



ОНЛАЙН-ОБРАЗОВАНИЕ



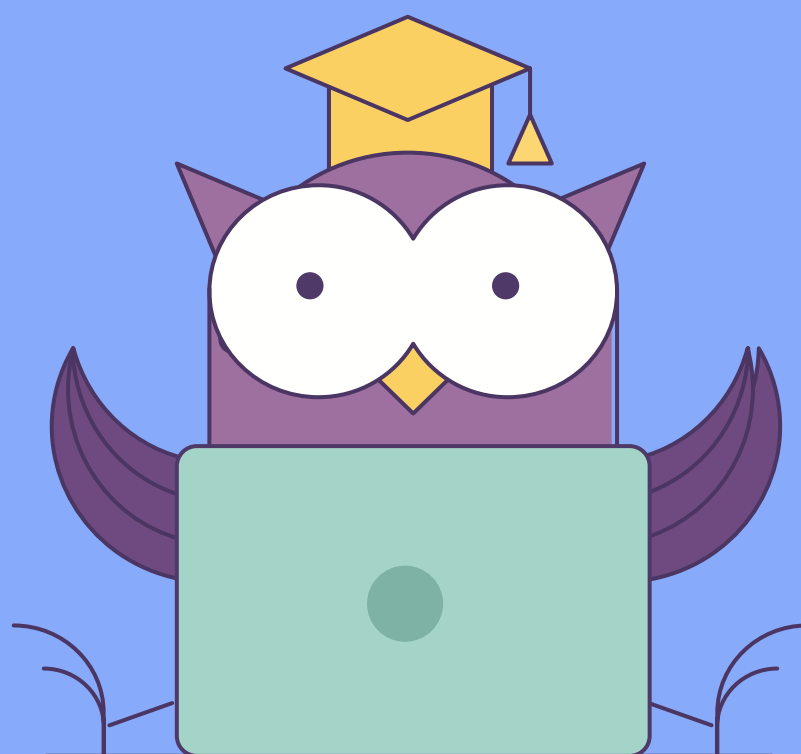
# Управление конфигурациями. Ansible

Курс «Администратор Linux»

Занятие № 9



# Меня хорошо слышно && видно?



Напишите в чат, если есть проблемы!

Ставьте ☐ + если все хорошо  
Ставьте ☐ - если есть проблемы



Управление  
конфигурациями



Системы управления  
конфигурациями



Ansible

**Управление конфигурацией** подразумевает под собой процесс установки и поддержки консистентности продукта, функциональности на всем его жизненном цикле

- Повторное использование кода
- Версионирование. VCS
- Совместная работа
- Самодокументирование

