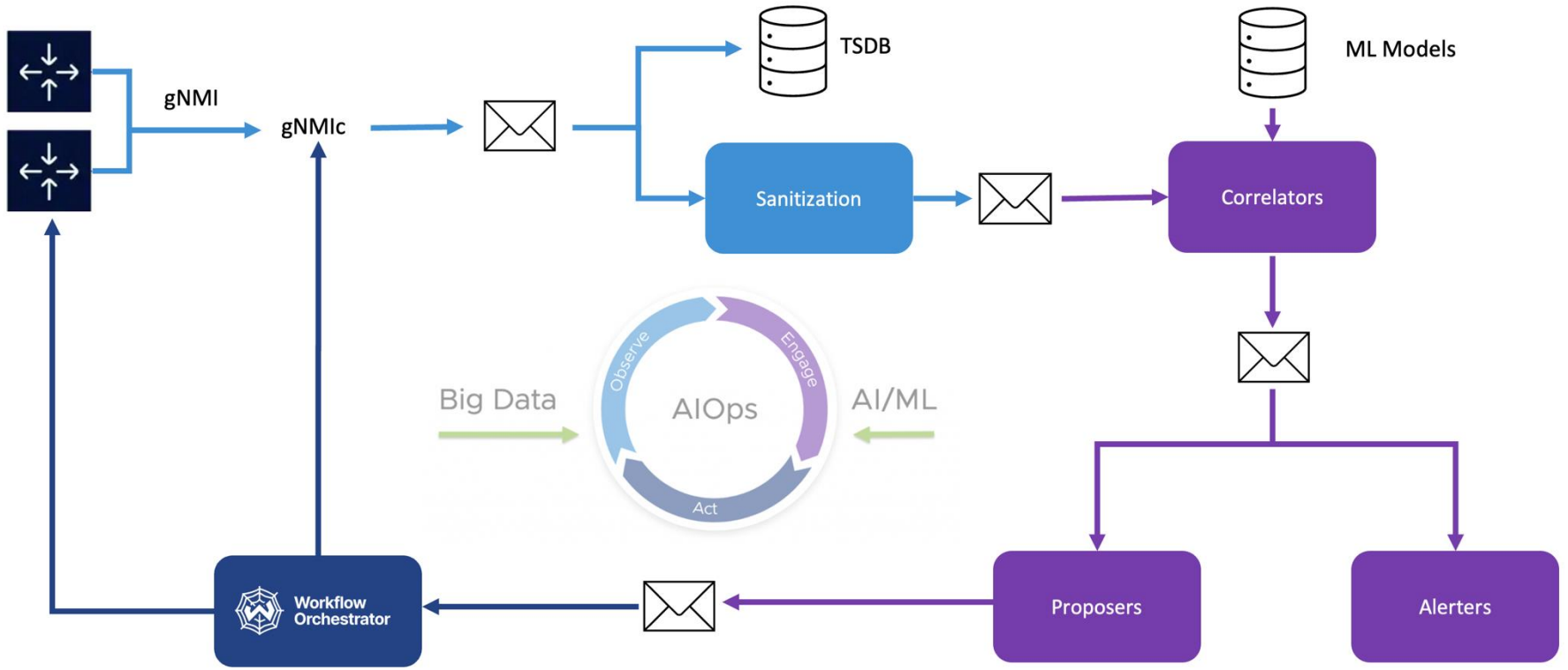
WFO – Telemetry Module – gnmic-cluster-chart

Peter Boers - SURF

Whoami?

- Peter Boers
- Software Engineer/Architect for the networking department @SURF
- Tech-Lead of the Workfloworchestrator Programme
- Involved in all sorts of Automation and Orchestration endeavours over the past 12 years, 8 of which @SURF

Origins of the idea….

