ANSIBLE TYPES
SCALAIRES, LISTES, DICTIONNAIRES & STRUCTURES (1/2)

- Ansible manipule 4 grands types : scalaires, listes /tableaux , dictionnaires, structures

  - Les **types scalaires** (text, int, bool, …)

  - Les **listes ou tableaux**

Code : *déclaration d'une variable version de type liste*

```
vars:
  version:
    - v1
    - v2
```

```
vars:
  version: ["v1", "v2"]
```

En général plus visuel sous cette forme

Code : *accès à une valeur d'une liste*

```
current_version: "{{ version[0] }}"
```

Pour tester le type d'une variable : *ma_variable **is** string / number / float … / mapping / iterable*

Pour aller plus loin : https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html

ANSIBLE TYPES
SCALAIRES, LISTES, DICTIONNAIRES & STRUCTURES (2/2)

- Les **dictionnaires**

Code : *déclaration d'une variable users de type dictionnaire*

```
vars:
  users:
    - usr1: maria
    - usr2: peter
```

```
vars:
  users: { "usr1" : "maria", "usr2" : "peter"}
```

Code : *accès à une valeur d'un dictionnaire*

```
current_user: "{{ users['usr1'] }}"
current_user: "{{ users.usr1 }}"
```

- Les **structures** imbriquées (nested)

Code : *déclaration d'une variable cidrs de type nested*

```
vars:
  cidrs:
      production:
        vpc: "172.31.0.0/16"
      staging:
        vpc: "10.0.0.0/24"
```

Code : *accès à une valeur d'une structure*

```
current_cidr: "{{ cidrs['production']['vpc'] }}"
```


Pour aller plus loin : https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html

ANSIBLE VARIABLES
VARIABLES BUILT-IN

- Ansible propose plusieurs variables réservées qui sont pré-renseignées par Ansible
- Ces variables ne peuvent être forcées par le script
- Parmi celles-ci :
  - **groups** : un dictionnaire de tous les groupes & serveurs de l'inventaire
  - **group_names** : la liste des groupes auxquels appartient le serveur managé en cours
  - **hostvars** : un dictionnaire avec l'ensemble des variables pour chaque serveur de l'inventaire
  - **inventory_hostname** : le nom du serveur managé en cours

Les variables précédentes permettent par exemple de récupérer les hosts d'un cluster composés de plusieurs éléments lors de son déploiement.


Pour aller plus loin : https://docs.ansible.com/ansible/latest/reference_appendices/special_variables.html

ANSIBLE FACTS
PLAY ET FAITS (1/2)

- Ansible peut récupérer des informations très détaillées sur les hosts appartenant à l'inventaire.
- Ces informations sont accessibles via le dictionnaire built-in **ansible_facts**
- Cette récupération est automatique en début de chaque script.
  Elle peut être désactivée.

Code : *exemple d'un script désactivant une collecte de faits*

```
- hosts: groupe44
  gather_facts: false
  tasks:
  - ...
```

Si les facts ne sont pas utilisés il vaut mieux désactiver.
Attention cependant car c'est très pratique ;)

Pour aller plus loin : https://docs.ansible.com/ansible/latest/user_guide/playbooks_vars_facts.html

ANSIBLE FACTS
PLAY ET FAITS (2/2)

- Une fois les faits sur un hôte collectés, on peut les utiliser

Code : *exemple d'un script utilisant un fait*

```
- hosts: all
  tasks:
  - name: compilation de l'application
    command: "make -j {{ ansible_processor_cores }}"
```

- La liste des facts disponibles peut être trouvé directement au sein de la documentation ansible

https://docs.ansible.com/ansible/latest/user_guide/playbooks_vars_facts.html#ansible-facts

Ne pas hésiter à parcourir l'ensemble des facts disponibles pour en avoir une idée globale

ANSIBLE FACTS
MODULE SETUP

- Le module **setup** permet de récupérer les faits

Code : *appel du module setup en filtrant certains attributs*

```
ansible all -i inventories/formation -m setup -a
'filter=ansible_distribution,ansible_distribution_version,
ansible_os_family,ansible_processor,ansible_python'
```

ansible_processor est une liste (cf. [])

