Ansible. Практическое занятие

Задача: попрактиковаться в написании плейбука, со следующими условиями:
- подготовить стенд на Vagrant,
- настроить ansible для доступа к стенду
- написать плейбук, который устанавливает NGINX в конфигурации по умолчанию, с применением модуля yum
- подготовить шаблон jinga2 новой конфигурации для nginx, чтобы сервис слушал на нестандартном порту 8080. В шаблоне для номера порта использовать переменные ansible
- добавить в плейбук копирование новой конфигурации сервиса на стенд
- должен быть использован механизм notify для рестарта nginx после установки или изменения конфигурации

Установка Ansible

Версия Ansible =>2.4 требует для своей работы Python 2.6 или выше
Проверьте, что версии python и ansible достаточно новые:

```
[user@fedora ansible]$ python -V
Python 3.9.7
```

• Если необходимо, установите ansible (yum install ...) и убедитесь что он установлен корректно:

```
[user@fedora ansible]$ ansible --version
ansible 2.9.25
```

Подготовка стенда

Создать каталог ansible
В этот каталог скопировать присланные файлы, убедиться что файл Vagrant не содержит ошибок
[user@ubunta ~ansible]$ vagrant status

```
nginx                     not created (virtualbox)
```

[user@ubunta ~ansible]$ vagrant up
…

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 (значения у всех могут отличаться, учитывайте различия):

```
[user@fedora ansible]$ vagrant ssh-config
Host nginx
 HostName 127.0.0.1
 User vagrant  (имя пользователя, под которым подключаемся)
 Port 2203     (порт для доступа к виртуальной машине)
 UserKnownHostsFile /dev/null
 StrictHostKeyChecking no
 PasswordAuthentication no
 IdentityFile
/home/user/sirius/10_Ansible/ansible/.vagrant/machines/nginx/virtu
albox/private_key    (путь до приватного ключа)
 IdentitiesOnly yes
 LogLevel FATAL
```

Inventory

Используя эти параметры, создадим свой первый inventory файл. Выглядеть он будет так

```
[user@fedora ansible]$ cat inventory
[webservers]
nginx  ansible_host=127.0.0.1 ansible_port=2203 ansible_user=vagrant
ansible_private_key_file=/home/user/sirius/10_Ansible/ansible/.vagrant/machines/nginx/
virtualbox/private_key
```

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

```
[user@fedora ansible]$ ansible nginx -i inventory -m ping
nginx | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
```

Если результат не SUCCESS, то ищите причину в inventory

ansible.cfg

Чтобы не пришлось в дальнейшем каждый раз явно указывать наш инвентори файл в командной строке, создадим файл конфигурации ansible.cfg

Для этого в текущем каталоге создадим файл ansible.cfg со следующим содержанием:

```
[user@fedora ansible]$ cat ansible.cfg
[defaults]
inventory = inventory
remote_user= vagrant
host_key_checking = False
transport=smart
```

Полезно почитать справку, понять какой параметр что означает
https://docs.ansible.com/ansible/2.6/reference_appendices/config.html

- Теперь из инвентори можно убрать информацию о пользователе:

```
[user@fedora ansible]$ cat inventory
[webservers]
nginx ansible_host=127.0.0.1 ansible_port=2203 ansible_ssh_private_key_file=.vagrant/ma
chines/nginx/virtualbox/private_key
```

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

```
[user@fedora ansible]$ ansible -m ping nginx
nginx | SUCCESS => {
   "ansible_facts": {
      "discovered_interpreter_python": "/usr/bin/python"
   },
   "changed": false,
   "ping": "pong"
}
```

AD-HOC, или однострочные команды

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:

```
[user@fedora ansible]$ ansible nginx -m command -a "uname -r"

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

• Проверим статус сервиса firewalld

```
[user@fedora ansible]$ ansible nginx -m systemd -a name=firewalld

nginx | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "name": "firewalld",
    "status": {
...
        "ActiveState": "inactive",
...
```

Установим пакет epel-release на наш хост

```
[user@fedora ansible]$ ansible nginx -m yum -a "name=epel-release state=present" -b

nginx | CHANGED => {
...
"changed": true,          пакет установился
…
Installed:\n  epel-release.noarch 0:7-11
 \n\nComplete!\n"
```

Playbook epel

Напишем простой Playbook который будет выполнять одну из задач, которые мы делали на прошлом слайде - а именно: установку пакета epel-release. Создайте файл epel.yml со следующим содержимым. Внимательно соблюдайте отступы, т. к. YaML очень чувствителен к синтаксису:

```
[user@fedora ansible]$ cat epel.yml
---
- name: Install EPEL Repo
  hosts: webservers
  become: true
  tasks:
    - name: Install EPEL Repo package from standard repo
      yum:
        name: epel-release
        state: present
```

После чего запустите выполнение Playbook:

```
[user@fedora ansible]$ ansible-playbook epel.yml

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

TASK [Gathering Facts]
*****************************************************************
ok: [nginx]


TASK [Install EPEL Repo package from standard repo]
***********************************
ok: [nginx]

PLAY RECAP
*****************************************************************
**********
nginx : ok=2    changed=0 unreachable=0 failed=0 skipped=
0 rescued=0 ignored=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.
О чем говорит полученный результат?

Playbook nginx

Теперь собственно приступим к выполнению домашнего задания и написания Playbook-а для установки NGINX. Будем писать его постепенно, шаг за шагом.

