

## Задание

Необходимо развернуть микросервис на виртуальной машине (ВМ), используя систему управления конфигурациями и описать выполненные действия.

1. В качестве гипервизора 2-го типа разрешается использовать любой, по своему усмотрению: VMware Workstation, VirtualBox, HyperV, Qemu, Vagrant и т.п. Операционная система для ВМ: RedHat-подобные системы, например, Rocky Linux, AlmaLinux и т.д.
2. Микросервис должен доставляться на виртуальную машину с помощью системы управления конфигурациями, после чего микросервис должен работать в фоне на ВМ. Необходимая CMS (Configuration Management Systems / Система Управления Конфигурациями): Ansible.
3. Микросервис: HTTP сервер, который экспортирует на 8080 порт Prometheus метрики. Одна из метрик должна предоставлять данные о том, на какой типе хоста запущен сервер: виртуальная машина, контейнер или физический сервер. Для написания можно использовать любой язык программирования (желательно, Python или Golang). Рекомендуется использовать готовые библиотеки для поднятия HTTP сервера и проброса Prometheus метрик.
4. В браузере на виртуальной машине по адресу <http://localhost:8080> должны отображаться соответствующие Prometheus метрики.

## 1. Настройка хоста и ВМ

### Гипервизор:

В качестве гипервизора 2-го типа я буду использовать VMware Workstation

### Операционная система для ВМ:

CentOS-Stream-9

Данная ОС была выбрана мной, так как является RedHat-подобной системой, и... на ПК имелся ее образ :)