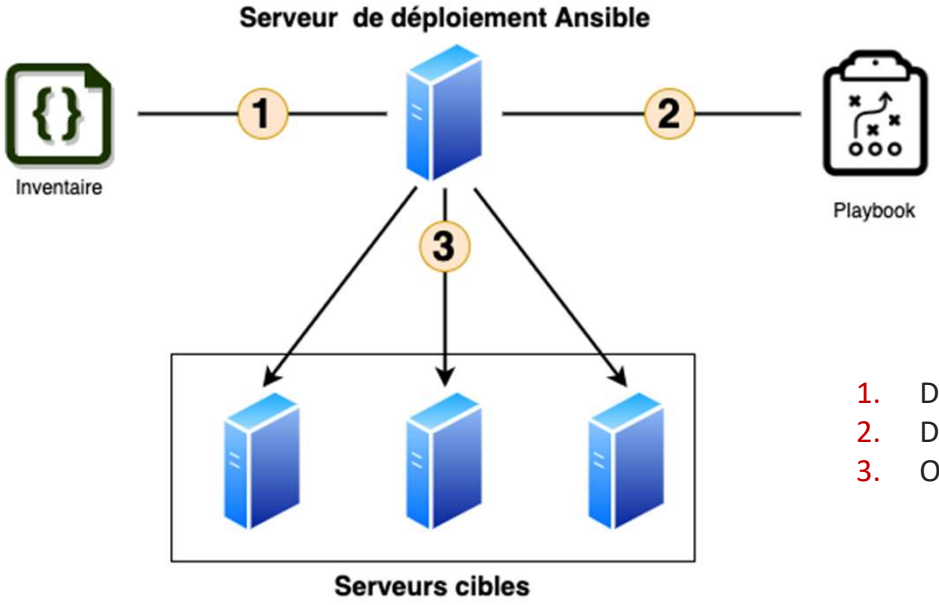
ansible_python est un dictionnaire (cf. {})


```
"ansible_facts": {
            "ansible_distribution": "Ubuntu",
            "ansible_distribution_version": "20.04",
            "ansible_os_family": "Debian",
            "ansible_processor": [
                        "0",
                        "GenuineIntel",
                        "Intel(R) Core(TM) i5-8365U CPU @ 1.60GHz",
                        "1",
                        "GenuineIntel",
                        "Intel(R) Core(TM) i5-8365U CPU @ 1.60GHz"
            ],
            "ansible_python": {
                        "executable": "/usr/bin/python3",
                        "has_sslcontext": true,
                        "type": "cpython",
                        "version": {
                                    "major": 3,
                                    "micro": 10,
                                    "minor": 8,
                                    "releaselevel": "final",
                                    "serial": 0
                        },
                        "version_info": [
                                    3,
                                    8,
                                    10,
                                    "final",
                                    0
                        ]
            },
            "discovered_interpreter_python": "/usr/bin/python3"
}
```


Pour aller plus loin :
https://docs.ansible.com/ansible/latest/collections/ansible/builtin/setup_module.html

PLAYBOOK
DÉFINITION (1/2)

- Objectifs
  - Décrire un **enchainement rejouable** d'appel à des modules dans un fichier
  - Attribuer les commandes à jouer en fonction des machines et/ou groupes de l'inventaire
  - **Eviter de taper les commandes à la main**



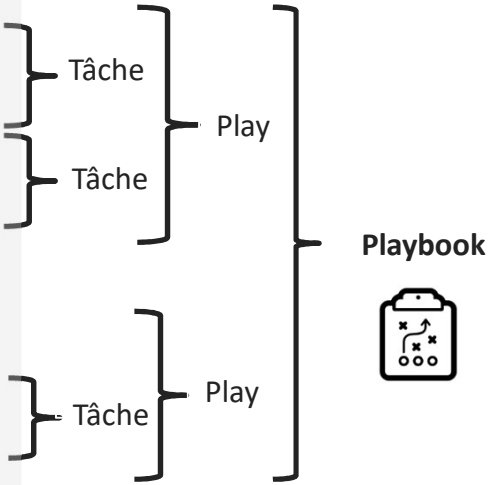
1. Définition de l'accès aux serveurs cibles (Inventaire)
2. Description de l'état cible des machines (Playbook)
3. Orchestration des appels (Ansible)

