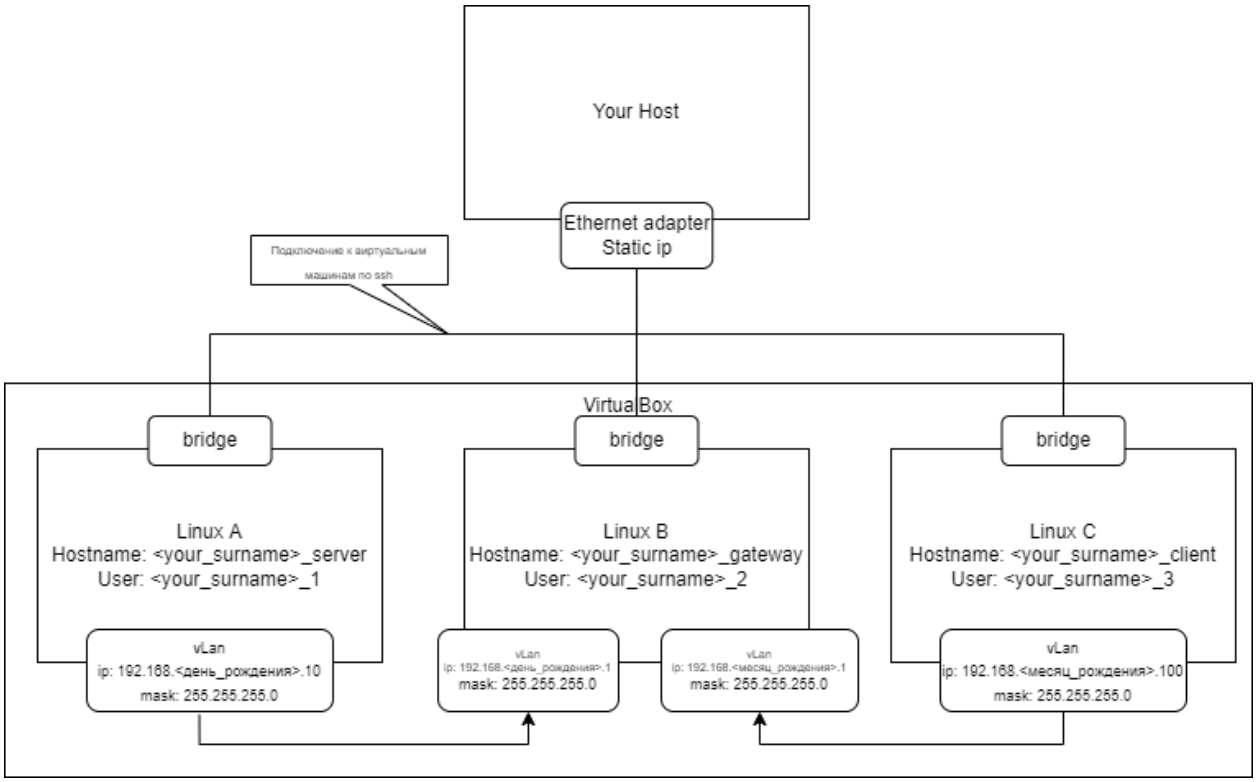
Лабораторная работа 1. Практика по Linux

Выполнила студент гр. 5142704/30801 Гайна А. А.

Задание

Для выполнения данного задания необходимо:

1. Развернуть три виртуальные машины Linux, согласно схеме ниже



2. Linux A
    i. Сконфигурировать Hostname следующим образом: <your_surname>_server (пропустить если делаете через Play with docker)
    ii. Создать пользователя <your_surname>_1 (пропустить если делаете через Play with docker)
    iii. Сконфигурировать виртуальный интерфейс со следующим ip адресом: 192.168.<день рождения>.10/24
    iv. Развернуть Http сервер на виртуальной машине на порту 5000. Необходимо реализовать минимум три эндпоинта (запрос /get, /post, /put)
3. Linux B
    i. Сконфигурировать Hostname следующим образом: <your_surname>_gateway (пропустить если делаете через Play with docker)
    ii. Создать пользователя <your_surname>_2 (пропустить если делаете через Play with docker)

