Model Driven Telemetry - Foundation for Big Data Analytics

Rada Stanic
Principal Systems Engineer, Cisco

Agenda

- Current Challenges
- Model Driven Approach
- Key Lessons Learnt
- Collectors
- Conclusion

Current Challenges

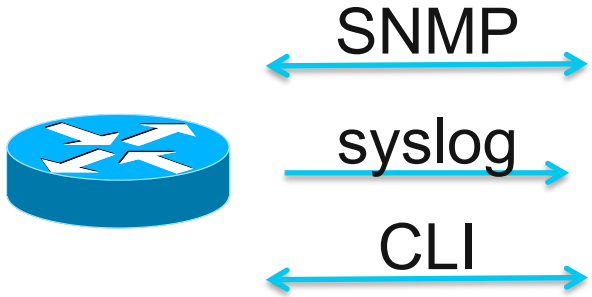
It’s All About Operations

- Precedent is well established

> “Today, our time series metric ingestion service handles more than 2.8 billion write requests per minute, stores 4.5 petabytes of time series data, and handles 25,000 query requests per minute.”
> https://blog.twitter.com/2016/observability-at-twitter-technical-overview-part-i

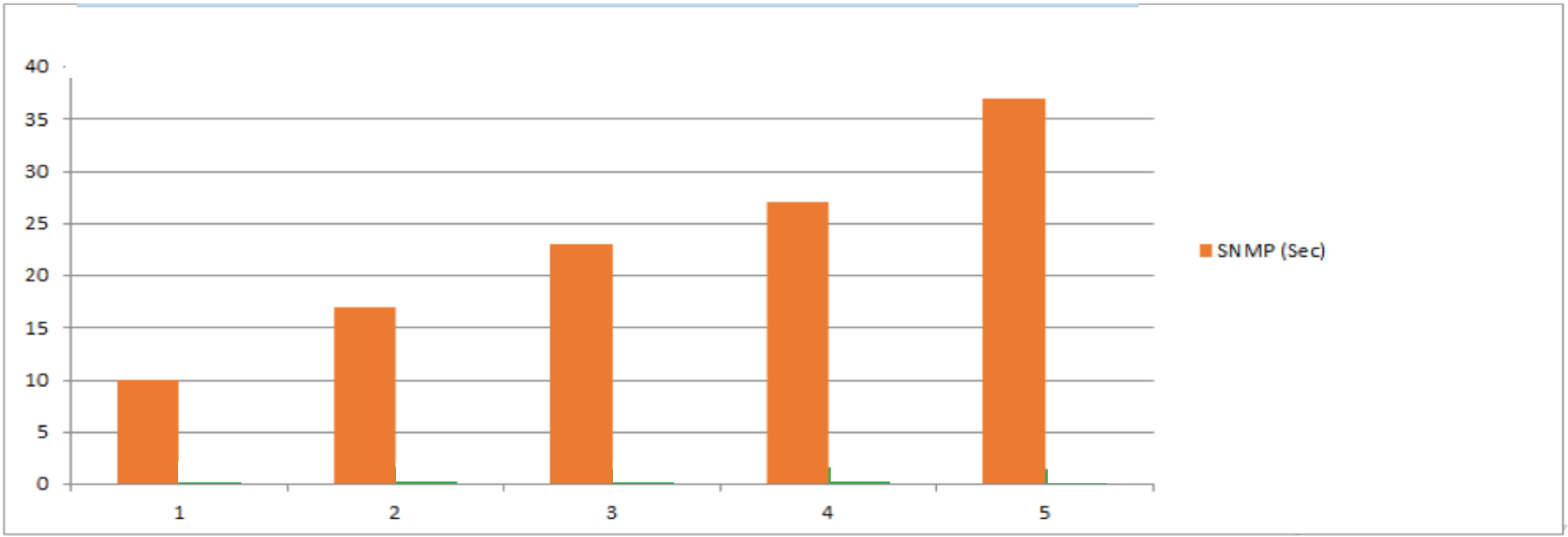
- Network monitoring has work to do.

Why Network Visibility Is Hard



- Too Slow
- Incomplete
- Network-Specific
- Hard to Operationalize

What Happens When You Push SNMP Too Hard



- 10 second poll
- 3 pollers
- 30 minute measurement intervals

- 288 100Gig E Interfaces (Line Rate)
- SNMP: IF-MIB (query by row)

Why This Matters Now

Real-Time Use Cases

- Network Health
- Troubleshooting / Remediation
- SLAs, Performance Tuning
- Capacity Planning
- Security

Trends

- Centralized / Software-defined
- Speed
- Scale

Capabilities

Analytics Ready…or Not

Not Ready

- Send unstructured text
- Invent a new stack from scratch
- Make it networking-specific
- Support minimal programming languages and toolchains

Traditional Networking Approach

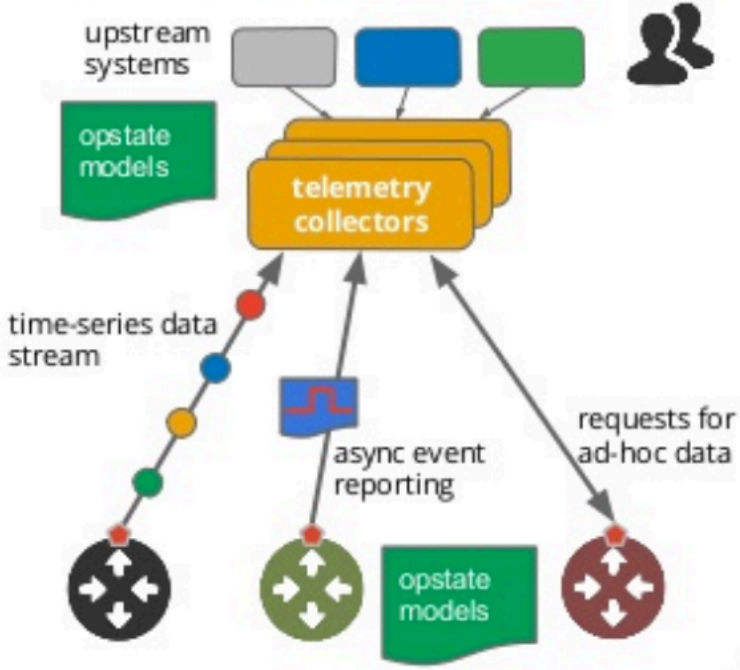
Ready

- Structured data only
- Re-use common encodings, RPCs, transports
- Learn/borrow from other domains
- Support many programming languages and toolchains