- Ansible
- Chef
- Puppet
- SaltStack



**puppet**  
labs®



**CHEF**™

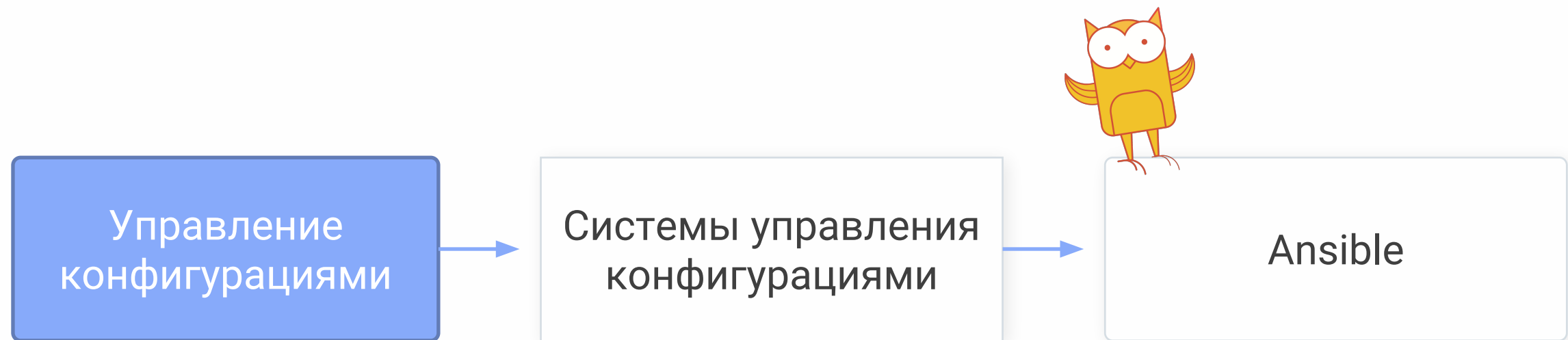


**SALTSTACK**



**ANSIBLE**

|                | Chef          | Puppet        | Ansible     | SaltStack     |
|----------------|---------------|---------------|-------------|---------------|
| Code           | Open source   | Open source   | Open source | Open source   |
| Cloud          | All           | All           | All         | All           |
| Type           | Config Mgmt   | Config Mgmt   | Config Mgmt | Config Mgmt   |
| Infrastructure | Mutable       | Mutable       | Mutable     | Mutable       |
| Language       | Procedural    | Declarative   | Procedural  | Declarative   |
| Architecture   | Client/Server | Client/Server | Client-Only | Client/Server |

