МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ им. В. И. ВЕРНАДСКОГО» ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ

Кафедра компьютерной инженерии и моделирования

# ОТЧЕТ ПО ПРАКТИЧЕСКОМУ ЗАДАНИЮ №6

**«Знакомство с Ansible»**

Практическая работа

по дисциплине «Современные технологии программирования» студента 1 курса группы ПИ-б-о-231(2)

Покидько Максим Сергеевич

направления подготовки 09.03.04 «Программная инженерия»

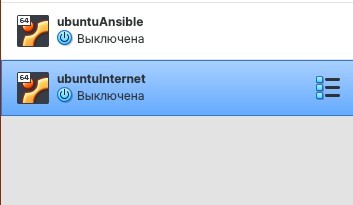
Симферополь, 2024

# Цель:

Ознакомиться на практике с инструментом для удаленного управления конфигурациями Ansible.

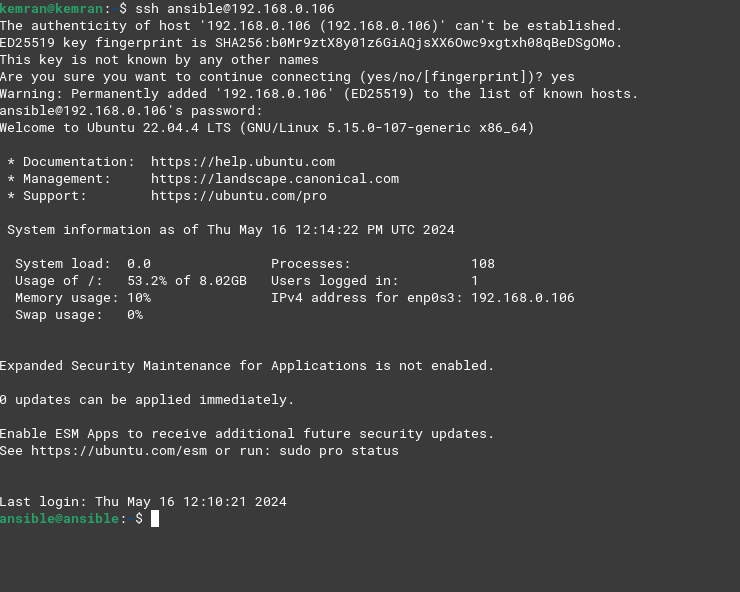
# Ход выполнения задания. Подготовка виртуальных машин

1. Нам понадобится две чистые виртуальные машины. Если у вас дефицит дискового пространства, то ВМ из предыдущих работ можно удалить, здесь они нам не понадобятся. На одной из виртуальных машин будет размещаться ansible, поэтому я её так и назову - "Ansible", а вторая будет просто "коробкой" для других виртуальных машин, которыми мы будем управлять при помощи ansible, поэтому я назову её "Internet" (имена можете выбрать по своему вкусу). По итогу, с точки зрения сети, мы хотим получить такую топологию:
2. Создайте две виртуальные машины с Ubuntu Server . В процессе установки используйте стандартные настройки. Из дополнительного софта понадобится только ssh-сервер. Для машины с именем "Internet" создайте диск большого объёма 50+ГБ.
3. В настройках сети VirtualBox установите "Сетевой мост" для обеих машин.



# Машина с Ansible

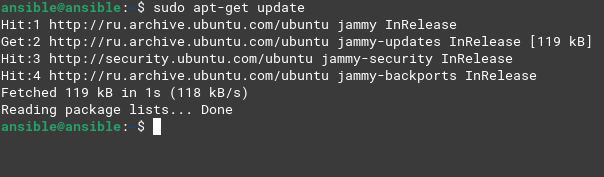
1. Для удобства дальнейшей работы, подключитесь к машине по ssh.



1. Добавьте в список dns-серверов гугловский (8.8.8.8) и(или) яндексовый (77.88.8.8).



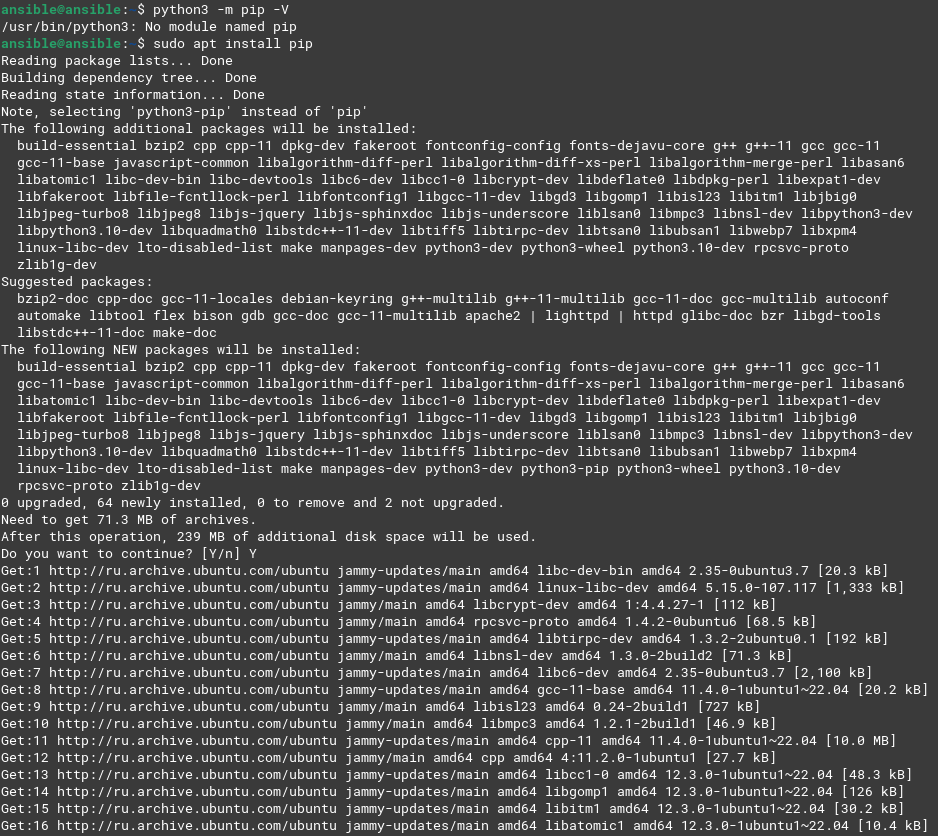
1. Обновите индексы пакетов: sudo apt-get update.



1. Проверьте, что в системе установлен python 3: python3 --version. Если нет, установите.

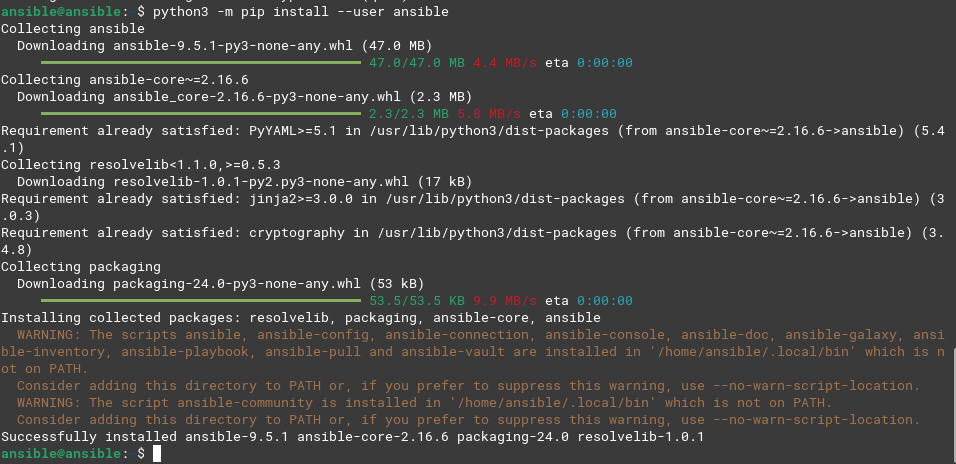


1. Проверьте, что в системе установлен pip: python3 -m pip -V. Если нет, установите.



1. Установите последнюю доступную версию ansible: python3 -m pip install -- user ansible.

Флаг --user установит пакет ansible как локальный, т.е. ansible будет доступен только текущему пользователю.



1. После установки выскочит предупреждение, что запустить ansible по имени не получится, т.к. каталог в который он установился (~/.local/bin) не добавлен в PATH.

Нам не придётся делать это вручную, т.к. путь ~/.local/bin будет добавлен в PATH автоматически после перезагрузки или перелогина. Вместо перезагрузки можно выполнить команду source ~/.profile, что вызовет принудительное обновление PATH.



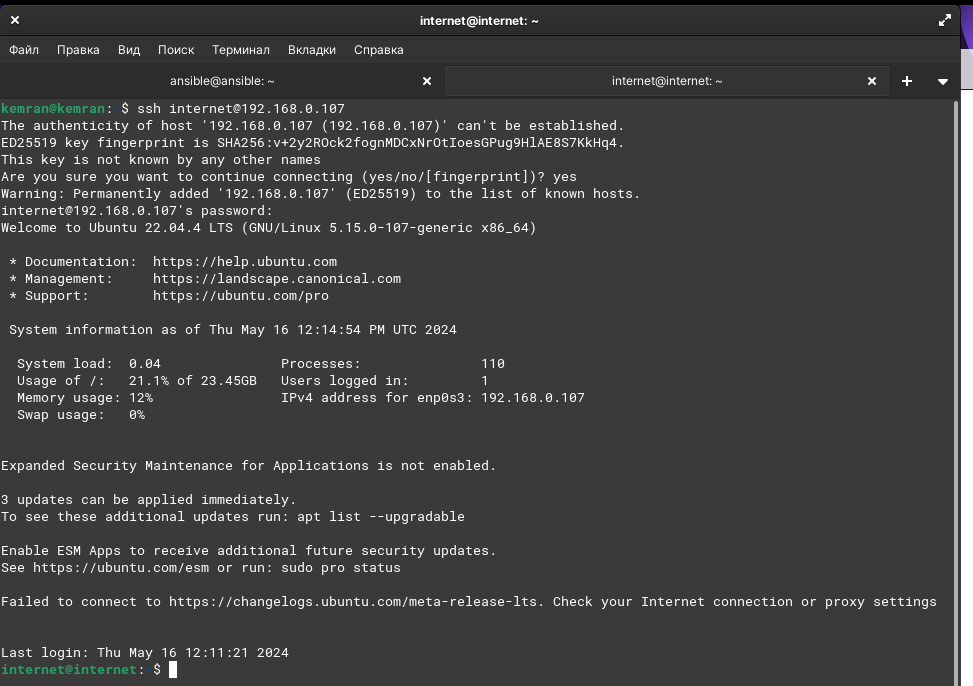
1. Теперь убедитесь, что ansible установился: ansible --version.

В результате вы увидите довольно подробный вывод о самой версии ansible, о путях к конфигам, версии python и т. д.

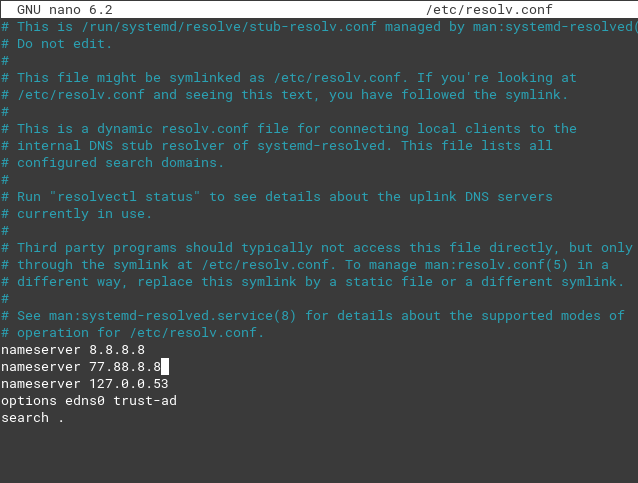


# Машина с Docker Compose

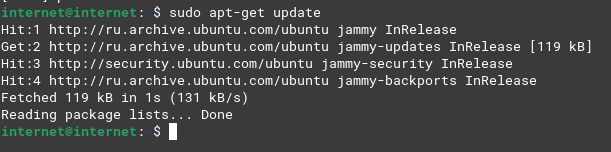
1. Для удобства дальнейшей работы, подключитесь к машине по ssh например при помощи KiTTY.



