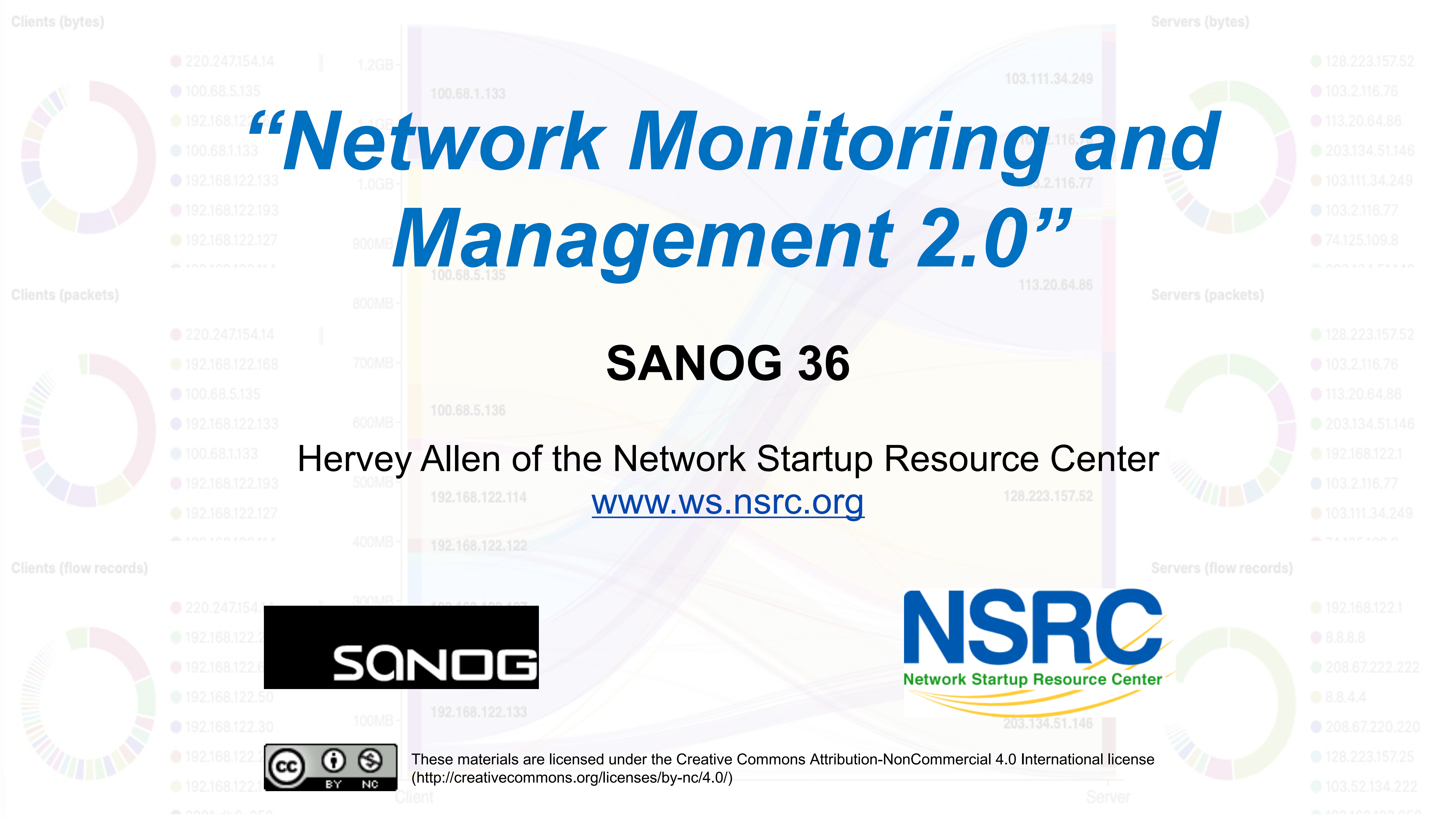


# *"Network Monitoring and Management 2.0"*

## SANOG 36

Hervey Allen of the Network Startup Resource Center
www.ws.nsrc.org

SANOG

NSRC
Network Startup Resource Center

# A few “Walls of Text”

I promise pictures after these initial slides…

# NMM 2.0

Why?

# The Why of NMM 2.0

- Finer-grained metrics (“real time network telemetry”)
  - Network telemetry streams vs. occasional data pulls
- Scaling (hyper scale)
  - Ability to measure monitor hyper-scale projects
  - Polling 10,000 devices/containers… that’s hard
  - Can have operational impact
- Portability:
  - Gather data once, use with multiple tools

# NMM 2.0

How?

# NMM 2.0

## Traditional vs. Present Day Practices

Push vs. Pull or…

Network telemetry / push / passive vs. polling / pull

After this we would start talking about…

Monitoring vs. *Observing* (o11y)

A wonderful discussion at https://twitter.com/isotopp/status/1328653624470331392

# NMM 2.0

## Traditional vs. Present Day Practices*

## Push vs. Pull or…

## Network telemetry / push / passive vs. polling / pull

- Traditional: standards-based like snmp or agents (Nagios, Check MK)
- Present: some push protocols:
  - Cisco compact Google Protocol Buffers
  - Google Protocol Buffers
  - Json
- Newer agents used with present day network monitoring stacks
  - Telegraf, beats, node exporter, Promtail, logstash, etc…

**Sort of… Depends on your needs, resources, goals, etc.*

# NMM 2.0

## Traditional vs. Present Day Practices

How we store our network metrics (NoSQL vs. Relational)

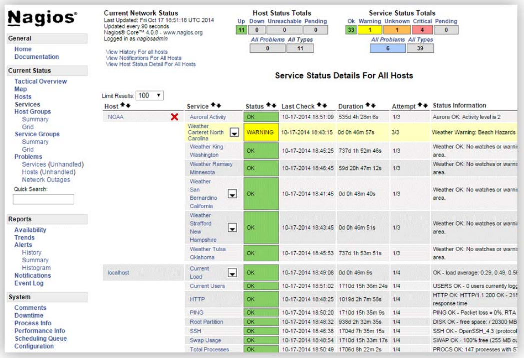
- Traditional: relational data stores for network metrics
  - MySQL, PostgreSQL, SQLite, Oracle, DB2, SQL Server, MariaDB, etc.
- Present: a few time series data stores or NoSQL databases:
  - Cassandra
  - CouchDB
  - ElastiSearch
  - InfluxDB
  - MongoDB
  - Prometheus
  - RRDTool (Old school time series data store! Heavily used.)
  - TimescaleDB

# NMM 2.0

## Traditional vs. Present Day Practices*

### Dashboards vs. Monolithic interfaces to network metrics

- Traditional: Constrained interfaces with less extensibility
  - Nagios
  - Cacti
  - LibreNMS
  - SmokePing



Current Network Status
Last Updated: Fri Oct 17 18:51:18 UTC 2014
Updated every 90 seconds
Nagios® Core™ 4.0.8 - www.nagios.org
Logged in as *nagiosadmin*

Host Status Totals

| Up | Down | Unreachable | Pending |
|---|---|---|---|
| 11 | 0 | 0 | 0 |

| All Problems | All Types |
|---|---|
| 0 | 11 |

Service Status Totals

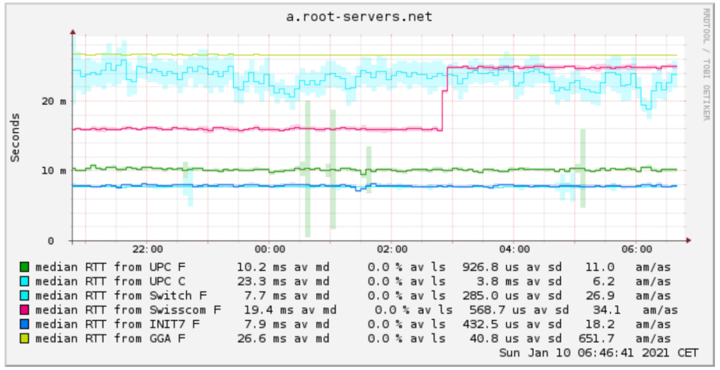
| Ok | Warning | Unknown | Critical | Pending |
|---|---|---|---|---|
| 33 | 1 | 1 | 4 | 0 |

| All Problems | All Types |
|---|---|
| 6 | 39 |

Service Status Details For All Hosts

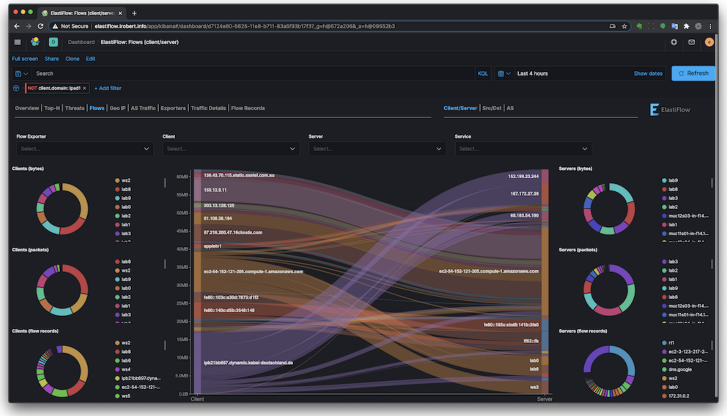
Limit Results: 100

| Host | Service | Status | Last Check | Duration | Attempt | Status Information |
|---|---|---|---|---|---|---|
| NOAA | Auroral Activity | OK | 10-17-2014 18:51:09 | 535d 4h 28m 6s | 1/3 | Aurora OK: Activity level is 2 |
| | Weather Carteret North Carolina | WARNING | 10-17-2014 18:43:15 | 0d 0h 46m 57s | 3/3 | Weather Warning: Beach Hazards |
| | Weather King Washington | OK | 10-17-2014 18:45:25 | 737d 1h 52m 46s | 1/3 | Weather OK: No watches or warni area. |
| | Weather Ramsey Minnesota | OK | 10-17-2014 18:46:45 | 59d 20h 47m 12s | 1/3 | Weather OK: No watches or warni area. |
| | Weather San Bernardino California | OK | 10-17-2014 18:41:45 | 0d 0h 48m 40s | 1/3 | Weather OK: No watches or warni area. |
| | Weather Strafford New Hampshire | OK | 10-17-2014 18:43:45 | 0d 0h 46m 51s | 1/3 | Weather OK: No watches or warni area. |
| | Weather Tulsa Oklahoma | OK | 10-17-2014 18:45:53 | 737d 1h 53m 51s | 1/3 | Weather OK: No watches or warni area. |
| localhost | Current Load | OK | 10-17-2014 18:49:08 | 0d 0h 46m 9s | 1/4 | OK - load average: 0.29, 0.49, 0.5 |
| | Current Users | OK | 10-17-2014 18:51:02 | 1710d 15h 36m 24s | 1/4 | USERS OK - 0 users currently logg |
| | HTTP | OK | 10-17-2014 18:48:25 | 1019d 2h 7m 58s | 1/4 | HTTP OK: HTTP/1.1 200 OK - 21 response time |
| | PING | OK | 10-17-2014 18:50:20 | 1710d 15h 35m 9s | 1/4 | PING OK - Packet loss = 0%, RTA |
| | Root Partition | OK | 10-17-2014 18:48:32 | 938d 2h 32m 35s | 1/4 | DISK OK - free space: / 20300 MB |
| | SSH | OK | 10-17-2014 18:46:38 | 1704d 7h 35m 15s | 1/4 | SSH OK - OpenSSH_4.3 (protocol |
| | Swap Usage | OK | 10-17-2014 18:48:54 | 1710d 15h 33m 17s | 1/4 | SWAP OK - 100% free (255 MB ou |
| | Total Processes | OK | 10-17-2014 18:50:49 | 1706d 8h 22m 2s | 1/4 | PROCS OK: 147 processes with S |



- Present: Dashboards massively configurable, harder to get started (for some)
  - Chronograf, Grafana, Kibana*
    - *Elastiflow: a flow collection tool that use Kibana and Elasticsearch with preconfigured dashboards

➔


ElastiFlow: Flows (client/server)
Not Secure | elastiflow.irobert.info/app/kibana#/dashboard/d7124e80-5625-11e8-b711-83a5f93b17f3?_g=h@572a206&_a=h@09552b3
Dashboard / ElastiFlow: Flows (client/server)
Full screen Share Clone Edit
Search
KQL
Last 4 hours
Show dates
Refresh
NOT client.domain: ipad1
+ Add filter
Overview | Top-N | Threats | Flows | Geo IP | AS Traffic | Exporters | Traffic Details | Flow Records
Client/Server | Src/Dst | AS
ElastiFlow
Flow Exporter
Client
Server
Service
Select...
Clients (bytes)
Clients (packets)
Clients (flow records)
Servers (bytes)
Servers (packets)
Servers (flow records)
138.43.70.115.static.exetel.com.au
155.12.9.11
203.13.128.125
61.108.38.194
67.216.200.47.16clouds.com
152.199.23.244
167.172.27.39
68.183.54.190
ws2
lab8
lab9
lab0
lab2
lab1
lab3
Client
Server

