

Аббревиатуры

IaC (Инфраструктура как код) — это подход для управления и описания инфраструктуры ЦОД через конфигурационные файлы, а не через ручное редактирование конфигураций на серверах или интерактивное взаимодействие.

IaaS. Вычислительные мощности для запуска своих решений и развертывания IT-инфраструктуры компании.

PaaS. Позволяет использовать уже готовые и настроенные платформы для специализированных задач.

SaaS. Модель, когда в облаках размещены готовые сервисы для конкретных прикладных функций — отправки почты, ведения базы клиентов, создания сайта.

1. Введение в Ansible

"Инфраструктура как код" (IaC) — это метод управления информационно-технологической инфраструктурой, который использует принципы разработки в рамках DevOps. Этот подход применяется для управления ресурсами облачной инфраструктуры, такими как виртуальные машины, сети, балансировщики нагрузки, базы данных и другие сетевые приложения.

IaC-обработка — это способ управления конфигурацией, при котором инфраструктурные ресурсы организации зашифровываются в текстовых файлах. Затем эти файлы инфраструктуры помещаются в систему контроля версий, например Git. Репозиторий системы контроля версий позволяет использовать рабочие процессы с применением функциональных веток и запросов pull, [лежащие в основе CI/CD](#).

Традиционно существует два подхода к IaC, декларативный или императивный, и два возможных метода, push и pull. Декларативный подход описывает конечную цель и определяет требуемое состояние ваших ресурсов. Императивный подход отвечает на вопрос о том, **как** нужно изменить инфраструктуру для достижения конкретной цели, обычно выдавая последовательность различных команд. [Ansible playbooks](#) — отличный пример императивного подхода. Разница между методом push и pull заключается в том, каким образом серверам сообщается информация о конфигурации. В pull режиме каждый сервер будет пулить свою конфигурацию из мастер-сервера, а в push режиме мастер-сервер сам распушивает конфигурацию по целевой системе.

Наибольшие преимущества для команд, использующих IaC, можно объединить в несколько ключевых характеристик:

- **Скорость и уменьшение затрат:** IaC позволяет быстрее конфигурировать инфраструктуру и направлен на обеспечение прозрачности, чтобы помочь другим командам со всего предприятия работать быстрее и эффективнее.
- **Масштабируемость и стандартизация:** IaC предоставляет стабильные среды быстро и на должном уровне. Командам разработчиков не нужно прибегать к ручной настройке - они обеспечивают корректность, описывая с помощью кода требуемое состояние сред. Развертывания инфраструктуры с помощью IaC повторяемы и предотвращают проблемы во время выполнения, вызванных дрейфом конфигурации или отсутствием зависимостей. IaC полностью стандартизирует сетап инфраструктуры, что снижает вероятность ошибок или отклонений.
- **Документация:** IaC также служит некой формой документации о правильном способе создания инфраструктуры и страховкой на случай, если сотрудники покинут вашу компанию, обладая важной информацией. Поскольку код можно версионировать, IaC позволяет документировать, регистрировать и отслеживать каждое изменение конфигурации вашего сервера.
- **Восстановление в аварийных ситуациях:** IaC — чрезвычайно эффективный способ отслеживания вашей инфраструктуры и повторного развертывания последнего работоспособного состояния после сбоя или катастрофы любого рода.

2. Установка и настройка Ansible

Перед написанием playbooks потребуются следующие компоненты:

- **Один узел управления Ansible:** узел управления Ansible — это система, которую мы будем использовать для подключения к хостам Ansible через SSH и управления этими хостами. В качестве узла управления Ansible может выступать локальный компьютер или специально выделенный для Ansible сервер.
- **Один или несколько хостов Ansible:** хост Ansible — это любой компьютер, для автоматизации которого настроен узел управления Ansible. Убедитесь, что каждый хост Ansible соответствует следующим требованиям:
Открытый ключ SSH узла управления Ansible добавлен в файл `authorized_keys` пользователя системы.

2.1 Установка Ansible

Установите Ansible в зависимости от вашей операционной системы, подробнее можно узнать об установке в [документации](#)

Здесь мы будем рассматривать установку на **Centos OS**.

Сначала обновите установленные пакеты системы с помощью следующей команды:

```
sudo apt update
```

После этого обновления вы можете установить программное обеспечение Ansible следующим образом:

```
sudo apt install ansible -y
```

Программа успешно установилась и можно приступать к дальнейшему обучению.