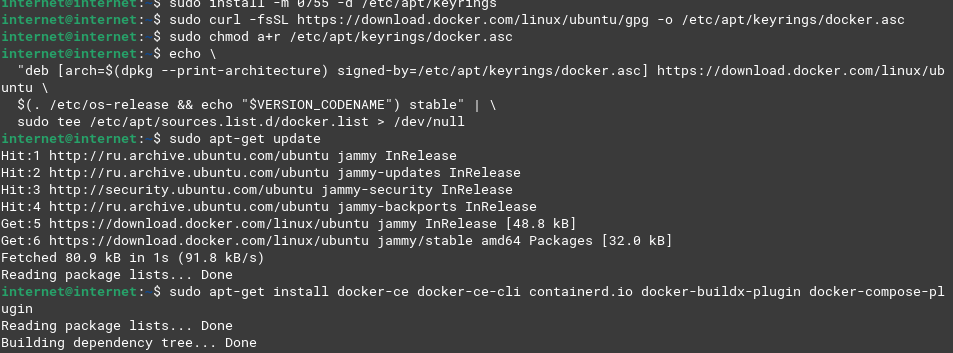
1. Добавьте в список dns-серверов гугловский (8.8.8.8) и(или) яндексовый (77.88.8.8).

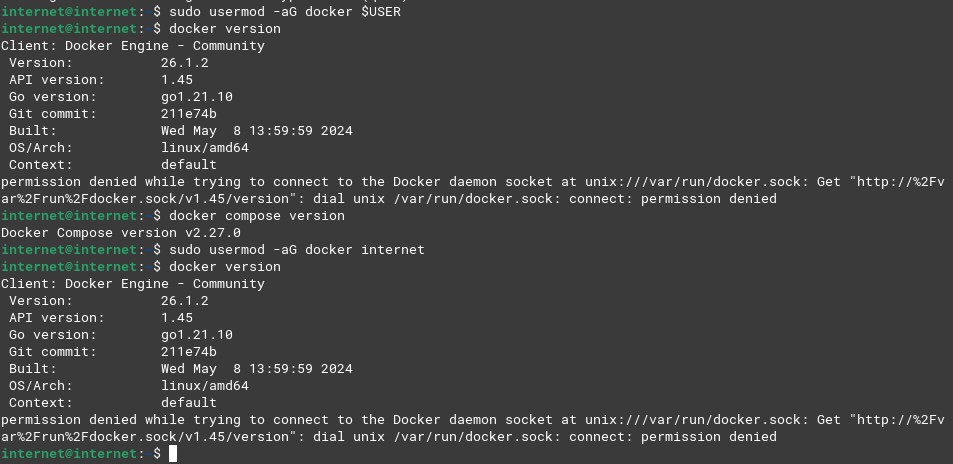


1. Обновите индексы пакетов: sudo apt-get update.



1. Устанавливать docker и docker compose будем из официальных репозиториев Docker (копируйте команды построчно, до конечного \, если он есть):





# Установим некоторые нужные пакеты:

sudo apt-get install ca-certificates curl

Добавим официальный ключ для доступа к репозиторию Docker:

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o

/etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

Следующая команда настроит репозиторий:

echo \

"deb [arch=$(dpkg --print-architecture)

signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu

\

$(. /etc/os-release && echo "$VERSION\_CODENAME") stable" | \ sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

После добавления репозитория обновим индексы пакетов:

sudo apt-get update

Теперь установим docker и docker compose:

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

Добавьте текущего пользователя в группу docker:

sudo usermod -aG docker $USER

* Перезагрузите сервер или перелогиньтесь;
* Проверьте, что установка прошла успешно:

docker compose version

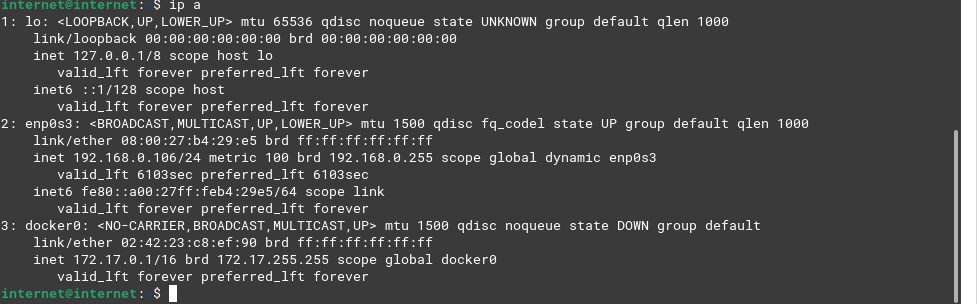
В данном случае мы установили плагин "compose" для docker, но есть и отдельный пакет "docker-compose" (как видно разница в наличии -). На данный момент он считается устаревшим.

# Создаём сервера

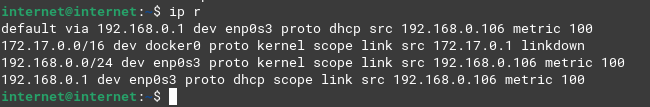
1. Выполните команду ip a и найдите сетевой интерфейс, который подключён к вашему роутеру.

Здесь же можно определить адрес подсети. После ip-адреса машины указано /24 - это маска подсети. Она говорит о том, что первые 24 бита ip-адреса фиксированы (т.е. первые 3 числа), а оставшиеся 8 бит (т.е. последнее число) могут изменяться.

Очевидно что в такой подсети есть всего 256 различных ip-адресов, два из которых служебные: 192.168.1.255 - занят под broadcast (для отправки пакета всем в подсети) и 192.168.1.0 - который и называется **адресом подсети**.



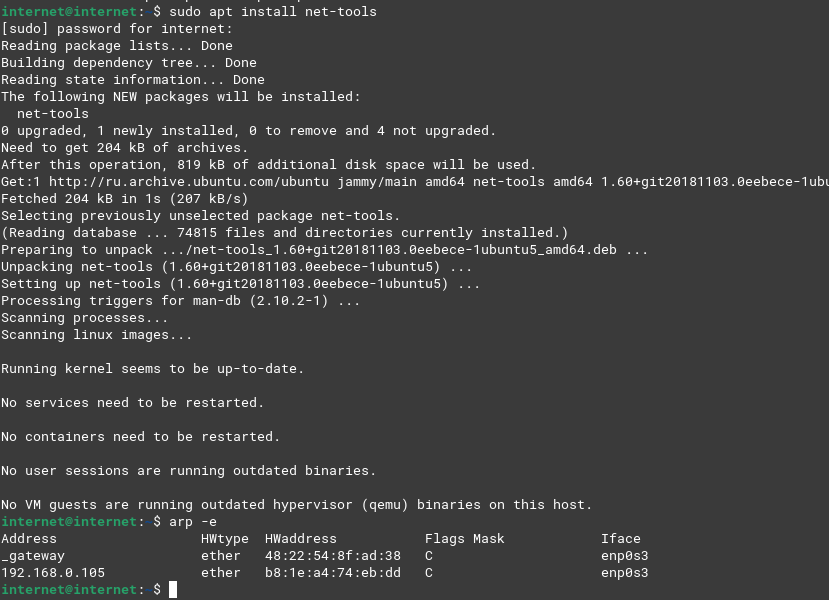
1. Выполните команду ip r и найдите default. Этот ip - **gateway** (в нашем случае это ip роутера).



1. Теперь нужно посмотреть занятые ip адреса в нашей локальной сети. Это нужно, чтобы выбрать диапазон свободных для наших серверов.

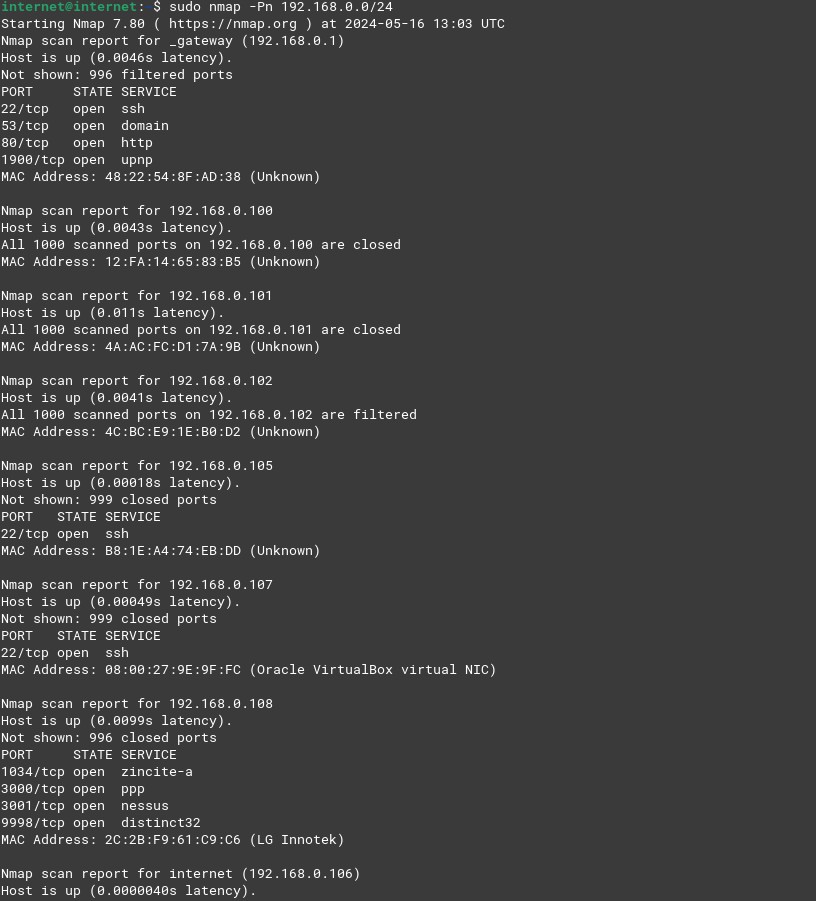
Установите утилиту arp при помощи команды: sudo apt install net-tools, а затем выполните arp -e. IP-адреса из списка заняты.

Список скорее всего будет не полным, т.к. он содержит только те IP к которым обращалась ваша машина или которые обращались к ней.



1. Чтобы уточнить список воспользуемся утилитой nmap. Установите её (sudo apt- get install nmap) и попросите просканировать все ip в сети (sudo nmap -Pn 192.168.1.0/24). К сожалению, при помощи nmap тоже не всегда можно получить полный список занятых адресов, т.к. некоторые устройства могут временно отключаться от сети для экономии батарее и т.д.

Точный способ узнать занятые IP - посмотреть их в web-интерфейсе роутера.