PLAYBOOK
DÉFINITION (2/2)

***Code :** structure d'un playbook*

```
---
# Mon super play
- hosts: all
  tasks:
  - name: fait un truc
    module1:
      argument: valeur44
  - name: fait un autre truc
    module2:
      argument: valeur66

# Un autre play
- hosts: groupe2:groupe3
  tasks:
  - name: fait un machin
    module3:
      argument: valeur99
```



Le fichier Playbook est au format YAML

***Code :** exemple d'installation d'un serveur web Nginx*

```
---
- hosts: webservers
  tasks:
  - name: Install Nginx
    package:
      name: nginx
  - name: template conf file
    template:
      src: nginx.conf.j2
      dest: /etc/nginx/nginx.conf
  - name: enable / start Nginx
    service:
      name: nginx
      state: started
      enabled: true
```

TIP En général, 1 Playbook = 1 Play

PLAYBOOK
EXÉCUTION (1/3)

- Ansible se connecte à l'hôte et récupère les faits
- Il traduit les tâches en commandes, qu'il transmet aux machines via SSH
- 4 résultats possibles :

Code : *ensemble des états retour possible*

```
changed
ok
skipped
failed
```

`changed`	L'état constaté n'est pas l'état attendu, Ansible à exécuté avec succès la tâche pour atteindre l'état cible.
`ok`	L'état constaté est l'état attendu. Aucune action n'a été réalisée. (**idempotence**)
`skipped`	Une condition liée à l'exécution de la tâche n'était pas réalisée pour réaliser la tâche.
`failed`	Une erreur est survenue durant l'exécution de la tâche.

PLAYBOOK
EXÉCUTION (2/3)

- Une **tâche** doit être **intégralement exécutée sur toutes les machines** concernées avant de passer à la suivante
- Un **play** doit être **intégralement exécuté sur toutes les machines concernées** avant de passer au suivant
- Un **playbook** est **exécuté au moyen d'une stratégie** qui défini le flux d'exécution des tâches

notions expert

Stratégie	Description
linear (default)	Le Playbook s'exécute de manière linéaire, chaque tâche est exécutée dans l'ordre et la tâche suivante n'est pas lancée tant que tous les hôtes n'ont pas terminé la tâche en cours.
debug	L'exécution se fait dans un mode de débogage qui permet à l'utilisateur **d'exécuter interactivement** le playbook à des fins de dépannage.
free	Le playbook est exécuté de manière linéaire, mais chaque hôte exécute les tâches à son propre rythme et n'a pas besoin que tous les hôtes aient terminé une tâche avant de passer à la suivante.
host_pinned	Fonctionnant comme la stratégie free, host_pinned exécutera les tâches aussi vite qu'il le peut sur le nombre d'hôtes spécifié dans le mot-clé serial (batch de machines), en démarrant immédiatement un hôte dès qu'un autre se termine.

***Code :** forcer la stratégie à « free »*

```
- hosts: groupe44
  strategy: free
  tasks:
  - ...
```

***Code :** Modifier le fichier de configuration d'Ansible*

```
[defaults]
strategy = free.    # forcer la stratégie à « free »
```

Pour aller plus loin : https://docs.ansible.com/ansible/latest/user_guide/playbooks_strategies.html

PLAYBOOK
EXÉCUTION (3/3)

notions expert

- Un play **est par défaut joué en parallèle** sur un ensemble de machines (par défaut 5) via l'option **fork**.

Code : *forcer le // à 10*

```
$ ansible-playbook -f 10 my_playbook.yml
```

Code : *Modifier le fichier de configuration d'Ansible*

```
[defaults]
forks = 10.    # forcer le // à 10
```

- Il peut aussi être joué **par lotissement**, utile notamment lors de rolling update

Code : *exemple d'un play par lot de 3 serveurs*

```
- hosts: groupe44
  serial: 10
  max_fail_percentage: 25%
  tasks:
  - ...
```

N lots comprenant au max. 10 serveurs du groupe44

Pour chaque lot, si plus de 25% du lot est en échec, l'exécution sera abandonnée

Code : *exemple d'un play par lot de 20% d'un groupe de serveurs*

```
- hosts: groupe44
  serial: 20%
  tasks:
  - ...
```

