ESnet

ENERGY SCIENCES NETWORK

ESnet’s Experience (so far) with Streaming Network Telemetry

Sowmya Balasubramanian

Bruce Mah

6th SIG-PMV Meeting, Dublin, Ireland

July 2, 2019

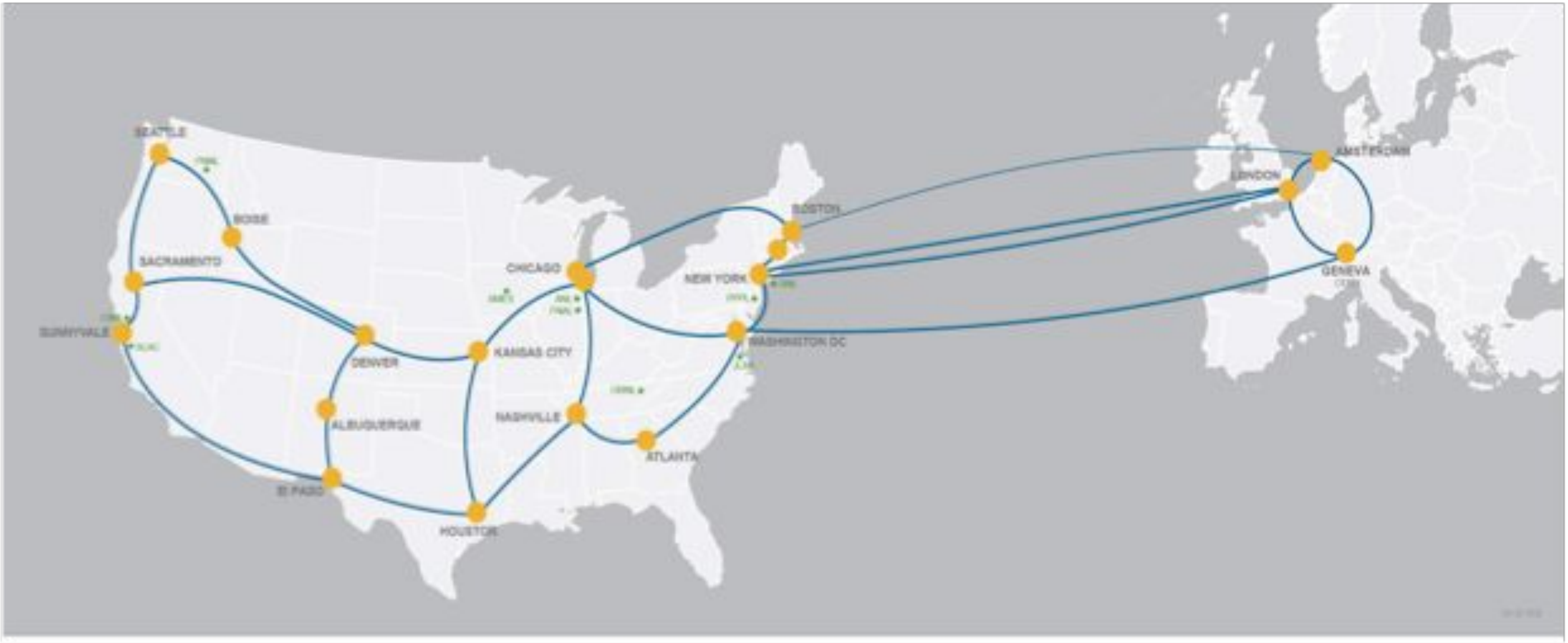
U.S. DEPARTMENT OF ENERGY Office of Science

BERKELEY LAB

Agenda

- ESnet
- Overview of Streaming Telemetry
- Perceived Benefits
- Telemetry Architecture
- Deployment Model
- Experiences
- Lessons Learned
- Open Questions

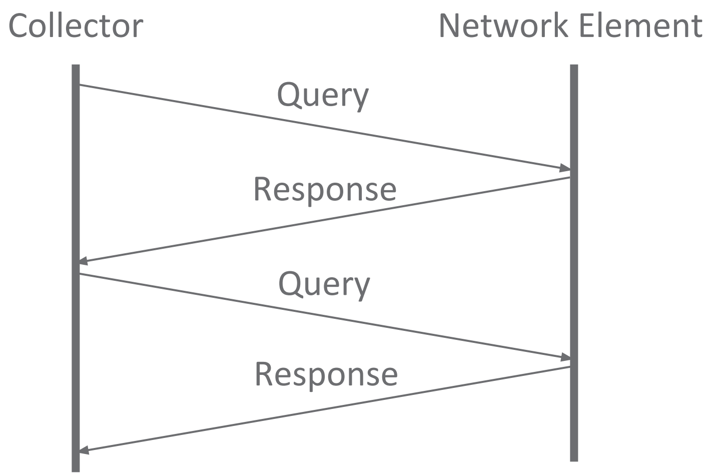
ESnet – Energy Sciences Network



U.S. Department of Energy's international research and education network.

