Лабораторная работа 4. Управление конфигурациями в Ansible

Ansible – система управления конфигурациями, обеспечивающая простую, но мощную автоматизацию настроек, обслуживания и развертывания серверов, служб, программного обеспечения, сервисов и др. одновременно в целой сети удаленно расположенных устройств [1].

В других подобных программных средах, таких как Chef, Puppet, SaltStack, используется pull метод для отправки команд, при котором на управляемых хостах требуется установка специальных агентов, которые сравнивают конфигурации и, при наличии изменений, «вытягивают» инструкции из главного компьютера. Ansible поддерживает как pull, так и push метод, «проталкивающий» команды и не требующий наличия дополнительного ПО на клиентских машинах. При этом все конфигурации хранятся локально в удобном для администратора виде.

Кроме того, Ansible активно развивается в сторону поддержки сетевого оборудования, и в нём постоянно появляются новые возможности и модули для работы с сетевым оборудованием. Некоторое сетевое оборудование поддерживает другие системы управления конфигурациями (позволяет установить агента).

Подключение к устройствам в Ansible происходит по SSH. Ansible написана на Python и использует язык yaml для описания конфигураций. Оба языка являются достаточно простыми для понимания, поэтому с Ansible легко начать работать. Также существует отдельный сайт Ansible Galaxy (https://galaxy.ansible.com/) с большой базой готовых пользовательских решений для тех или иных задач.

Один из главных принципов Ansible – идемпотентность, то есть независимо от количества вызовов какой-либо операции результат будет одинаковым. Конфигурация в данной системе осуществляется с помощью нескольких основных конфигурационных файлов, которые будут рассмотрены далее более подробно. На рисунке 1 изображена общая архитектура систем с Ansible [2].

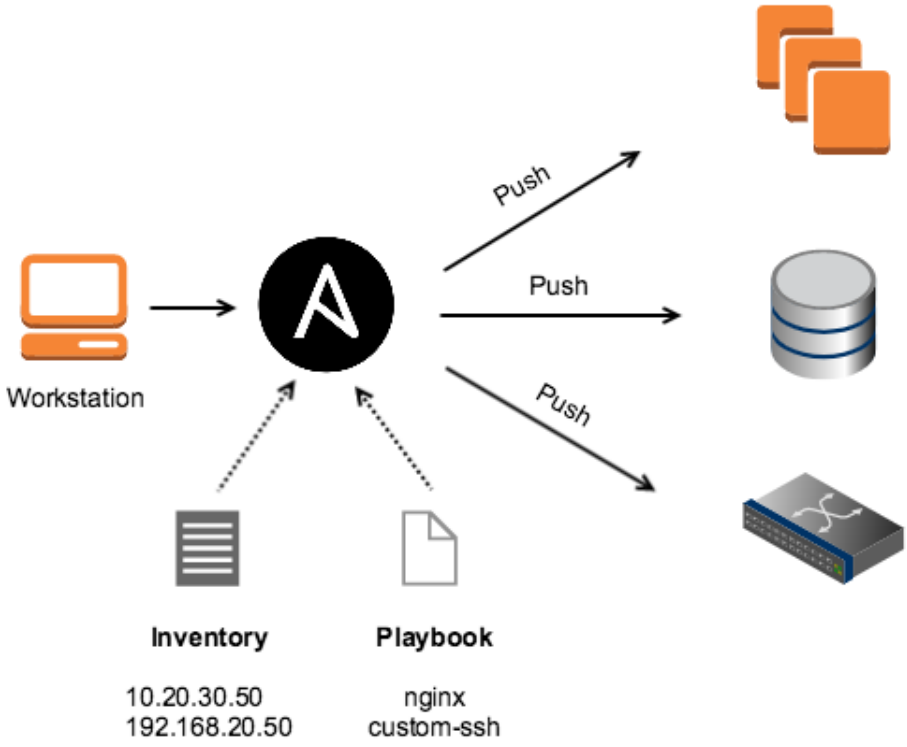

Рисунок 1 – Архитектура Ansible

1 Начало работы

Для управляющих хостов в Ansible существуют следующие требования:

- Windows не может быть операционной системой (подходит любой Linux: RedHat, Debian, CentOS, OS X);
- Версия Python 2.7 или выше.

