Matyáš Prokop

Principle Architect, Natilik

Starting off as a founder of Bubakov.net (a community-led ISP) whilst still in his teens, seeing the potential of a good network, is in Matyas' DNA. He has since designed data centers at Google and now stands as Principal Architect at Natilik focusing on service providers, datacenters and DevOps. Alongside this he organizes Network Automation Meet-Ups in London, constantly pushing boundaries and supporting this ever-evolving and exciting industry.

Using Streaming Telemetry for Network Automation and Monitoring

MATYÁŠ PROKOP

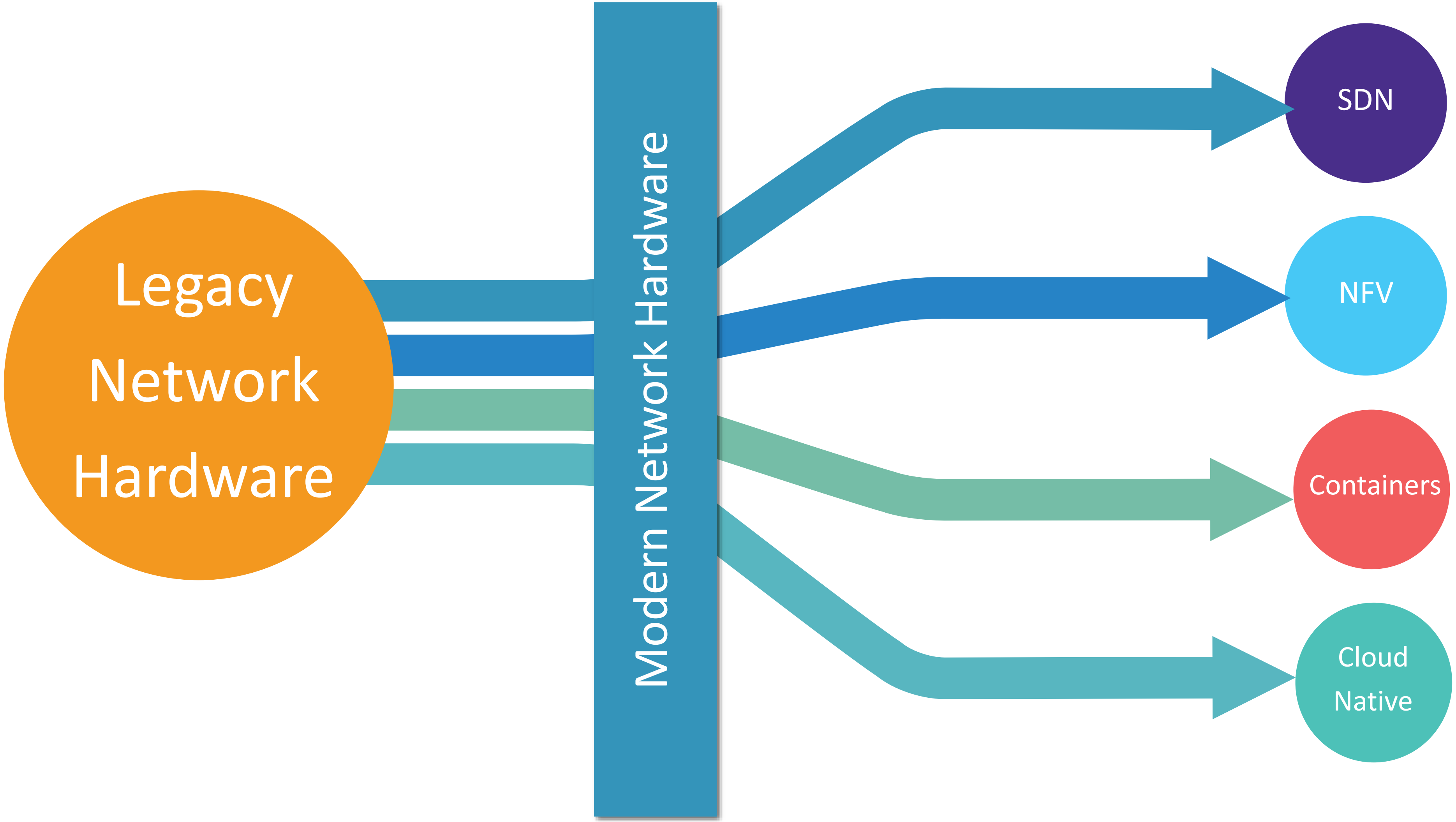
PRINCIPLE ARCHITECT, NATILIK

@MATY0609

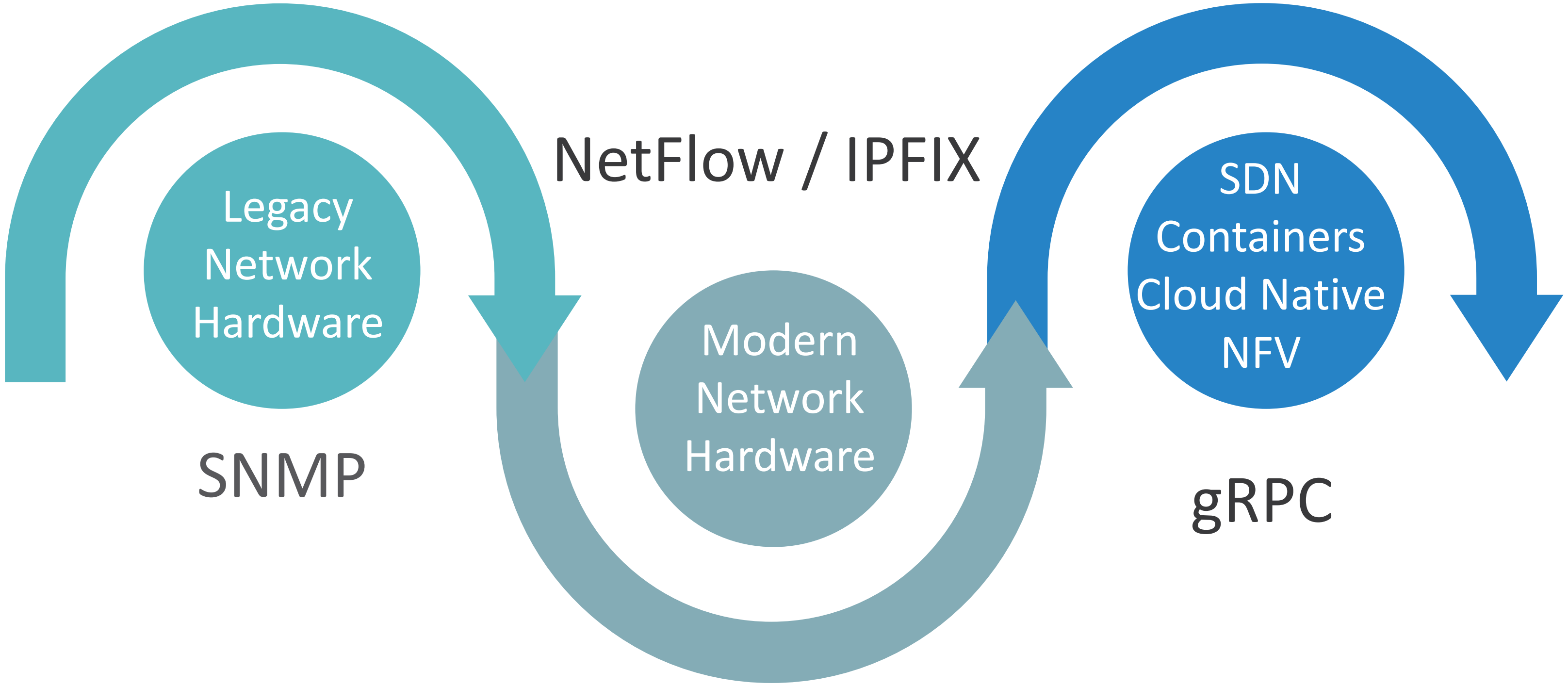
