

Работа с Playbook



Алексей
Метляков



Алексей Метляков

DevOps Engineer

OpenWay



Алексей Метляков

План занятия

1. [Что такое Playbook?](#)
2. [Play](#)
3. [Task](#)
4. [Handler](#)
5. [Tag](#)
6. [Role](#)
7. [Как запустить Playbook?](#)
8. [Как тестировать Playbook?](#)
9. [Итоги](#)
10. [Домашнее задание](#)

Что такое Playbook?

Ansible Playbook - набор plays, содержащих в себе roles и\или tasks, которые выполняются на указанных в inventory хостах с определёнными параметрами для каждого из них или для их групп.

Иными словами, **Playbook** описывает:

- Что делать?
- Какой результат ожидается?
- Какими средствами результат достигается?
- На каких хостах?
- С какими параметрами?

Playbook выполняется последовательно от верхнего **play** к нижнему

Play

Play нужны для **перечисления** действий, которые необходимо воспроизвести на хосте или на указанной группе хостов.

Play состоит из:

- **name** - имя **Play**
- **hosts** - перечисление хостов
- **pre_tasks** - необязательный параметр, **tasks** которые нужно выполнить в первую очередь, до **roles** и **tasks**, могут обращаться к **handlers**
- **tasks** - перечисление действий, которые нужно сделать на хостах, могут обращаться к **handlers**, не рекомендуется указывать, если есть **roles**
- **post_tasks** - необязательный параметр, перечисление **tasks**, которые необходимо выполнить после **tasks** и **roles**

Play

Также, **Play** может содержать:

- **roles** - перечисление ролей, которые необходимо запустить на хостах, могут обращаться к **handlers**, не рекомендуется указывать, если есть **tasks**
- **handlers** - обработчики событий, запускаются, если какая-то **task** или **role** обратилась к **handler**, могут быть сгруппированы через **listen**
- **tags** - позволяет группировать **roles**, **tasks** и вызывать их запуск отдельно от остальных сущностей

Play

И ещё немного зарезервированных параметров:

- **any_errors_fatal** - любая **tasks**, выполняемая на любом хосте может вызвать fatal и завершить выполнение **play** на всех хостах
- **become** - позволяет повысить привилегии для выполняемых **tasks**
- **become_user** - позволяет выбрать пользователя под которым будут повышаться привилегии
- **check_mode** - булевая переменная, помогающая определить, происходит ли запуск в **check** моде или нет
- **collections** - позволяет указать имена коллекций
- **debugger** - позволяет запустить **debugger**
- **diff** - переключатель вывода информации при использовании флага **diff**
- **force_handlers** - позволяет принудительно оповестить **handlers** на запуск

Play

И ещё немного зарезервированных параметров:

- **gather_facts** - булевая переменная для контроля запуска сбора фактов с хостов
- **ignore_errors** - позволяет игнорировать ошибки при выполнении **tasks** и продолжить выполнение **play**
- **ignore_unreachable** - позволяет игнорировать ошибки недоступности хоста и продолжать выполнение **play**
- **max_fail_percentage** - позволяет указать процент ошибок среды выполненных **tasks**, в рамках текущего **batch**, при котором можно продолжить **play**
- **order** - позволяет указать порядок сортировки хостов из **inventory**
- **run_once** - запустить данный **play** только на первом хосте из **inventory** в рамках одного **batch**
- **serial** - позволяет определить количество хостов, на которых запускается данный **play** в рамках одного **batch**

Play

Чаще всего, **play** будет выглядеть следующим образом:

```
---
- name: Try run Vector # Произвольное название play
  hosts: all # Перечисление хостов
  tasks: # Объявление списка tasks
    - name: Get Vector version # Произвольное имя для task
      ansible.builtin.command: vector --version # Что и как необходимо сделать
      register: is_installed # Запись результата в переменную is_installed
    - name: Get RPM # Произвольное имя для второй task
      ansible.builtin.get_url: # Объявление использования module get_url, ниже указание его
        url: "https://package.timber.io/vector/{{ vector_version }}/vector.rpm"
        dest: "{{ ansible_user_dir }}/vector.rpm"
        mode: 0755
      when: # Условия при которых task будет выполняться
        - is_installed is failed
        - ansible_distribution == "CentOS"
```

Play

При этом, для жизнеобеспечения **play** достаточно указать **hosts** и хотя бы один **task**:

```
---
- hosts: all # Перечисление хостов
  tasks: # Объявление списка tasks
    - name: Get Vector version # Произвольное имя для task
      ansible.builtin.command: vector --version # Что и как необходимо сделать
```

Task

Tasks нужны для **указания** действий, которые необходимо воспроизвести на хосте или на указанной в **play** группе хостов.