# NMM 2.0

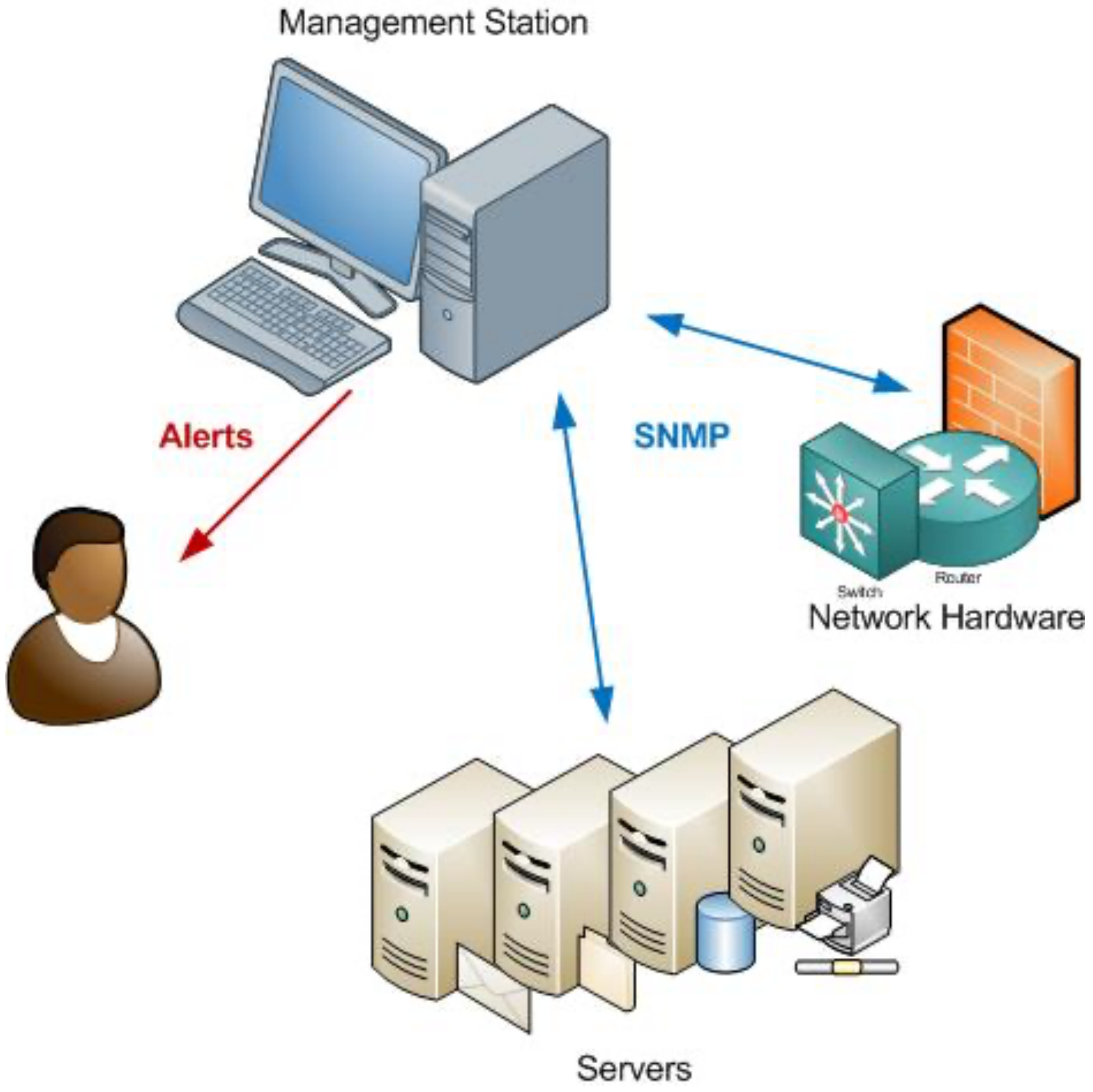
## Traditional vs. Present Day Practices

### Alerting

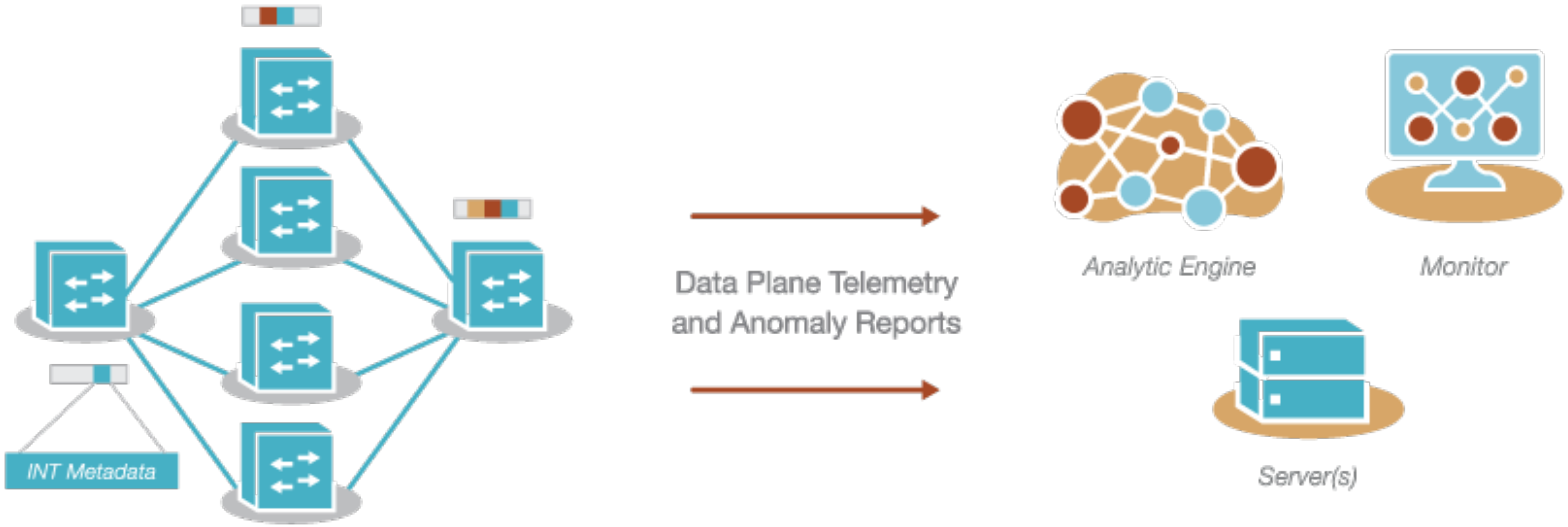
- Traditional: If available, built-in to the tool. Often minimal.
  - *SmokePing*: alerts.cfg with custom regex language
  - *Nagios*: template based. Very well implemented.
  - *Cacti*: plugins required. Variable.
  - *LibreNMS*: built-in. Not intuitive. Improving over time.
- Present: Often a separate tool or built-in to dashboard tool
  - *AlertManager* (Prometheus solution)
  - *Grafana* (visualizer/analyzer)
  - *Kapacitor* (TICK Stack)
  - *Kibana* (ELK Stack)

Stacks: ELK, TICK, Prometheus. We’ll get to these! ☺

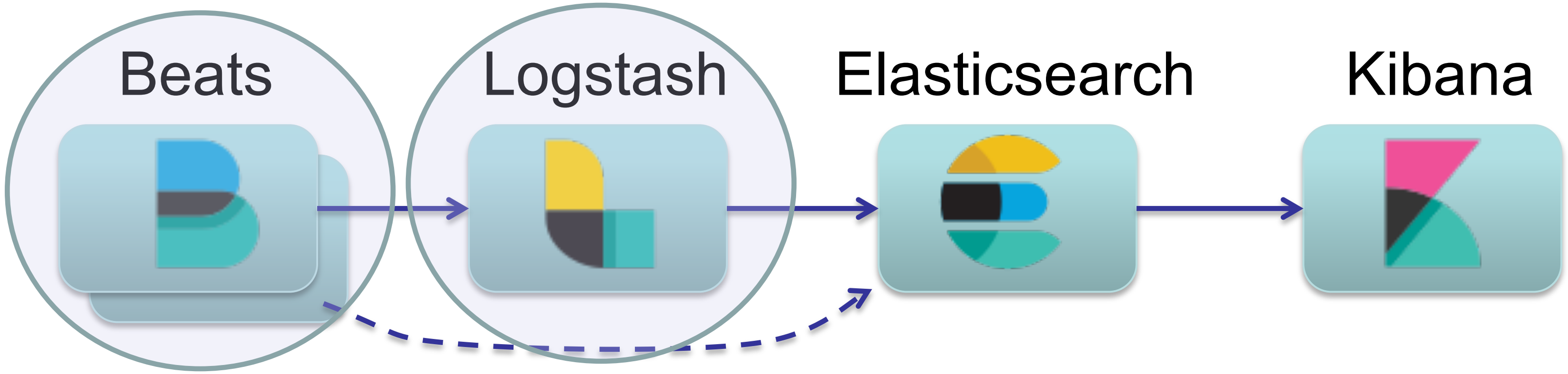
# Classical Polling Model



# “Network Telemetry” or “Push Model”



# The Elastic Stack (ELK)

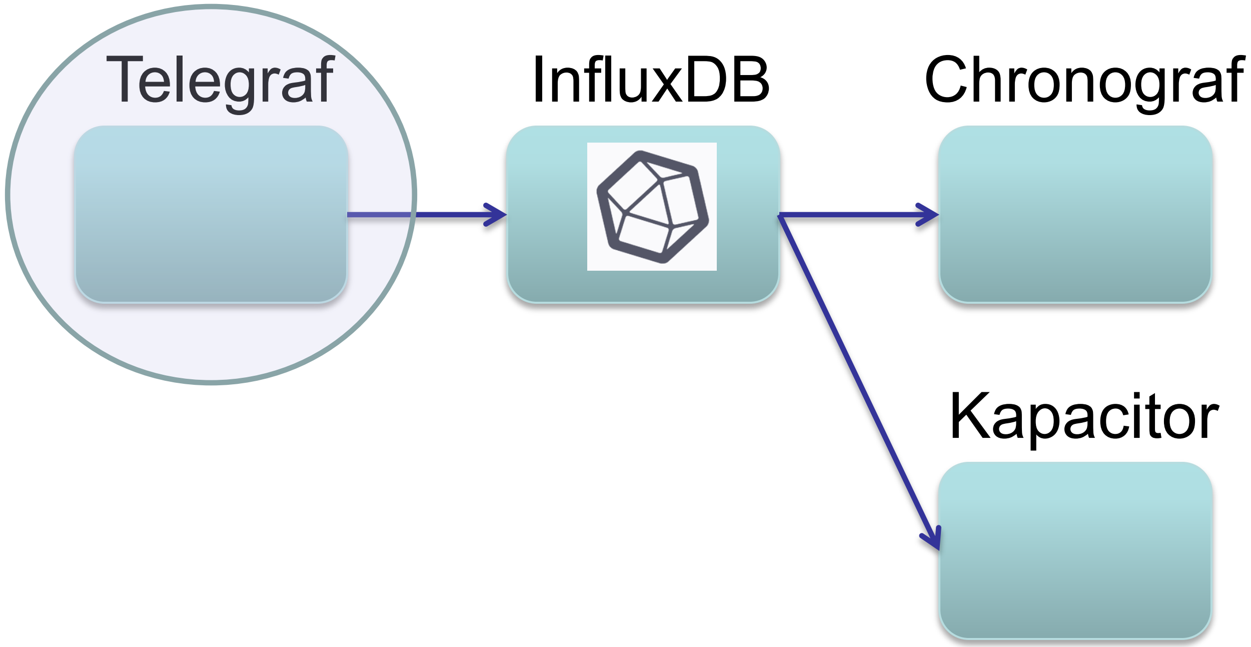


*Present day network measurement "Stacks" are a group of software components that work together to form a monitoring and management solution.*

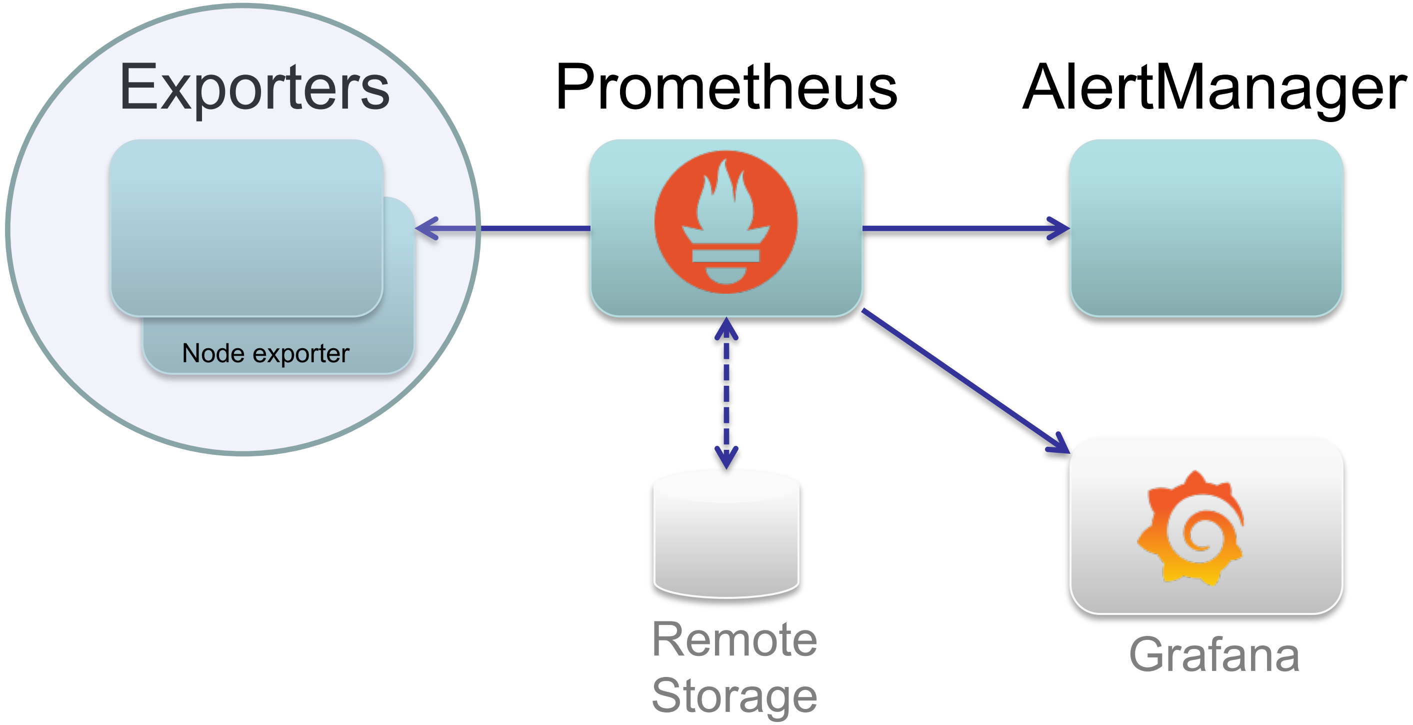
Typical stacks include (more or less):

- Mechanism(s) to push data to a data store (agents, protocols, both)
- A time series or NoSQL data store
- An engine to query the data store and present results in a graphical format in a dashboard format.
- A built-in or separate alerting component that works with the data store
- Note that many components are interchangeable between stacks

# The TICK Stack



# Prometheus


Exporters
Node exporter
Prometheus
AlertManager
Remote Storage
Grafana

# Typical Relational Store (MySQL)

```
  CREATE TABLE `device_metrics` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `timestamp` int(11) NOT NULL,
  `metric1` smallint(6) NOT NULL,
  `metric2` int NOT NULL,
  `metric3` float NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`),
  UNIQUE KEY `idposition_UNIQUE` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

# What this looks like