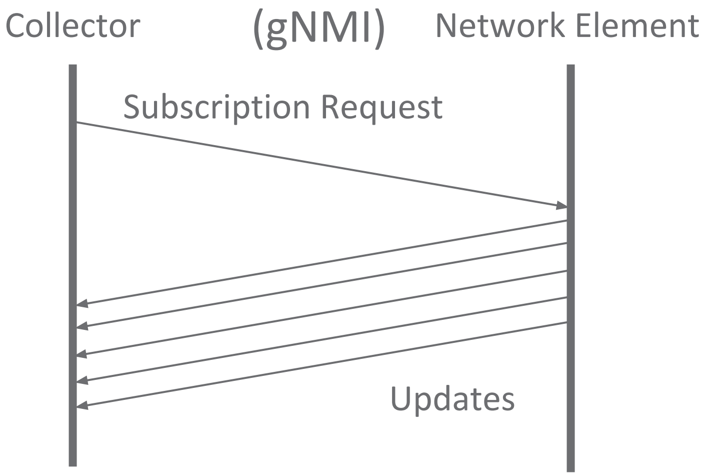
Streaming Telemetry: Benefits

Polling (SNMP)



Polls for all values of interest
Fixed polling interval can miss events
Uncertainty of measurement times

Streaming Telemetry (gNMI)

