Comprendre l’inventaire Ansible

Inventaire Ansible

Objectifs de certification

RHCE EX294 (RHEL8)

Si vous poursuivez des objectifs de certification ce document tente de répondre à ceux-ci :

- **2. Maîtrise des composants de base d’Ansible**
  - 2.1. Inventaires
- **3. Installation et configuration d’un nœud de contrôle Ansible**
  - 3.4. Créer et utiliser des inventaires statiques pour définir des groupes d’hôtes
- **9. Utilisation des fonctions Ansible avancées**
  - 9.3. Utilisation de variables et de facts Ansible

1. Inventaire Ansible

1.1. Définition d’un inventaire Ansible

Un inventaire est une source de données connue d’avance sur les cibles de gestion Ansible organisée en groupes. Les tâches sont exécutées pour des hôtes ou des groupes d’hôtes dans un inventaire défini.[1]

A quoi sert l’inventaire ?

1. Sélectionner les hôtes qui subissent des tâches.

2. Attribuer des valeurs communes ou spécifiques de variables à des hôtes.

La portée d'un jeu au sein du livre de jeu est limitée aux groupes d'hôtes définis dans l'inventaire (option `--limit` de `ansible-playbook`). Cela est aussi vrai pour le mode "ad-hoc" pour l'exécution de tâches "ad-hoc" (argument de la commande `ansible`).

Sans inventaire défini[2], Ansible est inutilisable sur des machines distantes !

On trouve habituellement l'inventaire sous forme de fichiers en format YAML ou INI mais n'importe quelle source de données peut faire office d'inventaire.

L'inventaire par défaut est habituellement situé dans le fichier `/etc/ansible/hosts`.

1.2. Sources d'inventaire

Un inventaire peut être une collection d'hôtes définis, dans une liste, dans un fichier plat, dans un dossier organisé de manière conventionnelle ou être nourri d'un script dynamique (qui interroge un CMDB par exemple) qui génère une liste de périphériques à cibler dans un livre de jeu. Il peut donc être statique, soit défini d'avance ; il peut aussi se créer dynamiquement et même mis à jour dynamiquement.

Finalement, n'importe quelle source de données peut devenir un inventaire qu'Ansible comprend grâce aux **plugins d'inventaire**. Par défaut, les plugins d'inventaire[3] suivants sont activés :

- `host_list` : une simple liste d'hôte,
- `script` : un script qui construit un inventaire dynamique contre une source de données,
- `yaml` : l'inventaire en format de fichier de données YAML,
- `ini` : l'inventaire en format de fichier de données INI.

La commande `ansible-config` nous permet de vérifier cette configuration :

```
ansible-config view | grep -B 2 -A 2 'inventory plugins'
```

```
[inventory]
# enable inventory plugins, default: 'host_list', 'script', 'yaml', 'ini'
#enable_plugins = host_list, virtualbox, yaml, constructed
```

Voici des fournisseurs supportés par un inventaire dynamique : BSD Jails, DigitalOcean, Google Compute Engine, Linode, OpenShift, **OpenStack**, Ovirt, SpaceWalk, Scaleway, Vagrant (à ne pas confondre avec le "provisioner" dans Vagrant, qui est préféré), **Zabbix**, ...

La commande `ansible-doc -t inventory -l` permet de lister les plugins d'inventaire disponibles :

Plugin d’inventaire Ansible	Description
advanced_host_list	Parses a ‘host list’ with ranges
auto	Loads and executes an inventory plugin specified in a YAML config
aws_ec2	EC2 inventory source
aws_rds	rds instance source
azure_rm	Azure Resource Manager inventory plugin
cloudscale	cloudscale.ch inventory source
constructed	Uses Jinja2 to construct vars and groups based on existing inventory
docker_machine	Docker Machine inventory source
docker_swarm	Ansible dynamic inventory plugin for Docker swarm nodes
foreman	foreman inventory source
gcp_compute	Google Cloud Compute Engine inventory source
generator	Uses Jinja2 to construct hosts and groups from patterns
gitlab_runners	Ansible dynamic inventory plugin for GitLab runners
hcloud	Ansible dynamic inventory plugin for the Hetzner Cloud
host_list	Parses a ‘host list’ string
ini	Uses an Ansible INI file as inventory source
k8s	Kubernetes (K8s) inventory source
kubevirt	KubeVirt inventory source
linode	Ansible dynamic inventory plugin for Linode
netbox	NetBox inventory source
nmap	Uses nmap to find hosts to target
online	Online inventory source
openshift	OpenShift inventory source
openstack	OpenStack inventory source
scaleway	Scaleway inventory source
script	Executes an inventory script that returns JSON
toml	Uses a specific TOML file as an inventory source
tower	Ansible dynamic inventory plugin for Ansible Tower
virtualbox	virtualbox inventory source
vmware_vm_inventory	VMware Guest inventory source

