OST
Eastern Switzerland
University of Applied Sciences

Spicing Up Network Application Development

SwiNOG #38, Berne

Severin Dellsperger

21 June 2023

INS Institute for Network and Security, Rapperswil (CH)

Agenda

→ **Introduction**

- Jalapeno
- Jalapeno API GW
- Applications

OST

INS @ OST

INS | Institute for Network and Security

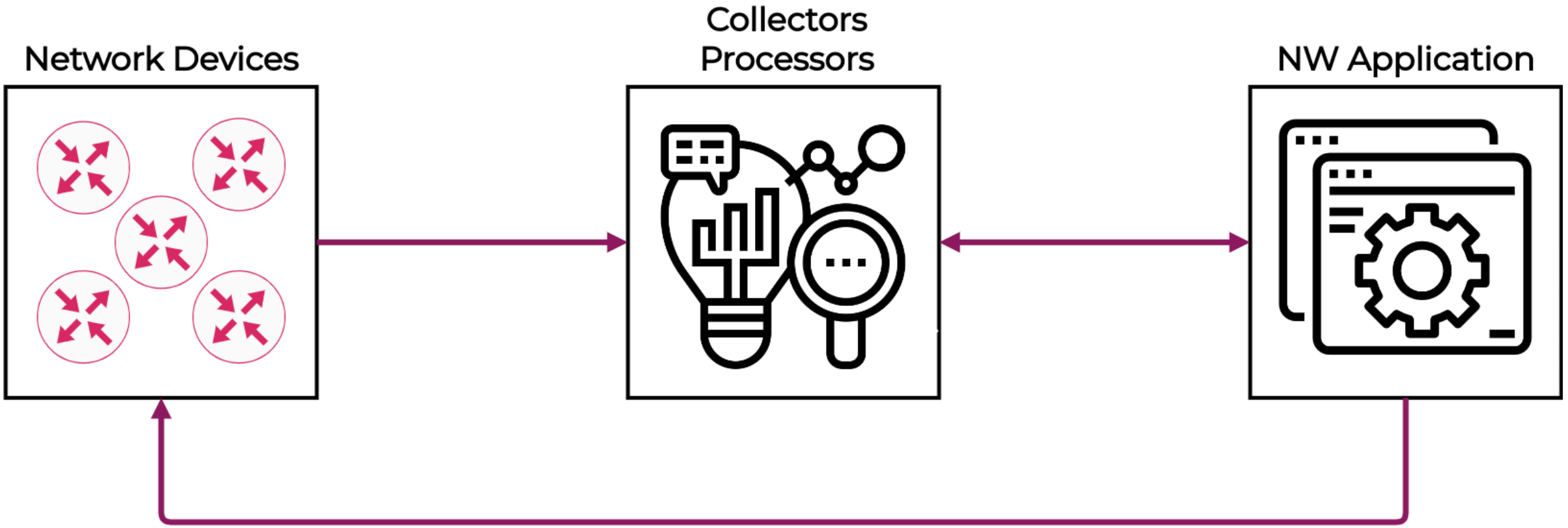
- OST - Eastern Switzerland University of Applied Science
  - Rapperswil · St.Gallen · Buchs SG
- INS Institute for Network and Security
  - Research latest technologies
  - Networking · Cloud · Security
  - Bridging the gap between industry and research

OST

About me

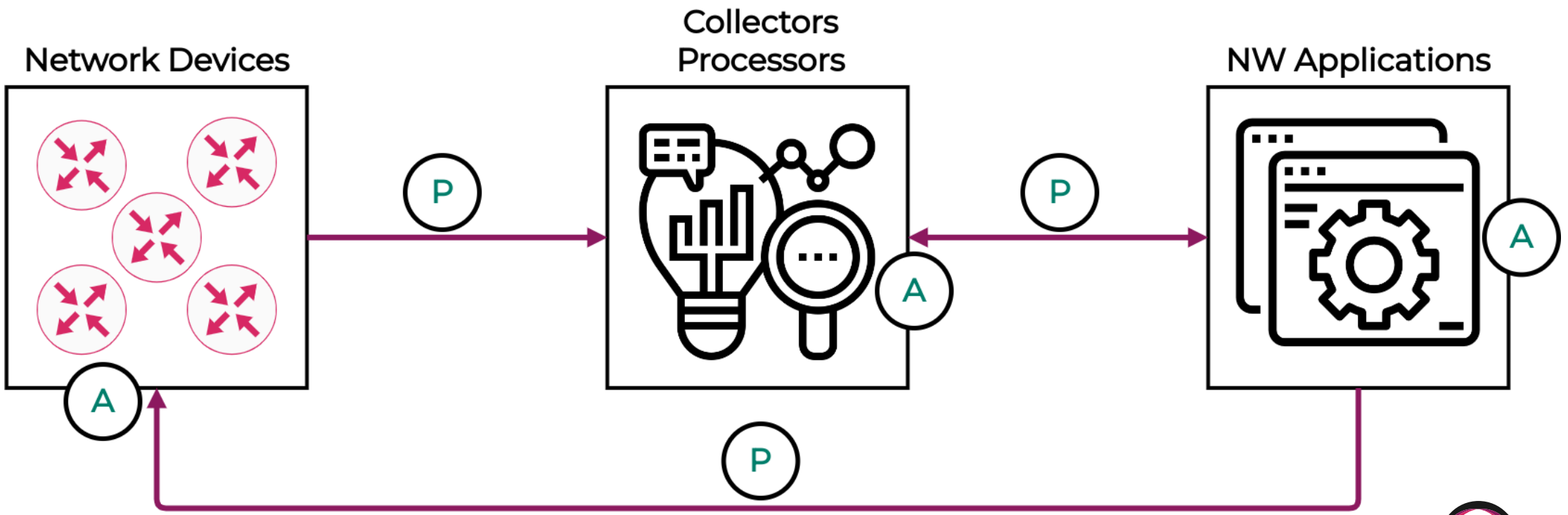
- Severin Dellsperger
- Network and Research Engineer
- Pursue MSc. in CS (SDN)
- Focus on modern network topics
  - Segment Routing
  - Network Applications
- IETF

