ntop in 2022:
Community Meeting and Future Plans

Luca Deri <deri@ntop.org>

Highlights

Title	Speaker
Introduction	Luca Deri
nDPId	Toni Uhlig
ntopng in 2022	Matteo Biscosi
What's new in PF_RING and nBox	Alfredo Cardigliano
New Developments: nTap, Kafka, ntopng Scripting	Luca Deri
nTap and Scripts at Work	Martin Scheu
Open Discussion	

ntop

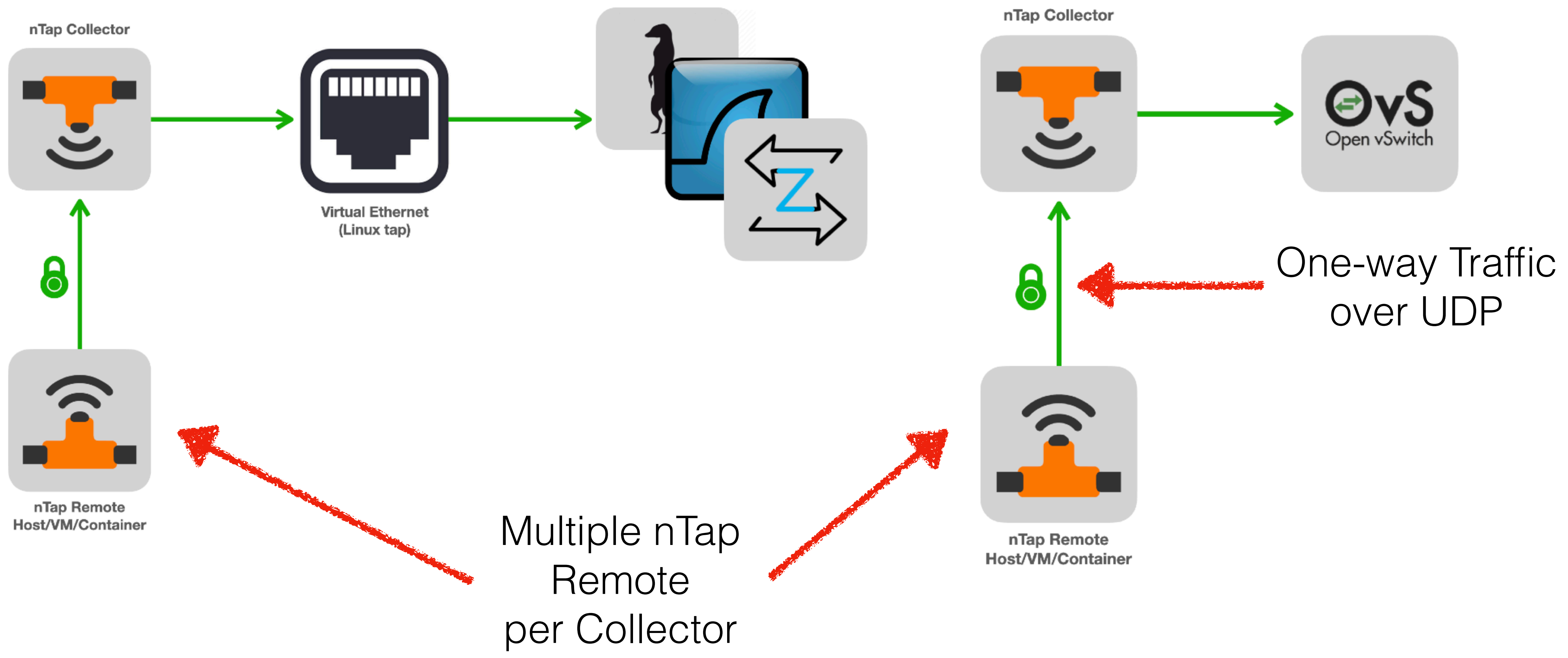
nTap, Kafka, Scripting

1. nTap

Welcome to nTap

- nTap is a *software* network tap that can be used to provide access to network packets from a remote location.
- Use Cases:
    - Need to collect selected traffic from a remote location and send it in a secure fashion to a monitoring center.
    - Troubleshooting: something is not working as expected and you need to temporarily send traffic for inspection.
    - Distributed topology that does not allow mirror/tap and so you need to grab traffic from remote devices.