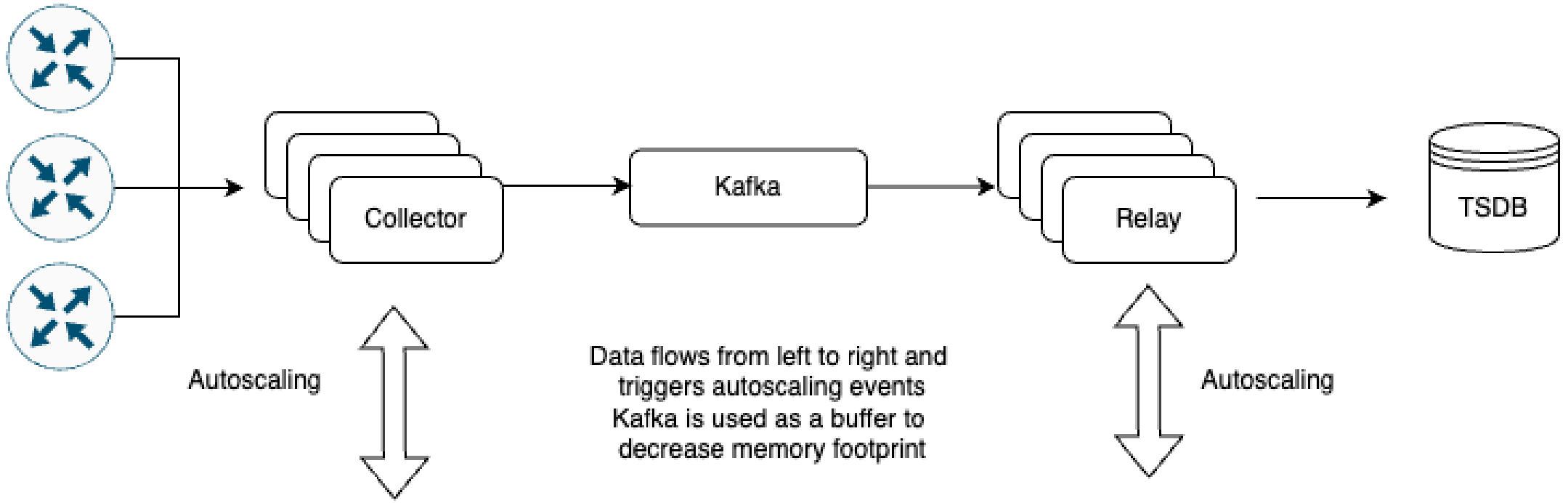


Project Rationale

- Rationale for creating this project:
  - Experimenting with upcoming technologies
  - Exploring solutions for vendor-agnostic monitoring
  - SURFnet∞
  - NMS-Like look and feel for the WFO orchestrator
  - ML
  - Fase out of legacy monitoring like SNMP in favor of gNMI
- Project goal: Develop a highly scalable telemetry solution based on the gNMI protocol

Introduction to the Tech Stack

gnmic-cluster-chart



WFO telemetry module tech stack

- gNMIc - gNMI collector
- OpenConfig – vendor agnostic
- Kubernetes – scalability
- Kafka - scalability
- Prometheus/InfluxDB – storage
- Helm – package management <- gnmic-cluster chart

WFO telemetry module tech stack

- **gNMIc - gNMI collector**
- OpenConfig – vendor agnostic
- Kubernetes – scalability
- Kafka - scalability
- Prometheus/InfluxDB – storage
- Helm – package management <- gnmic-cluster chart

gNMI(c)

- gRPC Network Management Interface
  - Protocol for the modification and retrieval of configuration form a target device as wel as the control and generation of telemetry streams from a target device to a data collection system.
  - https://datatracker.ietf.org/meeting/98/materials/slides-98-rtgwg-gnmi-intro-draft-openconfig-rtgwg-gnmi-spec-00
- Uses HTTP/2 to setup a bi-directional secure communication channel between router and subscriber
- Efficient use of resources built into the protocol
  - Less buffering and less information sent by the router

gNMI(c)

Streaming telemetry benefits over SNMP

- devices stream data based on a specified frequency or upon state change
- data is sent as soon as it is available, reducing the need to buffer
- no single large request for all data (unlike SNMP polling)
- data sent incrementally, e.g., only for those data items that have changed
- ability to distribute the telemetry sources (e.g., directly to linecards)
- users issue subscription requests via RPC for data of interest
- data exported in a well-structured, common format, e.g., based on YANG models
- device and collector communicate over a secure, authenticated, reliable channel

- https://datatracker.ietf.org/meeting/98/materials/slides-98-rtgwg-gnmi-intro-draft-openconfig-rtgwg-gnmi-spec-00

gNMI(c)

- gNMIc is a gNMI capable client application.
  - Built in go-lang
  - Able to be setup in a clustered manner (scalable)
- Pluggable
  - Different outputs and inputs
  - Data pipeline support
- Clustering
- Collector mode – collects from the routers
- Relay mode – streams data to the TSDB
- Easily processes 100k+ events p/s

- https://gnmic.openconfig.net/

Tool choices

- gNMIc - gNMI collector
- **OpenConfig – vendor agnostic**
- Kubernetes – scalability
- Kafka - scalability
- Prometheus/InfluxDB – storage
- Helm – package management <- gnmic-cluster chart

OpenConfig

- Vendor-neutral, model-driven network management
- Defined in Yang
- “Vendor Agnostic”
  - Not all implementations are equal across all vendors
- Lots of support, but incomplete

- https://openconfig.net/

WFO telemetry module tech stack

- gNMIc - gNMI collector
- OpenConfig – vendor agnostic
- **Kubernetes – scalability**
- Kafka - scalability
- Prometheus/InfluxDB – storage
- Helm – package management <- gnmic-cluster chart

Kubernetes

- Obvious choice for setting up scalable workloads
- StatefulSet workload
  - Pods need to communicate cluster state to one another
  - Statefulset is suitable for target re-allocation
- Kubernetes leases for target locking
  - Depends on scale and requirements
- HorizontalPodAutoscaling
  - For scaling against predefined CPU or Mem usage
  - Future work is to scale with a custom metric e.g: Locked targets