### Настройка ОС в гипервизоре

При создании ВМ я пользовался New VM wizard, встроенным в VMware Workstation



Рисунок 1 – New VM wizard

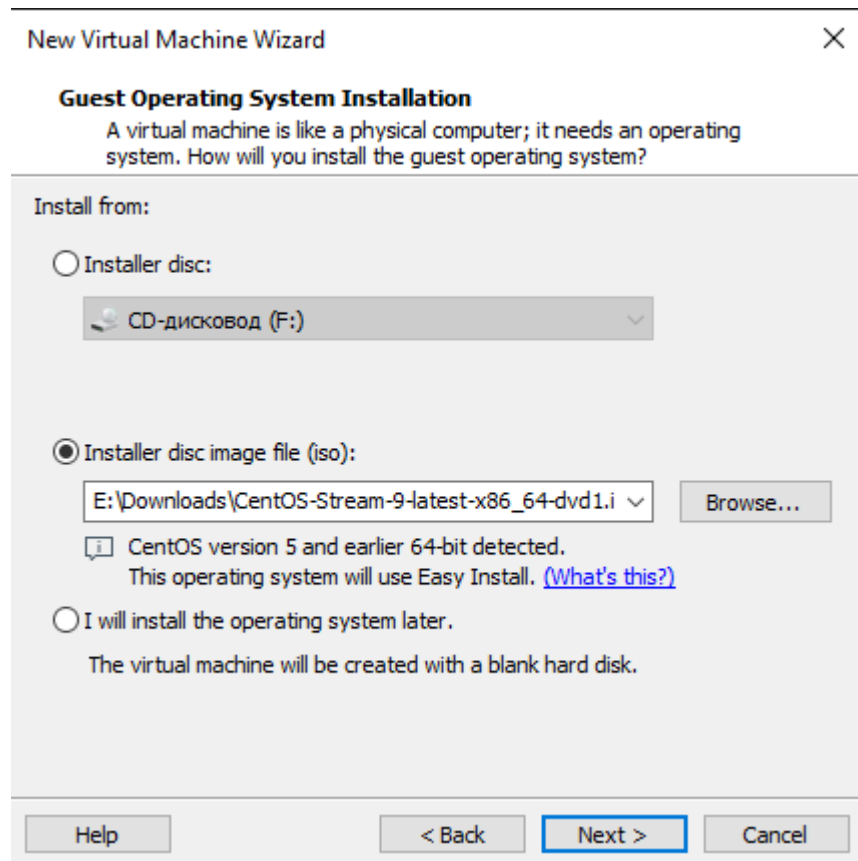


Рисунок 2 – Выбор образа

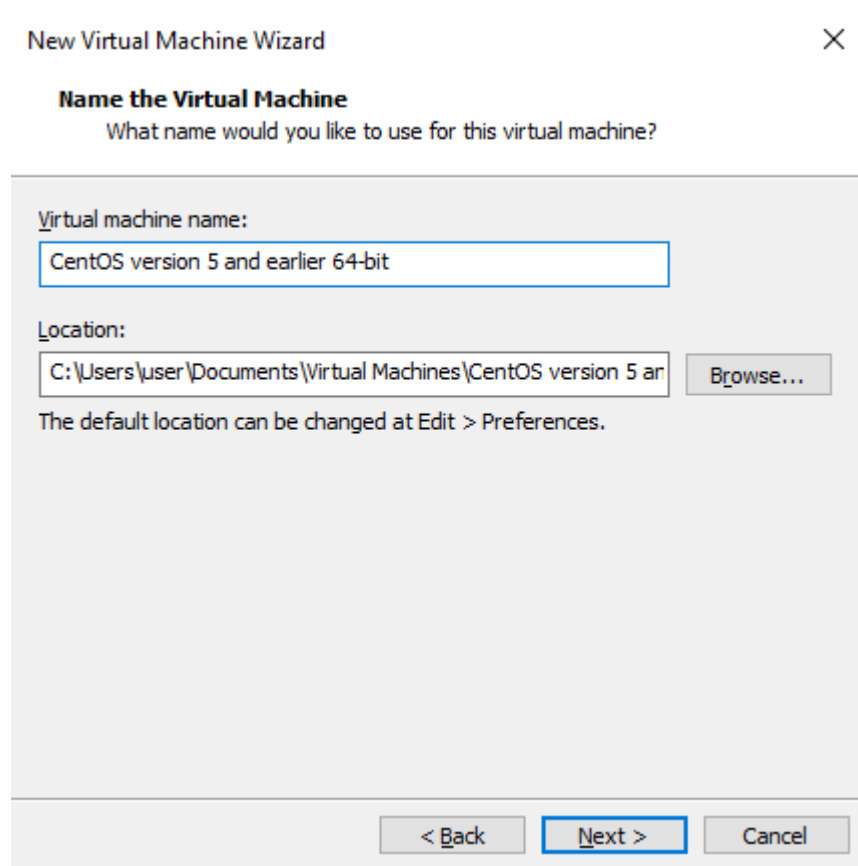


Рисунок 3 – Выбор расположения будущей ВМ

По неизвестной мне причине VMware Workstation определил образ как CentOS version 5 and earlier 64-bit. В дальнейшем это не повлияет на работу ВМ.