nTap Architecture



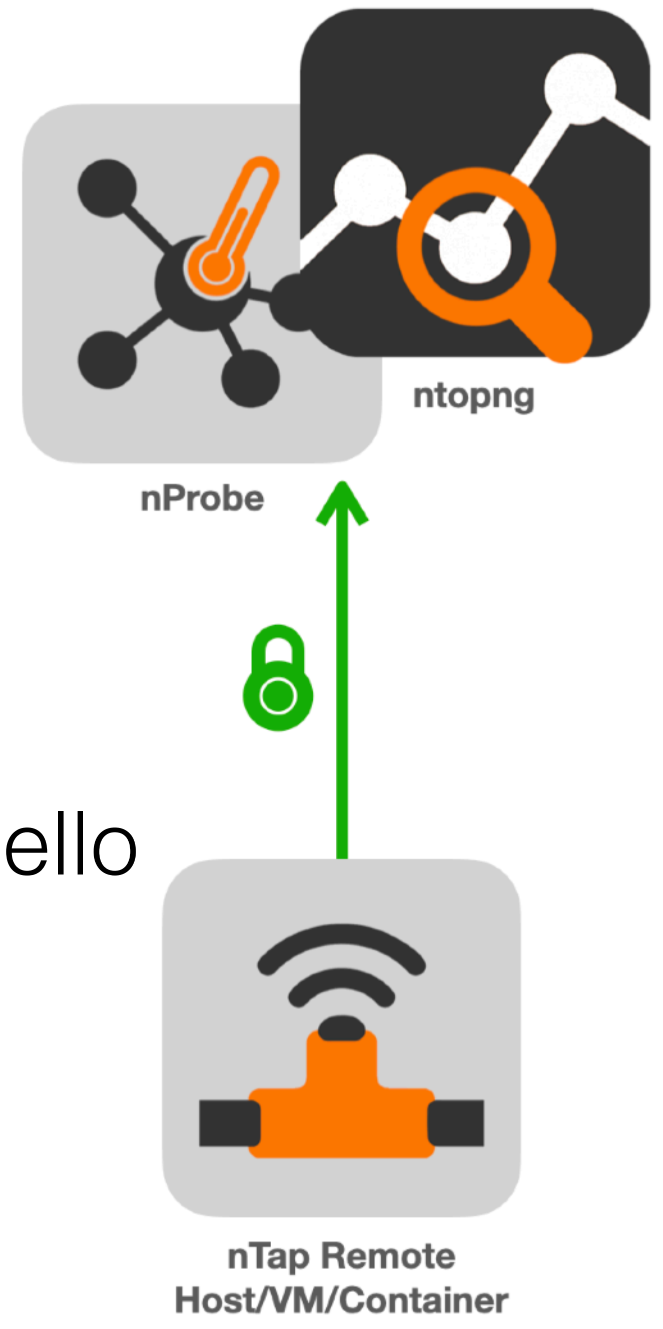
Using nTap with ntopng/nProbe

Native remote traffic collection built-in in nProbe and ntopng

with <u>no nTap license</u> required.

Example:

- [remote host]
  ntap_remote -i eth0 -c 1.2.3.4:5678 -k hello
- [local host]
  nprobe -3 5678 -n none --ntap hello



nTap Availability

Availability

- nTap Remote: MacOS, Linux, FreeBSD, Windows
- nTap Collector: Linux. ← No license required

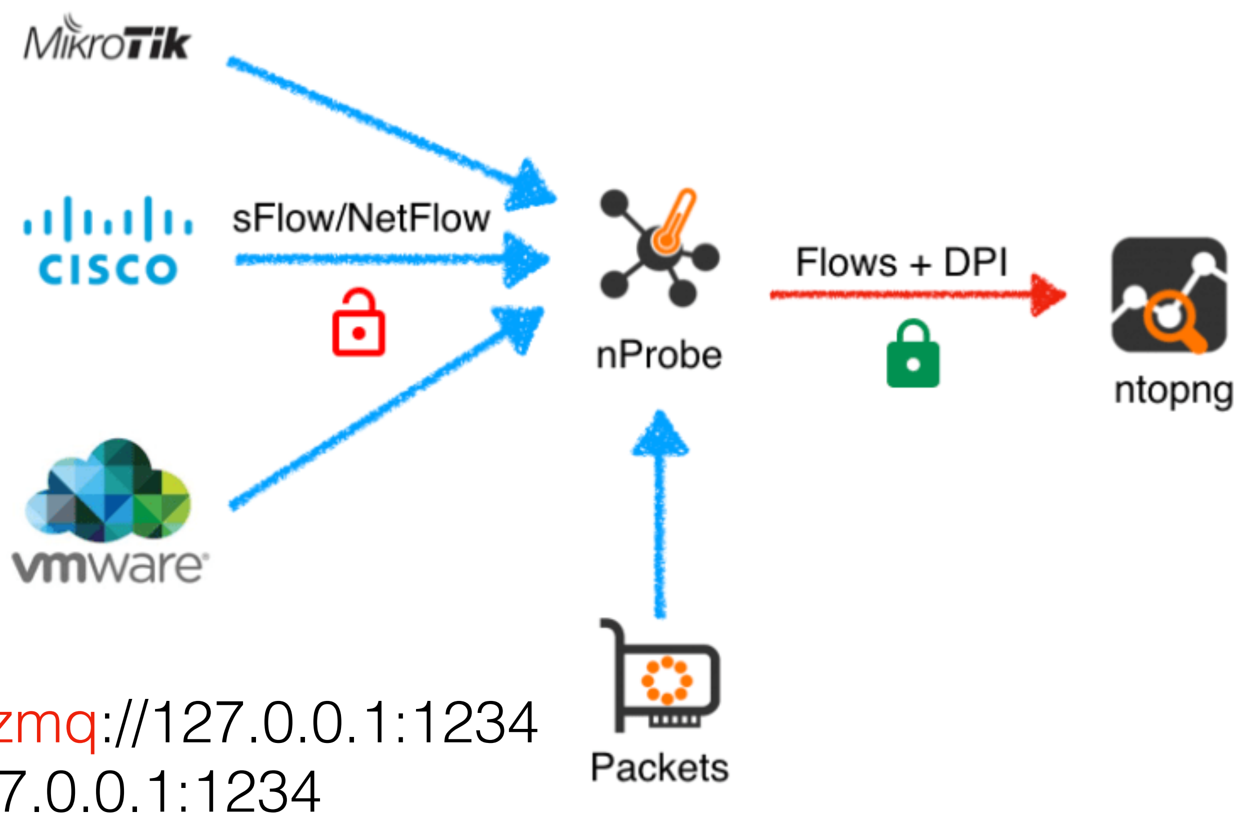
Performance (1 Gbit)

- With encryption: no loss up to 400 Mbps.
- Without encryption: no loss up to 450 Mbps.

2. Kafka

Kafka Everywhere

- Traditionally we have embedded ZMQ in our tools and optimised transmission and data encoding so that we are able to deliver data at high speed among ntopng, nProbe and nProbe Cento.