IT / Cloud / Web

Source: Google @ Bay Area OpenDaylight Meetup 06/16

Streaming telemetry -- moving forward from SNMP



- stream data continuously -- with incremental updates
- telemetry sent based on subscriptions
- observe network state through a time-series data stream
- device data follows a common model
- efficient, secure transport protocols (gRPC)

Aim to deprecate SNMP in our network by 2017

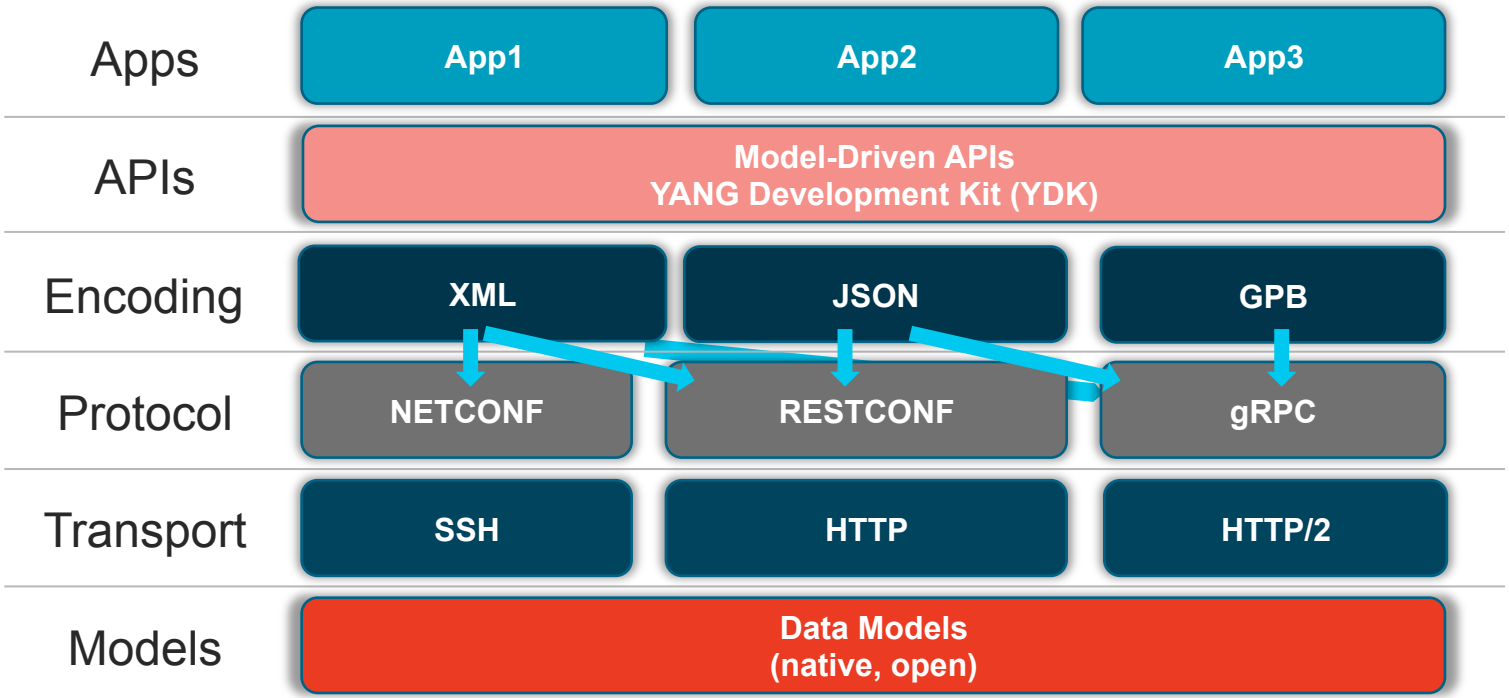
OPENCONFIG

Model Driven Approach

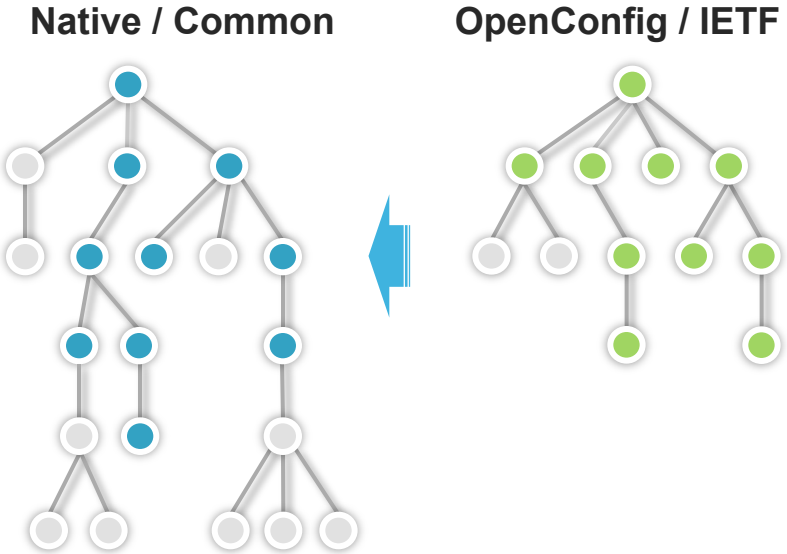
Data Model Driven Management: Architectural Pillars

Network Programmability	• Data accessible via published model driven interfaces or APIs • Machine friendly (CLI is not) • Enables automation @ scale
Visibility & Telemetry	• Operational Data, Deep analytical hooks • Policy-based, flexible, Push Model, more granular, greater scalability • Model driven data structures
Application Hosting	• Operators may run their own or 3rd party off the shelf applications built with Linux tool chains • Run custom applications built with vendor provided SDK • Can be run natively co-located with router OS (or) inside an LXC or Docker container

Model-Driven Programmability Stack



YANG Data Models



- Yang Modeling language (initially for NETCONF)
- Model structures data (config and operational) as a tree
- .yang files are self-documented and ship with devices
- Native models provide full feature coverage
- OpenConfig and IETF models are mapped to native models
- Benefits: automation, automation, automation
  - Machine friendly tree structure with key:value pairings (vs. CLI grammar)
  - Auto-generation of code
  - Support for programming language bindings