Agenda

- Evolution of the Network and Telemetry
- OpenConfig – gRPC, gNMI and gNOI
- Where gRPC sits in modern model based management?
- gRPC implementation and how to start?
- Proof of Concept
- The future

The Network Evolves....



The Evolution of Telemetry



Legacy Network Hardware

SNMP

NetFlow / IPFIX

Modern Network Hardware

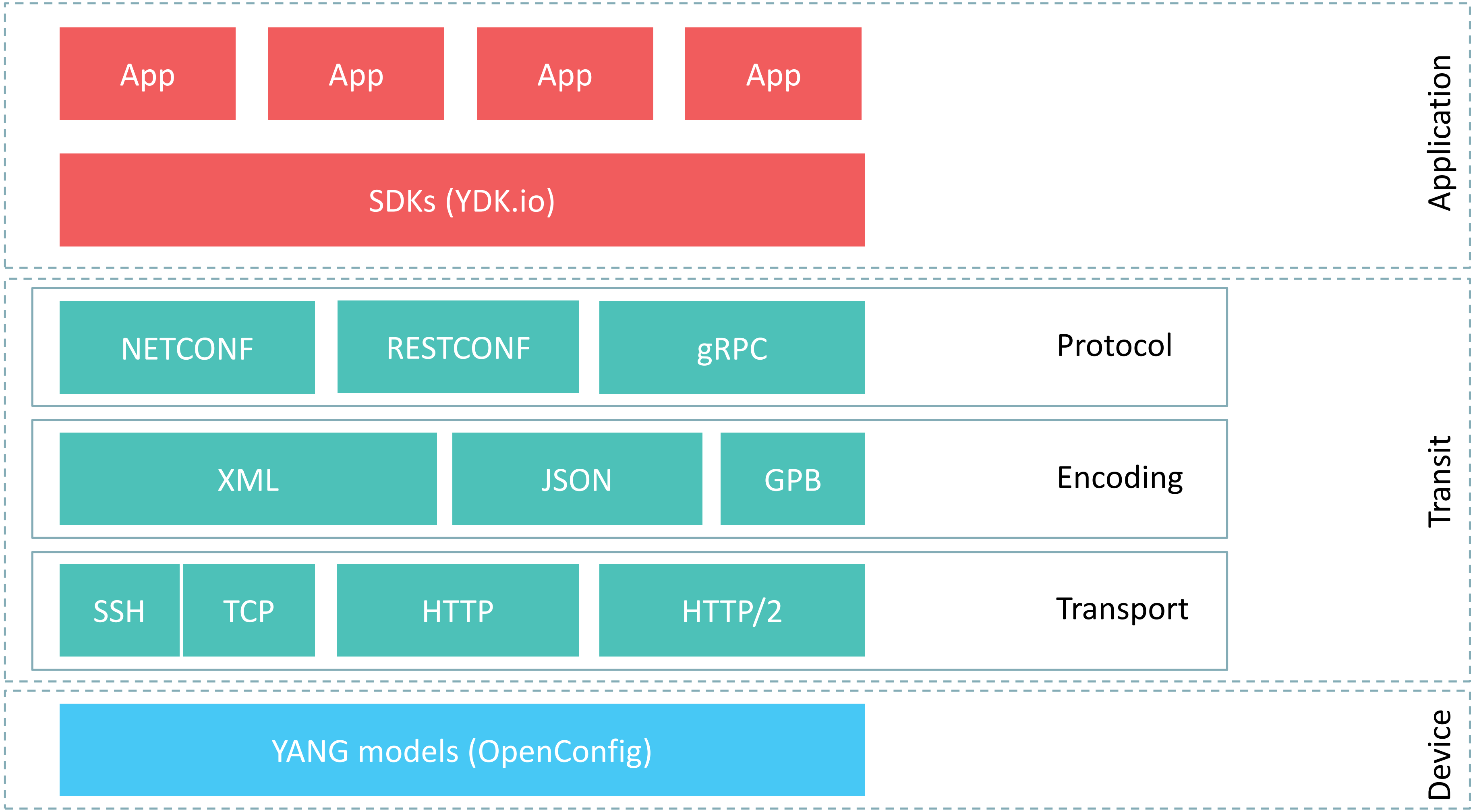
SDN Containers Cloud Native NFV

gRPC

gRPC Overview

- OpenConfig network management protocol
- gRPC is a open source framework that provides secure and reliable transport of data
- gRPC leverages standard HTTP/2 as its transport layer
- Real-time, bidirectional and standardized protocol