- **Task** - одно атомарное **действие**
- Можно сохранить результат выполнения в **переменную**
- Директива **when** помогает указать при каких условиях необходимо выполнить **task**
- Директива **notify** позволяет обратиться к **handlers**, чтобы он был исполнен в конце выполнения всех **tasks**

```
tasks: # Объявление списка tasks
- name: Get Vector version # Произвольное имя для task
  ansible.builtin.command: vector --version # Что и как необходимо сделать
  register: is_installed # Запись результата в переменную is_installed
  notify:
    - Restart Vector # Вызов handler Restart Vector
- name: Get RPM # Произвольное имя для второй task
  ansible.builtin.get_url: # Объявление использования module get_url, ниже указание его
    url: "https://package.timber.io/vector/{{ vector_version }}/vector.rpm"
  when: # Условия при которых task будет выполняться
    - is_installed is failed
```

Task

Tasks можно принудительно воспроизвести на указанном хосте:

- Для того, чтобы определить на каком хосте необходимо выполнить task - нужно использовать **delegate_to**
- Если действие необходимо делать на localhost, можно использовать инструкцию **local_action**

```
tasks: # Объявление списка tasks
- name: Take out of balance # Произвольное имя для task
  ansible.builtin.command: "/usr/bin/pool/take_out {{ inventory_hostname }}" # Что и как
  # необходимо сделать
  delegate_to: 127.0.0.1
- name: Install Latest NGINX # Произвольное имя для второй task
  ansible.builtin.yum:
    name: nginx
    state: latest
```

```
tasks: # Объявление списка tasks
- name: Take out of balance # Произвольное имя для task
  local_action:
    ansible.builtin.command:
      cmd: "/usr/bin/pool/take_out {{ inventory_hostname }}" # Что и как необходимо сделать
- name: Install Latest NGINX # Произвольное имя для второй task
  ansible.builtin.yum:
    name: nginx
    state: latest
```

Task

Tasks можно и нужно разделять на **pre_tasks**, **tasks** и **post_tasks**:

- Разделение группируется по вашей собственной внутренней логике
- Если **task** внутри этих групп вызвала **handler**, то он выполнится после того, как закончит исполнение последней **task** из текущей группы
- Внутри набора **tasks** их можно также группировать при помощи **group**

```
tasks: # Объявление списка tasks
- name: Get Vector version # Произвольное имя для task
  ansible.builtin.command: vector --version # Что и как необходимо сделать
  register: is_installed # Запись результата в переменную is_installed
  notify:
    - Restart Vector # Вызов handler Restart Vector
- name: Get RPM # Произвольное имя для второй task
  ansible.builtin.get_url: # Объявление использования module get_url, ниже указание его
    url: "https://package.timber.io/vector/{{ vector_version }}/vector.rpm"
  when: # Условия при которых task будет выполняться
    - is_installed is failed
```

Handler

Handlers используются для проведения одного действия в рамках одного **play**, например, для рестарта сервиса, после обновления конфигурации

- **Указывается** в директиве **Play**
- На **handler** могут ссылаться **task**, **role**, **pre_task**, **post_task**
- Вне зависимости от того, сколько раз **handler** был вызван - он исполнится один раз
- Если **handler** вызван в **role** - он исполнится после всех **roles**
- Если **handler** вызван в **tasks** любого вида - он исполнится в конце **tasks**, в рамках которого был вызван
- **Handler**, который определён в рамках одного **playbook** может быть вызван в любом **play**