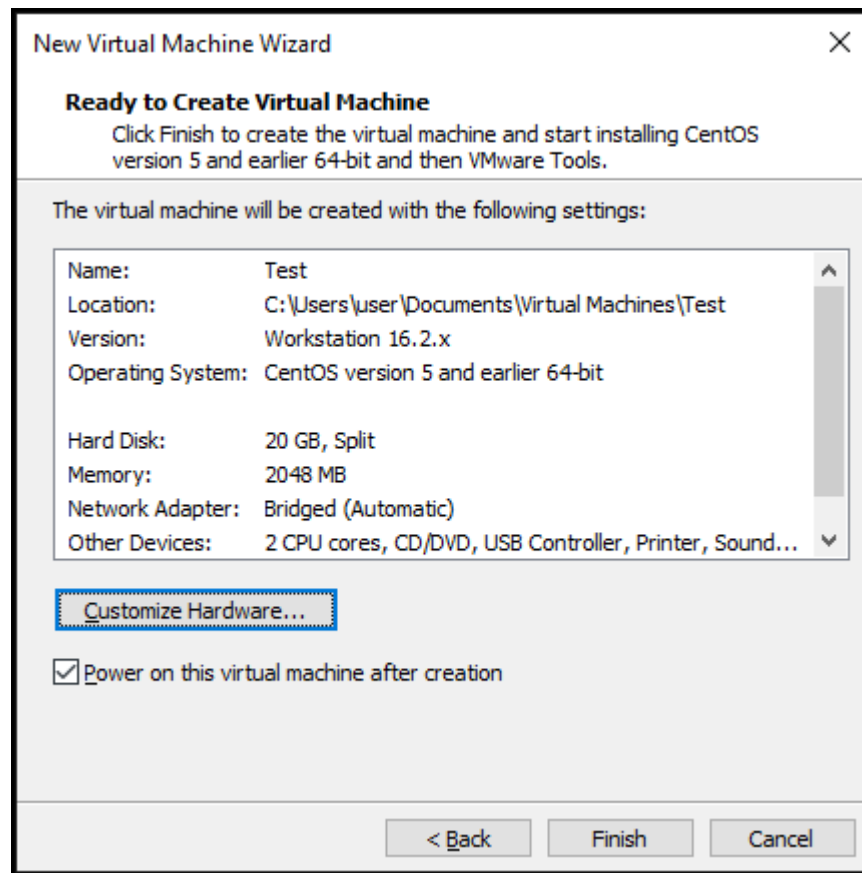
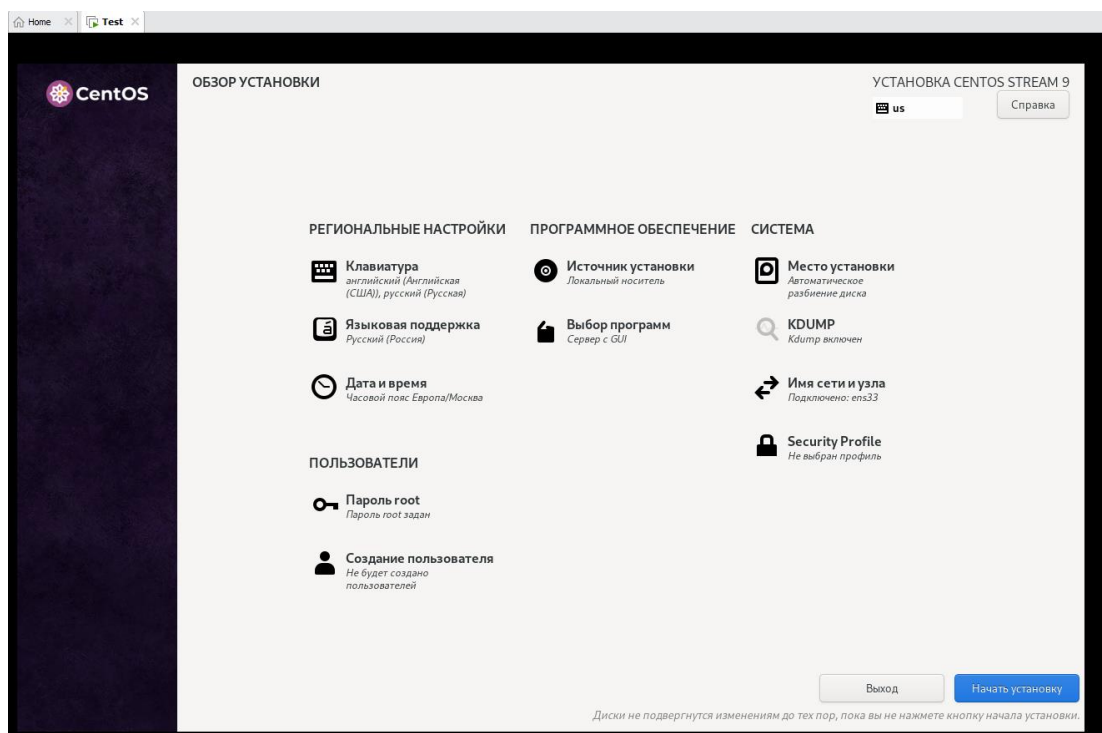


Рисунок 4 – Параметры ВМ



## Рисунок 5 – Установщик

После запуска установщика появляется время, чтобы настроить хост. Так как я использую машину под управлением Windows для дальнейшего выполнения задания (Ansible не работает нативно на Windows) мне необходимо установить WSL (Windows Subsystem for Linux).

