Обзор ToxiProxy, Conteinerlab и Kathara

ToxiProxy

Toxiproxy — это TCP-прокси с открытым исходным кодом, разработанный инженерной командой Shopify. Он помогает имитировать хаотические сетевые и системные условия из реальной жизни.

Как поставить?
https://github.com/Shopify/toxiproxy/releases
С помощью http api добавляются прокси и их токсики, например:

latency
Add a delay to all data going through the proxy. The delay is equal to latency +/- jitter.
Attributes:
latency : time in milliseconds
jitter : time in milliseconds
down
Bringing a service down is not technically a toxic in the implementation of Toxiproxy. This is done by POST ing to /proxies/{proxy} and setting the enabled field to false.
bandwidth
Limit a connection to a maximum number of kilobytes per second.
Attributes:
rate : rate in KB/s
slow_close
Delay the TCP socket from closing until delay has elapsed.
Attributes:
delay : time in milliseconds
timeout
Stops all data from getting through, and closes the connection after timeout . If timeout is 0, the connection won't close, and data will be dropped until the toxic is removed.
Attributes:
timeout : time in milliseconds
reset_peer
Simulate TCP RESET (Connection reset by peer) on the connections by closing the stub Input
immediately or after a timeout.
Attributes:
timeout : time in milliseconds
slicer
Slices TCP data up into small bits, optionally adding a delay between each sliced "packet".
Attributes:

average_size : size in bytes of an average packet
size_variation : variation in bytes of an average packet (should be smaller than average_size)
delay : time in microseconds to delay each packet by

limit_data
Closes connection when transmitted data exceeded limit.
bytes : number of bytes it should transmit before connection is closed

Что можно моделировать:
- задержка, ограничение соединения на максимальное число кб/с
- TCP RESET - разрывы соединений
- Slicer - делит пакет на биты и каждый бит отправляет с задержкой

Containerlab

Containerlab -- это инструмент, позволяющий управлять сетевыми лабораториями на основе контейнеров. Принцип работы следующий: он запускает контейнеры, создает виртуальное соединение между ними для создания топологий лабораторий и управляет жизненным циклом эмуляции.

Инструмент подходит для создания произвольных linux-контейнеров, в которых могут размещаться сетевые приложения и виртуальные функции, также Containerlab может выступать тестовым клиентом. При всем этом он предоставляет единый интерфейс IaaC (Infrastructure as a Code) для управления лабораториями, который может охватывать все необходимые варианты узлов.

Для того, чтобы развернуть Containerlab на Windows, требуется использовать WSL (Windows Subsystem Linux). WSL позволяет пользователям запускать легковесные Linux VM внутри Windows, это можно использовать для развертывания Containerlab.

Если WSL не установлен, исправить это можно одной командой:

```
wsl --install
```

Далее рассмотрим установку Containerlab. В Powersher от имени администратора включаем компоненты Windows, необходимые для работы подсистемы Linux: активируем базовую функциональность WSL и платформу виртуальных машин для использования более производительной версии WSL 2:

```
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux
/all /norestart

dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all
/norestart
```

После выполнения этих команд и перезагрузки компьютера, нужно с помощью команды `wsl --set-default-version 2` задать WSL 2 как версию по умолчанию.

Загрузим файл `.wsl` со страницы релизов. На данном этапе было выявлено, что файл не скачивается. Просто дважды щелкнем по файлу, и дистрибутив будет установлен. Альтернативно его можно установить с помощью `wsl -install -from-file C:\path\to\clab.wsl`. Containerlab должен появиться как программа в меню "Пуск". Запустим Containerlab:

```
wsl -d Containerlab
```

При первом запуске Containerlab WSL выполняется начальный этап настройки, который помогает настроить дистрибутив. Настраивается оболочка и командная строка. После этого SSH-ключи будут скопированы или сгенерированы (если ключей не существует). Это необходимо для обеспечения доступа к SSH без пароля, что улучшает интеграцию с DevPod.

Kathara

Kathara — это инструмент для эмуляции сетей на основе Docker-контейнеров. Этот инструмент может быть очень полезен для проведения интерактивных демонстраций/уроков, тестирования производственных сетей в песочной среде или разработки новых сетевых протоколов.

В среде Kathara каждое сетевое устройство реализовано в виде контейнера, а каждое соединение эмулируется с п2омощью виртуальной сети.

Каждое устройство можно настроить для работы с произвольным количеством (виртуальных) сетевых интерфейсов.