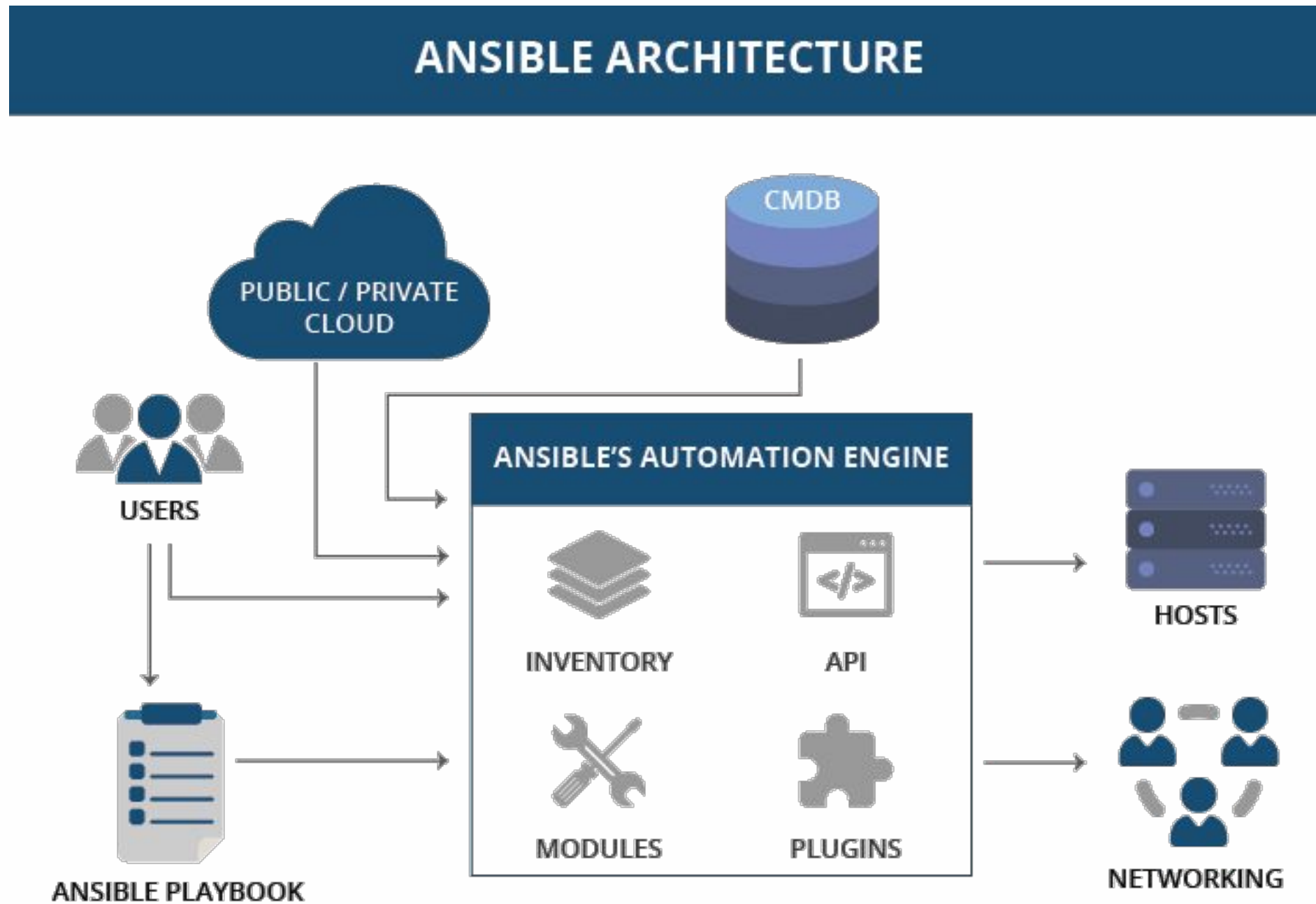






ANSIBLE

- Ansible 2.7
- Низкий порог вхождения/прост в установке
- Отличная [документация](#)
- Отсутствие агента/минимальные требования к хостам
- Идемпотентность
- Готовые модули >1000
- Готовые роли. [Ansible Galaxy](#)
- Язык YAML
- Поддержка [Windows](#). Какая никакая



Что храним и версионируем:

- Файл конфигурации для управляющего хоста
- Inventory файлы в случае статичного inventory
- Описание окружения - сами компоненты проекта, yml файлы, шаблоны
- Вспомогательные файлы
- Документацию о проекте
- Ваши тесты (molecule, infra, ...)

# Ansible. Пример репозитория

```
[root@ansible ~]$ tree -L 3
```

```
├── ansible.cfg
├── dev
│   └── hosts
├── group_vars
│   └── all.yml
├── host_vars
│   └── host1.yml
├── production
├── provision
│   └── playbook.yml
├── README.md
├── roles
│   └── fail2ban
│       ├── defaults
│       ├── files
│       ├── handlers
│       ├── meta
│       ├── tasks
│       ├── templates
│       └── vars
├── site.yml
├── staging
│   └── hosts
└── Vagrantfile
```

Конфиг файл

Групповые переменные

Переменные хостов

Роли

Инвентори для тестирования

- Файл в **ini** формате в котором хранятся predetermined параметры, например:
  - Inventory
  - Способ подключения
  - Другие параметры по умолчанию
- Пример конфигурации от [разработчиков](#). Полный листинг всех доступных опций можно посмотреть [тут](#). Начиная с версии 2.4 для получения доступных опций и просмотра текущих значений можно использовать утилиту [ansible-config](#)

- Группировка и разделение хостов
- Вложенные группы
- Inventory файлов может быть несколько
- [Динамический Inventory](#)
- Позволяет переопределить параметры указанные в ansible.cfg

# Ansible. Inventory (ini формат)

[app] ← Название группы  
app01.mydomain.com

[db]  
mysql\_master.mydomain.com  
mysql\_slave.mydomain.com ← Хосты входящие в группу

[frontend]  
nginx[1..4].mydomain.com http\_port=80 ← Переменные хоста

[us\_region:children] ← Группа групп  
app  
db

[db:vars] ← Переменные группы  
postgresql\_version=9.6



# Ansible. Inventory (YAML)

```
app:
  hosts:
    app01.mydomain.com
  vars:
    os_release: redhat
  children:
    frontend:
      hosts:
        nginx01.mydomain.com
        nginx02.mydomain.com
```

← Название группы

← Хост в группе

← Переменная хоста

← Подгруппа



Команды для просмотра инвентори:

```
[root@packages ~]# ansible-inventory --list
```

```
{
  "_meta": {
    "hostvars": {
      "haproxy": {
        "ansible_host": "35.193.16.112"
      },
      "nginx01": {
        "ansible_host": "35.188.154.136",
        "nginx_repo": "epel"
      }
    }
  }
}
```

```
[root@packages ~]# ansible-inventory --graph
```

```
@all:
  |--@app:
  | |--@balancer:
  | | |--haproxy
  | |--@web:
  | | |--nginx01
  | | |--nginx02
  |--@ungrouped:
```

- Ad-hoc - они же однострочники.

```
[root@packages ~]# ansible host1 -m ping
host1 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