Modern model-based management



Application

App App App App

SDKs (YDK.io)

Transit

Protocol: NETCONF RESTCONF gRPC

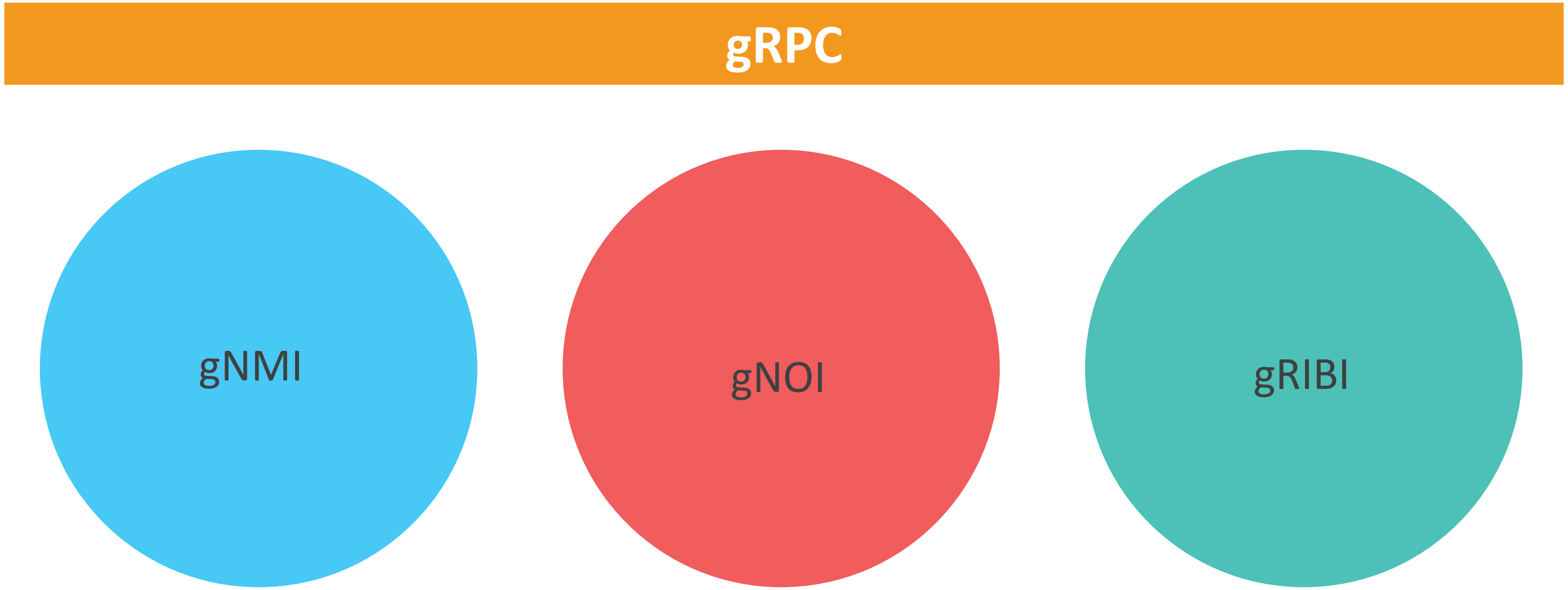
Encoding: XML JSON GPB

Transport: SSH TCP HTTP HTTP/2

Device

YANG models (OpenConfig)

OpenConfig Interfaces and protocols



gRPC

gNMI

gNOI

gRIBI

gRPC Network Management Interface

gRPC Network Operations Interface

gRPC Routing Information Base Interface

gRPC and gNMI – How does it work?

gRPC Capabilities

```
service gNMI {
        rpc Capabilities(CapabilityRequest) returns (CapabilityResponse);
}
```

gRPC Subscribe

```
service gNMI {
        rpc Subscribe(stream SubscribeRequest) returns (stream SubscribeResponse);
}
```

Source: https://github.com/openconfig/gnmi/blob/master/proto/gnmi/gnmi.proto

gRPC and gNMI – How does it work?

gRPC GET

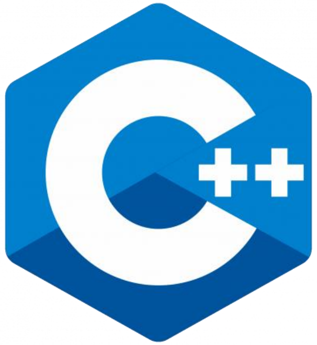
```
service gNMI {
        rpc Get(GetRequest) returns (GetResponse);
}
```

gRPC SET

```
service gNMI {
        rpc Set(SetRequest) returns (SetResponse);
}
```