Network Application Cycle



Requirements

Protocols	Collectors / Processors	APIs



APIs

Access processed NW data

Provide an easy-to-use interface for users

Program network devices

Straightforward

Consistent

Easy-to-use / -extend

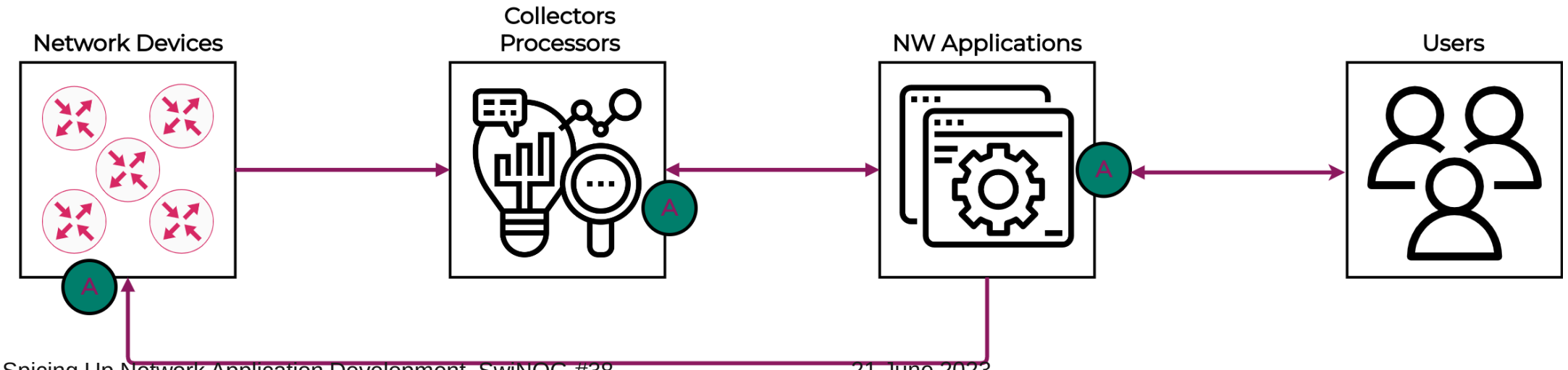
Reliable

Documented

REST

gRPC

GraphQL



Collectors / Processor

Collectors

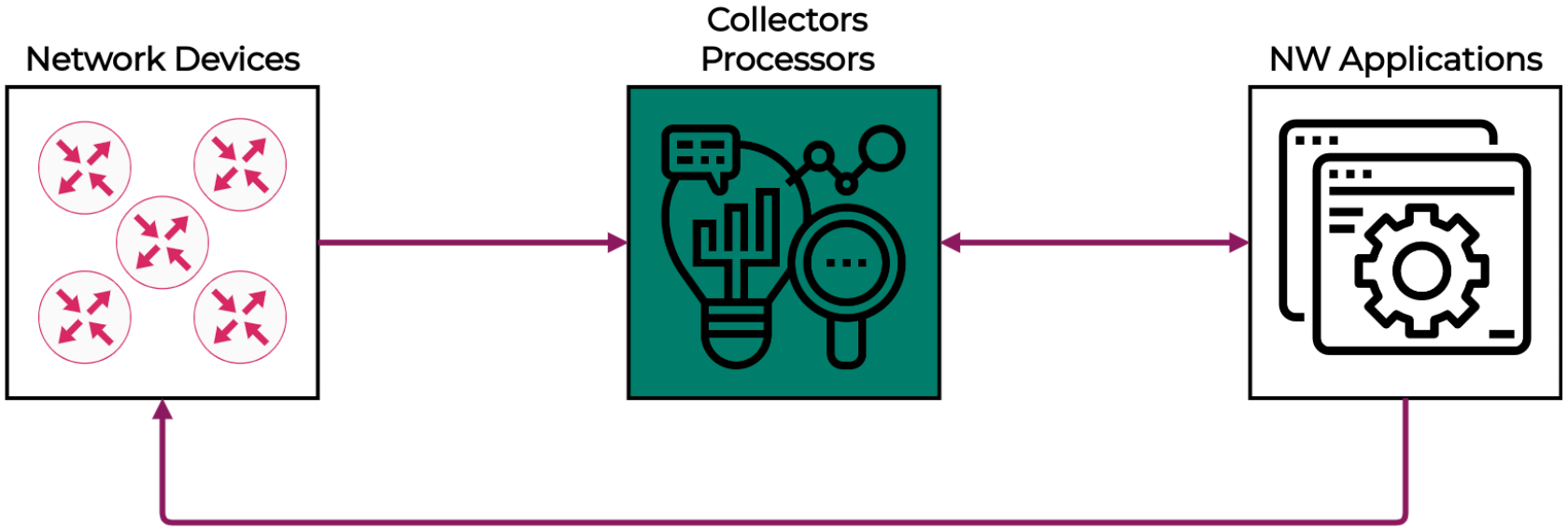
Receive network data from devices

Make data available for processors

Processors

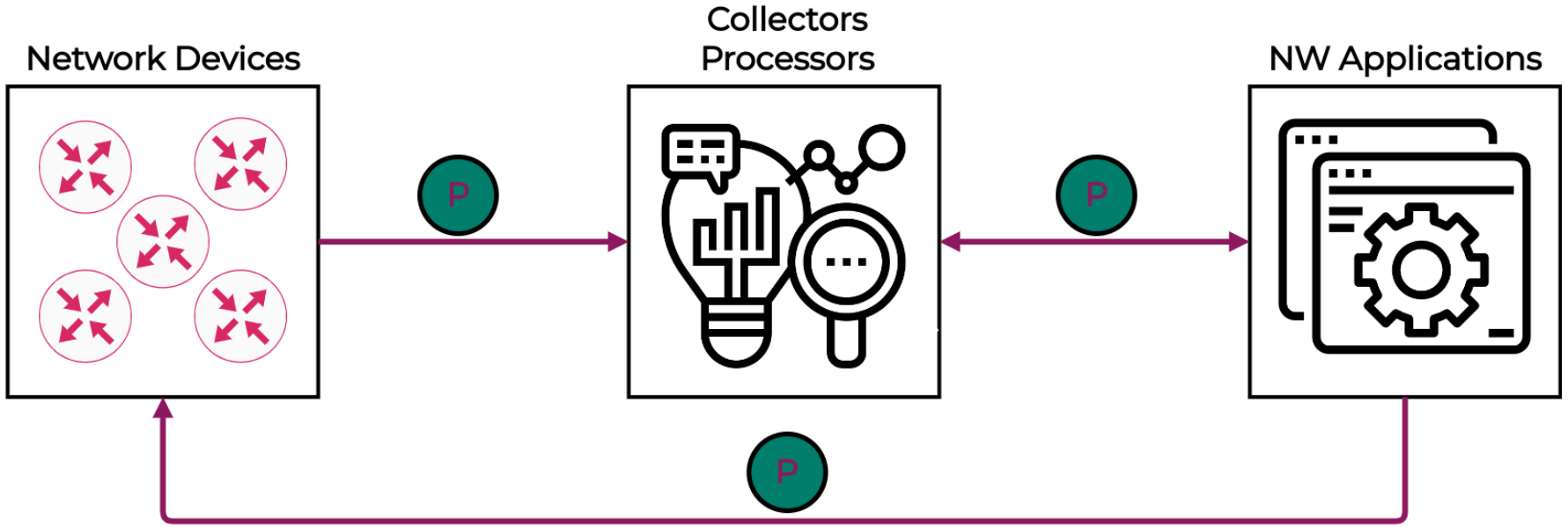
Extract information out of network data

Save the data to use easily



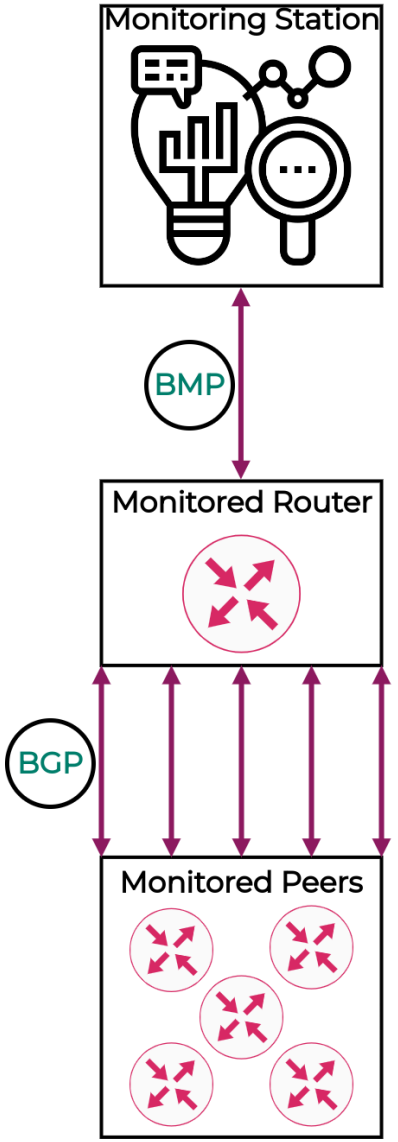
Protocols

NW devices → collector(s)	Applications → Processors	Applications → NW devices
BMP	HTTPS	NETCONF
YANG PUSH	(gRPC, REST, GraphQL)	RESTCONF
(gNMI, IPFIX)		(gNMI, SSH)



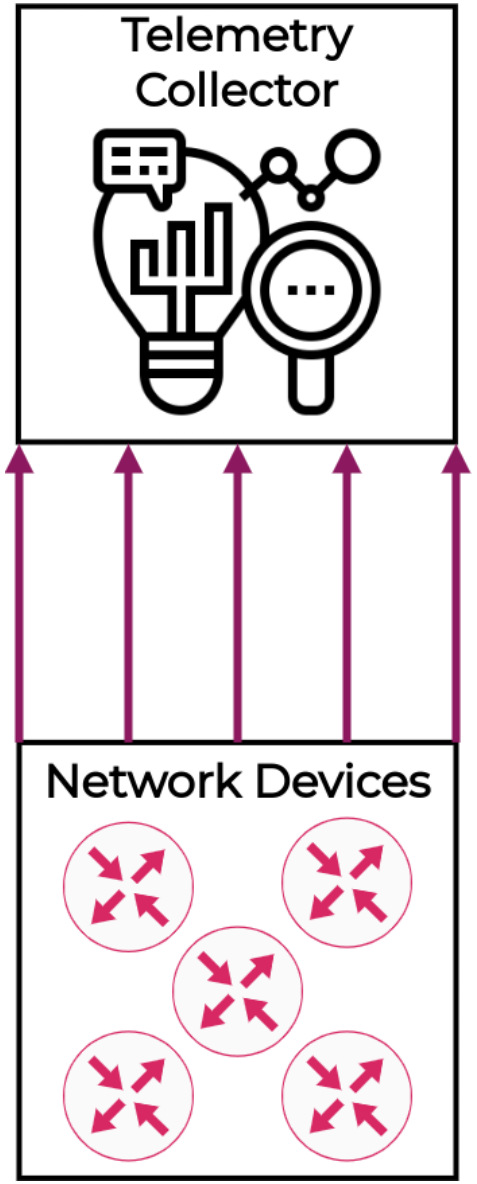
BMP – BGP Monitoring Protocol

- Standardized: RFC7854
- Monitor & analyze BGP
- Real-time updates, notifications and statistics
  - BGP sessions
  - Route information
  - Other relevant data



Network Telemetry

- Gain network insight
- Optimize network management
- Different approaches/technologies
  - Push/Streaming vs. Pull
  - YANG Push
    - Event Stream vs. Datastore Subscription
    - Periodic vs On-Change Subscription
  - gNMI, IPFIX, NETFLOW, etc.
- Important topic IETF OPSAWG



Yes, we do!


ONE DOES NOT SIMPLY
WRITE A NETWORK APPLICATION
imgflip.com

Agenda

- Introduction
- → **Jalapeno**
- Jalapeno API GW
- Applications

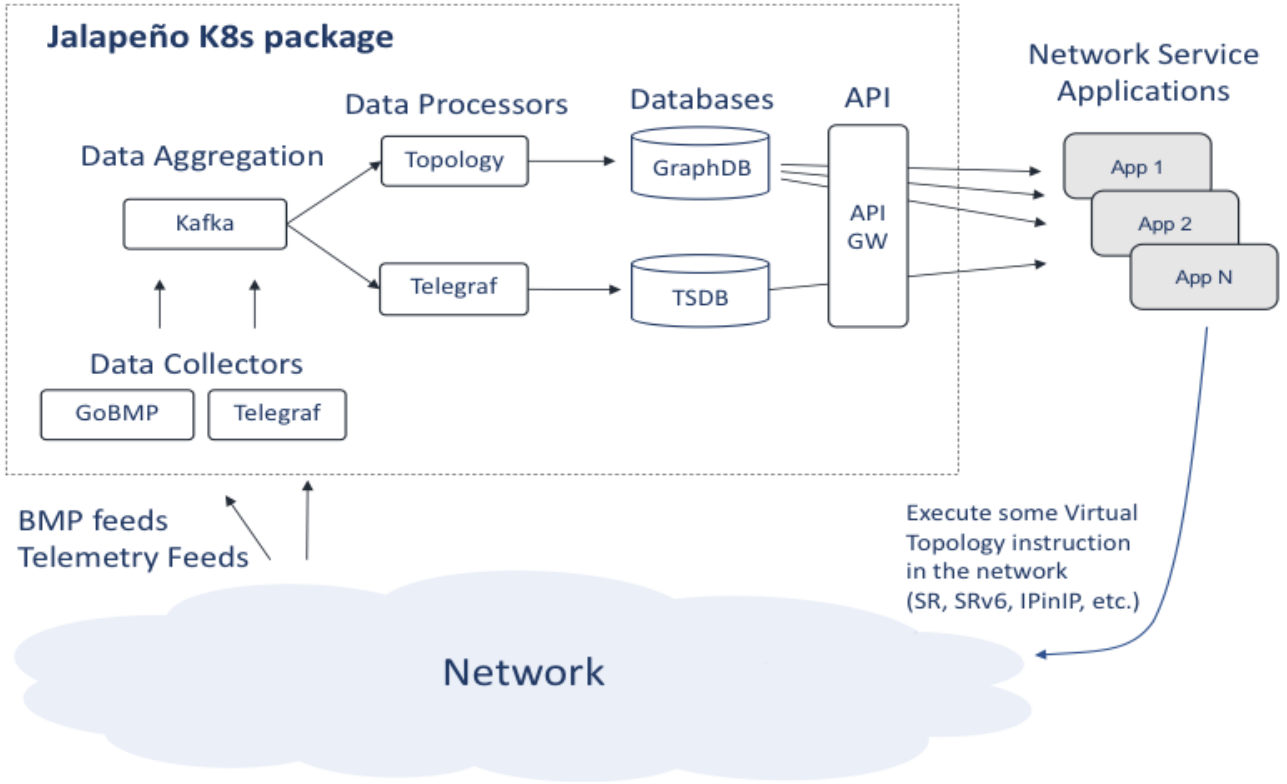
OST

Overview

A cloud-native infrastructure platform to enable development of network services

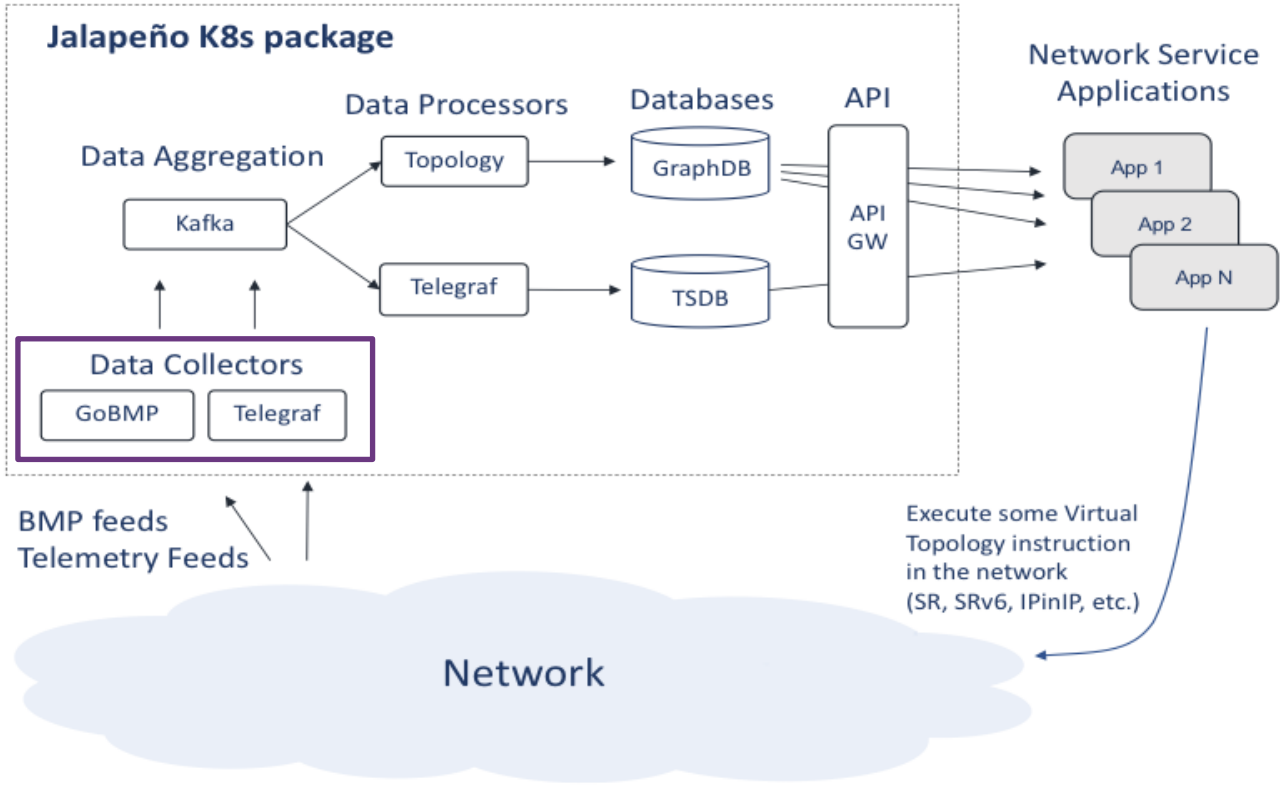
- Originally developed by Cisco
- Open source
  - [https://github.com/cisco-open/jalapeno]
- Micro-service architecture
- Build network applications
  - quickly
  - easily
- Many possible applications
  - Traffic Engineering
  - VPN Overlays
  - Service Chaining
  - …

High-level Architecture



Data Collectors

Collect the data messages from the network devices and forward them to the data aggregation



GoBMP

goBMP is an implementation of Open BMP (RFC 7854) protocol's collector in go language.

