Управление инфраструктурой. Ansible.

Задание 0. Построение стенда

Схема виртуального лабораторного стенда

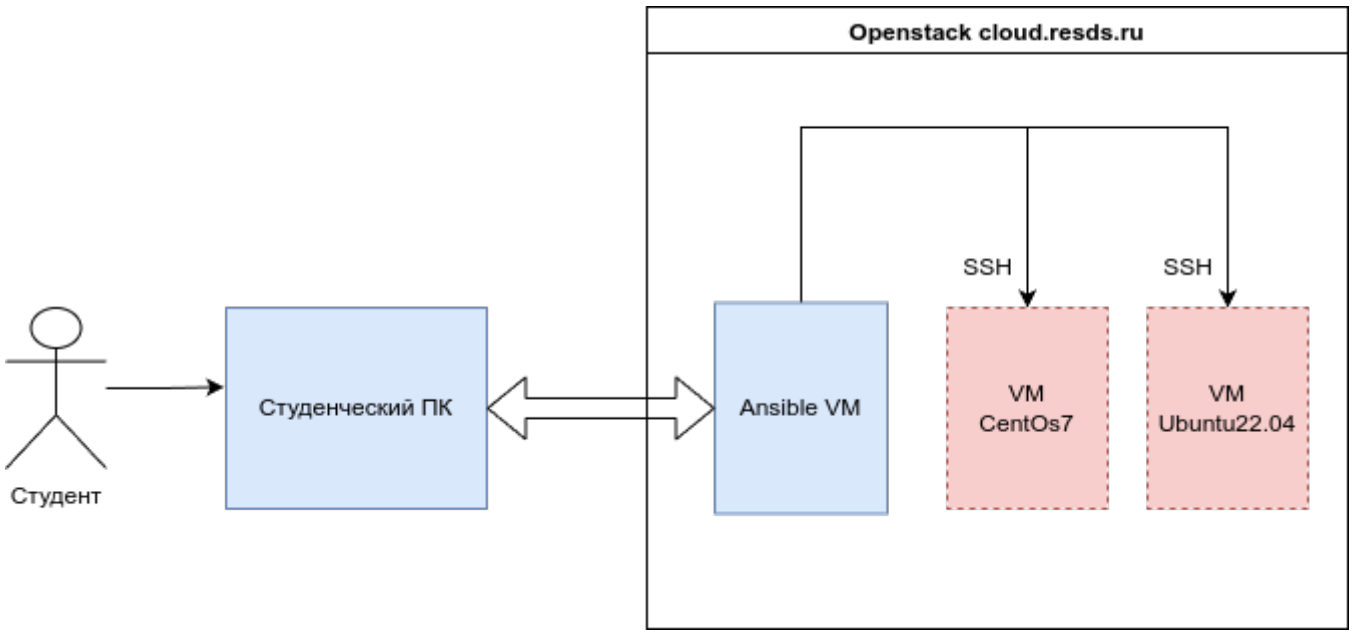


Рисунок 1. Схема стенда

1. Создать виртуальный стенд для работы

Название виртуальной машины	Источник	Тип инстанса	Сети для внешнего подключения	Размер диска
Ansible	Образ-Ubuntu-server20.04	small	external-net	15GB
node1	Образ-Ubuntu-server20.04	small	external-net	15GB

Название виртуальной машины	Источник	Тип инстанса	Сети длявнешнего подключения	Размер диска
node2	Образ-CentOS-7	small	external-net	15GB

2. Установить ANSIBLE

Ниже будут представлены несколько способов установки Ansible. От выбора метода установки зависит удобство использования и доступные функции, поэтому важно выбрать оптимальный способ в соответствии с вашими потребностями и предпочтениями и операционной системы

PIP

1. Проверить наличие `python` на узле:

   ```
   python3 -v
   ```

2. При отсутствии `python`, его необходимо установить:

   ```
   sudo apt install python3
   ```

3. Проверяем наличие менеджера пакетов `pip`:

   ```
   python3 -m pip --version
   ```

4. При отсутствии `pip`, его необходимо установить:

   ```
   sudo apt install python3-pip
   ```

При наличии python с менеджером пакетов pip, можно установить используя его.