2.2 Настройка файла инвентаризации

Файл инвентаризации содержит информацию о хостах, которыми вы будете управлять с помощью Ansible. Вы можете включить в файл инвентаризации от одного до нескольких сотен серверов и распределить хосты по группам и подгруппам. Файл инвентаризации также часто задает переменные, действующие только для определенных хостов или групп, чтобы их можно было использовать с playbook и шаблонами.

Чтобы изменить заданное по умолчанию содержание файла инвентаризации Ansible, откройте файл `/etc/ansible/hosts` в предпочитаемом редакторе на узле управления Ansible:

```
sudo vim /etc/ansible/hosts
```



Tip

Ansible обычно создает файл инвентаризации по умолчанию в `etc/ansible/hosts`, вы можете создавать файлы инвентаризации в любом месте, которое соответствует вашим потребностям. В нашем случае вам нужно будет предоставить путь к вашему настраиваемому файлу инвентаризации с помощью параметра `-i` при запуске команд Ansible и плейбуков. Использование файлов инвентаризации на каждом проекте поможет минимизировать риск запуска плейбука на несоответствующей группе серверов.

Пример файла инвентаризации с описанием хост-серверов и переменных:

```
[servers]
server1 ansible_host=192.168.55.231

[all:vars]
ansible_python_interpreter=/usr/bin/python3
```

Если вы захотите проверить корректности файла инвентаризации, вы можете запустить команду:

```
ansible-inventory --list -y
```

Результат вывода команды:

```
all:
  children:
    servers:
      hosts:
        server1:
          ansible_host: 192.168.55.231
          ansible_python_interpreter: /usr/bin/python3
    ungrouped: {}
```

2.2.1 Объединение хостов в группы и подгруппы

В файле инвентаризации вы можете распределить серверы по различным группам и подгруппам. Это не только помогает поддерживать порядок на ваших хостах, но и позволяет использовать групповые переменные.

Один хост может входить в несколько групп. Следующий файл инвентаря демонстрирует настройку с двумя группами: webservers, dbservers. Обратите внимание, что серверы сгруппированы по двум разным характеристикам: их назначение (интернет и база данных) и среда их использования (разработка и производство).

```
[webservers]
203.0.113.111
203.0.113.112
[dbservers]
203.0.113.113
server_hostname
```

В Ansible есть встроенные группы, например такие как all, ungrouped. Первая включает в себя в хосты указанные в файле инвентаризации, а вторая - хосты, которые не находятся ни в одной группе.

Ansible позволяет назначать группы другим группам, используя синтаксис [groupname:children] в инвентаре. Это дает вам возможность назначить определенные группы членами других групп. Дочерние группы имеют возможность переопределять переменные, установленные родительскими группами.

Ниже приведен пример использования дочерних групп:

```
[ubuntu]
host1

[debian]
host2

[linux:children]
ubuntu
debian
```

2.2.2 Настройка переменных хоста

Файл инвентаря можно использовать для настройки переменных, которые изменят поведение Ansible по умолчанию при подключении и выполнении команд на ваших нодах. Фактически мы сделали это на предыдущем шаге, при настройке псевдонимов хостов. По сути переменная ansible_host сообщает Ansible, где найти удаленные ноды, если для ссылки на этот сервер используется псевдоним.

Переменные инвентаря можно установить для любого хоста или для любой группы. Помимо переопределения параметров Ansible по умолчанию, эти переменные также позволяют выполнять дальнейшую настройку для отдельных хостов и групп в playbooks.

В следующем примере показано, как определить удаленного пользователя по умолчанию при подключении к каждой ноде, перечисленной в этом инвентаре:

```
server1 ansible_host=203.0.113.111 ansible_user=user server2 ansible_host=203.0.113.112
ansible_user=root  server3 ansible_host=203.0.113.113 ansible_user=user server4
ansible_host=server_hostname ansible_user=root
```

В следующей таблице можно найти несколько общих шаблонов, которые вы можете использовать при запуске команд и playbook Ansible:

Шаблон	цель
all	Все хосты из указанного инвентаря
host1	Единственный хост (host1)
host1:host2	Оба хоста, host1 и host2
group1	Единственная группа (group1)
group1:group2	Все серверы в группах group1 и group2
group1:&group2	Серверы, которые состоят и в группе group1, и в group2
group1:!group2	Серверы, которые состоят только в group1, исключая те, что состоят одновременно в группе group2

2.3 Тестирование подключения к хостовым серверам

После настройки файла инвентаризации для вашего сервера, вы можете проверить способность Ansible подключаться к этим серверам и запускать команды через SSH.