WFO telemetry module tech stack

- gNMIc - gNMI collector
- OpenConfig – vendor agnostic
- Kubernetes – scalability
- **Kafka - scalability**
- Prometheus/InfluxDB – storage
- Helm – package management <- gnmic-cluster chart

Kafka

- Kafka is used as a buffer between collector and relay
- Potentially can be used as a place to enact filtering and/or event processing before storage in the TSDB
- Enables other tools to act on the raw telemetry stream
- Significantly decreases the memory footprint the collectors and relays need
- Buffers billions of messages quite easily
- Easily deployed on Kubernetes with the strimzi operator

- https://strimzi.io

WFO telemetry module tech stack

- gNMIc - gNMI collector
- OpenConfig – vendor agnostic
- Kubernetes – scalability
- Kafka - scalability
- Prometheus/**InfluxDB – storage**
- Helm – package management <- gnmic-cluster chart

Time Series Database (TSDB)

- This is where you store your network events
- gNMIc supports Prometheus and InfluxDB
  - Prometheus is pull based
  - InfluxDB is push based
- @SURF we work with InfluxDB
- The discussion of which TSDB to use and why was is of scope for this project.

WFO telemetry module tech stack

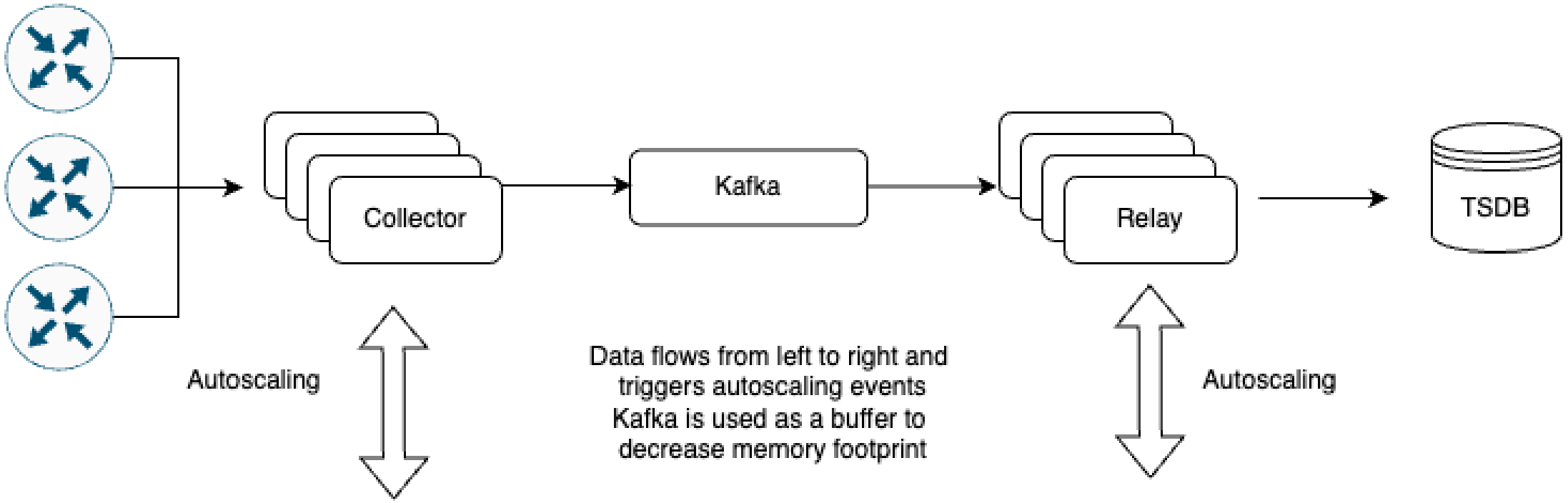
- gNMIc - gNMI collector
- OpenConfig – vendor agnostic
- Kubernetes – scalability
- Kafka - scalability
- Prometheus/InfluxDB – storage
- **Helm – package management <- gnmic-cluster chart**

Helm

- Helm is the tool we used to package all parts of this softwarestack
- It provides a documented, standardised and user friendly manner to deploy on Kubernetes
- It allows enough customisation to setup the software…
- …. but at the same it makes sure you do not have to worry about the kubernetes specifics
- **A helm chart is one of the outputs of this project.**