```
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| id        | int(11)      | NO   | PRI | NULL    | auto_increment |
| timestamp | int(11)      | NO   |     | NULL    |                |
| metric1   | smallint(6)  | NO   |     | NULL    |                |
| metric2   | int(11)      | NO   |     | NULL    |                |
| metric3   | float        | NO   |     | 0       |                |
+-----------+--------------+------+-----+---------+----------------+
```

This is moderately efficent vs. putting every metric in to a different table. But, you still only get one data set per row.

# What this looks like with inserted data

```
SELECT * FROM device_metrics;
```

```
+----+-------------+----------+-------------+----------+
| id | timestamp   | metric1  | metric2     | metric3  |
+----+-------------+----------+-------------+----------+
|  1 | 1610232093  |    29001 | 1800789199  |    79.86 |
|  2 | 1610232094  |    29002 | 1800789200  |    79.98 |
|  3 | 1610232095  |    29003 | 1800789201  |    77.67 |
|  4 | 1610232065  |    29004 | 1800789223  |    78.32 |
|  5 | 1610232097  |    29077 | 1800789456  |    80.01 |
|  6 | 1610232098  |    29232 | 1800723455  |    79.11 |
+----+-------------+----------+-------------+----------+
```

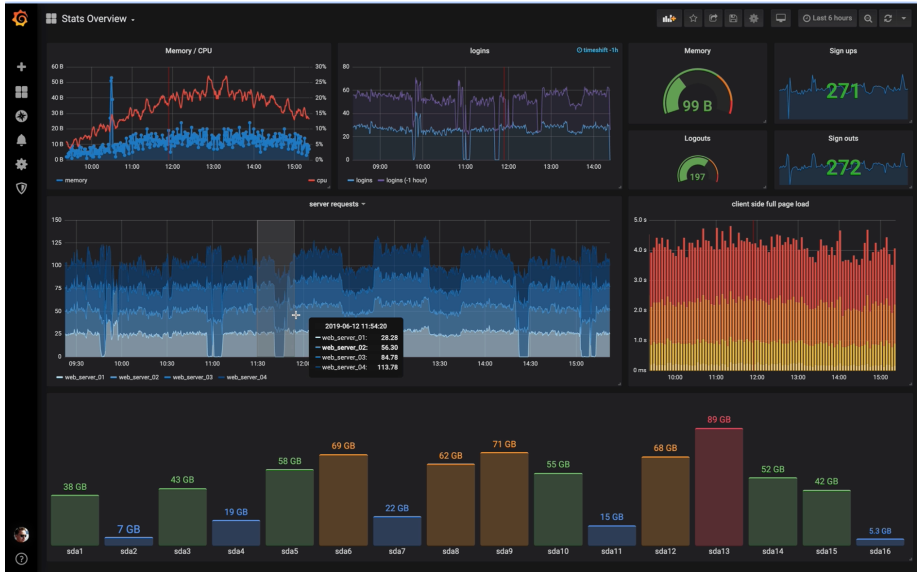
# Table Growth