Plugin d’inventaire Ansible	Description
vultr	Vultr inventory source
yaml	Uses a specific YAML file as an inventory source

1.3. Appel direct à l’inventaire

L’option `-i` ou `--inventory` suivie du chemin du fichier d’inventaire est utilisée pour préciser le chemin de l’inventaire avec les programmes `ansible`, `ansible-playbook` ou encore `ansible-inventory`.

1.4. Chemin par défaut de l’inventaire

La configuration du chemin d’un inventaire par défaut, ici démontrée avec la configuration d’une variable d’environnement `ANSIBLE_INVENTORY` :

```
echo "127.0.0.1 ansible_connection=local" > ~/ansible_hosts
export ANSIBLE_INVENTORY=~/ansible_hosts
```

On peut aussi préciser le chemin du fichier d’inventaire par défaut dans le fichier de configuration `ansible.cfg` avec l’entrée `inventory` sous la section `[defaults]` :

```
[defaults]
inventory = ~/ansible_hosts
```

1.5. Commande ansible-inventory

Le programme `ansible-inventory` liste les informations de l’inventaire en format JSON ou YAML.

Avec l’action –list, l’inventaire sort en format JSON :

```
ansible-inventory -i inventory --list
```

Mais il pourrait sortir en format YAML :

```
ansible-inventory -i inventory --list -y
```

Ou sous forme de graphe :

```
ansible-inventory -i inventory --graph
```

2. Structure d’un inventaire

2.1. Hôtes et groupes d’hôtes

L’inventaire Ansible est constitué d’hôtes (`hosts:`), de groupes qui comprennent les hôtes et des variables (`vars:`) attachées à ces hôtes et à ces groupes. un groupe peut imbriqué d’autres groupes (`children:`).

Un hôte appartient toujours au moins à deux groupes, car deux groupes d’hôtes sont toujours présents par défaut :

- `all` : qui contient **tous les hôtes**.
- `ungrouped` : qui contient tous les hôtes qui ne sont pas dans un autre groupe que `all`.

Un groupe permet de “regrouper” les hôtes sur base de :

- La fonction - Une application, une pile ou un microservice (par exemple, des serveurs de base de données, des serveurs web, etc.)
- L’emplacement - Un centre de données ou une région, pour parler au DNS local, au stockage, etc. (par exemple, à l’est, à l’ouest).
- Le moment - La phase de développement, pour éviter les tests sur les ressources de production (par exemple, prod, test).

On peut aussi placer des groupes dans d’autres groupes, soit imbriquer des groupes qui sont alors organisé selon une nomenclature parent/enfant.

2.2. Liste d’hôtes

L’inventaire le plus simple est une liste d’hôte en argument de l’option `-i`. Par exemple, pour désigner un inventaire directement comme une liste d’hôtes avec les binaires `ansible-playbook` ou `ansible` :

```
ansible -i '192.168.122.10,192.168.122.11,192.168.122.12,' -m ping all
```

ou encore par leur nom :

```
ansible -i 'node0,node1,node2,' -m ping all
```

ou encore avec une liste d’un seul hôte :

```
ansible -i '192.168.122.10,' -m ping all
```

2.3. Formats d’inventaire YAML ou INI

Dans cet exemple, on trouve cinq hôtes et deux groupes d’hôtes.

Les sections entre crochets identifient des groupes d’hôtes.

```
mail.example.com

[webservers]
foo.example.com
bar.example.com

[dbservers]
one.example.com
two.example.com
three.example.com
```

En réalité, il faudra compter sur les deux groupes par défaut `all` et `ungrouped`. Ici le même inventaire présenté en format YAML :

```
all:
  children:
    dbservers:
      hosts:
        one.example.com: {}
        three.example.com: {}
        two.example.com: {}
    ungrouped:
      hosts:
        mail.example.com: {}
    webservers:
      hosts:
        bar.example.com: {}
        foo.example.com: {}
```

On constate que le groupe `all` imbrique tous les autres. `all` est le parent de tous les autres groupes, il contient les trois autres groupes enfants `dbservers`, `ungrouped` et `webservers`.