iii. Сконфигурировать 2 виртуальных интерфейс со следующими ip адресом: 192.168.<день рождения>.1/24, 192.168.<месяц рождения>.10/24

iv. С помощью утилит ip route и iptables настроить маршрут пакетов от Linux A до C. Должны быть запрещены все пакеты, кроме http пакетов через порт 5000 (маршруты обязательно, файрвол опционально)

v. Запустить программу tcpdump с фильтрацией по портам 5000

4. Linux C
   i. Сконфигурировать Hostname следующим образом: <your_surname>_client (пропустить если делаете через Play with docker)
   ii. Создать пользователя <your_surname>_3 (пропустить если делаете через Play with docker)
   iii. Сконфигурировать виртуальный интерфейс со следующим ip адресом: 192.168.<месяц рождения>.100/24
   iv. С помощью команды curl на машине C послать 3 запроса на машину A в http сервер (/get, /post, /put)
5. При перезагрузки системы все сервисы и сетевая архитектура должны также функционировать (сохранить свои настройки)
6. Сделать скриншоты всех этапов задания
7. Оформить отчет в виде Markdown файла. Приложить конфигурационные файлы в репозиторий
8. Для play with docker также нужно написать bash скрипты для воспроизведения на новых вм

Решение

Шаг 1: создание виртуальных машин в PWD

С помощью возможностей `https://labs.play-with-docker.com/` создаем три виртуальные машины при помощи нажатия `ADD NEW INSTANCE`

