Привет! Добро пожаловать в инструкцию по Ansible! Ansible — это мощный инструмент для автоматизации конфигурации и управления системами. С его помощью вы можете значительно упростить процессы развёртывания, настройки вашей инфрастурктуры и управления ей.

Ansible основан на принципе «инфраструктура как код» — это означает, что вы можете определить состояние вашей системы и управлять ей, используя язык разметки и конфигурации. Это позволяет автоматизировать множество задач, устранить рутинную ручную работу и обеспечить единообразие в настройках и конфигурации.

В этой инструкции вы познакомитесь с основными концепциями и возможностями Ansible, узнаете:

- как установить Ansible,
- настроить свой инвентарь,
- создавать плейбуки и роли,
- использовать переменные и шаблоны,
- применять автоматизацию к различным аспектам вашей инфраструктуры.

Эта инструкция поможет вам освоить основы Ansible и начать использовать его в своих проектах независимо от того, кем вы являетесь: администратором, разработчиком или просто увлекаетесь технологиями.

Содержание

1. Что такое Ansible
2. Установка Ansible
   2.1. Установка на Windows
   2.2. Установка на Linux
   2.3. Установка на macOS
3. Inventory file
   3.1. Как создать Inventory file
4. Ansible modules
5. Ansible playbooks
6. Ansible roles

1. Что такое Ansible

Ansible — это инструмент для автоматизации конфигурации и управления системами, который позволяет управлять инфраструктурой как кодом. Он предоставляет простую и эффективную платформу для развёртывания, настройки множества узлов и приложений и управления ими.

Архитектура Ansible основана на клиент-серверной модели, где сервером является машина управления (control machine), а клиентами — узлы (nodes), которые управляются с помощью Ansible. На машине управления установлен Ansible и настроен инвентарь (inventory), который содержит информацию о доступных узлах и их конфигурации.

Основные компоненты Ansible:

1. Машина управления (Control Machine).
   На ней установлен Ansible, она используется для написания и запуска автоматизированных задач.
2. Инвентарь (Inventory).
   Инвентарь содержит информацию об узлах, которыми управляет Ansible. Это может быть список IP-адресов или доменных имён узлов, а также группировка узлов для организации их по функциональности или ролям.
3. Модули (Modules).
   Модули — это программы, которые выполняют конкретные задачи на управляемых узлах. Ansible предоставляет широкий набор встроенных модулей, таких как управление пакетами, файлами, сервисами, пользователем и другими аспектами системы.
4. Плейбуки (Playbooks).
   Плейбуки — это файлы, написанные на языке разметки YAML и описывающие желаемое состояние системы и действия, которые нужно выполнить для достижения этого состояния. Плейбуки объединяют модули и задают порядок их выполнения.
5. Роли (Roles).
   Роли — это способ организации и повторного использования плейбуков и переменных. Они позволяют группировать задачи и файлы конфигурации в логические блоки, которые можно применять к различным узлам.
6. Переменные (Variables).
   Переменные используются для параметризации конфигурации и передачи данных между различными компонентами Ansible. Они позволяют создавать более гибкие и настраиваемые плейбуки.

Аналоги Ansible в мире автоматизации и управления системами включают:

1. Chef.
   Это популярный инструмент для управления конфигурацией, который также работает по принципу «инфраструктура как код». Chef предоставляет свой язык конфигурации и имеет богатую экосистему.
2. Puppet.
   Ещё один инструмент для автоматизации и управления конфигурацией, который широко используется в индустрии. Puppet также позволяет описывать системную конфигурацию в виде кода и управлять ей.
3. SaltStack.
   Это инструмент для автоматизации и конфигурации, который основан на модели «клиент — сервер» и использует язык Python для описания конфигурации.
4. PowerShell Desired State Configuration (DSC).

Это фреймворк автоматизации и управления конфигурацией от Microsoft. Он использует язык PowerShell для описания желаемого состояния системы.

2. Установка Ansible

Ниже вы можете ознакомиться с порядком действий для установки Ansible на Windows, Linux и macOS.