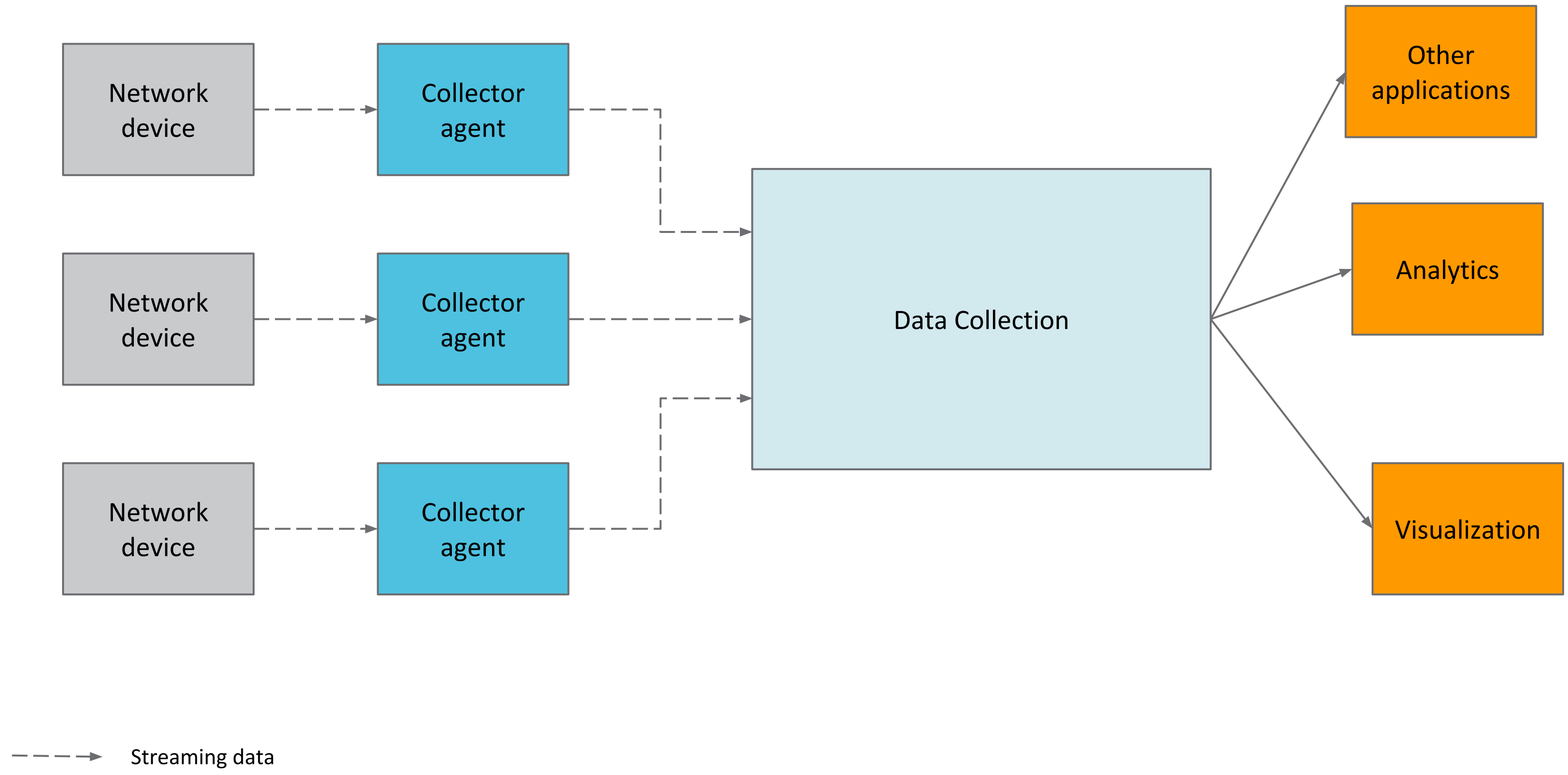


Only changed values are sent
More frequent updates
All measurements are timestamped

Perceived Benefits of Streaming Telemetry

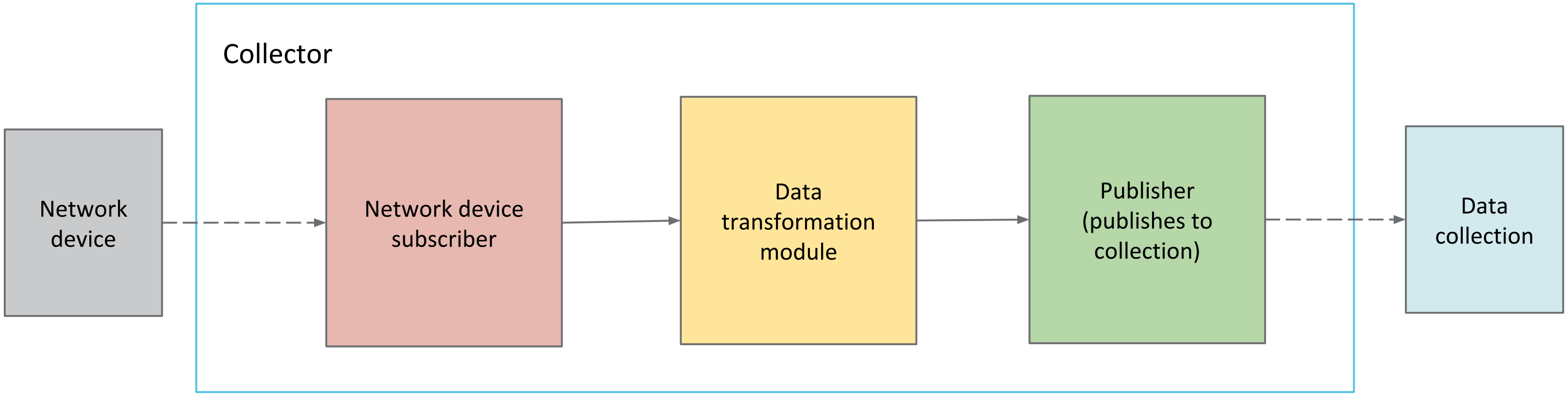
- Timestamped Measurements
- Less load on network equipment (push vs pull)
- More scalable updates
- More frequent / responsive updates
- Built-in security (TLS)

Telemetry and Analytics Architecture



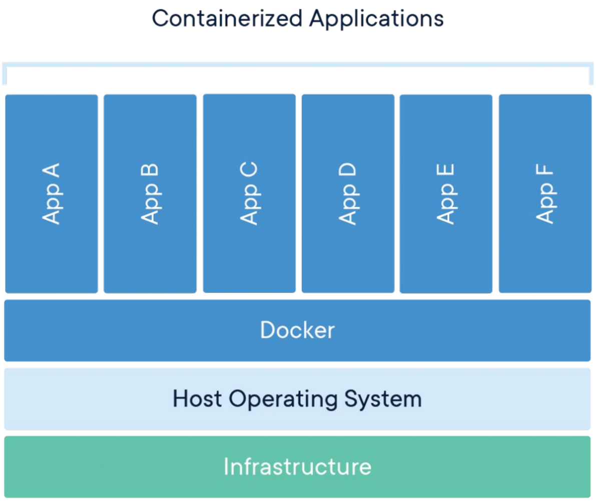
Collector

- Simple lightweight process
- Modular
- Transforms the data into a normalized model