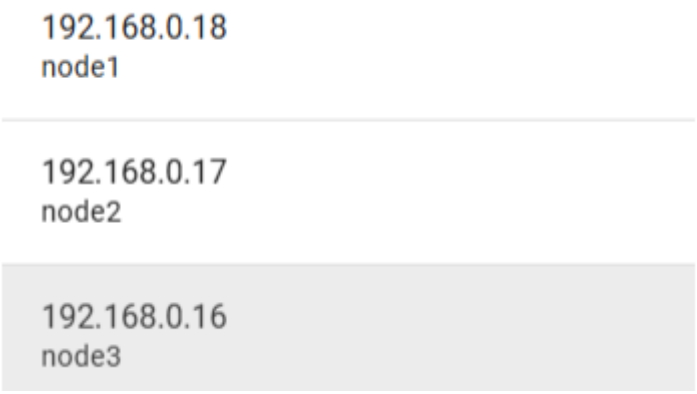


Рисунок 1 - Три виртуальные машины в Play With Docker

Шаг 2: генерация SSH-ключа

Для подключения к виртуальным машинам по ssh необходимо сгененрировать ключ

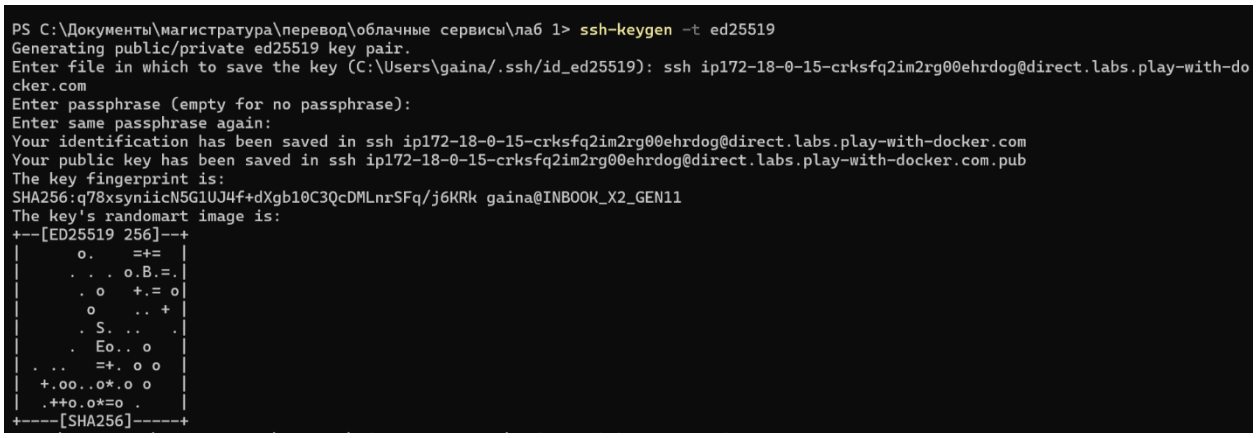


Рисунок 2 - Генерация ключа ssh протоколом ed25519 для 1-го сервера

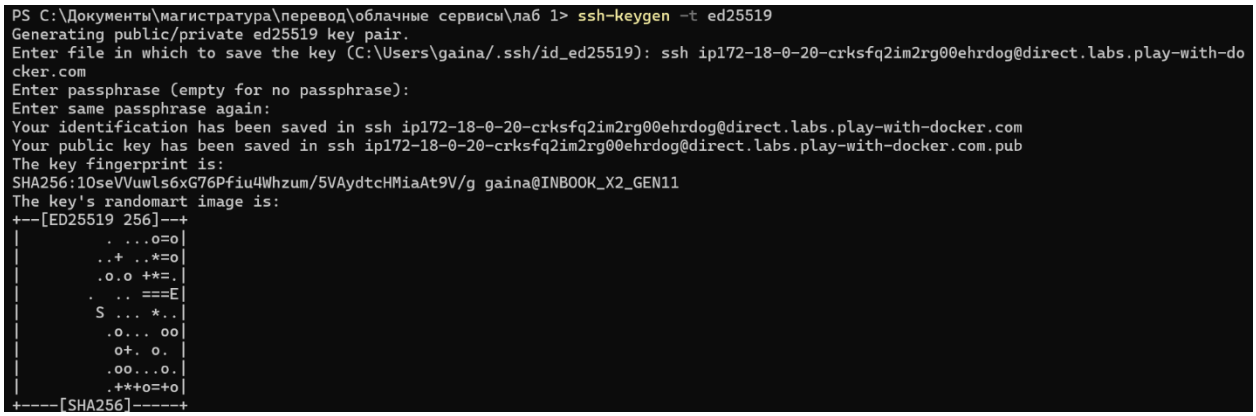


Рисунок 3 - Генерация ключа ssh протоколом ed25519 для 2-го сервера

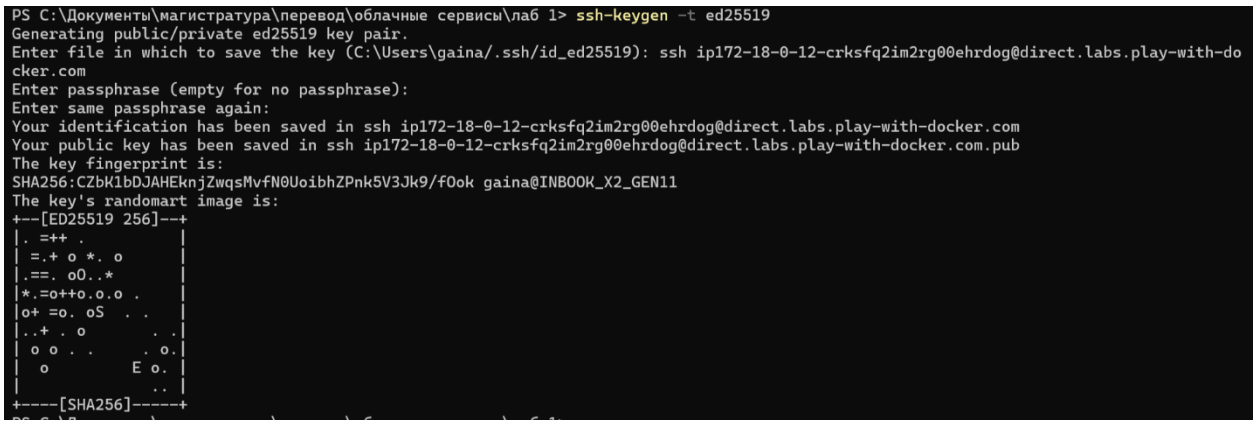


Рисунок 4 - Генерация ключа ssh протоколом ed25519 для 3-го сервера

Подключение к виртуальным машинам происходит по командам, указанным для каждой ВМ в PWD:

1. ssh ip172-18-0-15-crksfq2im2rg00ehrdog@direct.labs.play-with-docker.com

2. ssh ip172-18-0-20-crksfq2im2rg00ehrdog@direct.labs.play-with-docker.com
3. ssh ip172-18-0-12-crksfq2im2rg00ehrdog@direct.labs.play-with-docker.com

Шаг 3: Настройка маршрутов

По заданию каждому адаптеру необходимо назначить следующие ip-адреса:

Linux A : 192.168.18.10 / 24

Linux C : 192.168.7.100 / 24

Linux B_1 : 192.168.18.1 / 24

Linux B_2 : 192.168.7.1 / 24

```
################################################################
#                          WARNING!!!!                         #
# This is a sandbox environment. Using personal credentials    #
# is HIGHLY! discouraged. Any consequences of doing so are     #
# completely the user's responsibilites.                       #
#                                                              #
# The PWD team.                                                #
################################################################
[node1] (local) root@192.168.0.18 ~
$ ip link add macvlan1 link eth0 type macvlan mode bridge
[node1] (local) root@192.168.0.18 ~
$ ip address add dev macvlan1 192.168.18.10/24
[node1] (local) root@192.168.0.18 ~
$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    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
2: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN
    link/ether 02:42:1e:bc:04:f3 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
3: macvlan1@eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN qlen 1
    link/ether 32:af:64:72:65:6a brd ff:ff:ff:ff:ff:ff
    inet 192.168.18.10/24 scope global macvlan1
       valid_lft forever preferred_lft forever
7884: eth0@if7885: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
    link/ether b2:f1:03:78:7f:18 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.18/23 scope global eth0
       valid_lft forever preferred_lft forever
7888: eth1@if7889: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
    link/ether 02:42:ac:12:00:0f brd ff:ff:ff:ff:ff:ff
    inet 172.18.0.15/16 scope global eth1
       valid_lft forever preferred_lft forever
[node1] (local) root@192.168.0.18 ~
$ ip link set macvlan1 up
```