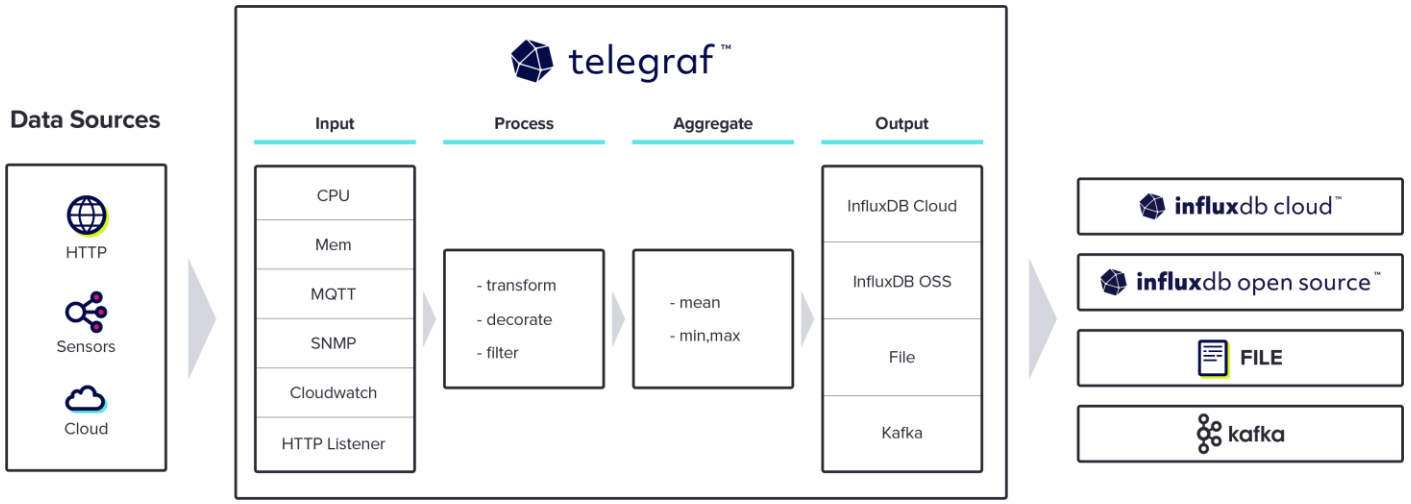
- Open source
- Collect BGP data
- Parse of many BMP message types
  - *peer, unicast_prefix, ls_node,* etc.
- Publish to Kafka topic, file or stdout


```
gobmp: 18:37:06.573667 {MsgType:74,
MsgHash:190dafab69706a67221c1226360de7dc,
{
   "action":"add",
   "router_hash":"190dafab69706a67221c1226360de7dc",
   "router_ip":"10.0.0.1",
   "base_attrs":{
      "base_attr_hash":"9afa75f04728a2262a6ad9d9628bc847",
      "origin":"igp",
      "local_pref":100,
      "is_atomic_agg":false
   },
   "peer_hash":"3e29df1fd1efc90e53184caf2d456bd9",
   "peer_ip":"10.0.0.2",
   "peer_type":0,
   "peer_asn":65000,
   "timestamp":"2023-06-06T18:36:57.0006576Z",
   "prefix":"172.16.2.0",
   "prefix_len":24,
   "is_ipv4":true,
   "nexthop":"10.0.0.2",
   "is_nexthop_ipv4":true,
   "is_adj_rib_in_post_policy":false,
   "is_adj_rib_out_post_policy":false,
   "is_loc_rib_filtered":false
}}
```

Telegraf

Telegraf is an agent for collecting, processing, aggregating, and writing metrics.

- Open source
- Originally developed by InfluxData
- Plugin-based
  - 300+ plugins
  - Input
  - Processor
  - Aggregator
  - Output



Telegraf Ingress

Jalapeno uses Cisco MDT plugin:

1. Collect Model-driven Telemetry messages
   - via gRPC
2. Process data
   - Find embedded tags
   - Use aliases
3. Send messages to Kafka

```
[agent]
  interval = "10s"
  round_interval = true
  metric_buffer_limit = 10000
  flush_buffer_when_full = true
  collection_jitter = "0s"
  flush_interval = "10s"
  flush_jitter = "0s"
  debug = false
  quiet = false
  hostname = "telegraf"


[[inputs.cisco_telemetry_mdt]]
  transport = "grpc"
  service_address = ":57400"
  embedded_tags = ["Cisco-IOS-XR-fib-common-oper:mpls-forwarding/nodes/node/label-fib/forwarding-details/forwarding-
detail/label-information/outgoing-interface"]
  [inputs.cisco_telemetry_mdt.aliases]
    ifstats = "ietf-interfaces:interfaces-state/interface/statistics"


[[outputs.kafka]]
  brokers = ["broker.jalapeno.svc:9092"]
  topic = "jalapeno.telemetry"
```

Source: https://github.com/cisco-open/jalapeno/blob/main/install/collectors/telegraf-ingress/telegraf_ingress_cfg.yaml

Data Aggregation

Apache Kafka is an open-source distributed event streaming platform […] for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications.

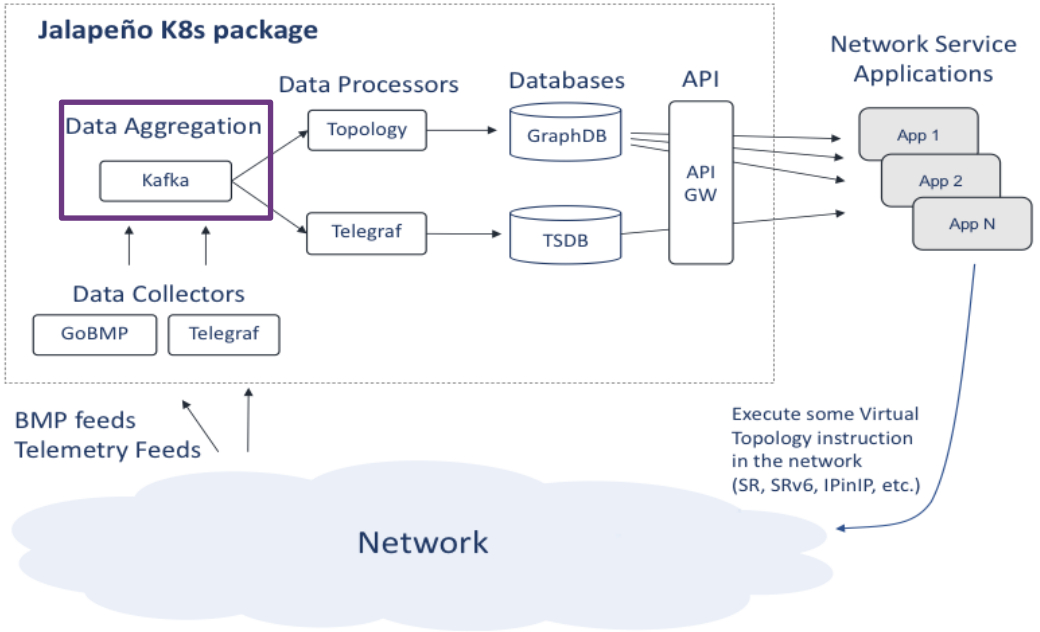
- High-performance messaging system
- Share messages between different applications easily

  App X: Publish messages into topic

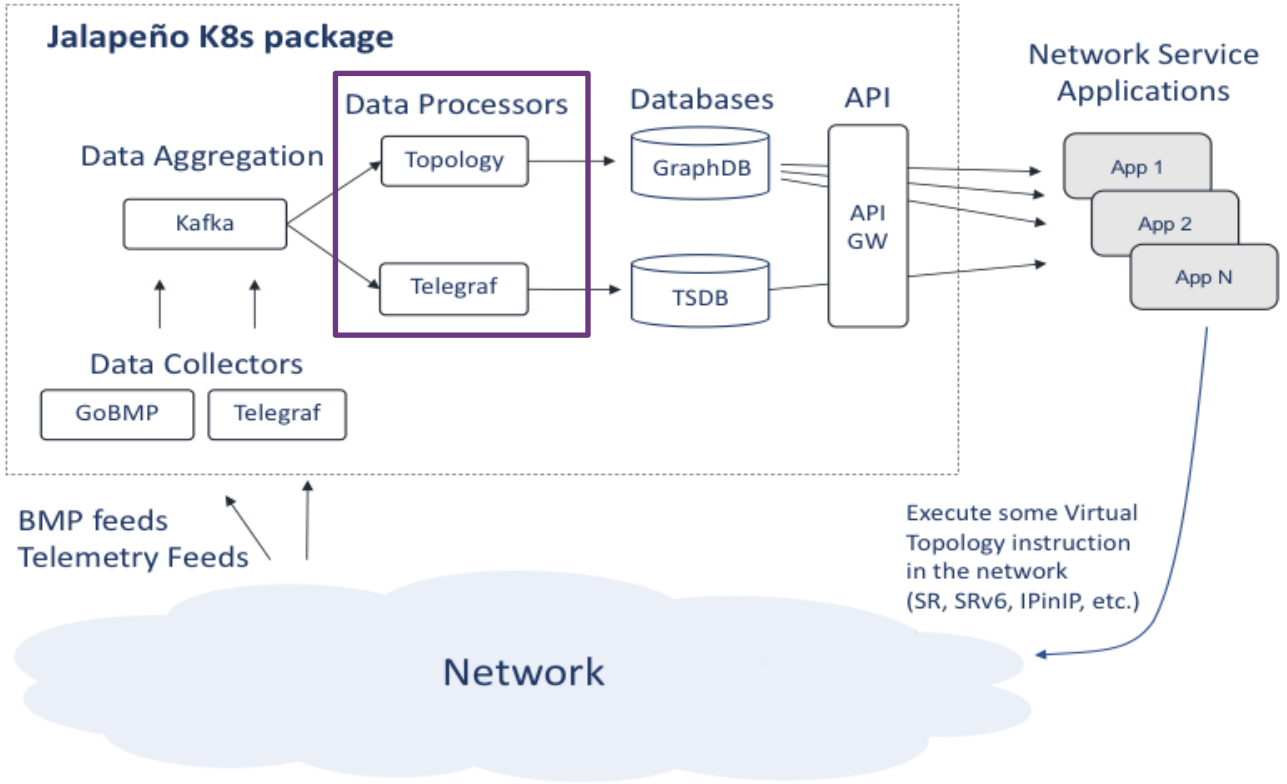
  Kafka: Store messages in fault-tolerant way