Для установки с помощью `pip` необходимо ввести команду

```
python3 -m pip install --user ansible
```

При необходимости можно установить пакет `ansible-core` он отличается тем, что с помощью него возможно использовать только язык и рантайм Ansible, и отсутствует интеграция с galaxy

```
python3 -m pip install --user ansible-core
```

> Ansible и Ansible Core тесно связаны, но есть небольшая разница между ними. Ansible Core представляет собой базовый движок автоматизации, который включает основные функции управления конфигурациями и выполнения задач через SSH. Он является основой для всей экосистемы Ansible. С другой стороны, Ansible как платформа включает в себя не только ядро, но и дополнительные инструменты, модули, плагины и библиотеки, расширяющие функциональность и возможности автоматизации. Таким образом, Ansible Core представляет собой базовую часть, в то время как Ansible включает в себя эту базу и дополнительные компоненты для расширения функциональности и упрощения управления инфраструктурой.

Ubuntu

Для установки на Ubuntu можно использовать стандартный менеджер пакетов apt

```
sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository --yes --update ppa:ansible/ansible
sudo apt install ansible
```

Debian

Репозиторий debian стал deprecated, возможно использование обходных путей для установки пакета

Debian	Ubuntu	UBUNTU_CODENAME
Debian 12 (Bookworm)	Ubuntu 22.04 (Jammy)	`jammy`
Debian 11 (Bullseye)	Ubuntu 20.04 (Focal)	`focal`
Debian 10 (Buster)	Ubuntu 18.04 (Bionic)	`bionic`

Пример для Debian12

```
UBUNTU_CODENAME=jammy

wget -O-
"https://keyserver.ubuntu.com/pks/lookup?fingerprint=on&op=get&search=0x6125E2A8C77F2818FB7BD15B93C4A3FD7BB9C367" | sudo gpg --dearmour -o /usr/share/keyrings/ansible-archive-keyring.gpg

echo "deb [signed-by=/usr/share/keyrings/ansible-archive-keyring.gpg] http://ppa.launchpad.net/ansible/ansible/ubuntu $UBUNTU_CODENAME main" | sudo tee
```

```
/etc/apt/sources.list.d/ansible.list


sudo apt update && sudo apt install ansible
```

Проверка установки ansible

```
ansible --version
```

3. Минимальная настройка ANSIBLE

Хоть Ansible может работать из коробки и не требует дополнительной настройки, мы выполним настройки которые улучшат опыт использования системы автоматизации для этого мы можем использовать конфигурационные параметры и они могут храниться в различных местах:

1. ANSIBLE_CONFIG (переменная окружения)
2. ansible.cfg (в текущем каталоге, откуда происходит запуск)
3. ~/.ansible.cfg (в домашнем каталоге пользователя)
4. /etc/ansible/ansible.cfg
   В ходе работы мы предлагаем такой конфиг, его необходимо разместить в домашней директории пользователя:

```
[defaults]
# Отключение проверки хостовых ключей SSH. Позволяет Ansible подключаться к хостам без подтверждения их хостовых ключей.
host_key_checking   = False
# Настройка метода сбора информации о системе хоста. "smart" означает автоматически определить наилучший метод, исходя из условий.
gathering           = smart
# Указание метода передачи данных между хостами.
transfer_method     = piped
# Настройка параметров SSH.
ssh_args            = "-o ControlMaster=auto -o ControlPersist=60s"
# Максимальное количество параллельных процессов (форков) Ansible. Определяет, сколько хостов может обрабатываться параллельно.
forks               = 20
```

> Тут представлена малая часть параметров которые можно использовать, полный список можно получить в документации