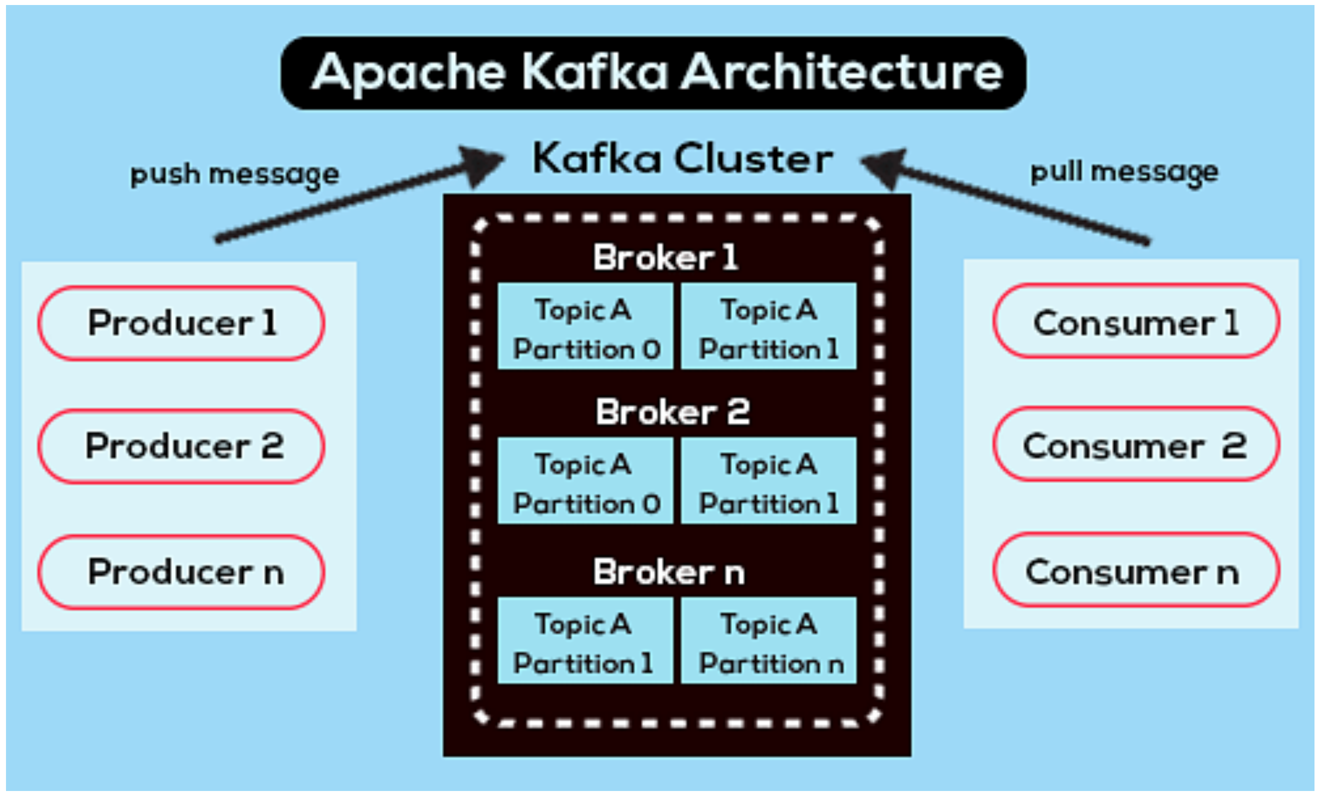


- nprobe -i eno1 --zmq://127.0.0.1:1234
- ntopng -i tcp://127.0.0.1:1234

From ZMQ to Kafka

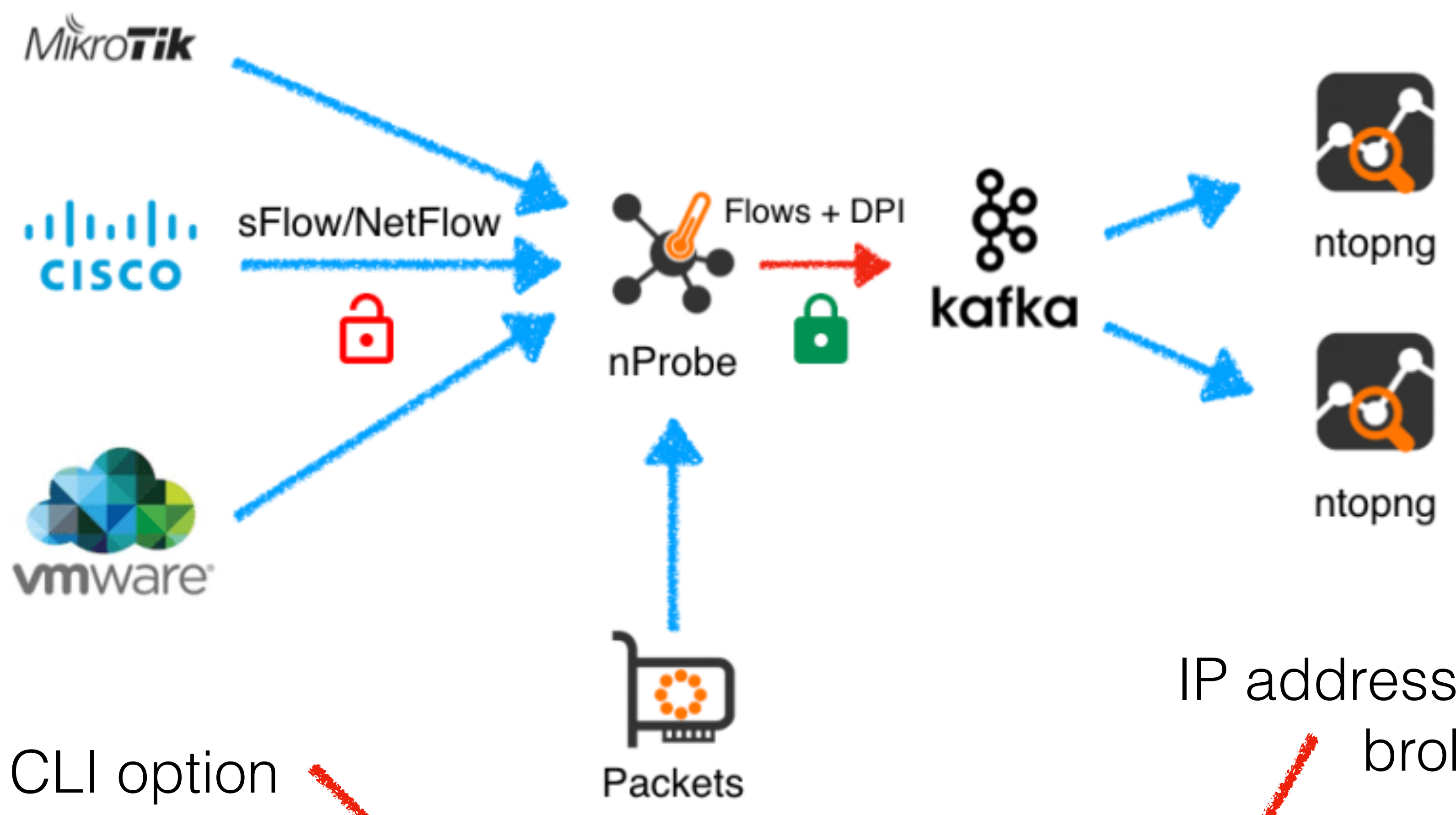
ZMQ is a broker-less solution whereas Kafka is broker-based that offers in particular some features not present in ZMQ:

- Persistency
- Fault Tolerance
