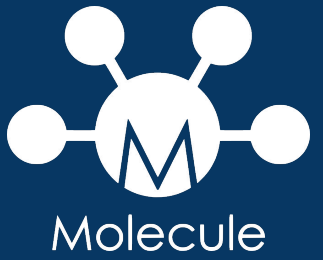


Molecule

Introduction à Molecule

Introduction à Molecule

La première partie d'un ensemble de trois presentation.

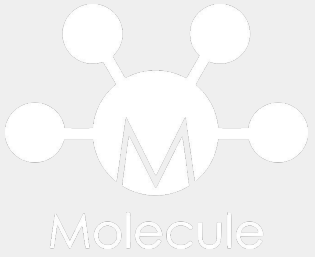
Les suivantes sont :

- **Approfondissement de Molecule**
- **Utilisations avancés de Molecule**.

Pierre BLANC
Architecte de solutions

pierre@redhat.com

Présentation



Molecule permet d'aider le développement et le test des rôles Ansible.

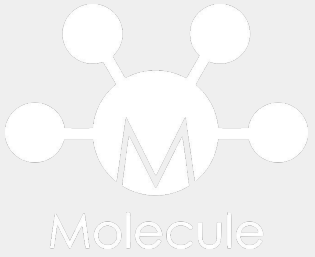
Molecule est un projet opensource.
La version 3 est sortie le 20 Février 2020

Il propose de tester les rôles et playbook Ansible :

- Sur plusieurs serveurs virtualisées ou conteneurs
- Sur différents système d'exploitation (potentiellement de version différentes)
- Pour valider l'idempotence d'un rôle.
- Via une validation syntaxique.
- Via des tests unitaire et fonctionnelles.

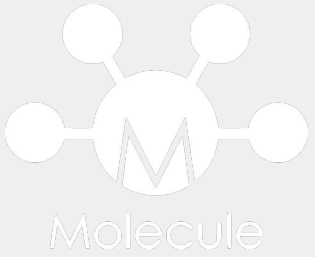
https://github.com/ansible/molecule

Concepts de Base



- Multiple drivers
- Vérificateur de syntaxe
- Tests de validation
- Scénarios
- Séquences
- Configurations

Multiple drivers



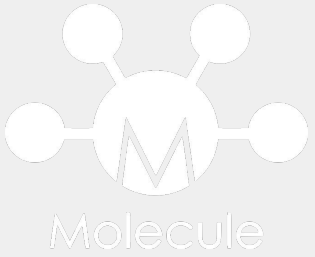
Les rôles peuvent être aussi bien testés sur des machines virtuelles que sur des conteneurs.

Prise en charge de plus d'une multitude de drivers différents dont:

- Openstack
- Azure
- Amazon
- Docker
- Podman
- LXC
- LXD
- Vagrant
- DigitalOcean
- ...

Le connecteur Vagrant permet d'étendre encore la liste des drivers disponibles.

Vérificateur de syntaxe



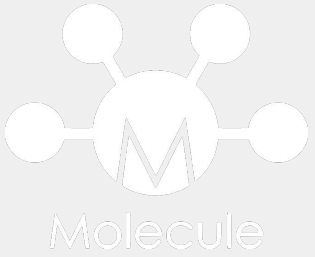
Les vérificateurs de syntaxe permettent de s'assurer que les rôles et plugins sont correctement écrits.

Ils permettent de s'assurer du respect des bonnes pratiques.

Liste non exhaustive des vérificateurs de syntaxe utilisés

- ansible-lint
- yamllint
- flake8
- ...

Scénario



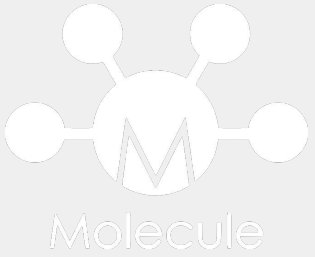
Les scénarios sont les point de départ d'un jeu de test.

Il englobe tous les fichiers dont molecule a besoin comme les fichiers de configuration, les tests, le playbook.

Le format des fichiers de configuration est en YAML

ansible-role-apache-0/ ├── defaults │ └── main.yml ├── handlers │ └── main.yml ├── meta │ └── main.yml ├── tasks │ └── main.yml └── molecule └── dev └── molecule.yml	Rôle Ansible Fichiers spécifiques à Ansible Fichiers liés à molecule

Scénario

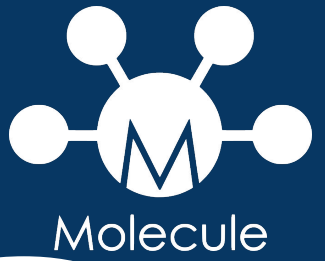


La commande **molecule list** permet de lister l'ensemble des scénarios disponible pour un rôle.

```
molecule list
--> Validating schema /home/redhat/molecule/httpd/molecule/default/molecule.yml.
Validation completed successfully.
Instance Name    Driver Name    Provisioner Name    Scenario Name    Created    Converged
---------------  -------------  ------------------  ---------------  ---------  -----------
instance         docker         ansible             default          false      false
```

