



ОНЛАЙН-ОБРАЗОВАНИЕ

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

- Версия Ansible =>2.4 требует для своей работы Python 2.6 или выше
- Убедитесь что у Вас установлена нужная версия:

```
[root@nginx ~#] python -V  
Python 2.7.15
```

- Далее произведите установку для Вашей ОС по [инструкции](#) и убедитесь что Ansible установлен корректно:

```
[root@nginx ~#] ansible --version  
ansible 2.6.0
```

- Для управления хостами Ansible использует SSH соединение. Поэтому перед стартом необходимо убедиться что у Вас есть доступ до управляемых хостов.
- Также на управляемых хостах должен быть установлен Python 2.X

- Создайте каталог **Ansible** и положите в него этот [Vagrantfile](#)
- Поднимите управляемый хост командой **vagrant up** и убедитесь что все прошло успешно и есть доступ по ssh
- Для подключения к хосту **nginx** нам необходимо будет передать множество параметров - это особенность Vagrant. Узнать эти параметры можно с помощью команды **vagrant ssh-config**. Вот основные необходимые нам:

**[root@nginx ~#] vagrant ssh-config**

Host nginx ← имя хоста

HostName 127.0.0.1 ← IP адрес

User vagrant ← имя пользователя под которым подключаемся

Port 2222 ← порт, который проброшен на 127.0.0.1

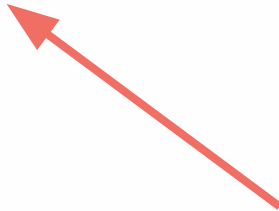
IdentityFile .vagrant/machines/nginx/virtualbox/private\_key

← путь до приватного ключа

- Используя эти параметры создадим свой первый **inventory** файл. Выглядеть он будет так:

[web]

```
nginx ansible_host=127.0.0.1 ansible_port=2222 ansible_user=vagrant  
ansible_private_key_file=.vagrant/machines/nginx/virtualbox/private_key
```



Это все одна строка

И наконец убедимся, что Ansible может управлять нашим хостом. Сделать это можно с помощью команды:

```
[root@nginx ~#] ansible nginx -i staging/hosts -m ping
```

```
nginx | SUCCESS => {  
    "changed": false,  
    "ping": "pong"  
}
```

- Как видимо нам придется каждый раз явно указывать наш инвентори файл и вписывать в него много информация. Это можно обойти используя **ansible.cfg** файл - прописав конфигурацию в нем.
- Для этого в текущем каталоге создадим файл **ansible.cfg** со следующим содержанием:

```
[defaults]
inventory = staging/hosts
remote_user = vagrant
host_key_checking = False
retry_files_enabled = False
```

- Теперь из инвентори можно убрать информацию о пользователе:

```
[web]
nginx ansible_host=127.0.0.1 ansible_port=2222
ansible_private_key_file=.vagrant/machines/nginx/virtualbox/private_key
```



- Еще раз убедимся, что управляемый хост доступен, только теперь без явного указания inventory файла:

```
[root@nginx ~#] ansible nginx -m ping
```

```
nginx | SUCCESS => {  
    "changed": false,  
    "ping": "pong"  
}
```

- Теперь, когда мы убедились, что у нас все подготовлено - установлен Ansible, поднят хост для теста и Ansible имеет к нему доступ, мы можем конфигурировать наш хост.
- Для начала воспользуемся **Ad-Hoc** командами и выполним некоторые удаленные команды на нашем хосте.

- Посмотрим какое ядро установлено на хосте:

```
[root@nginx ~#] ansible nginx -m command -a "uname -r"
```

```
nginx | SUCCESS | rc=0 >>  
3.10.0-862.2.3.el7.x86_64
```

- Проверим статус сервиса firewalld

```
[root@nginx ~#] ansible nginx -m systemd -a name=firewalld
```

```
nginx | SUCCESS => {  
  "changed": false,  
  "name": "firewalld",  
  "status": {  
    "ActiveEnterTimestampMonotonic": "0",  
    "ActiveExitTimestampMonotonic": "0",  
    "ActiveState": "inactive", ← не активен
```

- Установим пакет epel-release на наш хост

```
[root@nginx ~]# ansible nginx -m yum -a "name=epel-release state=present" -b
```

```
nginx | SUCCESS => {  
  "changed": true, ← пакет установлен
```

- Напишем простой **Playbook** который будет делать одно из действий, которое мы делали на прошлом слайде - а именно: установку пакета **epel-release**. Создайте файл `epel.yml` со следующим содержимым:

---

```
- name: Install EPEL Repo
  hosts: nginx
  become: true
  tasks:
    - name: Install EPEL Repo package from standard repo
      yum:
        name: epel-release
        state: present
```

- После чего запустите выполнение Playbook:

```
[root@nginx ~#] ansible-playbook epel.yml
```

```
PLAY [Install EPEL Repo] *****
```

```
TASK [Gathering Facts] *****
```

```
ok: [nginx]
```

```
TASK [Install EPEL Repo package from standart repo] *****
```

```
ok: [nginx]
```

```
PLAY RECAP *****
```

```
nginx : ok=2 changed=0 unreachable=0 failed=0
```

- Затем выполните команду `ansible nginx -m yum -a "name=epel-release state=absent" -b`, еще раз запустите Playbook и посмотрите на разницу в выводе.