По умолчанию устройства используют образ Docker, который включает сетевое программное обеспечение, такое как демоны маршрутизации (RIP, OSPF и т.д.), HTTP-сервер, инструменты для работы с фаерволом (iptables(8)) и диагностические утилиты (ping(1), traceroute(1), tcpdump(1) и т.д.).
Путем настройки соответствующего программного обеспечения можно точно эмулировать конкретное сетевое устройство (например, маршрутизатор).

Kathara предоставляет два альтернативных интерфейса для запуска и настройки устройств. Набор команд с префиксом v- (vstart, vclean, vconfig), которые позволяют запускать и управлять отдельными устройствами, обеспечивая детальный контроль над их конфигурацией; и набор команд с префиксом l- (lstart, lclean, linfo, lrestart, lconfig), которые упрощают настройку предварительно сконфигурированных сетевых сценариев, состоящих из нескольких устройств.

Kathara также предоставляет набор глобальных команд:

- connect - подключиться к устройству kathara;
- info - показывает информацию о сетевой лаборатории kathara;
- wipe - удалить все устройства Kathara и домены коллизий, опционально также удалить настройки;
- settings - показать и редактировать настройки;
- check - проверить системное окружение.

Пример

Построим топологию, состоящую из двух хостов и маршрутизатора.

Соединения между хостами прописываются в файле lab.conf:

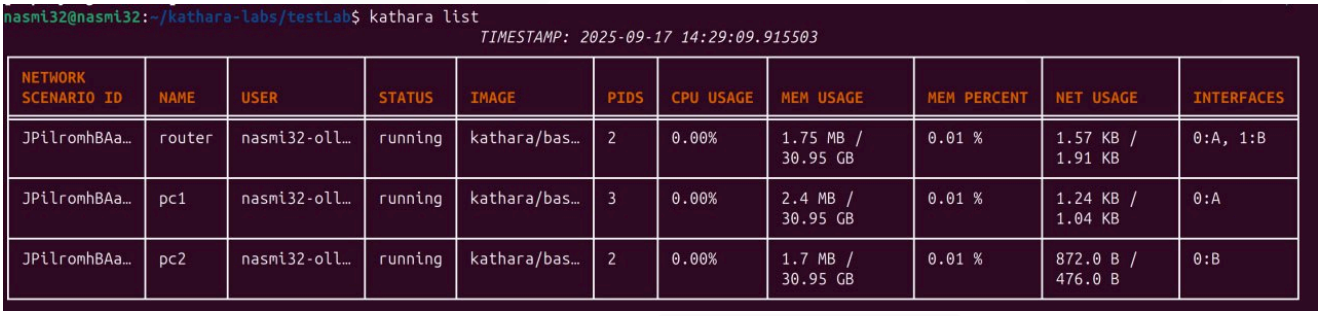
```
pc1[0]="A"

router[0]="A"
router[1]="B"

pc2[0]="B"
```

Опишем также оба хоста в файлах pc1.startup, pc2.startup:

```
ip link set eth0 up
ip addr add 10.0.0.1/24 dev eth0
ip route add default via 10.0.0.254
```

```
ip link set eth0 up
ip addr add 10.0.1.1/24 dev eth0
ip route add default via 10.0.1.254
```