5 lots comprenant 20% des serveurs du groupe44

Pour aller plus loin : https://docs.ansible.com/ansible/latest/user_guide/playbooks_strategies.html

PLAYBOOK

APPELER UN PLAYBOOK

- Appeler un playbook sans paramètre

Code : *appel d'un playbook portant sur un inventaire*

```
$ ansible-playbook -i inventories/dev playbook.yml
```

- Appeler plusieurs playbooks

Code : *appel de plusieurs playbooks*

```
$ ansible-playbook -i inventories/dev playbook1.yml playbook2.yml
```


Pour aller plus loin : https://docs.ansible.com/ansible/latest/cli/ansible-playbook.html#ansible-playbook

PLAYBOOK

1ER DÉPLOIEMENT VS MISE À JOUR

Code : *exemple d'installation d'un serveur web Nginx*

```
---
- hosts: webservers
  tasks:
  - name: Install Nginx
    package:
      name: nginx
  - name: template conf file
    template:
      src: nginx.conf.j2
      dest: /etc/nginx/nginx.conf
  - name: enable / start Nginx
    service:
      name: nginx
      state: started
      enabled: true
```

PLAYBOOK

1ER DÉPLOIEMENT VS MISE À JOUR

Code : *exemple d'installation d'un serveur web Nginx*

```
---
- hosts: webservers
  tasks:
  - name: Install Nginx
    package:
      name: nginx
  - name: template conf file
    template:
      src: nginx.conf.j2
      dest: /etc/nginx/nginx.conf
  - name: enable / start Nginx
    service:
      name: nginx
      state: started
      enabled: true
```

1ère exécution

changed

changed

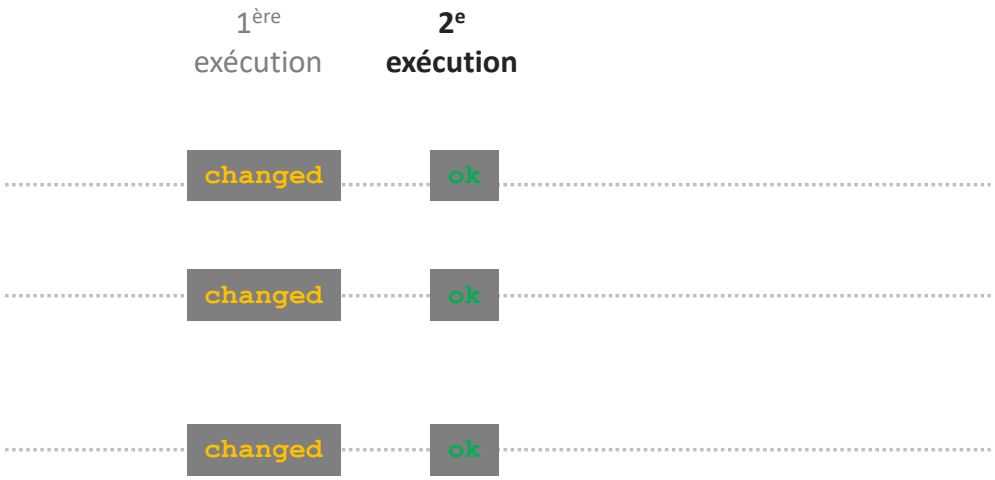
changed

PLAYBOOK

1ER DÉPLOIEMENT VS MISE À JOUR

Code : *exemple d'installation d'un serveur web Nginx*

```
---
- hosts: webservers
  tasks:
  - name: Install Nginx
    package:
      name: nginx
  - name: template conf file
    template:
      src: nginx.conf.j2
      dest: /etc/nginx/nginx.conf
  - name: enable / start Nginx
    service:
      name: nginx
      state: started
      enabled: true
```