Deployment architecture

Architecture diagram

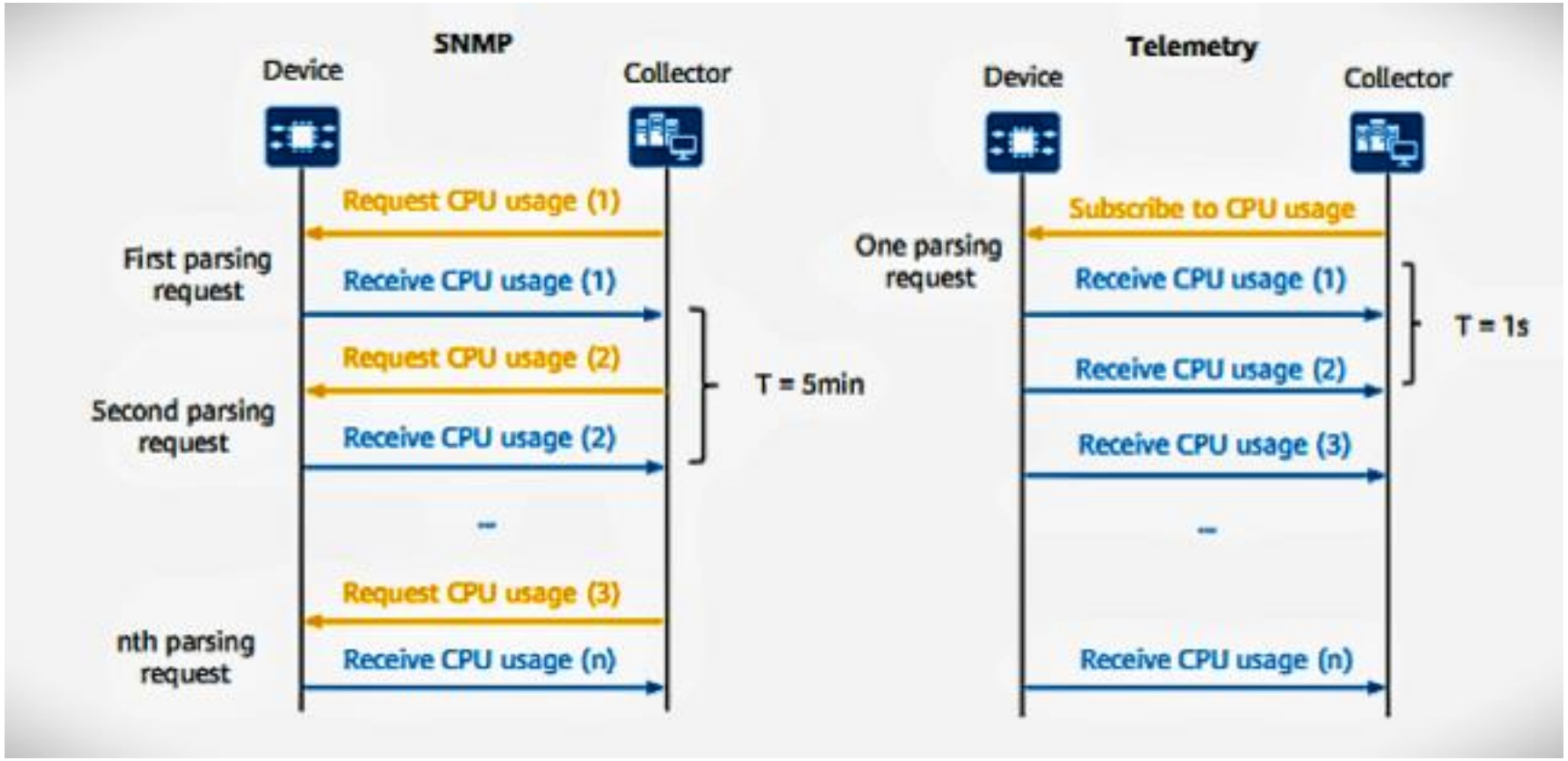


How does gNMIc work?

- It executes RPC calls on the Router
  - Capabilities
  - Get RPC
  - Set RPC
  - Subscribe RPC
- It uses an x-path like notation to describe to the router what information nodes it would like to stream
  - /interfaces/interface[name=xe-0/0/0]/state/counters
  - /components/component/state
- May use vendor specific yang or OpenConfig yang
  - https://openconfig.net/projects/models/paths/
  - https://apps.juniper.net/telemetry-explorer/

What is the difference between gNMI and SNMP?

- Stream modes:
  - Target Defined
  - Once
  - Sampled
  - On change
- Multiple information sources within one request



https://thinkpalm.com/blogs/a-guide-to-gnmi-grpc-how-are-they-revolutionizing-network-management/

How does a single gNMIc process work?

```yaml
username: admin
password: NokiaSrl1!
insecure: true
encoding: json_ietf
log: true

targets:
  # Add targets configuration here
  # eg:
  # 192.168.1.131:57400:
  #   username: gnmic
  #   password: secret_password


subscriptions:
  # Add subscriptions configuration here
  # e.g:
  # sub1:
  #   paths:
  #     - /interface/statistics
  #   stream-mode: sample
  #   sample-interval: 1s

outputs:
  influxdb-output:
    type: influxdb
    url: http://influxdb:8086
    bucket: telemetry # db name
    token: gnmic:gnmic # username:password
    batch-size: 1000
    flush-timer: 10s
```

```
❯ gnmic -a ut042a-jnx-03.dcn.surf.net:32767 --config config-prd.yml  subscribe --path  "/interfaces" --mode stream --stream-mode target_defined
{
  "source": "ut042a-jnx-03.dcn.surf.net:32767",
  "subscription-name": "default-1730792452",
  "timestamp": 1730792457015440836,
  "time": "2024-11-05T08:40:57.015440836+01:00",
  "prefix": "interfaces/interface[name=em2]/subinterfaces/subinterface[index=32768]",
  "updates": [
    {
      "Path": "state/counters/in-octets",
      "values": {
        "state/counters/in-octets": 31085082158
      }
    },
    {
      "Path": "state/counters/in-pkts",
      "values": {
        "state/counters/in-pkts": 89336731
      }
    },
    {
      "Path": "state/counters/out-octets",
      "values": {
        "state/counters/out-octets": 5061128757
      }
    },
    {
      "Path": "state/counters/out-pkts",
      "values": {
        "state/counters/out-pkts": 49717692
      }
    }
  ]
}
```

