

Модуль «Автоматизация администрирования инфраструктуры» **Ansible. Часть 2**



Алексей
Хмелевских



Алексей Хмелевских

DevOps Group Manager

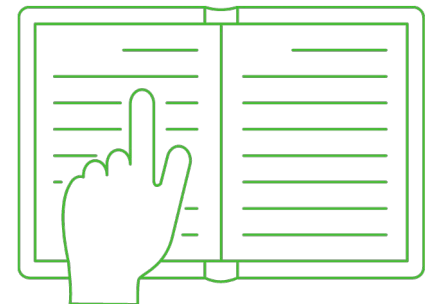
Netcracker

Предисловие

На этом занятии мы узнаем:

- как устроен синтаксис YAML
- как писать плейбуки Ansible
- что такое роли и Ansible Galaxy

После занятия научитесь писать плейбуки и работать с ролями



План занятия

1. [Введение](#)
2. [YAML. Playbook. Role](#)
3. [Практика. Пишем плейбуки](#)
4. [Ansible roles](#)
5. [Практика. Работа с ролями Ansible](#)
6. [Ansible. Расширенные возможности](#)
7. [Практика. Переменные и теги в Ansible](#)
8. [Итоги](#)
9. [Домашнее задание](#)



Введение



Основные термины Ansible

Узел управления — устройство с установленным и настроенным Ansible. Ноутбук или специальный узел в сети или подсети, выделенный для задач управления

Управляемые узлы — узлы, конфигурация которых выполняется

Файлы инвентаризации (inventory) — файл или файлы, в которых перечислены управляемые узлы: *.ini, *.yaml или динамический

Модули (modules) отвечают за действия, которые выполняет Ansible, другими словами, инструментарий Ansible



Основные термины Ansible

Задачи (tasks) — отдельный элемент работы, который нужно выполнить. Могут выполняться самостоятельно или в составе плейбука

Плейбук (playbook) — состоит из списка задач и других директив, указывающих на то, какие действия и где будут производиться

Обработчики (handlers) — элемент, который служит для экономии кода и способен перезапускать службу при его вызове

Роли (roles) — набор плейбуков и других файлов, которые предназначены для выполнения какой-либо конечной задачи. Упрощают, сокращают код и делают его переносимым



YAML. Playbook

YAML

Yet Another Markup Language — это язык для хранения информации в формате, понятном человеку

```
yml = yaml
```

YAML	JSON	XML
Читать легко	Читать легко	Читать сложно
Синтаксис не очень строгий	Строгий синтаксис	Строгий синтаксис
Есть комментарии	Нет комментариев	Есть комментарии

YAML syntax

В файлах YAML для разделения структур данных используются пробелы. Не используйте tab, только пробелы, IDE облегчит задачу

Элементы данных одного уровня должны иметь одинаковое количество отступов. Подчинённые элементы должны выделяться бóльшим количеством отступов, чем родительские

Пример:

```
level1:
  level2:
    Level3:
      item1: "one"
      item2: "two"
```

Playbook

Playbook — это основной рабочий скрипт. Он позволяет объединять задачи (task), роли в одном файле и передавать их для выполнения на определённые группы хостов

Пример запуска плейбука:

```
ansible-playbook myscript.yaml
```

Пример проверки синтаксиса плейбука:

```
ansible-playbook myscript.yaml --syntax-check
```

Зачем нужны плейбуки

- Ad-hoc команды удобны для запуска одной или двух команд
- Ad-hoc команды удобны для тестирования
- С их помощью можно запускать множество задач для множества хостов
- С их помощью удобно управлять запусками и сокращать код

Пример плейбук

В примере плейбук состоит из двух разделов, для каждого отдельно выбираются целевые хосты и запускаются роли или задачи:

```
---
- name: Playbook 1 name
  hosts: group1
  roles:
    - role1

- name: Playbook 2 name
  hosts: group2,group3
  tasks:
    - name: task 1 in playbook 2
      debug:
        msg: "Hello World"
    - name: task 2 in playbook 2
      service:
        name: sshd
        state: restarted

...
```

Vagrant provision

При создании ВМ с помощью Vagrant может понадобится производить донастройку ВМ. Vagrant позволяет делать это с помощью provision. Можно использовать shell или ansible

Пример:

```
Vagrant.configure("2") do |config|  
  
  config.vm.provision "ansible" do |ansible|  
    ansible.playbook = "playbook.yml"  
  end  
  
end
```



Практика. Пишем плейбуки



Ansible roles



Role

Role — это набор файлов, задач, шаблонов, переменных и обработчиков, которые объединены вместе в логическую структуру и служат определённой цели

Например, настройка сервера времени. Роли можно подключать в плейбуки. Роли удобно передавать коллегам, как целостную сущность

Для интерактивной работы с ролями используется команда `ansible-galaxy`

Преимущества использования ролей

- Можно использовать для нескольких проектов
- Удобно передавать
- Удобно отправить на внешние ресурсы
- Удобно управлять зависимостями

Структура ролей

- **defaults** — содержит переменные по умолчанию для роли, которые должны быть легко перезаписаны
- **vars** — содержит стандартные переменные для роли, которые не должны быть перезаписаны в вашей книге
- **tasks** — содержит набор задач, которые должна выполнять роль
- **handlers** — содержит набор обработчиков, которые будут использоваться в роли
- **templates** — содержит шаблоны Jinja2, которые будут использоваться в роли

Структура ролей

- **files** — содержит статические файлы, необходимые из ролевых задач
- **tests** — может содержать дополнительный файл инвентаря, а также `playbook test.yml`, который можно использовать для тестирования роли
- **meta** — содержит метаданные роли — информацию об авторе, лицензию, зависимости и т. д.