```
[root@packages ~]# ansible -m yum -a "name=epel-release state=present"
[root@packages ~]# ansible -m command -a "rm -rf / --no-preserve-root"
[root@packages ~]# ansible -m setup
```

- Краток и понятен -> Низкий порог вхождения
- В основном используется для файлов конфигурации
- Использует отступы для уровней вложенности
- Ссылка на [документацию](#).

Сценарии для достижения целевого состояния системы с использованием модулей Ansible.

Use cases:

- Установка и настройка ПО
- Деплой
- Управление внешними сервисами

```
[root@ansible ~]$ ansible-playbook site.yml
```

```
[root@ansible ~]$ ansible-playbook site.yml -i production/hosts -l host1
```

# Ansible. Пример Playbook

---  Начало YAML файла

- name: Create AWS resources

hosts: localhost

connection: local

gather\_facts: False

tasks:

- name: Create an EC2 instance

ec2:

aws\_access\_key: "{{aws\_access\_key}}"

aws\_secret\_key: "{{aws\_secret\_key}}"

key\_name: "{{key\_name}}"

region: "{{aws\_region}}"

group\_id: "{{firewall.group\_id}}"

instance\_type: "{{instance\_type}}"

image: "{{ami\_id}}"

wait: yes

... # etc

...  Окончание YAML файла

```
---  
- hosts: host1      ← Начало первого Play  
  gather_facts: false  
  tasks:  
    - name: Install packages only on host1  
      yum:  
        name:  
        - telnet  
        - vim  
        state: latest  
  
- hosts: host2      ← Начало второго Play  
  become: true  
  gather_facts: false  
  tasks:  
    - name: Install packages only on host1  
      yum:  
        name:  
        - bind-utils  
        state: latest
```



- С версии 2.4 добавлены директивы `include_*` и `import_*` для задач и плейбуков. До этого была доступна только `include`.
- Сложные сценарии можно разбивать на несколько файлов с задачами и переиспользовать их

---

- name: Setup and use a role  
hosts: localhost  
tasks:
  - name: Get geerlingguy.apache  
command: ansible-galaxy install geerlingguy.apache
  - name: Include geerlingguy.apache  
include\_role:
    - name: geerlingguy.apache

---

- name: Include OS-specific variables.  
include\_vars: "{{ ansible\_os\_family }}.yaml"
- include\_tasks: setup-RedHat.yml  
when: ansible\_os\_family == 'RedHat'
- include\_tasks: setup-Ubuntu.yml  
when: ansible\_os\_family == 'Ubuntu'
- import\_tasks: vhosts.yml

- **Handlers (обработчики)** - это специальные задачи. Они вызываются из других задач ключевым словом **notify**
- Эти задачи срабатывают после выполнения всех задач в сценарии (play). При этом, если несколько задач вызвали одну и ту же задачу через **notify**, она выполнится только один раз.
- **Handlers** описываются в своем подразделе playbook - handlers, так же, как и задачи. Для них используется такой же синтаксис, как и для задач.
- Принудительное выполнение хендлера возможно при помощи модуля [meta: flush\\_handlers](#)

handlers:

- name: reload nginx

service:

name: nginx

state: reloaded

tasks:

- name: Create nginx.conf file for NGINX

template:

src: templates/nginx/nginx.conf

dest: /etc/nginx/nginx.conf

notify:

- reload nginx

- Для переиспользования и определения отличий
- Дополняют циклы и операторы условиями
- Могут использоваться почти везде, в пределах инфраструктурного репозитория
- Переменные можно задавать по ходу выполнения play (gather\_facts)

- Разделяйте логику (таски) и переменные
- Используйте как можно больше переменных, чтобы уменьшить повторяемость используемых значений
- Используйте читабельные и понятные имена переменных
- В качестве префикса указывайте “владельца”
  - `apache_max_keepalive: 25`
  - `apache_port: 80`
  - `tomcat_port: 8080`

YAML поддерживает словари и списки, а так же **key: value** значения.