- Replication
- Reliability



This has been the driving force for adding the option to use both ZMQ and Kafka for ntopng-IPC.

Using Kafka



New CLI option

IP address of the Kafka broker(s)

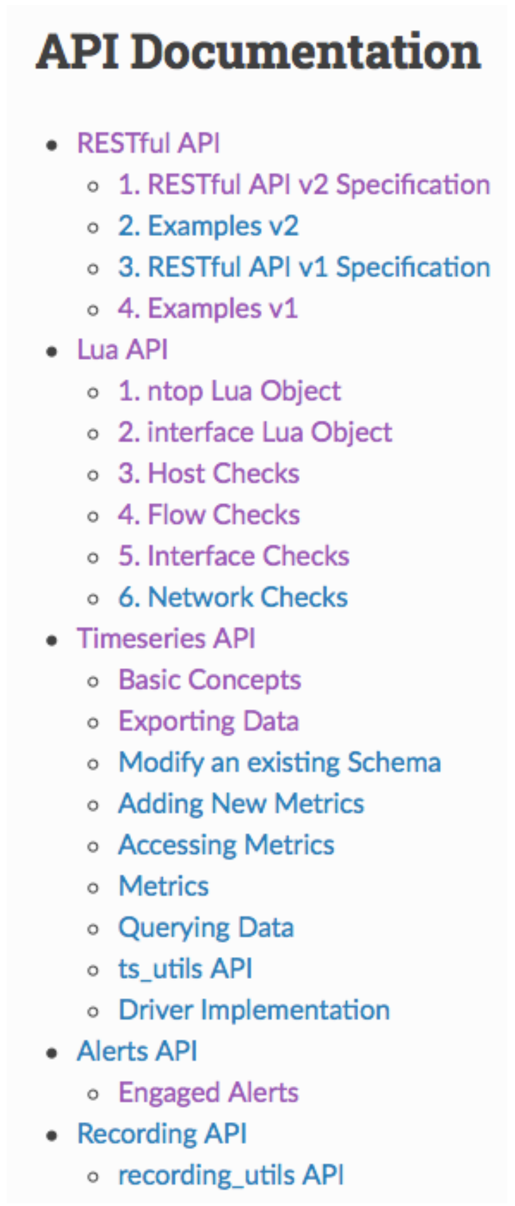
- nprobe -i eno1 --ntopng kafka://192.168.2.225
- ntopng -i kafka://192.168.2.225