Обычно используется учетная запись **root** в Ubuntu, поскольку чаще всего это единственная учетная запись, которая доступна по умолчанию на новых серверах. Если на ваших хостах Ansible уже имеются учетные записи sudo, вам рекомендуется использовать эту учетную запись.

Вы можете использовать аргумент `-u`, чтобы задать пользователя удаленной системы. Если не указано иное, Ansible попытается подключиться от имени текущего пользователя системы на узле управления.

```
ansible all -m ping -u timofey
```

Перед этим обязательно добавить публичный ssh ключ узла управления в файл `~/.ssh/authorized_keys` на хостовых машинах

В случае удачного тестирования мы получим такое сообщение:

```
[timofey@linux-os-ansible-adm:/etc/ansible$ ansible all -m ping -u timofey
server1 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
```

В случае фатального тестирования мы получим такое сообщение:

```
timofey@linux-os-ansible-adm:~/.ssh$ ansible all -m ping -u root
server1 | UNREACHABLE! => {
  "changed": false,
  "msg": "Failed to connect to the host via ssh: root@192.168.55.231: Permission denied (publickey,password).",
  "unreachable": true
}
```

Когда вы получите от хоста ответ `"pong"`, это будет означать, что вы готовы запускать команды и playbook Ansible на этом сервере.

2.4 Запуск ситуативных команд (опционально)

Убедившись, что узел управления Ansible может взаимодействовать с хостами, вы можете запускать на серверах ситуативные команды и плейбуки.

Пример простой команды, которая выведет использование дисковых ресурсов на всех серверах:


```
ansible all -a "df -h" -u root
```

```
[timofey@linux-os-ansible-adm:~/.ssh$ ansible all -a "df -h" -u timofey
server1 | CHANGED | rc=0 >>
Filesystem      Size  Used Avail Use% Mounted on
udev            1.9G     0   1.9G   0% /dev
tmpfs           392M  520K   392M   1% /run
/dev/vda1       62G   3.5G   56G    6% /
tmpfs           2.0G     0   2.0G   0% /dev/shm
tmpfs           5.0M     0   5.0M   0% /run/lock
tmpfs           392M     0   392M   0% /run/user/1000
```

Также можно указать несколько хостов указать их через ":":

```
ansible server1:server2 -m ping -u root
```


3. Playbook для Ansible

Для начала необходимо создать файл `playbook.yaml` в директории `/etc/ansible/`

'---' - начало playbook

'... ' - конец playbook

'#' - можно сделать комментарий к записи

 **Tip**

Каждый play обязательно содержит - `hosts: "all/..."`, где указывается на какие сервера будут выполнены данные tasks

play - набор действий, которые будут выполнены

task - отдельное атомарное действие

Ниже показан пример написания простого playbook:

```
-
  name: testing connection between devices
  hosts: all
  tasks:
    - name: task1
      ping:
```

Для запуска playbook можно использовать команду:

```
ansible-playbook playbook.yaml
```

Флаги `-v/-vv/-vvv` позволяют увидеть более подробную информацию о выполнении выше указанной команды.

4. Переменные

Переменные в Ansible представляют собой динамические значения, используемые в playbook и ролях для конфигурации и повторного использования настроек. Они аналогичны переменным в программировании и обеспечивают эффективное управление сложными задачами. Переменные позволяют применять один и тот же сценарий или роль в различных средах, системах или контекстах, избегая необходимости жесткого кодирования конкретной информации.

Переменные можно объявлять в инвентарном файле, playbook и в отдельном файле с переменными.

Пример объявления переменных в инвентарном файле:

```
[variables]
server.example.com
ansible_connection=ssh/winrm/local/docker
ansible_user=root
ansible_port=22/5986
ansible_ssh_pass=password
ansible_password=password
proxy=proxy.server.com
ansible_become=yes
```

Переменные для групп:

```
[ubunta]
host1
host2

[centos]
host2
host3

[linux:children]
ubunta
centos

[linux:vars]
some_server=server.com
dw_timeout=30
```

Хотя вы можете хранить переменные в основном файле инвентаризации, хранение отдельных файлов хост- и групповых переменных может помочь вам легче организовать значения переменных. Вы также можете использовать списки и хэш-данные в файлах хоста и групповых переменных, что вы не можете сделать в основном файле инвентаризации.

Файлы переменных хоста и группы должны использовать синтаксис YAML. Допустимые расширения файлов включают '.yml', '.yaml', '.json' или без расширения файла.