2.4. Hôtes dans plusieurs groupes

Un hôte peut appartenir à plusieurs groupes, comme par exemple ici en format INI :

```
mail.example.com

[webservers]
foo.example.com
bar.example.com

[dbservers]
one.example.com
two.example.com
three.example.com

[east]
foo.example.com
one.example.com
two.example.com

[west]
bar.example.com
three.example.com

[prod]
foo.example.com
one.example.com
two.example.com

[test]
bar.example.com
three.example.com
```

L’inventaire sembable en format YAML :

```
all:
  children:
    dbservers:
      hosts:
        one.example.com: {}
        three.example.com: {}
        two.example.com: {}
    east:
      hosts:
        foo.example.com: {}
        one.example.com: {}
        two.example.com: {}
    prod:
      hosts:
        foo.example.com: {}
        one.example.com: {}
        two.example.com: {}
    test:
      hosts:
        bar.example.com: {}
        three.example.com: {}
    ungrouped:
      hosts:
        mail.example.com: {}
    webservers:
      hosts:
        bar.example.com: {}
        foo.example.com: {}
    west:
      hosts:
        bar.example.com: {}
        three.example.com: {}
```

2.5. Groupes imbriqués

La clé `children` indique les groupes qui s’imbriquent :

```
mail.example.com

[webservers]
foo.example.com
bar.example.com

[dbservers]
one.example.com
two.example.com
three.example.com

[east]
foo.example.com
one.example.com
two.example.com

[west]
bar.example.com
three.example.com

[prod:children]
east

[test:children]
west
```

```
all:
  children:
    dbservers:
      hosts:
        one.example.com: {}
        three.example.com: {}
        two.example.com: {}
    prod:
      children:
        east:
          hosts:
    test:
      children:
```

```
      west:
  ungrouped:
    hosts:
      mail.example.com: {}
  webservers:
    hosts:
      bar.example.com: {}
      foo.example.com: {}
```

Vous pouvez simplifier votre inventaire avec des motifs.

Ici sur une étendue :

```
[webservers]
www[01:50].example.com
```

```
...
  webservers:
    hosts:
      www[01:50].example.com:
```

Avec un incrément :

```
[webservers]
www[01:50:2].example.com
```

```
  webservers:
    hosts:
      www[01:50:2].example.com:
```

2.6. Hôte implicite localhost

Un inventaire implicite existe toujours pour l'hôte spécial `localhost`.[4] Pour l'hôte local, soit la machine qui exécute Ansible, il n'est donc pas nécessaire de créer un inventaire. Autrement dit on peut exécuter des tâches sur `localhost` sans qu'il soit défini dans l'inventaire. Ceci dit, la variable `ansible_connection` valorisée avec `local` est conseillée pour une communication interne directe. Si la configuration ou l'inventaire ne le précise pas, il faudrait le faire dans le jeu.

```
...


hosts:
  localhost:
   vars:
     ansible_connection: local
     ansible_python_interpreter: "{{ ansible_playbook_python }}"
```

3. Variables d’inventaire

L’inventaire est sans doute l’endroit idéal pour valoriser des variables propres à des hôtes ou à des groupes d’hôtes.

3.1. Variable comportementales de connexion

Outre des variables personnalisées, on y trouvera volontiers des variables comportementales spéciales concernant les paramètres de connexion en vue de gérer les cibles.[5]

- `ansible_host` et `ansible_port`
- `ansible_connection`
- `ansible_ssh_user` et `ansible_ssh_pass`
- `ansible_become` et `ansible_become_method`

3.2. Variables d’inventaire sur les hôtes

Vous pouvez associer des variables aux hôtes mais seulement sur la ligne de l’hôte en format INI :

```
web ansible_port=22 ansible_host=192.168.122.50
```

```
all:
  children:
    ungrouped:
      hosts:
        web:
          ansible_host: 192.168.122.50
```

```
    ansible_port: 22
```

La variable `ansible_host` est un alias du nom d'hôte qui permet par exemple de définir l'adresse IP de l'hôte à gérer. La variable `ansible_port` défini le port de connexion de l'hôte.