Collector
Network device
Network device subscriber
Data transformation module
Publisher (publishes to collection)
Data collection

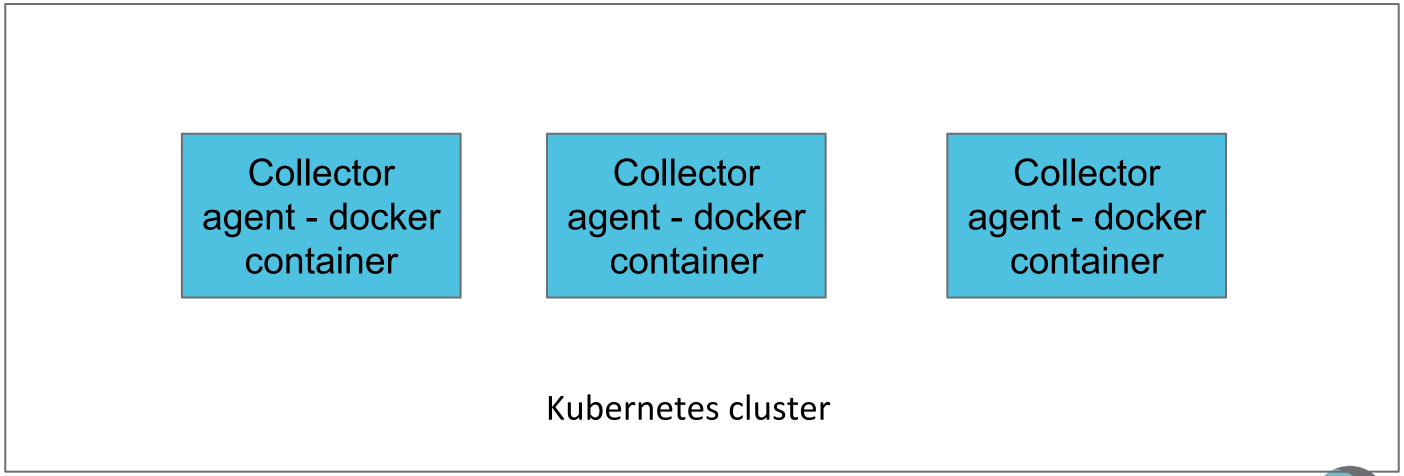
Deployment Model

- One collector for one network device (may extend to 2-3 network devices).
- Containerized environment
  - Better isolation than shared processes
  - Lightweight than traditional VMs



Kubernetes

- Automated deployment, scaling, and management of containerized applications
- Handles reloading of configs, replication, etc
- Keeps collector simple


Collector agent - docker container
Collector agent - docker container
Collector agent - docker container
Kubernetes cluster

Experiments

- Explore / investigate
  - Protocol versions, functionality, and features
  - Availability of models (defined using YANG)
  - Use available open-source tooling
- Stream data
  - Python-based collector:
    - Subscribes to telemetry updates for interface counters
    - Normalize data as necessary
    - Pushes updates to ESnet analytics back-end
  - Work with developers of data collection system
    - Generalize data models
    - Remove SNMP dependencies
  - Visualize data on experimental version of ESnet portal

Results: Arista

- Arista 7504R, EOS 4.20.3F
- gNMI 0.4, OpenConfig models

- Protocol / model testing
  - Update messages received every few seconds
  - Counter values sent on changes
  - TLS optional

- Prototype telemetry collector
  - Import interface counters into data collection system

Results: Nokia

- Nokia 7500 Service Router, SROS 15/16
- gNMI 0.3 (SROS 15) and gNMI 0.4 (SROS 16), proprietary and OpenConfig models
- OpenConfig data models require model-drive configuration mode
- OpenConfig telemetry requires configuration via OpenConfig

- Protocol / model testing
  - Counter updates sent every 10 seconds (minimum interval)
  - All counter values sent with every update
  - Proprietary and OpenConfig models are similar
  - TLS optional (SROS 16)

Lessons Learned (So Far)

- Streaming telemetry is new and evolving
- gNMI protocol functionality is vendor/version-specific
  - For example, ON_CHANGE is handled differently in the two vendors we tested
- Model support is vendor/version-specific
- May require changes to configuration process
- Streaming telemetry differences with SNMP
  - Data model vs. MIB organization
  - Subscription and push updates paradigm
  - Require different tooling

Open Questions

- Update frequency - How often and how useful?
- Less load on network equipment - Verify and quantify
- Scalable updates - Reduction in message size, data points
- OpenConfig vs. vendor specific data models
- Vendor implementations of gNMI differ
- Investigate other vendors?