- nprobe -i eno1 --ntopng zmq://127.0.0.1:1234
- ntopng -i zmq://127.0.0.1:1234

New Format

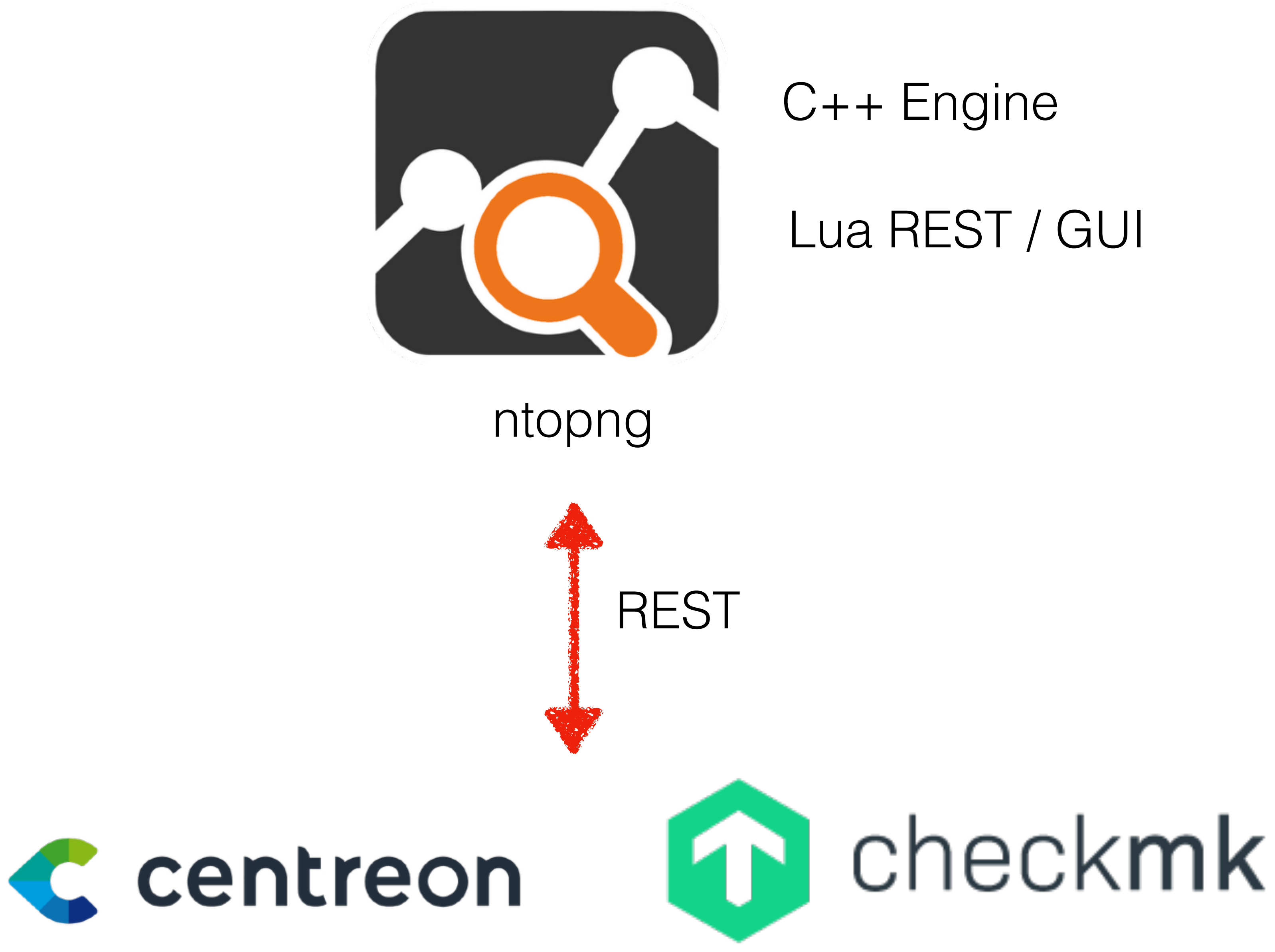
3. Scripting ntopng

ntopng APIs [1/2]



API Documentation

- RESTful API
  - 1. RESTful API v2 Specification
  - 2. Examples v2
  - 3. RESTful API v1 Specification
  - 4. Examples v1
- Lua API
  - 1. ntop Lua Object
  - 2. interface Lua Object
  - 3. Host Checks
  - 4. Flow Checks
  - 5. Interface Checks
  - 6. Network Checks
- Timeseries API
  - Basic Concepts
  - Exporting Data
  - Modify an existing Schema
  - Adding New Metrics
  - Accessing Metrics
  - Metrics
  - Querying Data
  - ts_utils API
  - Driver Implementation
- Alerts API
  - Engaged Alerts
- Recording API
  - recording_utils API



ntop

ntopng APIs [2/2]

- The ntopng engine is programmable in C++ for efficiency reasons.
- Lua scripts are limited to non critical tasks including:
  - Render the web GUI.
  - Perform periodic tasks (e.g. write timeseries or SNMP polling).
- North/South interaction is only via REST APIs.

What is missing in ntopng?

REST API is good for using ntopng as a server but it is not enough for many other tasks including the ability to address some use cases such as:

- Efficiently coding a behavioural check in Lua without affecting the overall performance. (e.g. trigger an alert when observing flows with certificate hash XXXX).
- Create some periodic “sanity checks” that allow users to enforce specific conditions (e.g. identify hosts that sent more than X GB to network Y, using protocol Z).
- Programmability using popular languages such as Python (i.e. similar to what pyshark is for Wireshark).

Using the Lua API for Behaviour Analysis [1/3]

- In addition to the existing Lua API used in the GUI, we have created a “nano size” Lua API used to create code able to run at a speed close to C++.
- Trigger alerts based on specific host and flow conditions:
    - Host Lua API is called every minute on hosts to check specific conditions.
    - The Flow Lua API is called as soon as nDPI has completed its detection.