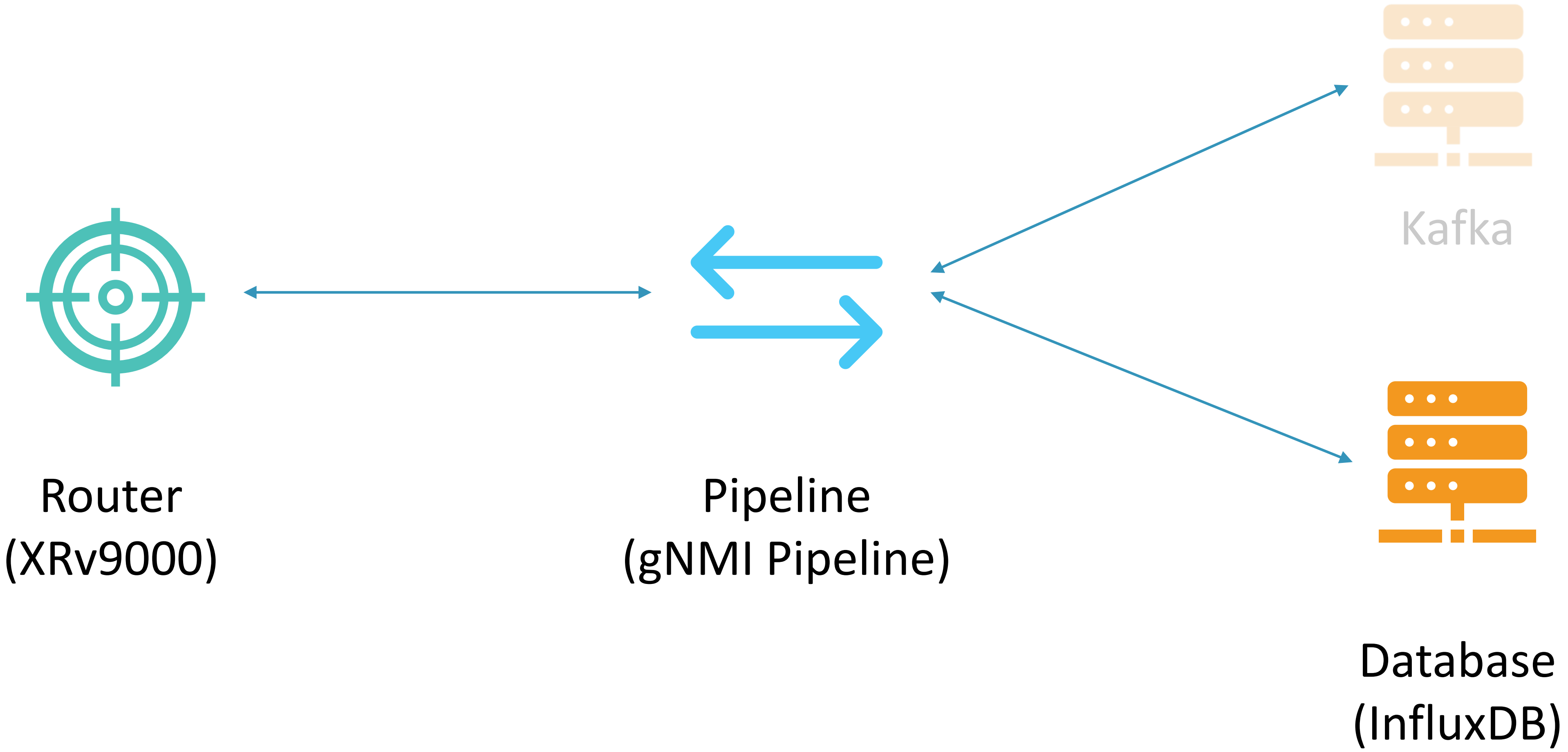
Source: https://github.com/openconfig/gnmi/blob/master/proto/gnmi/gnmi.proto

How to start?



ydk.io

gNMI - Proof of Concept



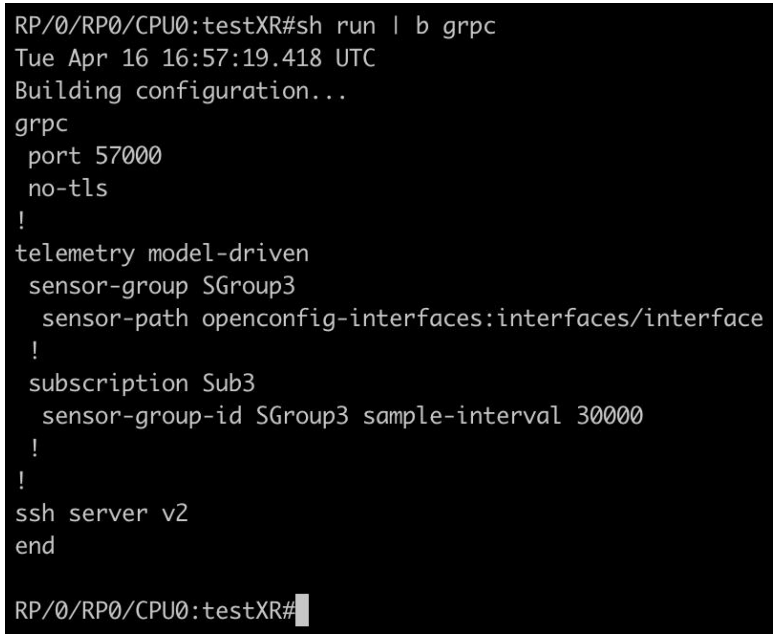
Router
(XRv9000)

Pipeline
(gNMI Pipeline)