Рисунок 5 - Создание адаптера и установка IP - адреса для Linux A

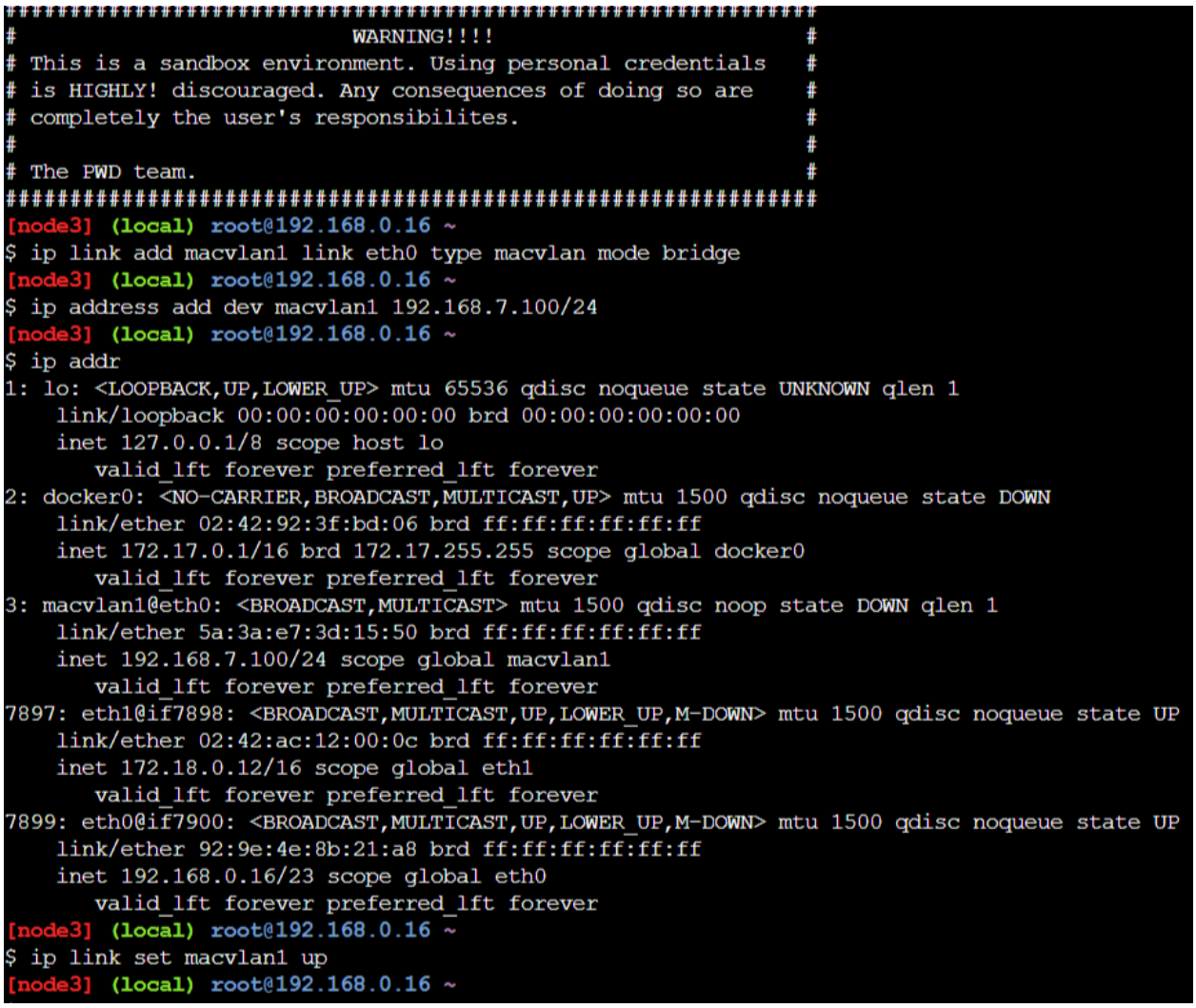


Рисунок 6 - Создание адаптера и установка IP - адреса для Linux C

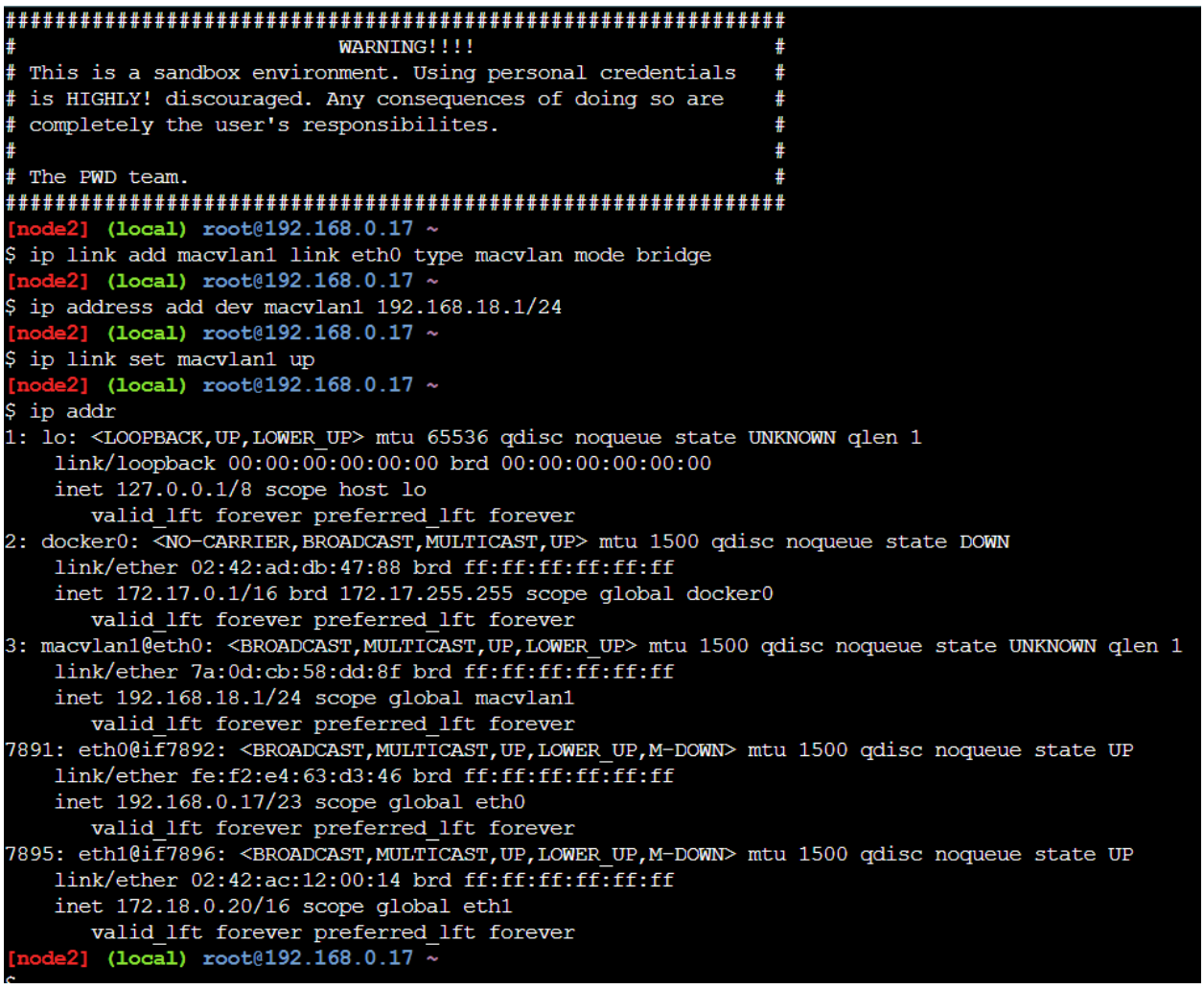


Рисунок 7 - Создание 1-го адаптера и установка IP - адреса для Linux B

```
[node2] (local) root@192.168.0.17 ~
$ ip link add macvlan2 link eth0 type macvlan mode bridge
ip: RTNETLINK answers: File exists
[node2] (local) root@192.168.0.17 ~
$ ip address add dev macvlan1 192.168.7.1/24
[node2] (local) root@192.168.0.17 ~
$ ip link set macvlan2 up
[node2] (local) root@192.168.0.17 ~
$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    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
2: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN
    link/ether 02:42:ad:db:47:88 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
3: macvlan1@eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN qlen
 1
    link/ether 7a:0d:cb:58:dd:8f brd ff:ff:ff:ff:ff:ff
    inet 192.168.18.1/24 scope global macvlan1
       valid_lft forever preferred_lft forever
    inet 192.168.7.1/24 scope global macvlan1
       valid_lft forever preferred_lft forever
4: macvlan2@eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN qlen