```
+----+------------+---------+------------+---------+
| id | timestamp  | metric1 | metric2    | metric3 |
+----+------------+---------+------------+---------+
|  1 | 1610232093 |   29001 | 1800789199 |   79.86 |
|  2 | 1610232094 |   29002 | 1800789200 |   79.98 |
|  3 | 1610232095 |   29003 | 1800789201 |   77.67 |
|  4 | 1610232065 |   29004 | 1800789223 |   78.32 |
|  5 | 1610232097 |   29077 | 1800789456 |   80.01 |
|  6 | 1610232098 |   29232 | 1800723455 |   79.11 |
+----+------------+---------+------------+---------+
```

🡐 A new data point every second!

- With “push” model and agents much more telemetry data.
- Querying and displaying large numbers of metrics become inefficent in a relational model.

How to get to this? ➔
(Grafana)



# Inefficiencies of relations…

Inserting, Updating and Selecting, or…

- Adding data
- Changing data
- Getting data

Each row increases

- Index size
- Compute

*NoSQL / Time Series data stores allow for very large sets of metrics in sequence and ability to query these metrics at large scale*

# Time series data stores / NoSQL

A few ways to store time series data (there are many):

- `timestamp, metric, timestamp, metric`

  or
- `timestamp, metric, metric, …, timestamp, metric, metric, …`

  or
- `metric, metric, metric, metric, metric, … timestamp`

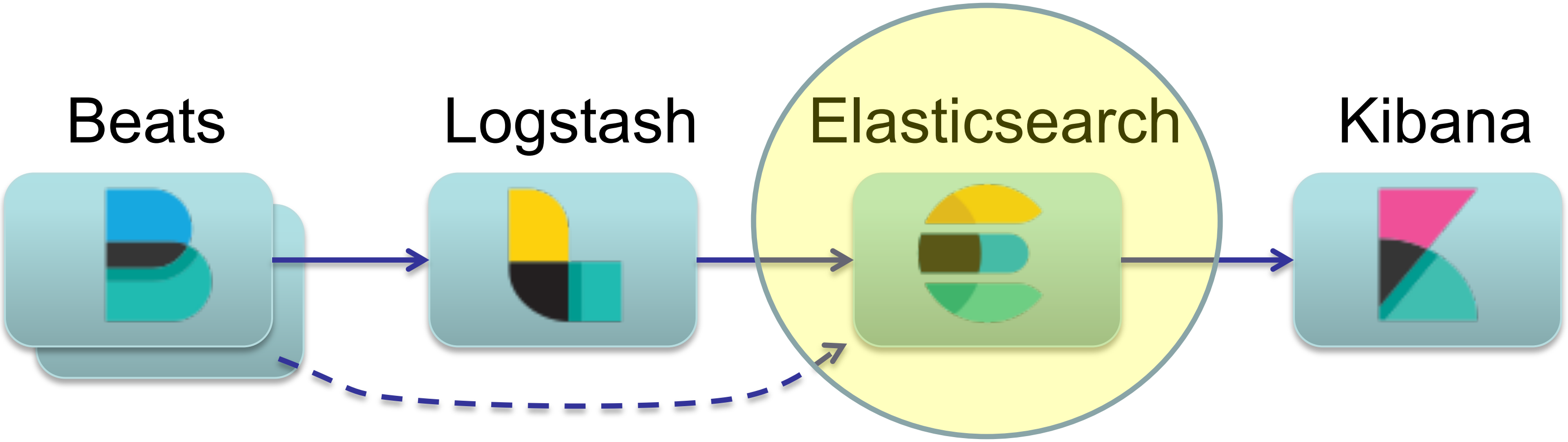
Per row. Each row can have *many* columns.

- For example, Cassandra DB can support up to 2 billion columns per row!
- Nice discussion on what is time series data: https://www.influxdata.com/what-is-time-series-data/

# NMM 2.0

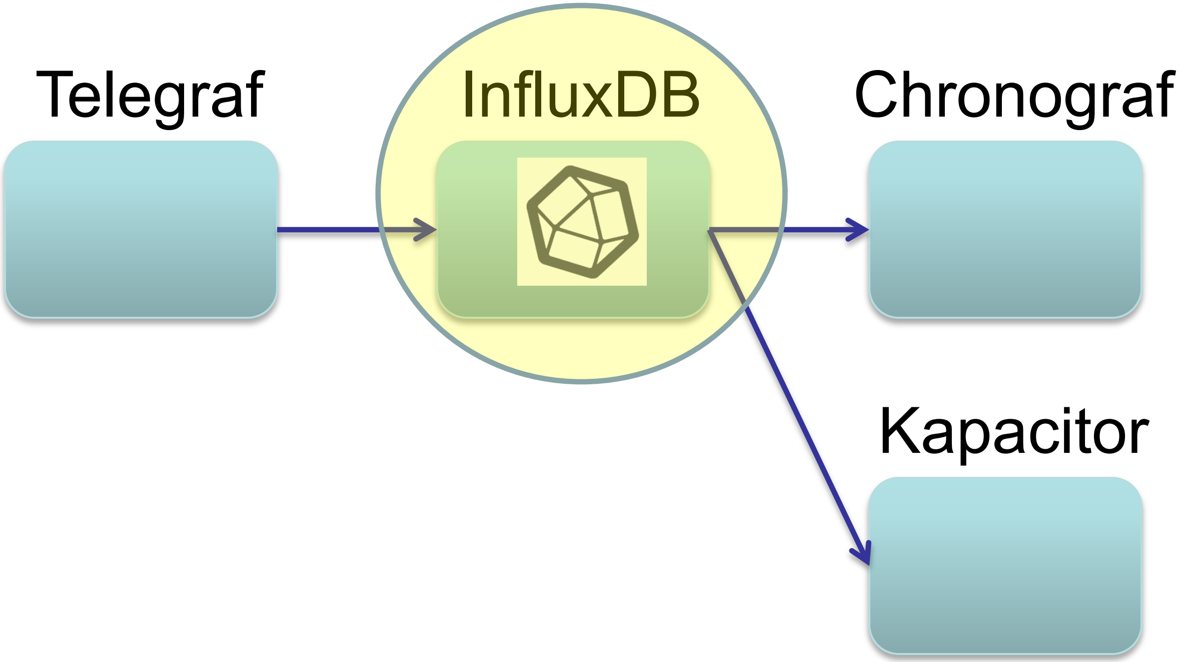
## The Datastores

# The Elastic Stack (ELK)

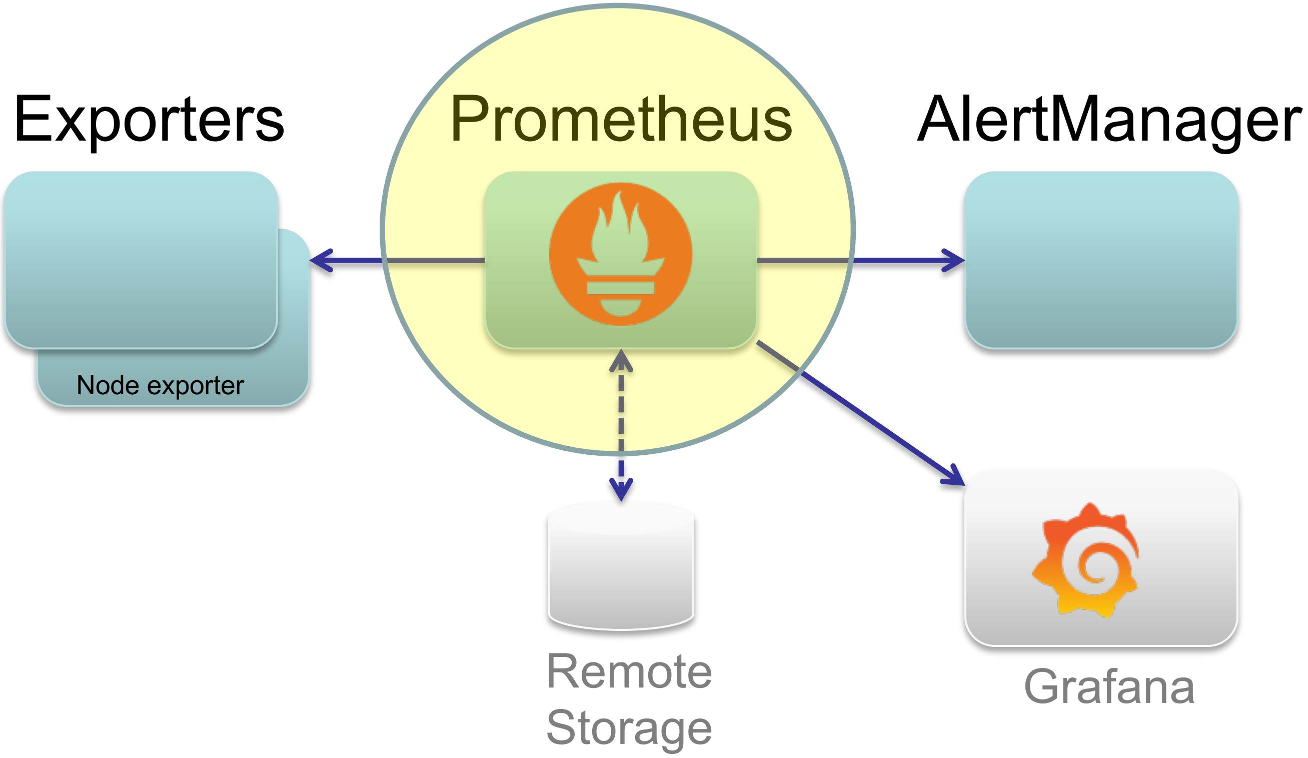


*("The BLEK Stack" doesn't sound as good)*

# The TICK Stack



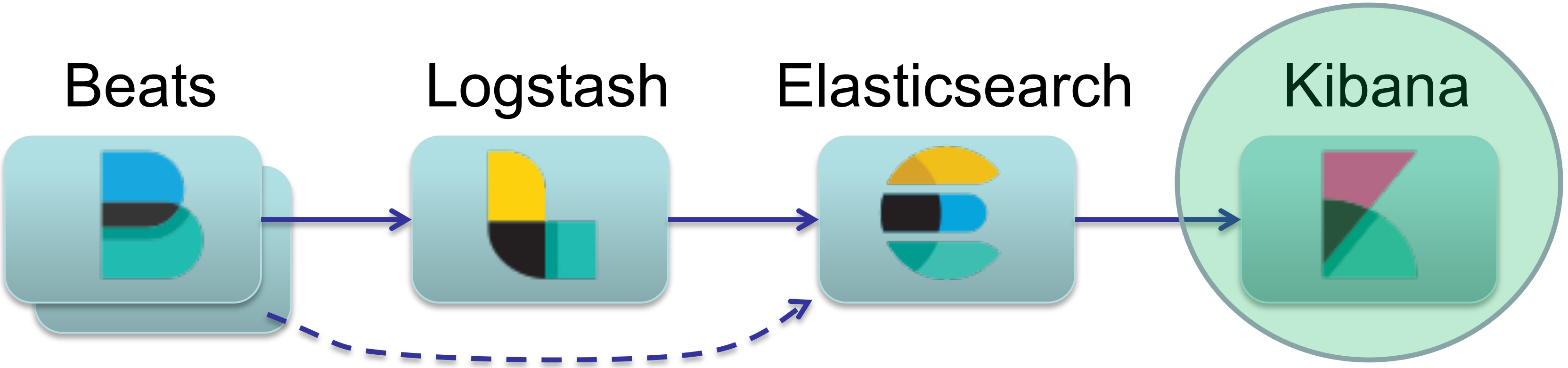
# Prometheus


Exporters
Node exporter
Prometheus
AlertManager
Remote Storage
Grafana

# NMM 2.0

## The Dashboards

# The Elastic Stack (ELK)



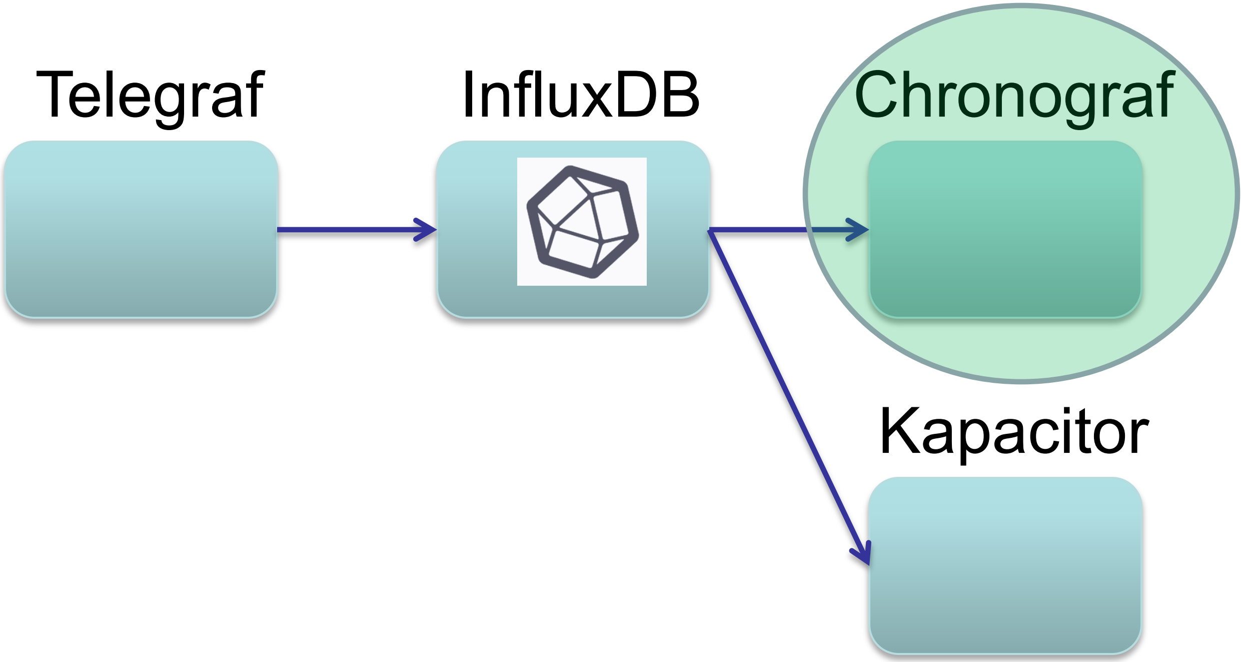
**Not sure whether to use Logstash or Beats?**

Beats are lightweight data shippers that you install as agents on your servers to send specific types of operational data to Elasticsearch. Beats have a small footprint and use fewer system resources than Logstash.

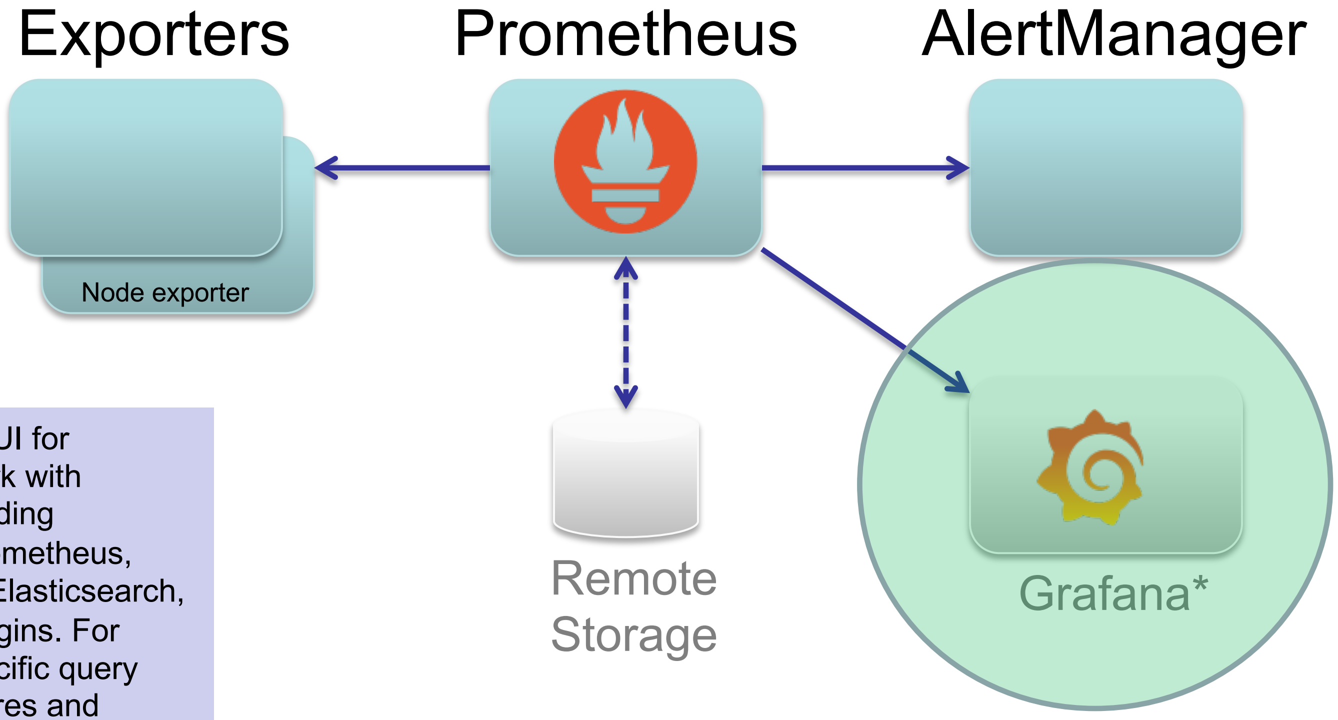
