нетология

Тестирование Roles

Алексей
Метляков

Алексей Метляков

DevOps Engineer

OpenWay

Алексей Метляков

План занятия

1. Как тестировать Role?
2. Molecule
3. Tox
4. Итоги
5. Домашнее задание

Как тестировать Role?

В данном виде тестирование Role превращается в достаточно сложную задачу, так как нужно:

- Провести проверку синтаксиса;
- Подготовить тестовое окружение;
- Провести проверку на работоспособность;
- Провести проверку на идемпотентность;
- Исправить ошибки на каждом из этих этапов и повторять весь сценарий, пока не будет получен положительный результат.

Molecule

Данный фреймворк позволяет избавить нас от рутины и заниматься только созданием и исправлением ошибок. Он умеет:

- Создавать новые **roles;**
- Создавать **scenarios** тестирования;
- Тестировать **roles** против разного окружения;
- Поддерживает docker, podman, delegated как драйверы подключений.

Официальная документация тут

Как создать Role?

- Создать стандартную структуру директорий и файлов при помощи **molecule;**
- Создать необходимые **tasks**, **handlers;**
- Определить все необходимые переменные в **defaults** и **vars;**
- Создать готовый тестовый **playbook** в **tests**, заполнить файлы **molecule** для проведения тестирования;
- Заполнить **meta** всю информацию о роли, наиболее полно описать её в **README.md.**

Структура директорий

После инициализации новой роли мы получаем следующие директории и файлы:

- стандартный набор директорий и файлов для **role;**
- molecule – набор **scenarios** для тестирования.

Внутри любого **scenario** находятся следующие файлы:

- **molecule.yml** – основной файл для **molecule;**
- **converge.yml** – **playbook**, который **molecule** будет использовать для запуска тестов;
- **verify.yml** – дополнительные тесты после исполнения **role**.

Структура molecule.yml

Внутри файла находятся следующие директивы:

- **dependency** – перечисление зависимостей роли;
- **driver** – указание параметров выбранного **driver;**
- **platform** – перечисление хостов для выбранного **driver;**
- **provisioner** – указание поставщика для **molecule;**
- **verifier** – выбор **framework** для проведения проверок.

Туда же можно добавить:

- **lint** – конфигурирование **linter** для тестирования;
- **scenario** – перечисление сценариев тестирования.

Как правильно тестировать Role?

В данном виде тестирование **role** упрощается, так как нужно:

- Подготовить тестовое окружение;
- Запустить сценарий проверки через **molecule;**
- Исправить ошибки на каждом из этих этапов и повторять весь сценарий, пока не будет получен положительный результат.

Как правильно тестировать Role?

Тестирование условно можно разделить на три вида:

- Использование полного сценария тестирования;
- Использование собственных сценариев тестирования;
- Использование отдельных частей сценария самостоятельно.

Как правильно тестировать Role?

molecule test запускает полный сценарий тестирования role.

Полный сценарий включает в себя:

- **lint** – прогон линтеров;
- **destroy** – удаление старых инстансов с прошлого запуска;
- **dependency** – производит установку ansible-зависимостей, если есть;
- **syntax** – проверка синтаксиса с помощью `ansible-playbook --syntax-check`;
- **create** – создание инстансов для тестирования;
- **prepare** – подготовка инстансов, если это необходимо.

Как правильно тестировать Role?

molecule test запускает полный сценарий тестирования role.

Полный сценарий включает в себя:

- **converge** – запуск тестируемого плейбука;
- **idempotence** – проверка на идемпотентность при помощи повторного запуска;
- **side_effects** – действия, которые не относятся к role, но необходимые для тестирования;
- **verify** – запуск тестов с помощью указанного фреймворка тестирования;
- **cleanup** – очистка внешней инфраструктуры от результатов тестирования;
- **destroy** – уничтожение инстансов для тестирования.

Как правильно тестировать Role?

- Каждую из указанных частей сценария можно вызвать отдельно, но нужно держать в уме, что у каждой из них могут быть зависимости.
- Для того чтобы понимать, что каждая из **tasks** будет запускать, в случае отдельного вызова – необходимо пользоваться конструкцией **molecule matrix <task_name>.**
- Перед тем, как уничтожать инстансы, к ним можно подключиться и в ручном режиме проверить все изменения в системе.
- Оставить возможность подключения можно и с полным сценарием тестирования, воспользовавшись параметром **--destroy=never.**

Как правильно тестировать Role?

Очерёдность сценариев можно переопределить через директиву **scenario.** Формат записи в **molecule.yml** будет выглядеть так:

```
---
scenario:
  <task>_sequence:
    - list
    - of
    - tasks
...
---
#Example of redefined of test scenario
scenario:
  test_sequence:
    - create
    - converge
    - idempotence
    - destroy
...
```

Список основных команд molecule

- molecule init role --driver-name <driver> <rolename>
- molecule init scenario --driver-name <driver> <scenarioname>
- molecule test
- molecule test --destroy=never
- molecule matrix <taskname>
- molecule matrix -s <scenarioname> <taskname>
- molecule <taskname>

Tox

Данный фреймворк позволяет проводить тестирование любого python кода, грубо говоря, tox – менеджер виртуальных окружений:

- Выбирать, какой версии **python** использовать (версии должны быть установлены в системе);
- **Выбирать,** какие **дополнительные** модули должны быть установлены;
- **Тестировать** разные версии модулей друг против друга (создавать матрицы тестирования).

Официальная документация тут

Установка Tox

Так как он является стандартной библиотекой **python** кода (как и **ansible**), то его установка происходит достаточно просто:

- pip3 install tox
- pip install tox

Настройка Tox

В нашем случае, нам необходимо два файла для использования tox:

- **tox.ini** – основной файл настройки, содержит в себе перечисление **python** и возможных **модулей** плюс описание их перечисления друг против друга, а также указание **команды**, которую необходимо выполнить для проведения тестирования
- **test-requirements.txt** – перечисление дополнительных **модулей**, которые **не должны** иметь итерирования против разных версий **python**.

Запуск Tox

- **tox** – запуск на всех возможных окружениях,
- **tox -l** – показ всех возможных окружений,
- **tox -r** – принудительное пересоздание окружений,
- **tox -e <env_names>** – запуск на указанных окружениях

Итоги

Итоги

Сегодня мы узнали, что:

- существует удобный **framework** для тестирования – **molecule;**
- **molecule** позволяет проверить работоспособность роли против разного окружения на **managed node;**
- второй **framework** для тестирования – **tox;**
- **tox** позволяет проверить работоспособность роли против разного окружения на **control node.**

Домашнее задание

Давайте посмотрим ваше домашнее задание.

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

нетология

Задавайте вопросы и пишите отзыв о лекции!

Алексей Метляков

Алексей Метляков