WORK IN PROGRESS

Using the Lua API for Behaviour Analysis [2/3]

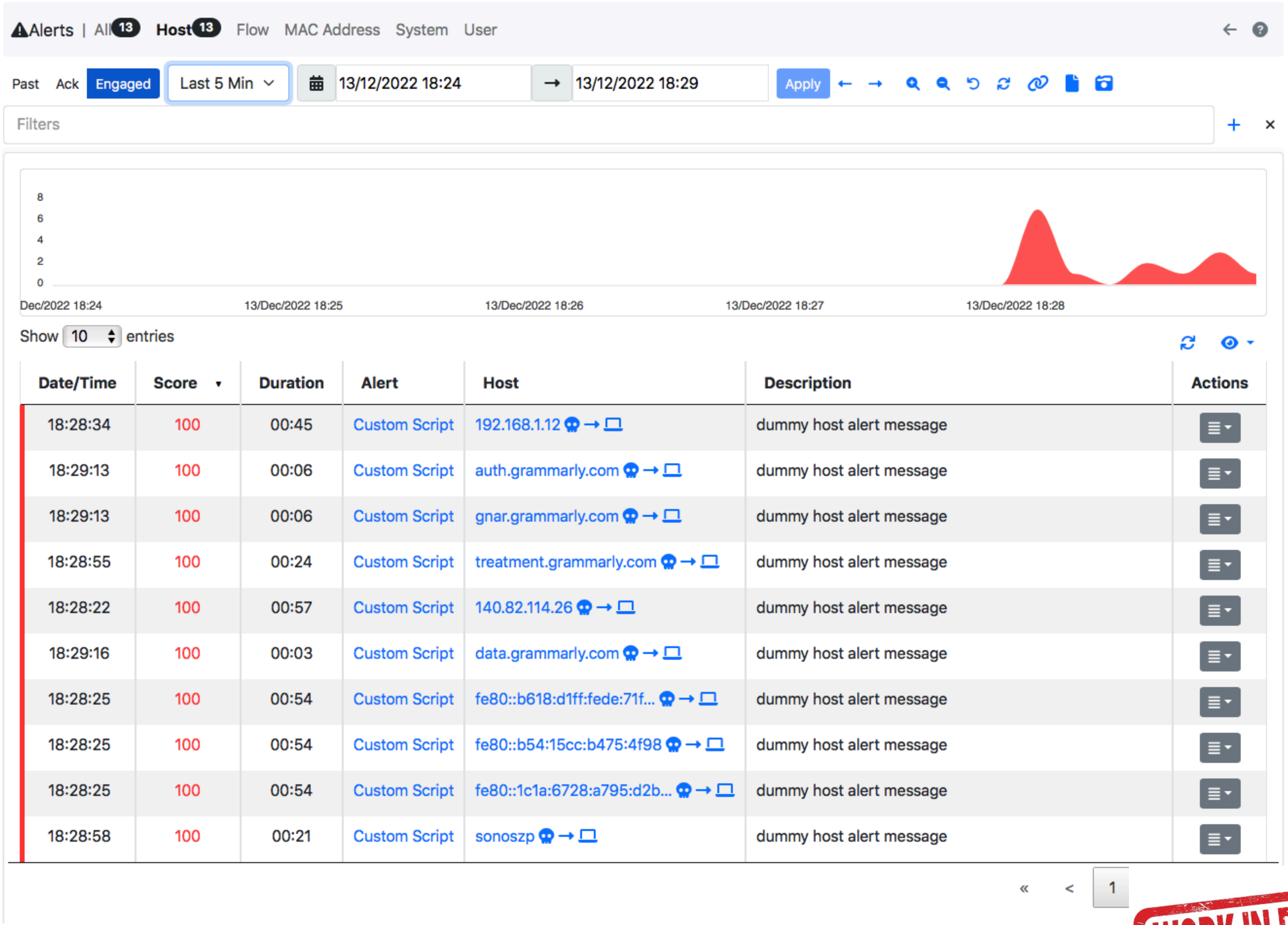
```
$ cat scripts/callbacks/checks/hosts/custom_host_lua_script.lua
--
-- (C) 2013-22 - ntop.org
--

--
-- NOTE: this script is called periodically (i.e. every minute) for every host
--       that ntopng has in memory
--

if(host.bytes_sent() > host.bytes_rcvd()) then
   local score   = 100
   local message = "dummy host alert message"

   host.triggerAlert(score, message)
   host.skipVisitedHost()
end

-- IMPORTANT: do not forget this return at the end of the script
return(0)
```

Script Performance Overhead: 0.78 usec/host

Tested on 3,2 GHz Intel Core i3 (2010)

WORK IN PROGRESS

Using the Lua API for Behaviour Analysis [3/3]



Alerts | All 13 Host 13 Flow MAC Address System User
Past Ack Engaged Last 5 Min
13/12/2022 18:24 → 13/12/2022 18:29 Apply
Filters
8
6
4
2
0
Dec/2022 18:24
13/Dec/2022 18:25
13/Dec/2022 18:26
13/Dec/2022 18:27
13/Dec/2022 18:28
Show 10 entries
« < 1

Date/Time	Score	Duration	Alert	Host	Description	Actions
18:28:34	100	00:45	Custom Script	192.168.1.12	dummy host alert message	
18:29:13	100	00:06	Custom Script	auth.grammarly.com	dummy host alert message	
18:29:13	100	00:06	Custom Script	gnar.grammarly.com	dummy host alert message	
18:28:55	100	00:24	Custom Script	treatment.grammarly.com	dummy host alert message	
18:28:22	100	00:57	Custom Script	140.82.114.26	dummy host alert message	
18:29:16	100	00:03	Custom Script	data.grammarly.com	dummy host alert message	
18:28:25	100	00:54	Custom Script	fe80::b618:d1ff:fede:71f...	dummy host alert message	
18:28:25	100	00:54	Custom Script	fe80::b54:15cc:b475:4f98	dummy host alert message	
18:28:25	100	00:54	Custom Script	fe80::1c1a:6728:a795:d2b...	dummy host alert message	
18:28:58	100	00:21	Custom Script	sonoszp	dummy host alert message	