ESnet

Thank you!
Questions?

bmah@es.net
sowmya@es.net

ESnet

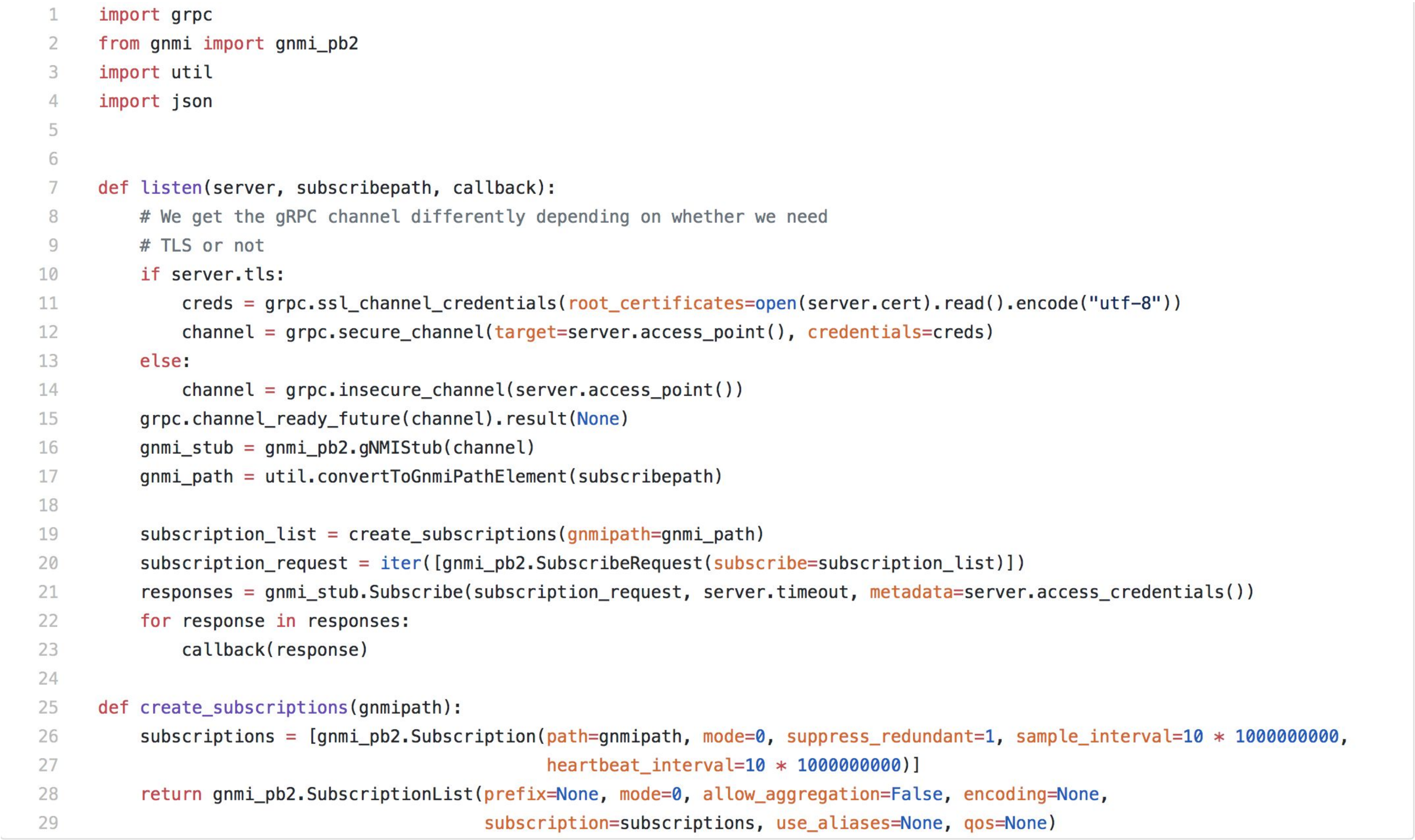
Additional Slides

Openconfig Data Models

- Standard Yang based models
- Models for interfaces, telemetry, vlan, etc
- SNMP to openconfig mapping:

ifInDiscards	in-discards
ifInErrors	in-errors
ifInUcastPkts	in-unicast-pkts (from ifHCInUcastPkts)
ifOutDiscards	out-discards
ifOutErrors	out-errors
ifOutUcastPkts	out-unicast-pkts (from ifHCOutUcastPkts)
ifHCInOctets	in-octets
ifHCOutOctets	out-octets
ifName	name
ifAlias	description

gNMI Client

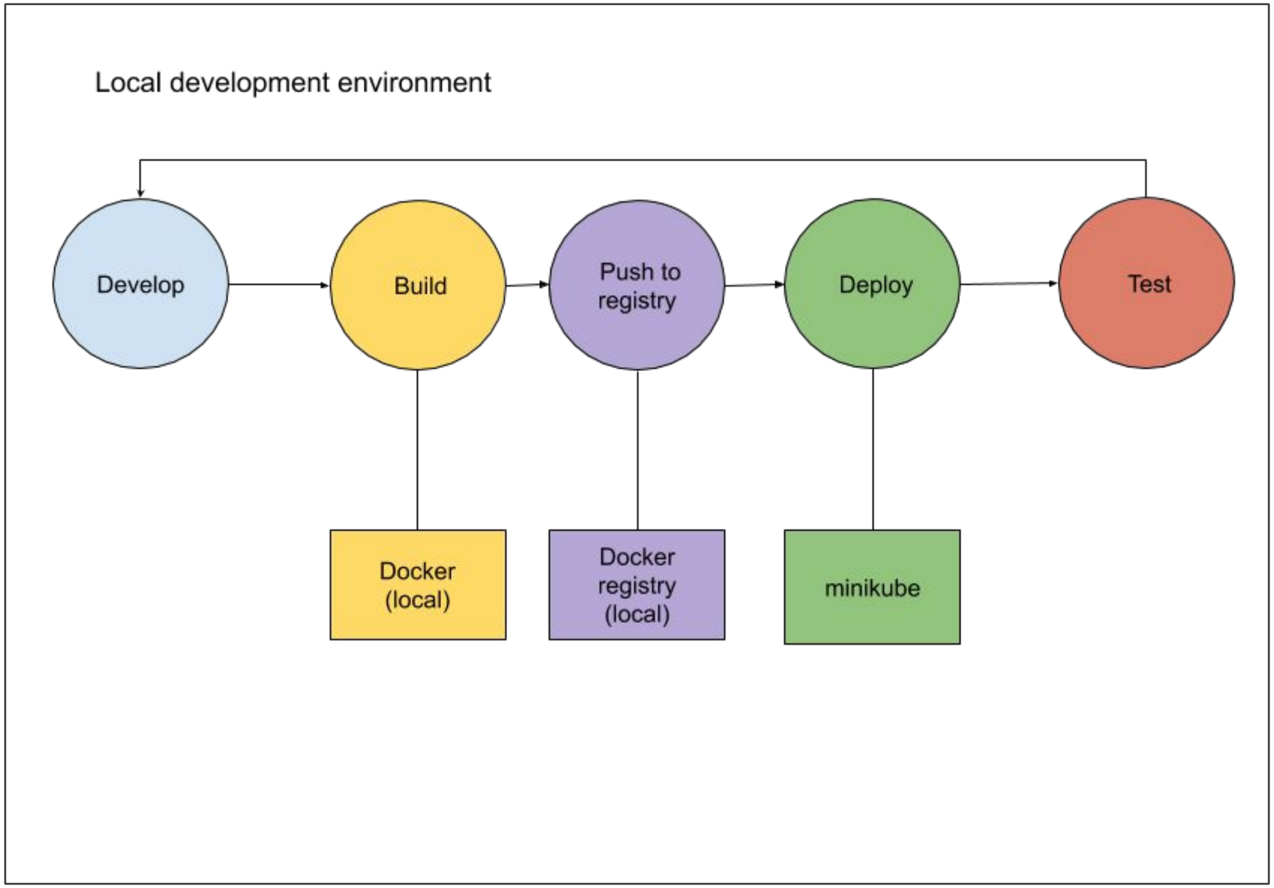


```python
import grpc
from gnmi import gnmi_pb2
import util
import json


def listen(server, subscribepath, callback):
    # We get the gRPC channel differently depending on whether we need
    # TLS or not
    if server.tls:
        creds = grpc.ssl_channel_credentials(root_certificates=open(server.cert).read().encode("utf-8"))
        channel = grpc.secure_channel(target=server.access_point(), credentials=creds)
    else:
        channel = grpc.insecure_channel(server.access_point())
    grpc.channel_ready_future(channel).result(None)
    gnmi_stub = gnmi_pb2.gNMIStub(channel)
    gnmi_path = util.convertToGnmiPathElement(subscribepath)

    subscription_list = create_subscriptions(gnmipath=gnmi_path)
    subscription_request = iter([gnmi_pb2.SubscribeRequest(subscribe=subscription_list)])
    responses = gnmi_stub.Subscribe(subscription_request, server.timeout, metadata=server.access_credentials())
    for response in responses:
        callback(response)

def create_subscriptions(gnmipath):
    subscriptions = [gnmi_pb2.Subscription(path=gnmipath, mode=0, suppress_redundant=1, sample_interval=10 * 1000000000,
                                           heartbeat_interval=10 * 1000000000)]
    return gnmi_pb2.SubscriptionList(prefix=None, mode=0, allow_aggregation=False, encoding=None,
                                     subscription=subscriptions, use_aliases=None, qos=None)
```