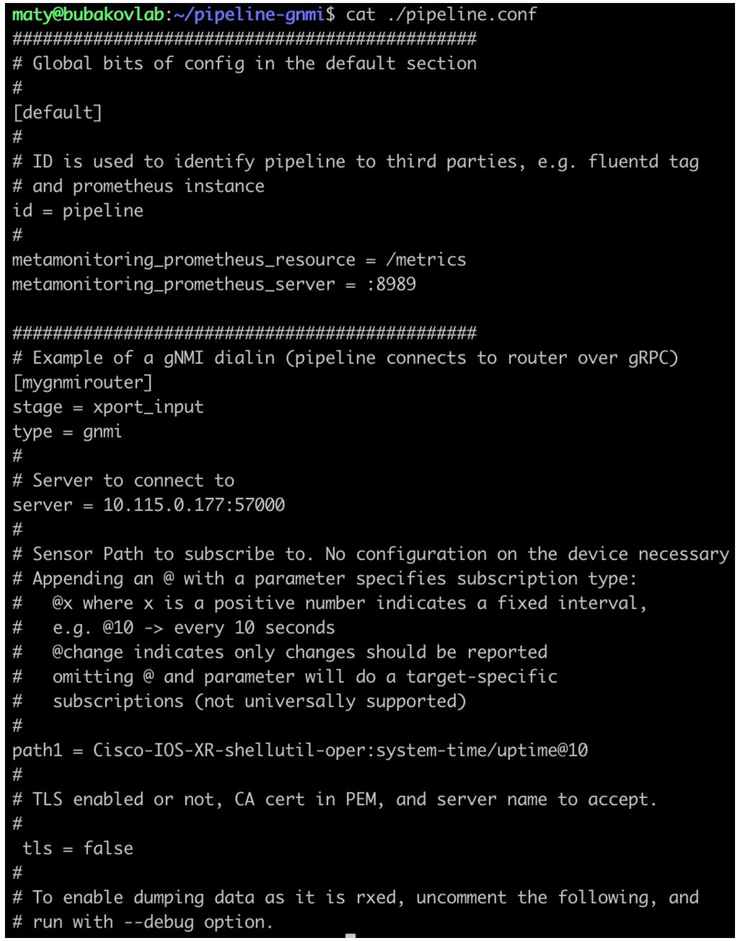
Kafka

Database
(InfluxDB)

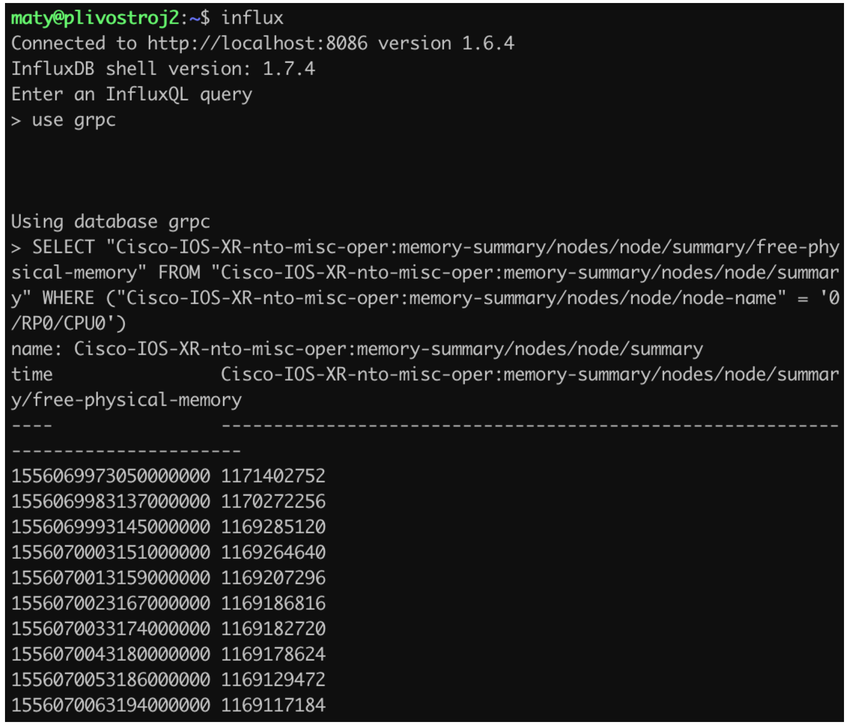
gNMI - Proof of Concept



```
RP/0/RP0/CPU0:testXR#sh run | b grpc
Tue Apr 16 16:57:19.418 UTC
Building configuration...
grpc
 port 57000
 no-tls
!
telemetry model-driven
 sensor-group SGroup3
  sensor-path openconfig-interfaces:interfaces/interface
 !
 subscription Sub3
  sensor-group-id SGroup3 sample-interval 30000
 !
!
ssh server v2
end

RP/0/RP0/CPU0:testXR#
```

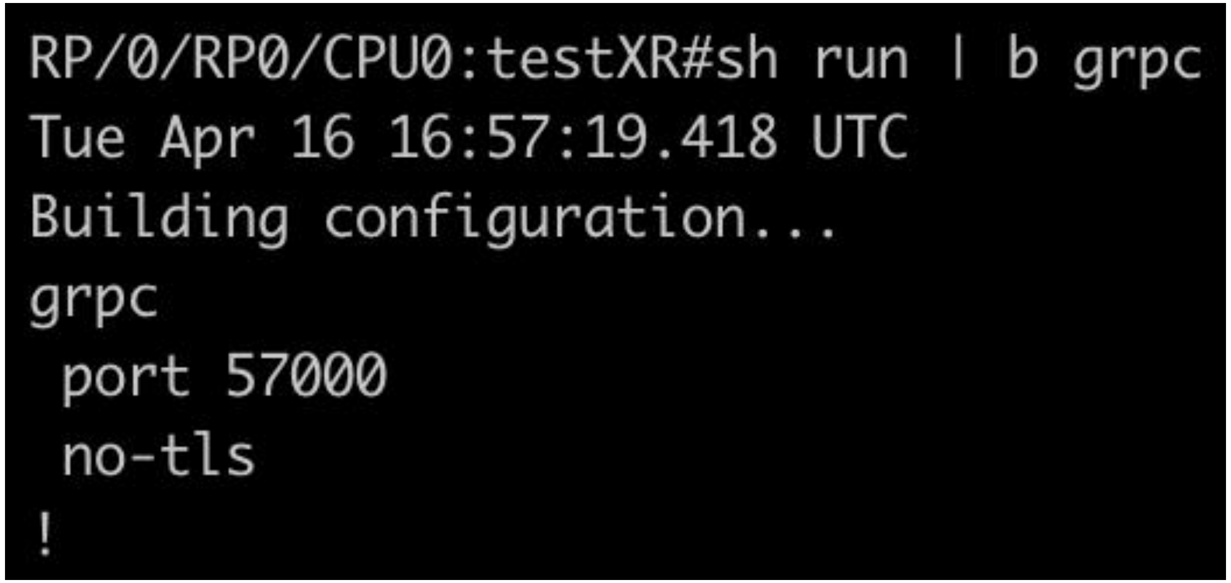


```
maty@bubakovlab:~/pipeline-gnmi$ cat ./pipeline.conf
##############################################
# Global bits of config in the default section
#
[default]
#
# ID is used to identify pipeline to third parties, e.g. fluentd tag
# and prometheus instance
id = pipeline
#
metamonitoring_prometheus_resource = /metrics
metamonitoring_prometheus_server = :8989

##############################################
# Example of a gNMI dialin (pipeline connects to router over gRPC)
[mygnmirouter]
stage = xport_input
type = gnmi
#
# Server to connect to
server = 10.115.0.177:57000
#
# Sensor Path to subscribe to. No configuration on the device necessary
# Appending an @ with a parameter specifies subscription type:
#   @x where x is a positive number indicates a fixed interval,
#   e.g. @10 -> every 10 seconds
#   @change indicates only changes should be reported
#   omitting @ and parameter will do a target-specific
#   subscriptions (not universally supported)
#
path1 = Cisco-IOS-XR-shellutil-oper:system-time/uptime@10
#
# TLS enabled or not, CA cert in PEM, and server name to accept.
#
 tls = false
#
# To enable dumping data as it is rxed, uncomment the following, and
# run with --debug option.
```