На управляемых устройствах может быть установлена Windows. Подключение к устройствам может осуществляться как по логину/паролю администратора, так и по SSH ключу (если установлен Linux). Желательно наличие Python 2.7+ на клиентах, на Windows также необходим PowerShell 3.0 и .NET 4.0 или выше [3].

Для работы с Ansible необходимо развернуть несколько экземпляров операционных систем. Это можно сделать разными способами: с помощью программ для виртуализации, таких как VirtualBox, VMware, или с помощью облачных сервисов, например, AWS. Также для более простого и быстрого создания множества копий виртуальных машин можно использовать Proxmox.

Один из самых простых вариантов – это создать четыре копии Ubuntu Server (использовалась версия 22.04.2) в VirtualBox, один из которых будет

управляющим и на который будет установлен Ansible, три другие будут использоваться в качестве клиентов.

Чтобы виртуальные машины видели друг друга через сеть, в настройках сети в качестве типа подключения нужно выбрать *Сетевой мост*. Для удобного администрирования всех серверов из одного окна предлагается использовать специальные терминалы, такие как MobaXterm, PuTTY или mRemoteNG.

После запуска операционных систем обновляем списки репозиториев и сами пакеты:

```
sudo apt update
sudo apt upgrade
```

Ansible для отправки команд использует SSH. При установке Ubuntu Server предлагается установить SSH сервер (SSH клиент установлен по умолчанию), однако, если он не установлен, установить, запустить и открыть нужный порт сервера можно с помощью следующих команд:

```
sudo apt install openssh-server
sudo systemctl enable ssh
sudo ufw allow ssh
```

С помощью `ip address` получаем IP адреса каждой системы. Открыть терминал в MobaXterm можно введя ip адрес системы в поле *Session→SSH→Basic SSH settings→Remote host*, после чего происходит авторизация с помощью логина и пароля в терминале.

На следующем шаге устанавливаем Ansible на управляющую систему. Это можно сделать разными способами. Например, через pip:

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

Или с помощью менеджера пакетов:

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

Более подробную информацию об установке Ansible в том числе и на другие операционные системы можно найти в официальной документации [4].

Чтобы при вызове некоторых команд не запрашивался пароль sudo и для упрощения дальнейших настроек, в файле /etc/sudoers для клиентских хостов были изменены настройки группы sudo следующим образом:

```
%sudo ALL=(ALL) NOPASSWD: ALL
```

2 Основные файлы конфигурации

Проверить установленный Ansible можно с помощью `ansible –version`, также с помощью нее можно узнать адрес основной директории. В случае установки с помощью pip базовые файлы конфигурации не создаются (в строке *config file* будет написано None), поэтому второй способ предпочтительнее.

Ansible.cfg – это основной конфигурационный файл ini формата, в котором можно указать стандартные настройки для Ansible, например, стандартный путь к данным о хостах [5]. Если он не был создан автоматически, то создадим новую директорию и в ней создадим конфигурационный файл:

```
sudo mkdir ansible

cd ansible

sudo touch ansible.cfg
```

Указание IP адреса напрямую в командах Ansible не очень удобный способ управления инфраструктурой. Чтобы не указывать его каждый раз, в Ansible существует inventory файл (или инвентарный файл), который используется для создания списка IP адресов, имён хостов для управления. В простейшем случае в этом файле можно указать список IP адресов по одному на строчку. Файл имеет ini формат. Если ansible.cfg не был создан автоматически, то его также нужно создать вручную:

```
sudo touch hosts
```

При установке через менеджер пакетов стандартная директория Ansible следующая:

```
cd /etc/ansible/
```

Откроем инвентарный файл:

```
sudo nano hosts
```

Здесь нужно прописать адреса созданных linux систем в качестве управляемых серверов. Для ознакомления с принципами работы будет достаточно произвести авторизацию по паролю, хотя стоит учитывать, что прописывать пароли в текстовом файле является не самым оптимальным и безопасным вариантом, для этих целей лучше использовать SSH ключи. Введенные адреса выглядят следующим образом:

```
host1 ansible_host=192.168.1.7 ansible_user=user ansible_password=1111
```

```
host2 ansible_host=192.168.1.8 ansible_user=user ansible_password=1111

host3 ansible_host=192.168.1.9 ansible_user=user ansible_password=1111
```

В каждой строчке записывается сначала необязательное имя хоста, в параметре ansible_host его IP адрес, в параметрах ansible_user и ansible_password записывается логин и пароль пользователей управляемых систем.