2.1. Установка Ansible на Windows

- Сначала убедитесь, что на вашей системе установлен Python версии 3.x. Если Python не установлен, загрузите и установите его с официального сайта Python.
- Откройте командную строку или PowerShell с правами администратора.
- Установите Ansible с помощью pip, выполнив команду pip install ansible.
- После установки Ansible может потребоваться добавление пути к исполняемому файлу в переменную среды PATH. Для этого:
    - Нажмите правой кнопкой мыши на значок «Мой компьютер» или «Этот компьютер» и выберите «Свойства».
    - Нажмите «Дополнительные параметры системы» и выберите «Переменные среды».
    - В разделе «Системные переменные» найдите переменную Path и нажмите «Изменить».
    - Добавьте путь к папке с исполняемым файлом Ansible (обычно C:\Python\Scripts) и сохраните изменения.
- Теперь Ansible должен быть успешно установлен в вашей системе Windows.

2.2. Установка Ansible на Linux

- Откройте терминал в вашей системе Linux.
- Убедитесь, что у вас установлен Python версии 3.x. Если Python не установлен, выполните команду sudo apt install python3.
- Установите Ansible, выполните следующую команду: sudo apt install ansible.
- Во время установки вам может потребоваться введение пароля администратора.
- После завершения установки Ansible будет доступен в вашей системе Linux.

2.3. Установка Ansible на macOS

- Откройте терминал в вашей системе macOS.
- Убедитесь, что у вас установлен Python версии 3.x. Если Python не установлен, вы можете установить его с помощью Homebrew: brew install python3.
- Установите Ansible с помощью pip, выполните команду pip3 install ansible.
- После завершения установки Ansible будет доступен в вашей системе macOS.

После успешной установки Ansible вы можете проверить его версию, запустив команду ansible --version в терминале или командной строке.

Теперь давайте перейдём к работе с Ansible и изучим его основные концепты.

3. Inventory file

Inventory file в Ansible является текстовым файлом, содержащим информацию о хостах, на которых будет выполняться конфигурация и управление с помощью Ansible. Он определяет группы хостов, переменные окружения и другие параметры, необходимые для выполнения задач.

Inventory file нужен:

1. Для определения целевых хостов. Inventory file указывает Ansible, с какими хостами он будет взаимодействовать. Можно определить хосты по их IP-адресам, доменным именам или другим идентификаторам.
2. Группировки хостов. Inventory file позволяет организовать хосты в группы. Группы могут быть использованы для применения задач и плейбуков к определённым наборам хостов — это облегчает управление и конфигурацию систем.
3. Определения переменных окружения. Inventory file может содержать переменные, которые будут использованы в плейбуках для настройки системы. Например, можно определить переменные, содержащие конфигурационные параметры, учётные данные или любую другую информацию, необходимую для задач.

3.1. Как создать Inventory file

Inventory file может быть создан вручную в текстовом редакторе. Его расширение может быть .ini или .yaml. В файле можно определить хосты и группы хостов, используя следующий синтаксис:

```
[group_name]
hostname ansible_host=IP_address ansible_user=username ansible_ssh_private_key_file=path/to/private_key
```

Здесь и далее — изображения Skillbox

Здесь:

- group_name — имя группы хостов;
- hostname — имя хоста или его IP-адрес;
- ansible_host — IP-адрес хоста;
- ansible_user — имя пользователя, под которым будет выполняться подключение к хосту;
- ansible_ssh_private_key_file — путь к приватному SSH-ключу, используемому для аутентификации.

Давайте попробуем!

1. У нас есть виртуальная машина, развёрнутая в облаке. На её примере мы всё разберём. На скриншоте ниже вы можете видеть обновлённый инвентори-файл.

Указаны:

- имя хоста;
- его IP;
- имя пользователя, под которым будет происходить подключение к виртуальной машине;
- полный путь к SSH-ключу.

```
> $ cat inventory.yml
[skb_ansible]
skb_test ansible_host=158.160.70.172 ansible_user=evgeny ansible_ssh_private_key_file=/Users/evgeny/.ssh/id_rsa
```

2. Теперь давайте запустим ansible и подробно разберём команду запуска:

```
> $ ansible skb_ansible -i inventory.yml -m ping
skb_test | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

2.1.

```
> $ ansible skb_ansible -i inventory.yml -m ping
```

Эта команда выполняет модуль ping на группе хостов skb_ansible, используя инвентарный файл inventory.yml.

- ansible — команда, которая запускает выполнение задач.
- skb_ansible — имя группы хостов, для которой будет применяться команда. В инвентарном файле должна быть определена группа с таким именем.
- -i inventory.yml — опция для указания пути к инвентарному файлу.
- -m ping — опция для указания модуля, который будет выполнен на хостах. В данном случае модуль ping будет выполнен для проверки доступности хостов в группе skb_ansible.

2.2.