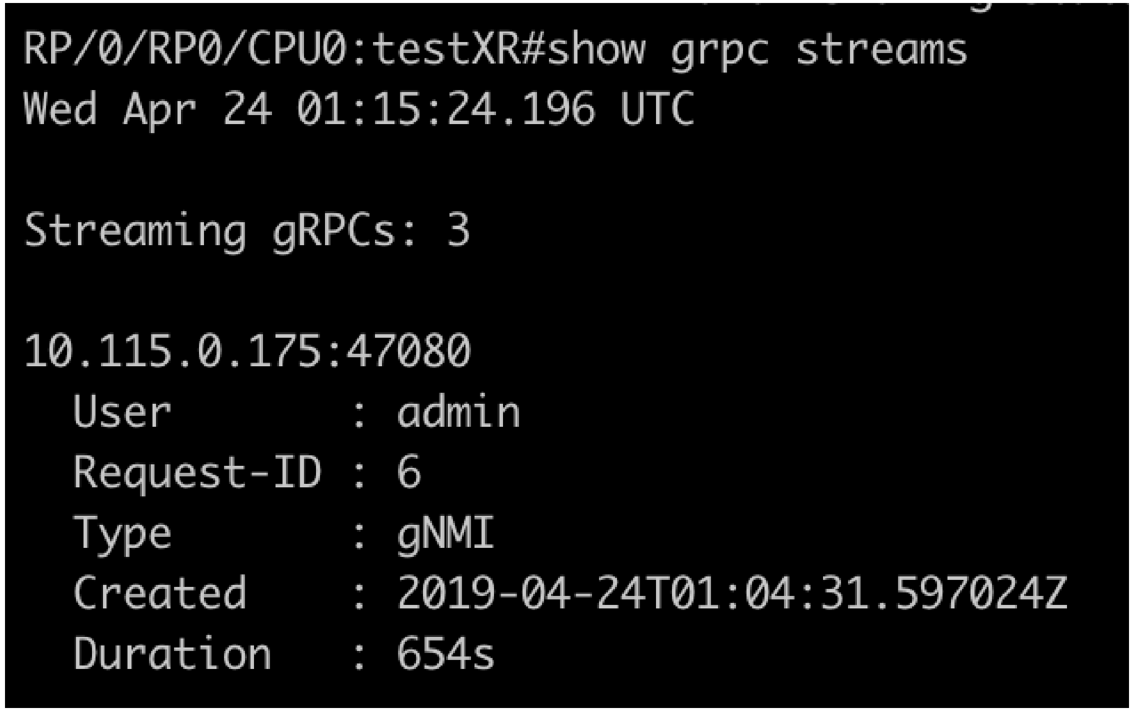


```
maty@plivostroj2:~$ influx
Connected to http://localhost:8086 version 1.6.4
InfluxDB shell version: 1.7.4
Enter an InfluxQL query
> use grpc



Using database grpc
> SELECT "Cisco-IOS-XR-nto-misc-oper:memory-summary/nodes/node/summary/free-phy
sical-memory" FROM "Cisco-IOS-XR-nto-misc-oper:memory-summary/nodes/node/summar
y" WHERE ("Cisco-IOS-XR-nto-misc-oper:memory-summary/nodes/node/node-name" = '0
/RP0/CPU0')
name: Cisco-IOS-XR-nto-misc-oper:memory-summary/nodes/node/summary
time                Cisco-IOS-XR-nto-misc-oper:memory-summary/nodes/node/summar
y/free-physical-memory
----                -----------------------------------------------------------
----------------------
1556069973050000000 1171402752
1556069983137000000 1170272256
1556069993145000000 1169285120
1556070003151000000 1169264640
1556070013159000000 1169207296
1556070023167000000 1169186816
1556070033174000000 1169182720
1556070043180000000 1169178624
1556070053186000000 1169129472
1556070063194000000 1169117184
```