3 Ad-hoc команды

Теперь рассмотрим процесс выполнения запросов на удаленных серверах. Самый простой вариант запуска таких команд – это ad-hoc команды, когда запрос к серверу выполняется напрямую из командной строки. Ad-hoc команды выглядят следующим образом [6]:

```
ansible [pattern] -m [module] -a "[module options]"
```

Паттерн используется для указания на конкретные хосты, которым адресуются команды. После флага -m следует название модуля, который отвечает за выполнение конкретной задачи. В настоящее время в Ansible существует более 1000 различных модулей. Полный список модулей можно найти на официальном сайте [7].

Как правило, при вызове модуля ему нужно передать аргументы. Какие-то аргументы будут управлять поведением и параметрами модуля, а какие-то передавать, например, команду, которую надо выполнить. Аргументы прописываются после флага -a, как правило в виде пар "ключ=значение" разделенных пробелами.

Попробуем ввести следующую простую команду:

```
ansible all -i hosts -m ping,
```

где `all` – передача команды всем хостам, записанным в инвентарном файле,

`-i hosts` – после флага -i записывается название инвентарного файла,

`ping` – модуль для проверки доступности хоста по сети, также проверяет доступную версию python.

Возможно, что при выполнении этой команды отобразится ошибка «Using a ssh password instead of a key is not possible…». Она связана с тем, что включена проверка ключа узла, но отпечаток ключа узла, к которому происходит подключение, отсутствует в соответствующем пользовательском файле. Чтобы отключить эту проверку, изменим файл ansible.cfg:

```
[defaults]
```

```
host_key_checking = false
```

Также сразу запишем путь до инвентарного файла, чтобы каждый раз не прописывать это в команде:

```
inventory = ./hosts
```

Если все настроено правильно, должно отобразиться сообщение об успешном выполнении команды для всех клиентских хостов.

Команду также можно выполнить для конкретного хоста, введя его имя вместо ключевого слова all:

```
ansible host1 -i hosts -m ping
```

4 Группы и переменные

Для обращения к определенному множеству хостов в Ansible существует возможность создания групп. Для этого в инвентарном файле в квадратных скобках записывается название группы, а ниже адресы хостов. Ниже представлен пример создания групп:

```
[group1]
host1 ansible_host=192.168.1.7 ansible_user=user ansible_password=1111
[group2]
host2 ansible_host=192.168.1.8 ansible_user=user ansible_password=1111
host3 ansible_host=192.168.1.9 ansible_user=user ansible_password=1111
[group3:children]
group1
group2
```

С помощью ключевого слова children можно создать группу из групп. При выполнении ad-hoc команд можно указывать группу, для которой она будет выполнена.

Модули Ansible, как правило, идемпотентны. Это означает, что модуль можно выполнять сколько угодно раз, но при этом модуль будет выполнять изменения, только если система не находится в желаемом состоянии. Это можно увидеть, создав файл и скопировав его два раза с помощью модуля copy:

```
sudo touch file.txt
ansible group2 -m copy -a "src=file.txt dest=/home mode=0777" -b
```

Флаг -b здесь нужен для выполнения команды с повышенными привилегиями. При втором выполнении в поле changed результата в консоли будет стоять false, то есть команда была проигнорирована.

В Ansible для каждого хоста или группы можно создавать любый переменные. Это можно сделать как в инвентарном файле, с помощью конструкции [[название_группы]:vars], так и с помощью создания отдельной папки group_vars в каталоге с конфигурационными файлами [8].

Попробуем перенести параметры ansible_user и ansible_password в отдельные файлы, чтобы не прописывать их каждый раз при добавлении нового хоста, а также добавить одну новую переменную. Для этого создадим отдельные файлы для каждой группы с одноименным названием в формате yaml в папке group_vars. Файл group1.yaml выглядит следующим образом:

```
---
ansible_name: user
ansible_password: 1111
group_name: group1
```

group2.yaml:

```
---
ansible_name: user
ansible_password: 1111
group_name: group2
```

Отобразим новую переменную с помощью модуля debug:

```
ansible all -m debug -a "var=group_name"
```

5 Playbooks

Ad-hoc команды могут выполнять только один запрос за раз. Для выполнения целых наборов команд с помощью скриптов в Ansible существуют playbooks. Playbook (плейбук) – это конфигурационный yaml-документ, который задает определенную последовательность действий управляемым компьютерам или серверам [9].