Для его установки воспользуюсь Powershell с правами админа и командой:

***wsl –install***



## Рисунок 6 – Ярлык WSL

После установки WSL и запуска Linux-дистрибутива (Ubuntu), обновлю его и установлю Ansible:

***sudo apt update && sudo apt upgrade -y***

***sudo apt install ansible -y***



```
[gleb@localhost ~]$ ip addr | grep inet
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
    inet 192.168.0.107/24 brd 192.168.0.255 scope global dynamic noprefixroute ens33
    inet6 fe80::20c:29ff:fe4f:1b7e/64 scope link noprefixroute
[gleb@localhost ~]$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.0.107 netmask 255.255.255.0  broadcast 192.168.0.255
    inet6 fe80::20c:29ff:fe4f:1b7e prefixlen 64  scopeid 0x20<link>
    ether 00:0c:29:4f:1b:7e  txqueuelen 1000  (Ethernet)
    RX packets 19916  bytes 14614641 (13.9 MiB)
    RX errors 0  dropped 1  overruns 0  frame 0
    TX packets 5992  bytes 800326 (781.5 KiB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128  scopeid 0x10<host>
    loop txqueuelen 1000  (Local Loopback)
    RX packets 469  bytes 44129 (43.0 KiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 469  bytes 44129 (43.0 KiB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

[gleb@localhost ~]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:4f:1b:7e brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.0.107/24 brd 192.168.0.255 scope global dynamic noprefixroute ens33
        valid_lft 5220sec preferred_lft 5220sec
    inet6 fe80::20c:29ff:fe4f:1b7e/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[gleb@localhost ~]$ s
```

Рисунок 8 – Получение IP VM с помощью команд в терминале

IP моей VM: 192.168.0.107

Так же добавлю автозапуск SSH при загрузке системы и сразу запущу его(Он нужен, так-как Ansible использует этот протокол для связи с VM)

***sudo systemctl enable sshd***

***sudo systemctl start sshd***

***sudo systemctl status sshd***

```

[gleb@localhost ~]$ sudo systemctl enable sshd
sudo systemctl start sshd
[gleb@localhost ~]$ sudo systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; preset: enabled)
   Active: active (running) since Sat 2025-04-26 15:31:57 MSK; 1h 45min ago
     Docs: man:sshd(8)
           man:sshd_config(5)
    Main PID: 994 (sshd)
      Tasks: 1 (limit: 10709)
     Memory: 2.2M
        CPU: 389ms
    CGroup: /system.slice/sshd.service
            └─994 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Apr 26 15:57:26 localhost.localdomain sshd[10357]: pam_unix(sshd:session): session opened for user root(uid=0) by root(uid=0)
Apr 26 15:59:15 localhost.localdomain sshd[13082]: Accepted password for root from 192.168.0.110 port 49898 ssh2
Apr 26 15:59:15 localhost.localdomain sshd[13082]: pam_unix(sshd:session): session opened for user root(uid=0) by root(uid=0)
Apr 26 16:04:55 localhost.localdomain sshd[14567]: Accepted password for root from 192.168.0.110 port 49922 ssh2
Apr 26 16:04:55 localhost.localdomain sshd[14567]: pam_unix(sshd:session): session opened for user root(uid=0) by root(uid=0)
Apr 26 16:08:17 localhost.localdomain sshd[16039]: Accepted password for root from 192.168.0.110 port 49910 ssh2
Apr 26 16:08:17 localhost.localdomain sshd[16039]: pam_unix(sshd:session): session opened for user root(uid=0) by root(uid=0)
Apr 26 17:11:11 localhost.localdomain sshd[18463]: Connection closed by authenticating user gleb 192.168.0.110 port 49932 [preauth]
Apr 26 17:11:32 localhost.localdomain sshd[18468]: Accepted password for root from 192.168.0.110 port 49934 ssh2
Apr 26 17:11:32 localhost.localdomain sshd[18468]: pam_unix(sshd:session): session opened for user root(uid=0) by root(uid=0)
[gleb@localhost ~]$

```

## Рисунок 9 – Проверка SSH сервера

На данный момент у нас есть работающая ВМ с SSH сервером и хост, на котором установлен Ansible с помощью WSL



## 2.Ansible

### Проект

Необходимо создать директорию для проекта:

```
mkdir -p ~/microservice_ansible
```