https://gnmic.openconfig.net/deployments/deployments_intro/

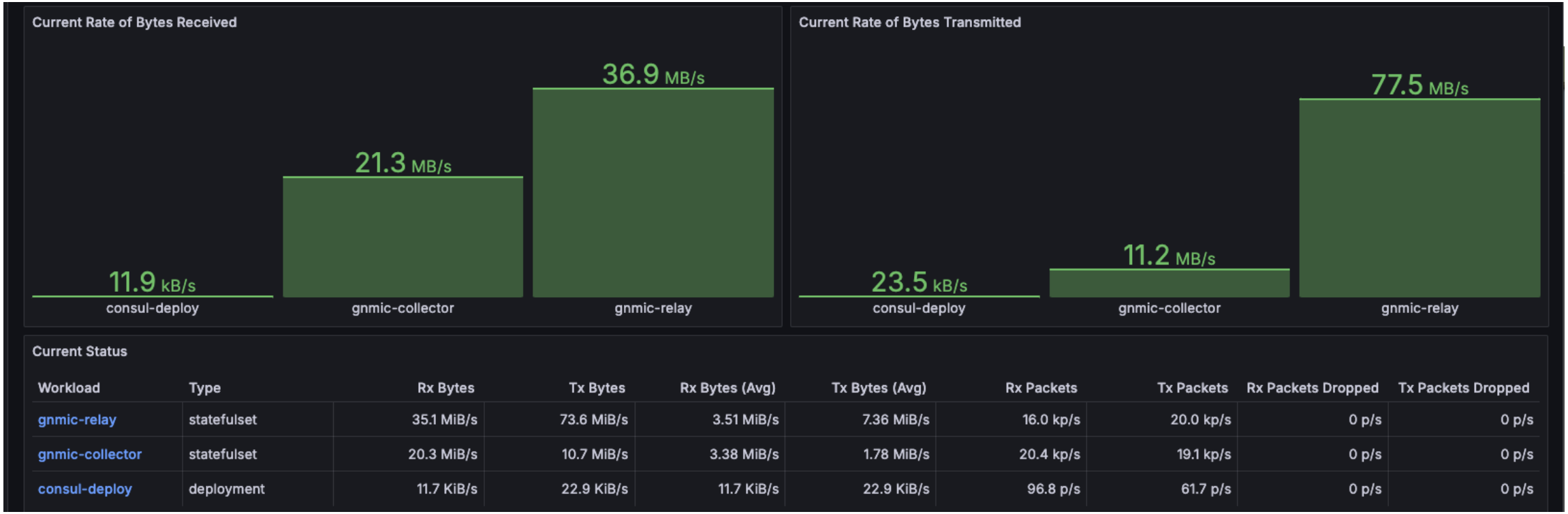
How to handle scale?

- A single router produces thousands of events per minute if you subscribe to ”/”
- This quickly requires too many resources from a single instance when scaling the amount of routers
- What do you need to do when you have around 400 nodes as is the case @SURF
- **gNMIc Clustering mode**

Scaling is complex, what do you run into?

- Resource limits
  - CPU and Memory
- I/O
  - Network – Currently using around +/- 25MB/s in +/- 75MB/s out
  - Storage – TSDB has a certain way of writing to disk (batched)
- Target acquisition and distribution
  - gNMIc cluster mode
- Application constraints
  - Many Routers
  - One database
- Kubernetes API Rate limits
  - In some cases
- **Continuous trade-off**
  - To compress or not to compress???

Scaling is complex, what do you run into?