```
skb_test | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

Зелёный блок текста — это ответ, который выдал ansible. Давайте разберём его подробно.

Результат выполнения команды ansible skb_test -i inventory.yml -m ping показывает успешное выполнение модуля ping на хосте skb_test. Вот расшифровка вывода:

- SUCCESS указывает, что задача выполнена успешно и без ошибок;
- "ansible_facts": — секция, которая содержит информацию об ансибл-фактах, в данном случае — о расположении Python-интерпретатора на хосте;
- "discovered_interpreter_python": "/usr/bin/python3" — ансибл-факт, который указывает путь к Python-интерпретатору, используемому на хосте;
- "changed": false указывает, что в результате выполнения задачи не было внесено изменений на хосте;
- "ping": "pong" — это результат выполнения модуля ping. Если значение "ping" равно "pong", это означает, что хост успешно ответил на ICMP-пакеты и доступен.

В данном случае вывод показывает, что хост skb_test успешно ответил на пинг и доступен.

4. Ansible modules

Модуль в контексте Ansible — это самостоятельный исполняемый файл или скрипт, который выполняет определённые операции на удалённых хостах. Модули дают возможность автоматизировать различные задачи: они предоставляют готовые инструменты для выполнения определённых действий на удалённых хостах. Они также упрощают процесс автоматизации и позволяют легко и гибко настраивать системы.

В Ansible уже предоставляется большое количество встроенных модулей для выполнения различных задач. Некоторые из них включают:

- ping — модуль для проверки доступности хоста;
- command — модуль для выполнения команд на удалённом хосте;
- copy — модуль для копирования файлов на удалённый хост;
- template — модуль для генерации конфигурационных файлов на основе шаблонов и переменных.

В Ansible также можно создавать собственные кастомные модули. Это полезно, если вам необходимо выполнить специфические действия, которые не поддерживаются встроенными модулями.

5. Ansible playbooks

Плейбук в Ansible представляет собой описательный файл в формате YAML, который содержит набор задач и инструкций для автоматизации конфигурации и управления системами. Плейбуки позволяют определить желаемое состояние системы и указать, каким образом достичь этого состояния.

Основные компоненты плейбука включают:

- Задачи (Tasks).

  Задачи представляют собой фундаментальные единицы работы в плейбуке. Каждая задача определяет конкретное действие, которое должно быть выполнено на удалённых хостах. Задачи выполняются в порядке их указания в плейбуке.

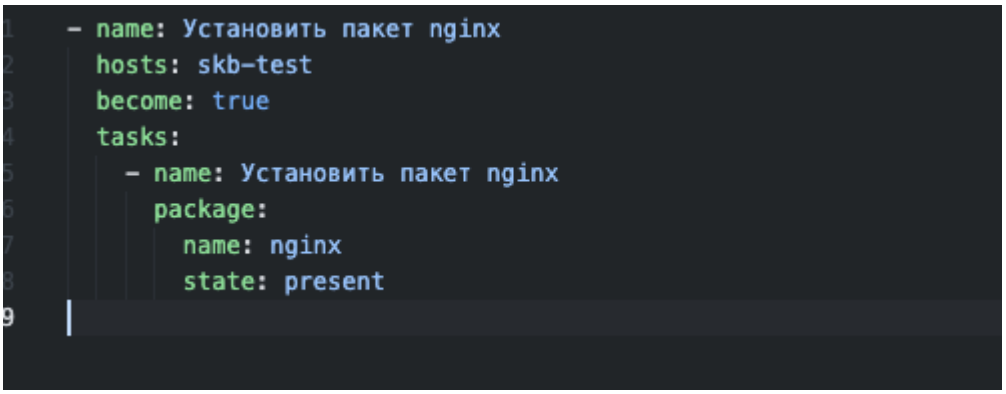
- Модули (Modules).

  Модули представляют собой исполняемый код, который выполняет конкретные задачи на удалённых хостах. Ansible предоставляет множество встроенных модулей для различных операций, таких как управление пакетами, работа с файлами, настройка сервисов и многое другое. Модули используются в задачах для выполнения конкретных действий.

