Scalable monitoring tools - a mile-high view

Network Startup Resource Center

www.ws.nsrc.org

UNIVERSITY OF OREGON

NSRC Network Startup Resource Center

Contents

- Problem
- Requirements
- Solutions
- ELK stack
  - ElastiFlow
- TICK stack
- Prometheus
- TimescaleDB
- Kafka

Overview

The Problem(s)

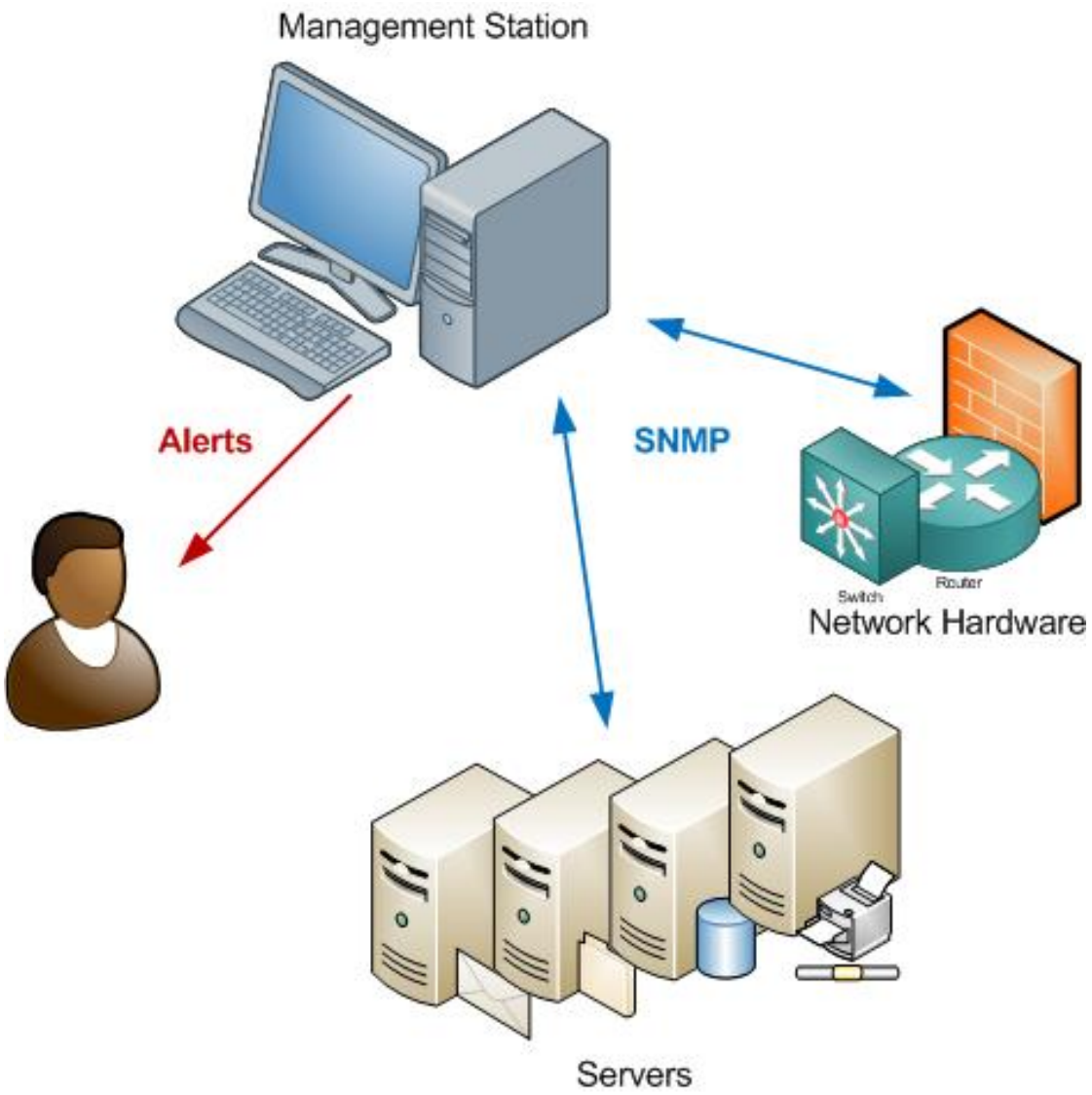
SNMP polling is…

- Slow
- Resource intensive
- Not fine-grained enough

This leads to…

- Slow response to network changes
- Difficult to implement tools on large networks
- Harder to datamine and alert

