

# Лабораторная работа №8. Первое знакомство с Containerlab. Захват сетевого трафика.

## 1. Тема

Основы работы с эмулятором сетей Containerlab. Создание простой топологии и анализ трафика с помощью Edgeshark/Wireshark.

## 2. Цель работы

- **Теоретическая:** Изучить концепции контейнеризации сетевых устройств, принципы работы инструмента Containerlab и основы анализа сетевого трафика.
- **Практическая:** Приобрести навыки описания сетевой топологии в формате YAML, развертывания лабораторного стенда с помощью containerlab, базового управления узлами на базе FRR и захвата трафика для анализа.

## 3. Задачи

1. Создать файл топологии сети в формате YAML для containerlab.
2. Развернуть лабораторный стенд из двух маршрутизаторов и одного хоста.
3. Настроить базовую IP-адресацию на узлах.
4. Запустить захват сетевого трафика между узлами с помощью edgeshark.
5. Проверить связность между узлами и проанализировать захваченный трафик в Wireshark.

## 4. Оборудование и программное обеспечение

- **Программное обеспечение:** Terminal (CLI), текстовый редактор (nano/vim/vscode), Containerlab (v0.51.0+), Docker & Docker-Compose, Edgeshark, Wireshark.
- **Оборудование (виртуальное, контейнерное):**
  - Маршрутизаторы FRR (с образом `frrouting/frr:latest`) - 2 шт.
  - Хост-контейнер (с образом `alpine:latest`) - 1 шт.
  - Виртуальные сетевые связи (links) - 2 шт.

## 5. Краткие теоретические сведения

**Containerlab** — это инструмент с открытым исходным кодом для развертывания и управления лабораторными стендами сетевой инфраструктуры на основе Docker-

контейнеров. В отличие от графических симуляторов (Packet Tracer), топология описывается декларативно в YAML-файле, что ближе к современным практикам Infrastructure as Code (IaC). Containerlab создает изолированную сетевую среду, соединяя контейнеры через виртуальные интерфейсы (veth).

**FRR (Free Range Routing)** — это пакет маршрутизации с открытым исходным кодом, реализующий такие протоколы, как OSPF, BGP, RIP. В контексте containerlab FRR работает внутри контейнера, предоставляя полнофункциональный CLI (vtysh), похожий на интерфейсы Cisco/Juniper.

**Принцип работы:** Пользователь описывает в `topology.yaml` узлы (nodes) и связи (links) между ними. Containerlab, используя Docker, разворачивает контейнеры, создает для них сетевые пространства и виртуальные соединения. Узлы становятся доступны для конфигурации по SSH или через CLI контейнера.

## 6. Порядок выполнения работы

### 6.1. Подготовительный этап

1. Откройте терминал в Manjaro Linux.
2. Создайте директорию для лабораторной работы: `mkdir ~/lab1_ospf_intro && cd ~/lab1_ospf_intro`
3. Убедитесь, что Docker и containerlab работают: выполните команды `docker ps` и `clab version`. Ошибок быть не должно.

### 6.2. Основной этап

#### 1. Создание файла топологии

- Создайте файл с именем `topology.yaml` в текущей директории
- Скопируйте в него следующее содержимое:

```
name: lab1-first-steps

topology:
  nodes:
    router1:
      kind: linux
      image: frrouting/frr:latest
      binds:
        - router1:/etc/frr:/etc/frr # Монтируем директорию для
сохранения конфигурации FRR
    router2:
      kind: linux
      image: frrouting/frr:latest
```

```
binds:
  - router2:/etc/frr:/etc/frr
client:
  kind: linux
  image: alpine:latest

links:
  - endpoints: ["router1:eth1", "router2:eth1"]
  - endpoints: ["router2:eth2", "client:eth1"]
```

## 2. Развертывание лабораторного стенда

- В терминале, из директории с файлом `topology.yaml`, выполните команду развертывания: `clab deploy -t topology.yaml`
- После успешного выполнения вы увидите таблицу с узлами и их IP-адресами для управления.
- Убедитесь, что все контейнеры запущены: `docker ps`