  App Y: Subscribe to topic

  App Y: Receive messages when ready

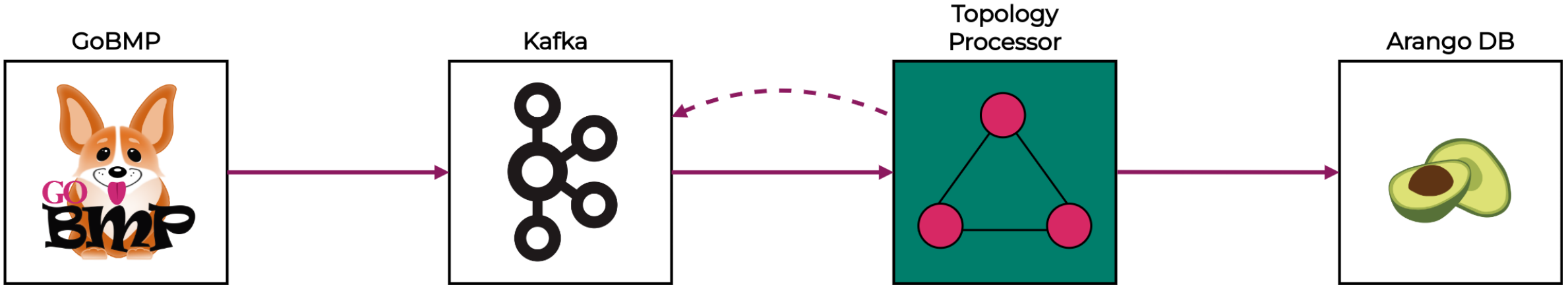


Data Processors



Topology Processor

- Subscribe to Kafka BMP topic
- Receive messages from goBMP via Kafka
- Create ArangoDB Collection based on goBMP messages
  - E.g. *ls_node / ls_link* collection from BGP-LS message data
- Write events about processed entities back to Kafka



Telegraf Egress

Jalapeno use Telegraf to process MDT info

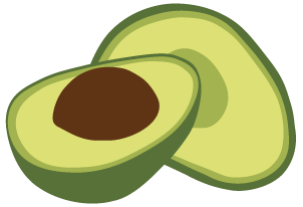
1. Get telemetry messages from Kafka
2. Send messages to outputs
   1. InfluxDB
   2. File

```
[[inputs.kafka_consumer]]
  brokers = ["broker.jalapeno.svc:9092"]
  topics = ["jalapeno.telemetry"]
  max_message_len = 1000000
  data_format = "influx"

[[outputs.influxdb]]
  urls = ["http://influxdb-np.jalapeno:8086"]
  database = "mdt_db"
  username = "jalapeno"
  password = "jalapeno"
  precision = "s"
  timeout = "5s"

[[outputs.file]]
  files = ["metrics.out"]
  rotation_interval = "1h"
  rotation_max_size = "20MB"
  rotation_max_archives = 3
  data_format = "json"
```

Source: https://github.com/cisco-open/jalapeno/blob/main/install/processors/telegraf-egress/telegraf_egress_cfg.yaml

Databases

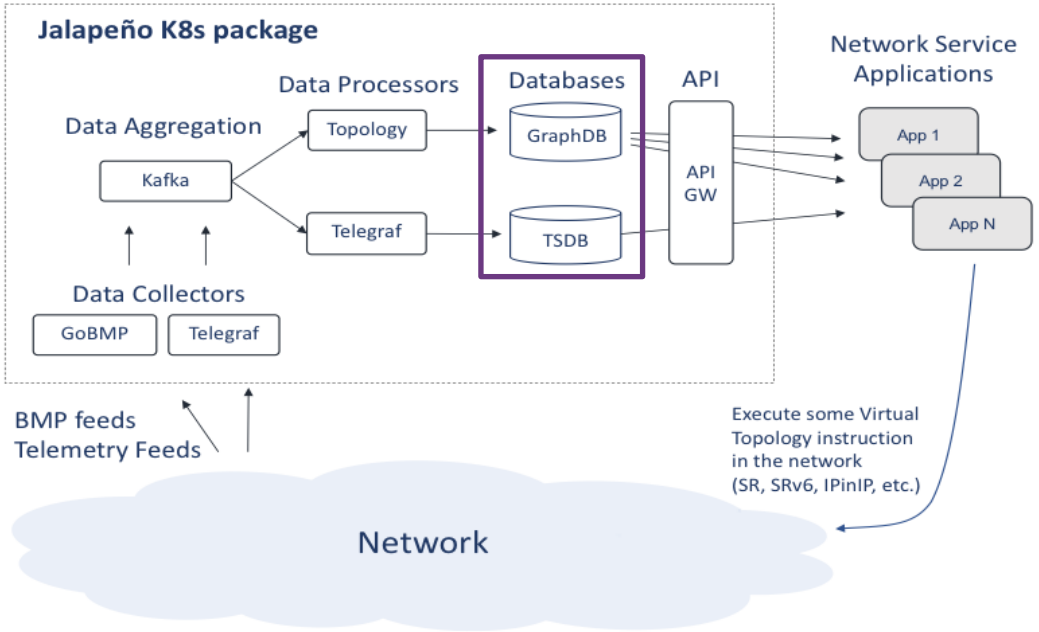
ArangoDB



- Graph database
- Contains BMP data
- “Topology view”

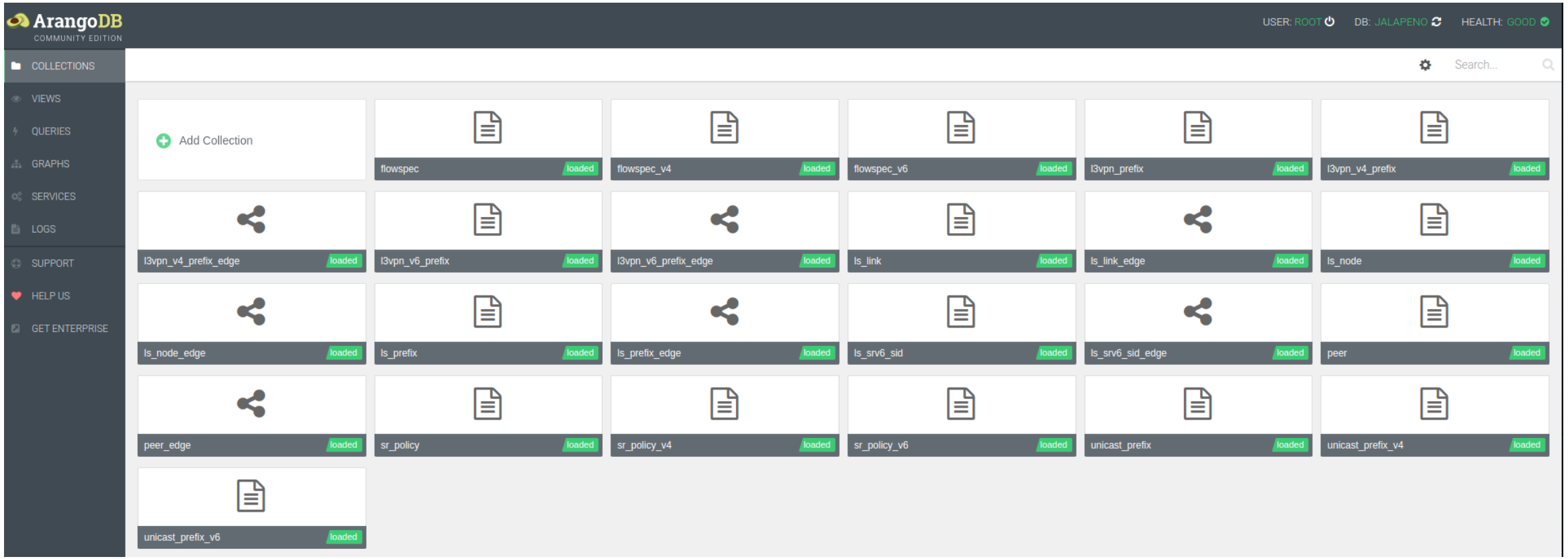
InfluxDB

- Timeseries database
- Contains Telemetry data
- “Assign network data a time tag”