- Условия и циклы.

  В плейбуках Ansible можно использовать условия и циклы для определения логики выполнения задач. Условия позволяют выполнить задачи только при определённых параметрах, например наличии определённого файла на хосте. Циклы позволяют повторить выполнение задачи для нескольких элементов, например для каждого хоста из списка.

Давайте теперь напишем и разберём простой плейбук, который можно поставить, например, на машину nginx:



1. Данный плейбук в Ansible представляет собой файл YAML, в котором описываются задачи для автоматизации конфигурации и управления системами.

Разберём каждую часть плейбука подробнее:

- name: — это описание задачи, которая будет отображаться при выполнении плейбука;
- hosts: skb-test — указывается конкретный хост или группа хостов, на которых должна быть выполнена задача. В данном случае указана группа хостов с именем skb-test, которая должна быть определена в вашем инвентаре Ansible;
- become: true — данный параметр указывает, что Ansible должен использовать привилегии суперпользователя (часто используется для выполнения команд с правами root). Если установлено значение true, Ansible попытается получить привилегии суперпользователя перед выполнением задач;
- tasks: — этот блок содержит список задач, которые нужно выполнить на выбранных хостах. В данном случае — только одна задача;
- - name: Установить пакет nginx — снова указывается описание задачи. Задача имеет имя «Установить пакет nginx»;
- package: — здесь используется модуль package, предоставляемый Ansible. Модуль package позволяет управлять пакетами на хосте. В данном случае мы хотим установить пакет с именем nginx;
- name: nginx — в параметре name указывается имя пакета, который нужно установить. В нашем случае мы указываем nginx;
- state: present — в параметре state указывается желаемое состояние пакета. Здесь мы указываем present — это означает, что Ansible должен убедиться, что пакет nginx установлен на хосте.

Таким образом, данный плейбук будет выполняться на группе хостов skb-test, получая привилегии суперпользователя и устанавливая пакет nginx на каждом хосте в состоянии present (если он ещё не установлен).

2. Запускаем наш плейбук следующей командой:

```
evgeny@MacBook-Pro-evgeny ~/ansible
> $ ansible-playbook pb.yml -i inventory.yaml
```

3. Дальше происходит магия:

```
PLAY [Установить пакет nginx] **************************************************

TASK [Gathering Facts] *********************************************************
ok: [skb-test]

TASK [Установить пакет nginx] **************************************************
ok: [skb-test]

PLAY RECAP *********************************************************************
skb-test                   : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

4. В консоли ansible выведет всю подробную информацию по выполнению плейбука.

Теперь давайте посмотрим на ещё один плейбук, в котором есть циклы:

```yaml
---
- name: Конфигурация сервера
  hosts: skb_test
  become: true
  tasks:
    - name: Обновить пакеты
      apt:
        update_cache: yes
        upgrade: yes

    - name: Установить пакеты
      apt:
        name:
          - nginx
          - postgresql
          - redis
        state: present

    - name: Скопировать конфигурационный файл nginx
      template:
        src: nginx.conf.test
        dest: /etc/nginx/nginx.conf

    - name: Настроить доступ к базе данных
      template:
        src: pg_hba.conf.test
        dest: /etc/postgresql/pg_hba.conf

    - name: Настроить конфигурацию Redis
      template:
        src: redis.conf.test
        dest: /etc/redis/redis.conf