# Теперь воспользуемся docker compose чтобы создать сервера. Для начала пропишем в нём только один сервер.

1. На сервере с docker compose создайте файл compose.yaml и откройте его в текстовом редакторе.
2. Поместите в него следующий текст (с учётом ваших параметров сети):

services:

test: # Так я назвал сервис

image: rastasheep/ubuntu-sshd # Тот самый базовый образ dns:

* 77.88.8.8 # Яндексовый
* 8.8.8.8 # Гугловый

networks: outside:

ipv4\_address: 192.168.1.100 # Свободный ip из нашей подсети

# Здесь создаём docker-сеть networks:

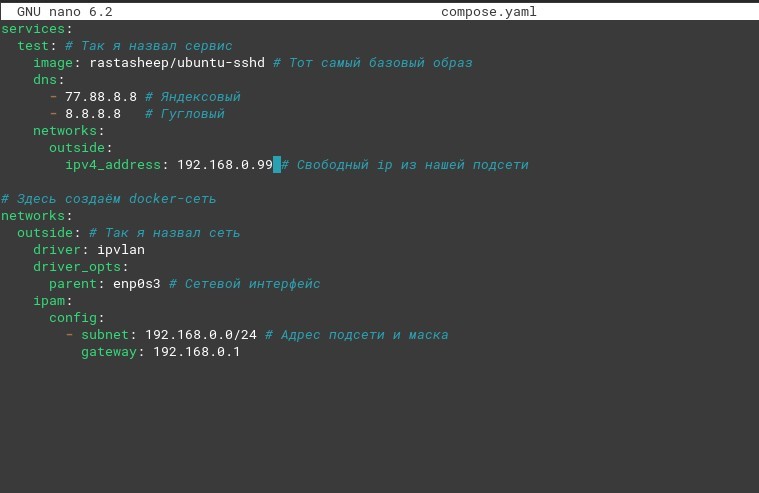
outside: # Так я назвал сеть driver: ipvlan driver\_opts:

parent: enp0s3 # Сетевой интерфейс ipam:

config:

- subnet: 192.168.1.0/24 # Адрес подсети и маска gateway: 192.168.1.1

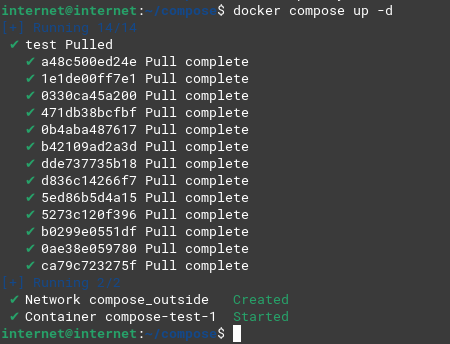
Как видно по описанию создаётся одна машина и одна сеть.



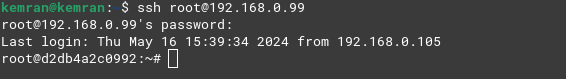
1. Наберите в терминале:

docker compose up -d

В результате, docker compose найдёт файл compose.yaml в текущей папке, создаст сеть outside и запустит машину test в фоновом режиме (-d), т.е не захватывая терминал и не выводя логи.



1. Проверьте, что к этой машине можно подключится как и к любой другой, например при помощи KiTTY (ip, port и пользователь вам известны).

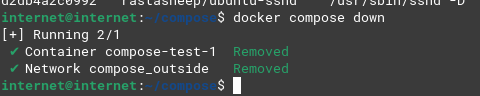


1. Наберите в терминале:

docker compose down

В результате, docker compose остановит машину и удалит созданные им контейнер и сеть.

**Внимание:** после остановки машины будут уничтожены и все изменения, которые в них были сделаны тоже исчезнут. Если хотите приостановить машины на время, без удаления, используйте docker compose stop и для последующего возобновления работы docker compose start. После остановки все запущенные на машине процессы тоже будут остановлены, но файлы и установленный софт останется.



# Модифицируем файл таким образом, чтобы создать 3 машины: одну под - балансировщик нагрузки, вторую под базу данных и третью под приложение (пока одну, затем их будет больше).

1. Откройте файл compose.yaml и поместите следующий текст (с учётом ваших параметров сети):

services:

load\_balancer: # Балансировщик нагрузки image: rastasheep/ubuntu-sshd

dns:

- 77.88.8.8

- 8.8.8.8

networks: outside:

ipv4\_address: 192.168.1.100

db: # База данных

image: rastasheep/ubuntu-sshd dns:

- 77.88.8.8

- 8.8.8.8

networks: outside:

ipv4\_address: 192.168.1.200

worker: # Сервер с приложением image: rastasheep/ubuntu-sshd dns:

- 77.88.8.8

- 8.8.8.8

networks: outside:

ipv4\_address: 192.168.1.101

networks: outside:

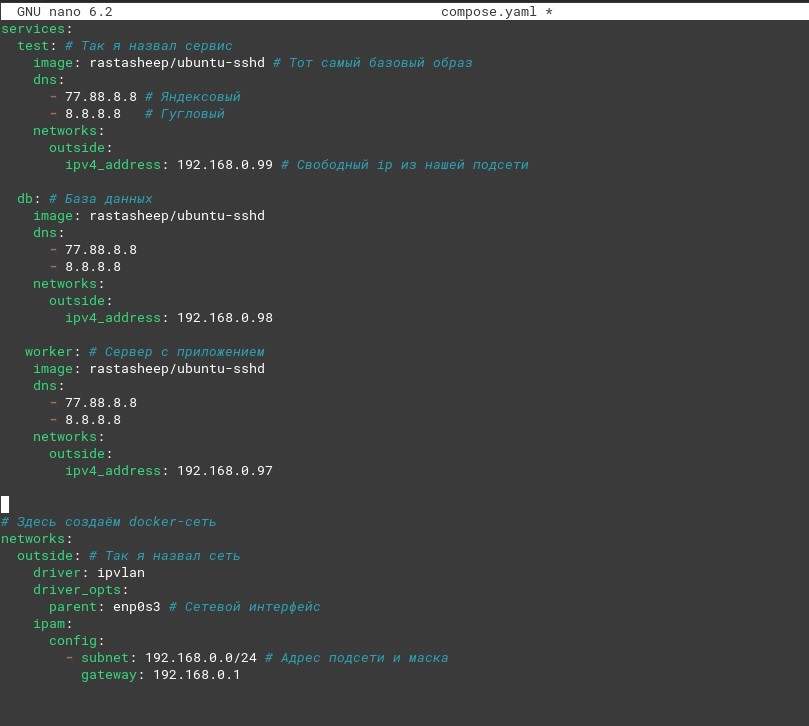
driver: ipvlan driver\_opts:

parent: enp0s3 ipam:

config:

- subnet: 192.168.1.0/24 gateway: 192.168.1.1

Запомните ip-адреса серверов, они будут нужны ansible.

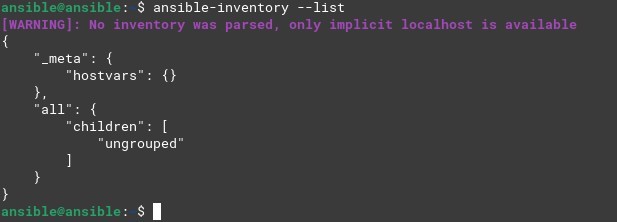


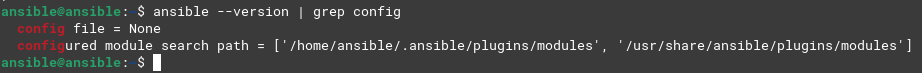
1. Запустите машины командой: docker compose up -d и на некоторое время мы перейдём на машину с ansible.

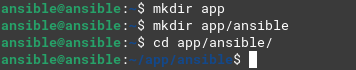


# Соединяем Ansible с управляемыми машинами

1. Выполните команду:

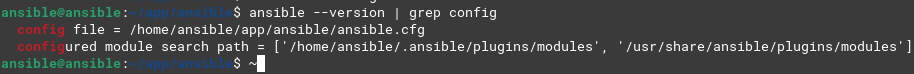


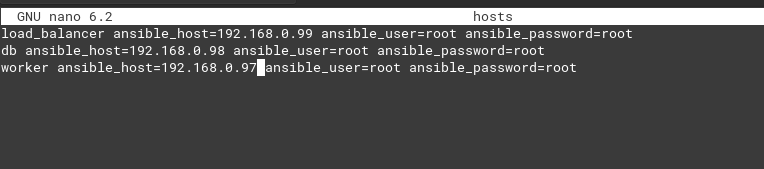
1. Выполните команду:
2. Создайте каталог "app" и в нём "ansible", затем перейдите туда. Это не какие-то специальные названия, просто чтобы была понятная структура.



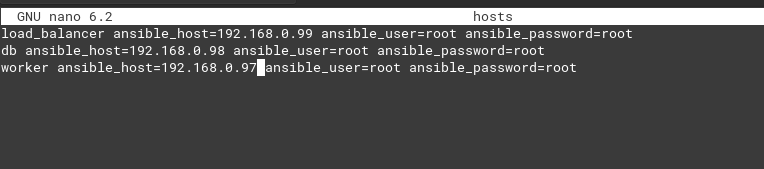
1. Создайте файл "ansible.cfg" содержащий:



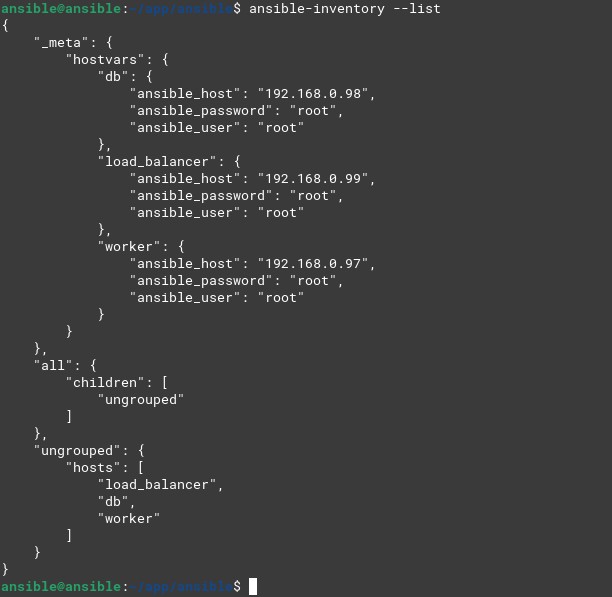
1. Снова выполните команду:
2. В конфиге мы указали, что файл инвентаря лежит в той же папке, что и конфиг и называется "hosts". Обычно так и поступают, но технически можно было положить инвентарь в другое место.



1. Создайте этот файл и напишите в нём следующее (только ip-адреса укажите свои):

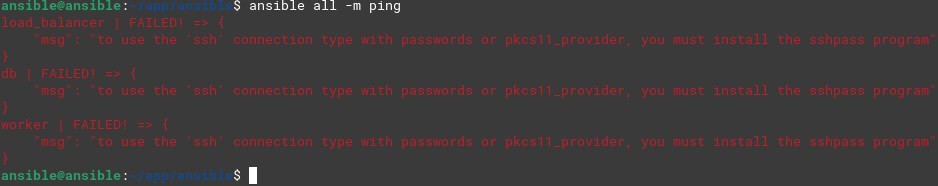


1. Убедитесь, что ansible подхватил информацию о хостах: ansible- inventory —list.

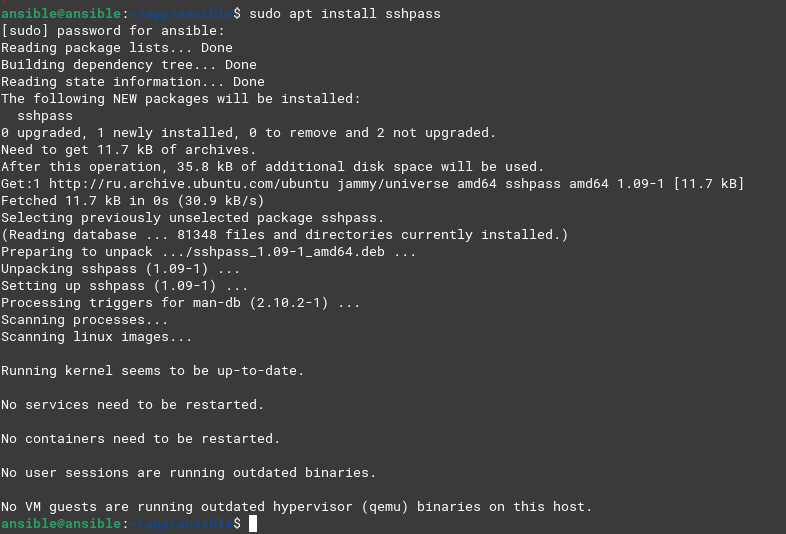


1. Попробуем подключится к ним.