Yang Models Are Created By Many

https://github.com/openconfig/public

http://www.openconfig.net/

Vendors

OpenConfig

IETF

Other

"Native Models"

Config Model Example

```
module: Cisco-IOS-XR-telemetry-model-driven-cfg
   +--rw telemetry-model-driven
      +--rw sensor-groups
      |  +--rw sensor-group* [sensor-group-identifier]
      |        ...
      +--rw subscriptions
      |  +--rw subscription* [subscription-identifier]
      |        ...
      +--rw destination-groups
      |  +--rw destination-group* [destination-id]
      |        ...
      +--rw enable?                empty
```

What Data to Collect

When

Where to Send and How

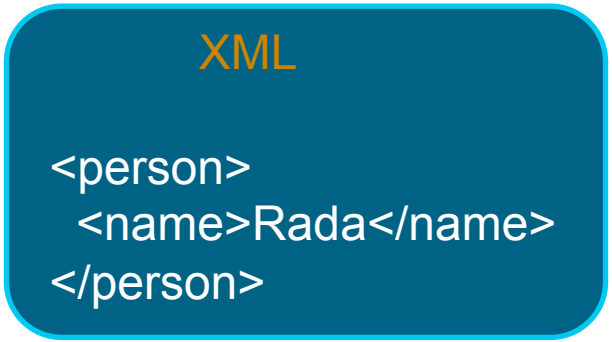
```
module: openconfig-telemetry
   +--rw telemetry-system
      +--rw sensor-groups
      |  +--rw sensor-group* [sensor-group-id]
      |        ...
      +--rw destination-groups
      |  +--rw destination-group* [group-id]
      |        ...
      +--rw subscriptions
         +--rw persistent
         |     ...
         +--rw dynamic
               ...
```

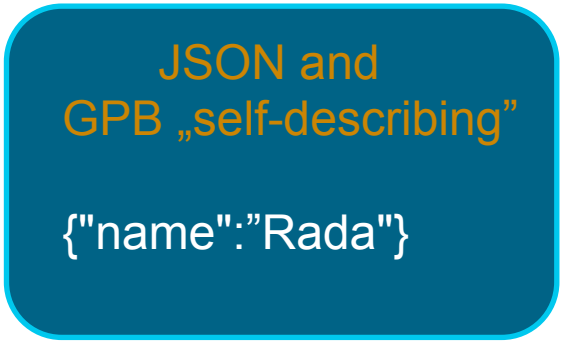
What Data to Collect

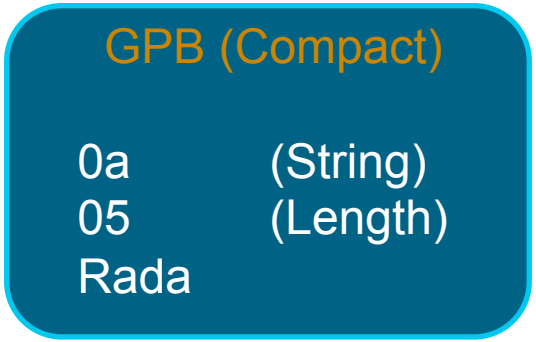
Where to Send and How

When

Encoding Types – What They Look Like





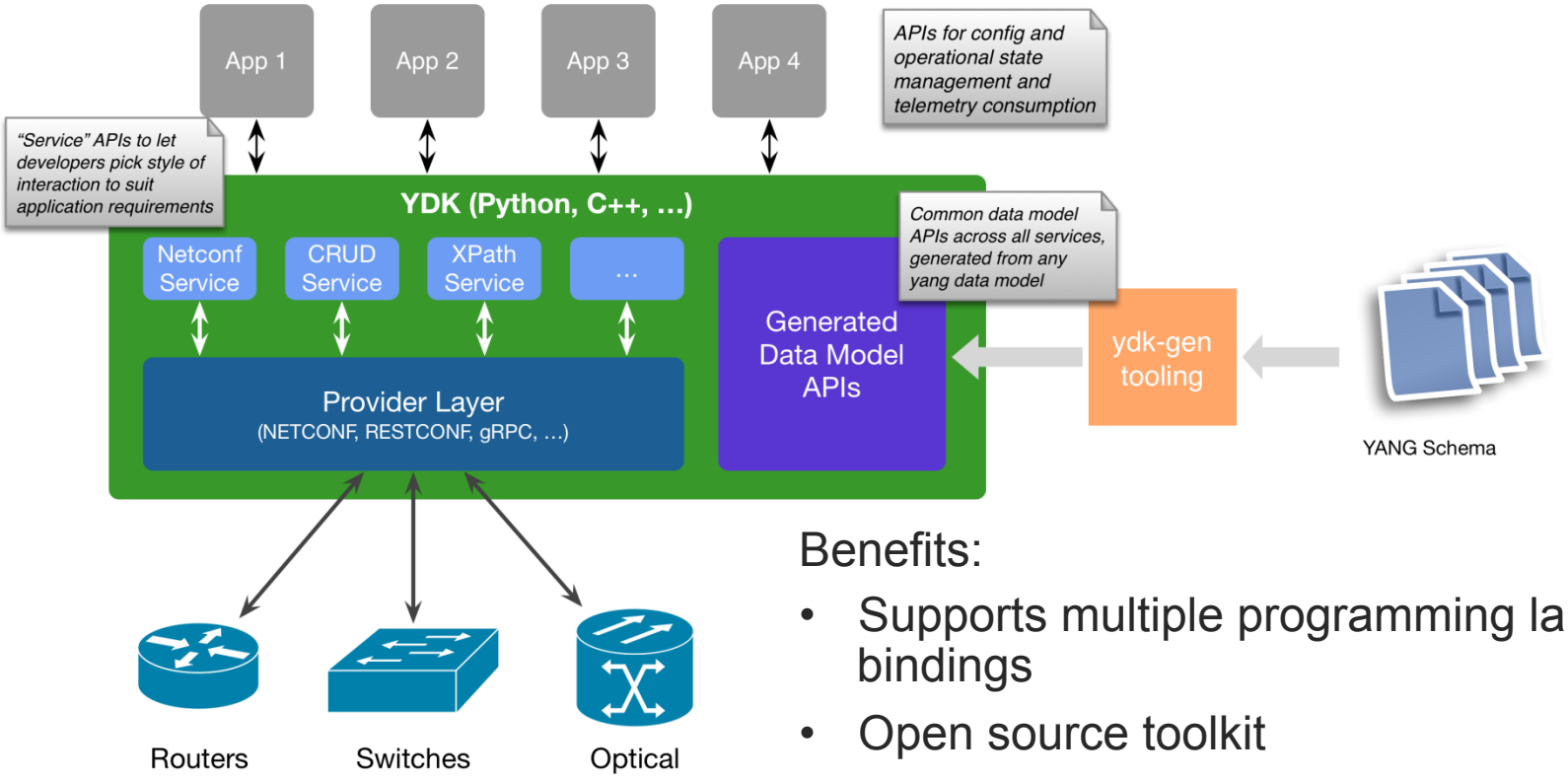


35 Bytes

16 Bytes

7 Bytes

YANG Development Kit (YDK)