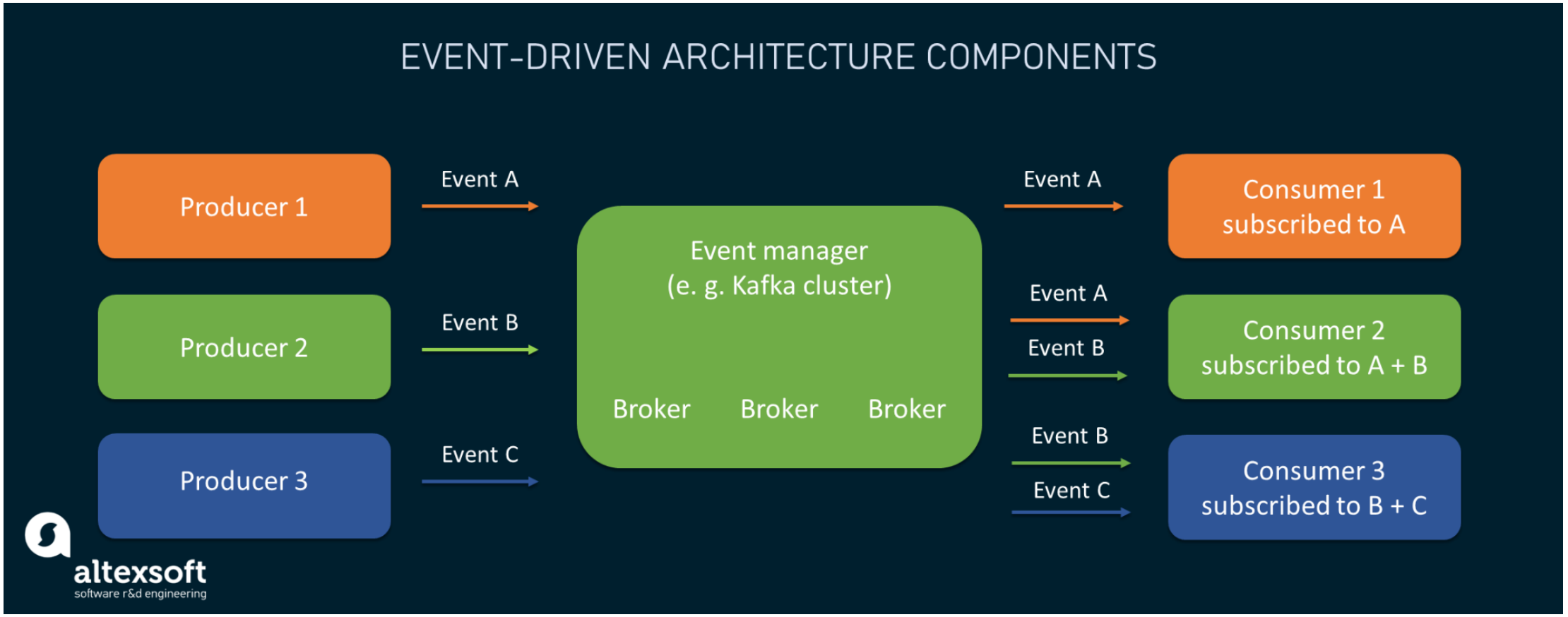


Workload	Type	Rx Bytes	Tx Bytes	Rx Bytes (Avg)	Tx Bytes (Avg)	Rx Packets	Tx Packets	Rx Packets Dropped	Tx Packets Dropped
gnmic-relay	statefulset	35.1 MiB/s	73.6 MiB/s	3.51 MiB/s	7.36 MiB/s	16.0 kp/s	20.0 kp/s	0 p/s	0 p/s
gnmic-collector	statefulset	20.3 MiB/s	10.7 MiB/s	3.38 MiB/s	1.78 MiB/s	20.4 kp/s	19.1 kp/s	0 p/s	0 p/s
consul-deploy	deployment	11.7 KiB/s	22.9 KiB/s	11.7 KiB/s	22.9 KiB/s	96.8 p/s	61.7 p/s	0 p/s	0 p/s