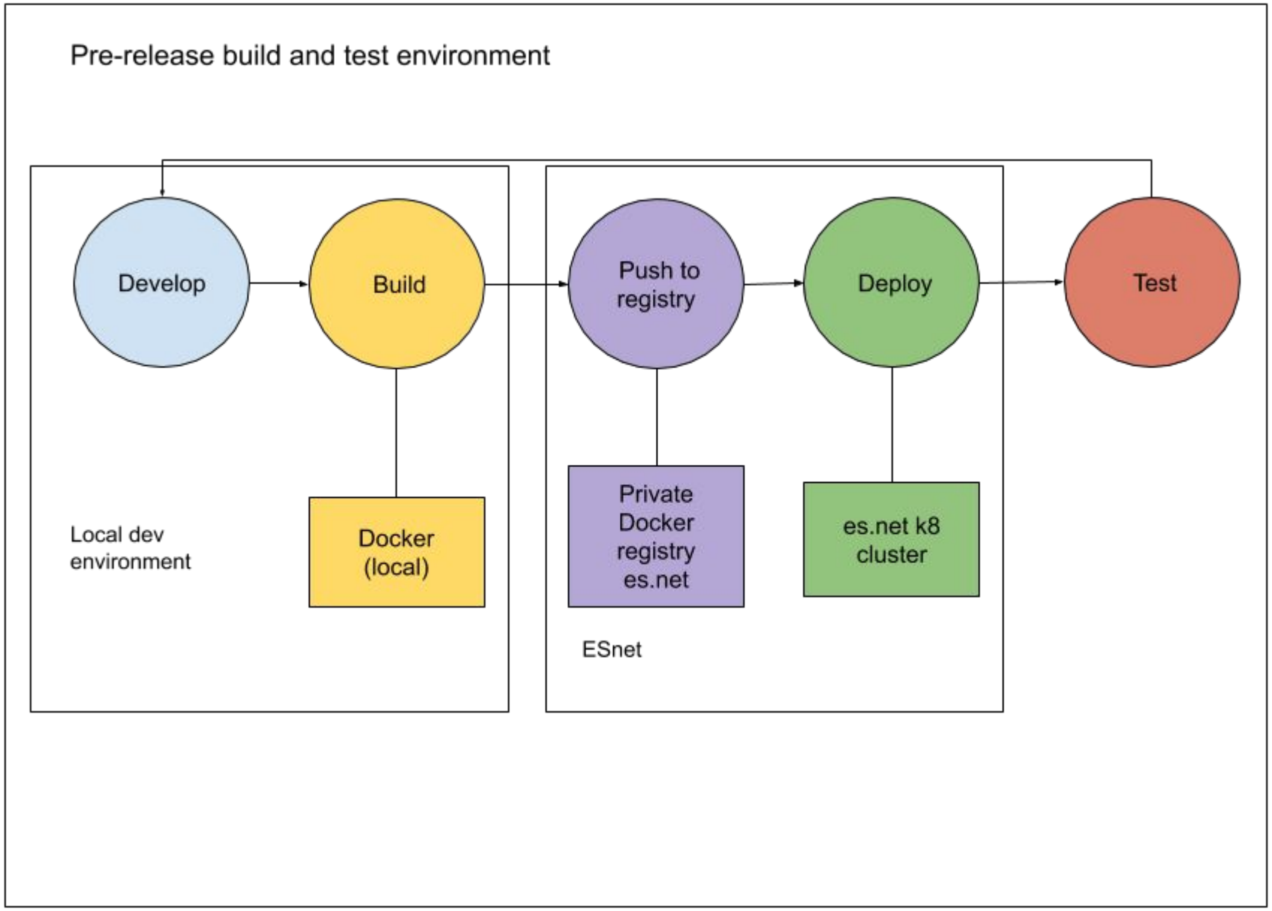
gNMI (gRPC Network Management Interface)