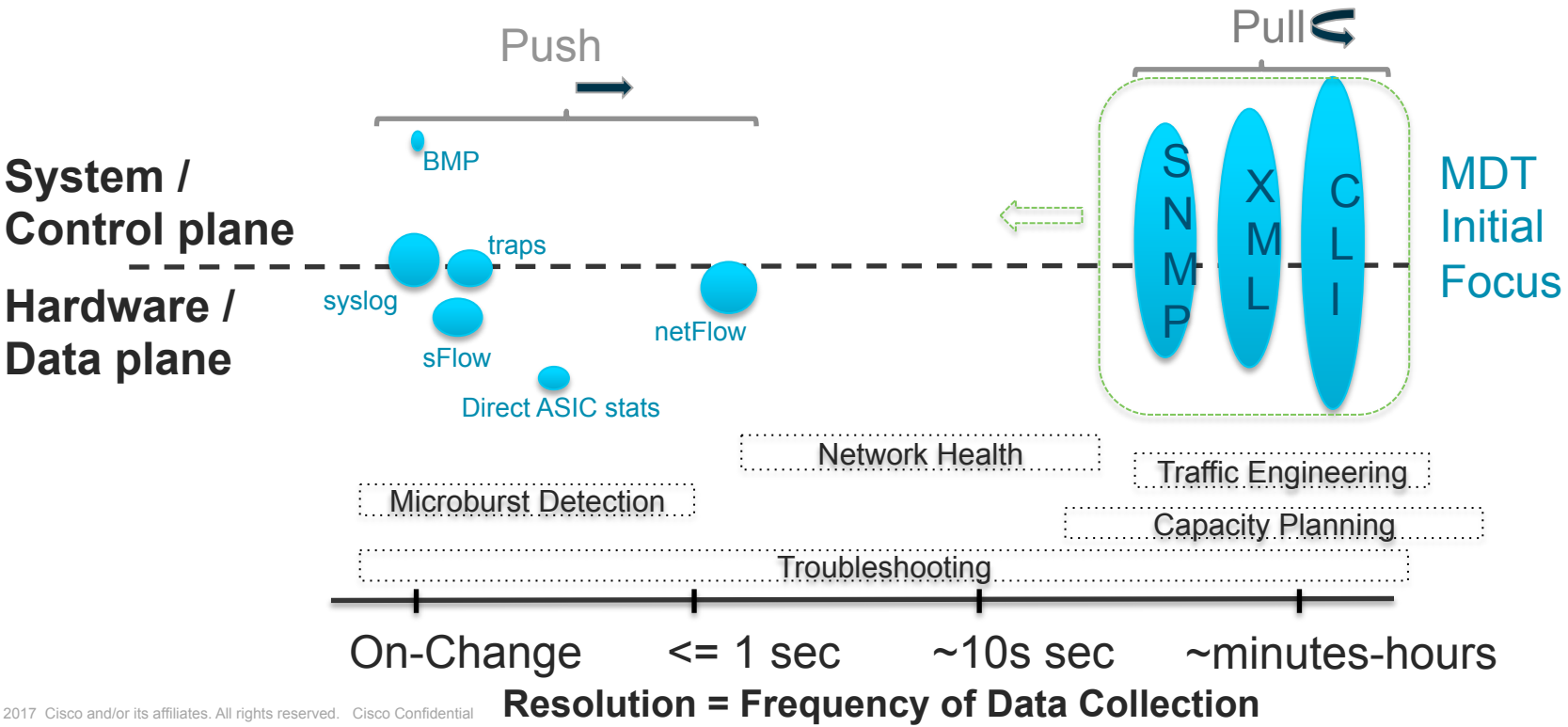


Benefits:

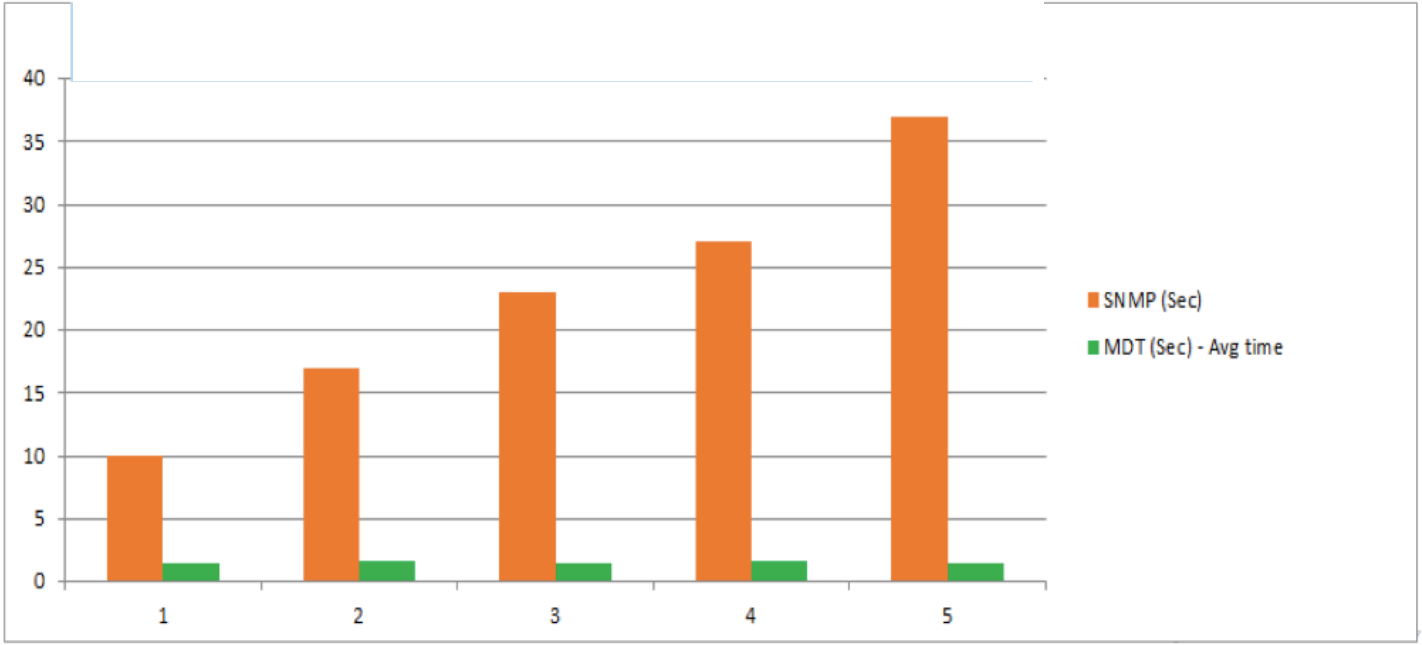
- Supports multiple programming language bindings
- Open source toolkit
- Community driven development