## 3. Базовая настройка маршрутизаторов

- Подключитесь к консоли первого маршрутизатора: `docker exec -it clab-lab10first-steps-router1 bash`
- Внутри контейнера запустите оболочку FRR для настройки: `vttysh`
- Перейдите в режим конфигурации: `configure terminal`
- Настройте интерфейс eth1:

```
interface eth1
ip address 10.0.12.1/24
exit
```

- Выйдите из режима конфигурации: `exit`
- Сохраните конфигурацию FRR: `write memory`
- Просмотрите IP-адреса интерфейсов: `show interface brief`
- Выйдите из vtysh ( `exit` ) и из контейнера ( `exit` ).
- **Аналогично настройте router2:** IP-адрес `10.0.12.2/24` на eth1 и `192.168.20.1/24` на eth2.

## 4. Запуск захвата трафика

- Запустите Edgeshark `edgeshark-up`.
- Перейдите по адресу `localhost:5001`
- Найдите контейнер с роутером на котором хотите захватить трафик. Нажмите на значок плавника рядом с требуемым интерфейсом.
- Wireshark начнет захват и отобразит пакеты в режиме реального времени. **Оставьте эту сессию запущенной.**

## 5. Проверка связности и анализ трафика

- Вернитесь к первому терминалу. Подключитесь к контейнеру client: `docker exec -it clab-lab1-first-steps-client sh`
- Настройте ip адрес (например с помощью утилиты ip)
- Попробуйте пропинговать router2: `ping 192.168.20.1`
- Попробуйте пропинговать router1 (должно **НЕ** сработать, так как на router1 нет маршрута к сети 192.168.20.0/24): `ping 10.0.12.1`
- В окне терминала с wireshark вы увидите ICMP-пакеты (эхо-запросы и эхо-ответы) от проверки связи с `192.168.20.1` . Трафика до `10.0.12.1` видно не будет.
- Остановите ping ( `Ctrl+C` ). Выйдите из контейнера client ( `exit` ).

## 6. Настройка маршрутизации

- Измените топологию таким образом что бы к router1 так же был подключен client2 в подсети 192.168.100.0/24
- Настройте ip адреса на клиенте и на роутере
- На router1 и router2 настройте статические маршруты
- Добейтесь того что бы клиенты могли пинговать друг друга.

### 6.3. Контрольный этап

1. Остановите захват трафика в edgeshark.
  - Проанализируйте захваченные данные
2. Остановите работу Edgeshark `edgeshark-down`
3. Уничтожьте лабораторный стенд, освободив ресурсы: `clab destroy -t topology.yaml --cleanup` . Файлы конфигурации в папках `router1/` и `router2/` сохраняются.

## 7. Контрольные вопросы

1. В чем основное концептуальное отличие между Cisco Packet Tracer и Containerlab?
2. Какую роль играет Docker в работе Containerlab?
3. Для чего нужен файл `topology.yaml` ? Опишите структуру его основных секций.
4. Каким образом внутри контейнера с образом `frrouting/frr` происходит настройка маршрутизатора?
5. Какую команду containerlab нужно использовать для просмотра состояния всех узлов после развертывания?
6. Какие преимущества дает использование edgeshark по сравнению с запуском tcpdump вручную внутри контейнера?
7. Опишите последовательность действий для развертывания уже готовой лабораторной работы, присланной преподавателем (файл `topology.yaml` ).

## 8. Содержание отчета

- Тема, цель и задачи работы.
- Схема сети, полученная из описания в `topology.yaml` , с указанием назначенных IP-адресов.
- Результаты выполнения заданий:
  - Листинг файла `topology.yaml` .
  - Скриншот таблицы с узлами после выполнения `clab deploy` .
  - Выводы команд `show ip interface brief` с `router1` и `router2`.
  - Результаты проверки связности (ping) с узла `client`.
  - Скриншот окна Wireshark с выделенным ICMP-пакетом из файла `router2-eth1.pcap` .
- Ответы на контрольные вопросы.
- Выводы по работе (что нового узнали, с какими трудностями столкнулись, как `containerlab` соотносится с ранее изученными инструментами).