Logstash has a larger footprint, but provides a broad array of input, filter, and output plugins for collecting, enriching, and transforming data from a variety of sources.

https://www.elastic.co/guide/en/beats/filebeat/current/diff-logstash-beats.html

# The TICK Stack



# Prometheus

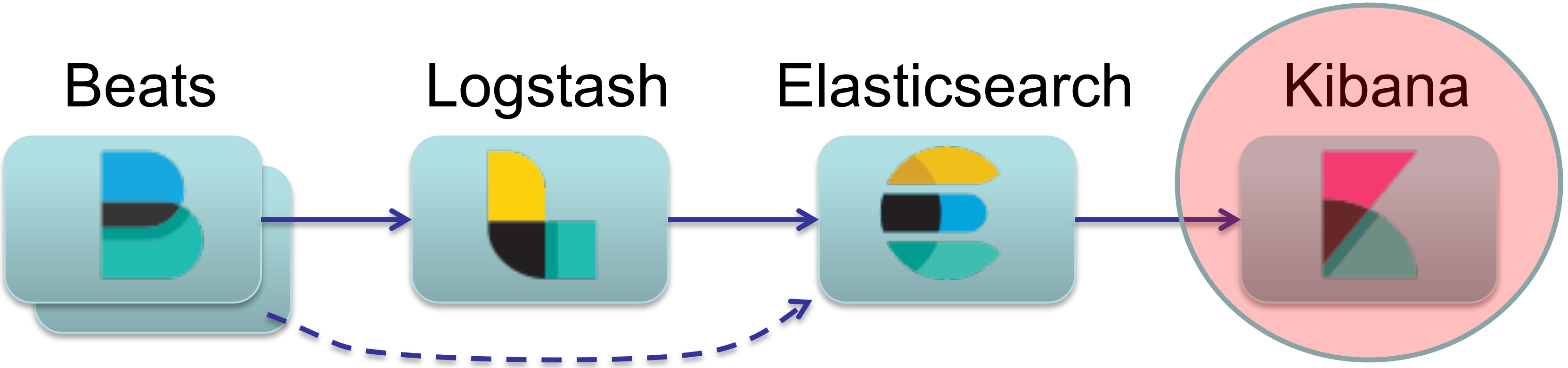


*Grafana was designed to work as a UI for analyzing metrics. As such, it can work with multiple time-series data stores, including built-in integrations with Graphite, Prometheus, InfluxDB, MySQL, PostgreSQL, and Elasticsearch, and additional data sources using plugins. For each data source, Grafana has a specific query editor that is customized for the features and capabilities that are included in that data source (https://logz.io/blog/grafana-vs-kibana/).

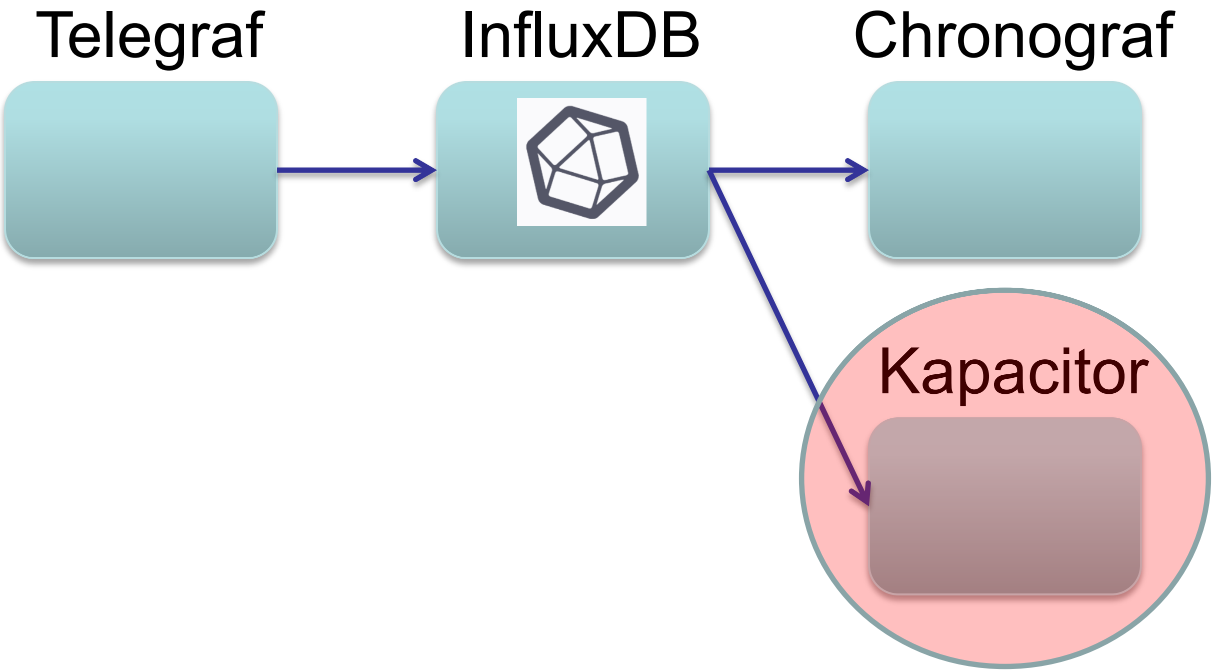
# NMM 2.0

## Alerting

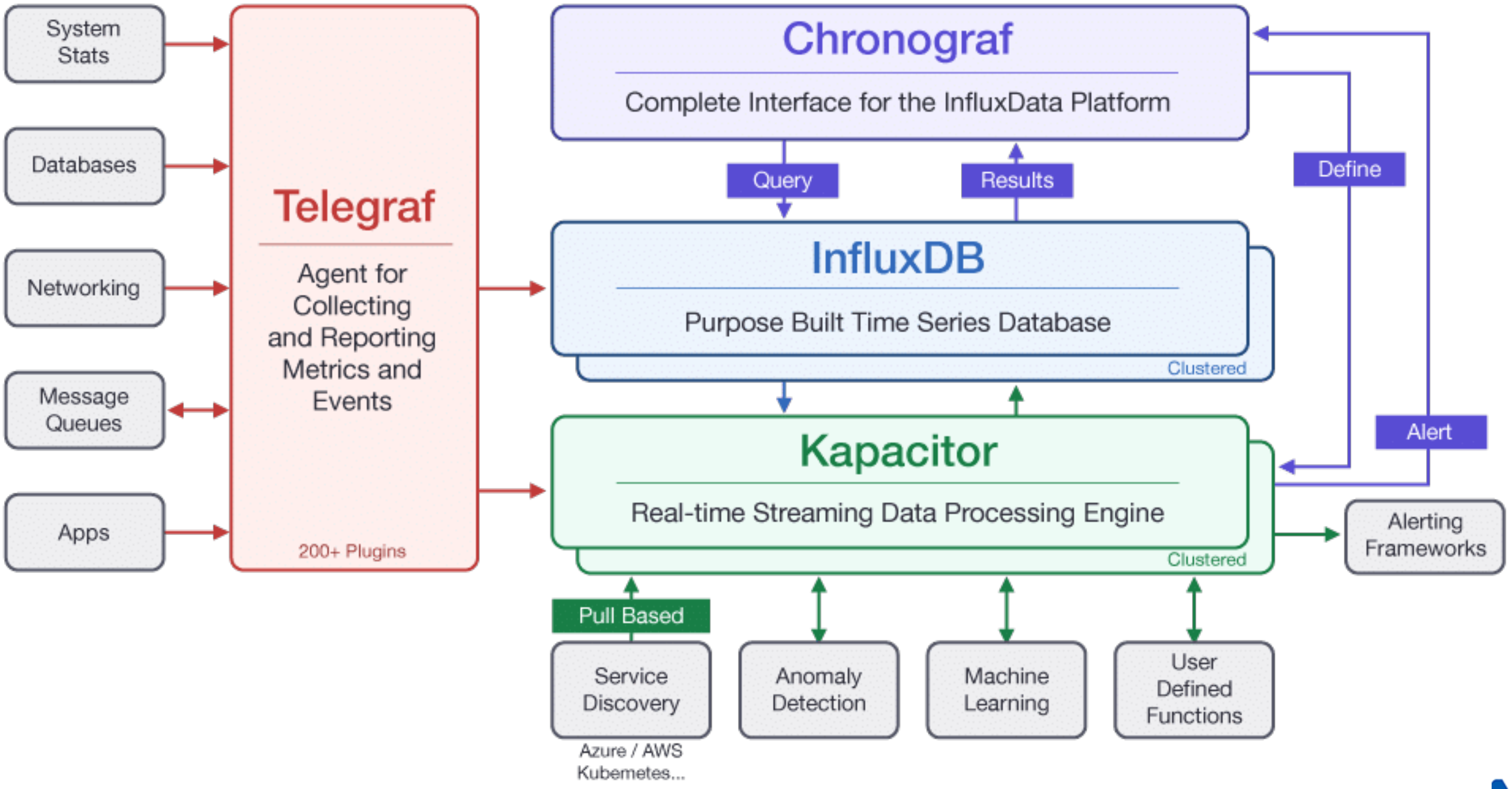
# The Elastic Stack (ELK)



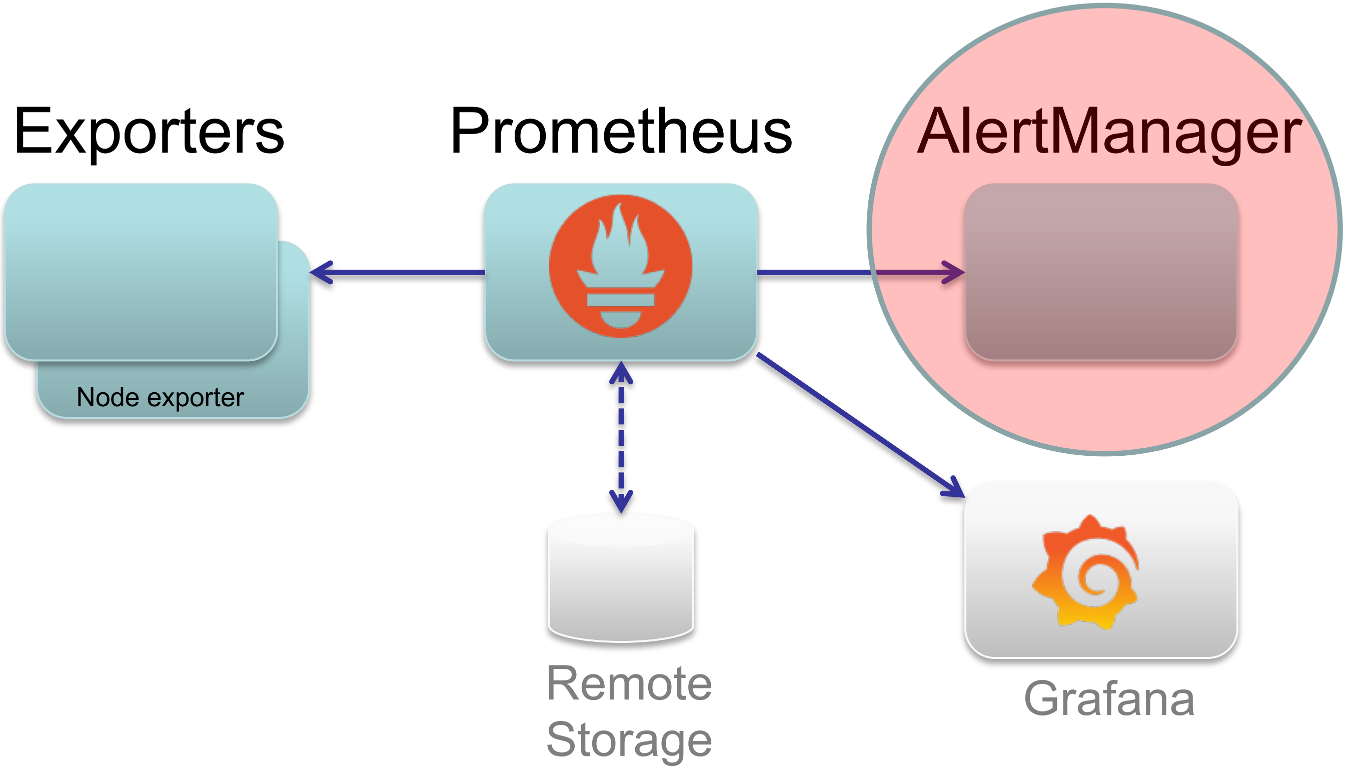
# The TICK Stack



# TICK stack detail

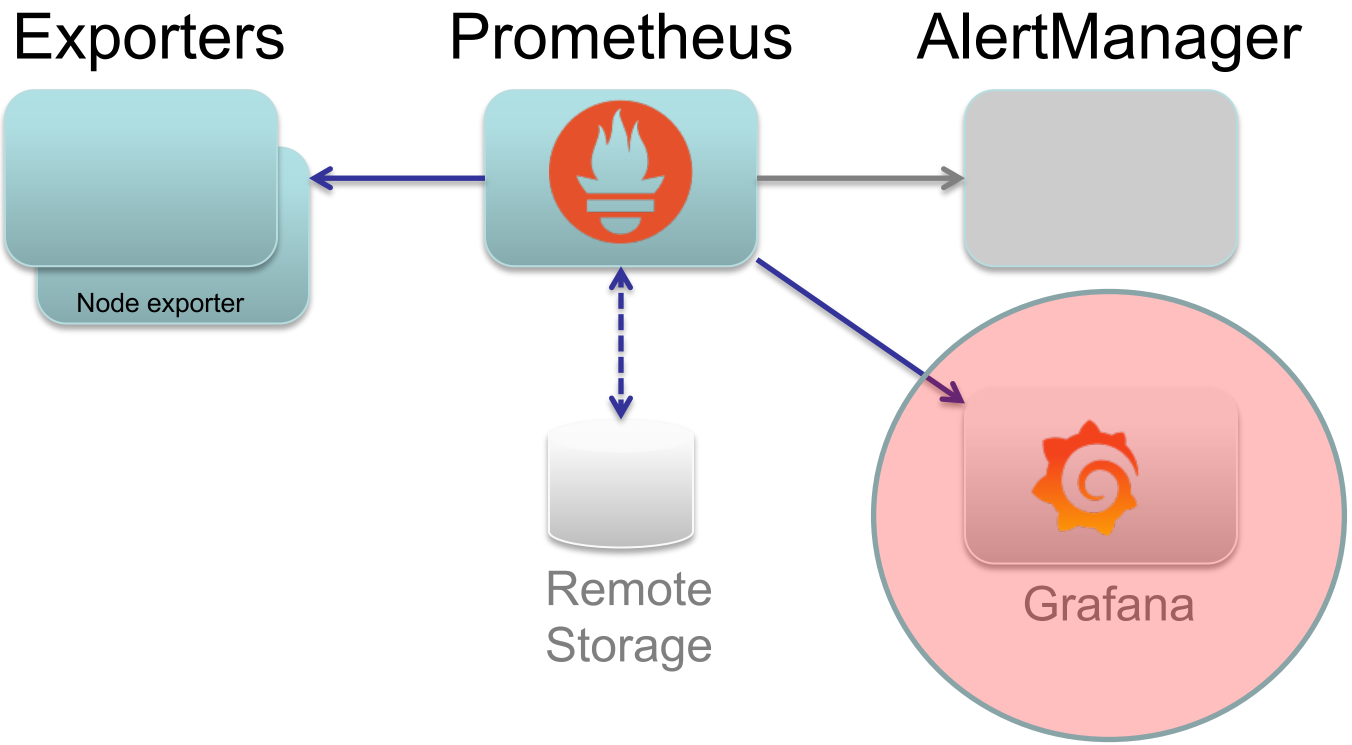


# Prometheus



Or… ➔

# Prometheus

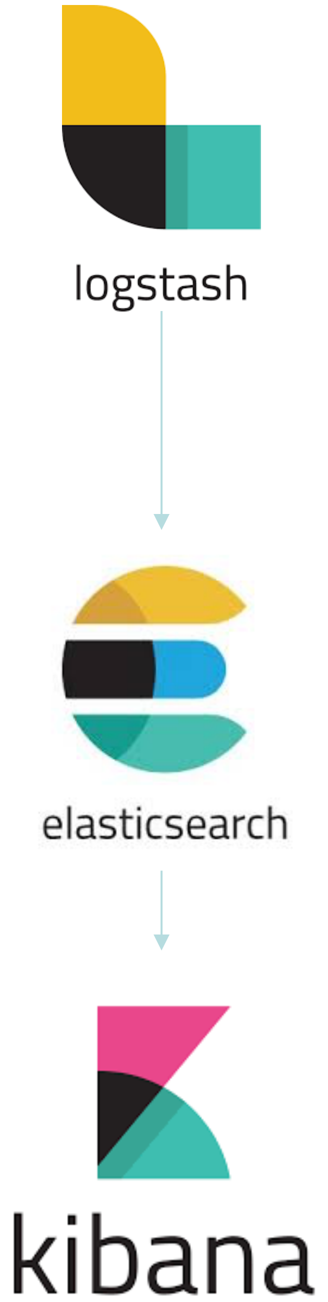

Exporters
Prometheus
AlertManager
Node exporter
Remote Storage
Grafana

# NMM 2.0

## Traditional vs. Present Day Practices

### Putting it all together

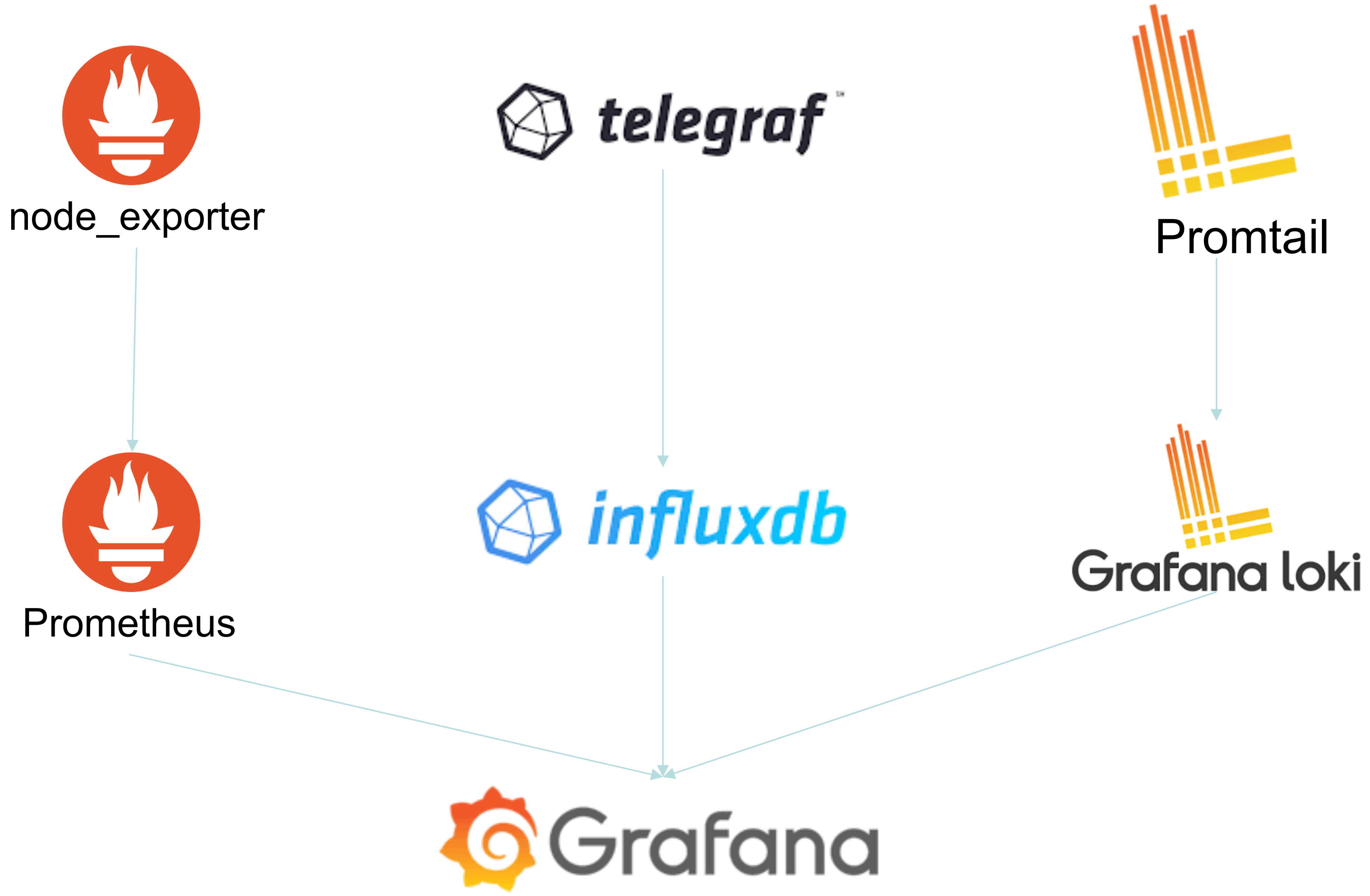
- Presentation of data often requires more resources
  - Disk and CPU
  - Fine-grained telemetry (seconds or less vs. minutes) == more data on disk
  - Large data stores and complex dashboards can == more CPU
- Regex knowledge
  - You figure out what you want to know (some preconfigured dashboards as well)
  - Stack Based. Multiple software projects working together



logstash

elasticsearch

kibana



node_exporter

Prometheus

telegraf

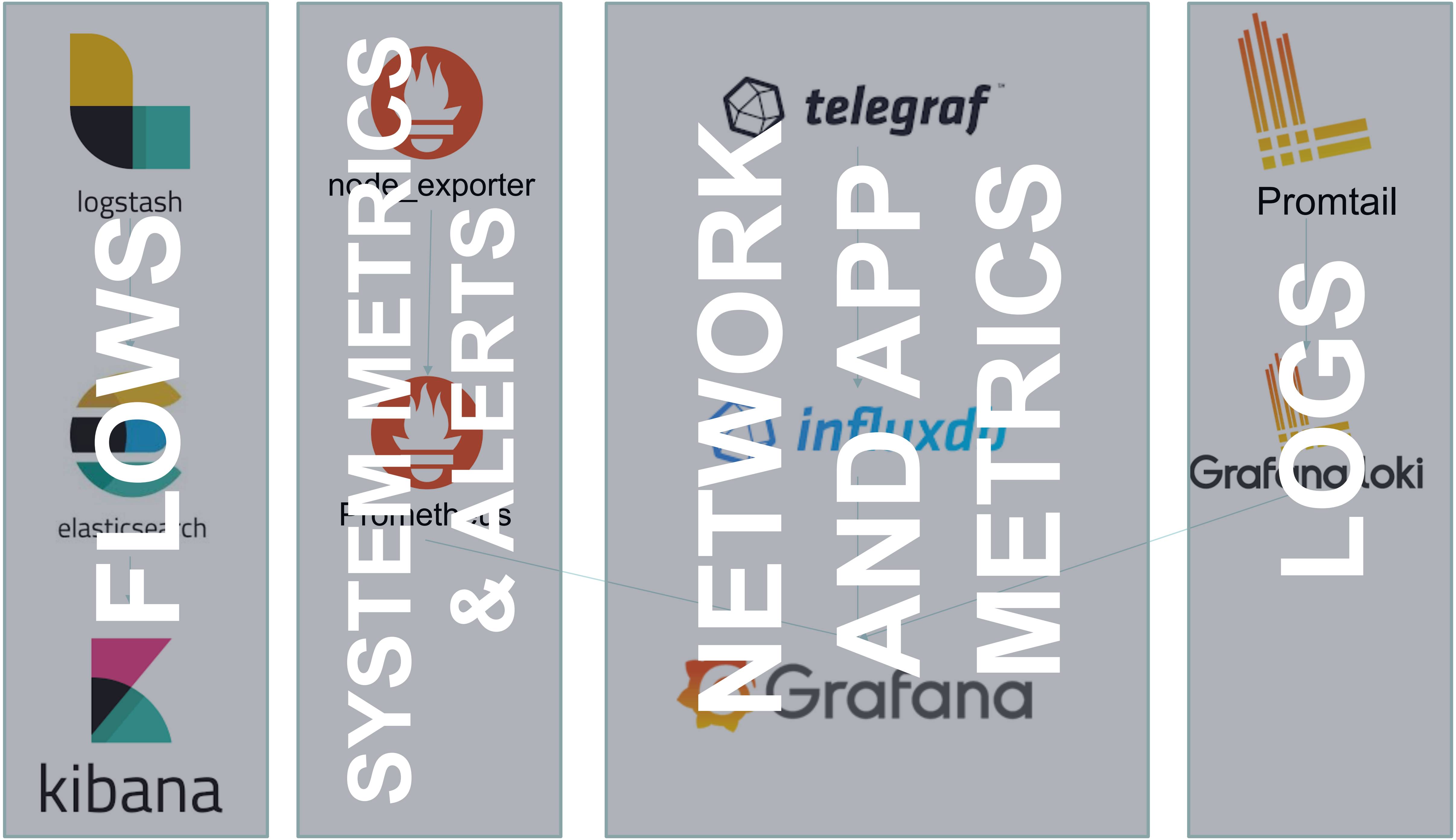
influxdb

Promtail

Grafana loki

Grafana

Thank you Dean Pemberton for the next 7 slides

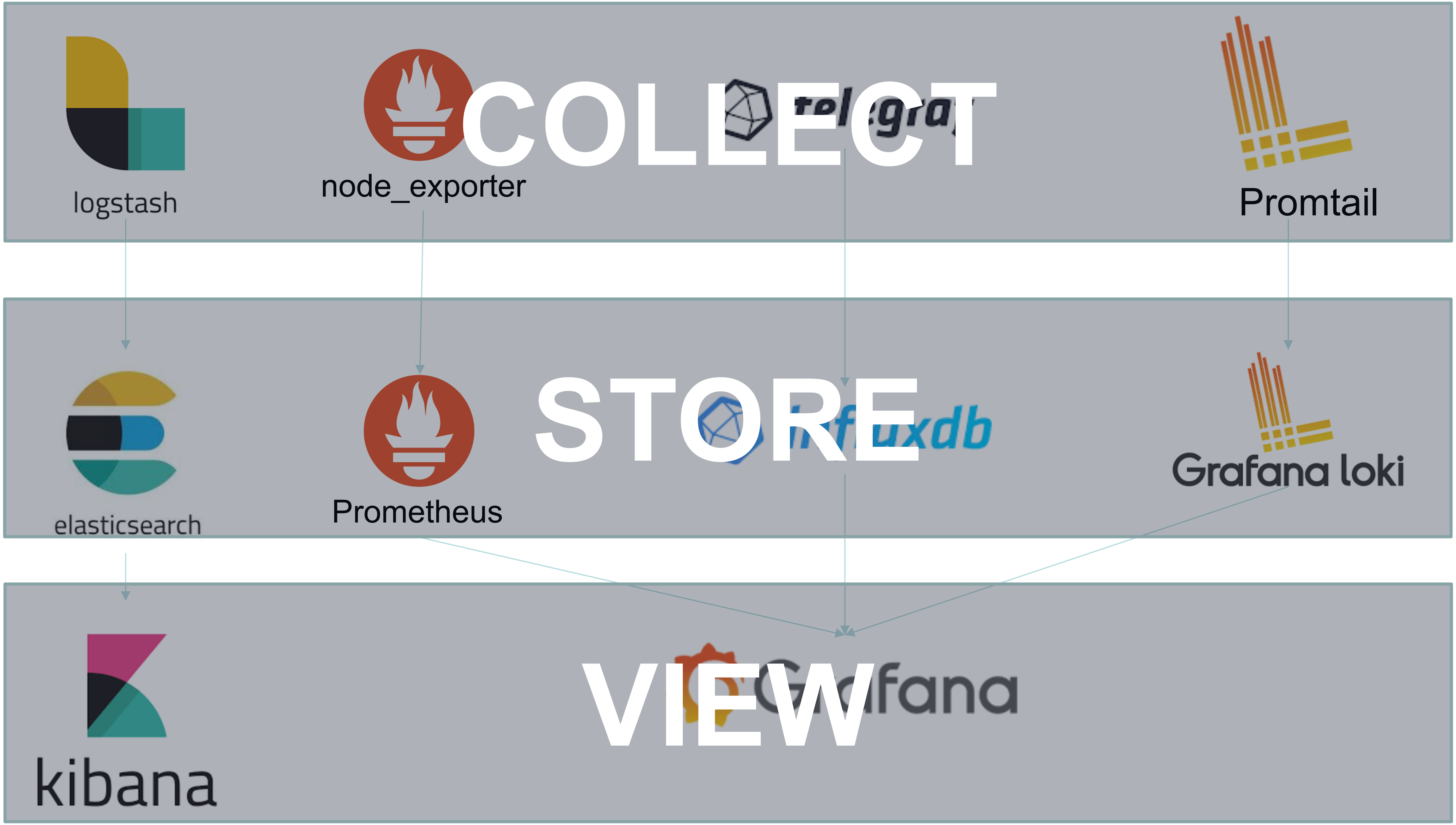


FLOWS

SYSTEM METRICS & ALERTS

NETWORK AND APP METRICS

LOGS



COLLECT

STORE

VIEW

logstash

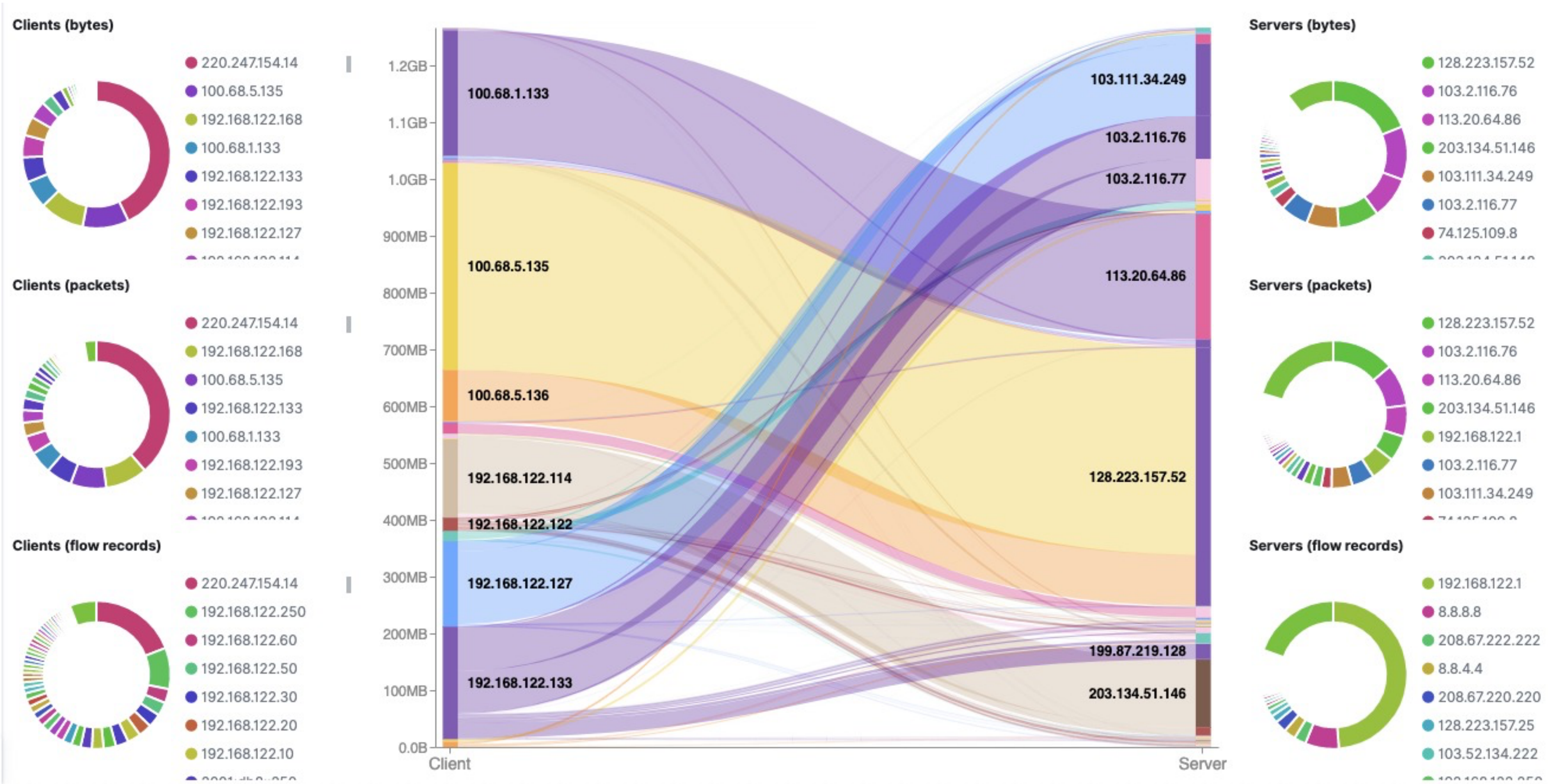
elasticsearch

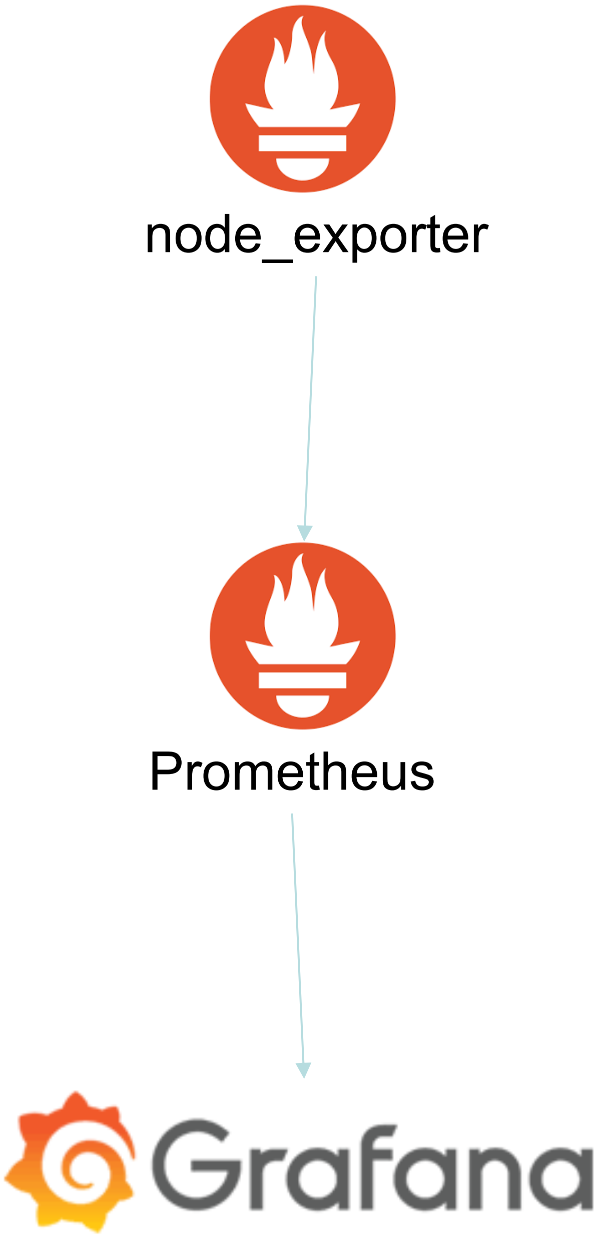
kibana

ElastiFlow

## Takes the following flow protocols

- ✓ Netflow
- ✓ IPFix
- ✓ SFlow



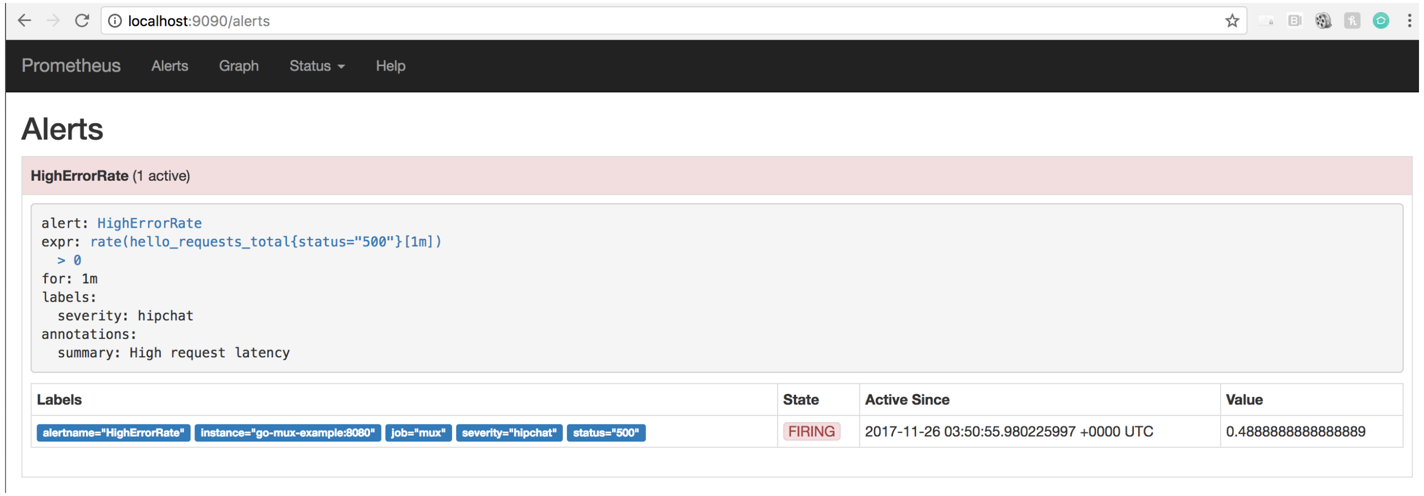


node_exporter

Prometheus

Grafana

# Generate alerts for reachability and metrics

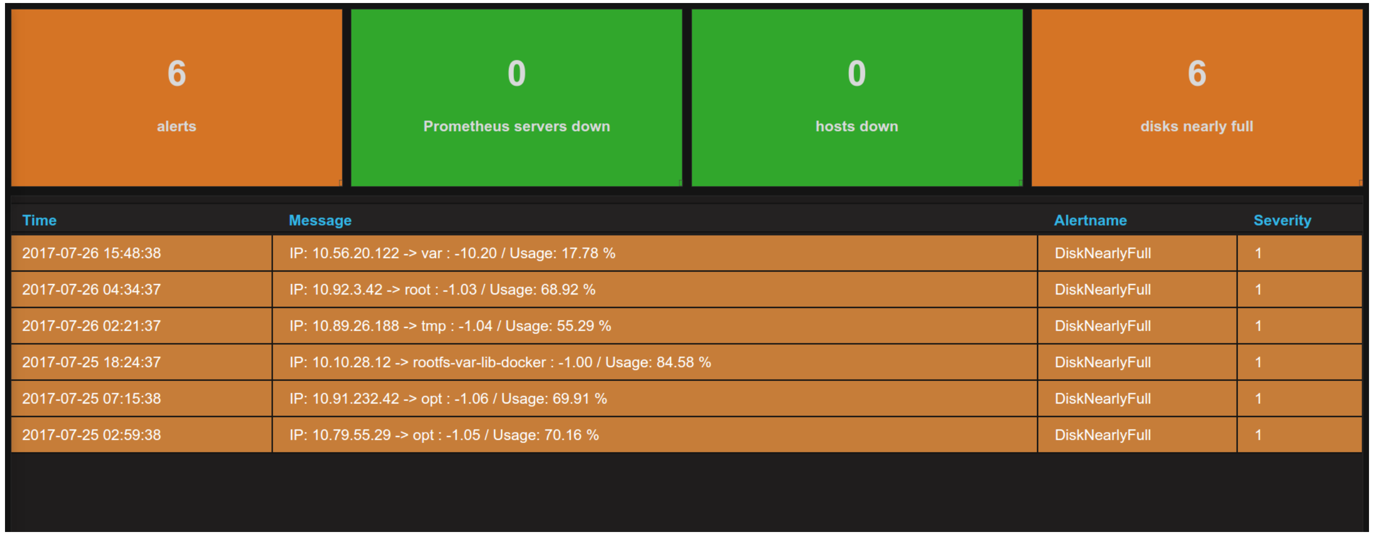


## Alerts

**HighErrorRate** (1 active)