За основу возьмем уже созданный нами плейбук epel.yml. Скопируйте этот файл с именем nginx.yml (командой cp)

Добавим в этот новый файл установку пакета nginx. Секция будет выглядеть так:

```
[user@fedora ansible]$ cat nginx.yml
- name: Install nginx package from epel repo
  hosts: webservers
  become: true
  tasks:
    - name: Install EPEL Repo package from standard repo
      yum:
        name: epel-release
        state: present
    - name: install nginx from repo
      yum:
        name: nginx
        state: latest
      tags:
         nginx-package
         packages
```

Обратите внимание - добавили tags. Теперь можно вывести в консоль список тегов и выполнить, например, только часть из задач описанных в плейбуке, а именно установку NGINX. В нашем случае так, например, можно осуществлять его обновление.
Выведем в консоль все теги:

```
[user@fedora ansible]$ ansible-playbook nginx.yml –list-tags

playbook: nginx.yml

  play #1 (webservers): Install nginx package from epel repo
TAGS: []
      TASK TAGS: [nginx-package, packages]
```

Запустим только установку NGINX, использовав любой из подходящих тегов:

```
[user@fedora ansible]$ ansible-playbook nginx.yml --tag packages
PLAY [Install nginx package from epel repo]
*******************************************

TASK [Gathering Facts]
*************************************************************
ok: [nginx]


TASK [install nginx from repo]
*****************************************************
changed: [nginx]
```

```
PLAY RECAP
*********************************************************************
**********
nginx                      : ok=2    changed=1    unreachable=0
failed=0    skipped=
0    rescued=0    ignored=0
```

Если в процессе выполнения плейбука возникли ошибки, проанализируйте вывод и постарайтесь определить причину.

Шаблон

Далее создадим файл шаблона для конфига NGINX, имя файла nginx.conf.j2. Обратите внимание, что в шаблоне используется переменная, которую в дальнейшем надо где-то определить.

```
[user@fedora ansible]$ cat nginx.conf.j2
events {
 worker_connections 1024;
}

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

И добавим в плейбук задачу, которая копирует подготовленный шаблон на хост. Для этой задачи используется модуль template, можно ознакомится с документацией на модуль https://docs.ansible.com/ansible/latest/collections/ansible/builtin/template_module.html

Также в плейбук добавлено определение переменной в секции vars

```
[user@fedora ansible]$ cat nginx.yml

---
- name: Install EPEL Repo
  hosts: webservers
  become: true
  vars:
    nginx_listen_port: 8080
  tasks:
    - name: Install EPEL Repo package from standard repo
      yum:
        name: epel-release
        state: present
    - name: install nginx from repo
      yum:
        name: nginx
        state: latest
      tags:
        nginx-package
        packages
    - name: Create config file from template
      template:
        src: nginx.conf.j2
        dest: /etc/nginx/nginx.conf
      tags:
        nginx-configuration
```

Handlers

Плейбук сейчас уже почти работоспособен, осталось только добавить секции handler и notify для рестарта nginx не при любом старте плейбука, а только при изменения в конфигурации. Для рестарта сервисов применяется модуль systemd, документация к модулю https://docs.ansible.com/ansible/latest/collections/ansible/builtin/systemd_module.html

```
[user@fedora ansible]$ cat nginx.yml
---
- name: Install nginx package from epel repo
 hosts: webservers
 become: true
 vars:
   nginx_listen_port: 8080
 tasks:
   - name: Install EPEL Repo package from standard repo
     yum:
       name: epel-release
       state: present
   - name: install nginx from repo
     yum:
       name: nginx
       state: latest
     notify:
       - restart nginx
     tags:
      - nginx-package
      - packages
   - name: Create config file from template
     template:
       src: nginx.conf.j2
       dest: /etc/nginx/nginx.conf
     notify:
       - reload nginx
     tags:
       - nginx-configuration
 handlers:
   - name: restart nginx
     systemd:
       name: nginx
       state: reconfigured
       enabled: yes
```

Исправление ошибок

В этот плейбук намеренно внесены ошибки. Прочитайте документацию, постарайтесь найти и исправить их.

Отлаживать работу плейбука можно, запуская и анализируя сообщения об ошибке, если таковые будут.

После исправления ошибок, плейбук успешно отработает примерно так:

```
[user@fedora ansible]$ ansible-playbook nginx.yml


PLAY [Install nginx package from epel repo]
*******************************************

TASK [Gathering Facts]
*************************************************************
ok: [nginx]


TASK [Install EPEL Repo package from standard repo]
***********************************
ok: [nginx]

TASK [install nginx from repo]
*****************************************************
ok: [nginx]

TASK [Create config file from template]
*********************************************
changed: [nginx]

RUNNING HANDLER [restart nginx]
*****************************************************
changed: [nginx]

PLAY RECAP
*************************************************************************
**********
nginx                      : ok=5    changed=2    unreachable=0
failed=0    skipped=
0    rescued=0    ignored=0
```

Проверка

Теперь, чтобы проверить работу NGINX на нестандартном порту, нам надо выяснить IP адрес, который получила виртуальная машина при создании vagrant-ом
Это можно сделать командой

```
[user@fedora ansible]$ vagrant ssh -c "ip addr show"
```

В полученном выводе найдите описание сетевых интерфейсов. Первый интерфейс — локальный loopback, второй — автоматически добавленный вагрантом, предназначеный для интерконнекта ядра вагранта к виртуальной машине и не доступен снаружи виртуалки. А третий как раз публичный и к нему можно обращаться.

Попингуйте этот адрес, убедитесь, что он отвечает.

Затем в браузере откройте страницу

```
http://ip_address:8080
```