Un autre exemple dans lequel on spécifie le type de connexion (`ansible_connection`) et l'utilisateur SSH (`ansible_ssh_user`) :

```
localhost ansible_connection=local


[targets]
other1.example.com ansible_connection=ssh ansible_ssh_user=user
other2.example.com ansible_connection=ssh ansible_ssh_user=user
```

```
all:
  children:
    targets:
      hosts:
        other1.example.com:
          ansible_connection: ssh
          ansible_ssh_user: user
        other2.example.com:
          ansible_connection: ssh
          ansible_ssh_user: user
    ungrouped:
      hosts:
        localhost:
          ansible_connection: local
```

3.3. Variables d'inventaire sur les groupes

On appliquera volontiers des variables comportementales de connexion sur les groupes :

- `ansible_connection`
- `ansible_ssh_user` et `ansible_ssh_pass`
- `ansible_become` et `ansible_become_method`

Par exemple, dans ce fichier d'inventaire :

```
[webservers]
app1
app2

[dbservers]
db

[all:vars]
ansible_connection=ssh
ansible_ssh_user=root
ansible_ssh_pass=testtest
```

ou encore :

```
[debian]
app[01-02]

[rhel]
app[03-04]

[ubuntu]
app[05-06]

[webservers]
app[01-06]

[others:children]
debian
rhel

[ubuntu:vars]
ansible_ssh_user=ubuntu
ansible_ssh_pass=testtest
ansible_become=true

[others:vars]
ansible_ssh_user=root
```

```
ansible_ssh_pass=testtest

[all:vars]
ansible_connection=ssh
```

3.4. Précédence des variables

- Les “Group variables” s’appliquent pour tous les périphériques d’un groupe : `all`, le groupe parent, le groupe enfant.
- Les “Host variables” s’appliquent uniquement sur le périphériques et outrepassent les “Group variables”.

3.5. Organisation de variables d’inventaire en fichiers

Au lieu de placer les variables des hôtes et des groupes dans le fichier d’inventaire, il est possible de les encoder dans fichiers séparés prenant le nom de l’hôte ou du groupe dans les dossiers par défaut `group_vars` et `host_vars` en format YAML.[6]

```
group_vars/
  all.yml
  webservers.yml
  dbservers.yml
host_vars/
  www01.yml
  www02.yml
  db01.yml
  db02.yml
```

4. Inventaire dynamique

Ansible offre la possibilité d’utiliser un inventaire dynamique plutôt qu’une liste statique d’adresses.[7] Si le chemin de l’inventaire est un exécutable, il l’exécutera et interprétera la sortie JSON dans un format particulier.

4.1. Retour en format JSON

Cela signifie que vous pouvez exécuter le code que vous souhaitez pour retourner votre inventaire, à condition de respecter le format, comme par exemple celui-ci :

```
{
  "_meta": {
    "hostvars": {
      "web": {
        "ansible_host": "192.168.122.50",
        "ansible_port": 22
      }
    }
  },
  "all": {
    "children": [
      "ungrouped"
    ]
  },
  "ungrouped": {
    "hosts": [
      "web"
    ]
  }
}
```

Les scripts d'inventaire doivent accepter les arguments `--list` et `--host <hostname>`.[8]

4.2. Exemple en Python

On trouvera ici un exemple formel en Python d'un script d'inventaire nommé `sample-inventory.py`.

```
#!/usr/bin/env python3


'''
Example custom dynamic inventory script for Ansible, in Python.
https://github.com/geerlingguy/ansible-for-devops/blob/master/dynamic-inventory/custom/inventory.py
'''
```

```
import os
import sys
import argparse
import json

class ExampleInventory(object):

    def __init__(self):
        self.inventory = {}
        self.read_cli_args()

        # Called with `--list`.
        if self.args.list:
            self.inventory = self.example_inventory()
        # Called with `--host [hostname]`.
        elif self.args.host:
            # Not implemented, since we return _meta info `--list`.
            self.inventory = self.empty_inventory()
        # If no groups or vars are present, return empty inventory.
        else:
            self.inventory = self.empty_inventory()

        print(json.dumps(self.inventory));

    # Example inventory for testing.
    def example_inventory(self):
        return {
            "_meta": {
                "hostvars": {
                    "web": {
                        "ansible_host": "192.168.122.50",
                        "ansible_port": 22
                    }
                }
            },
            "all": {
                "children": [
                    "ungrouped"
                ]
            },
```

```
        "ungrouped": {
            "hosts": [
                "web"
            ]
        }
    }

    # Empty inventory for testing.
    def empty_inventory(self):
        return {'_meta': {'hostvars': {}}}

    # Read the command line args passed to the script.
    def read_cli_args(self):
        parser = argparse.ArgumentParser()
        parser.add_argument('--list', action = 'store_true')
        parser.add_argument('--host', action = 'store')
        self.args = parser.parse_args()

# Get the inventory.
ExampleInventory()
```

```
chmod +x sample-inventory.py
ansible-inventory -i sample-inventory.py --list
```