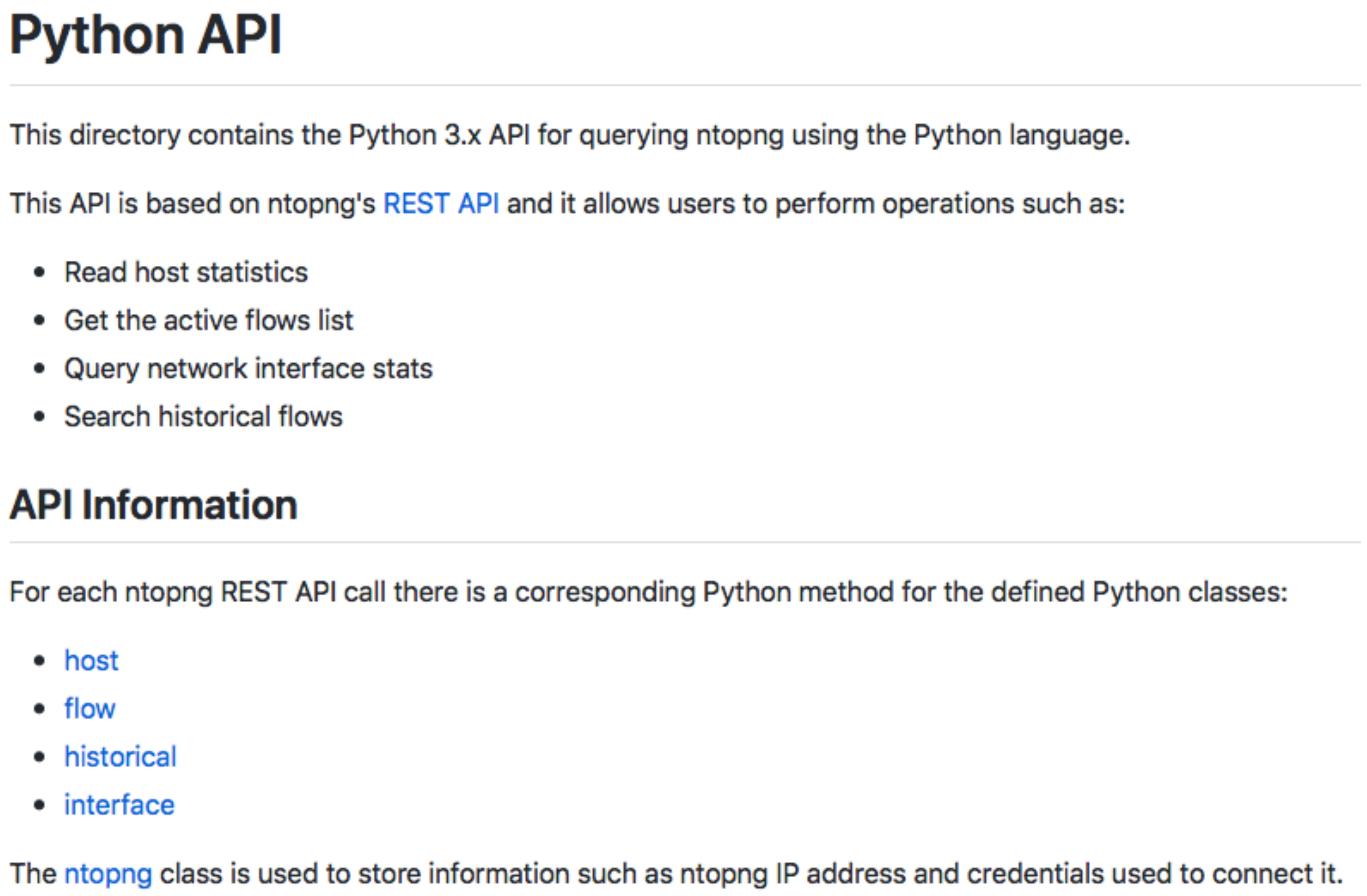
WORK IN PROGRESS

ntop

Python API [1/2]

- This API is designed to withdraw information from ntopng rather than to modify the ntopng behaviours as you do with the Lua API.
- Python is a popular language that can be used by unskilled users to use ntopng as a powerful tool for traffic analysis and cybersecurity.
- Through this API, we envisage the creation of ntopng-based monitoring applications that can take advantage of the monitoring engine we have created.

Python API [2/2]



Python API

This directory contains the Python 3.x API for querying ntopng using the Python language.

This API is based on ntopng's REST API and it allows users to perform operations such as:

- Read host statistics
- Get the active flows list
- Query network interface stats
- Search historical flows