Source: https://github.com/cisco-open/jalapeno

ArangoDB Collections



ArangoDB Node Collection

ArangoDB COMMUNITY EDITION Collection: ls_node

USER: ROOT DB: JALAPENO HEALTH: GOOD

COLLECTIONS
VIEWS
QUERIES
GRAPHS
SERVICES
LOGS
SUPPORT
HELP US
GET ENTERPRISE

3.6.12

Content Indexes Info Settings

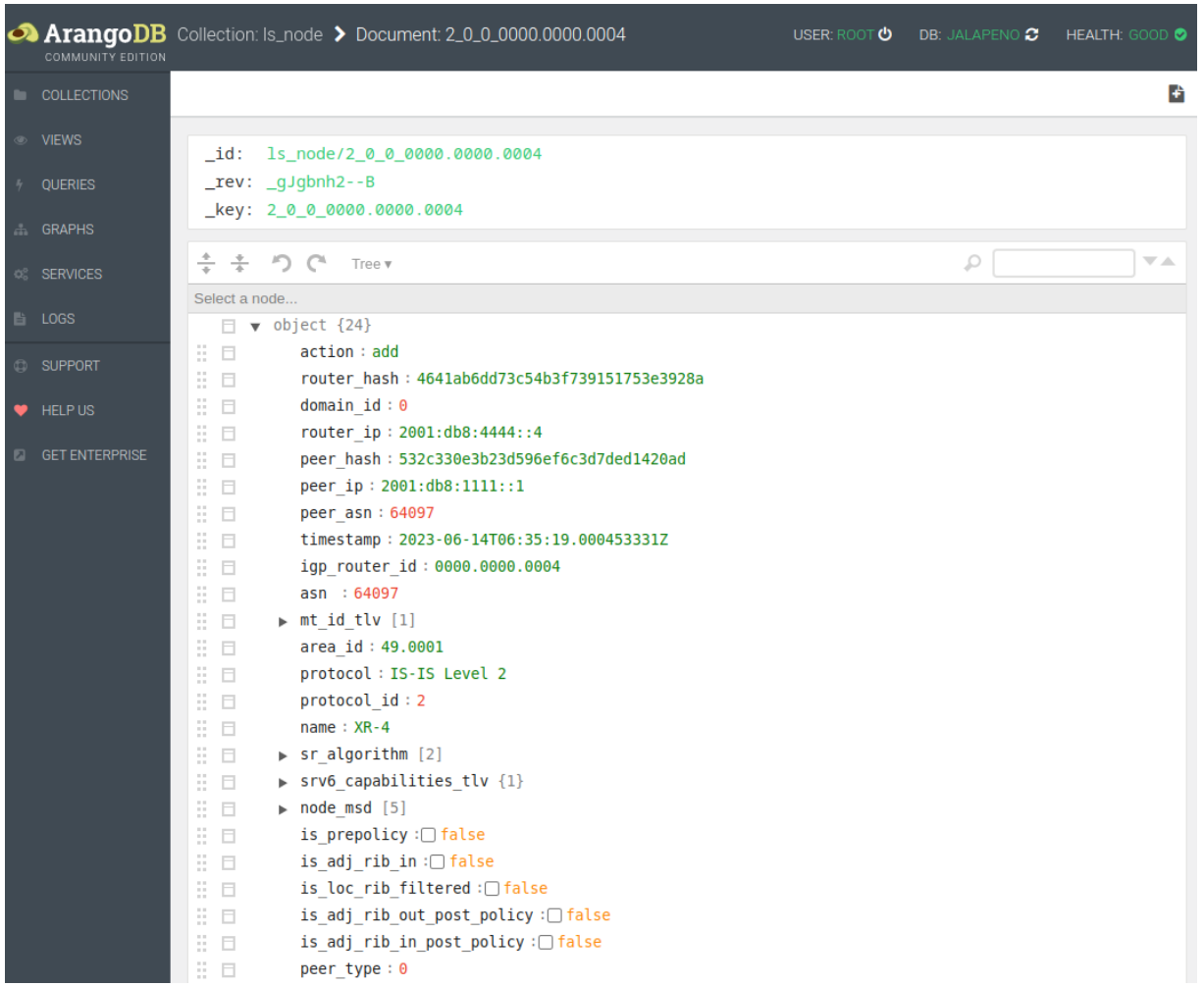
10 results

Content	_key
{"_id":"ls_node/2_0_0_0000.0000.0004","_key":"2_0_0_0000.0000.0004","_rev":"_gJgbnh2--B","action":"add","area_id":"49.0001","asn":64097,"domain_id":0,"igp_router_…	2_0_0_0000.0000.0004
{"_id":"ls_node/2_0_0_0000.0000.0002","_key":"2_0_0_0000.0000.0002","_rev":"_gJgbni---_","action":"add","area_id":"49.0001","asn":64097,"domain_id":0,"igp_router_…	2_0_0_0000.0000.0002
{"_id":"ls_node/2_0_0_0000.0000.0006","_key":"2_0_0_0000.0000.0006","_rev":"_gJgbni---C","action":"add","area_id":"49.0001","asn":64097,"domain_id":0,"igp_router_…	2_0_0_0000.0000.0006
{"_id":"ls_node/2_0_0_0000.0000.0007","_key":"2_0_0_0000.0000.0007","_rev":"_gJgbniW--D","action":"add","area_id":"49.0001","asn":64097,"domain_id":0,"igp_router_…	2_0_0_0000.0000.0007
{"_id":"ls_node/2_0_0_0000.0000.0005","_key":"2_0_0_0000.0000.0005","_rev":"_gJgbni2--I","action":"add","area_id":"49.0001","asn":64097,"domain_id":0,"igp_router_…	2_0_0_0000.0000.0005
{"_id":"ls_node/2_0_0_0000.0000.0008","_key":"2_0_0_0000.0000.0008","_rev":"_gJgbni6--D","action":"add","area_id":"49.0001","asn":64097,"domain_id":0,"igp_router_…	2_0_0_0000.0000.0008
{"_id":"ls_node/2_0_0_0000.0000.0003","_key":"2_0_0_0000.0000.0003","_rev":"_gJgbnji--D","action":"add","area_id":"49.0001","asn":64097,"domain_id":0,"igp_router_…	2_0_0_0000.0000.0003
{"_id":"ls_node/2_0_0_0000.0000.0001","_key":"2_0_0_0000.0000.0001","_rev":"_gJgbnkW--_","action":"add","area_id":"49.0001","asn":64097,"domain_id":0,"igp_router_…	2_0_0_0000.0000.0001

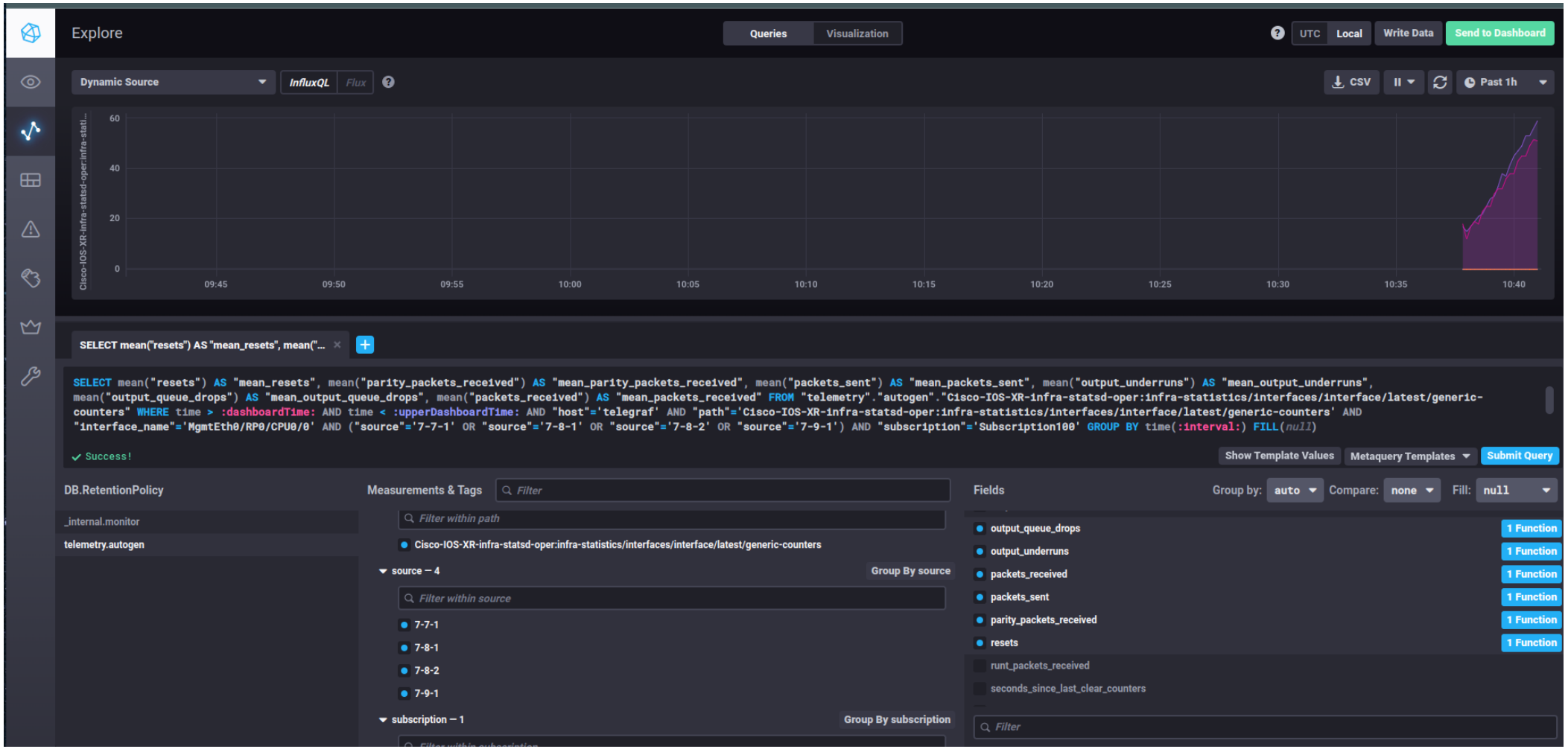
8 doc(s)