4.3. Exemple en Bash

On trouvera ici un même exemple formel en Bash nommé `sample-inventory.sh`.

```
#!/bin/bash
#https://stackoverflow.com/questions/56280806/ansible-dynamic-inventory-with-bash-script

if [ "$1" == "--list" ]; then
cat<<EOF
{
    "_meta": {
        "hostvars": {
            "web": {
                "ansible_host": "192.168.122.50",
```

```
            "ansible_port": 22
        }
      }
    },
    "all": {
      "children": [
        "ungrouped"
      ]
    },
    "ungrouped": {
      "hosts": [
        "web"
      ]
    }
}
EOF
elif [ "$1" == "--host" ]; then
  echo '{"_meta": {hostvars": {}}}'
else
  echo "{ }"
fi
```

```
chmod +x sample-inventory.sh
ansible-inventory -i sample-inventory.sh --list
```

4.4. Scripts d’inventaire communautaires

On trouvera dans le lien community.general/scripts/inventory/ toute une série de script d’inventaire maintenus par la communauté pour des prestataires ou des solutions comme Azure, Cobbler, Consul, Docker, Foreman, FreeIPA, GCP, Infoblox, Jail, LXC, LXD, Nagios, OpenShift, Ovirt, OpenStack, Packet, Proxmox, Scaleway, Vagrant, VirtualBox, ...

5. Dossier d’inventaire et sources multiples

5.1. Sources multiples

On peut préciser plusieurs sources d'inventaire à l'exécution d'un livre de jeu[[9] :

```
ansible-playbook get_logs.yml -i staging -i production
```

5.2. Dossier d'inventaire

Si l'emplacement de l'inventaire est un dossier, Ansible peut utiliser plusieurs sources d'inventaire en même temps. Il est donc possible de mélanger des sources d'inventaire gérées de manière dynamique et statique dans la même exécution d'ansible. Par exemple ce dossier d'inventaire nommé `inventory/` :

```
inventory/
  openstack.yml          # configure un plugin d'inventaire pour obtenir les hôtes d'un nuage Openstack
  dynamic-inventory.py   # ajoute dynamiquement des hôtes avec un script
  static-inventory       # ajoute des hôtes des groupes statiques
  group_vars/
    all.yml              # attribue des variables à tous les hôtes
```

Dans un dossier d'inventaire, les fichiers exécutables sont traités comme des sources d'inventaire dynamiques et la plupart des autres fichiers comme des sources statiques.

6. Variables magiques de l'inventaire

Ces variables sont dites "magiques"[9] car elle ne peuvent pas être définies directement par l'utilisateur. Ansible les remplacera toujours pour refléter l'état interne réel.

Les variables dynamiques d'inventaire (dépendant du contexte d'exécution) comme `hostvars`, `groups`, `group_names` et `inventory_hostname` permettent d'accéder à des variables appartenant à d'autres hôtes que ceux du jeu lui-même.

Parmi elles, nous citons celles qui dépendent de l'inventaire :

- `hostvars` : Un dictionnaire avec tous les hôtes en inventaire et les variables qui leur sont attribuées

- `groups` : Un dictionnaire avec tous les groupes en inventaire et chaque groupe a la liste des hôtes qui lui appartiennent
- `group_names` : Liste des groupes dont l'hôte actuel fait partie
- `inventory_hostname` : Le nom d'inventaire de l'hôte "actuel" qui est répété dans le jeu
- `inventory_hostname_short` : La version courte de inventory_hostname
- `inventory_dir` : Le répertoire de la source de l'inventaire dans lequel le nom d'hôte de l'inventaire a été défini pour la première fois

6.1. Variable hostvars et groups

Il ne faut pas confondre la variable `hostvars` avec le nom par défaut du dossier qui contient les variables des hôtes `host_vars/`.

La variable magique `hostvars` est dictionnaire avec tous les hôtes en inventaire et les variables qui leur sont attribuées. On peut la combiner avec l'autre variable magique `groups` qui est un dictionnaire avec tous les groupes en inventaire, chacun a la liste des hôtes qui lui appartiennent.