 1
    link/ether 1e:a7:ba:f3:15:48 brd ff:ff:ff:ff:ff:ff
7891: eth0@if7892: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
    link/ether fe:f2:e4:63:d3:46 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.17/23 scope global eth0
       valid_lft forever preferred_lft forever
7895: eth1@if7896: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
    link/ether 02:42:ac:12:00:14 brd ff:ff:ff:ff:ff:ff
    inet 172.18.0.20/16 scope global eth1
       valid_lft forever preferred_lft forever
[node2] (local) root@192.168.0.17 ~
```

Рисунок 8 - Создание 2-го адаптера и установка IP - адреса для Linux B

После настройки сети на ВМ необходимо прописать маршруты у клиентов A и C к их подсетям через машину B при помощи команды: `ip route add <subnet A vm>/<mask> via <gateway ip B vm>`

```
[node1] (local) root@192.168.0.18 ~
$ ip route add 192.168.7.0/24 via 192.168.18.1
```

Рисунок 9 - Настройка маршрута передачи от A к C через B

```
[node3] (local) root@192.168.0.16 ~
$ ip route add 192.168.18.0/24 via 192.168.7.1
```

Рисунок 10 - Настройка маршрута передачи от C к A через B

Шаг 3: Настройка файрвола

На Linux B необходимо настроить файрвол таким образом, чтобы Linux B пропускал только http и только через порт 5000.