Handler

Пример синтаксиса **Handlers**:

```
handlers: # Объявление списка handlers
- name: restart-vector # Произвольное имя для handler
  ansible.builtin.service: # Вызов module, обрабатывающего
операции с сервисами
    name: vector # Имя сервиса
    state: restarted # Ожидаемый результат работы модуля
    listen: "restart monitoring" # Группировка handlers для
возможности вызова группы
- name: restart-memcached
  ansible.builtin.service:
    name: memcached
    state: restarted
    listen: "restart monitoring"
```

Role

В рамках использования role в playbook нам достаточно знать несколько пунктов:

- **Role** можно скачивать через **ansible-galaxy**
- Какие **role** скачивать - лучше указать в **requirements.yml**
- Чтобы использовать **role** в **playbook** - необходимо использовать следующий синтаксис

```
roles: # Объявление списка roles
  - vector
  - java
```

Tag

Tag позволяет пометить какую-либо сущность **ansible** для отдельного исполнения. Их можно выставлять для:

- Отдельной task
- Группы tasks
- Include
- Play
- Role
- Import

Tag

Синтаксис **Tag** достаточно прост:

```
tasks: # Объявление списка tasks
- name: Get Vector version # Произвольное имя для task
  ansible.builtin.command: vector --version # Что и как необходимо сделать
  register: is_installed # Запись результата в переменную is_installed
  notify:
    - Restart Vector # Вызов handler Restart Vector
  tags:
    - vector
    - info
- name: Get RPM # Произвольное имя для второй task
  ansible.builtin.get_url: # Объявление использования module get_url, ниже указание его
    url: "https://package.timber.io/vector/{{ vector_version }}/vector.rpm"
  when: # Условия при которых task будет выполняться
    - is_installed is failed
  tags:
    - vector
    - install
```

Tag

Существует два выделенных **tags**:

- **always** - выполняется всегда, если явно не указано пропустить
- **never** - не выполняется никогда, если явно не указано запустить

Tag

Список ключей для использования **tags**:

- `--tags all`
- `--tags tagged`
- `--tags untagged`
- `--tags [tag1, tag2]`
- `--skip-tags [tag1, tag2]`
- `--list-tags`
- `--list-tasks` (with `--tags` or `--skip-tags`)



Практическое применение

Как запустить Playbook?

- Первый запуск стоит осуществлять или на тестовом окружении или с флагом: `ansible-playbook -i inventory/<inv_file>.yaml <playbook_name>.yaml --check`
- Если были найдены ошибки: `ansible-playbook -i inventory/<inv_file>.yaml <playbook_name>.yaml --start-at-task <task_name>`
- Для запуска исполнения в полуинтерактивном виде: `ansible-playbook -i inventory/<inv_file>.yaml <playbook_name>.yaml --step`
- Полноценный запуск **playbook** в целевом виде должен выглядеть: `ansible-playbook -i inventory/<inv_file>.yaml <playbook_name>.yaml`

Как тестировать Playbook?

- В первую очередь, нужно использовать **debugger**
- Необходимо использовать возможности **check**
- Организовать запуск на тестовом окружении
- Нужно не забывать про идемпотентность и использовать флаг:
`ansible-playbook -i inventory/<inv_file>.yaml
<playbook_name>.yaml --diff`
- Тестировать **playbook** против разного окружения на **control node**
- Тестировать **playbook** против разного окружения на **managed node**
- Активно использовать флаг **-vvv**

Как тестировать Playbook?

Чтобы использовать debugger, об этом необходимо объявить:

- через ключевое слово **debugger**
- через переменную окружения (ANSIBLE_STRATEGY)
- указать **debug** как стратегию (deprecated)



Итоги

Итоги

- **Playbook** содержит информацию о том **что** и **где** необходимо сделать
- То, **что** необходимо сделать описывается в блоке **play**
- **Где** необходимо выполнять **play** написано в **inventory**
- **Play** состоит из перечислений **task** и **role**
- **Task** - атомарное действие над **host** из **inventory**
- **Task** и их группы можно описывать при помощи **tag**
- **Tag** можно использовать для ограничения **task** на исполнение
- **Task** может оповестить **handler** на исполнение
- Вызванный несколько раз **handler** исполнится один раз
- Переменные **playbook** лучше всего хранить в **group vars**
- **ansible-playbook** - запуск полноценного **playbook**

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**.

**Задавайте вопросы и
пишите отзыв о лекции!**

Алексей Метляков