1

ArangoDB Node Document



InfluxDB Interface Counters

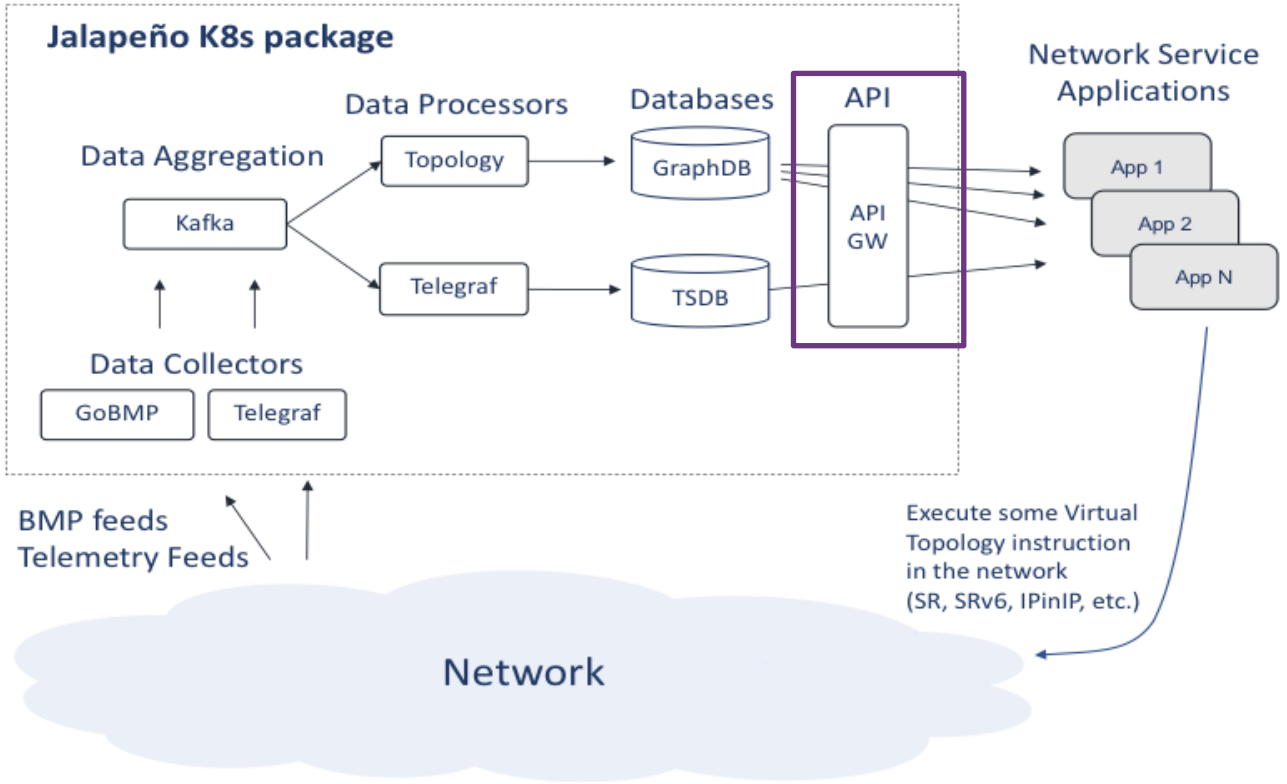


Agenda

- Introduction
- Jalapeno
- → **Jalapeno API GW**
- Applications

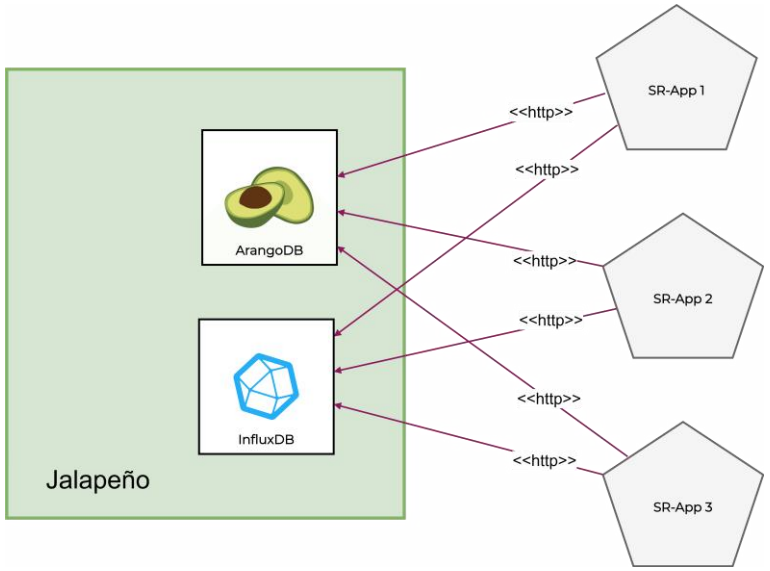
OST

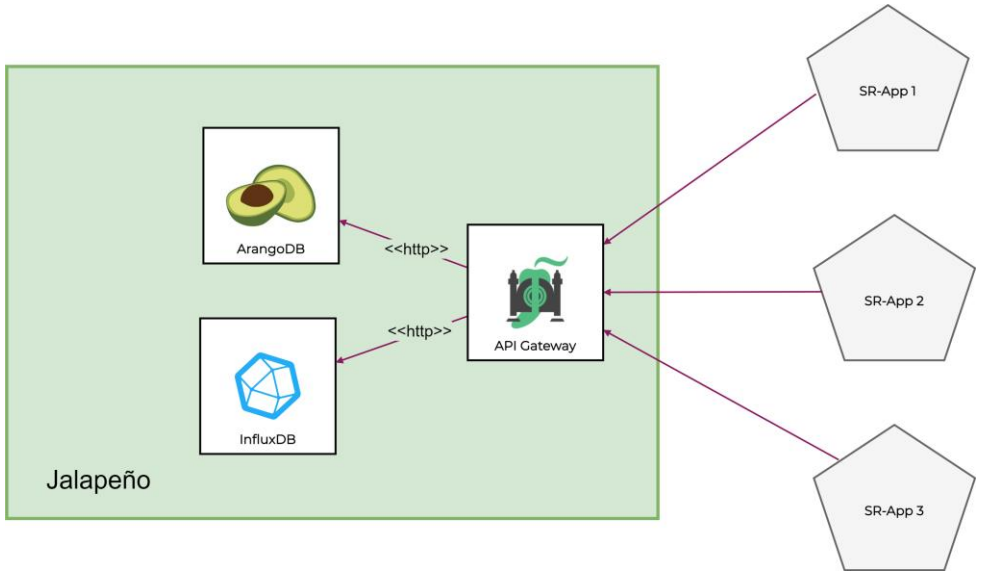
Overview


Jalapeño K8s package
Data Processors
Databases
API
Network Service Applications
Data Aggregation
Topology
GraphDB
Kafka
API GW
App 1
App 2
App N
Telegraf
TSDB
Data Collectors
GoBMP
Telegraf
BMP feeds
Telemetry Feeds
Execute some Virtual Topology instruction in the network (SR, SRv6, IPinIP, etc.)
Network

Why an API Gateway?

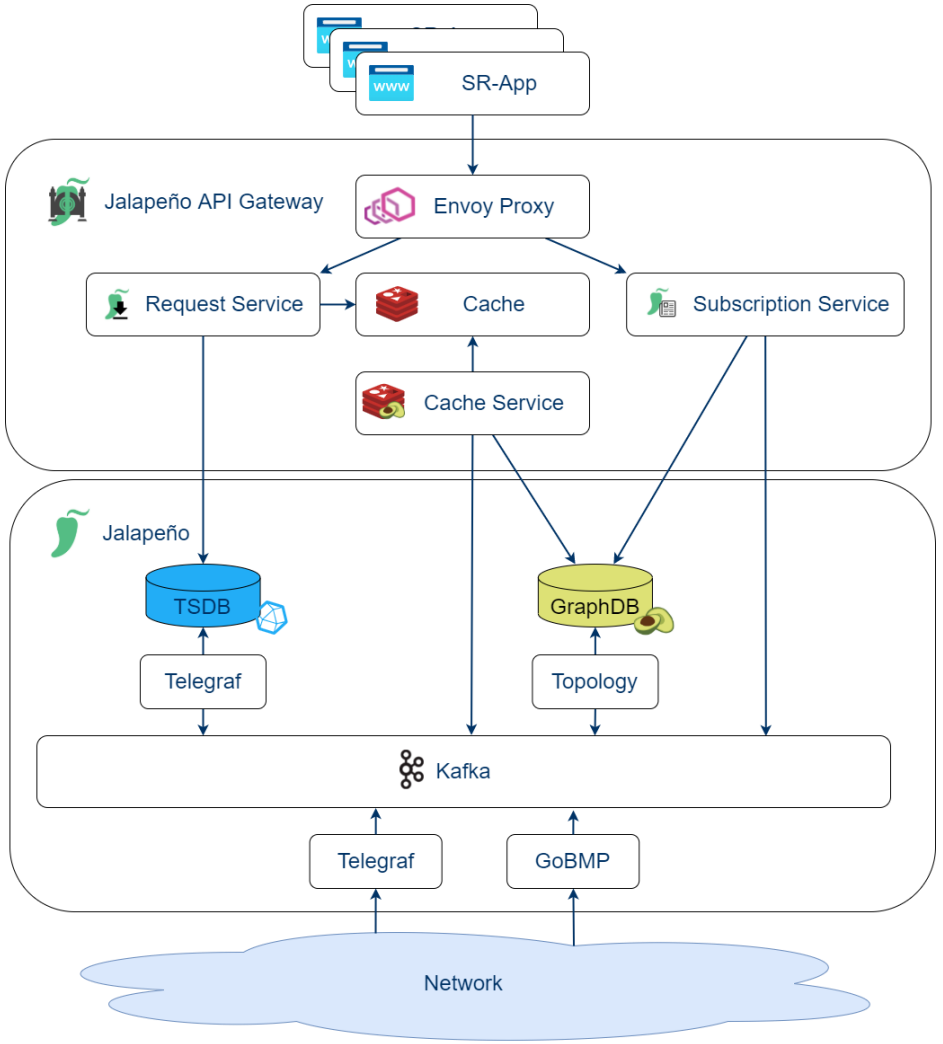
- Single point for access
- Simplifies data retrieval
  - Request data easily
  - Subscribe on live updates
- Decouples apps from internal Jalapeno
- Caching (for frequently requested data)
- Capabilities to introduce new features easily
  - Load Balancing
  - Security
  - …





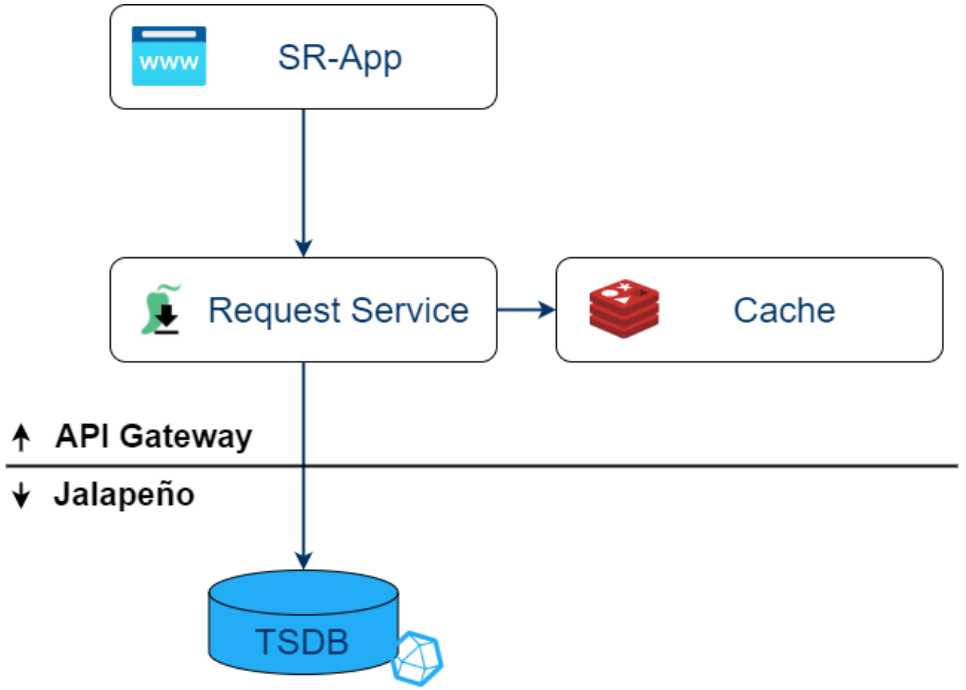
An Overview

- Open source:
  - [https://github.com/jalapeno-api-gateway]
- Well-documented 🙌
  - [https://jalapeno-api-gateway.github.io/jagw/docs/]
- Developed by the INS
- Cloud-native
- Easy-to-use via gRPC
- Well-defined contracts via protobuf
- Easy-to-maintain via Helm chart



Request Service

- Allow app to request data via gRPC
- Request service gets
  - Topology data from cache
    - Node, Link, Prefixes, SRv6 SIDs, etc.
  - Telemetry data from InfluxDB
    - YANG models
- Return data to caller




```
service RequestService {
    rpc GetLsNodes(TopologyRequest) returns (LsNodeResponse) {}
    rpc GetLsLinks(TopologyRequest) returns (LsLinkResponse) {}
    rpc GetLsPrefixes(TopologyRequest) returns (LsPrefixResponse) {}
    rpc GetLsSrv6Sids(TopologyRequest) returns (LsSrv6SidResponse) {}
    rpc GetLsNodeEdges(TopologyRequest) returns (LsNodeEdgeResponse) {}
    rpc GetTelemetryData(TelemetryRequest) returns (TelemetryResponse) {}
}
```

Example

```
rpc GetLsNodes(TopologyRequest) returns (LsNodeResponse) {}
```

```
message TopologyRequest {
    repeated string keys = 1;
    repeated string properties = 2;
}
```

```
message LsNodeResponse {
    repeated LsNode ls_nodes = 1;
}
```

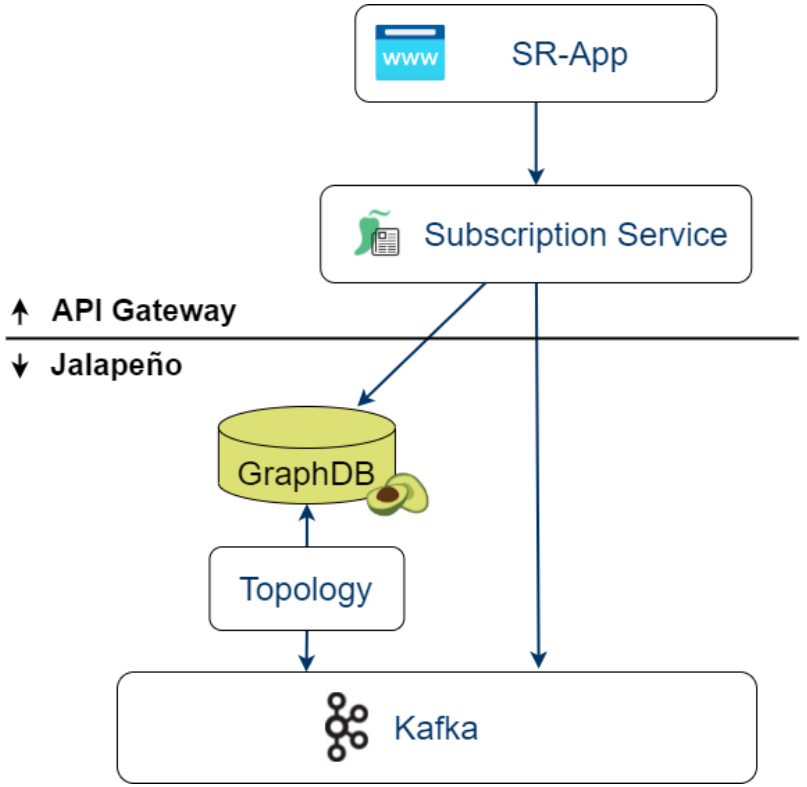
- Omitting **keys** returns all available LsNodes.
- Omitting **properties** returns LsNodes with all available properties.

```
// Request specific properties of specific LsNodes.
TopologyRequest {
    "keys": [
        "2_0_0_0000.0000.0001",
        "2_0_0_0000.0000.0002"
    ],
    "properties": [
        "Key",
        "Name",
        "Asn"
    ]
}
```

```
LsNodeResponse {
    "LsNodes": [
        {
            "Key": "2_0_0_0000.0000.0001",
            "Name": "XR-1",
            "Asn": 65001
        },
        {
            "Key": "2_0_0_0000.0000.0002",
            "Name": "XR-2",
            "Asn": 65001
        }
    ]
}
```