- gRPC - A high performance, open-source universal RPC framework
- Uses protocol buffers for serialization
- Works across different languages and platforms
- Client calls procedures in server
  - Procedures are defined using service definitions
  - “Subscribe” call for streaming telemetry
- Different subscription modes
  - STREAM, POLL, ONCE

Workflow management - Skaffold

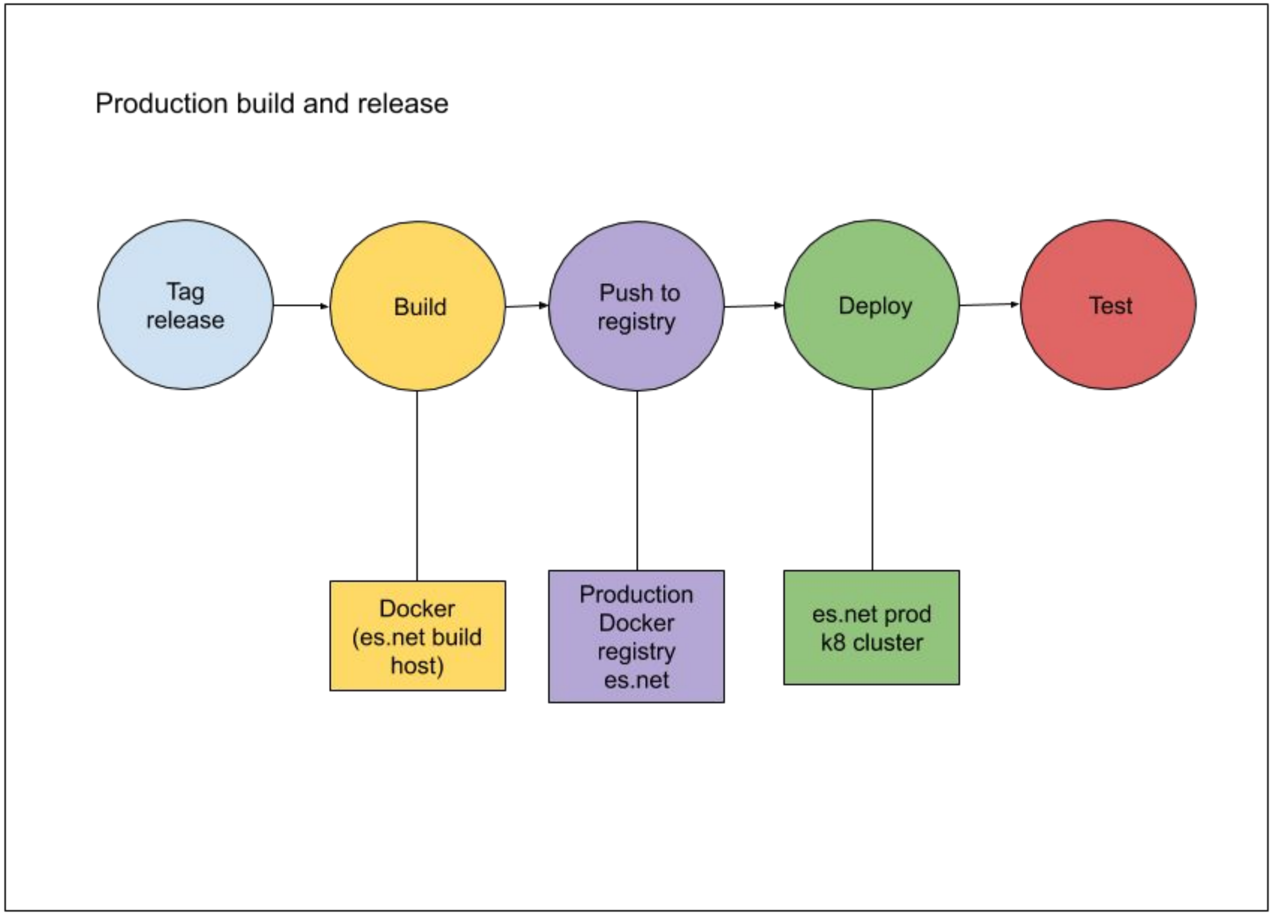


Build/Test environment



Pre-release build and test environment

Production Workflow



Production build and release

Tag release

Build

Push to registry

Deploy

Test

Docker (es.net build host)

Production Docker registry es.net

es.net prod k8 cluster