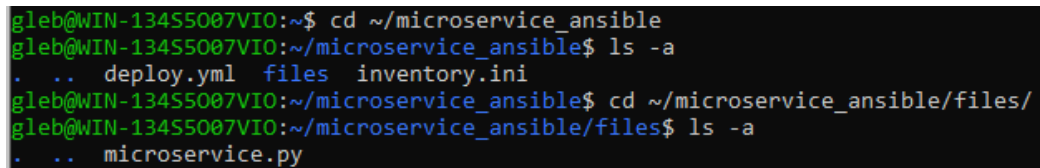
```
cd ~/microservice_ansible
```

И файлы, которые мы будем использовать:

```
nano files/microservice.py
```

```
nano deploy.yml
```

```
nano inventory.ini
```



```
gleb@WIN-134S5007VIO:~$ cd ~/microservice_ansible
gleb@WIN-134S5007VIO:~/microservice_ansible$ ls -la
.  ..  deploy.yml  files  inventory.ini
gleb@WIN-134S5007VIO:~/microservice_ansible$ cd ~/microservice_ansible/files/
gleb@WIN-134S5007VIO:~/microservice_ansible/files$ ls -la
.  ..  microservice.py
```

Рисунок 10 – Директория и файлы в ней

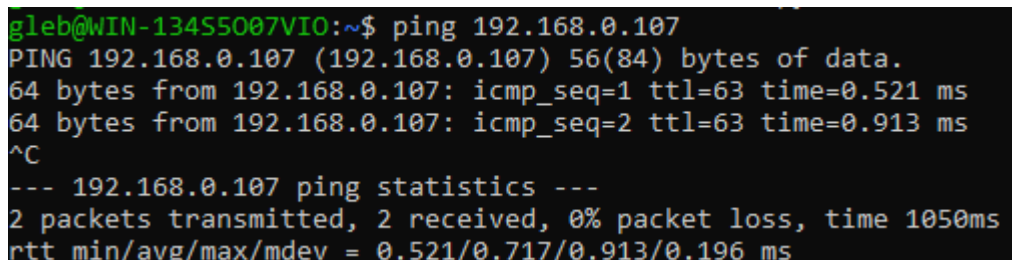
### Доступ к ВМ:

Для доступа к виртуальной машине необходимо заполнить `inventory.ini` файл. Это конфигурационный файл Ansible, в котором указывается список управляемых хостов (серверов, виртуальных машин, сетевых устройств) и переменные для хостов/групп (IP-адреса, порты, учетные данные и т.д.).

```
[vm]
```

```
192.168.0.107      ansible_user=root      ansible_password=211112
```

```
ansible_python_interpreter=/usr/bin/python3
```



```
gleb@WIN-134S5007VIO:~$ ping 192.168.0.107
PING 192.168.0.107 (192.168.0.107) 56(84) bytes of data.
64 bytes from 192.168.0.107: icmp_seq=1 ttl=63 time=0.521 ms
64 bytes from 192.168.0.107: icmp_seq=2 ttl=63 time=0.913 ms
^C
--- 192.168.0.107 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1050ms
rtt min/avg/max/mdev = 0.521/0.717/0.913/0.196 ms
```

Рисунок 11 – Проверка связи с ВМ

### Ansible playbook для доставки микросервиса

Листинг кода `deploy.yml` будет представлен в приложении 1

В общем и целом Playbook в Ansible — это YAML-файл, содержащий набор инструкций для автоматизации развертывания и настройки инфраструктуры. В моем случае playbook будет копировать код микросервиса на виртуальную машину, устанавливать зависимости (Python, библиотеки) и настраивать systemd-сервис для автоматического запуска.

### **Микросервис**

Из задания: микросервис - HTTP сервер, который экспортирует на 8080 порт Prometheus метрики. Одна из метрик должна предоставлять данные о том, на каком типе хоста запущен сервер: виртуальная машина, контейнер или физический сервер

Листинг кода *microservice.py* будет представлен в приложении 1

### 3. Запуск и отладка

#### Запуск playbook

На хосте выполняем команду

*ansible-playbook -i inventory.ini deploy.yml*

```
gleb@WIN-134S5007VI0:~$ cd ~/microservice_ansible/
gleb@WIN-134S5007VI0:~/microservice_ansible$ ansible-playbook -i inventory.ini deploy.yml

PLAY [Установка микросервиса с Prometheus метриками] *****
*****
TASK [Gathering Facts] *****
*****fatal: [192.168.0.107]: FAILED! => {"msg": "Using a SSH password instead of
a key is not possible because Host Key checking is enabled and sshpass does not support this. Ple
ase add this host's fingerprint to your known_hosts file to manage this host."}

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

Рисунок 12 – Результат выполнения

Для того, чтобы такой проблемы не возникало можно использовать два способа.

Первый заключается в том, чтобы использовать флаг *ANSIBLE\_HOST\_KEY\_CHECKING=False*. Этот флаг временно отключит проверку ключей.