Subscription Service

- Allow app to subscribe for updates via gRPC
- Subscription Service constantly observes Kafka topics
  - Watch events for topology changes and get update from GraphDB
  - Watch telemetry topic and extract data directly
- Return updates to subscribers




```
service SubscriptionService {
    rpc SubscribeToLsNodes(TopologySubscription) returns (stream LsNodeEvent) {}
    rpc SubscribeToLsLinks(TopologySubscription) returns (stream LsLinkEvent) {}
    rpc SubscribeToLsPrefixes(TopologySubscription) returns (stream LsPrefixEvent) {}
    rpc SubscribeToLsSrv6Sids(TopologySubscription) returns (stream LsSrv6SidEvent) {}
    rpc SubscribeToLsNodeEdges(TopologySubscription) returns (stream LsNodeEdgeEvent) {}
    rpc SubscribeToTelemetryData(TelemetrySubscription) returns (stream TelemetryEvent) {}
}
```

Example

```
rpc SubscribeToLsNodes(TopologySubscription) returns (stream LsNodeEvent){}
```

```
message TopologySubscription {
    repeated string keys = 1;
    repeated string properties = 2;
}
```

```
message LsNodeEvent {
    required string action = 1;
    required string key = 2;
    optional LsNode ls_node = 3;
}
```

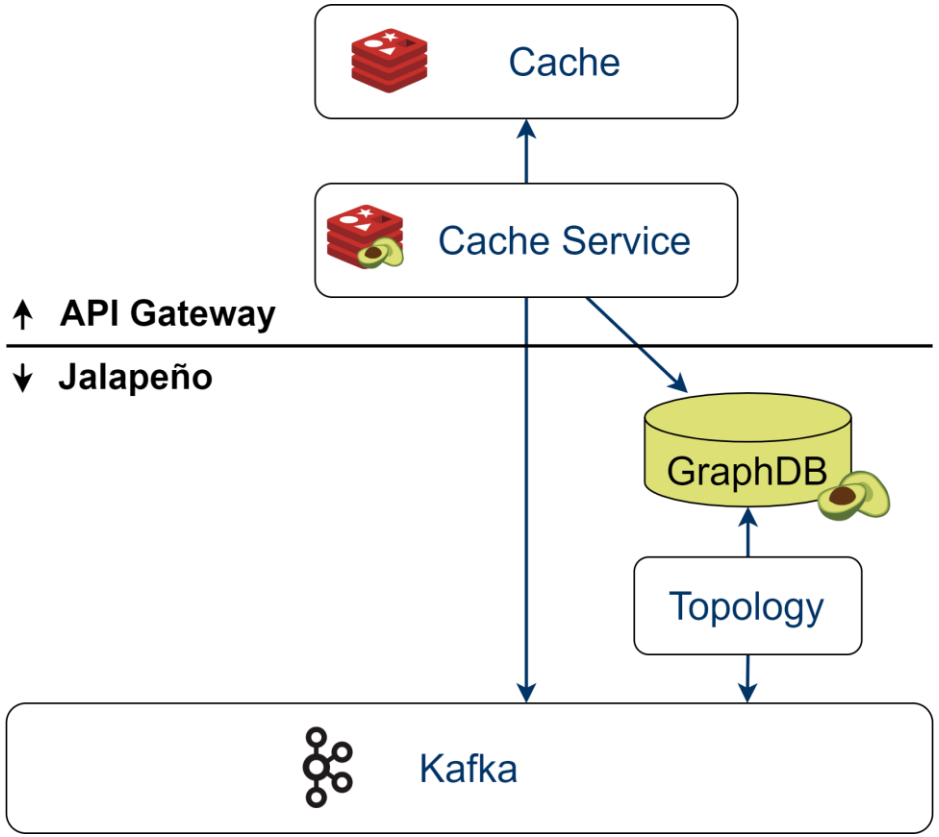
- Omitting **keys** streams LsNodeEvents for all available LsNodes.
- Omitting **properties** streams LsNodeEvents for LsNodes with all available properties.

```
// Subscribe to all available LsNodes and return all their properties.
TopologySubscription {}
```

```
// Continuously streams LsNodeEvents
// for all available nodes

LsNodeEvent {
    "Action": "update",
    "Key": "2_0_0_0000.0000.0007",
    "LsNode": {
        "Key": "2_0_0_0000.0000.0007",
        "Id": 773,
        "RouterHash": "7eb583cb3c17c496cfa9370d9bc2a3eb",
        ...
    }
}
```

Cache Service

- Mirror GraphDB
  - Reduce the load on the GraphDB
  - Reduce response time to API Request
- Telemetry data is not cached
  - Updates are too frequently to justify caching
  - Amount of telemetry data makes cache unfeasible
- During startup:
  1. Subscribe first to all Kafka event/update topics
  2. Fetch current state of all existing documents and save them into the cache
  3. Process update messages



Agenda

- Introduction
- Jalapeno
- Jalapeno API GW
- → **Applications**

OST

Request all nodes in Go

```
requestEndpoint := jagw.JagwEndpoint{
        EndpointAddress: '172.16.19.66',
        EndpointPort:    9903,
}
//Dial in
requestConnection, err := jagw.NewJagwConnection(requestEndpoint)
if err != nil {
        panic(err)
}
defer requestConnection.Close()
client := jagw.NewRequestServiceClient(requestConnection)
```

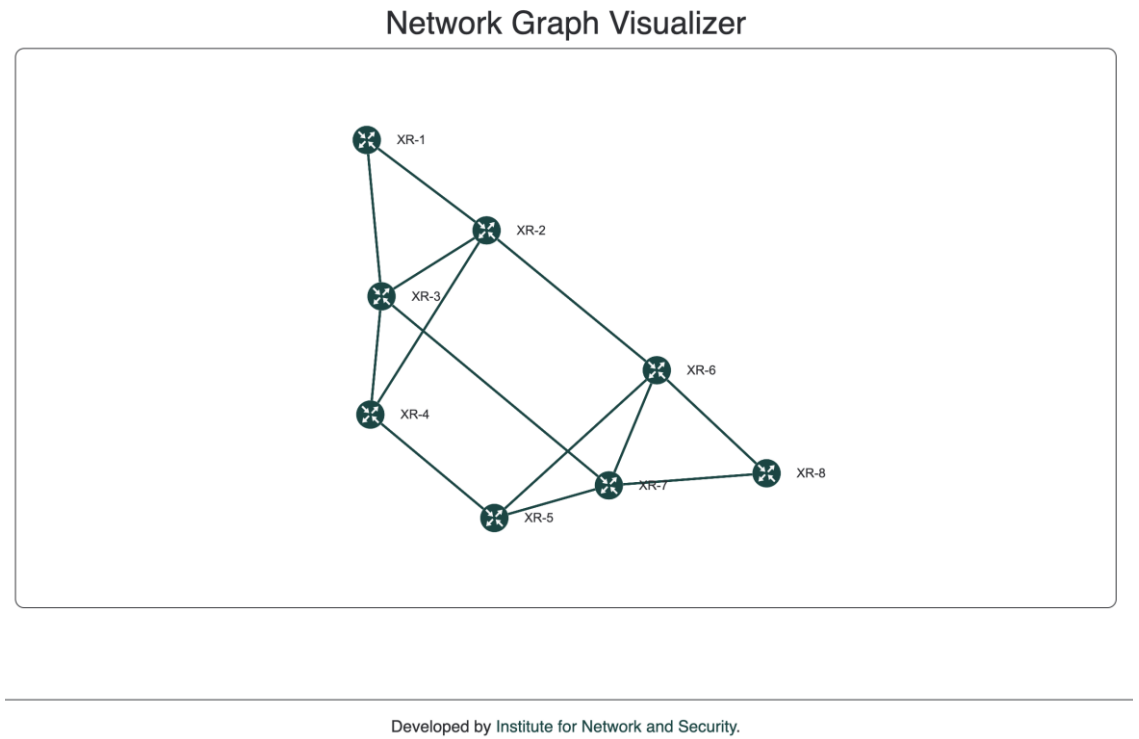
```
request := &jagw.TopologyRequest{
        Keys:       []string{},
        Properties: []string{},
}

// Get all nodes with all properties
response, err := client.GetLsNodes(context.Background(), request)
if err != nil {
        log.Fatalf("Error when calling GetLsNodes on request service: %s", err)
}
prettyPrint(response)
```

…. or simply in Python

```
HOST = '172.16.19.66'
PORT = '9903’

def get_all_lsnodes():
    channel = create_channel(HOST, PORT)
    with channel:
        stub = RequestServiceStub(channel)
        response = stub.GetLsNodes(TopologyRequest())
        print(response)

get_all_lsnodes()
```

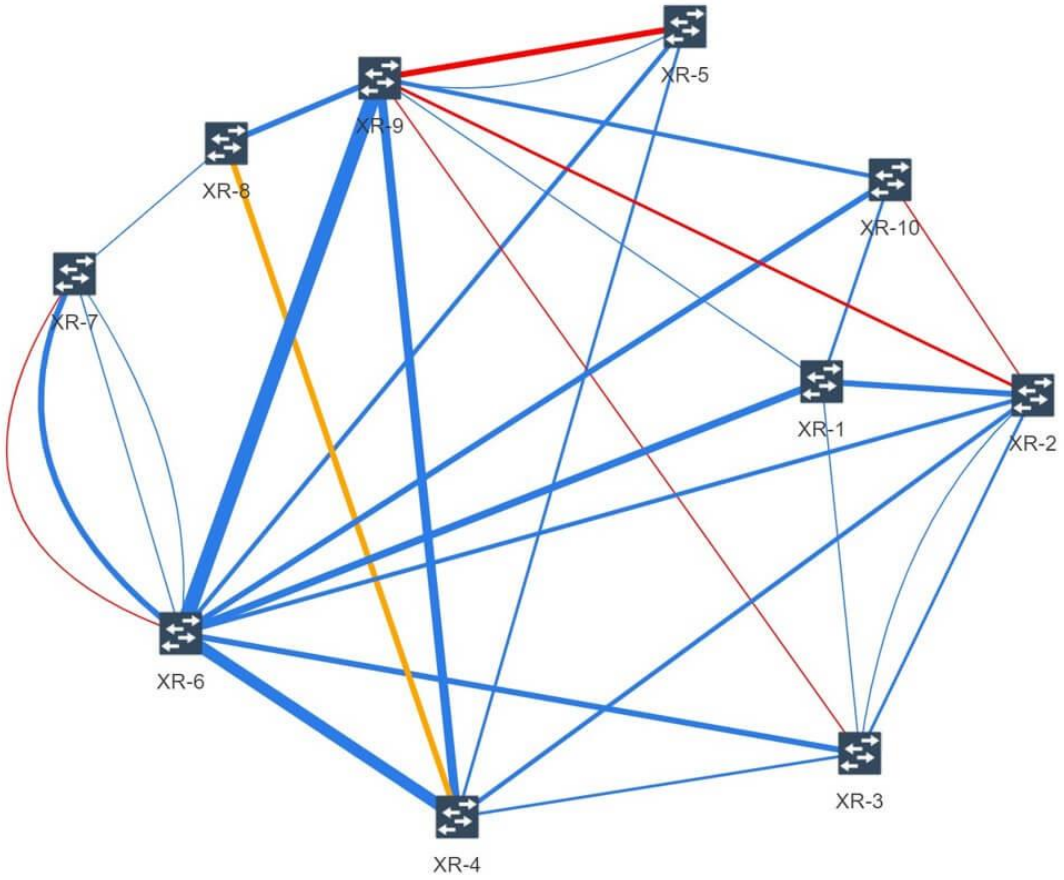
Demos

- Basic examples for interacting with the gateway: demo-sr-app
- Example app for showing the topology: network_graph_visualizer