Lessons Learnt

Telemetry Means Different Things to Different People



It’s Not Hard to Beat SNMP



- 10 second poll / push
- 3 pollers / telemetry receivers
- 30 minute measurement intervals

- 288 100Gig E Interfaces (Line Rate)
- SNMP: IF-MIB (query by row)

You Don’t Know What You Love Until It’s Gone

- Well-Known Name:

OID: 1.3.6.1.2.1.2.2.1.10.4
MIB: IF-MIB
Object: ifInOctets

In reality, not enough of these

- Not-So-Well-Known Name:

"RootOper.InfraStatistics.Interface({'InterfaceName' : 'GigabitEthernet0/0/0/0'}).GenericCounters",
{'BytesReceived': 443140326}

Data Models are Good for Telemetry

- Follow the industry standard: YANG

Vendor-Specific

```
module: Cisco-IOS-XR-infra-statsd-oper
  +--ro infra-statistics
    +--ro interfaces
      +--ro interface* [interface-name]
        +--ro latest
          |  +--ro generic-counters
          |     +--ro packets-received?  uint64
          |     +--ro bytes-received?    uint64
          |     +--ro packets-sent?      uint64
          |     +--ro bytes-sent?        uint64
        <snip>
```

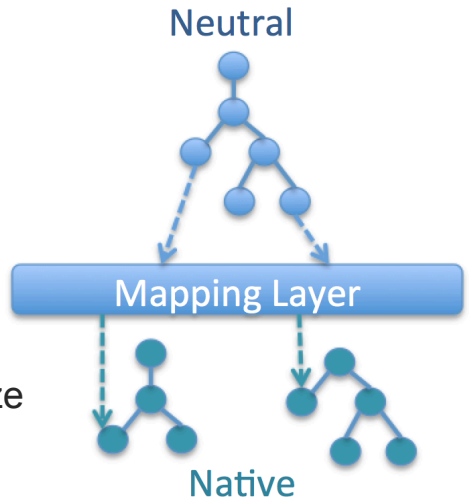
Vendor-Neutral

```
module: openconfig-interfaces
  +--rw interfaces
    +--rw interface* [name]
      |  +--ro counters
      |     +--ro in-octets?             yang:counter64
      |     +--ro in-unicast-pkts?       yang:counter64
      |     +--ro in-broadcast-pkts?     yang:counter64
      |     +--ro in-multicast-pkts?     yang:counter64
      |     +--ro in-discards?           yang:counter64
       <snip>
```

Benefits of YANG: Order by system, pull containers, readable, extensible, hierarchical, reusable, versioning, modularity, etc

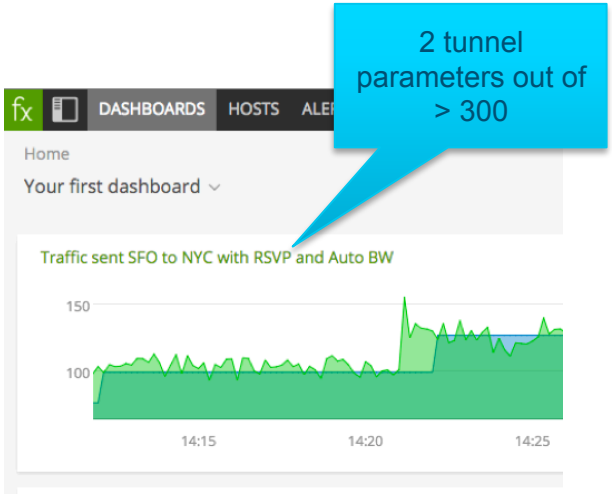
Some Models Are Harder Than Others

- Less “Impedance Mismatch” Than Config
  - Diminished Many to One Problem
  - Same underlying objects, different paths
  - Mapping oper data is mostly stateless
- Implications for Performance
  - Scatter-gather problem
  - Model mapping can be resource intensive -> optimize software architecture for high speed export
- Model Development Cycle May Lag



Filtering Is A Balancing Act

- Massive Amounts of Data
  - Bulk Collection is Efficient
  - Bulk Processing/Export Not So Much
  - Compression (Lossless) is Good
- On-Box Filtering Loses Data
  - Can’t Change Your Mind About What’s Important Later
  - Can’t Scale Out Embedded Processing
- Options
  - Balance Filtering Between Network and Collector
  - Don’t Rely on Periodic Push for “On-Change” Data (e.g. RIB events)



Self-Describing Data Has Pros and Cons

GPB – “compact”

```
  1: GigabitEthernet0/0/0/0
 50: 449825
 51: 41624083
 52: 360333
 53: 29699362
 54: 91299
    <snip>
```

2X faster
Operationally more complex (but not relative to SNMP!)