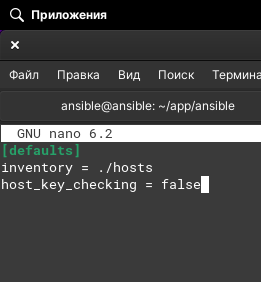
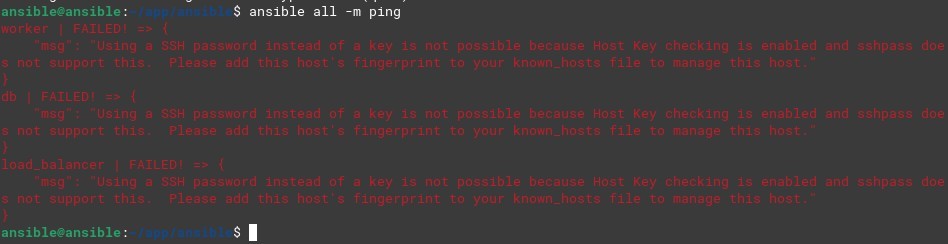
Ansible может выполнять команды в двух режимах: одиночная команда (ad hoc) или сценарий (play). Общий вид запуска одиночной команды выглядит так:

ansible all -m ping

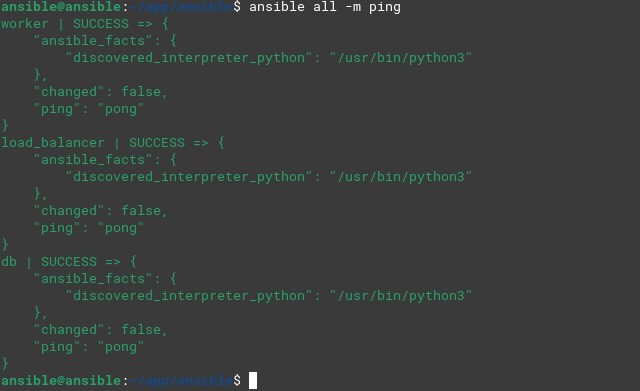
1. В итоге вы должны были получить ошибку, т.к. для авторизации по паролю нужен дополнительный пакет: sshpass. Установите его при помощи apt и повторите ping.



1. В этот раз текст ошибки должен измениться. Суть ошибки в том, что мы не подтвердили, что доверяем этим серверам (в работе №2 мы отвечали yes при первом подключении). Для решения этой проблемы есть 2 основных способа:

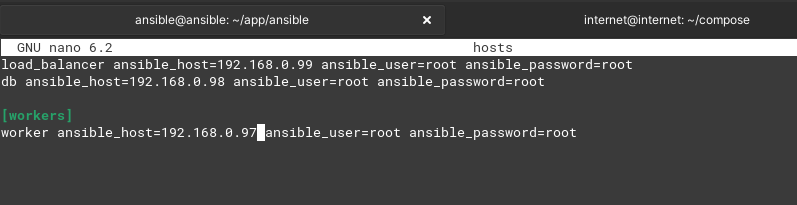


1. В этот раз подключение должно быть успешно, и все три сервера должны ответить "pong".

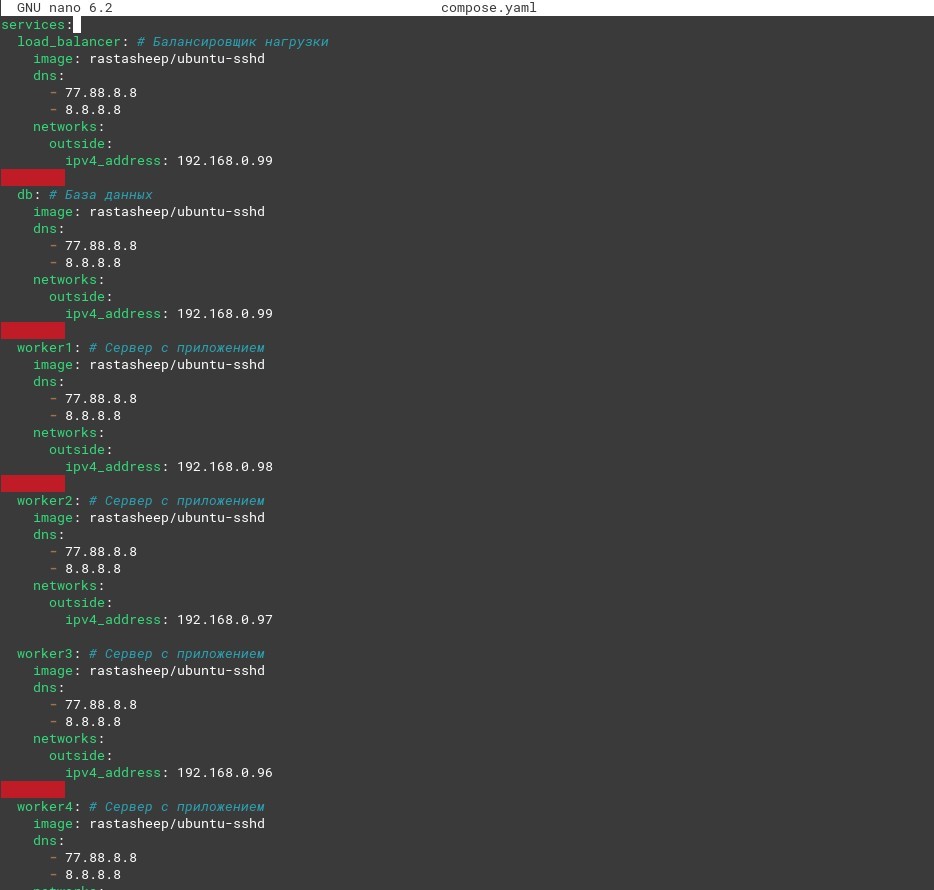


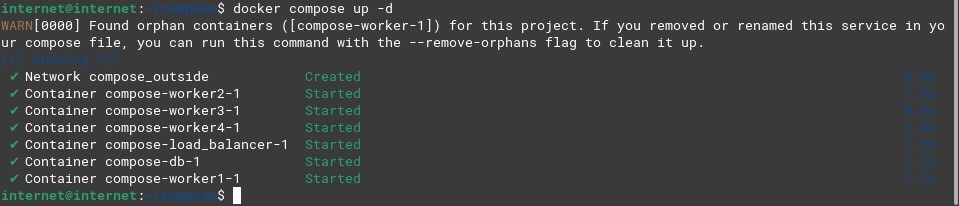
# Разделение на группы

1. Откройте файл "hosts" и замените его на (со своими ip):

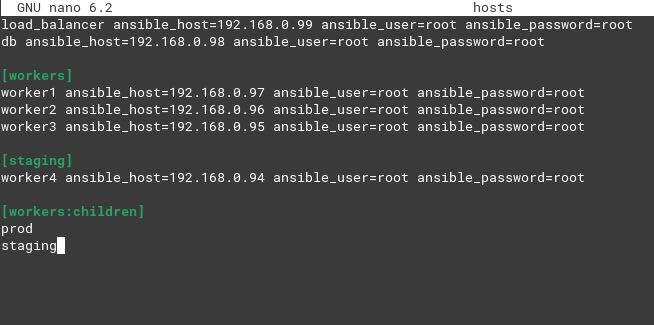


1. Перейдите на виртуальную машину с docker compos, остановите запущенные машины и добавьте в "compose.yaml" ещё 3 машины. В моём случае это будет выглядеть так:

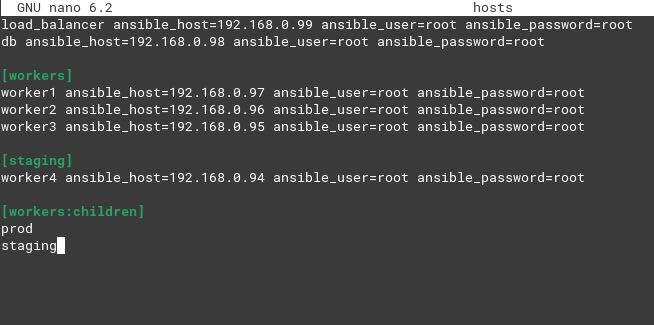


1. Запустите машины и вернитесь на сервер с ansible.
2. Т.к. у нас появилось несколько новых машин, нам нужно добавить их в инвентарь.

Снова откройте файл "hosts" и замените его на (со своими ip):



1. Проверьте, что машины доступны при помощи модуля ansible ping. Прошпигуйте все воркеры и только те, которые находятся в группе "prod".

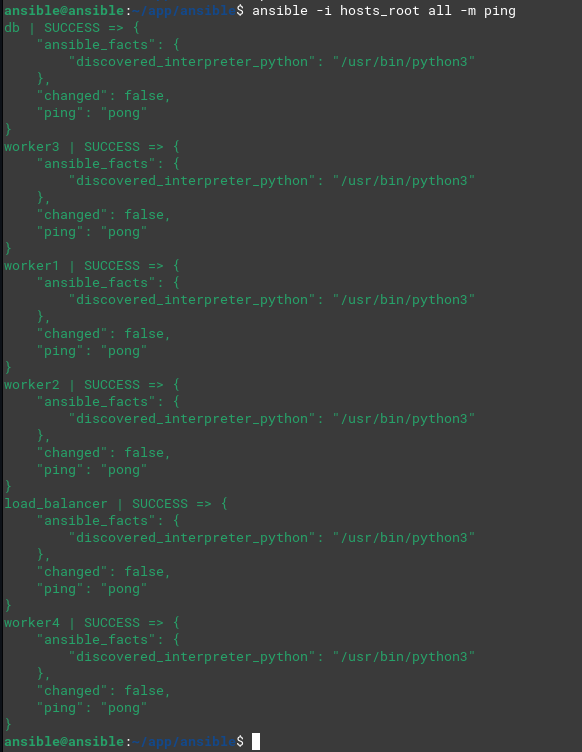


# Изменяем способ доступа к серверам

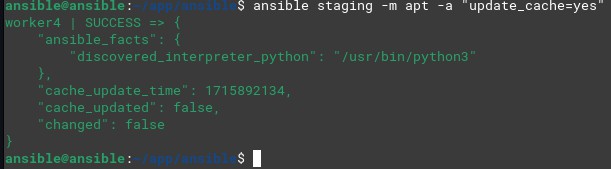
1. Выполните команду:



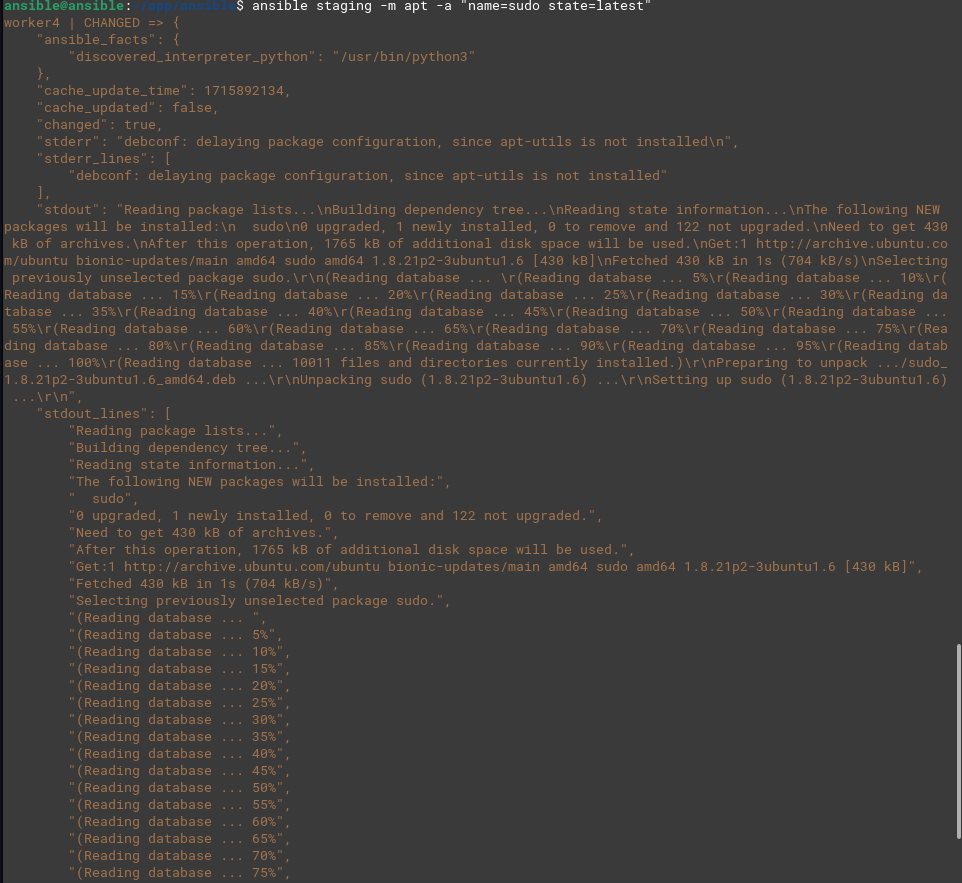
1. Проверьте, что копирование файла прошло успешно. Для этого пропингуйте все сервера командой:



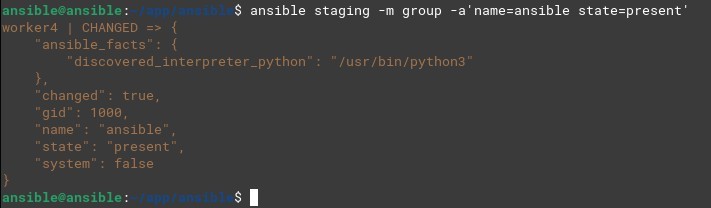
1. Обновим индексы apt-пакетов на сервере, для этого будем использовать модуль ansible [apt](https://docs.ansible.com/ansible/latest/collections/ansible/builtin/apt_module.html):



1. Установим пакет sudo (да, это отдельный пакет, а не встроенная команда):

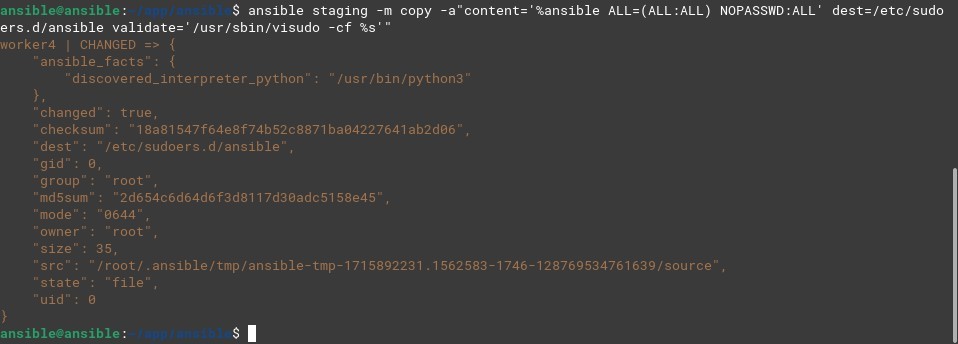


1. По умолчанию новые пользователи обладают минимальными правами, чтобы дать возможность пользователю повышать свои привилегии при помощи sudo он или группа в которой он состоит должны быть записаны в файле "/etc/sudoers" с указанием доступных привилегий. Создадим новую группу с названием "ansible" при помощи модуля [group](https://docs.ansible.com/ansible/latest/collections/ansible/builtin/group_module.html):

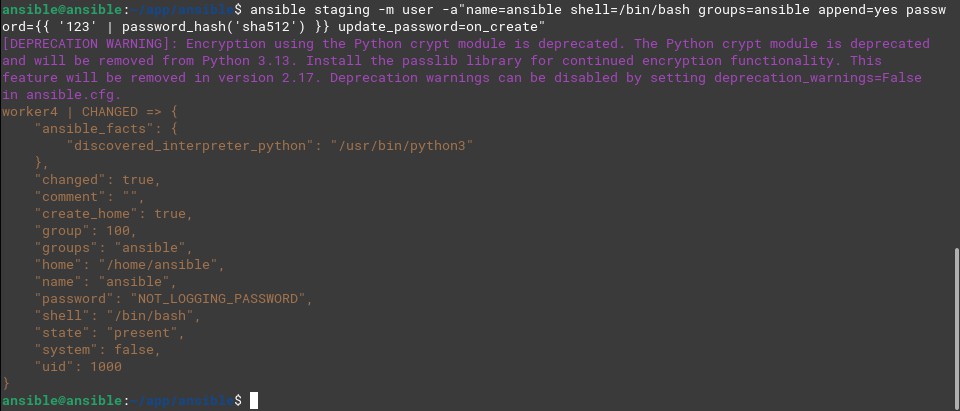


1. Разрешим пользователям из этой группы выполнять любые команды, при этом без необходимости вводить пароль (нужно для ansible).

Чтобы случайно не поломать файл "/etc/sudoers" внося в него изменения, мы создадим новый файл в каталоге "/etc/sudoers.d/" (файл "/etc/sudoers" подтягивает все файлы из этого каталога):

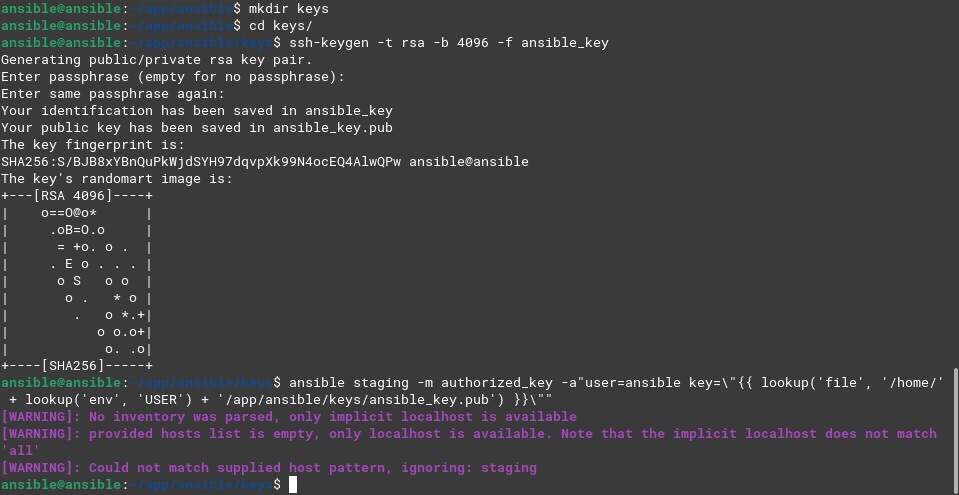


1. Теперь настало время создать пользователя под которым в дальнейшем будет работать ansible:

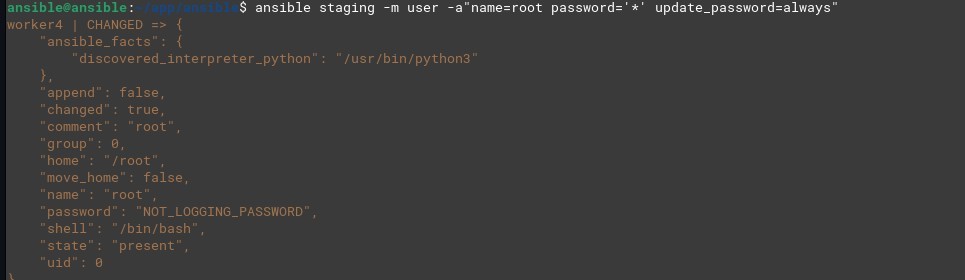


1. Настроим на машинах из группы "staging" доступ под пользователем "ansible" по ключу.

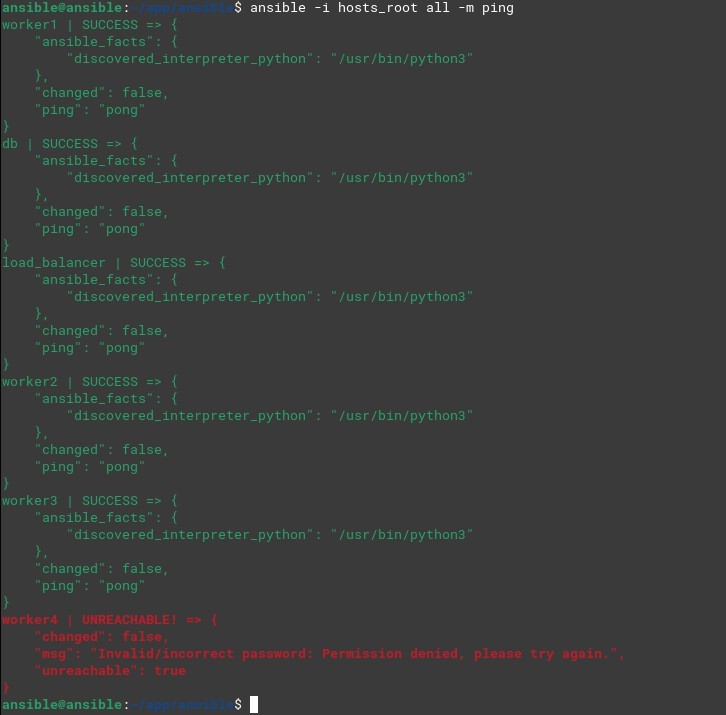
Для этого первым делом нам нужна пара приватный/публичный ключ которую мы будем использовать для доступа. В принципе на нашем сервере уже есть такая пара (она была создана при установке ssh-сервера), но мы создадим ещё одну.



1. Отключим пользователю "root" пароль, чтобы ни у кого не было возможности залогинится под ним:

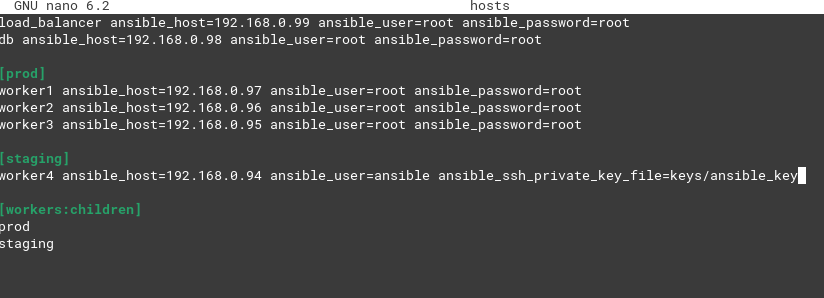


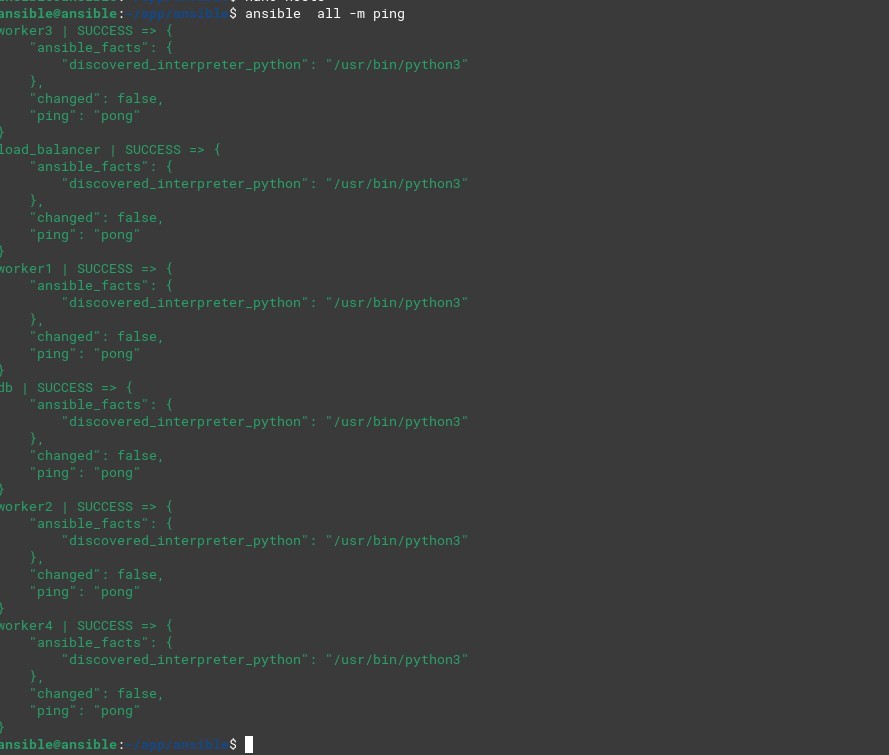
1. Подождите несколько минут и запустите пинг **всех** серверов (через ansible). Вы должны увидеть, что сервер из группы "staging" не ответит.