чтобы передавать переменные отдельным файлом необходимо создать файл и затем включить этот файл в `playbook`:

```
---
- name: My Playbook
  hosts: your_target_hosts
  vars_files:
    - path/to/my_vars.yml

  tasks:
    - name: Your task
      debug:
        msg: "Variable 1: {{ my_variable1 }}, Variable 2: {{ my_variable2 }}"
```

Помимо этого можно передавать файл групповых переменных.

Если у вас есть группа хостов в вашей директории, где находится инвентарный файл, создайте каталог `group_vars` , и внутри него создайте файл с именем группы хостов (например, `web_servers.yml`):

```
---
- name: My Playbook
  hosts: web_servers  # Имя вашей группы хостов
```

```
tasks:
  - name: Your task
    debug:
      msg: "Variable 1: {{ my_variable1 }}, Variable 2: {{ my_variable2 }}"
```

5. Модули в Ansible

Модули – это дискретные единицы кода, которые можно запускать с помощью командной строки или с помощью playbook для того, чтобы вносить определенные изменения в целевой узел или собирать с него информацию. Ansible реализует каждый модуль на удалённом целевом узле, а также собирает ответные значения. Модули Ansible также известны как плагины задач или библиотечные плагины.

Модули в playbook пишутся по принципу идемпотентности, это означает, что **модуль можно выполнять сколько угодно раз, но при этом модуль будет выполнять изменения, только если система не находится в желаемом состоянии.**

Важно!

Не стоит в написании playbook использовать shell, поскольку в Ansible почти на все случаи есть уже реализованные модули, которые способны сделать необходимые задачи с большей точностью и без повторений.

Например, если нам необходимо запустить сервис nginx, не нужно писать shell скрипт, который бы это делал, лучшим выбором будет пример кода ниже:

```
- name: nginx systemd
  systemd:
    name: nginx
    enabled: yes
    state: started
```

Где мы фактически передаем желаемый результат для сервиса nginx, то есть "started". Если служба уже запущена, ничего не произойдет, а если нет, то Ansible ее запустит.

Модуль Setup

Модуль Setup помогает нам в сборе данных о целевых узлах, таких как hostname, ip, os и других. Этот модуль автоматически реализуется playbook для сбора полезной информации об удаленном хосте. Вся информация представлена в удобном формате JSON, а всем значениям предшествует “ansible_”.

```
ansible all -m setup -u timofey
```