JSON/GPBKV – “self-describing”

```
{InterfaceName: GigabitEthernet0/0/0/0
    GenericCounters {
      PacketsSent: 449825
      BytesSent: 41624083
      PacketsReceived: 360333
      BytesReceived: 29699362
      MulticastPacketsReceived: 91299
      <snip>
```

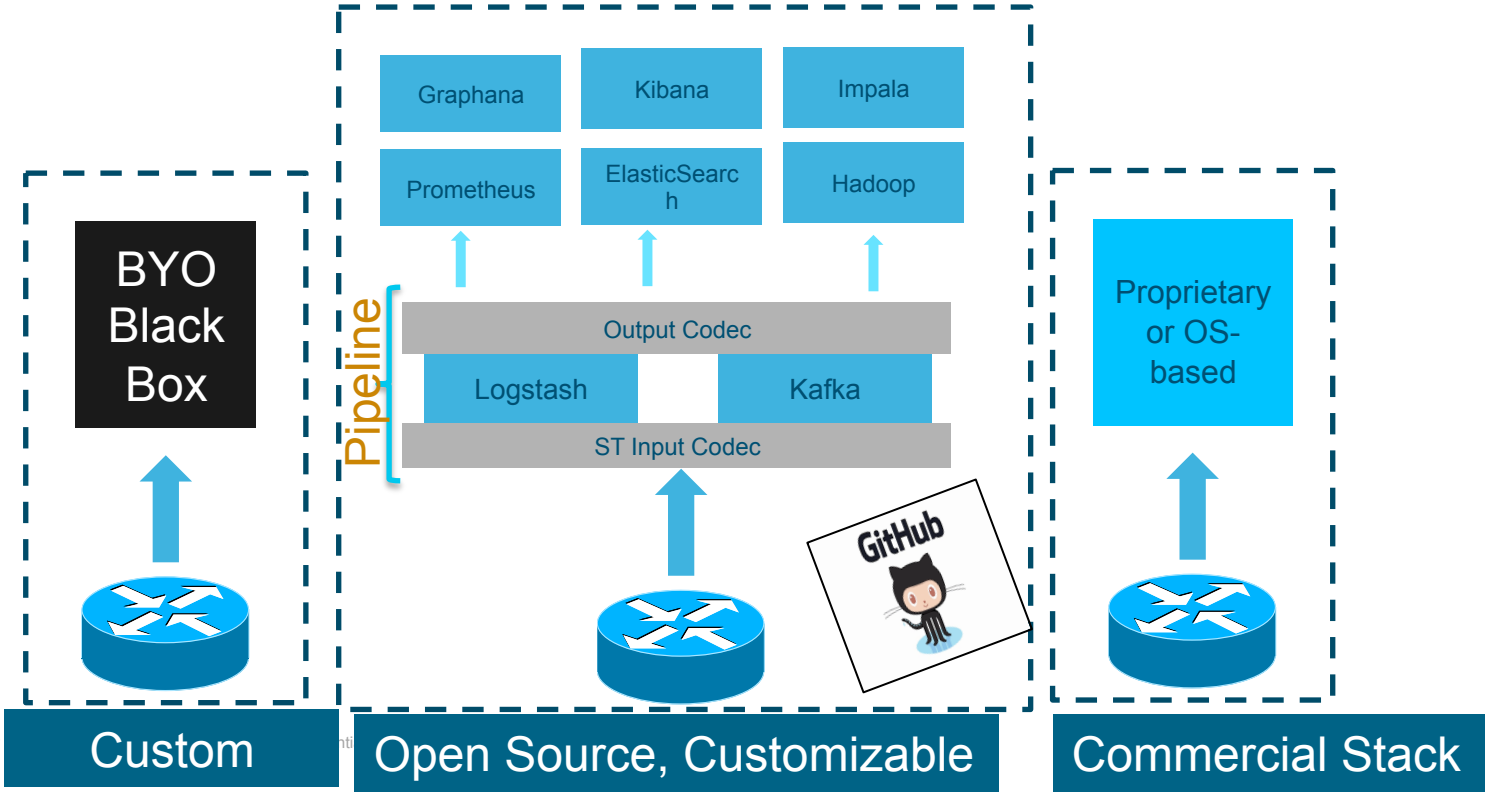
3X larger
Native models: still need heuristics for key names

Transports Have Trade-offs

- UDP – can't fragment, packets might get lost, insecure, fast
- TCP – can fragment, guaranteed delivery, insecure, vulnerable to loss
- Two motivations for gRPC
  - Unification of management protocol
  - Secure, two-way streaming

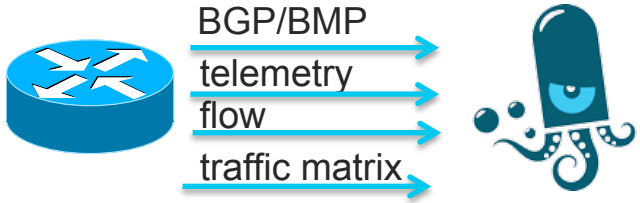
Data Model	Encoding	Transport	Score
Native	Binary	UDP	100%
		TCP	88%
	Self-describing		42%
		gRPC	38%

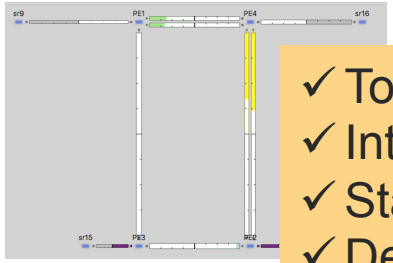
Time to Think About Analytics Pipeline & Platform



Diversity Is In the Eye of Beholder

- No Single Metric Describes the Network



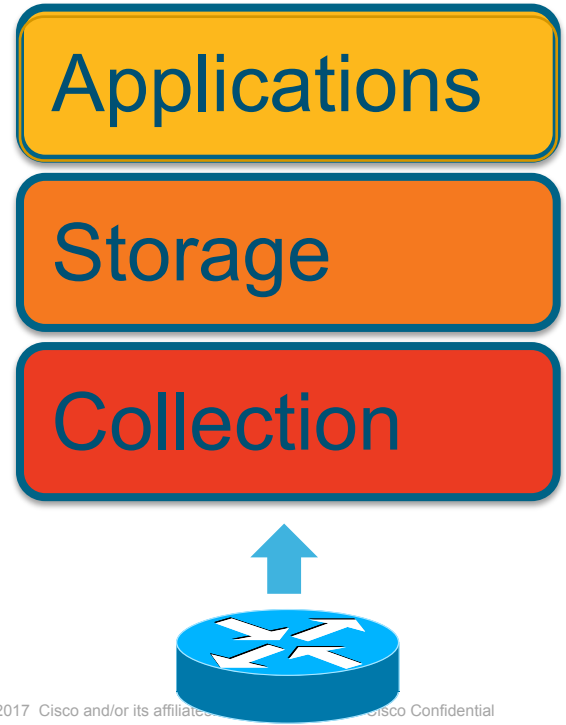


- ✓Topology
- ✓Interfaces
- ✓Stats
- ✓Demand

- Application + Networking Data ?