Grâce à ces variables, on peut donc accéder à des variables d'hôtes qui ne font pas partie d'un jeu. [10]

Avec cet inventaire par exemple :

```
[nodes]
node1 ansible_ssh_host=192.168.122.11 node_name=foo
node2 ansible_ssh_host=192.168.112.12 node_name=bar

[nodes:vars]
custom_var=asdasdasd
```

On peut accéder à la variable `custom_var` du groupe `nodes` avec cette variable en Jinja2 :

```
{{ hostvars['nodes'].custom_var }}
```

Et si on avait d'une valeur spécifique d'un hôte, par exemple la variable `node_name` du premier hôte du groupe `nodes`, on pourrait invoquer cette formule :

```
{{ hostvars[groups['nodes'][0]].node_name }}
```

Avec `groups`, une liste de tous les groupes (et hôtes) dans l'inventaire, vous pouvez énumérer tous les hôtes d'un groupe. Par exemple, on peut énumérer tous les hôtes d'un groupe :

```
{% for host in groups['app_servers'] %}
  # something that applies to all app servers.
{% endfor %}
```

6.2. Récupérer les facts d’un hôte

Si on souhaite configurer un serveur en utilisant la valeur d’un “fact” d’un autre nœud, ou la valeur d’une variable d’inventaire assignée à un autre nœud, vous pouvez utiliser les `hostvars` dans un template ou sur une ligne d’action :

```
{{ hostvars['test.example.com']['ansible_facts']['distribution'] }}
```

Vous pouvez utiliser `groups` et `hostvars` ensemble pour trouver toutes les adresses IP d’un groupe, ici dans un template Jinja2 :

```
{% for host in groups['nodes'] %}
  {{ hostvars[host]['ansible_facts']['eth0']['ipv4']['address'] }}
{% endfor %}
```

6.3. Variables de noms

Avec `group_names`, une liste (un tableau) de tous les groupes dans lesquels se trouve l’hôte actuel, on peut créer des fichiers modèles qui varient en fonction de l’appartenance au groupe (ou du rôle) de l’hôte :

```
{% if 'nodes' in group_names %}
  # some part of a configuration file that only applies to nodes group
{% endif %}
```

On peut utiliser la variable magique `inventory_hostname`, le nom de l’hôte tel que configuré dans votre inventaire, comme alternative à `ansible_hostname` lorsque la collecte de “facts” (*fact-gathering*) est désactivée. Si vous avez un FQDN long, vous pouvez utiliser la variable `inventory_hostname_short`, qui contient la partie jusqu’à la première tranche, sans le reste du domaine.

7. Ajout d’hôtes et de groupes d’inventaire par le livre de jeu

Les modules `add_host` et `group_by` permettent de créer des hôtes et des groupes d’inventaire par un livre de jeu lui-même, et de les réutiliser dans sa suite.

7.1. Module add_host

L’intérêt du module `add_host` est d’être en mesure de créer des hôtes et des groupes dans un livre de jeu courant, sans qu’ils soient définis d’avance et les utiliser comme cible dans des jeux ultérieurs du même livre de jeu. Ce sont des variables qui nourissent l’inventaire. Le module permet aussi d’attribuer des variables à ces nouveaux hôtes.

7.2. Module group_by

Le module `group_by` utilise des “facts” pour créer des groupes correspondants qui seront utilisés comme cible dans les jeux ultérieurs d’un livre de jeu. Autrement dit, `group_by` permettrait de créer dynamiquement des groupes cibles d’inventaire en fonction de “facts” communs comme la famille et la version de la distribution par exemple ou d’autres caractéristiques communes.

8. Exécution limitée

On peut limiter l’exécution des jeux sur certains hôtes seulement (option `-l`, `--limit`).

```
ansible-playbook playbook.yml --limit node0
```

1. How to build your inventory ↩
2. Excepté pour les tâches sur l’hôte local, l’hôte spécial “localhost” est une cible qu’il n’est pas nécessaire de définir dans l’inventaire. ↩
3. Inventory Plugins ↩
4. Implicit localhost ↩
5. Connecting to hosts: behavioral inventory parameters ↩
6. Best Practices, Content Organization et Working with Inventory, Splitting Out Host and Group Specific Data ↩
7. ansible.builtin.script – Executes an inventory script that returns JSON ↩

8. Working with dynamic inventory ↩
9. Magic variables ↩
10. Information about Ansible: magic variables ↩

Revision #1
Created 3 October 2021 21:07:05 by garfi
Updated 3 October 2021 21:07:49 by garfi