    - name: Запустить службы
      service:
        name: "{{ item }}"
        state: started
        enabled: yes
      loop:
        - nginx
        - postgresql
        - redis
```

В этом плейбуке определён один хост с именем skb_test. Для этого хоста выполняются следующие задачи:

- Обновление пакетов с использованием модуля apt.
- Установка пакетов nginx, postgresql и redis с помощью модуля apt.
- Копирование конфигурационного файла nginx.conf с помощью модуля template.
- Настройка доступа к базе данных с помощью модуля template.
- Настройка конфигурации Redis с помощью модуля template.
- Запуск и включение служб nginx, postgresql и redis с использованием модуля service.

Таким образом, этот плейбук позволяет выполнить комплексную конфигурацию и запустить несколько служб на одном хосте.
В этом плейбуке есть ещё одна примечательная деталь, которую вы, возможно, заметили, — loop. Что это такое?
В данном примере loop используется для выполнения итераций по списку служб, которые нужно запустить и включить. Цикл loop позволяет пройти по каждому

элементу списка и выполнить задачу для каждого элемента. В данном случае мы используем модуль service для запуска и включения каждой службы по отдельности. Таким образом, блок задачи будет выполнен три раза:

- первый раз — для службы nginx;
- второй раз — для службы postgresql;
- третий раз — для службы redis.

Использование loop позволяет сократить повторяющийся код и упростить плейбук. Теперь запустите этот плейбук и посмотрите на вывод.

6. Ansible roles

Роли в Ansible — это организационный механизм, который позволяет структурировать и переиспользовать код в ваших проектах Ansible. Они позволяют разделить функциональность и задачи на отдельные компоненты, которые могут быть легко включены и использованы в различных плейбуках.
Роль включает в себя набор файлов и директорий, предназначенных для выполнения определённых задач. Она может содержать:

- плейбуки,
- переменные,
- шаблоны,
- модули,
- файлы конфигурации и другие ресурсы, которые нужны для достижения конкретных целей.

Роли могут быть созданы для различных компонентов системы, таких как:

- веб-серверы,
- базы данных,
- приложения и так далее.

Основные преимущества использования ролей в Ansible:

- Переиспользование кода.
  Роли позволяют создавать модульный код, который можно легко использовать и повторно применять в различных проектах и сценариях.
- Упрощённая структура проекта.
  Роли позволяют организовать код в логические единицы, что делает его более понятным и поддерживаемым.
- Масштабируемость.
  Роли могут быть использованы для автоматизации сложных системных конфигураций с большим количеством компонентов и задач.
- Изоляция функциональности.
  Каждая роль может быть ответственна за определённую функциональность или компонент системы, что обеспечивает чёткое разделение обязанностей.

Для создания роли в Ansible обычно используется специальная команда ansible-galaxy:

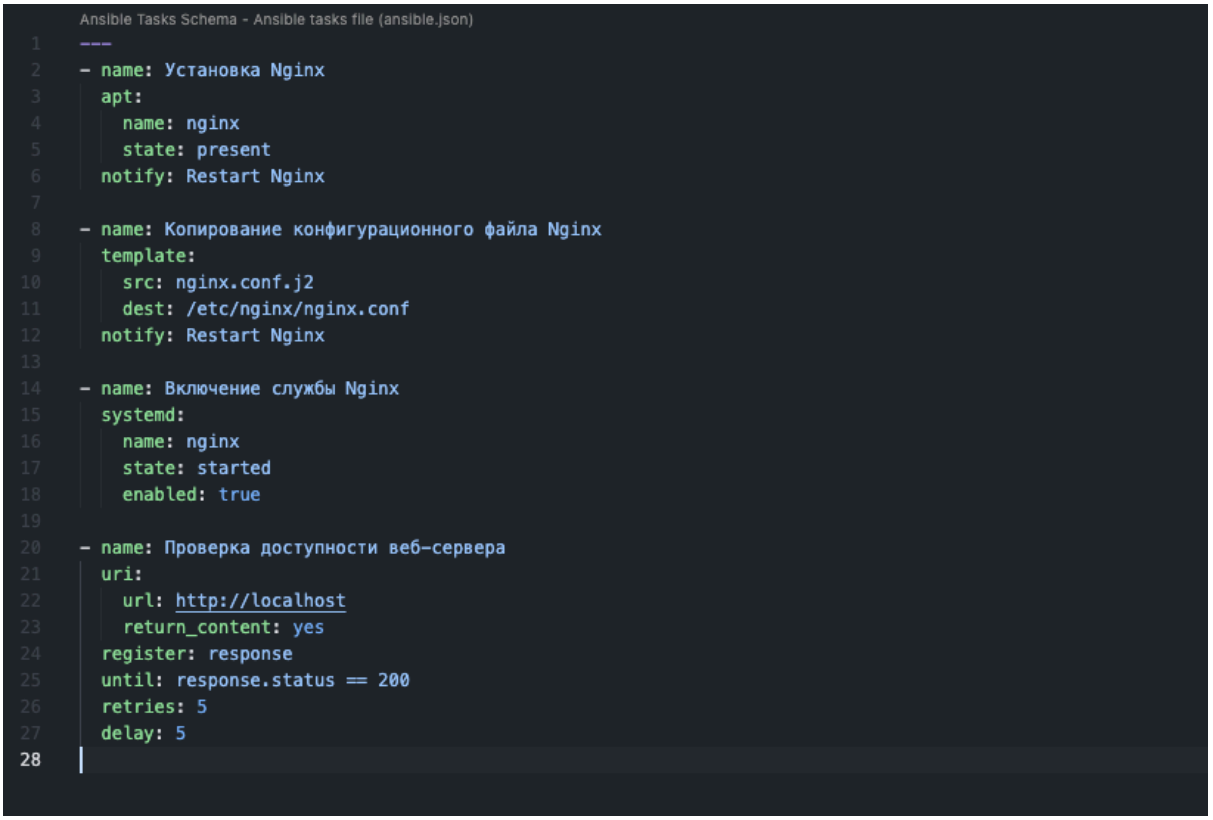
```
> $ ansible-galaxy init test
- Role test was created successfully
```

Она создает структуру каталогов и файлов для роли, включая основные файлы, такие как:

- main.yml — плейбук роли;
- vars — переменные роли;
- templates — шаблоны.

Роли могут быть использованы в плейбуках с помощью указания их названия в секции roles.
Использование ролей упрощает разработку, поддержку и масштабирование автоматизации в Ansible, обеспечивая модульную структуру и повторное использование кода.

Давайте посмотрим на практике, как это работает.
1. Файл main.yml будет выглядеть следующим образом:



В этом файле у нас задачи для установки пакета Nginx, копирования конфигурационного файла, включения службы и проверки доступности веб-сервера.

2. Конфигурационный файл Nginx представлен в виде шаблона nginx.conf.j2, который будет скопирован на хост и использован для настройки веб-сервера.
Файл с конфигурацией nginx, который мы копируем на машину:

```
worker_processes auto;

events {
    worker_connections 1024;
}

http {
    server {
        listen 80;
        server_name localhost;

        location / {
            root /var/www/html;
            index index.html;
        }
    }
}
```

3. Основной файл плейбука — pb.yml.

pb.yml > {} 0 > [ ] roles

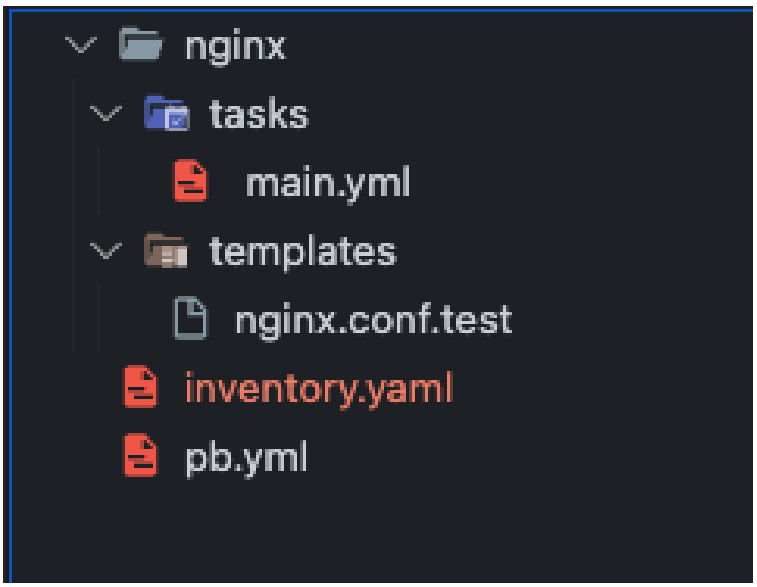
```
---
- name: Установка и конфигурирование Nginx
  hosts: skb_test
  become: true

  roles:
    - nginx

```

В этом примере у нас появился плейбук и роль для установки и конфигурирования Nginx на хосте. Плейбук:

- указывает группу хостов skb_test;
- включает привилегии суперпользователя (become: true);
- применяет роль nginx.

Роль nginx содержит задачи для установки пакета Nginx, копирования конфигурационного файла, включения службы и проверки доступности веб-сервера.

Итоговая структура файлов у нас будет выглядеть примерно так:



4. Запустите плейбук с помощью команды ansible-playbook pb.yml для установки и настройки Nginx на хосте.