Краткий вывод команды:

```
timofey@linux-os-ansible-adm:~$ ansible all -m setup -u timofey
[server2 | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "192.168.55.232",
            "172.17.0.1"
        ],
        "ansible_all_ipv6_addresses": [
            "fdf0:8cab:fd:0:78a6:55ff:fe93:78b8",
            "fe80::78a6:55ff:fe93:78b8"
        ],
        "ansible_apparmor": {
            "status": "enabled"
        },
        "ansible_architecture": "x86_64",
        "ansible_bios_date": "04/01/2014",
        "ansible_bios_vendor": "SeaBIOS",
        "ansible_bios_version": "rel-1.14.0-0-g155821a1990b-prebuilt.qemu.org",
        "ansible_board_asset_tag": "NA",
        "ansible_board_name": "NA",
        "ansible_board_serial": "NA",
        "ansible_board_vendor": "NA",
        "ansible_board_version": "NA",
        "ansible_chassis_asset_tag": "NA",
        "ansible_chassis_serial": "NA",
        "ansible_chassis_vendor": "QEMU",
        "ansible_chassis_version": "pc-i440fx-6.1",
        "ansible_cmdline": {
            "BOOT_IMAGE": "/boot/vmlinuz-5.10.0-23-amd64",
            "quiet": true,
            "ro": true,
            "root": "UUID=d840efa8-879d-440c-9bfd-5063900ff9b1"
        },
        "ansible_date_time": {
            "date": "2022-08-01",
            "day": "01",
            "epoch": "1659340800",
            "hour": "12",
            "iso8601": "2022-08-01T12:00:00",
            "iso8601_micro": "2022-08-01T12:00:00.000000",
            "microseconds": "0",
            "month": "08",
            "seconds": "00",
            "time": "12:00:00",
            "timezone": "UTC",
            "weekday": "Monday",
            "year": "2022"
        },
        "ansible_devicetree": {
            "bootargs": "root=UUID=d840efa8-879d-440c-9bfd-5063900ff9b1 ro console=ttyS0,0x3f8 console=ttyL0,0x3f01000 no_timer_check loglevel=7"
        },
        "ansible_distribution": "Ubuntu",
        "ansible_distribution_file_path": "/etc/os-release",
        "ansible_distribution_file_variety": "Ubuntu",
        "ansible_distribution_major_version": "22.04",
        "ansible_distribution_release": "jammy",
        "ansible_distribution_version": "22.04.1 LTS",
        "ansible_domain": "",
        "ansible_hostname": "server2",
        "ansible_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_ip4_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_ip6_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_lsb_release": {
            "description": "Ubuntu 22.04.1 LTS",
            "id": "Ubuntu",
            "pretty": "Ubuntu 22.04.1 LTS",
            "release": "22.04",
            "variant": "Ubuntu",
            "variant_id": "Ubuntu"
        },
        "ansible_machine_id": "d840efa8-879d-440c-9bfd-5063900ff9b1",
        "ansible_machine_vendor": "QEMU",
        "ansible_machine_version": "6.1.0",
        "ansible_memory_mb": {
            "available": "1024",
            "free": "1024",
            "total": "1024"
        },
        "ansible_mounts": [
            {
                "device": "lo",
                "filesystem": "tmpfs",
                "mount_options": "rw,relatime",
                "mount_point": "/",
                "size": "1024M",
                "type": "local"
            },
            {
                "device": "eth0",
                "filesystem": "xfs",
                "mount_options": "rw,relatime",
                "mount_point": "/dev/vda1",
                "size": "1024M",
                "type": "remote"
            }
        ],
        "ansible_net_hostname": "server2",
        "ansible_net_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_ipv4_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_ipv6_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l2_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l3_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l4_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l5_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l6_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l7_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l8_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l9_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l10_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l11_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l12_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l13_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l14_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l15_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l16_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l17_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l18_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l19_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l20_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l21_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l22_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l23_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l24_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l25_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l26_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l27_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l28_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l29_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l30_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l31_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l32_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l33_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l34_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l35_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l36_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l37_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l38_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l39_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l40_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l41_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l42_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l43_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l44_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l45_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l46_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l47_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l48_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l49_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l50_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l51_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l52_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l53_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l54_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l55_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l56_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l57_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l58_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l59_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l60_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l61_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l62_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l63_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l64_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l65_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l66_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l67_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l68_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l69_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l70_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l71_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l72_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l73_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l74_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l75_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l76_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l77_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l78_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l79_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l80_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l81_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l82_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l83_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l84_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l85_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l86_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l87_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l88_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l89_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l90_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l91_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l92_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l93_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l94_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l95_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l96_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l97_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l98_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l99_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l100_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l101_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l102_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l103_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l104_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l105_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l106_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l107_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l108_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l109_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l110_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l111_interfaces": [
            "eth0",
            "lo"
        ],
        "ansible_net_l112_interfaces": [
            "eth0
```

Модуль Ping

Модуль Ping используется при проверке на удаленном хосте, особенно при подключении к удаленному хосту, и проверке Python на нём. Он также может использоваться для проверки логина пользователя и доступов.

```
---
- name: ping module
  hosts: all
  become: false
  tasks:
    - name: test connection
      ping:
```

Модуль apt

Модуль apt package используется для управления ПО в ОС Linux на базе debian.

```

----
- name: Update Package on debian system
  hosts: all
  become: yes
  tasks:
    - name: Run the equivalent of "apt-get update" as a separate step
      apt:
        update_cache: yes
    - name: Install latest version of "nginx"

```



```
apt:
  name: nginx
  state: latest
  install_recommends: no
```

Модуль package

Этот модуль управляет пакетами на целевом объекте без указания модуля диспетчера пакетов (yum, apt, ...). Его удобно использовать в гетерогенной среде без необходимости создавать конкретную задачу под каждого менеджера пакетов. Этот модуль выступает в качестве прокси-сервера для базового модуля диспетчера пакетов, тогда как все параметры будут переданы базовому модулю:

```
---
- name: Update Packages on target system
  hosts: all
  become: yes
  tasks:
    - name: Install ntpdate
      ansible.builtin.package:
        name: ntpdate
        state: present

    - name: Install the latest version of Apache and MariaDB
      ansible.builtin.package:
        name:
          - httpd
          - mariadb-server
        state: latest
```

Модуль copy

Модуль copy может использоваться для копирования файла с локального/удаленного устройства и передаваться на другое устройство. Для компьютера с Windows можно использовать модуль win_copy.

```
---
- name: copy module
  hosts: localhost
  tasks:
    - name: copy a file from local machine to local machine
      copy:
        src: files/src.txt
        dest: files/dest.txt

---
- name: copy module
  hosts: all
  tasks:
    - name: copy a file from remote machine to remote machine
      copy:
        src: /etc/src.txt
        dest: /etc/dest.txt
```