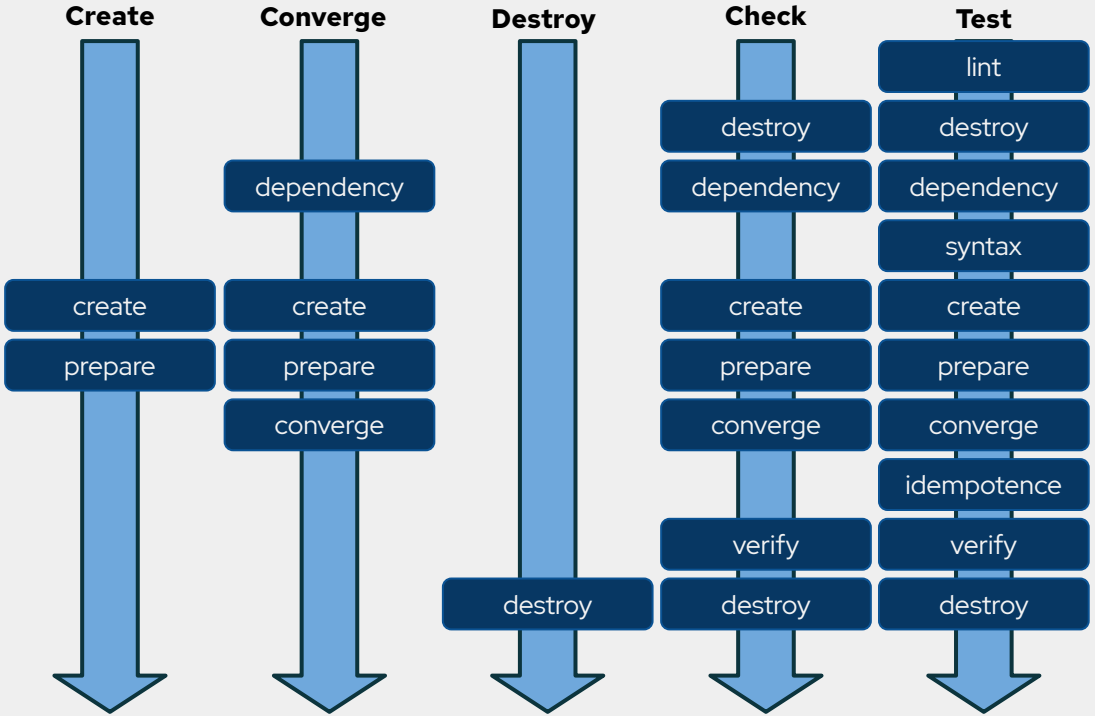
La commande **molecule init scenario** permet de créer un nouveau scénario

```
molecule init scenario -s kvm -d vagrant
--> Validating schema /home/redhat/molecule/httpd/molecule/kvm/molecule.yml.
Validation completed successfully.
Instance Name    Driver Name    Provisioner Name    Scenario Name    Created    Converged
---------------  -------------  ------------------  ---------------  ---------  -----------
instance         vagrant        ansible             kvm              false      false
```

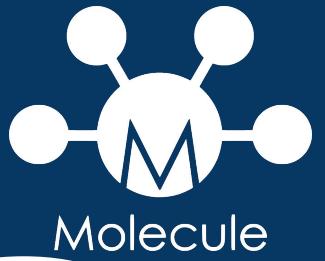


DEMO

Séquences

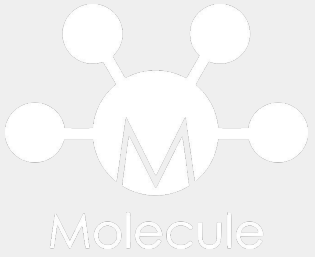


- Valide la syntaxe des fichiers
- Détruit l'environnement de test
- Installe les dependance
- Valide la syntaxe du playbook
- Créer l'environnement de test
- Installe les prérequis
- Applique le rôle ansible
- Valide que le rôle est idempotent
- Lance les tests unitaires
- Détruit l'environnement de test



DEMO

Tests de validation



Molecule intègre des outils de tests comme Testinfra, Goss, Inspect et Ansible.

Ce sont des projets opensource permettant de réaliser des tests unitaires / fonctionnels .

Un large choix de module vous permettra de tester toutes les couches d'un système.

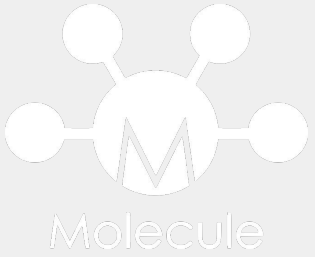
Exemple Testinfra

`def test_apache_is_installed(host):`
`  apache = host.package("httpd")`
`  assert apache.is_installed`
	Vérifie que le logiciel httpd est installé

`def test_apache_running_and_enabled(host):`
`  apache = host.service("httpd")`
`  assert apache.is_running`
`  assert apache.is_enabled`
	Vérifie que le service httpd est lancé et activé par défaut

`def test_port_80_is_listening(host):`
`  socket = host.socket("tcp://80")`
`  assert(socket.is_listening)`
	Vérifie que le port 80 est bien utilisé

Tests de validation

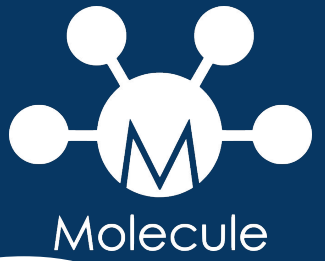


Exemple goss

```
package:
  httpd:
    installed: true

service:
  httpd:
    enabled: true
    running: true

port:
  tcp:80:
    listening: true
```

Exemple Ansible

```
- package_facts:
- service_facts:
- listen_ports_facts:

- assert:
    that:
      - "'httpd' in ansible_facts.packages"
      - "ansible_facts.services['httpd'].state == 'running'"
      - "'80' in ansible_facts.tcp_listen"
```



DEMO

Molecule

MERCI