```
gleb@WIN-134S5007VI0:~/microservice_ansible$ ANSIBLE_HOST_KEY_CHECKING=False ansible-playbook -i i
nventory.ini deploy.yml

PLAY [Установка микросервиса с Prometheus метриками] *****
*****
TASK [Gathering Facts] *****
*****ok: [192.168.0.107]

TASK [Установить зависимости] *****
*****ok: [192.168.0.107]

TASK [Установить библиотеки Python] *****
*****ok: [192.168.0.107]

TASK [Копировать микросервис] *****
*****changed: [192.168.0.107]

TASK [Создать systemd unit] *****
*****ok: [192.168.0.107]

TASK [Перезапустить systemd] *****
*****ok: [192.168.0.107]

TASK [Включить и запустить сервис] *****
*****changed: [192.168.0.107]

PLAY RECAP *****
*****192.168.0.107 : ok=7   changed=2   unreachable=0   failed=0
skipped=0   rescued=0   ignored=0
```

Рисунок 13 – Результат выполнения

Как видно из рисунка 13 playbook сработал корректно и развернул микросервис на ВМ.

Второй способ заключается в том, чтобы вручную подключиться к ВМ с использованием пароля от root.

```
gleb@WIN-134S5007VI0:~/microservice_ansible$ ssh root@192.168.0.107
root@192.168.0.107's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Sat Apr 26 17:12:11 2025 from 192.168.0.110

[root@localhost ~]# exit
logout
Connection to 192.168.0.107 closed.
gleb@WIN-134S5007VI0:~/microservice_ansible$ ansible-playbook -i inventory.ini deploy.yml

PLAY [Установка микросервиса с Prometheus метриками] *****
TASK [Gathering Facts] *****
*****ok: [192.168.0.107]

TASK [Установить зависимости] *****
*****ok: [192.168.0.107]

TASK [Установить библиотеки Python] *****
*****ok: [192.168.0.107]

TASK [Копировать микросервис] *****
*****ok: [192.168.0.107]

TASK [Создать systemd unit] *****
*****ok: [192.168.0.107]

TASK [Перезапустить systemd] *****
*****ok: [192.168.0.107]

TASK [Включить и запустить сервис] *****
*****ok: [192.168.0.107]