Модуль fetch

Модуль fetch можно использовать каждый раз, когда мы хотим передать файл с удаленного устройства на локальный. Файлы хранятся на локальном устройстве в озаглавленном каталоге, соответствующем имени хоста.

```
---
- name: fetch module
  hosts: all
  tasks:
  - name: copy a file from remote machine to local machine
    fetch:
      src: /var/log/access.log
      dest: /var/log/fetched
```

Модуль file

Модуль file отвечает за выполнение таких задач, как создание файлов и каталогов, их удаление, создание мягких и жестких символических ссылок, добавление и изменение разрешений для файлов и каталогов и многое другое. Для компьютера с Windows можно использовать модуль win_file.

```
---
- name: file module
  hosts: all
  tasks:
  - name: Create a file
    file:
      path: /etc/foo.conf
      state: touch
      mode: u=rw,g=r,o=r
  - name: Create a directory if it does not exist
    file:
      path: /etc/some_directory
      state: directory
      mode: '0755'
  - name: Remove file (delete file)
    file:
      path: /etc/foo.txt
      state: absent
  - name: Change file ownership, group and permissions
    file:
      path: /etc/foo.conf
      owner: foo
      group: foo
      mode: '0644'
  - name: Create a symbolic link
    file:
      src: /file/to/link/to
      dest: /path/to/symlink
      owner: foo
      group: foo
      state: link
```

Модуль find

Модуль find функционирует так же, как и команда поиска Linux, помогая находить файлы и каталоги на основе различных критериев поиска. Для Windows вместо этого следует использовать модуль win_find.

```
---
- name: find module
  hosts: all
  tasks:
  - name: Recursively find /tmp files older than 2 days
```

```
find:
  paths: /tmp
  age: 2d
  recurse: yes
```

Модуль lineinfile

Модуль lineinfile незаменим в тех случаях, когда вы хотите добавить, удалить или изменить одну строку в файле. Можно также использовать режим сопоставления строки перед изменением или удалением при помощи регулярных выражений. Вы можете повторно использовать и изменять строку, используя параметр обратной ссылки. Также можно использовать свойства insertafter и insertbefore для внесения изменений в указанную часть файла.

```
---
- name: lineinfile module
  hosts: all
  tasks:
  - name: adding a line
    lineinfile:
      path: /etc/selinux/config
      regexp: '^SELINUX='
      line: SELINUX=enforcing
```

При необходимости информацию можно также найти в официальной [документации](#)

6. Роли и сценарии Ansible

Роль в Ansible представляет собой набор задач или обработчиков переменных, файлов и других артефактов, которые объединены и подключаются к плейбуку как единое целое. Обычно роли предназначены для выполнения высокоуровневых задач, таких как установка баз данных, настройка веб-серверов и других сложных сервисов. Однако роли также могут использоваться для автоматизации низкоуровневых сервисов, не встроенных непосредственно в Ansible.

С ростом функциональности в ваших плейбуках они могут становиться сложными и трудными в обслуживании. Роли приходят на помощь, разбивая сложные плейбуки на отдельные, более мелкие фрагменты. Эти фрагменты могут быть легко повторно использованы и координированы центральной точкой входа.

Основная идея ролей заключается в том, чтобы обеспечить повторное использование общих шагов настройки между разными типами серверов. Это позволяет значительно упростить структуру ваших конфигураций, делая их более модульными и поддерживаемыми.

Роль в Ansible включает в себя определенные каталоги и файлы, которые предоставляют четкую структуру для организации задач, переменных, файлов шаблонов и других ресурсов. Вот основные компоненты роли:

- Каталог defaults :**
Файл `main.yml` в каталоге `defaults` содержит значения по умолчанию для переменных роли.
- Каталог vars :**
Файлы с переменными, которые могут быть переопределены при необходимости.
- Каталог tasks :**
Файл `main.yml` в каталоге `tasks` содержит задачи, которые будут выполнены при использовании роли.
- Каталог templates :**
Шаблоны Jinja2, используемые для создания конфигурационных файлов на целевых хостах.
- Каталог meta :**
Файл `main.yml` в каталоге `meta` содержит информацию о зависимостях роли и другие метаданные.
- Каталог handlers :**
Файл `main.yml` в каталоге `handlers` содержит задачи для управления сервисами, перезапуска и т.д.