```
alert: HighErrorRate
expr: rate(hello_requests_total{status="500"}[1m])
  > 0
for: 1m
labels:
  severity: hipchat
annotations:
  summary: High request latency
```

| Labels | State | Active Since | Value |
|---|---|---|---|
| alertname="HighErrorRate" instance="go-mux-example:8080" job="mux" severity="hipchat" status="500" | FIRING | 2017-11-26 03:50:55.980225997 +0000 UTC | 0.4888888888888889 |



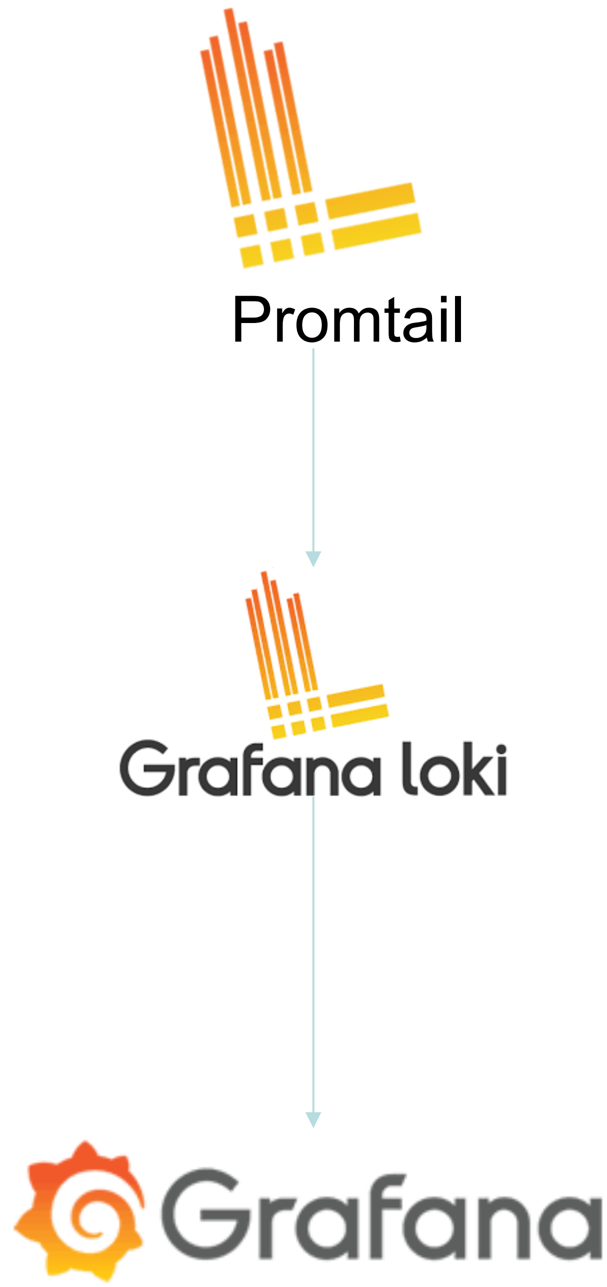
6 alerts

0 Prometheus servers down

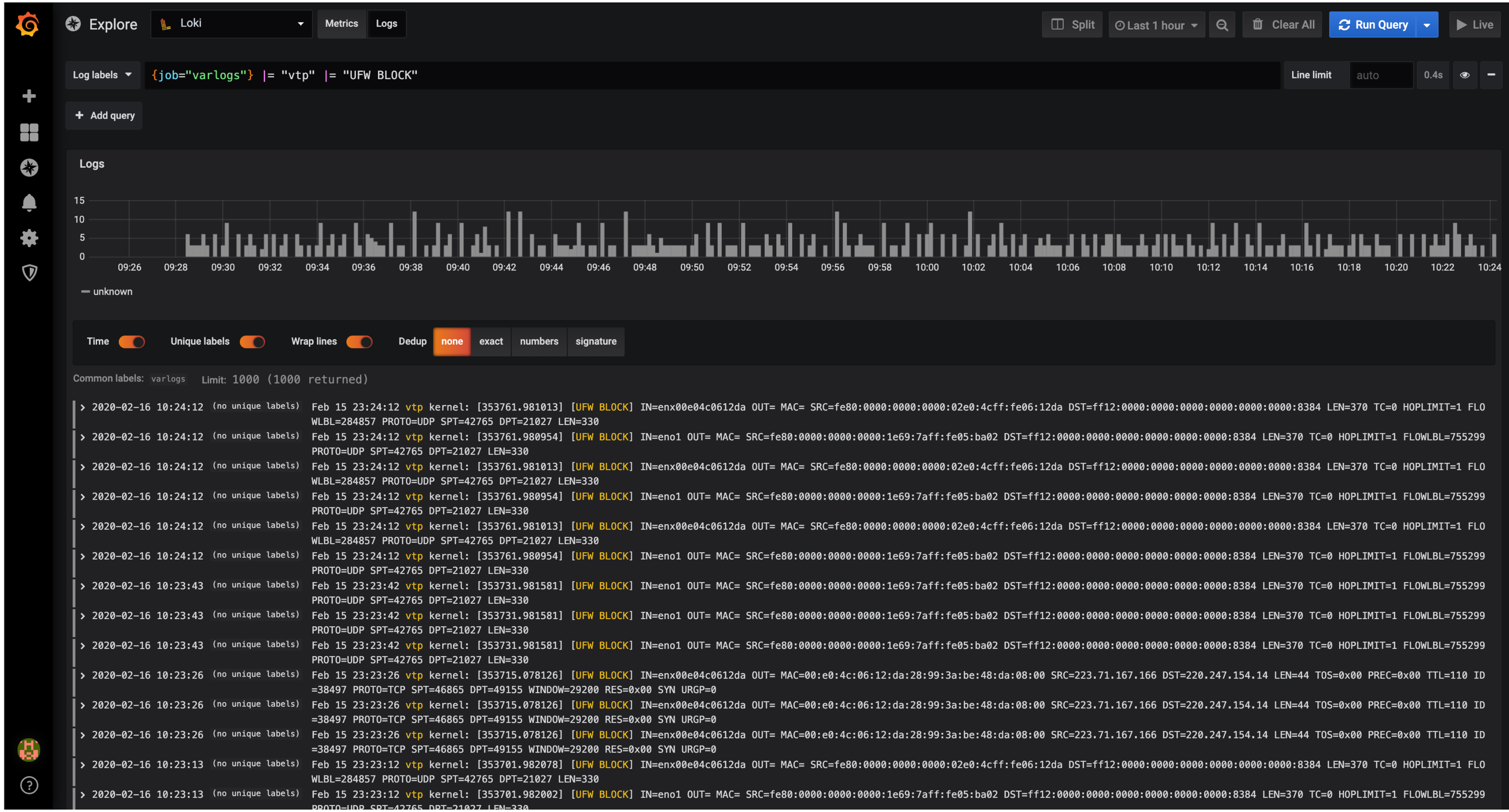
0 hosts down

6 disks nearly full

| Time | Message | Alertname | Severity |
|---|---|---|---|
| 2017-07-26 15:48:38 | IP: 10.56.20.122 -> var : -10.20 / Usage: 17.78 % | DiskNearlyFull | 1 |
| 2017-07-26 04:34:37 | IP: 10.92.3.42 -> root : -1.03 / Usage: 68.92 % | DiskNearlyFull | 1 |
| 2017-07-26 02:21:37 | IP: 10.89.26.188 -> tmp : -1.04 / Usage: 55.29 % | DiskNearlyFull | 1 |
| 2017-07-25 18:24:37 | IP: 10.10.28.12 -> rootfs-var-lib-docker : -1.00 / Usage: 84.58 % | DiskNearlyFull | 1 |
| 2017-07-25 07:15:38 | IP: 10.91.232.42 -> opt : -1.06 / Usage: 69.91 % | DiskNearlyFull | 1 |
| 2017-07-25 02:59:38 | IP: 10.79.55.29 -> opt : -1.05 / Usage: 70.16 % | DiskNearlyFull | 1 |



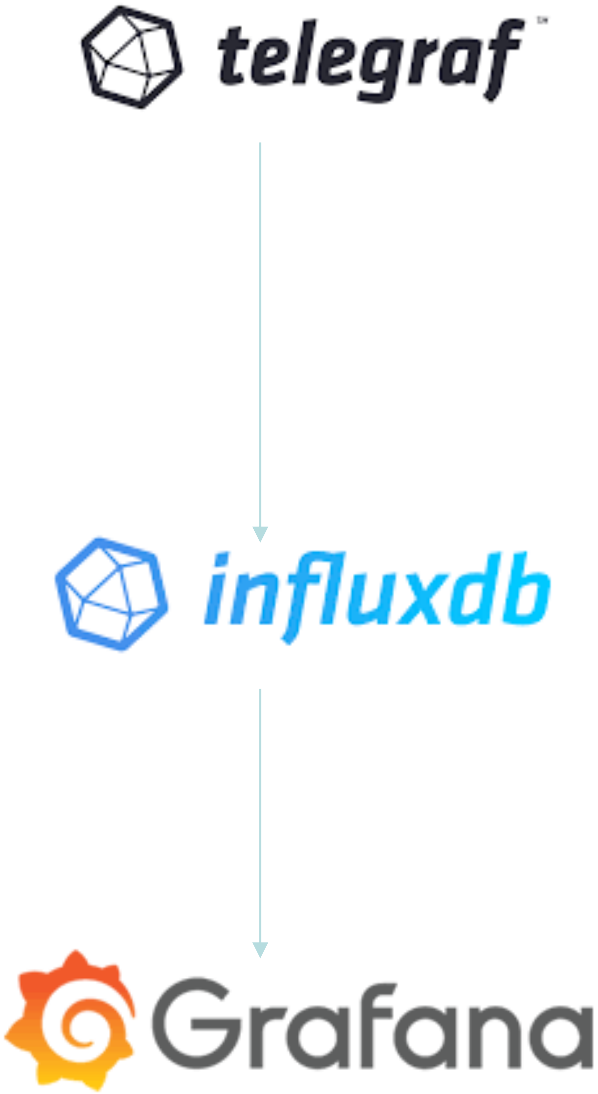
- Streaming logs from files
- Works with Prometheus
- Kubernetes build available