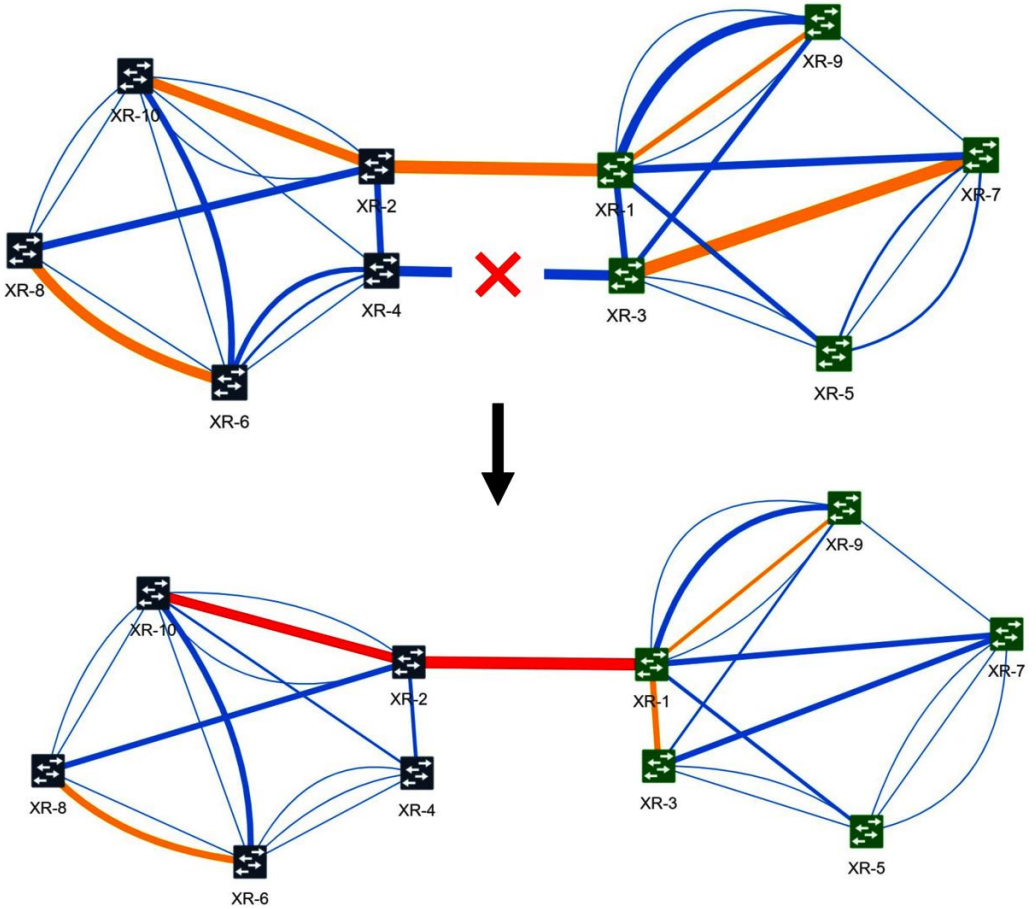


Network Analytics

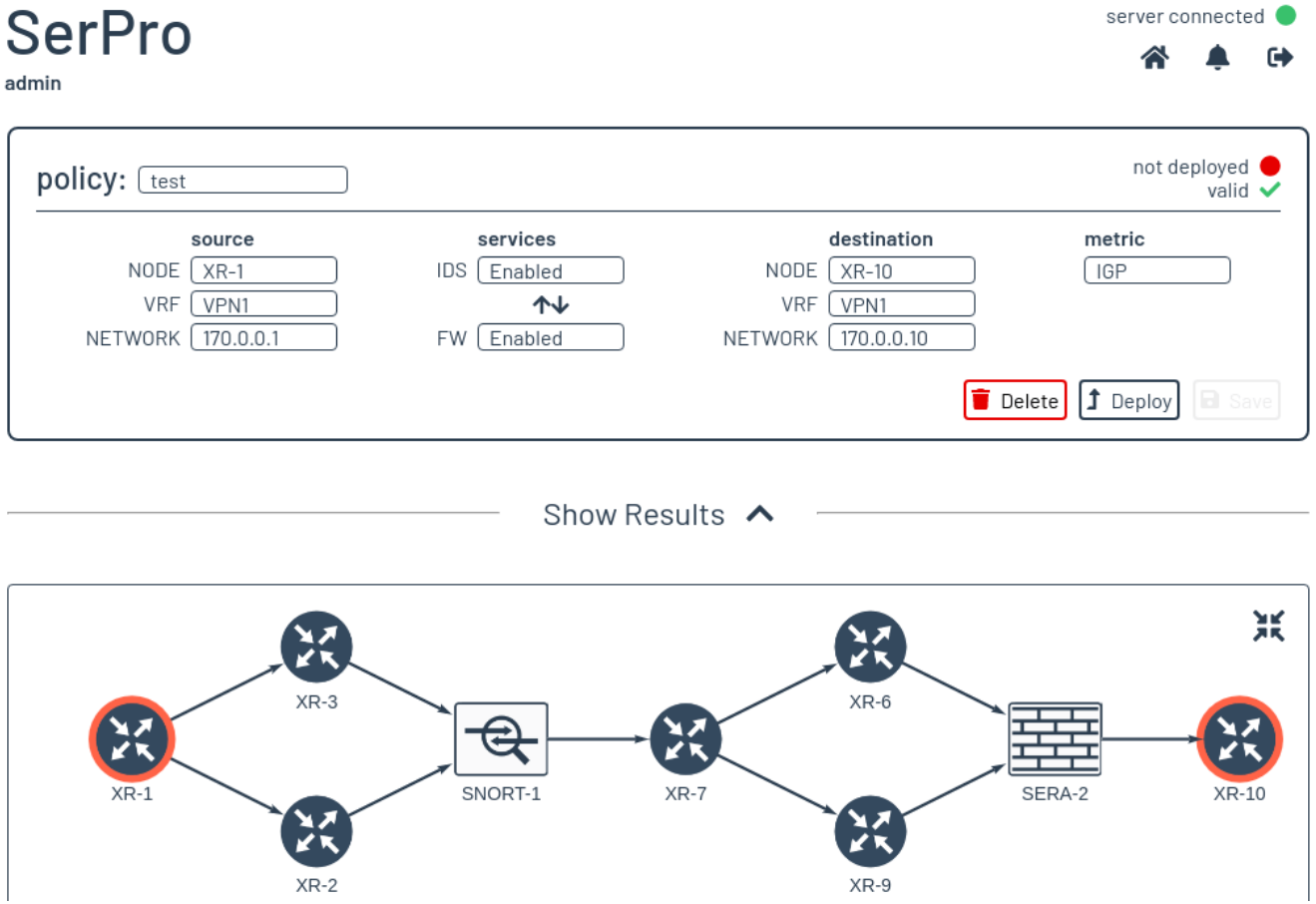
Link Quality Assessment



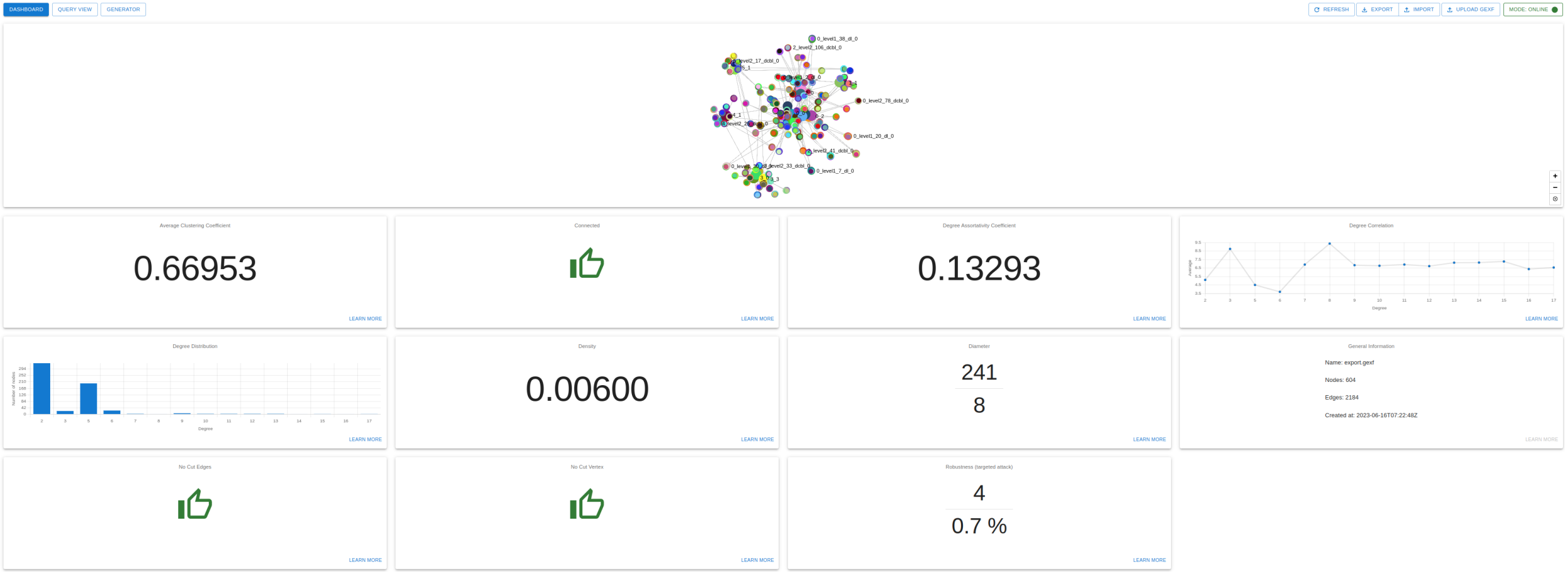
Link Saturation Prediction



Service Programming



Graph Analyzer / Generator



How to *

- All info in our GitHub Organization: [https://github.com/jalapeno-api-gateway]
  - Get started
  - Install/Update
  - Documentation
  - Go client library with examples
  - Python client library with examples
- More about us and our projects: [https://ost.ch/ins]
- Otherwise: [severin.dellsperger@ost.ch]

Questions?

Spicing Up Network Application Development, SwiNOG #38

INS Institute for Network and Security

Severin Dellsperger
severin.dellsperger@ost.ch

21 June 2023

OST
Eastern Switzerland
University of Applied Sciences

Sources

- https://www.ost.ch/de/die-ost/campus/campus-rapperswil-jona
- https://www.claise.be/srv6-flow-monitoring/
- https://datatracker.ietf.org/doc/html/rfc7854
- https://datatracker.ietf.org/wg/opsawg/documents/
- https://imgflip.com/memegenerator
- https://github.com/cisco-open/jalapeno
- https://github.com/sbezverk/gobmp
- https://github.com/influxdata/telegraf
- https://www.influxdata.com
- https://kafka.apache.org/
- https://www.arangodb.com/
- https://github.com/jalapeno-api-gateway/
- https://jalapeno-api-gateway.github.io/jagw/
- https://www.ost.ch/en/research-and-consulting-services/computer-science/ins-institute-for-network-and-security