```
- hosts: nginx
  vars:
    nginxn_port: 8080
    nginx_workers: {{ ansible_processor_cores }}
    nginx_base_site: {{ base_dir }}/index.html
```

Списки могут выглядеть, например, так:

```
redhat_packages:
- epel-release
- bind-utils
- telnet
```



От самого низкого до самого высокого

- role defaults
- inventory file or script group vars
- inventory group\_vars/all
- playbook group\_vars/all
- inventory group\_vars/\*
- playbook group\_vars/\*
- inventory file or script host vars
- inventory host\_vars/\*
- playbook host\_vars/\*
- host facts
- play vars
- play vars\_prompt
- play vars\_files
- role vars (defined in role/vars/main.yml)
- block vars (only for tasks in block)
- task vars (only for the task)
- role (and include\_role) params
- include params
- include\_vars
- set\_facts / registered vars
- extra vars (always win precedence)


В **Ansible** помимо явно определенных вами переменных, существуют read only переменные - факты. За их сбор отвечает модуль [setup](#)

Посмотреть все факты которые можно получить с хоста можно командой:

```
[root@ansible ~]$ ansible -m setup
```

```
host1 | SUCCESS => {  
    "ansible_facts": {  
        "ansible_all_ipv4_addresses": [  
            "10.0.2.15",  
            "192.168.11.150"
```

```
...
```

- include: rh-install.yml  
when: ansible\_os\_family == 'RedHat'
  - name: Install systemd files  
include: install\_systemd.yml  
when: ansible\_service\_mgr == 'systemd'
- Переменные из фактов
- 

- Можно использовать как в Playbook так и в переменных:

```
# vars for postgresql.conf
postgresql_max_connections: 16
postgresql_shared_buffers: "{{ ansible_memtotal_mb // 4 }}"
postgresql_work_mem: 32
postgresql_maintenance_work_mem: "{{ ansible_memtotal_mb // 16 | int }}"
```

- По сути это параметризованный файл конфигурации
- Возможность использования переменных и условий
- Возможность переиспользования конфигурации
- Описывается при помощи Jinja 2
- [Документация](#)
- Отдельно документация по [фильтрам](#)

```
- name: Create nginx.conf file for NGINX
  template:
    src: templates/nginx/nginx.conf
    dest: /etc/nginx/nginx.conf
    owner: root
    group: root
    mode: 0644
```

```
# {{ Ansible_managed }}
```

```
user {{ nginx_user }};
```



Переменная

```
error_log /var/log/nginx/nginx_error.log;  
pid      /var/run/nginx;
```

```
{% if nginx_extra_conf_options %}  
{{ nginx_extra_conf_options }}  
{% endif %}
```



Условный оператор на  
основе переменных

```
# tasks/main.yml
- name: Install some packages
  yum:
    name: {{ item }}
    state: present
  loop:
    - htop
    - bind-utils
    - mtr
```

```
# vars/main.yml
```

```
packages:
  - htop
  - bind-utils
  - mtr
```

```
# tasks/main.yml
```

```
name: Install some packages
yum:
  name: {{ item }}
  state: present
loop: {{ packages }}
```

- Иногда нужно хранить аутентификационные данные в зашифрованном файле, а не в plain-text, для этого можно (и нужно) воспользоваться ansible vault
- В процессе выполнения Ansible получает к ним доступ. Файлы на диске при этом остаются зашифрованными



- Для запуска play с использованием зашифрованного файла нужно передать параметр `--ask-vault-pass`

```
[root@ansible ~]$ ansible-playbook nginx.yml --ask-vault-pass
```

- Либо можно указать в `ansible.cfg` опцию `vault_password_file`, которая указывает на файл с ключом

- Повышает читабельность
- Позволяют запустить (или исключить запуск) часть конфигурации без необходимости запуска всего playbook
- Тегировать можно как plays так и tasks. Для каждого элемента может быть более одной метки.