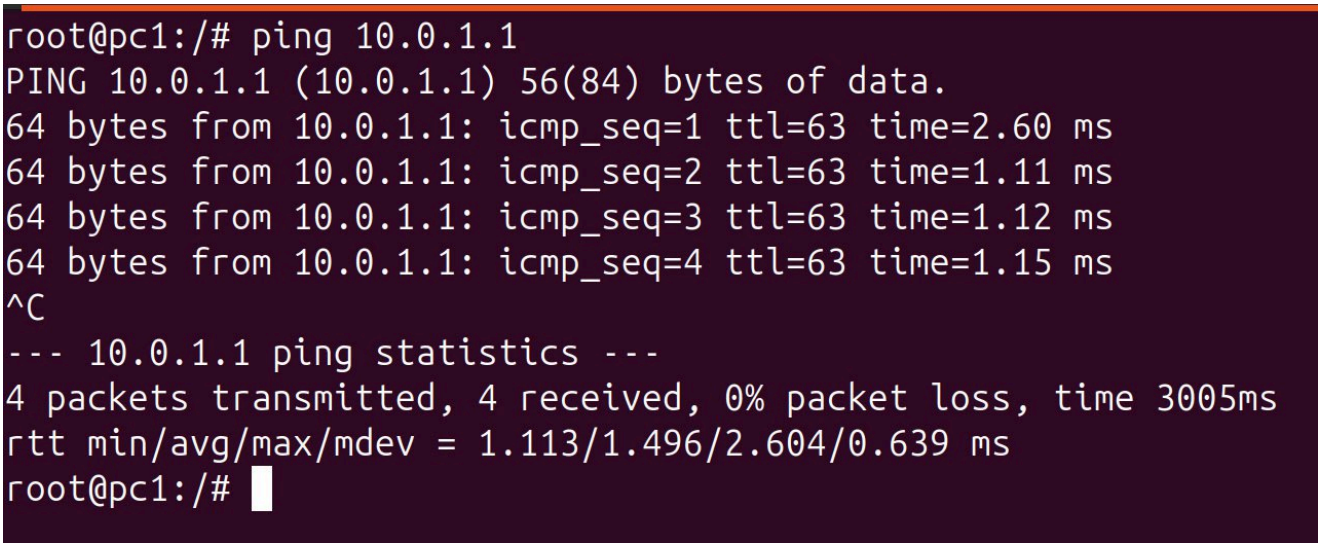
Запустим топологию `kathara lstart` и посмотрим список устройств `kathara list`:



```
nasmi32@nasmi32:~/kathara-labs/testLab$ kathara list
                                    TIMESTAMP: 2025-09-17 14:29:09.915503
```

NETWORK SCENARIO ID	NAME	USER	STATUS	IMAGE	PIDS	CPU USAGE	MEM USAGE	MEM PERCENT	NET USAGE	INTERFACES
JPilromhBAa…	router	nasmi32-oll…	running	kathara/bas…	2	0.00%	1.75 MB / 30.95 GB	0.01 %	1.57 KB / 1.91 KB	0:A, 1:B
JPilromhBAa…	pc1	nasmi32-oll…	running	kathara/bas…	3	0.00%	2.4 MB / 30.95 GB	0.01 %	1.24 KB / 1.04 KB	0:A
JPilromhBAa…	pc2	nasmi32-oll…	running	kathara/bas…	2	0.00%	1.7 MB / 30.95 GB	0.01 %	872.0 B / 476.0 B	0:B

Зайдем в устройство pc1 и пропингуем pc2 `kathara connect pc1`:



Пинг проходит успешно.

Сравнение

ToxyProxy и Kathara работают на разных уровнях абстракции, что определяет их архитектурные различия и области применения. ToxyProxy функционирует как TCP-прокси, действующий на транспортном уровне, что позволяет ему перехватывать и модифицировать трафик между приложением и его зависимостями. Однако это накладывает существенное ограничение — инструмент работает исключительно с TCP-протоколом и не поддерживает UDP, ICMP или другие сетевые протоколы. Это делает его непригодным для тестирования приложений, зависящих от UDP-трафика (например, VoIP, DNS, видеостриминг) или использующих ICMP для диагностики сети.

В отличие от этого, Kathara оперирует на сетевом уровне (L3) и ниже, обеспечивая полную эмуляцию сетевого стека. Это позволяет работать с любыми сетевыми протоколами, включая TCP, UDP, ICMP, а также специализированные протоколы маршрутизации like OSPF, BGP. Kathara создает изолированную среду, где можно тестировать поведение приложений при различных сетевых условиях, независимо от используемого транспортного протокола. Например, с помощью Kathara можно эмулировать потерю UDP-пакетов для VoIP-трафика или задержки ICMP-ответов для систем мониторинга.

Это фундаментальное различие в поддержке протоколов определяет и сферы применения этих инструментов. ToxyProxy наиболее эффективен для тестирования HTTP/REST API, баз данных и других TCP-ориентированных сервисов в микросервисной архитектуре. Kathara же незаменима для комплексного тестирования сетевой инфраструктуры, включая маршрутизацию, VPN-туннели, QoS-политики и многое другое, где важна работа с различными сетевыми протоколами.

При выборе между этими инструментами ключевым вопросом должен быть тип тестируемого трафика. Если речь идет исключительно о TCP-сервисах и необходимости быстрого внедрения в CI/CD — ToxyProxy будет оптимальным выбором. Если же требуется комплексное тестирование сетевого стека с поддержкой

различных протоколов — Kathara предоставляет гораздо более широкие возможности за счет большей сложности настройки и ресурсоемкости.