- Теперь собственно приступим к выполнению домашнего задания и написания Playbook-а для установки **NGINX**. Будем писать его постепенно, шаг за шагом. И в итоге трансформируем его в роль.
- За основу возьмем уже созданный нами файл `epel.yml` (я его переименую в `nginx.yml`). И первым делом добавим в него установку пакета NGINX. Секция будет выглядеть так:

- name: Install nginx package from epel repo

- yum:

- name: nginx

- state: latest

- tags:

- nginx-package

- packages



Как видите добавлены tags

- Целиком файл будет выглядеть [так](#)

- Обратите внимание - добавили **Tags**. Теперь можно вывести в консоль список тегов и выполнить, например, только установку NGINX. В нашем случае так, например, можно осуществлять его обновление.

- Выведем в консоль все теги:

```
[root@nginx ~#] ansible-playbook nginx.yml --list-tags
```

```
playbook: epel.yml
```

```
play #1 (nginx): NGINX | Install and configure NGINX TAGS: []
```

```
TASK TAGS: [epel-package, nginx-package, packages]
```

- Запустим только установку NGINX:

```
[root@nginx ~#] ansible-playbook nginx.yml -t nginx-package
```

```
PLAY RECAP *****
```

```
nginx                : ok=2 changed=0    unreachable=0    failed=0
```

- Далее добавим шаблон для конфига NGINX и модуль, который будет копировать этот шаблон на хост:
  - name: NGINX | Create NGINX config file from template  
template:  
src: templates/nginx.conf.j2  
dest: /tmp/nginx.conf  
tags:
    - nginx-configuration
- Сразу же пропишем в Playbook необходимую нам переменную. Нам нужно чтобы NGINX слушал на порту 8080:
  - name: NGINX | Install and configure NGINX  
hosts: nginx  
become: true  
vars:
    - nginx\_listen\_port: 8080



Добавлена только секция **vars**

В итоге на данном этапе Playbook будет выглядеть так



- Сам шаблон будет выглядеть так:

```
events {  
    worker_connections 1024;  
}  
  
http {  
    server {  
        listen      {{ nginx_listen_port }} default_server;  
        server_name default_server;  
        root        /usr/share/nginx/html;  
  
        location / {  
        }  
    }  
}
```

- Ссылка на [GIST](#)

- Теперь создадим **handler** и добавим **notify** к копированию шаблона. Теперь каждый раз когда конфиг будет изменяться - сервис перезагрузиться. Секция с handlers будет выглядеть следующим образом:

handlers:

- name: restart nginx

systemd:

name: nginx

state: restarted

enabled: yes

← Так же создадим handler для  
рестарта и включения сервиса  
при загрузке

- name: reload nginx

systemd:

name: nginx

state: reloaded

← Перечитываем конфиг

- **Notify будут выглядеть так:**
  - name: NGINX | Install NGINX package from EPEL Repo  
yum:  
  name: nginx  
  state: latest  
notify:  
  - restart nginx  
tags:  
  - nginx-package  
  - packages
  - name: NGINX | Create NGINX config file from template  
template:  
  src: templates/nginx.conf.j2  
  dest: /etc/nginx/nginx.conf  
notify:  
  - reload nginx  
tags:  
  - nginx-configuration
- Результирующий файл [nginx.yml](#). Теперь можно его запустить

```
[root@nginx ~#] ansible-playbook playbooks/nginx.yml
```

```
PLAY [NGINX | Install and configure NGINX] *****
```

```
TASK [Gathering Facts] *****
```

```
ok: [nginx]
```

```
TASK [NGINX | Install EPEL Repo package from standart repo] *****
```

```
changed: [nginx]
```

```
TASK [NGINX | Install NGINX package from EPEL Repo] *****
```

```
changed: [nginx]
```

```
TASK [NGINX | Create NGINX config file from template] *****
```

```
changed: [nginx]
```

```
RUNNING HANDLER [restart nginx] *****
```

```
changed: [nginx]
```

```
RUNNING HANDLER [reload nginx] *****
```

```
changed: [nginx]
```

```
PLAY RECAP *****
```

```
nginx : ok=6 changed=5 unreachable=0 failed=0
```

- Теперь можно перейти в браузере по адресу <http://192.168.11.150:8080> и убедиться, что сайт доступен.
- Или из консоли выполнить команду:

```
[root@nginx ~#] curl http://192.168.11.150:8080
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"  
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">  
  <head>  
    <title>Test Page for the Nginx HTTP Server on Fedora</title>  
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />  
    <style type="text/css">  
      /*<![CDATA[*/  
      body {
```

...