PLAY RECAP *****
*****192.168.0.107 : ok=7 changed=0
unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
```

Рисунок 14 – Результат выполнения

Из рисунка 14-ть видно, что playbook отработал без ошибок.

## Проверка на ВМ

После успешного запуска playbook'а можно проверить работу микросервиса на виртуальной машине.

Для этого откроем браузер и перейдем на <http://localhost:8080/metrics>

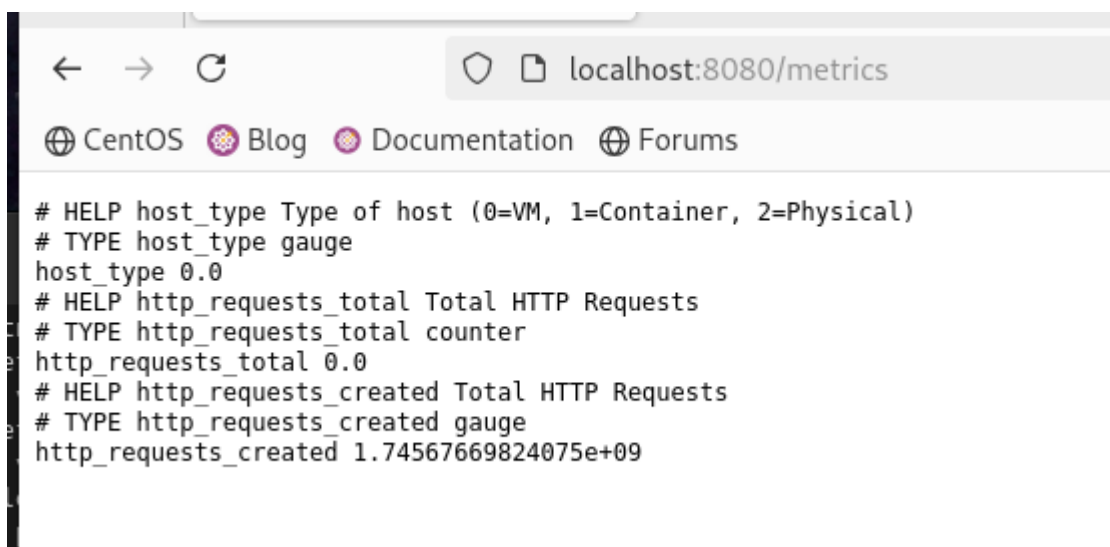
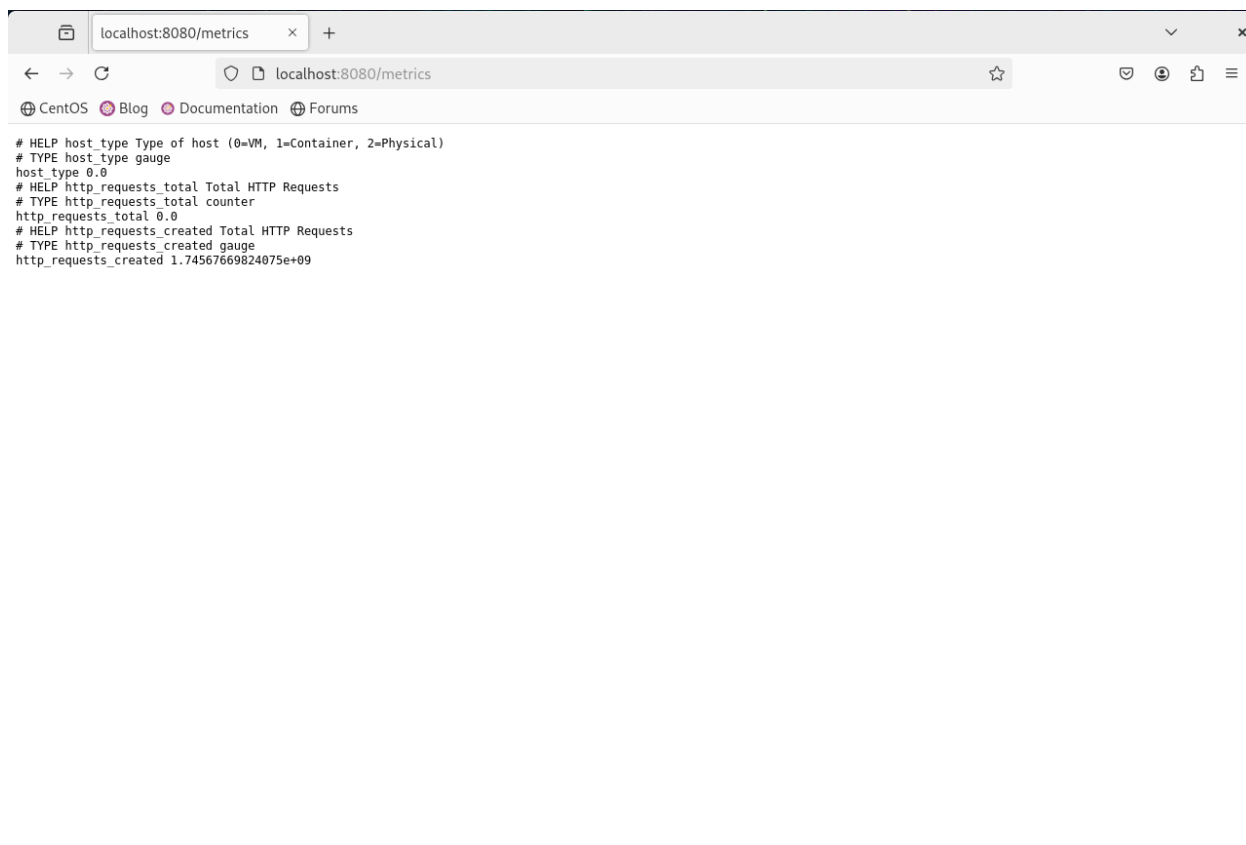


Рисунок 15 – Метрики

Как видно на рисунке 15 представленные не все метрики, а лишь их часть.

***host\_type*** показывает тип хоста, на котором работает сервис

0.0 - виртуальная машина (VM)

1.0 - контейнер

2.0 - физический сервер

***http\_requests\_total*** счетчик общего количества HTTP-запросов к серверу

*http\_requests\_created* временная метка создания метрики (timestamp в Unix-формате)