Collectors

A Very Basic Analytics Platform Architecture

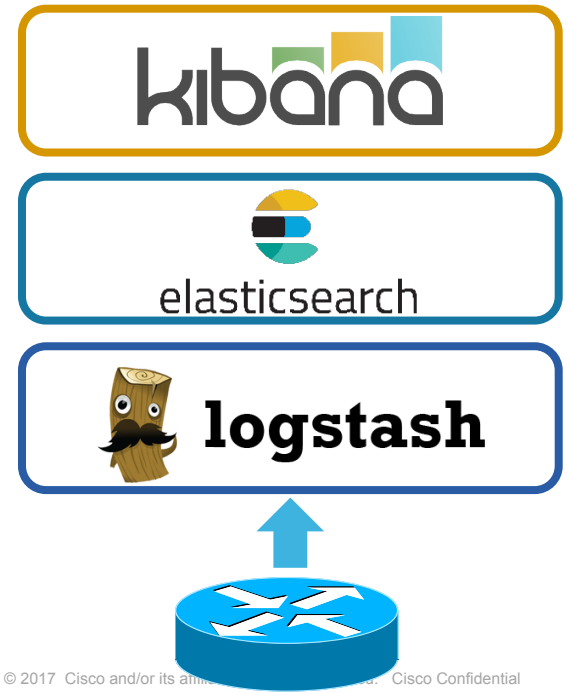


Visualize, Alert, Automate

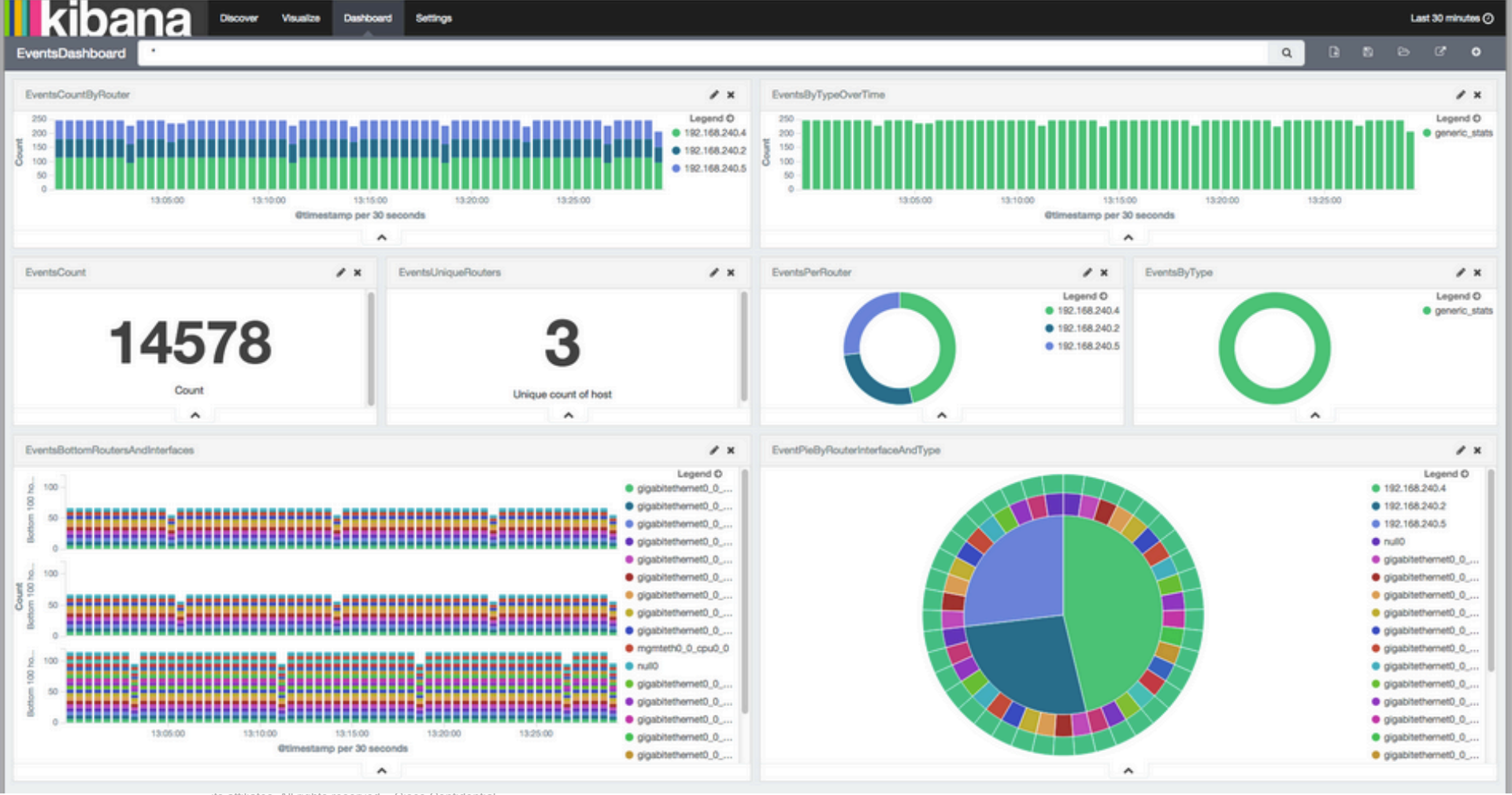
Index, Search, Store

Ingest, Aggregate, Normalize

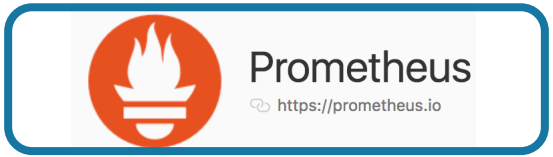
The Elastic Stack: A Popular OpenSource Stack



- Formerly known as “ELK”
- Commerical/Cloud via elastic.co
- Lucene based (inverted index)
- Data stored as documents
- Full text search and log management
- https://github.com/cisco/bigmuddy-network-telemetry-stacks/tree/master/stack_elk