> https://docs.ansible.com/ansible/latest/reference_appendices/config.html

4. Написание инвентари

Ansible использует файлы инвентаря для определения групп хостов и их параметров. Создайте файл `inventory.ini` и определите в нем хосты, с которыми будет взаимодействовать Ansible:

```
[node]
# Добавление узла  с именем web и ip адресом 172.17.5.5
node1 ansible_host=172.17.5.5


[all:vars]
# Добавление общей переменной для всех хостов в inventory, с указанием общего имени пользователя
ansible_user    =    cloudadmin
```

Проверить inventory, можно использовав встроенный модуль ansible `ping`:

```
ansible -i inventory.ini all -m ping
```

Пример валидного ответа

```
~/itt_ansible ansible -i inventory.ini  all -m ping
web | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

Также можно получить всю существующую информацию об хостах:

```
ansible -i inventory.ini all -m setup
```

Установить пакет `git` используя модуль `apt`:

```
ansible all -i inventory.ini -m apt -a "name=git state=present" --become
```

> Опция `--become` в командах Ansible используется для выполнения задач с привилегиями суперпользователя (обычно root). Грубо говоря как использования `sudo` при работе в терминале.

```
~/itt_ansible  ansible all -i inventory.ini -m apt -a "name=git state=present" --become
web | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1710766184,
    "cache_updated": false,
    "changed": false
}
```

Выполнить команду `ss -tulpan` на удаленных узлах:

```
ansible all -i inventory.ini -m command -a "ss -tulpan" --become
```

5. Написание плейбуков

Плейбуки в Ansible - это текстовые файлы в формате YAML, которые содержат описание задач, должны быть выполнены на целевых хостах. Их основное предназначение - автоматизация конфигурации и управления системами. Попробуем написать простой плейбук и будем его пополнять в процессе работы:

```
---
- name: Update packages
  hosts: node
  become: yes
  tasks:
  - name: Update all packages
    apt:
      update_cache: yes
      upgrade: 'yes'
```

Для выполнения плейбука можно выполнить команду:

```
ansible-playbook playbook.yaml -i inventory.ini
```

Пример того, когда при выполнение плейбука обновляются пакеты:

```
~/itt_ansible  ansible-playbook playbook.yaml -i inventory.ini                4 ✘  ilya@339-2-ilya  23:57:53

PLAY [Update packages] ************************************************************************************

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

TASK [Update all packages] ********************************************************************************
changed: [web]

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

Пример выполнения плейбука, когда пакеты не были обновлены:

```
~/itt_ansible  ansible-playbook playbook.yaml -i inventory.ini                ✔ | 8s | ilya@339-2-ilya | 00:03:11

PLAY [Update packages] ************************************************************************************************

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

TASK [Update all packages] ********************************************************************************************
ok: [web]

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

Можно заметить, что в данном случае плейбук состоит всего из одной задачи, но плейбуки могут состоять из множетсва задача, давайте добавим задачу для установки NGINX, и передачи ему конфигурации:

```
---
- name: Update packages
  hosts: node
  become: yes
  tasks:
  - name: Update all packages
    apt:
      update_cache: yes
      upgrade: 'yes'
  - name: Install nginx
    ansible.builtin.apt:
      package:
        -  nginx
      state: present
      update_cache: yes
  - name: Copy config nginx
    ansible.builtin.template:
      src:  nginx.conf.j2
      dest: /etc/nginx/nginx.conf
```

Также можно заметить, что в последней строке происходит копирования файлов, нам необходимо создать файл `nginx.conf.j2` и заполнить его:

```
user  {{ nginx_user }};
worker_processes  2048;
worker_priority     -1;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;

events {
```

```
  worker_connections {{worker_connections}};
}

http {
  include       /etc/nginx/mime.types;
  default_type  application/octet-stream;
  log_format  main  '$remote_addr - $remote_user [$time_local] "$request" ' '$status $body_bytes_sent
"$http_referer" ' '"$http_user_agent" "$http_x_forwarded_for"';
  access_log  /var/log/nginx/access.log  main;
  sendfile on;
  tcp_nopush on;
  tcp_nodelay on;
  keepalive_timeout  65;
  reset_timedout_connection  on;
  client_body_timeout        35;
  send_timeout               30;
  gzip on;
  gzip_min_length     {{gzip_min_length}};
  gzip_vary on;
  gzip_proxied        expired no-cache no-store private auth;
  gzip_types          text/plain text/css application/json application/x-javascript text/xml application/xml
application/xml+rss text/javascript application/javascript;    gzip_disable        "msie6";
  types_hash_max_size 2048;
  client_max_body_size {{client_max_body_size}};
  proxy_buffer_size   64k;
  proxy_buffers   4 64k;
  proxy_busy_buffers_size   64k;
  server_names_hash_bucket_size 64;
  include /etc/nginx/conf.d/*.conf;
  include /etc/nginx/sites-enabled/*;
}
```

Можно заметить, что в данном конфигурационном файле мы начинаем использовать переменные. Переменных могло быть и больше, но для демонстрации их использования, вполне, достаточно. Сами переменные заключены в двойные фигурные скобки и были нами определены в файле инвентари в данном случае (переменная `nginx_user`, будет определена в настройках плейбука инвентари, будет зависеть от установленной системы).
Модифицируем наш инвентари и добавим второй узел и переменные среды для `node1`:

```
[node]
node1 ansible_host=172.17.5.5 nginx_user=www-data
```

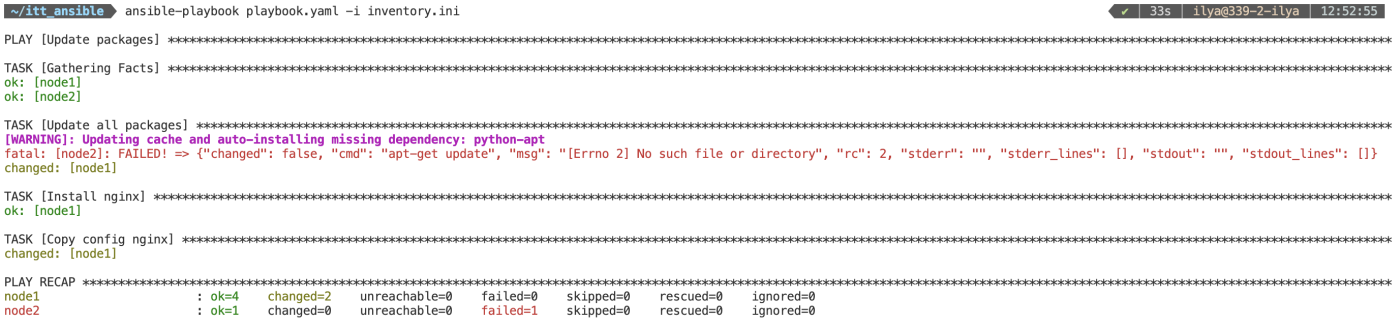
```
node2 ansible_host=172.17.5.6 nginx_user=root


[all:vars]
ansible_user    =    cloudadmin
```

Выполним плейбук

```
ansible-playbook playbook.yaml -i inventory.ini
```

Во время, выполнения мы можем увидеть ошибку, связанную с тем, что centos использует yum, а не apt, как и все redhat семейство:



Для того, чтобы сделать плейбук более универсальным, нам необходимо добавить условия и сделать установку для семейства redhat и тогда плейбук, будет выглядеть так:

```
---
- name: Update packages
  hosts: node
  become: yes
  tasks:
  - name: Install nginx "Debian"
    ansible.builtin.apt:
      package:
        - nginx
      state: present
      update_cache: yes
    when: ansible_facts['os_family'] == "Debian"
  - name: Install nginx "redhat"
    ansible.builtin.yum:
      name:
        - nginx
      state: present
    when: ansible_facts['os_family'] == "RedHat"
  - name: Copy config nginx
```

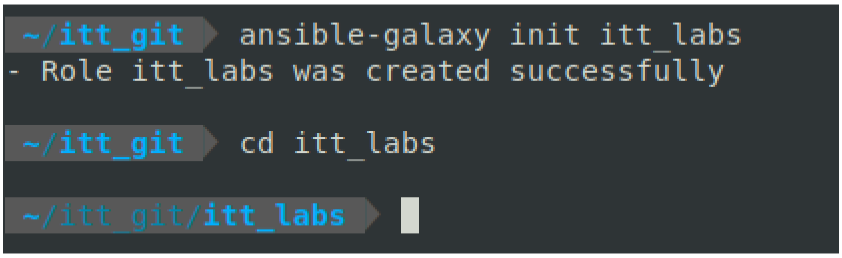
```
      ansible.builtin.template:
        src:  nginx.conf.j2
        dest: /etc/nginx/nginx.conf
```

Мы успешно освоили навык написания простых Ansible плейбуков, что позволяет нам автоматизировать рутинные операции и упрощает управление инфраструктурой, повышая эффективность и надежность работы.

6. Написание ролей

Для написания ролей в первую очередь можно создать роль с помощью `ansible galaxy`

```
ansible-galaxy init itt_labs
```



Посмотрим на созданную нами роль:

```
itt_labs/
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   └── main.yml
```

```
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml

8 directories, 8 files
```

Данные папки в Ansible служат для структурирования и организации кода, обеспечивая четкое разделение между различными типами ресурсов и функциональными блоками в проекте автоматизации. Вот краткое описание каждой из папок:

- `defaults`: Эта папка содержит файлы YAML, которые определяют значения по умолчанию для переменных, используемых в роли. Переменные, определенные здесь, могут быть переопределены пользователями роли.
- `files`: В этой папке хранятся файлы, которые не требуют обработки шаблонами и должны быть просто копированы на целевые хосты.
- `handlers`: Здесь располагаются файлы YAML, содержащие определения обработчиков. Обработчики используются для выполнения действий в ответ на определенные события, например, перезапуск сервисов после изменения конфигурации.
- `meta`: Папка, содержащая файл YAML с метаданными о роли. Метаданные включают автора, зависимости, лицензию и другую информацию о роли.
- `tasks`: В этой папке находятся файлы YAML, определяющие задачи, которые должна выполнить роль. Здесь содержатся инструкции по управлению конфигурацией системы.
- `templates`: Папка, содержащая шаблоны файлов, которые могут быть обработаны Jinja2 для динамической генерации конфигурационных файлов или других текстовых файлов.
- `tests`: В этой папке могут располагаться файлы для тестирования роли, такие как инвентарные файлы для запуска тестов и сценарии тестирования.
- `vars`: Здесь хранятся файлы YAML, содержащие объявления переменных, используемых в роли. Эти переменные могут быть использованы в задачах и шаблонах.
- `README.md`: Файл с описанием роли, ее назначением, используемыми переменными и примерами использования.

Перейдите в директорию проекта и инициализируйте в нем репозиторий и при каждом шаге модификации, делайте коммит.

Начнем с заполнения самого простого для понимания пункта, это метаданные, там мы описываем роль, указываем теги и указываем зависимости если такие существует, он находится на пути `meta/main.yml`:

```
galaxy_info:
  author: student
  description: A simple role for ansible learning
  company: ITT Department
  license: BSD-3-Clause
  min_ansible_version: "2.1"

  galaxy_tags: []

dependencies: []
```

В нашем курсе мы не рассматриваем тестирование ролей, поэтому удалим папку `tests`

```
rm -rf tests
```

Перенесем существующий плейбук в `tasks/main.yml` приводя его к виду:

```
---
# tasks file for itt_labs
- name: Install and configure nginx
  hosts: node
  become: true
  tasks:
  block:
    - name: Install nginx "Debian"
      ansible.builtin.apt:
        package:
          - nginx
        state: present
        update_cache: true
      when: ansible_facts['os_family'] == "Debian"
    - name: Install nginx "redhat"
      ansible.builtin.yum:
        name:
          - nginx
        state: present
      when: ansible_facts['os_family'] == "RedHat"
    - name: Copy config nginx
      ansible.builtin.template:
        src: nginx.conf
```

```
        dest: /etc/nginx/nginx.conf
        owner: root
        group: root
        mode: '0644'
```

Создадим файл `nginx.conf.j2` в директории `templates`:

```
user  {{ nginx_user }};
worker_processes  2048;
worker_priority     -1;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;

events {
  worker_connections {{worker_connections}};
}

http {
  include       /etc/nginx/mime.types;
  default_type  application/octet-stream;
  log_format  main  '$remote_addr - $remote_user [$time_local] "$request" ' '$status $body_bytes_sent
"$http_referer" ' '"$http_user_agent" "$http_x_forwarded_for"';
  access_log  /var/log/nginx/access.log  main;
  sendfile on;
  tcp_nopush on;
  tcp_nodelay on;
  keepalive_timeout  65;
  reset_timedout_connection  on;
  client_body_timeout        35;
  send_timeout               30;
  gzip on;
  gzip_min_length     {{gzip_min_length}};
  gzip_vary on;
  gzip_proxied        expired no-cache no-store private auth;
  gzip_types          text/plain text/css application/json application/x-javascript text/xml application/xml
application/xml+rss text/javascript application/javascript;    gzip_disable        "msie6";
  types_hash_max_size 2048;
  client_max_body_size {{client_max_body_size}};
  proxy_buffer_size   64k;
```

```
  proxy_buffers   4 64k;
  proxy_busy_buffers_size   64k;
  server_names_hash_bucket_size 64;
  include /etc/nginx/conf.d/*.conf;
  include /etc/nginx/sites-enabled/*;
}
```

Количество переменных изменилось, и теперь нам можно добавить новые переменные для роли, они будут находится по пути `vars/main.yml`

```
---
# vars file for itt_labs
worker_connections: 512
gzip_min_length: 1000
client_max_body_size: 512m
```

Также необходимо создать хендлеры которые будут работать с сервисом во время изменения его конфигураций, так создадим два хендлера для перезапуска и добавление в автозапуск демона(`handlers/main.yml`):

```
---
# handlers file for itt_labs
- name: Enable nginx service
  ansible.builtin.systemd_service:
    name: nginx.service
    state: restarted
    enabled: true

- name: Reload nginx service
  ansible.builtin.systemd_service:
    name: nginx.service
    state: restarted
```

И для того, чтобы эти хендлеры вызывались, нам необходимо явно прописать у задач `notify` после которых будет необходимо перезапускать или добавлять в автозапуск `nginx`, так общий файл с задачами примет вид:

```
---
# tasks file for itt_labs
- name: Install and configure nginx
```

```
hosts: node
become: true
tasks:
block:
  - name: Install nginx "Debian"
    ansible.builtin.apt:
      package:
        - nginx
      state: present
      update_cache: true
    when: ansible_facts['os_family'] == "Debian"
    notify: Enable nginx service
  - name: Install nginx "redhat"
    ansible.builtin.yum:
      name:
        - nginx
      state: present
    when: ansible_facts['os_family'] == "RedHat"
    notify: Enable nginx service
  - name: Copy config nginx
    ansible.builtin.template:
      src: nginx.conf
      dest: /etc/nginx/nginx.conf
      owner: root
      group: root
      mode: '0644'
    notify: Reload nginx service
```

Задание для самостоятельного выполнения

1. Напишите роль для автоматизации развертывания проекта
2. Создайте репозиторий и закоммитьте роль

Версия #19
Тарабанов Илья Федорович создал 15 января 2024 12:07:49
Тарабанов Илья Федорович обновил 16 мая 2024 19:16:27