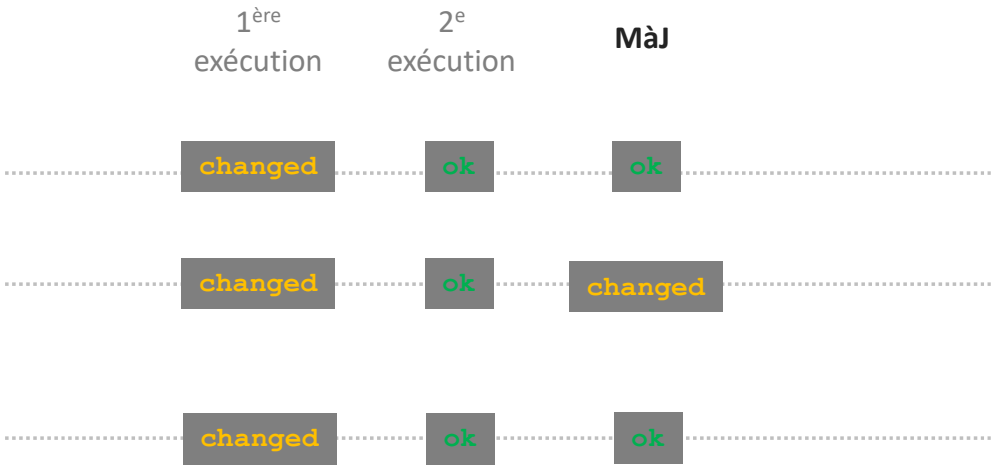


PLAYBOOK

1ER DÉPLOIEMENT VS MISE À JOUR

Code : *exemple d'installation d'un serveur web Nginx*

```
---
- hosts: webservers
  tasks:
  - name: Install Nginx
    package:
      name: nginx
  - name: template conf file
    template:
      src: nginx.conf.j2
      dest: /etc/nginx/nginx.conf
  - name: enable / start Nginx
    service:
      name: nginx
      state: started
      enabled: true
```

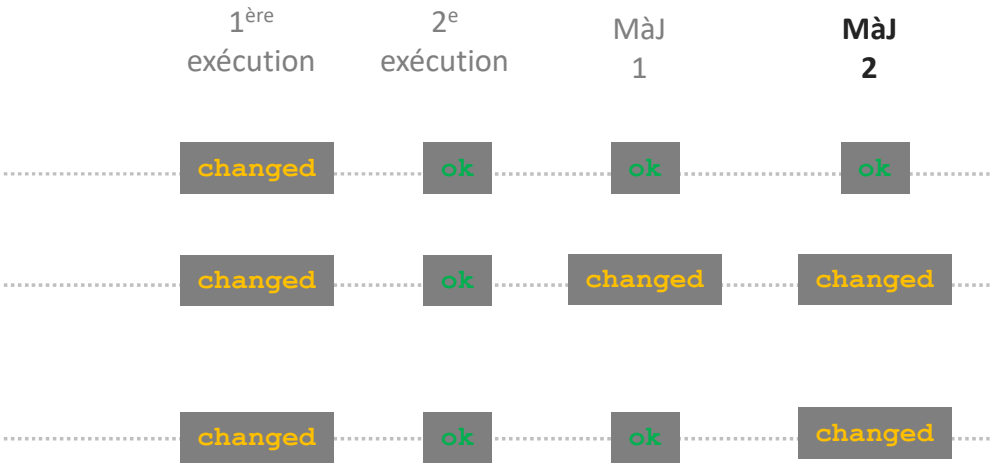


PLAYBOOK

1ER DÉPLOIEMENT VS MISE À JOUR

Code : *exemple d'installation d'un serveur web Nginx*

```
---
- hosts: webservers
  tasks:
  - name: Install Nginx
    package:
      name: nginx
  - name: template conf file
    template:
      src: nginx.conf.j2
      dest: /etc/nginx/nginx.conf
  - name: enable / start Nginx
    service:
      name: nginx
      state: restarted
      enabled: true
```