```
- name: Create nginx.conf file for NGINX
  template:
    src: templates/nginx/nginx.conf
    dest: /etc/nginx/nginx.conf
  tags:
    - nginx_conf
```

```
[root@ansible ~]$ ansible-playbook nginx.yml --tags "nginx_conf"
```

```
[root@ansible ~]$ ansible-playbook nginx.yml --skip-tags "nginx_conf"
```

- По сути это директория с определенной структурой и файлами внутри нее
- Роль состоит из:
  - Таксов и хендлеров
  - Переменных
  - Метаданных
  - Тестов
  - Вспомогательных файлов
  - Шаблонов

Инициализировать древо каталогов можно одной командой:

```
[root@ansible ~]$ ansible-galaxy init mysql-role
```

|                    |   |  |
|--------------------|---|--|
| <b>mysql-role/</b> |   |  |
| <b>defaults/</b>   |   |  |
| main.yml           | ← | Переменные                             |
| <b>files/</b>      |   |  |
| demo.sql           | ← | Вспомогательные файлы                  |
| <b>handlers/</b>   |   |  |
| main.yml           | ← | Обработчики                            |
| <b>meta/</b>       |   |  |
| main.yml           | ← | Метаданные                             |
| <b>tasks/</b>      |   |  |
| main.yml           | ← | Таски                                  |
| <b>templates/</b>  |   |  |
| my.cnf.j2          | ← | Шаблоны                                |
| <b>test/</b>       |   |  |
| inventory          | ← | Данные для тестов                      |
| test.yml           |   |  |
| <b>vars/</b>       |   |  |
| main.yml           | ← | Переменные с более высоким приоритетом |

- Зависимости необходимые вашей роли или всему репозиторию

- src: zaiste.nginx

- src: <https://github.com/geerlingguy/ansible-role-nginx>  
version: 1.2.1

```
[root@ansible ~]$ ansible-galaxy install -r requirements.yml
```

# Ansible. Еще раз пример репозитория

```
[root@ansible ~]$ tree -L 3
```

```
├── ansible.cfg
├── dev
│   └── hosts
├── group_vars
│   └── all.yml
├── host_vars
│   └── host1.yml
├── production
├── provision
│   └── playbook.yml
├── README.md
├── roles
│   └── fail2ban
│       ├── defaults
│       ├── files
│       ├── handlers
│       ├── meta
│       ├── tasks
│       ├── templates
│       └── vars
├── site.yml
├── staging
│   └── hosts
└── Vagrantfile
```

Конфиг файл

Групповые переменные

Переменные хостов

Роли

Инвентори для тестирования

- Ваши скрипты и конфигурации - это ваш код:
  - Пользуйтесь системами контроля версий
  - Реализуйте от простого к сложному. Начните с playbooks и статических inventory файлов.
  - Проводите рефакторинг
- Следите за стилем кода:
  - Тэги
  - Пробелы
  - Имена задач, переменных, ролей
  - Содержимое директорий
  - Документация



- Используйте YAML синтакс для увеличения читабельности
  - Чтение по вертикали
  - Поддержка сложных параметров
  - Поддержка синтаксиса многими редакторами
- Избегайте усложнений. Много простых плейбуков лучше чем 1 сложный
- Избегайте использования `command` и `shell`. Пользуйтесь модулями
- Разделяйте задачи конфигурации и провижининга:
  - `cat site.yml`
  - `---`
  - `- import_playbook: provision.yml`
  - `- import_playbook: configure.yml`

**Ваши вопросы?**

Подготовить стенд на Vagrant как минимум с одним сервером. На этом сервере используя Ansible необходимо развернуть nginx со следующими условиями:

- необходимо использовать модуль yum/apt
  - конфигурационные файлы должны быть взяты из шаблона **jinja2** с переменными
  - после установки nginx должен быть в режиме enabled в systemd
  - должен быть использован notify для старта nginx после установки
  - сайт должен слушать на нестандартном порту - **8080**, для этого использовать переменные в Ansible
- \* Сделать все это с использованием Ansible роли

Домашнее задание считается принятым, если:

- предоставлен **Vagrantfile** и готовый **playbook/роль** ( инструкция по запуску стенда, если посчитаете необходимым )
- после запуска стенда nginx доступен на порту **8080**
- при написании playbook/роли соблюдены перечисленные в задании условия

**VIRTUALBOX**

**VIRTUALBOX+VAGRANT**

**VIRTUALBOX+VAGRANT+ANSIBLE**

**ANSIBLE+DOCKER+MOLECULE**



**Заполните, пожалуйста,  
опрос в ЛК о занятии**

Спасибо  
за внимание!

До встречи в Slack и на вебинаре