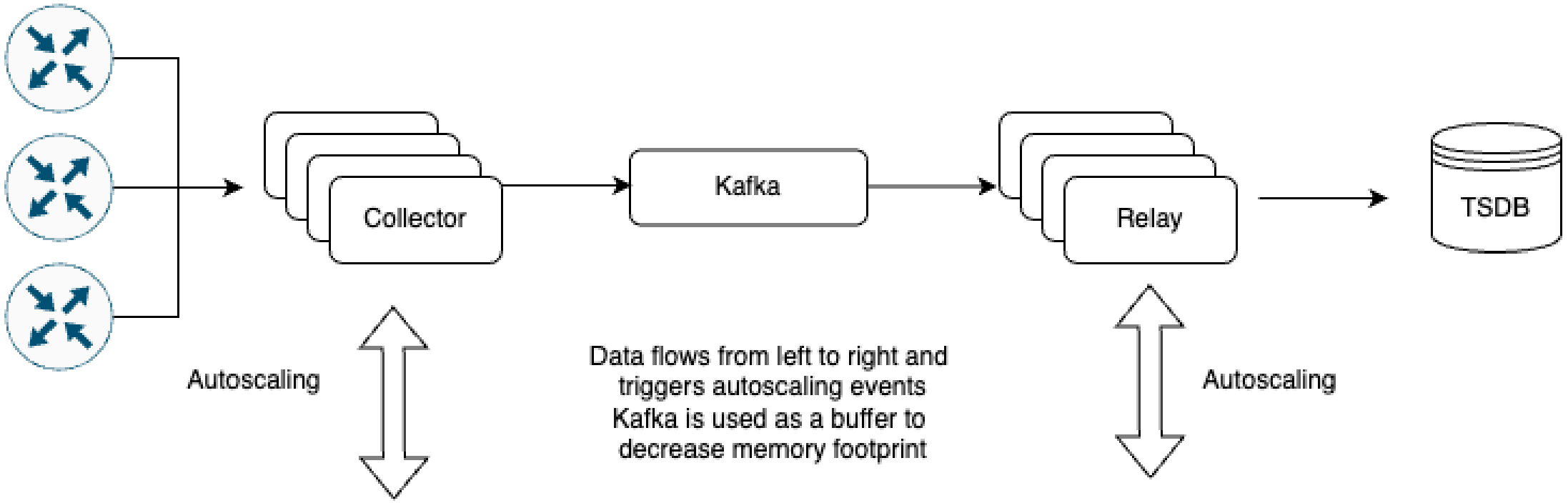
Gnmic-cluster-chart architecture

- Setup an event driven architecture
- Implement separation of concerns
  - Collector role
  - Relay role
  - Coordinators
  - Buffering
- Scaling the roles separately

Event driven architecture



Event driven architecture

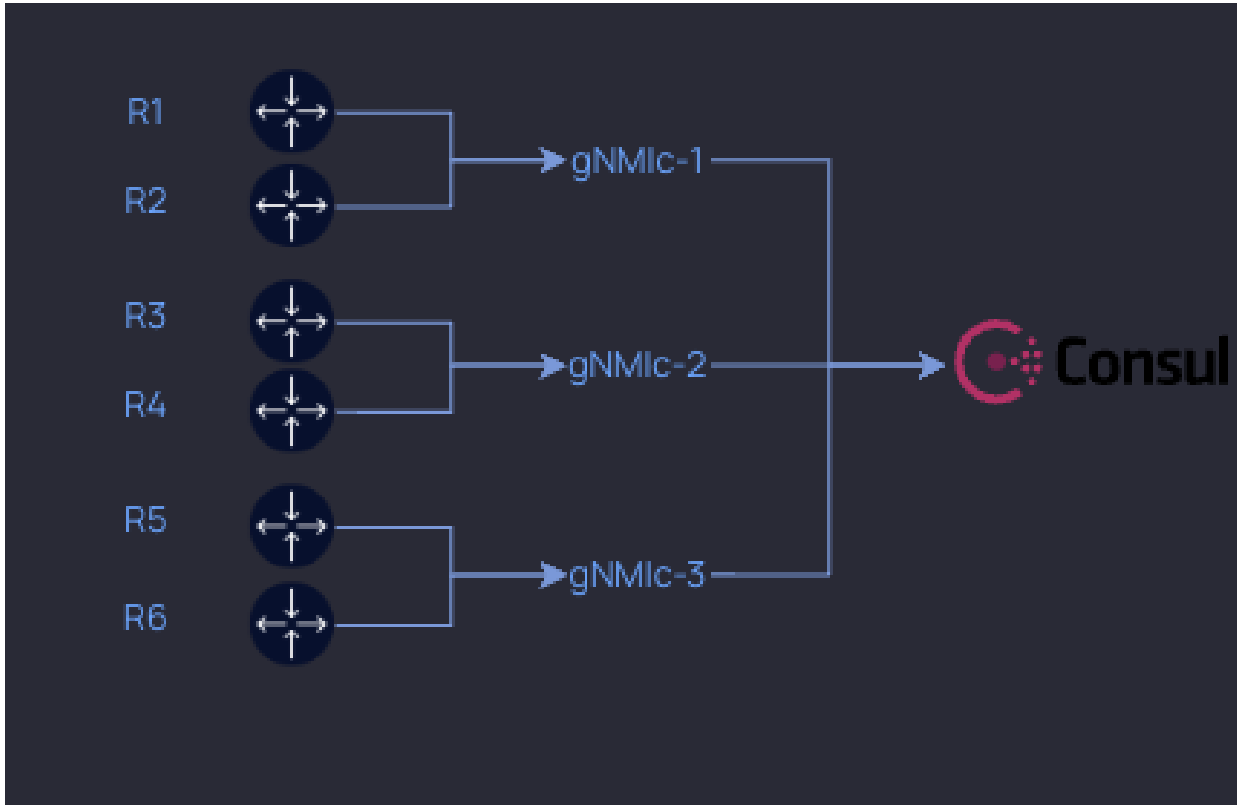


gNMIc clustering process

1. Service discovery
2. Leader Election
3. Target distribution
4. **On failure, repeat from step 1.**

https://gnmic.openconfig.net/user_guide/HA/

gNMIc clustering mode



KV Store		
Session	**Lock**	**TTL**
gNMIc-1	gnmic/default-cluster/targets/R1	10s
gNMIc-1	gnmic/default-cluster/targets/R2	10s
gNMIc-2	gnmic/default-cluster/targets/R3	10s
gNMIc-2	gnmic/default-cluster/targets/R4	10s
gNMIc-3	gnmic/default-cluster/targets/R5	10s
gNMIc-3	gnmic/default-cluster/targets/R6	10s