7. **Каталог tests :**

Тесты для роли.

8. **– **files** :** содержит статические файлы и файлы сценариев, которые могут быть скопированы на удалённый сервер или выполнены на нём.

6.1 Создание роли

1. Создайте роль с именем, например, `webserver` , воспользовавшись командой:

```
ansible-galaxy init webserver
```

Это создаст структуру каталогов роли, как я описывал ранее.

2. **Определите конфигурацию роли:**

Внесите необходимые настройки в файлы роли, такие как переменные (`defaults/main.yml`), задачи (`tasks/main.yml`), шаблоны (`templates/`), и так далее.

Например, в файле `defaults/main.yml` вы можете установить переменные по умолчанию для веб-сервера:

```
# defaults/main.yml
webserver_port: 80
webserver_document_root: "/var/www/html"
```

Создайте шаблон `templates/webserver.conf.j2`:

```
<VirtualHost *:{{ webserver_port }}>
    DocumentRoot {{ webserver_document_root }}
    # Дополнительные настройки
</VirtualHost>
```

В файле `tasks/main.yml` определите задачи для установки веб-сервера и его настройки:

```
# tasks/main.yml
- name: Install web server
  apt:
    name: apache2
    state: present

- name: Configure web server
  template:
    src: webserver.conf.j2
    dest: "/etc/apache2/sites-available/default"
  notify: restart apache
```

- name: Configure web server :** Это описание задачи, которое будет отображаться в выводе Ansible. Это просто информация для удобства чтения.
- template :** Это модуль Ansible, который используется для применения шаблонов к конфигурационным файлам.
- src: webserver.conf.j2 :** Указывает на исходный шаблон, который будет использоваться для генерации конфигурационного файла. Здесь `webserver.conf.j2` - это имя файла шаблона в каталоге `templates` вашей роли.
- dest: "/etc/apache2/sites-available/default" :** Указывает на путь, по которому будет сохранен созданный конфигурационный файл. В данном случае, конфигурационный файл будет сохранен по пути `/etc/apache2/sites-available/default`.
- notify: restart apache :** Уведомление (notification) обозначает, что после выполнения этой задачи Ansible должен выполнить определенный обработчик. В данном случае, после того как будет настроен

веб-сервер, будет вызван обработчик с именем `restart apache`. Обработчики обычно определены в каталоге `handlers` вашей роли.

В файле `handlers/main.yml` определите обработчик для перезапуска веб-сервера:

```
```.yaml
handlers/main.yml
- name: restart apache
 service:
 name: apache2
 state: restarted
```

Используйте роль в `playbook`. Создайте Ansible `playbook`, в котором используйте вашу роль `webserver`. Пример `playbook`:

```
site.yml
- name: Configure web servers
 hosts: webservers
 roles:
 - webserver
```

Запустите плейбук:

```
ansible-playbook -i inventory.ini site.yml
```

В этом примере роль `webserver` может быть повторно использована в других проектах, и изменения в конфигурации могут быть легко внесены, обновив только соответствующие файлы роли.

## 7. Ansible Galaxy

Ansible Galaxy - это веб-платформа и инструмент командной строки для обмена, поиска, оценки и использования ролей Ansible.

Вот некоторые ключевые аспекты Ansible Galaxy:

1. **Роль:**

- В Ansible Galaxy, роль представляет собой коллекцию плейбуков, переменных, файлов и других ресурсов, упакованных вместе для легкого использования и обмена. Роли могут быть созданы сообществом или предоставлены организациями для автоматизации конкретных задач.

2. **Коллекция:**

- Коллекция представляет собой новый способ упорядочивания и структурирования ролей, плейбуков, модулей и дополнений. Коллекции были введены для улучшения организации и повторного использования кода Ansible.

3. **Команда `ansible-galaxy`:**

- Ansible Galaxy поставляется с командой командной строки `ansible-galaxy`, которая предоставляет возможность взаимодействия с ролями и коллекциями. Например, вы можете использовать `ansible-galaxy install` для установки роли из Galaxy в ваш проект.

4. **Роли и Коллекции Ansible Galaxy:**

- Ansible Galaxy содержит множество общедоступных ролей и коллекций, созданных сообществом. Вы можете легко искать и устанавливать роли с помощью Ansible Galaxy. Это позволяет избежать написания кода с нуля и ускоряет процесс разработки.

5. **Качество и Обратная связь:**

- Роли и коллекции на Ansible Galaxy часто снабжаются описаниями, документацией, версиями и тегами, что облегчает выбор и использование. Пользователи также могут предоставлять обратную связь и оценки, помогая другим сообществам оценить качество и полезность ролей.

6. **Интеграция с Ansible Tower:**



- Ansible Tower, коммерческое решение для управления инфраструктурой на основе Ansible, обеспечивает интеграцию с Ansible Galaxy. Это позволяет управлять ролями и коллекциями, а также легко внедрять их в автоматизированные рабочие процессы.

Примеры использования Ansible Galaxy:

- **Установка роли:**