1. Внесём изменения в файл "hosts", чтобы наладить связь с сервером. Откройте "hosts" в текстовом редакторе и для "worker4" (т.к. он один у нас в группе "staging") замените ansible\_user=root на ansible\_user=ansible и ansible\_password=root на ansible\_ssh\_private\_key\_file=keys/ansible\_key.

Здесь путь к приватному ключу указан относительно расположения файла "hosts".



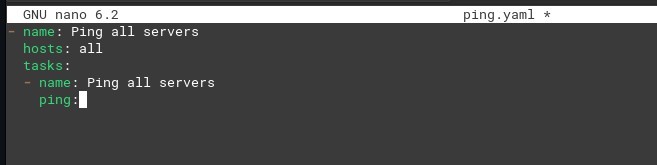
1. Убедитесь, что теперь все сервера отвечают на ping.

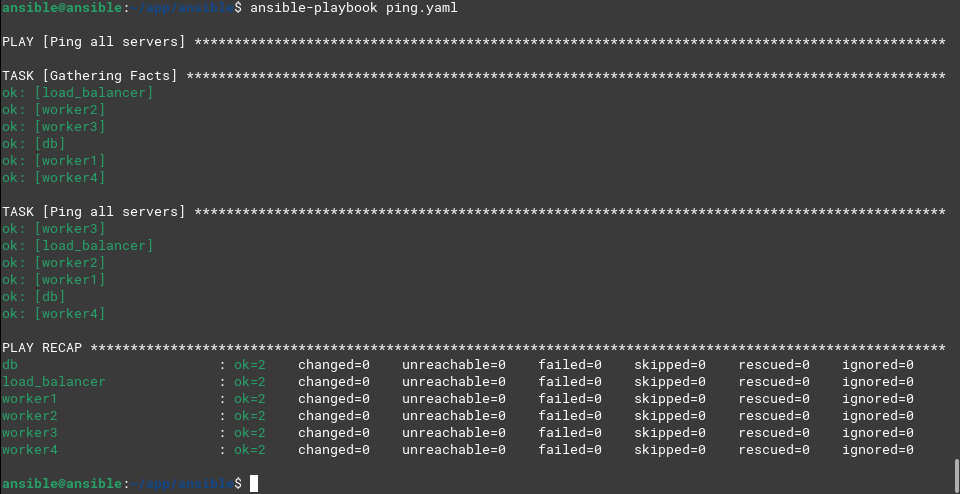
# Пишем сценарий

1. Создайте в каталоге "~/app/ansible" файл "ping.yaml";
2. Добавьте в файл следующие строки:



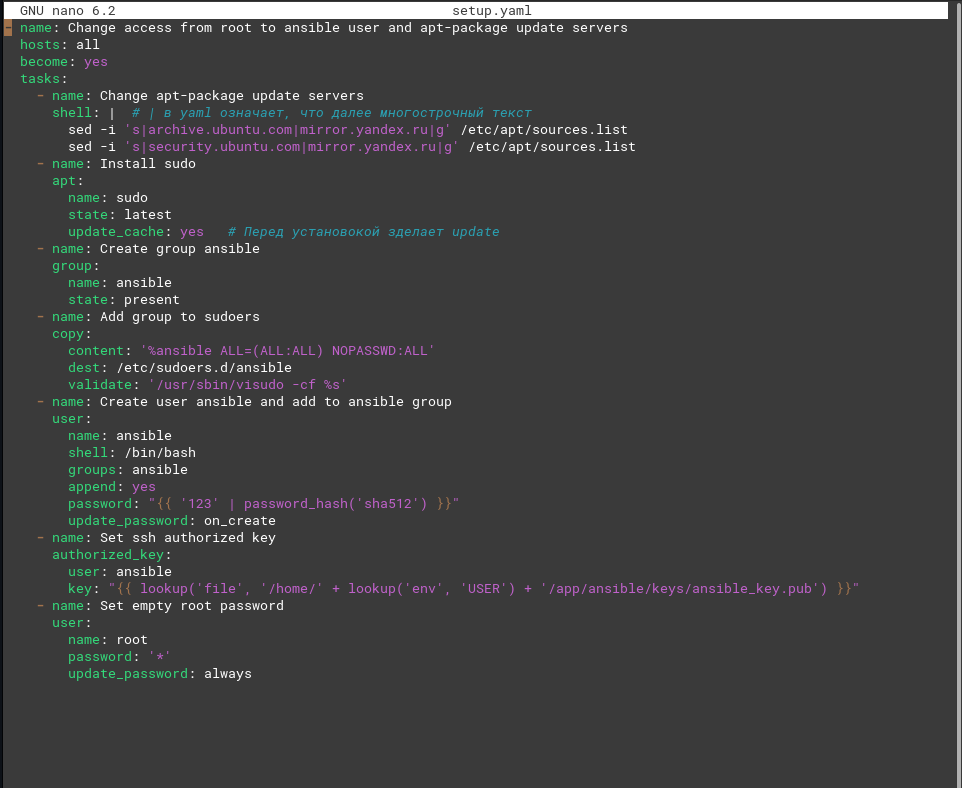
1. Теперь добавим непосредственно команды:



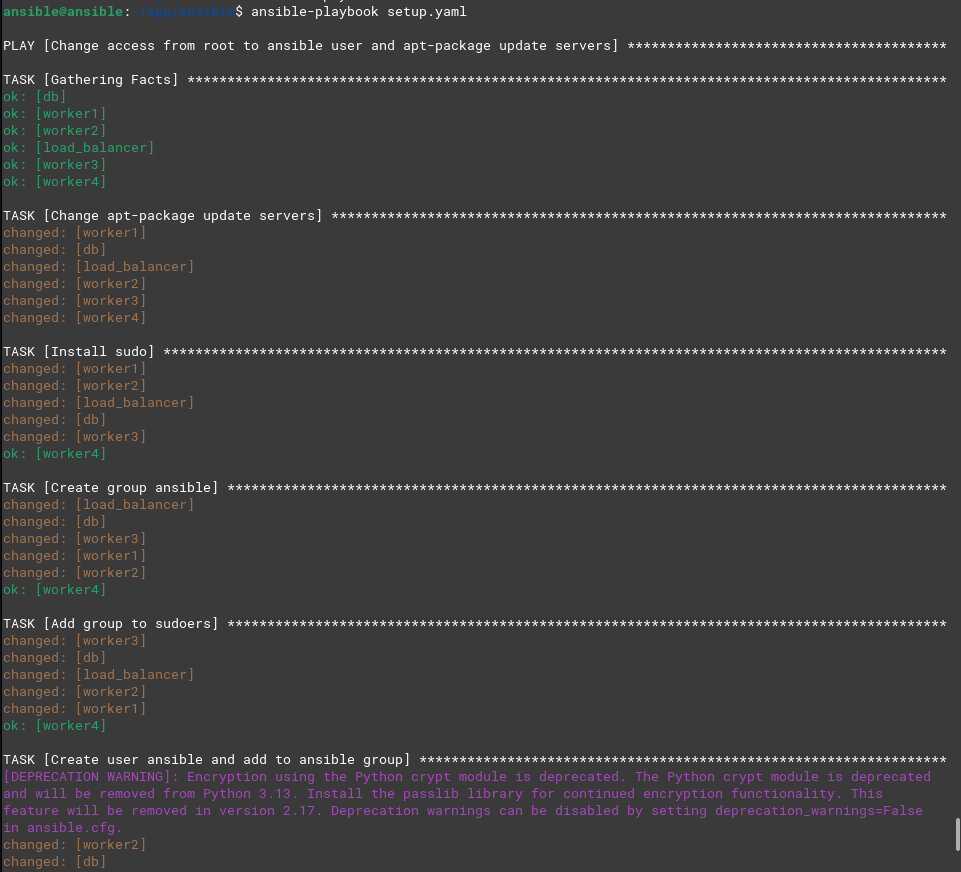
1. Запустите palybook при помощи команды:

# Теперь напишем palybook который будет делать ровно тоже, что мы делали руками в предыдущем разделе:

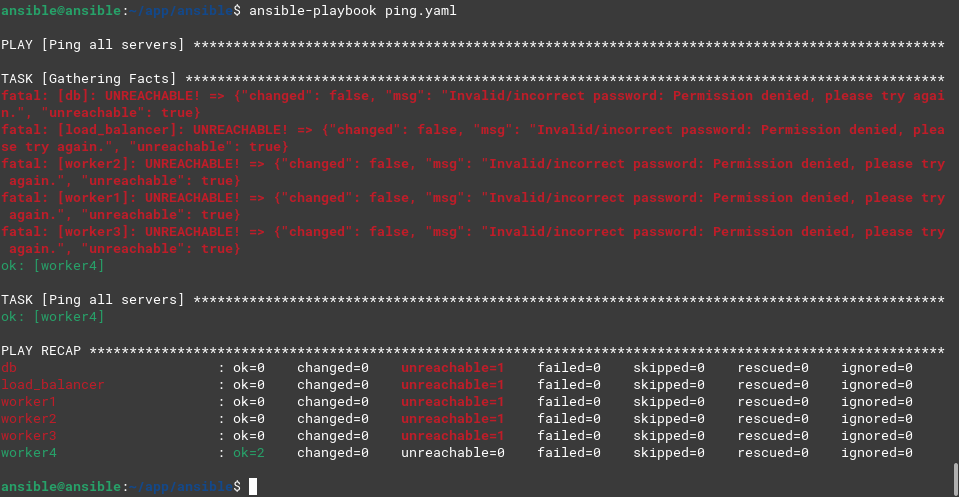
1. Создайте в каталоге "~/app/ansible" файл "setup.yaml";
2. Добавьте в файл следующие строки:
3. Теперь добавим непосредственно команды. Сравните ad hoc команды из предыдущего раздела с их вариантами в palybook:



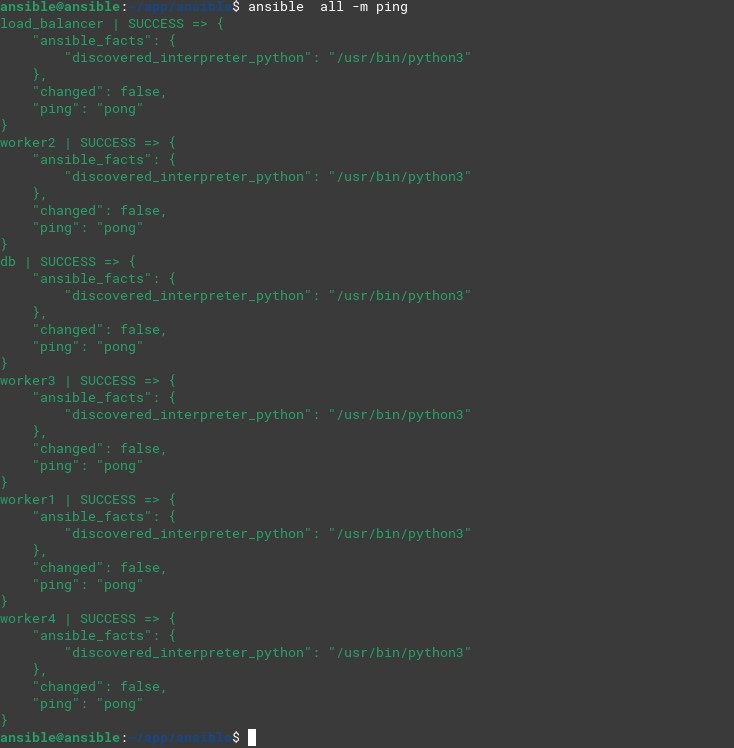
1. Запустите palybook и дождитесь его завершения. В логах вы должны увидеть, что на всех серверах кроме "worker4" на каждом этапе происходили изменения (changed), а на "worker4" изменений не было (ok).