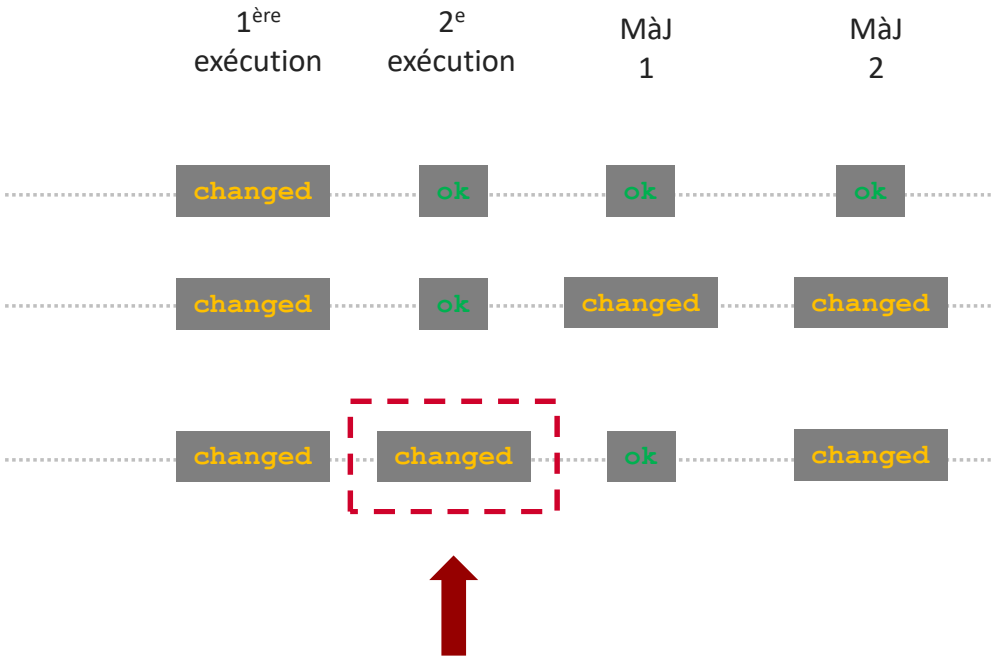


PLAYBOOK

1ER DÉPLOIEMENT VS MISE À JOUR

Code : *exemple d'installation d'un serveur web Nginx*

```
---
- hosts: webservers
  tasks:
  - name: Install Nginx
    package:
      name: nginx
  - name: template conf file
    template:
      src: nginx.conf.j2
      dest: /etc/nginx/nginx.conf
  - name: enable / start Nginx
    service:
      name: nginx
      state: restarted
      enabled: true
```



Si on relance l'exécution 2 avec la modification précédente

DIVERS
PLAY ET UTILISATEURS

- Un play peut être lancé avec sudo via l'option « **become** »

Code : *exemple d'un play lancé en sudo*

```
- hosts: groupe44
  become: true
  tasks:
  - ...
```

A noter : become permet de lancer les tâches en tant que root via la commande sudo. Les utilisateurs autorisés à utiliser sudo sont configurés par les administrateurs.

- « **become_user** » indique sous quel utilisateur le play sera lancé

Code : *exemple d'un play lancé en tant qu'un utilisateur toto*

```
- hosts: groupe44
  become: true
  become_user: toto
  tasks:
  - ...
```

A noter : become_user permet de préciser l'utilisateur bénéficiant de l'élévation de privilège.
Equivalent de la commande « su » sous linux

Pour aller plus loin : https://docs.ansible.com/ansible/latest/user_guide/become.html

DIVERS
PLAY ET REGISTER

- Il peut être utile de capturer la sortie des modules d'un play exécuté par Ansible via l'instruction **register.**

Code : *exemple d'un module exploitant register*

```
- hosts: all
  tasks:
  - name: print hello world
    command: echo "hello there"
    register: echo_return
  - debug: msg="stdout={{ echo_return.stdout }}"
  - debug: msg="stderr={{ echo_return.stderr }}"
```

Permet d'afficher les sorties standards

- Différentes propriétés sont accessibles au sein de la variable déclarée via register :

```
"echo_return": {
  "failed": false,
  "changed": true,
  "cmd": [...],
  "rc": 0,
  "start": "...",
  "end": "...",
  "delta": "...",
  "stderr": "...",
  "stderr_lines": [ …
  ],
  "stdout": "…",
  "stdout_lines": […
  ],
  "warnings": [
  ]
}
```

Le module est-il en échec ou a-t-il changé l'état du host ?