```
{job="varlogs"} |= "vtp" |= "UFW BLOCK"
```

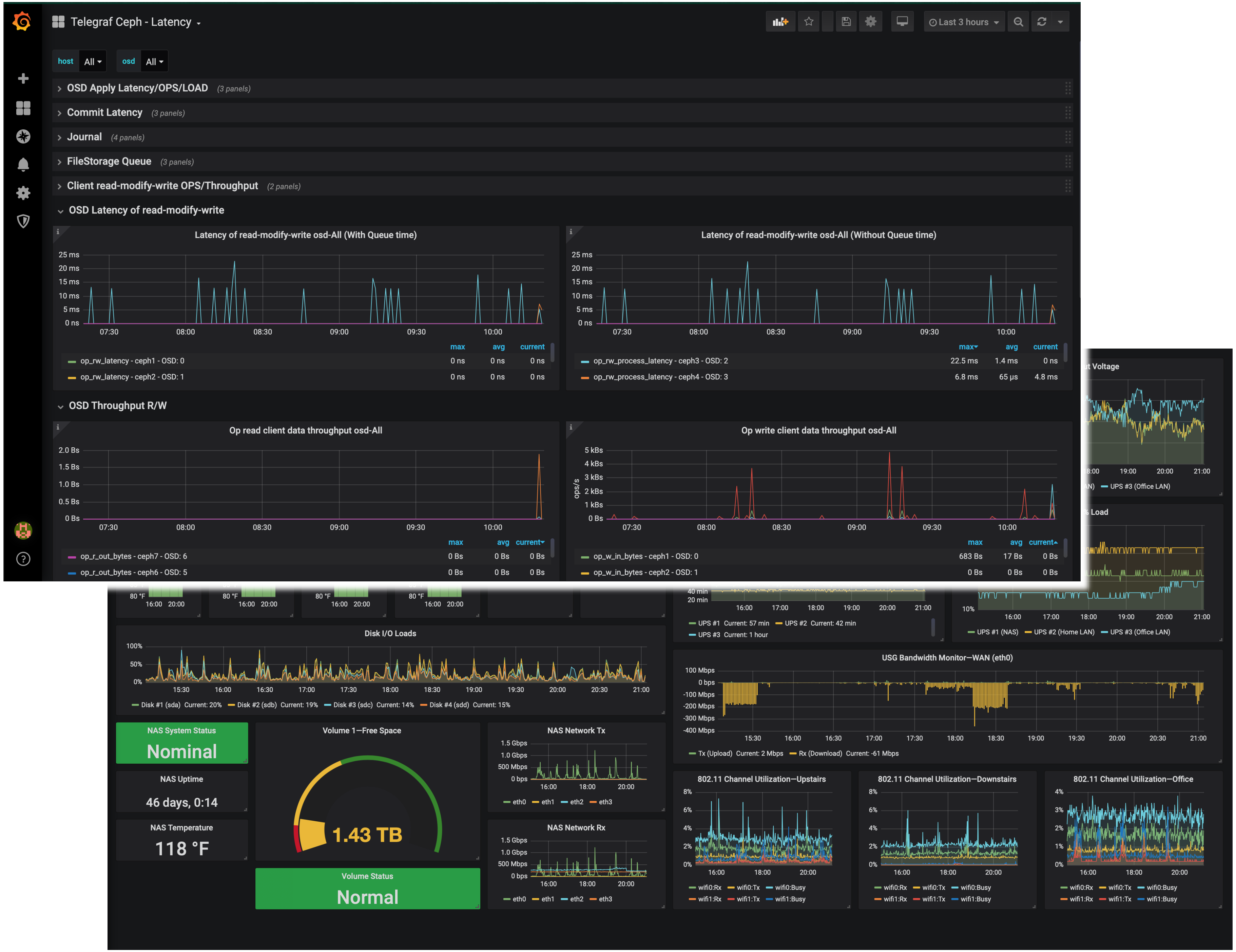
```
Common labels: varlogs   Limit: 1000 (1000 returned)