Это можно сделать при помощи tcpdump.

```
[node2] (local) root@192.168.0.17 ~
$ apk add tcpdump
fetch https://dl-cdn.alpinelinux.org/alpine/v3.18/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.18/community/x86_64/APKINDEX.tar.gz
(1/2) Installing libpcap (1.10.4-r1)
(2/2) Installing tcpdump (4.99.4-r1)
Executing busybox-1.36.1-r2.trigger
OK: 469 MiB in 164 packages
[node2] (local) root@192.168.0.17 ~
$ tcpdump -i any -s 0 'tcp port http' -w /tmp/http.cap and 'port 5000'
tcpdump: data link type LINUX_SLL2
tcpdump: listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 bytes
```

Рисунок 11 - Настройка файрвола на Linux B

Шаг 4: Организация клиент-серверного взаимодействия между A и C

После настройки маршрутов необходимо развернуть сервер на Linux A. Для этого нужно использовать библиотеку Flask.

```
[node1] (local) root@192.168.0.18 ~
$ pip install flask
Collecting flask
  Downloading flask-3.0.3-py3-none-any.whl (101 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 101.7/101.7 kB 2.8 MB/s eta 0:00:00
Collecting Werkzeug>=3.0.0 (from flask)
  Downloading werkzeug-3.0.4-py3-none-any.whl (227 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 227.6/227.6 kB 15.8 MB/s eta 0:00:00
Collecting Jinja2>=3.1.2 (from flask)
  Downloading jinja2-3.1.4-py3-none-any.whl (133 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 133.3/133.3 kB 12.5 MB/s eta 0:00:00
Collecting itsdangerous>=2.1.2 (from flask)
  Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
Collecting click>=8.1.3 (from flask)
  Downloading click-8.1.7-py3-none-any.whl (97 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 97.9/97.9 kB 15.7 MB/s eta 0:00:00
Collecting blinker>=1.6.2 (from flask)
  Downloading blinker-1.8.2-py3-none-any.whl (9.5 kB)
Collecting MarkupSafe>=2.0 (from Jinja2>=3.1.2->flask)
  Downloading MarkupSafe-2.1.5-cp311-cp311-musllinux_1_1_x86_64.whl (33 kB)
Installing collected packages: MarkupSafe, itsdangerous, click, blinker, Werkzeug, Jinja2, fl
ask
Successfully installed Jinja2-3.1.4 MarkupSafe-2.1.5 Werkzeug-3.0.4 blinker-1.8.2 click-8.1.7
 flask-3.0.3 itsdangerous-2.2.0
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting beha
viour with the system package manager. It is recommended to use a virtual environment instead
: https://pip.pypa.io/warnings/venv
[node1] (local) root@192.168.0.18 ~
$
```

Рисунок 12 - Установка Flask

Далее создаем файл app.py со следующим содержимым:

```python
from flask import Flask, request

app = Flask(__name__)

data = {"username":"", "password":""}
```

```
@app.route("/")
def get_():
    return data

@app.route("/",methods = ['POST'])
def post():
    data_json=request.get_json()
    if data_json is None:
        return 'Invalid JSON data', 400

    data['username']=data_json['username']
    data['password']=data_json['password']
    print(f"Data received {data_json}")

    return [data['username'], data['password']]

@app.route("/", methods =['PUT'])
def put():
    str = request.args.get('password')
    print(f"New password received {str}")
    data['password'] = str
    return [data['username'], data['password']]

app.run(host='0.0.0.0', port=5000)

EOF
```

