



ОНЛАЙН-ОБРАЗОВАНИЕ



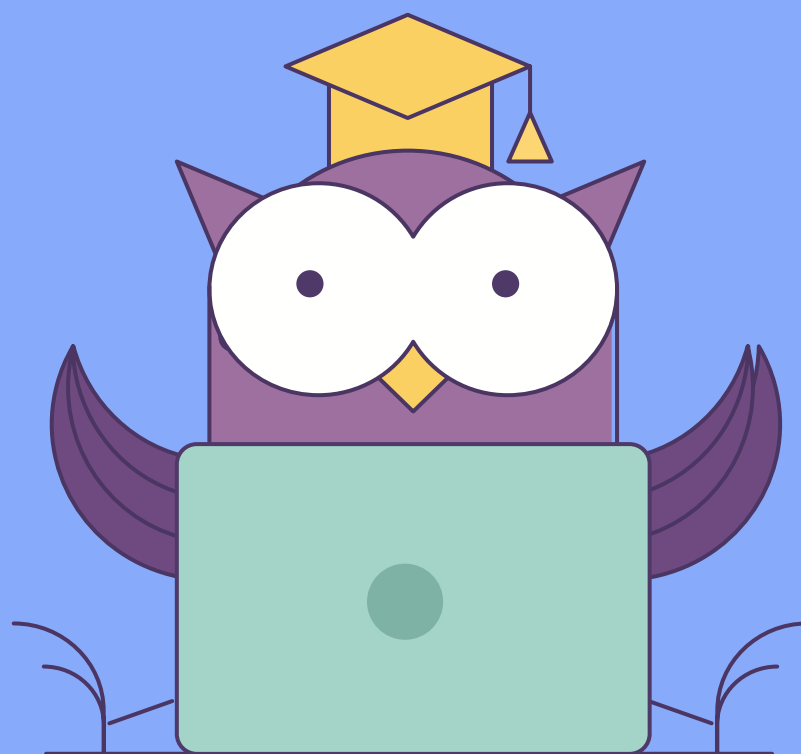
Управление конфигурациями. Ansible

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



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	Client-Only (No)	Client/Server Client



ANSIBLE

Плюсы:

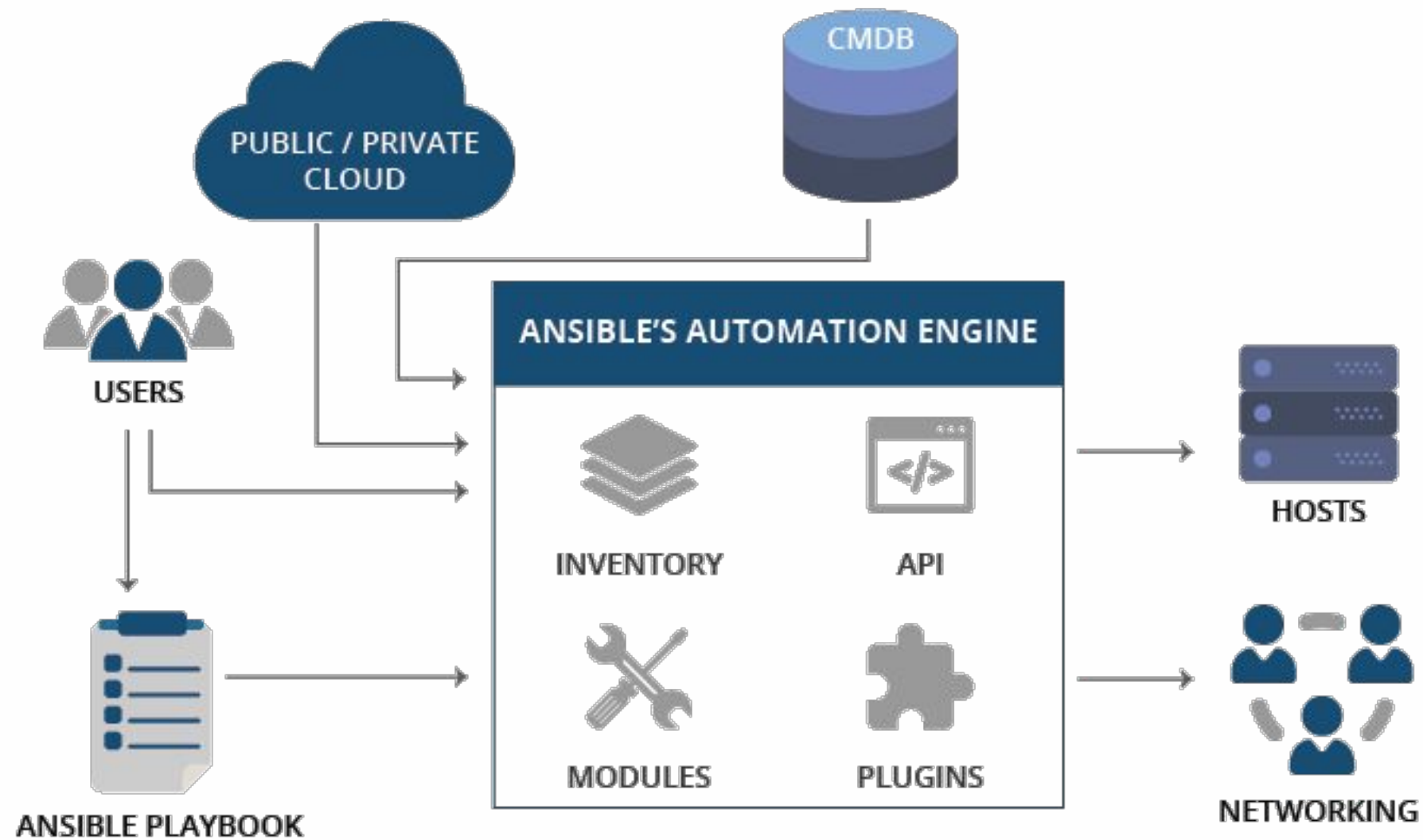
- Легкий порог вхождения
- Понятная документация
- SSH - отсутствие агента
- Большое кол-во ролей/модулей от сообщества
- Не нужно поддерживать дополнительную инфраструктуру
- Можно дописать свои модули на python
- Легко бутстрапить новые серваки
- Прозрачное выполнение Playbook
- Поддержка со стороны RedHat

Минусы:

- Из коробки статический инвентори
- Скорость выполнения (при большом кол-ве машин и/или их удаленности от деплой хоста)
- Возможное расхождение между желаемой и текущей конфигурации
- Актуальность верности плейбука
- Нет базы фактов обо всех серверах
- Нет истории деплоя, непонятно кто-что раскатал

- Ansible 2.9
- 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

ANSIBLE ARCHITECTURE



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 (статические файлы)
- Документацию о проекте (readme.md)
- Ваши тесты (molecule, infra, ...)

Ansible. Пример репозитория

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

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

Конфиг файл

Групповые переменные

Переменные хостов

Все инвентори

Плейбуки/сценарии

Роли

- 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

← Переменные

← 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:
```

- Библиотеки для выполнения и отслеживания состояния задач. По сути код который формирует другой код который выполняется на удаленной машине:
 - Типовые операции ОС
 - Управление ресурсами
 - Все остальное
- Основа для выполнения задач в Ansible
- [Список](#) модулей из документации

yum: ← Имя модуля
name: epel-release ← Параметры модуля
state: present

- 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

Ansible. Несколько Play

```
---  
- 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
```


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

- 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"        ]  
    }  
}
```

...

```
# 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

- 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.conftags:
 - nginx_conf

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

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

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 и официальный репозиторий NGINX
- конфигурационные файлы должны быть взяты из шаблона **jinja2** с переменными
- после установки nginx должен быть в режиме enabled в systemd
- сайт должен слушать на нестандартном порту - **8080**, для этого использовать переменные в Ansible

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

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

**Спасибо
за внимание!**

До встречи в Slack и на вебинаре