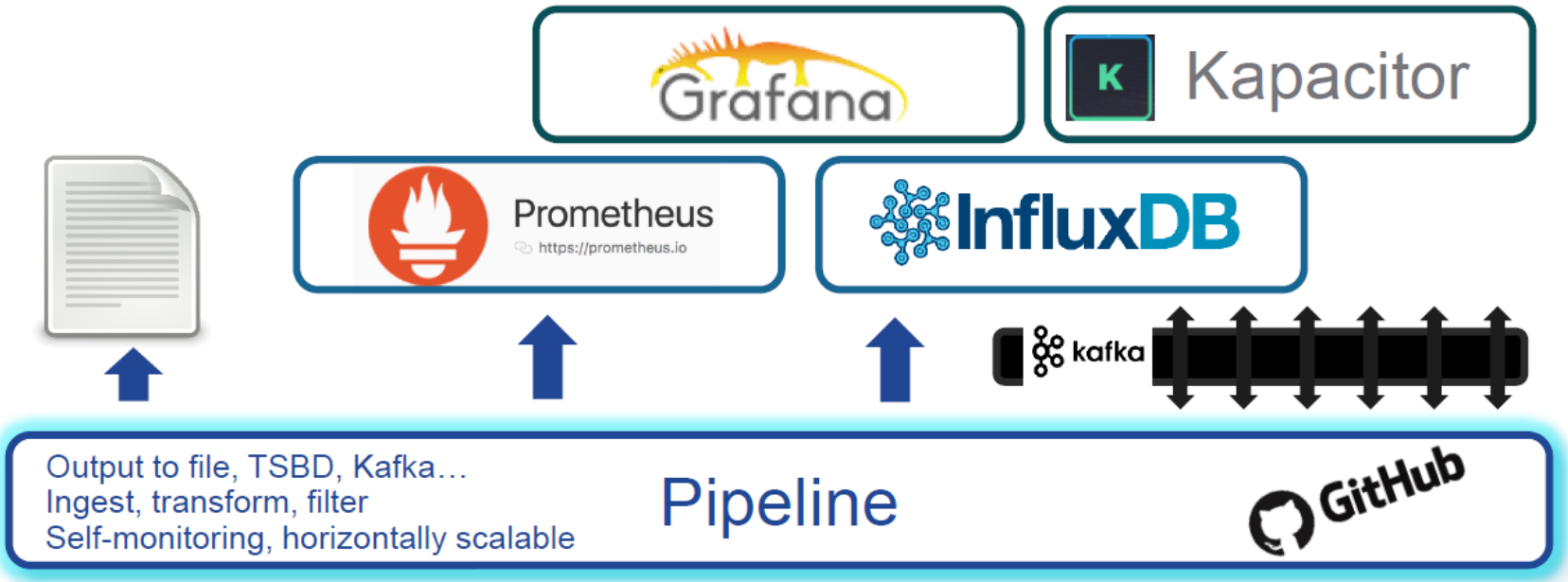
A Large, Fast-Moving Landscape



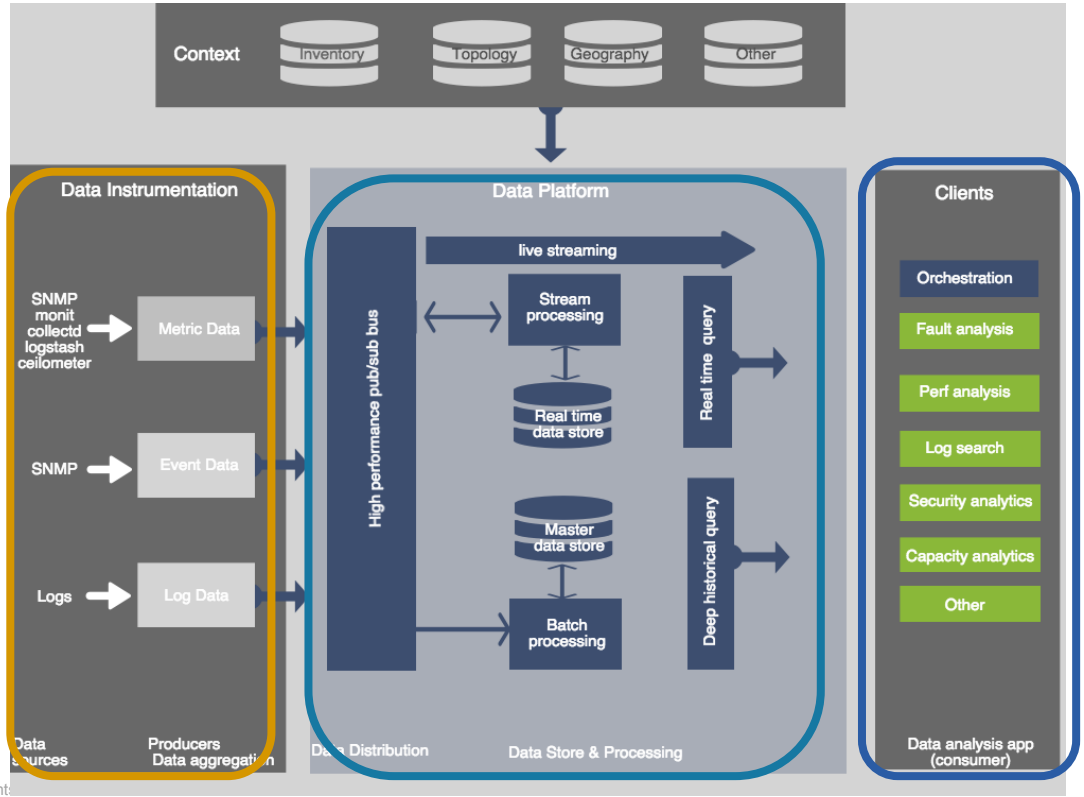
- Cloud Native Computing Foundation
- Focus on Reliability

- Commercial version: influxdata.com
- Flexible realtime query

Pipeline: An Open Source Collector

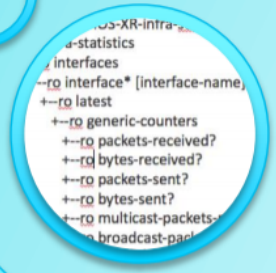


PNDA.IO: A More Complex Architecture



Key Takeaways

A Big Data Platform Is In Your Future



Data Models Are Your Friends

Speed & Scale Require Visibility

It’s Not Hard to Beat SNMP


CISCO