts` :

```
[defaults]
gathering = explicit
```

> Les facts seront remontés uniquement si la valeur de `gather_facts: True`

ou de les mettre en cache dans un fichier au format Json:

```
[defaults]
# les nouveaux serveurs seront scannés implicitement et mises en cache
gathering = smart
# 24-hour timeout, adjust if needed
fact_caching_timeout = 86400
fact_caching = jsonfile
fact_caching_connection = /tmp/ansible_fact_cache
```

> https://docs.ansible.com/ansible/2.5/reference_appendices/config.html#ansible-configuration-settings

Taches concurrentes avec Async

Utile pour ne pas attendre qu'une tache soit terminée pour en lancer une autre (parallélisme de taches) ou si l'exécution d'une tache excède la limite de session SSH (SSH timeouts)

```
- name: install git
        apt: name=git update_cache=yes
        become: yes
- name: clone Linus's git repo
        git:
        repo: git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git
        dest: /home/vagrant/linux

        # Si la tache prend plus de 3600s Ansible arrête la tache
        async: 3600

        # Indique à Ansible qu'il peut passer à la suite
        poll: 0

        # Avec async, il faut utiliser Register pour capturer
        # le ansible_job_id de async afin de connaitre le status
        register: linux_clone
```

```
- name: install several packages
        apt:
        name: "{{ item }}"
        with_items:
                - apt-transport-https
                - ca-certificates
                - linux-image-extra-virtual
                - software-properties-common
                - python-pip
        become: yes

- name: wait for linux clone to complete
        # Interroge le module async_status pour connaitre l'etat du Job
        async_status:
        # On stocke l'etat du Job dans la variable result
                jid: "{{ linux_clone.ansible_job_id }}"
        register: result

        # Le module async_status n'interroge qu'une seule fois le status.
        # Nous devons spécifier une clause Until afin qu'elle continue
        # à être interrogée jusqu'à la fin du travail
        # ou jusqu'à ce que le nombre de tentatives spécifié soit épuisé.
        until: result.finished
        retries: 3600
```