```
ansible-galaxy install username.role_name
```

- **Создание собственной роли:**

```
ansible-galaxy init my_role
```

- Поиск существующих ролей:

```
ansible-galaxy search "nginx"
```

Подробнее с существующими ролями можно ознакомиться на [Ansible galaxy](#).

Ansible Galaxy упрощает обмен и повторное использование кода Ansible в сообществе, способствуя более эффективному развертыванию и управлению инфраструктурой.

## 8. Ansible Vault

Ansible Vault - это инструмент в Ansible, предназначенный для шифрования файлов, содержащих конфиденциальные данные, такие как пароли, ключи и другие секреты. Использование Ansible Vault обеспечивает безопасное хранение и передачу конфиденциальной информации в автоматизированных конфигурационных задачах. Вот основные концепции и команды Ansible Vault:

### 1. Создание зашифрованного файла:

- Вы можете создать новый зашифрованный файл Ansible Vault с помощью следующей команды:

```
ansible-vault create filename.yml
```

- После выполнения этой команды Ansible предложит вам ввести и подтвердить пароль для зашифровки файла.

### 2. Редактирование зашифрованного файла:

- Для редактирования зашифрованного файла используйте команду:

```
ansible-vault edit filename.yml
```

- Это откроет файл в текстовом редакторе (обычно это будет VI), после чего вы сможете внести изменения.

### 3. Просмотр зашифрованного файла:

- Для просмотра содержимого зашифрованного файла используйте команду:

```
ansible-vault view filename.yml
```

### 4. Расшифровка файла:

- Если вам нужно расшифровать файл для внесения изменений, используйте команду:

```
ansible-vault decrypt filename.yml
```

- После этого Ansible запросит пароль, и файл будет расшифрован.

### 5. Шифрование существующего файла:

- Если у вас уже есть нешифрованный файл, вы можете зашифровать его с использованием команды:

```
ansible-vault encrypt filename.yml
```

## 6. Пароли:

- Ansible Vault использует пароли для шифрования и расшифровки файлов. Пароль может быть введен вручную при каждом использовании или сохранен в файле для автоматизации.

## 7. Использование Vault в Playbook:

- Для включения зашифрованных файлов в плейбуки, используйте ключ `--vault-id` при запуске playbook:

```
ansible-playbook --vault-id @prompt playbook.yml
```

### ⚠ Важно!

В примере выше необходимо создать папку `prompt` с паролем, в директории с `playbook`.

- Ansible также поддерживает различные методы предоставления пароля, такие как `--vault-password-file` или использование переменной окружения `ANSIBLE_VAULT_PASSWORD_FILE`.

## 8. Использование Vault в переменных:

- Можно также использовать Ansible Vault для шифрования переменных внутри playbook.

Для работы с файлами мы можем вводить парольную фразу с клавиатуры при каждом обращении к Vault или же использовать файл, в котором хранится парольная фраза. В первом случае в строку вызова добавляется ключ `--ask-vault-pass`, во втором добавляется параметр `--vault-password-file=<путь к файлу>`.

Далеко не всегда необходимо шифровать файл плейбука целиком. Иногда удобнее зашифровать значение только определенной переменной, например пароль, а все остальное оставить в открытом виде для того, чтобы другие специалисты также могли вносить свои правки при необходимости.

Для этого мы можем воспользоваться параметром `encrypt_string`. В примере ниже у нас имеется переменная `my_secret` значение которой неплохо бы скрыть от посторонних.

```
- name: inline secret variable demonstration
 hosts: all
 gather_facts: false
 vars:
 my_secret: my_very_long_password
 tasks:
 - name: print the secure variable
 debug:
 var: my_secret
```

Для этого используем следующую команду:

```
ansible-vault encrypt_string --vault-id @prompt my_very_long_password
```

Ansible Vault обеспечивает безопасное и гибкое управление секретами и конфиденциальной информацией, делая их доступными только тем, кто имеет доступ к соответствующим паролям или ключам.