La commande exécutée et son code retour

L'horaire de début, de fin et le temps écoulé pour exécuter le module

Les sorties standards et d'erreur avec les éventuels warnings

*std*_lines correspond à std*.split('\n')*

Pour aller plus loin : https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html#registering-variables

LINTING (1/2)
ANSIBLE-LINT

- Ansible propose un utilitaire « **ansible-lint** » pour s'assurer du respect de certaines règles de développement built-in

Code : *exemple d'usage d'ansible-lint*

```
> ansible-lint -p workshop-tp/tp2
```

A noter : si l'argument de –p est un répertoire, ansible-lint vérifie l'ensemble des fichiers présents dans le répertoire (et sous répertoire)

- Ces règles sont regroupées par catégorie

Code : *exemple de résultat de ansible-lint*

nom de la catégorie [nom de la règle]

```
WARNING  Listing 10 violation(s) that are fatal
workshop-tp/tp2/myplaybook.yml:16: yaml[trailing-spaces]: Trailing spaces
workshop-tp/tp2/myplaybook.yml:17: yaml[trailing-spaces]: Trailing spaces
workshop-tp/tp2/myplaybook.yml:24: yaml[trailing-spaces]: Trailing spaces
workshop-tp/tp2/myplaybook.yml:26: name[play]: All plays should be named.
workshop-tp/tp2/myplaybook.yml:26: yaml[colons]: Too many spaces after colon
workshop-tp/tp2/myplaybook.yml:30: yaml[trailing-spaces]: Trailing spaces
workshop-tp/tp2/myplaybook.yml:33: fqcn[action]: Use FQCN for module actions, such `community.postgresql.postgresql_user`.
workshop-tp/tp2/myplaybook.yml:37: yaml[truthy]: Truthy value should be one of
workshop-tp/tp2/myplaybook.yml:44: fqcn[action]: Use FQCN for module actions, such `community.postgresql.postgresql_db`.
workshop-tp/tp2/myplaybook.yml:62: yaml[trailing-spaces]: Trailing spaces
Read documentation for instructions on how to ignore specific rule violations.

                Rule Violation Summary
 count tag                  profile    rule associated tags
     1 name[play]           basic      idiom
     1 yaml[colons]         basic      formatting, yaml
     5 yaml[trailing-spaces] basic     formatting, yaml
     1 yaml[truthy]         basic      formatting, yaml
     2 fqcn[action]         production formatting

Failed after min profile: 10 failure(s), 0 warning(s) on 1 files.
```

A noter : il est possible de lancer les règles d'un seul profil avec l'option --profile <nom du profil>

Profil : min < basic < moderate < safety < shared < production

Un profil de niveau supérieur embarque les règles du niveau inférieur

Pour aller plus loin : *https://ansible.readthedocs.io/projects/lint/*

LINTING (2/2)
ANSIBLE-LINT

- L'argument ansible-lint qui va vous sauver : **--write**

Code : *exemple d'usage d'ansible-lint*

```
> ansible-lint -p <myplaybook> --write
```

--write va corriger automatiquement certaines erreurs de votre script ansible. Il va aussi reformater le script pour être consistant.

Code : *ansible-lint* ***sans write***

Rule Violation Summary

count	tag	profile	rule associated tags
4	name[play]	basic	idiom
62	yaml[colons]	basic	formatting, yaml
2	yaml[indentation]	basic	formatting, yaml
2	yaml[trailing-spaces]	basic	formatting, yaml
20	yaml[truthy]	basic	formatting, yaml
3	name[casing]	moderate	idiom
5	risky-file-permissions	safety	unpredictability
1	ignore-errors	shared	unpredictability
3	no-changed-when	shared	command-shell, idempotency
2	fqcn[action-core]	production	formatting
2	fqcn[action]	production	formatting

Code : *ansible-lint* ***avec write***

Rule Violation Summary

count	tag	profile	rule associated tags
4	name[play]	basic	idiom
3	name[casing]	moderate	idiom
5	risky-file-permissions	safety	unpredictability
1	ignore-errors	shared	unpredictability
3	no-changed-when	shared	command-shell, idempotency
2	fqcn[action-core]	production	formatting
2	fqcn[action]	production	formatting

Pour aller plus loin : *https://ansible.readthedocs.io/projects/lint/*

RÉCAPITULATIF

Qu'est ce qu'un playbook ?

Qu'est-ce qu'une tâche ?

Quels sont les états finaux possible d’une tâche ?

Qu'est-ce que la programmation par état ?

Comment lance-t-on l’exécution d'un playbook ?