1. Подождите несколько минут и запустите "ping.yaml". Теперь, все сервера, кроме "worker4" перестали отвечать, т.к. в hosts указаны данные для пользователя root.

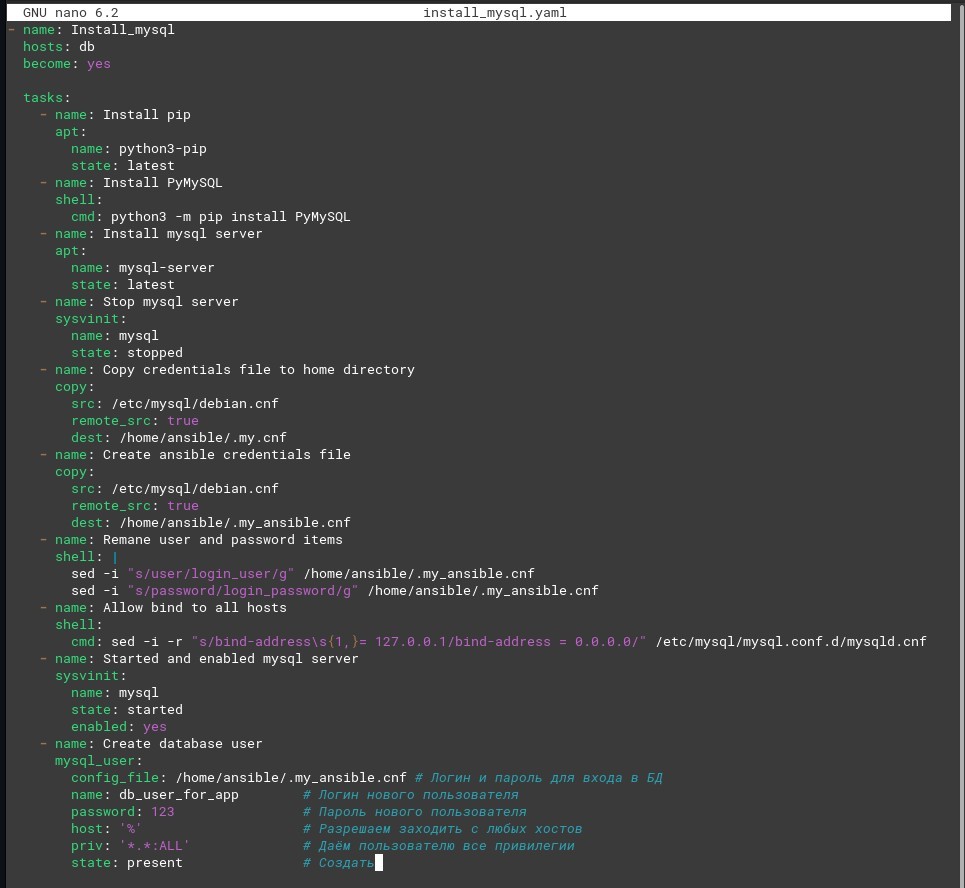


1. Поправьте файл hosts и убедитесь, что теперь все сервера на связи.

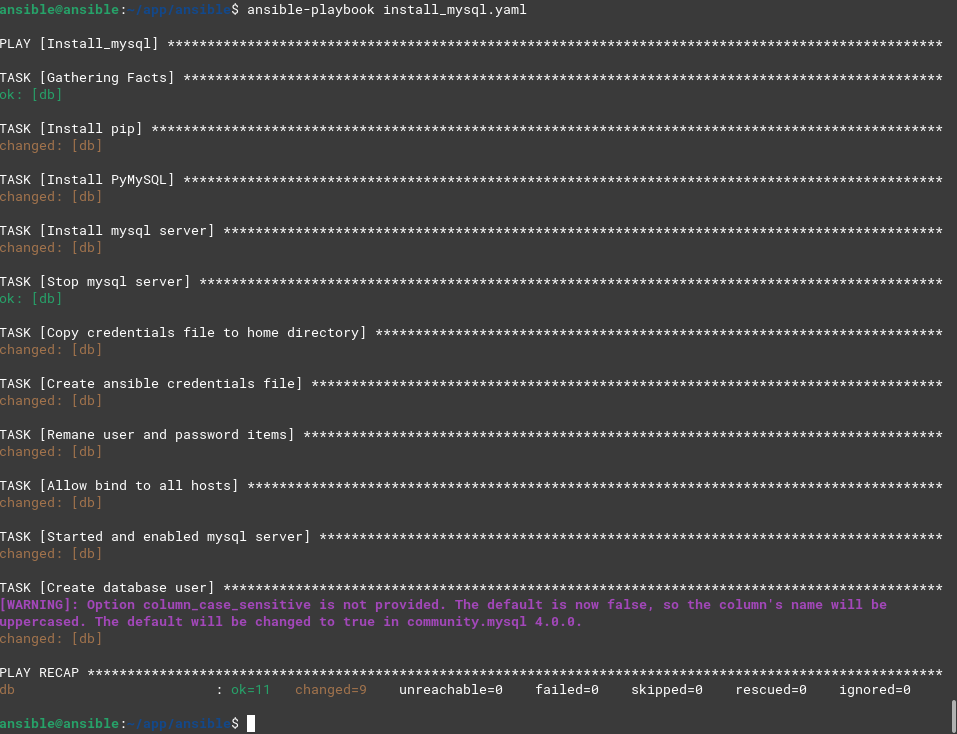


# Установка софта на сервер базы данных

1. Создайте в каталоге "~/app/ansible" файл "install\_mysql.yaml";
2. Добавьте в файл следующие строки:

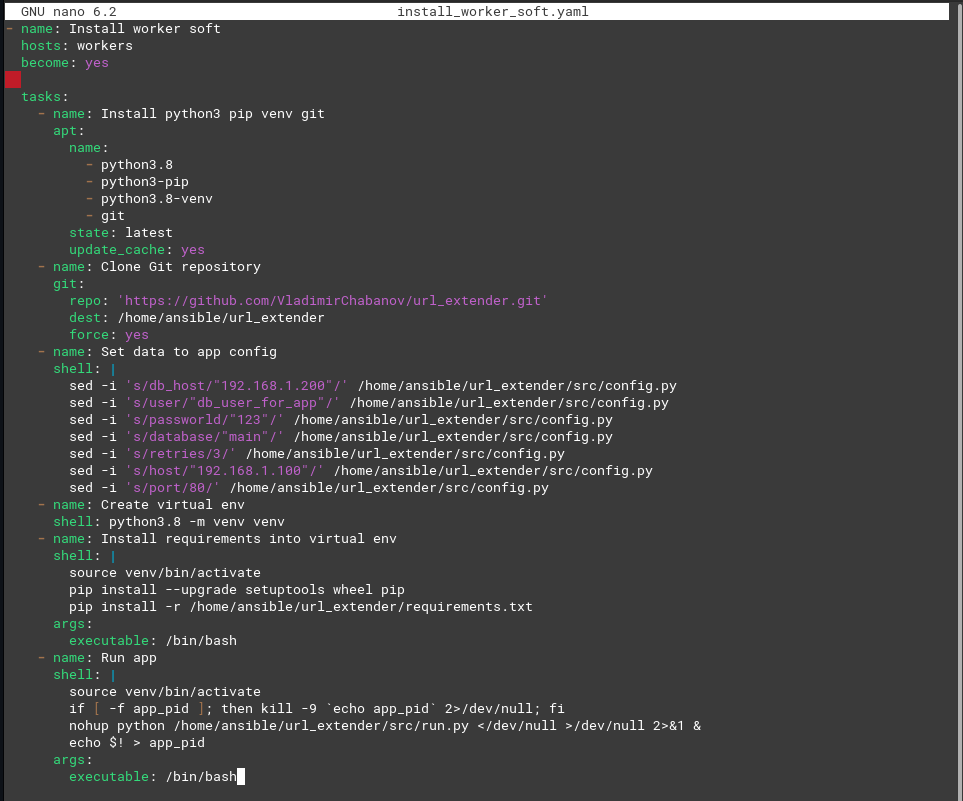


1. Запустите плейбук и дождитесь завершения.

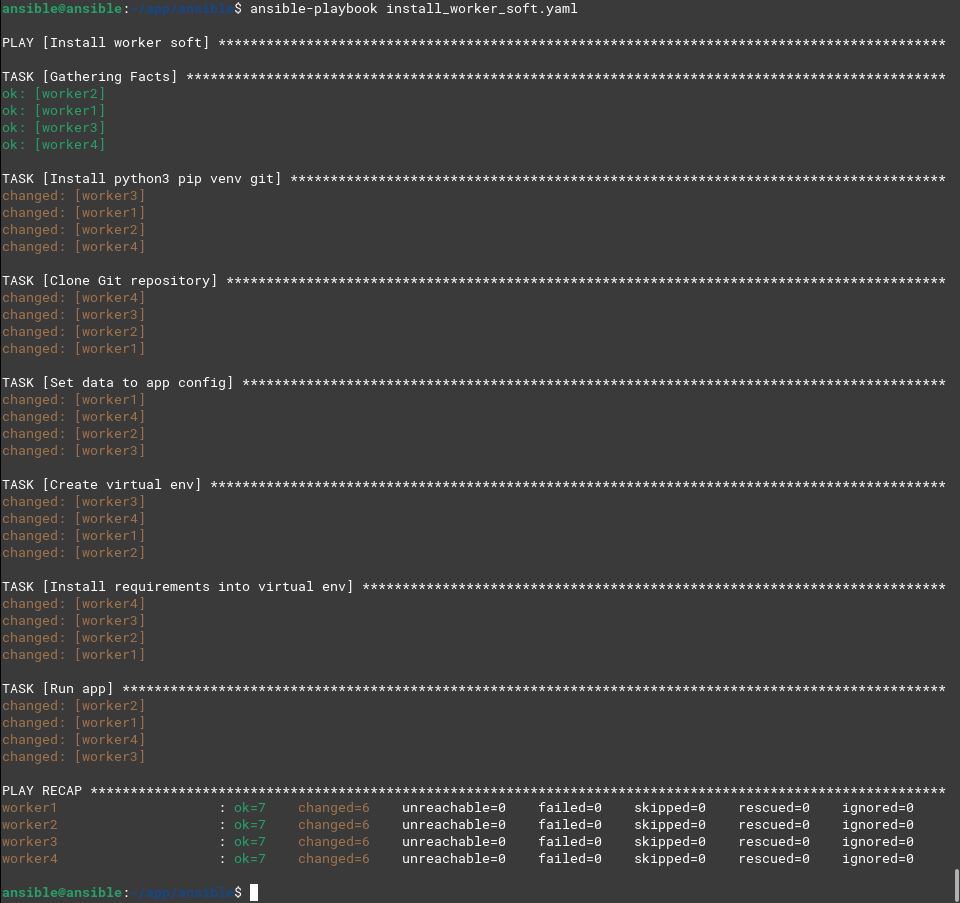


# Установка софта на рабочие сервера

1. Создайте в каталоге "~/app/ansible" файл "install\_worker\_soft.yaml";
2. Добавьте в файл следующие строки (с учётом своих ip-адресов):

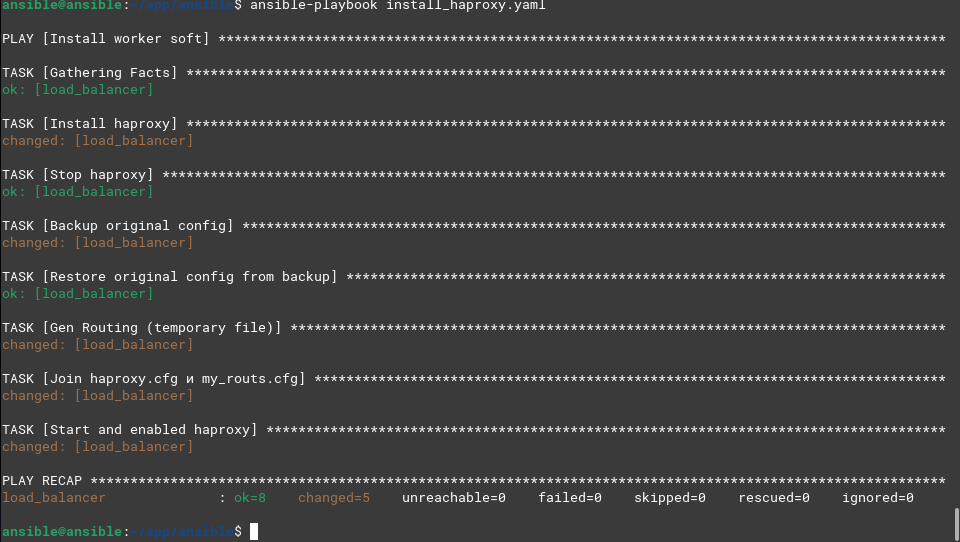


1. Запустите playbook и дождитесь установки.