Расположение ролей на файловой системе

Расположение файлов роли на файловой системе имеет значение:

- **./roles** — самый высокий приоритет
- **~/.ansible/roles**
- **/etc/ansible/roles**
- **/usr/share/ansible/roles** — самый низкий приоритет

Запуск ролей

Чтобы запустить роль, необходимо написать плейбук, и в нём вызвать роль. При необходимости можно переопределить переменные в момент вызова роли

```
---  
- name: roles example  
  hosts: servers  
  roles:  
    - role: role1  
    - role: role2  
      var1: one  
      var2: two
```

Galaxy

Ansible-galaxy — это [сайт](#), который является публичным репозиторием для ansible-ролей, написанных сообществом. Т. е. на нём можно искать, использовать и выкладывать роли


Примеры:

```
ansible-galaxy role list
ansible-galaxy role init myrole
ansible-galaxy role search nginx
ansible-galaxy role install nginx
ansible-galaxy role remove nginx
```



Практика.

Работа с ролями Ansible



Ansible. Расширенные ВОЗМОЖНОСТИ

Tags

Теги служат, чтобы пометать определённые задачи (task) внутри плейбука или роли и вызывать либо наоборот пропускать только их, не заставляя выполняться весь плейбук

```
ansible-playbook -i hosts.ini --tags prod playbook.yml
```

```
ansible-playbook -i hosts.ini --skip-tags prod playbook.yml
```

Handlers

Handlers — это специальные задачи. Они вызываются из других задач ключевым словом `notify`. Служат для сокращения кода

```
---
- name: handlers example
  hosts: all
  become: yes
  tasks:
    - name: Change ssh config
      lineinfile:
        path: /etc/ssh/sshd_config
        regexp: '^PasswordAuthentication'
        line: PasswordAuthentication yes
      notify:
        - Restart sshd
  handlers:
    - name: Restart sshd
      service:
        name: sshd
        state: restarted
```

Условие when

За счёт конструкции when можно выполнять задачу из плейбука только при выполнении какого-либо условия

```
tasks:
  - name: Configure SELinux to start mysql on any port
    ansible.posix.seboolean:
      name: mysql_connect_any
      state: true
      persistent: yes
    when: ansible_selinux.status == "enabled"
```

Jinja2

С помощью механизма шаблонов Jinja2 можно динамически создавать, например, конфигурационные файлы. Для работы с шаблонами Jinja2 используется модуль `template`. Пример использования:

```
cat index.j2
```

```
A message from {{ inventory_hostname }}
```

```
- name: Create index.html using Jinja2
```

```
  template:
```

```
    src: index.j2
```

```
    dest: /var/www/html/index.html
```

Пример задания тегов

Пример task с использованием тегов:

```
- name: tag usage example  
  shell: /bin/echo "Only running because of specified tag"  
  tags: mytag
```



Vars

В Ansible существует 22 способа задавать переменные. С их помощью можно реализовывать логику работы скрипта

В [документации](#) описано, какой приоритет имеет тот или иной способ задания переменных

Пример задания переменных

Переменная задаётся в процессе запуска плейбука из командной строки:

```
ansible-playbook example.yml --extra-vars "foo=bar"
```

Переменная задаётся в начале плейбука:

```
---  
- hosts: example  
  vars:  
    foo: bar  
  tasks:  
    debug:
```

Переменная задаётся в отдельном файле, на который ссылается плейбук:

```
---  
- hosts: example  
  vars_files:  
    - vars.yml  
  tasks:  
    debug
```


Special variables

В Ansible есть встроенные [переменные](#). Их имена нельзя переназначать, но использование встроенных переменных может быть очень полезно

Пример:

```
ansible_play_hosts_all
```

 – вернёт список хостов из inventory файла

Ansible-vault

Чтобы не хранить пароли в открытом виде, в Ansible существует vault — шифрованное хранилище паролей

Ansible-vault — утилита, которая позволяет работать с таким хранилищем

Пример команд для работы с ansible-vault:

```
ansible-vault create file2.yml  
ansible-vault decrypt file2.yml  
ansible-vault encrypt file2.yml  
ansible-vault rekey file2.yml
```

Ansible Tower



Ansible Tower — графический веб-интерфейс для управления и мониторинга работы Ansible

Позволяет:

- пользователям работать со скриптами без необходимости настраивать окружение и разбираться с ними
- управлять доступом и создавать группы для определённых категорий пользователей — есть интеграция с LDAP
- настраивать разного рода автоматизации при помощи REST API

Открытая версия (upstream) — **AWS project**

Molecule

Molecule — это фреймворк для тестирования Ansible ролей. В конфигурационных файлах Molecule описывается инфраструктура и тесты, которыми покрывается роль

Что позволяет сделать:

- проверять синтаксис
- поднимать тестовую инфраструктуру
- писать тесты вручную
- скачивать роли из Galaxy



Практика.

Переменные и теги в Ansible



Итоги

Итоги

Сегодня мы:

- познакомились с синтаксисом YAML
- научились писать и запускать плейбуки
- научились работать с ролями в Ansible





Домашнее задание

Домашнее задание

Ваше домашнее задание можно посмотреть [по ссылке](#)

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

⌘ нетология

**Буду рад вашим вопросам и
отзывам о лекции!**

Алексей Хмелевских