gNMI - Proof of Concept

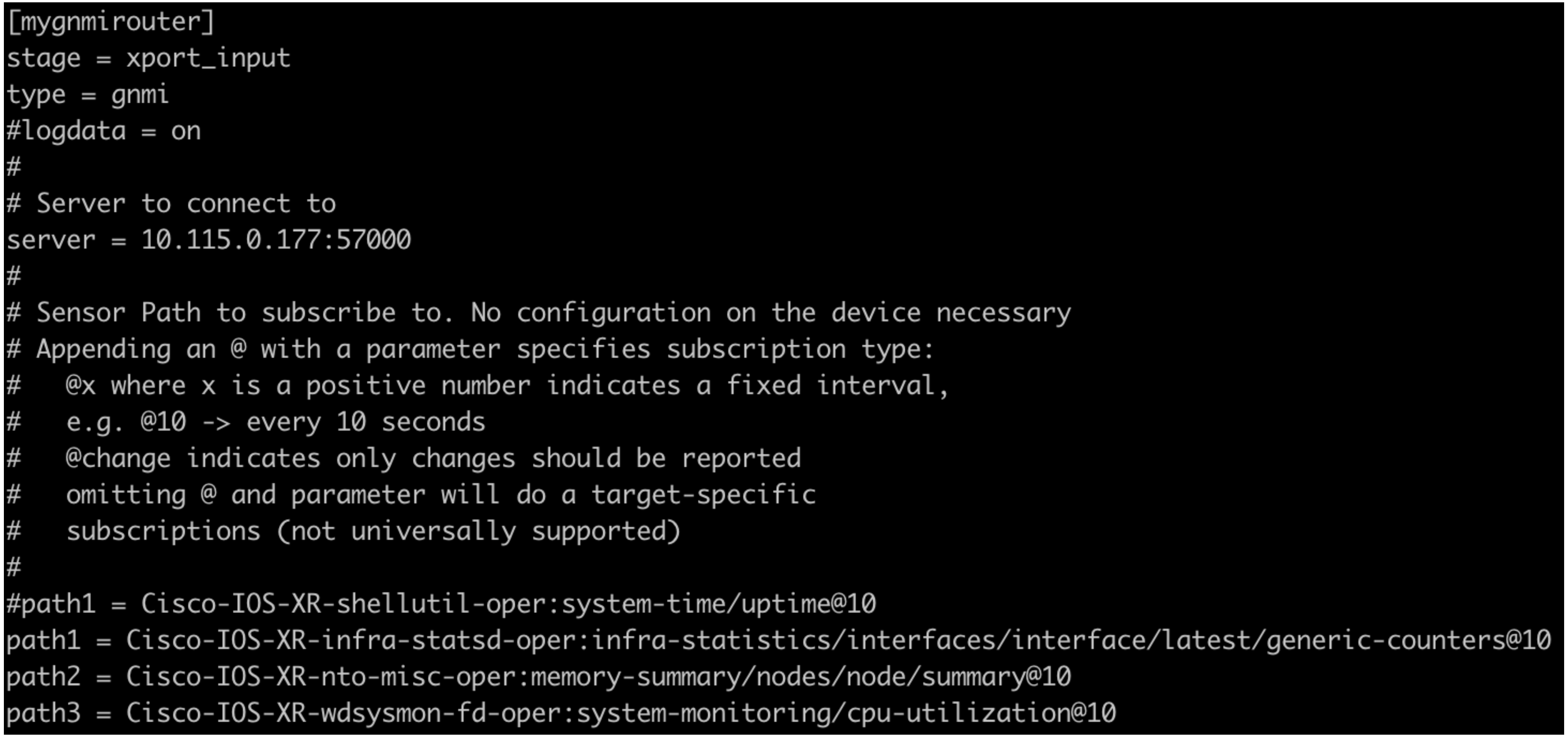


```
RP/0/RP0/CPU0:testXR#sh run | b grpc
Tue Apr 16 16:57:19.418 UTC
Building configuration...
grpc
 port 57000
 no-tls
!
```



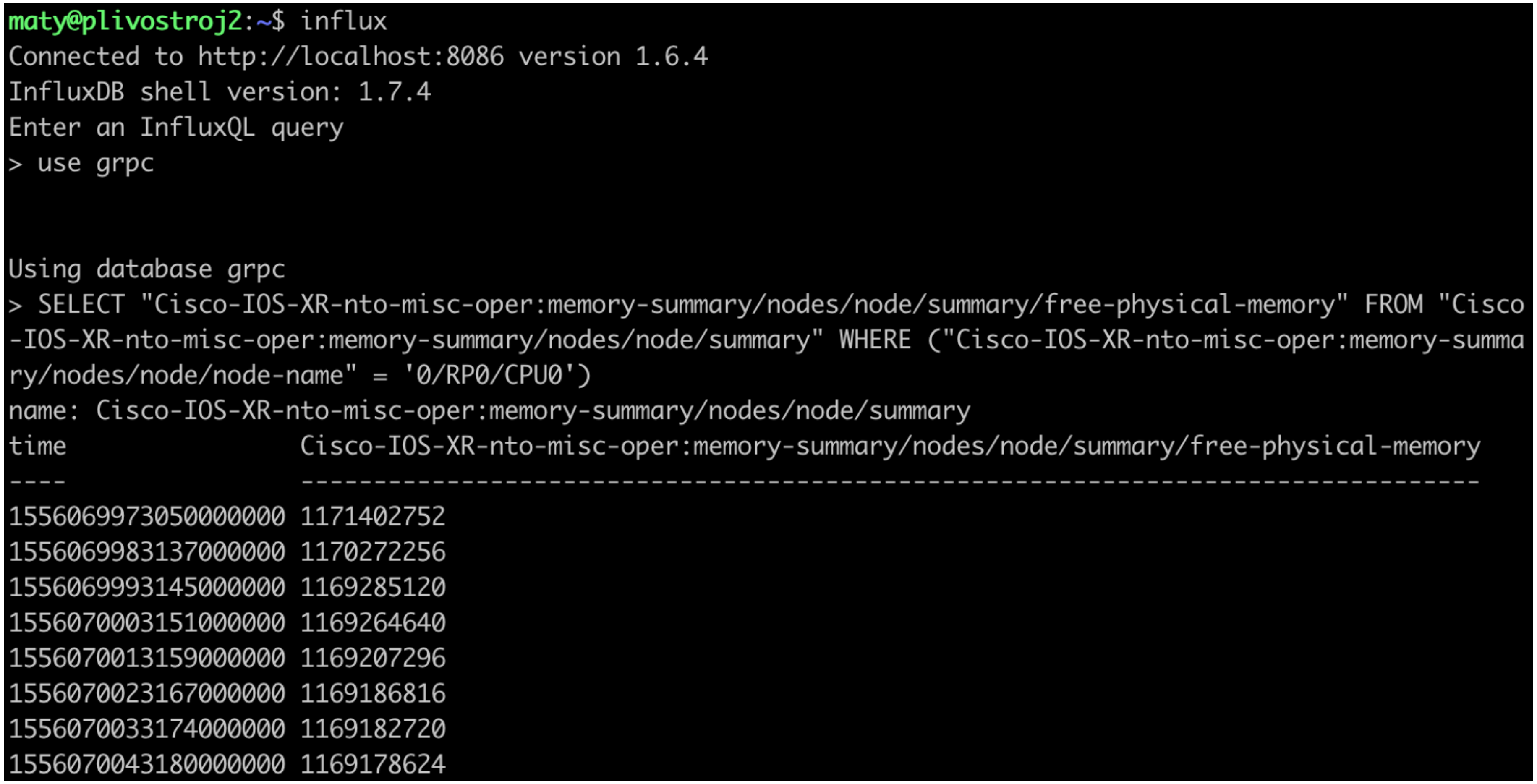
```
RP/0/RP0/CPU0:testXR#show grpc streams
Wed Apr 24 01:15:24.196 UTC

Streaming gRPCs: 3

10.115.0.175:47080
  User       : admin
  Request-ID : 6
  Type       : gNMI
  Created    : 2019-04-24T01:04:31.597024Z
  Duration   : 654s
```