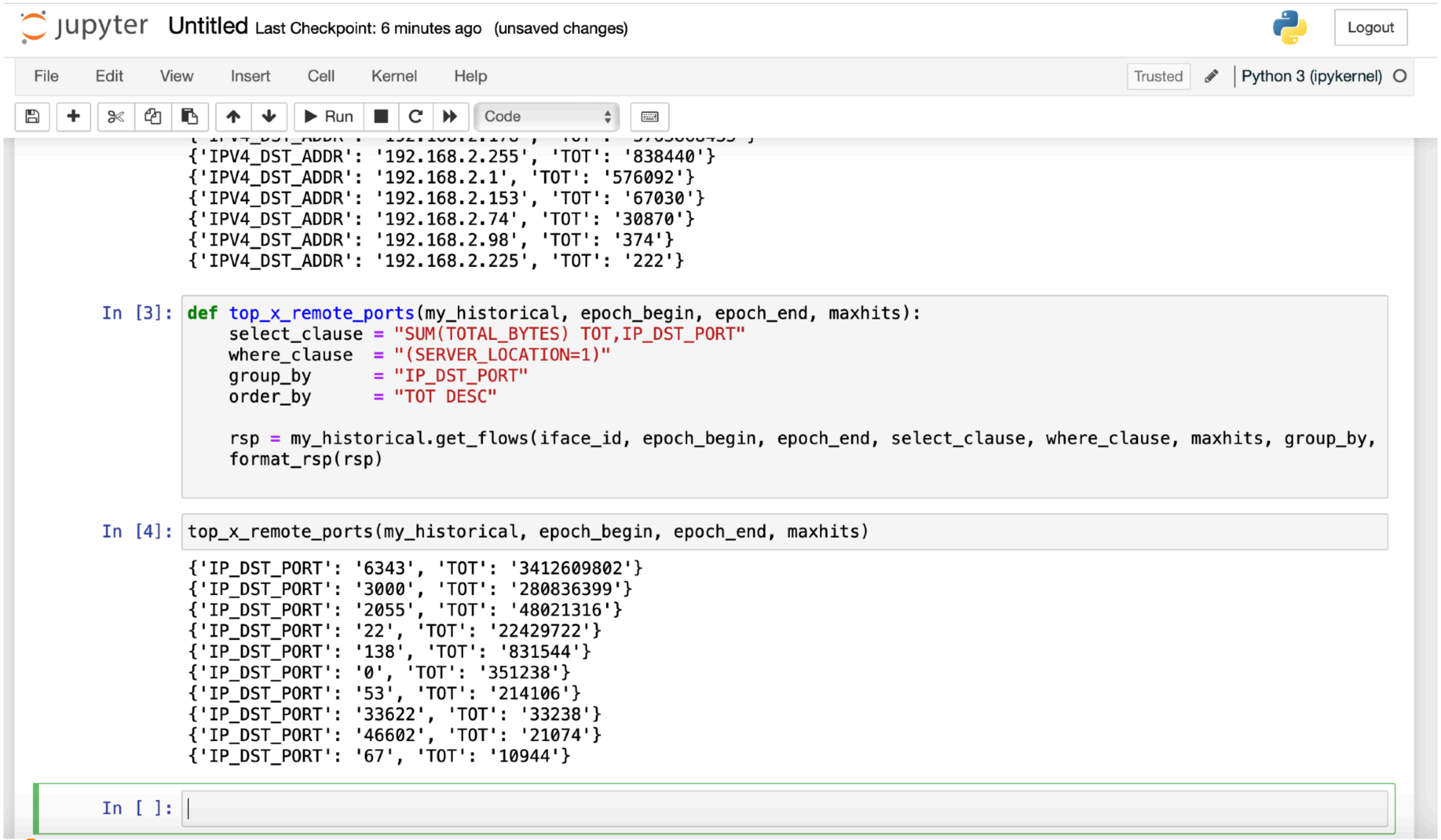
API Information

For each ntopng REST API call there is a corresponding Python method for the defined Python classes:

- host
- flow
- historical
- interface

The ntopng class is used to store information such as ntopng IP address and credentials used to connect it.

ntopng in Jupyter Notebook


Jupyter Untitled Last Checkpoint: 6 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3 (ipykernel)
Run
Code

```
{'IPV4_DST_ADDR': '192.168.2.255', 'TOT': '838440'}
{'IPV4_DST_ADDR': '192.168.2.1', 'TOT': '576092'}
{'IPV4_DST_ADDR': '192.168.2.153', 'TOT': '67030'}
{'IPV4_DST_ADDR': '192.168.2.74', 'TOT': '30870'}
{'IPV4_DST_ADDR': '192.168.2.98', 'TOT': '374'}
{'IPV4_DST_ADDR': '192.168.2.225', 'TOT': '222'}
```

```
In [3]: def top_x_remote_ports(my_historical, epoch_begin, epoch_end, maxhits):
            select_clause = "SUM(TOTAL_BYTES) TOT,IP_DST_PORT"
            where_clause  = "(SERVER_LOCATION=1)"
            group_by      = "IP_DST_PORT"
            order_by      = "TOT DESC"

            rsp = my_historical.get_flows(iface_id, epoch_begin, epoch_end, select_clause, where_clause, maxhits, group_by,
            format_rsp(rsp)
```

```
In [4]: top_x_remote_ports(my_historical, epoch_begin, epoch_end, maxhits)

        {'IP_DST_PORT': '6343', 'TOT': '3412609802'}
        {'IP_DST_PORT': '3000', 'TOT': '280836399'}
        {'IP_DST_PORT': '2055', 'TOT': '48021316'}
        {'IP_DST_PORT': '22', 'TOT': '22429722'}
        {'IP_DST_PORT': '138', 'TOT': '831544'}
        {'IP_DST_PORT': '0', 'TOT': '351238'}
        {'IP_DST_PORT': '53', 'TOT': '214106'}
        {'IP_DST_PORT': '33622', 'TOT': '33238'}
        {'IP_DST_PORT': '46602', 'TOT': '21074'}
        {'IP_DST_PORT': '67', 'TOT': '10944'}
```

```
In [ ]:
```

Some Future ntopng Plans for 2023 Q1 Release

- Store Timeseries in ClickHouse
- Scalability: support ClickHouse cluster where multiple ntopng write flows.
- Add host similarity for detecting equally behaving hosts.
- Improved ObservationPoint concept to split incoming traffic more efficiently than with sub-interfaces to promote scalability.

Summary

- In 1Q23 we will release a new ntopng release that includes all these new features that we’re developing.
- Out goal is to make ntopng fully scriptable and versatile creating a new generation of efficient yet programmatically open traffic analysis applications able to do things that no other tools is able to do today.
- We will schedule new “vertical” webinars in Jan/Feb timeframe to go deep into these new features and refine the tools before the release.