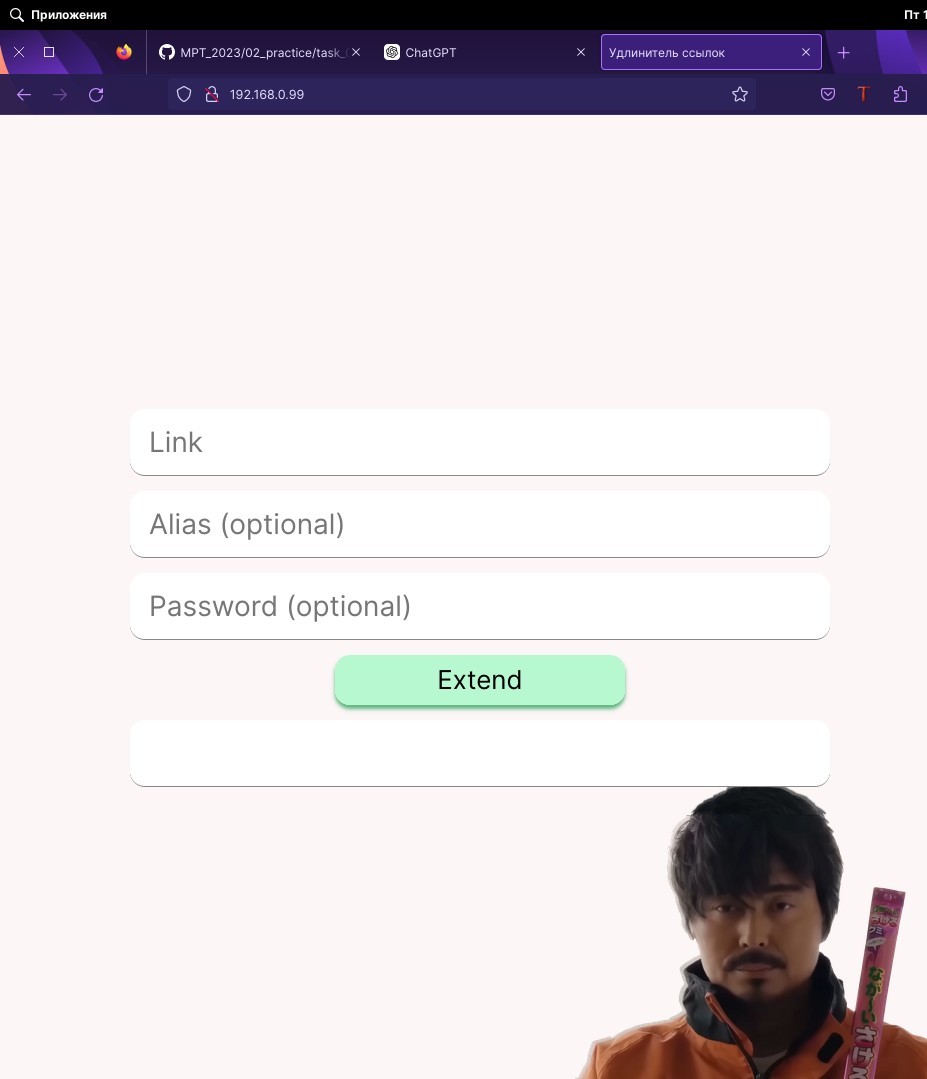
# Установка софта на сервер балансировщика нагрузки

1. Создайте в каталоге "~/app/ansible" файл "install\_haproxy.yaml";
2. Добавьте в файл следующие строки (с учётом своих ip-адресов):

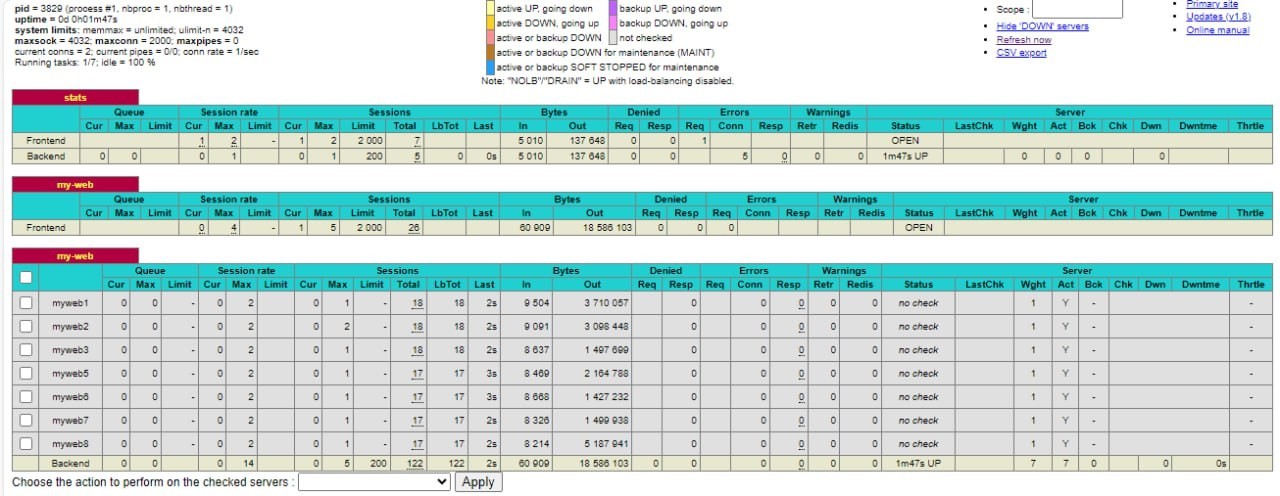


# Теперь нужно убедится, что приложение развёрнуто на серверах и корректно работает.

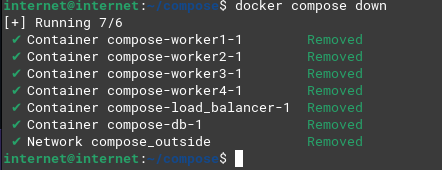
1. Откройте браузер на основной операционной системе, или на другом устройстве подключённом к той же сети, и введите ip-адрес сервера "load\_balancer" (у меня это 192.168.1.100).



1. Обновите несколько раз страницу и попробуйте воспользоваться функционалом приложения.
2. Зайдите на страницу "/haproxy\_stats" (порт 81) и введите учётные данные.
3. Изучите статистику распределения запросов к рабочим серверам столбец **Sessions** -> **Total** в нижней таблице. Количество запросов ко всем серверам должно быть примерно одинаковым, т.к. установлена политика балансировки roundrobin (карусель).

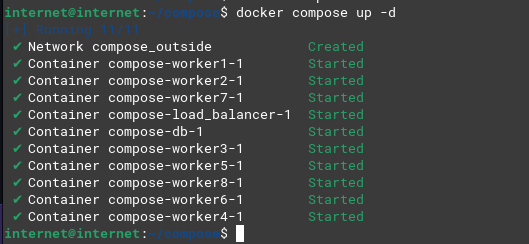


# Теперь увеличим количество рабочих серверов.

1. Перейдите на виртуальную машину с docker compose и остановите запущенные контейнеры;
2. Добавьте в файл compose.yaml дополнительные сервера, таки образом, чтобы *рабочих*

серверов было 8;



1. Запустите их при помощи docker compose up -d. Все машины будут чистыми с настройками по умолчанию;
2. Добавьте новые сарваера в файл hosts (учтите, что теперь там снова пользователь root), запустите начальное конфигурирование и установку всего необходимого софта;
3. В конфиге балансировщика, в раздел "backend" добавьте новые рабочие сервера;
4. Протестируйте работоспособность приложения, можно просто обновить вкладку раз 20-30;
5. Зайдите на страницу "/haproxy\_stats" и проверьте, что все сервера работают и загружены равномерно.

# Ответы на вопросы.

1. Команде `ansible-playbook` можно указать файл с серверами (hosts) несколькими способами:
   1. Указать имя файла с хостами через параметр `-i` (inventory):

```

ansible-playbook -i hosts\_file.txt playbook.yml

```

Здесь `hosts\_file.txt` - файл с хостами, а `playbook.yml` - файл с плейбуком.

* 1. Указать имя файла с хостами в конфигурационном файле `ansible.cfg`:

``` [defaults]

inventory = hosts\_file.txt

```

В этом случае можно выполнять команду без указания параметра `-i`:

```

ansible-playbook playbook.yml

```

1. При условии, что доступ по ssh уже есть, на Linux-серверах, которыми планируется управлять при помощи Ansible, нужно установить программное

обеспечение, которое поддерживает Ansible. Это, в основном, Python. Python должен быть установлен на серверах, а также на компьютере, с которого выполняются команды Ansible.

1. В логах выполнения playbook-а показатель "changed" в разделе "PLAY RECAP" означает, что состояние хоста было изменено. Это может быть изменение файлов, установка пакетов и другие операции. Если значение "changed" равно 0, значит состояние хоста не было изменено.
2. Ad hoc команда ansible для создания пользователя "thor", который состоит только в группах "avengers" и "asgard" с паролем "Strongest Avenger" на хостах из группы "stark\_industries":

```

ansible stark\_industries -m user -a "name=thor groups='avengers,asgard' password=Strongest Avenger"

```

Здесь `stark\_industries` - группа хостов, `user` - модуль для работы с пользователями, `name`, `groups` и `password` - параметры модуля.

1. Playbook для удаления половины файлов из каталога "/lifeforms" на серверах из группы "universe":

```

---

- hosts: universe tasks:

- name: Delete half of the files file:

path: "/lifeforms/{{ item }}" state: absent

when: item.endswith('1') or item.endswith('2') loop: "{{ query('fileglob', '/lifeforms/\*') }}"

when: query('fileglob', '/lifeforms/\*') | length % 2 == 0

```

Здесь `hosts: universe` указывает, что плейбук будет выполнен на серверах из группы "universe". `file` - модуль для работы с файлами, `path: "/lifeforms/{{ item }}"` - путь к файлу, `state: absent` - удаляем файл. `when: item.endswith('1') or item.endswith('2')` - условие для удаления файлов, заканчивающихся на 1 или 2. `loop: "{{ query('fileglob', '/lifeforms/\*') }}"` - цикл по всем файлам в каталоге "/lifeforms". `when: query('fileglob', '/lifeforms/\*') | length % 2 == 0` - условие для проверки количества файлов в каталоге, оно должно быть чётным.

# Вывод:

Я ознакомился на практике с инструментом для удаленного управления конфигурациями Ansible.