gNMI - Proof of Concept



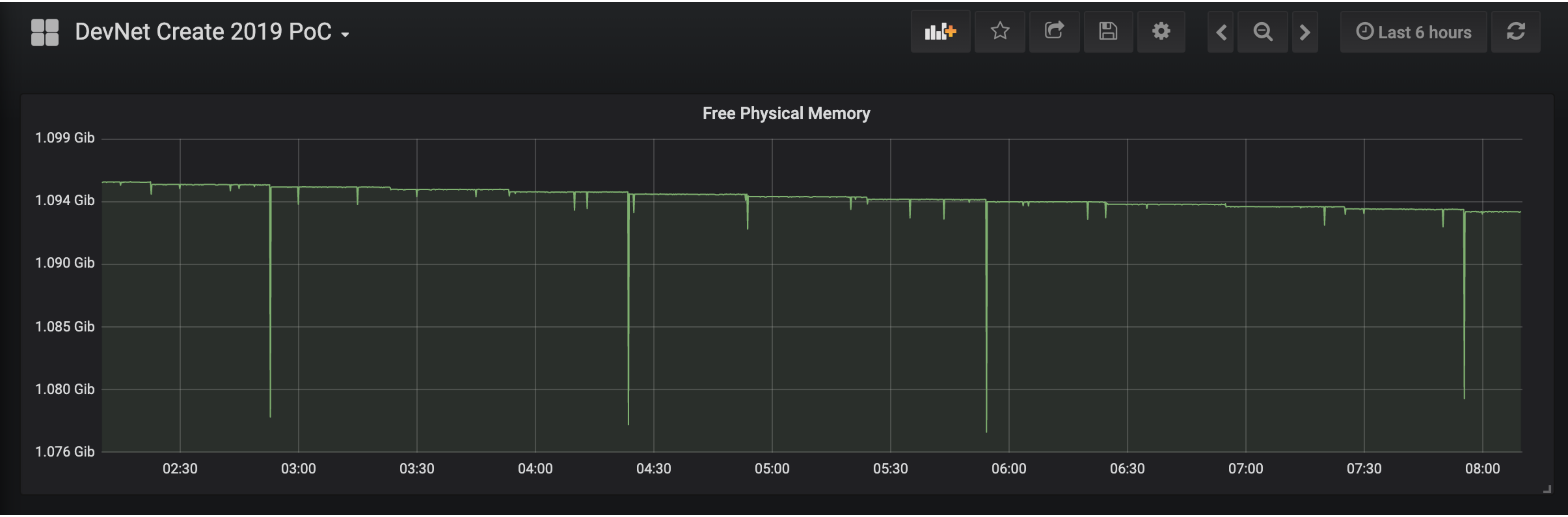
```
[mygnmirouter]
stage = xport_input
type = gnmi
#logdata = on
#
# Server to connect to
server = 10.115.0.177:57000
#
# Sensor Path to subscribe to. No configuration on the device necessary
# Appending an @ with a parameter specifies subscription type:
#    @x where x is a positive number indicates a fixed interval,
#    e.g. @10 -> every 10 seconds
#    @change indicates only changes should be reported
#    omitting @ and parameter will do a target-specific
#    subscriptions (not universally supported)
#
#path1 = Cisco-IOS-XR-shellutil-oper:system-time/uptime@10
path1 = Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/latest/generic-counters@10
path2 = Cisco-IOS-XR-nto-misc-oper:memory-summary/nodes/node/summary@10
path3 = Cisco-IOS-XR-wdsysmon-fd-oper:system-monitoring/cpu-utilization@10
```

gNMI - Proof of Concept



```
maty@plivostroj2:~$ influx
Connected to http://localhost:8086 version 1.6.4
InfluxDB shell version: 1.7.4
Enter an InfluxQL query
> use grpc


Using database grpc
> SELECT "Cisco-IOS-XR-nto-misc-oper:memory-summary/nodes/node/summary/free-physical-memory" FROM "Cisco
-IOS-XR-nto-misc-oper:memory-summary/nodes/node/summary" WHERE ("Cisco-IOS-XR-nto-misc-oper:memory-summa
ry/nodes/node/node-name" = '0/RP0/CPU0')
name: Cisco-IOS-XR-nto-misc-oper:memory-summary/nodes/node/summary
time                Cisco-IOS-XR-nto-misc-oper:memory-summary/nodes/node/summary/free-physical-memory
----                ---------------------------------------------------------------------------------
1556069973050000000 1171402752
1556069983137000000 1170272256
1556069993145000000 1169285120
1556070003151000000 1169264640
1556070013159000000 1169207296
1556070023167000000 1169186816
1556070033174000000 1169182720
1556070043180000000 1169178624
```

gNMI - Proof of Concept



gRPC Implementations

- Not Google and DevOps only
- NX-OS gRPC
- IOS XR gRPC and gNMI
- IOS XE gRPC and gNMI (limited)
- Arista EOS
- Junos OS
- Kubernetes

Future

- De facto standard for application and network health check
- gRPC will help to disaggregate the network functions
- A more responsive network allows greater adoption of automation

Stay in touch

mprokop@natilik.com
email@matyasprokop.com

@maty0609

https://git.matyasprokop.com