```
gleb@WIN-134S5007VIO:~/microservice_ansible$ curl http://192.168.0.107:8080/metrics
# HELP host_type Type of host (0=VM, 1=Container, 2=Physical)
# TYPE host_type gauge
host_type 0.0
# HELP http_requests_total Total HTTP Requests
# TYPE http_requests_total counter
http_requests_total 0.0
# HELP http_requests_created Total HTTP Requests
# TYPE http_requests_created gauge
http_requests_created 1.74567669824075e+09
```

Рисунок 16 – Обращение к серверу с хоста

## **Заключение**

Мне удалось создать микросервис на Python, с автоматизацией процесса через Ansible, который собирает метрики и предоставляет их на порту 8080.

## **ПРИЛОЖЕНИЕ 1**

В приложение входят:

- 1) Листинг `deploy.yml` (страница 19),
- 2) Листинг `microservice.py` (страница 20,21).



## Листинг deploy.yml

```
- name: Установка микросервиса с Prometheus метриками
  hosts: vm
  become: yes

  tasks:
    - name: Установить зависимости
      dnf:
        name:
          - python3
          - python3-pip
        state: present

    - name: Установить библиотеки Python
      pip:
        name:
          - flask
          - prometheus_client
        executable: pip3

    - name: Копировать микросервис
      copy:
        src: files/microservice.py
        dest: /opt/microservice.py
        mode: '0755'

    - name: Создать systemd unit
      copy:
        dest: /etc/systemd/system/microservice.service
        content: |
          [Unit]
          Description=Microservice with Prometheus metrics
          After=network.target

          [Service]
          ExecStart=/usr/bin/python3 /opt/microservice.py
          Restart=always

          [Install]
          WantedBy=multi-user.target

    - name: Перезапустить systemd
      systemd:
        daemon_reload: yes

    - name: Включить и запустить сервис
      systemd:
        name: microservice
        enabled: yes
        state: started
```

## Листинг microservice.py

```
from http.server import BaseHTTPRequestHandler, HTTPServer
import socket
import time
from prometheus_client import start_http_server, Gauge, Counter
from prometheus_client.core import REGISTRY

# Метрики Prometheus
METRIC_HOST_TYPE = Gauge('host_type', 'Type of host (0=VM, 1=Container, 2=Physical)')
METRIC_REQUEST_COUNT = Counter('http_requests_total', 'Total HTTP Requests')

class RequestHandler(BaseHTTPRequestHandler):
    def do_GET(self):
        METRIC_REQUEST_COUNT.inc()
        self.send_response(200)
        self.end_headers()
        self.wfile.write(b"OK")

def detect_host_type():
    # 0=VM, 1=Container, 2=Physical
    try:
        with open("/proc/1/cgroup", "r") as f:
            if "docker" in f.read() or "kubepods" in f.read():
                return 1 # Контейнер
    except:
        pass

    if "hypervisor" in open("/proc/cpuinfo").read().lower():
        return 0 # Виртуальная машина
    else:
        return 2 # Физический сервер

def remove_default_metrics():
    """Безопасное удаление стандартных метрик"""
    default_metrics = [
        'python_gc_objects_collected_total',
        'python_gc_objects_uncollectable_total',
        'python_gc_collections_total',
        'python_info',
        'process_virtual_memory_bytes',
        'process_resident_memory_bytes',
        'process_start_time_seconds',
        'process_cpu_seconds_total',
        'process_open_fds',
        'process_max_fds'
    ]

    for name in default_metrics:
        try:
            REGISTRY.unregister(REGISTRY._names_to_collectors[name])
        except KeyError:
            # Если метрика не найдена, пропускаем
            pass

if __name__ == "__main__":
    # Определяем тип хоста и задаём метрику
    METRIC_HOST_TYPE.set(detect_host_type())

    # Удаляем стандартные метрики
    remove_default_metrics()

    # Запускаем Prometheus-метрики на порту 8080
```

```
start_http_server(8080)

# Запускаем HTTP-сервер (опционально)
server = HTTPServer(('0.0.0.0', 8000), RequestHandler)
print("Server running on http://0.0.0.0:8000")
server.serve_forever()
```