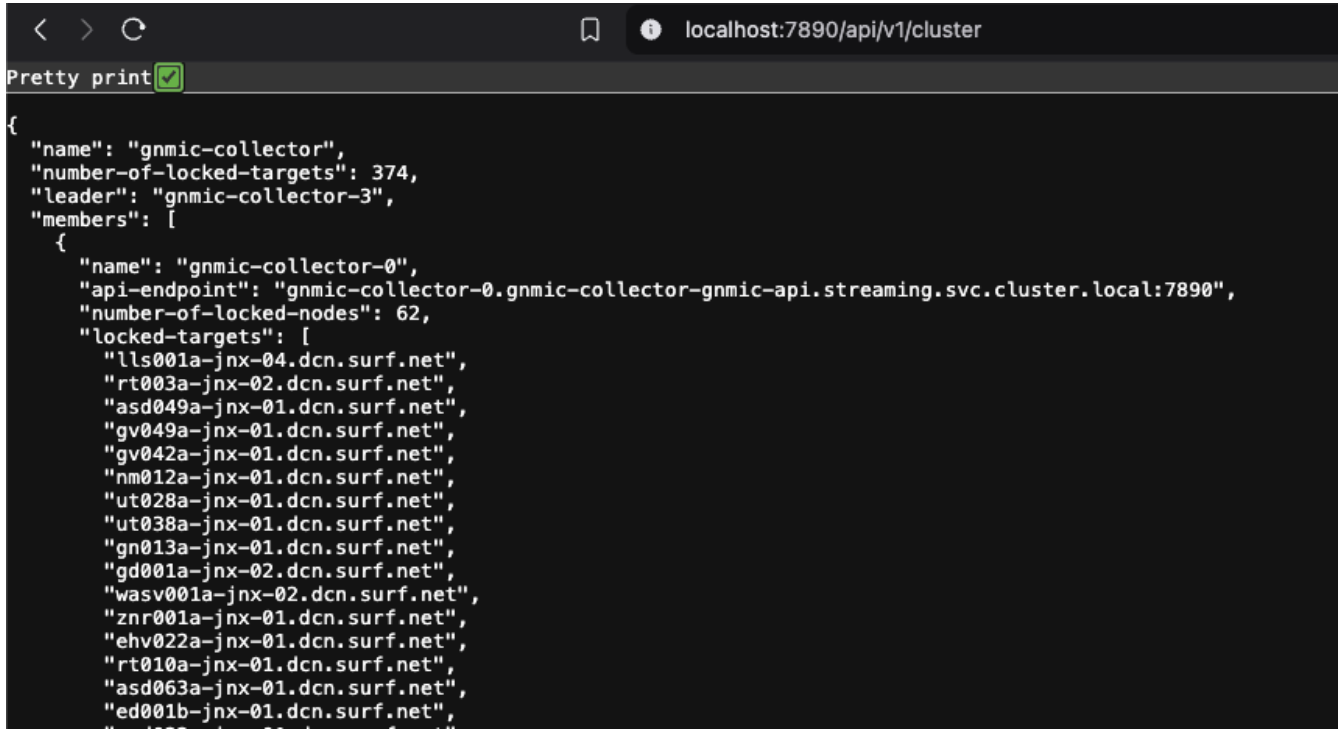
https://gnmic.openconfig.net/user_guide/HA/

Quorum state management

- Kubernetes leases
- Consul
- Separate state for the collector and relay processes
- Redis provides a caching layer and method to do inter process communication

What are other things gNMIc can do?

- Restful API
- Proxy service towards the Routers
  - Demo later
- CLI tool
- Data processing
  - Enrichment
  - Concatenation
  - Deletion
  - Conversion



Helm chart architecture and overview

The gnmic-cluster-chart

- Contains all manifests needed to setup gNMIc in cluster mode
- Pre-requisites
  - Strimzi operator installed
  - Kafka broker setup
  - Redis
  - Optional: Secret reflector
  - Sufficient CPU, Mem, Storage
- Resources for +/- 2 billion events per day

vCPU	Memory	Storage (90 days)
20	160Gi	3Ti

Chart components

- Statefulset configuration
- ConfigMaps
- HorizontalPodAutoscaling
- RBAC & ServiceAcounts
- Secrets and Secretproviders
- Ingress and services
- Optional: Kafka manifets
- Optional: Consul

Demo Time

Demo components

- Overview of the helm chart
- Deployment in Argo
- gNMIc
  - Cli
  - Openconfig models
  - Deployment
  - Change proces
  - gNMI Proxy
- Consul
- Kafka Monitoring
- Grafana Dashboard
- Orchestrator integrations

Wrap-up and Q&A

Future work

- Custom metrics for scaling
  - Acquired targets
- Pre-configured processors
- ML workloads that use the raw telemetry stream
- More integrations with WFO

Deliverables

- Helm chart: ✔
- Automatic target acquisition: WIP
- P.O.C: WFO integration example: ✔

Questions?