Запущенный сервер выглядит так:

```
[node1] (local) root@192.168.0.18 ~
$ echo -e "Run the server\n"
Run the server

[node1] (local) root@192.168.0.18 ~
$ python app.py
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a produc
tion WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.18.0.15:5000
Press CTRL+C to quit
```

Рисунок 13 - Сервер на машине А

С помощью команды curl на машине С было послано 3 запроса на машину А в http сервер (/get, /post, /put):

Рисунок 14 - Отправка GET-запроса

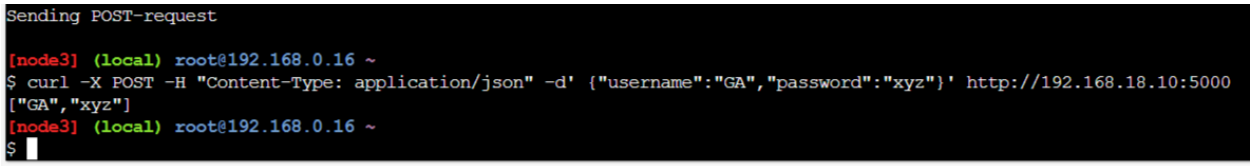


Рисунок 15 - Отправка POST-запроса

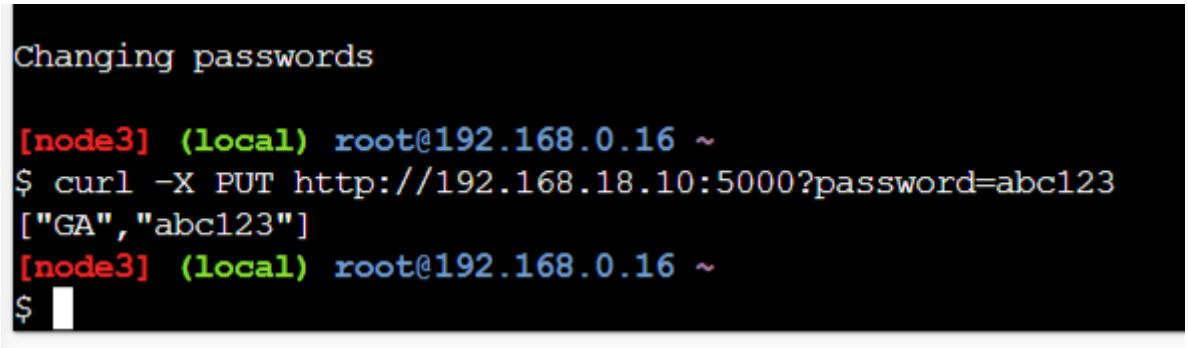


Рисунок 16 - Отправка PUT-запроса

Были получены ответы на отправленные запросы с машины C на A:

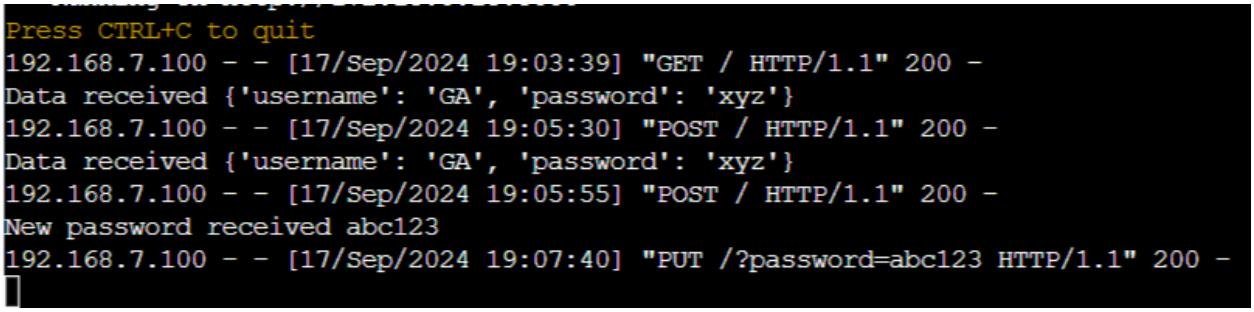


Рисунок 17 - Ответ на отправленные запросы с машины C на A