Рассмотрим строение плейбука на примере знакомого модуля ping. Для начала создадим файл:

```
sudo nano playbook.yml
```

И вставим в него следующий код:

```
---
```

```
- name: First playbook
  hosts: all
  become: yes


  tasks:
  - name: Ping
    ping:
```

В начале файла указывается имя плейбука, список хостов, строчка become: yes соответствует флагу -b. Ниже, под строкой tasks перечисляются вызываемые модули, начиная с произвольного названия для команды (name). Ниже прописываются параметры для модуля, если они требуются.

Вызов плейбука происходит с помощью простой команды:

```
ansible-playbook playbook.yml
```

В плейбуках также возможно использование таких конструкций как циклы, условия, разделения на блоки, а также переменных и обработчиков. Создадим еще один плейбук с другим названием и вставим в него следующий код, который будет обновлять репозитории, устанавливать несколько пакетов, запускать веб сервер и копировать в его папку html файл:

```
---
- name: Web server
  hosts: all
  become: yes

  vars:
    filehtml: ./index
    destination: /var/www/html

  tasks:
  - name: Update
    ansible_pkg_mgr:
      update_cache: yes

  - name: Upgrade
    ansible_pkg_mgr:
      upgrade: yes

  - block:
    - name: Install packages
      apt: name={{item}} state=present
      loop:
         - apache2
         - nmon
         - ghex
```

```
    - name: Copy html
      copy: src={{filehtml}} dest={{destination}} mode=0777
      notify: Restart

    - name: Start Web Server
      service: name=apache2 state=started enabled=yes
    when: ansible_hostname == "host1"

  handlers:
  - name: Restart
    service: name=apache2 state=restarted
```

Здесь встречаются сразу несколько новых конструкций. Переменные записываются над командами под ключевым словом vars. В дальнейшем эти переменные могут быть использованы с помощью двойных фигурных скобок. Можно использовать не только переменные, объявленные в начале плейбука, но и записанные в инвентарном файле или в папке group_vars, а также переменные, несущие различную информацию об удаленных хостах, которые можно посмотреть с помощью команды `ansible all -m setup`.

Любой набор команд может быть объединен в блок (ключевым словом block). В конце блока может быть записано условие выполнения этого блока с ключевым словом when. Условия могут быть использованы не только в конце блоков, но и после любой отдельной команды. В данном случае для проверки условия используется имя удаленного хоста. Таким образом несмотря на то, что вначале плейбука целевыми хостами выбраны все, выделенный блок кода исполнится только для одного хоста.

С помощью циклов можно вызывать выполнение команды для определенного набора параметров. Для этого используется зарезервированное слово item. Варианты значений переменной записываются в конце команды в последовательности loop. Обработчики реализуются с помощью отдельного словаря, располагающегося в конце документа под ключом handlers. Для вызова обработчика можно использовать, например, ключ notify, который следует вводить в конце команды. Если выполнение команды вызовет изменения на удаленном хосте, будет вызван обработчик с именем, соответствующим значению ключа notify. В данном случае при изменениях в html файле вызывается команда, перезапускающая сервис веб-сервера.

Перед запуском плейбука необходимо создать файл index.html в основной директории с простым кодом, который будет передаваться на веб-сервер:

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta    name="viewport"    content="width=device-width,    initial-
scale=1.0">
    <title>Document</title>
</head>
<body>
    Ansible
</body>
</html>
```

После успешного исполнения плейбука по IP адресу хоста №1 в браузере должна открыться переданная веб-страница.

6 Шаблонизация

При конфигурации удаленных серверов файлы могут отличаться несколькими параметрами. Использование статических файлов в данном случае не является эффективным решением. Шаблоны обычно используются для настройки сервисов на основе значений переменных, которые можно установить в самом плейбуке, определить во включенных файлах переменных или получить с помощью модуля setup. Это позволяет создавать универсальные настройки, которые адаптируют свое поведение с учетом динамической информации.

Рассмотрим возможности для шаблонизации в Ansible. Файлы шаблонов обычно имеют расширение .j2, которое обозначает используемый механизм создания шаблонов Jinja2. Jinja2, представляет собой предназначенный для проектирования язык шаблонов для Python. В нем можно работать с управляющими структурами, проводить различные манипуляции данными, сравнения и т. д. [10].

Первым делом изменим расширения файла index:

```
sudo mv index.html index.j2
```

И изменим содержимое тега body:

```
<br> Web Server installed on {{ansible_hostname}} </br>
<br>Server's group: {{group_names[0]}}</br>
```

Для генерации страницы по шаблону используется модуль template. В предыдущем плейбук файле уберем условие when у блока, а также добавим следующий код в словарь tasks:

```
- name: Generate Page
  template: src={{filehtml}} dest={{destination}}/index.html mode=0777
  notify: Restart
```

После запуска плейбука на всех трех серверах будут запущены веб-серверы. При переходе в браузере по их адресам на странице отобразится информация, относящаяся к конкретному удаленному серверу.

7 Роли

Роли применимы, когда используется очень большое количество серверов, управлять которыми написав один playbook становится сложно. Роли позволяют отдавать команды различным группам машин, разделяя их по выполняемой функции. Основная их идея заключается в том, чтобы позволить повторно использовать общие шаги настройки между различными типами серверов [11].

Роль в Ansible – это независимая сущность, решающая какой-то свой набор задач. То есть роль – это директория с поддиректориями и файлами, где расположены задачи. Обычно роли независимы от проектов и хранятся удаленно, если их зависимость не продиктована логикой проекта. Роли имеют строго заданную структуру и создаются командой ansible-galaxy init [название роли] [12, 13].

Для примера разобьем плейбук для запуска веб-сервера на две простые роли: upgrade_packages для обновления всех пакетов и setup_web_servers для установки и запуска веб-сервера. Первым делом создадим папку roles в основной папке проекта и перейдем в неё:

```
sudo mkdir roles
```

```
cd roles/
```

Сгенерируем необходимые папки для ролей:

```
sudo ansible-galaxy init upgrade_packages
```

```
sudo ansible-galaxy init setup_web_servers
```

Для каждой роли генерируется набор папок с файлами main.yml изображенный на рисунке 2.

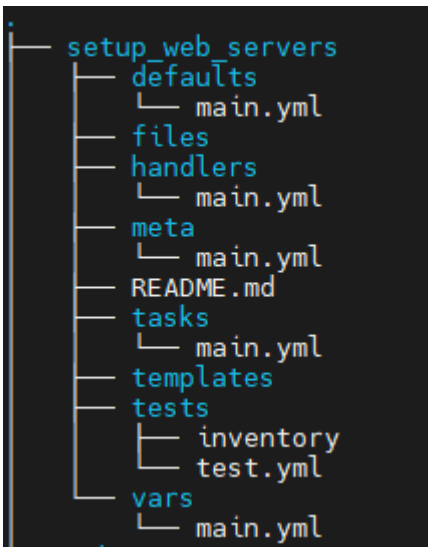

Рисунок 2 – Директории для роли в Ansible

Структура роли включает следующие основные каталоги [13]:

vars – переменные, связанные с этой ролью;

defaults – переменные с более низким приоритетом по умолчанию для этой роли;

files – файлы для использования с ресурсом;

handlers – файлы обработчиков;

meta – ролевые зависимости;

tasks – файл команд tasks, может включать файлы меньшего размера, если это необходимо;

templates – файлы для использования с ресурсом шаблона.

Чтобы сделать из плейбука роль достаточно перенести участки кода в файлы main.yml в нужных папках. Файл index.j2 в папку templates:

```
sudo mv index.j2 roles/setup_web_servers /templates
```

Содержимое словарей vars и handlers переносим в соответствующие файлы в папке setup_web_servers, так как они используются только для команд, запускающих веб-серверы. Первые две команды в tasks переносим в upgrade_packages/tasks/main.yml, остальные в папку setup_web_servers/tasks/main.yml.

Ansible знает откуда брать шаблоны для модуля template, поэтому в аргументе src этого модуля достаточно только указать название j2 файла, а строчку `"filehtml: ./index"` в vars можно удалить.

В итоге в файле плейбука должны остаться только основные параметры запуска и поле roles, в котором перечисляется последовательность запуска ролей:

```
---
```

```
- name: Web server
  hosts: all
  become: yes

  roles:
    - upgrade_packages
    - setup_web_servers
```

Таким же образом можно, например, добавлять роли для конфигурации баз данных или списка пользователей в операционной системе и гибко определять их набор для каждой группы хостов.

Если все сделано правильно, то измененный плейбук должен запуститься без ошибок. При необходимости можно также добавлять условия для списка ролей:

```
roles:
    - upgrade_packages
    - {role: setup_web_servers, when: ansible_hostname == "host1"}
```

Приложение А
(справочное)
Библиографический список

1. Самойленко, Н. Основы Ansible для сетевых инженеров : учебник / Н. Самойленко. – 2022 . – 145 с. – Текст : непосредственный
2. Ansible mode of operations [Электронный ресурс]. – Текст: электронный // O'reilly [сайт]. – URL: https://www.oreilly.com/library/view/enterprise-cloud-security/9781788299558/43f530d6-2296-4cc7-9ed1-bff6d70a2aa3.xhtml (дата обращения 05.05.2023) – Режим доступа: свободный.
3. Setting up a Windows Host [Электронный ресурс]. – Текст: электронный // Ansible documentation [сайт]. – URL: https://docs.ansible.com/ansible/latest/os_guide/windows_setup.html (дата обращения 06.05.2023) – Режим доступа: свободный.
4. Installation Guide [Электронный ресурс]. – Текст: электронный // Ansible documentation [сайт]. – URL: https://docs.ansible.com/ansible/latest/installation_guide/ (дата обращения 07.05.2023) – Режим доступа: свободный.
5. Ansible Configuration Settings [Электронный ресурс]. – Текст: электронный // Ansible documentation [сайт]. – URL: https://docs.ansible.com/ansible/latest/reference_appendices/config.html#ansible-configuration-settings-locations (дата обращения 08.05.2023) – Режим доступа: свободный.
6. Patterns: targeting hosts and groups [Электронный ресурс]. – Текст: электронный // Ansible documentation [сайт]. – URL: https://docs.ansible.com/ansible/latest/inventory_guide/intro_patterns.html (дата обращения 09.05.2023) – Режим доступа: свободный.
7. All modules [Электронный ресурс]. – Текст: электронный // Ansible documentation [сайт]. – URL: https://docs.ansible.com/ansible/2.9/modules/list_of_all_modules.html (дата обращения 10.05.2023) – Режим доступа: свободный.
8. Organizing host and group variables [Электронный ресурс]. – Текст: электронный // Ansible documentation [сайт]. – URL: https://docs.ansible.com/ansible/latest/inventory_guide/intro_inventory.html#organizing-host-and-group-variables (дата обращения 11.05.2023) – Режим доступа: свободный.
9. Ansible Playbook: определение, параметры, модули и примеры [Электронный ресурс]. – Текст: электронный // CoderNet [сайт]. – URL:

https://codernet.ru/articles/drugoe/ansible_playbook_opredelenie_parametryi_moduli_i_primeryi (дата обращения 12.05.2023) – Режим доступа: свободный.

10. Расширяем возможности Ansible: шаблоны [Электронный ресурс]. – Текст: электронный // Хабр [сайт]. – URL: https://habr.com/ru/companies/otus/articles/719276/ (дата обращения 13.05.2023) – Режим доступа: свободный.

11. ANSIBLE ROLES [Электронный ресурс]. – Текст: электронный // Сервис мониторинга сайтов и серверов [сайт]. – URL: https://server-gu.ru/ansible-roles/ (дата обращения 14.05.2023) – Режим доступа: свободный.

12. Ansible: развёртывание Django-стека на 3 VM [Электронный ресурс]. – Текст: электронный // 1cloud [сайт]. – URL: https://1cloud.ru/blog/ansible_django_3_vm (дата обращения 15.05.2023) – Режим доступа: свободный.

13. Roles [Электронный ресурс]. – Текст: электронный // Ansible documentation [сайт]. – URL: https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_reuse_roles.html (дата обращения 16.05.2023) – Режим доступа: свободный.