> 2020-02-16 10:24:12 (no unique labels) Feb 15 23:24:12 vtp kernel: [353761.981013] [UFW BLOCK] IN=enx00e04c0612da OUT= MAC= SRC=fe80:0000:0000:0000:02e0:4cff:fe06:12da DST=ff12:0000:0000:0000:0000:0000:0000:8384 LEN=370 TC=0 HOPLIMIT=1 FLO
                                          WLBL=284857 PROTO=UDP SPT=42765 DPT=21027 LEN=330
> 2020-02-16 10:24:12 (no unique labels) Feb 15 23:24:12 vtp kernel: [353761.980954] [UFW BLOCK] IN=eno1 OUT= MAC= SRC=fe80:0000:0000:0000:1e69:7aff:fe05:ba02 DST=ff12:0000:0000:0000:0000:0000:0000:8384 LEN=370 TC=0 HOPLIMIT=1 FLOWLBL=755299
                                          PROTO=UDP SPT=42765 DPT=21027 LEN=330
> 2020-02-16 10:24:12 (no unique labels) Feb 15 23:24:12 vtp kernel: [353761.981013] [UFW BLOCK] IN=enx00e04c0612da OUT= MAC= SRC=fe80:0000:0000:0000:02e0:4cff:fe06:12da DST=ff12:0000:0000:0000:0000:0000:0000:8384 LEN=370 TC=0 HOPLIMIT=1 FLO
                                          WLBL=284857 PROTO=UDP SPT=42765 DPT=21027 LEN=330
> 2020-02-16 10:24:12 (no unique labels) Feb 15 23:24:12 vtp kernel: [353761.980954] [UFW BLOCK] IN=eno1 OUT= MAC= SRC=fe80:0000:0000:0000:1e69:7aff:fe05:ba02 DST=ff12:0000:0000:0000:0000:0000:0000:8384 LEN=370 TC=0 HOPLIMIT=1 FLOWLBL=755299
                                          PROTO=UDP SPT=42765 DPT=21027 LEN=330
> 2020-02-16 10:24:12 (no unique labels) Feb 15 23:24:12 vtp kernel: [353761.981013] [UFW BLOCK] IN=enx00e04c0612da OUT= MAC= SRC=fe80:0000:0000:0000:02e0:4cff:fe06:12da DST=ff12:0000:0000:0000:0000:0000:0000:8384 LEN=370 TC=0 HOPLIMIT=1 FLO
                                          WLBL=284857 PROTO=UDP SPT=42765 DPT=21027 LEN=330
> 2020-02-16 10:24:12 (no unique labels) Feb 15 23:24:12 vtp kernel: [353761.980954] [UFW BLOCK] IN=eno1 OUT= MAC= SRC=fe80:0000:0000:0000:1e69:7aff:fe05:ba02 DST=ff12:0000:0000:0000:0000:0000:0000:8384 LEN=370 TC=0 HOPLIMIT=1 FLOWLBL=755299
                                          PROTO=UDP SPT=42765 DPT=21027 LEN=330
> 2020-02-16 10:23:43 (no unique labels) Feb 15 23:23:42 vtp kernel: [353731.981581] [UFW BLOCK] IN=eno1 OUT= MAC= SRC=fe80:0000:0000:0000:1e69:7aff:fe05:ba02 DST=ff12:0000:0000:0000:0000:0000:0000:8384 LEN=370 TC=0 HOPLIMIT=1 FLOWLBL=755299
                                          PROTO=UDP SPT=42765 DPT=21027 LEN=330
> 2020-02-16 10:23:43 (no unique labels) Feb 15 23:23:42 vtp kernel: [353731.981581] [UFW BLOCK] IN=eno1 OUT= MAC= SRC=fe80:0000:0000:0000:1e69:7aff:fe05:ba02 DST=ff12:0000:0000:0000:0000:0000:0000:8384 LEN=370 TC=0 HOPLIMIT=1 FLOWLBL=755299
                                          PROTO=UDP SPT=42765 DPT=21027 LEN=330
> 2020-02-16 10:23:43 (no unique labels) Feb 15 23:23:42 vtp kernel: [353731.981581] [UFW BLOCK] IN=eno1 OUT= MAC= SRC=fe80:0000:0000:0000:1e69:7aff:fe05:ba02 DST=ff12:0000:0000:0000:0000:0000:0000:8384 LEN=370 TC=0 HOPLIMIT=1 FLOWLBL=755299
                                          PROTO=UDP SPT=42765 DPT=21027 LEN=330
> 2020-02-16 10:23:26 (no unique labels) Feb 15 23:23:26 vtp kernel: [353715.078126] [UFW BLOCK] IN=enx00e04c0612da OUT= MAC=00:e0:4c:06:12:da:28:99:3a:be:48:da:08:00 SRC=223.71.167.166 DST=220.247.154.14 LEN=44 TOS=0x00 PREC=0x00 TTL=110 ID
                                          =38497 PROTO=TCP SPT=46865 DPT=49155 WINDOW=29200 RES=0x00 SYN URGP=0
> 2020-02-16 10:23:26 (no unique labels) Feb 15 23:23:26 vtp kernel: [353715.078126] [UFW BLOCK] IN=enx00e04c0612da OUT= MAC=00:e0:4c:06:12:da:28:99:3a:be:48:da:08:00 SRC=223.71.167.166 DST=220.247.154.14 LEN=44 TOS=0x00 PREC=0x00 TTL=110 ID
                                          =38497 PROTO=TCP SPT=46865 DPT=49155 WINDOW=29200 RES=0x00 SYN URGP=0
> 2020-02-16 10:23:26 (no unique labels) Feb 15 23:23:26 vtp kernel: [353715.078126] [UFW BLOCK] IN=enx00e04c0612da OUT= MAC=00:e0:4c:06:12:da:28:99:3a:be:48:da:08:00 SRC=223.71.167.166 DST=220.247.154.14 LEN=44 TOS=0x00 PREC=0x00 TTL=110 ID
                                          =38497 PROTO=TCP SPT=46865 DPT=49155 WINDOW=29200 RES=0x00 SYN URGP=0
> 2020-02-16 10:23:13 (no unique labels) Feb 15 23:23:12 vtp kernel: [353701.982078] [UFW BLOCK] IN=enx00e04c0612da OUT= MAC= SRC=fe80:0000:0000:0000:02e0:4cff:fe06:12da DST=ff12:0000:0000:0000:0000:0000:0000:8384 LEN=370 TC=0 HOPLIMIT=1 FLO
                                          WLBL=284857 PROTO=UDP SPT=42765 DPT=21027 LEN=330
> 2020-02-16 10:23:13 (no unique labels) Feb 15 23:23:12 vtp kernel: [353701.982002] [UFW BLOCK] IN=eno1 OUT= MAC= SRC=fe80:0000:0000:0000:1e69:7aff:fe05:ba02 DST=ff12:0000:0000:0000:0000:0000:0000:8384 LEN=370 TC=0 HOPLIMIT=1 FLOWLBL=755299
                                          PROTO=UDP SPT=42765 DPT=21027 LEN=330
```


telegraf

influxdb

Grafana


Telegraf Ceph - Latency

host All osd All

- OSD Apply Latency/OPS/LOAD (3 panels)
- Commit Latency (3 panels)
- Journal (4 panels)
- FileStorage Queue (3 panels)
- Client read-modify-write OPS/Throughput (2 panels)

OSD Latency of read-modify-write

Latency of read-modify-write osd-All (With Queue time)

| | max | avg | current |
|---|---|---|---|
| op_rw_latency - ceph1 - OSD: 0 | 0 ns | 0 ns | 0 ns |
| op_rw_latency - ceph2 - OSD: 1 | 0 ns | 0 ns | 0 ns |

Latency of read-modify-write osd-All (Without Queue time)

| | max | avg | current |
|---|---|---|---|
| op_rw_process_latency - ceph3 - OSD: 2 | 22.5 ms | 1.4 ms | 0 ns |
| op_rw_process_latency - ceph4 - OSD: 3 | 6.8 ms | 65 µs | 4.8 ms |

OSD Throughput R/W

Op read client data throughput osd-All

| | max | avg | current |
|---|---|---|---|
| op_r_out_bytes - ceph7 - OSD: 6 | 0 Bs | 0 Bs | 0 Bs |
| op_r_out_bytes - ceph6 - OSD: 5 | 0 Bs | 0 Bs | 0 Bs |

Op write client data throughput osd-All

| | max | avg | current |
|---|---|---|---|
| op_w_in_bytes - ceph1 - OSD: 0 | 683 Bs | 17 Bs | 0 Bs |
| op_w_in_bytes - ceph2 - OSD: 1 | 0 Bs | 0 Bs | 0 Bs |

Disk I/O Loads

Disk #1 (sda) Current: 20% Disk #2 (sdb) Current: 19% Disk #3 (sdc) Current: 14% Disk #4 (sdd) Current: 15%

NAS System Status
Nominal

NAS Uptime
46 days, 0:14

NAS Temperature
118 °F

Volume 1—Free Space
1.43 TB

Volume Status
Normal

NAS Network Tx

NAS Network Rx

UPS #1 Current: 57 min UPS #2 Current: 42 min
UPS #3 Current: 1 hour

Voltage

Load

USG Bandwidth Monitor—WAN (eth0)

Tx (Upload) Current: 2 Mbps Rx (Download) Current: -61 Mbps

802.11 Channel Utilization—Upstairs

802.11 Channel Utilization—Downstairs

802.11 Channel Utilization—Office

# Thanks!

Questions?

# References

- Cisco Telemetry with Google Protocol Buffers https://blogs.cisco.com/sp/streaming-telemetry-with-google-protocol-buffers
- Cisco Model Driven Telemetry https://www.cisco.com/c/en/us/solutions/service-provider/cloud-scale-networking-solutions/model-driven-telemetry.html
- Graphite https://graphiteapp.org/
- InfluxDB https://www.influxdata.com/
- Kafka https://docs.confluent.io/current/streams-ksql.html

# References

- Logz.io (Information on *Elastic Stack*, others)
  https://logz.io/
- Monitoring vs. Observing
  https://twitter.com/isotopp/status/1328653624470331392
- Prometheus
  https://prometheus.io/
- Splunk
  https://www.splunk.com/
- Tick Stack on CentOS
  https://www.digitalocean.com/community/tutorials/how-to-monitor-system-metrics-with-the-tick-stack-on-centos-7
- TimescaleDB
  https://www.timescale.com/

# References from Dean ☺

- to docker-compose stacks
  - https://github.com/robcowart/elastiflow
  - https://github.com/grafana/loki
  - https://github.com/nicolargo/docker-influxdb-grafana
  - https://github.com/vegasbrianc/prometheus
- Other
  - https://peter.run/blog/2019-07-28-visualising-latency-variance-in-grafana-in-2019/
  - https://hveem.no/visualizing-latency-variance-with-grafana