Classical Polling Model



Requirements

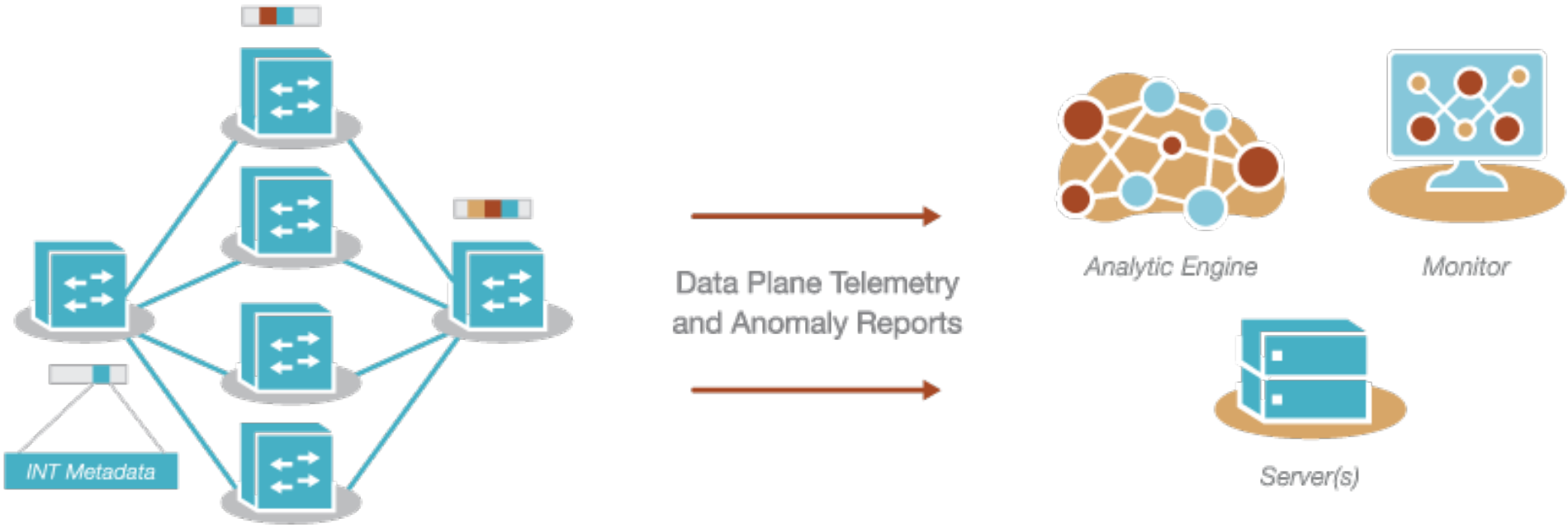
- Timeseries storage
- EVENT LOGS and METRICS
  - Events => syslog, snmp traps, netflow
  - Metrics => counters, gauges
- Searching and Visualisation
- Alerting
- Scale to large volumes of data
- APIs and integration options

Some Solutions

Go from polling to push/passive model

- Network telemetry, not standardized yet
  - Cisco compact Google Protocol Buffers
  - Google Protocol Buffers
  - JSON
- Logging data push solutions
  - Agent-based
  - Collectors and parsers
  - NoSQL database stores
  - Visualization and alerting systems

“Network Telemetry”



ELK vs. TICK

The “Elastic Stack”

ELK or “Elastic Stack”

- **E**lasticsearch: NoSQL Database / Search and analytics engine
- **L**ogstash: Log analysis and aggregator
- **K**ibana: Visualization layer

Beats - Outsourcing some Logstash functions.

Some examples include:

- **Filebeat**: logging agent on clients sending log data to Logstash or Elasticsearch directly.
- **Metribeat**: forwards server metrics
- **Packetbeat**: forwards network data

ELK vs. TICK

Data Collection

Data Aggregation & Processing

Indexing & storage

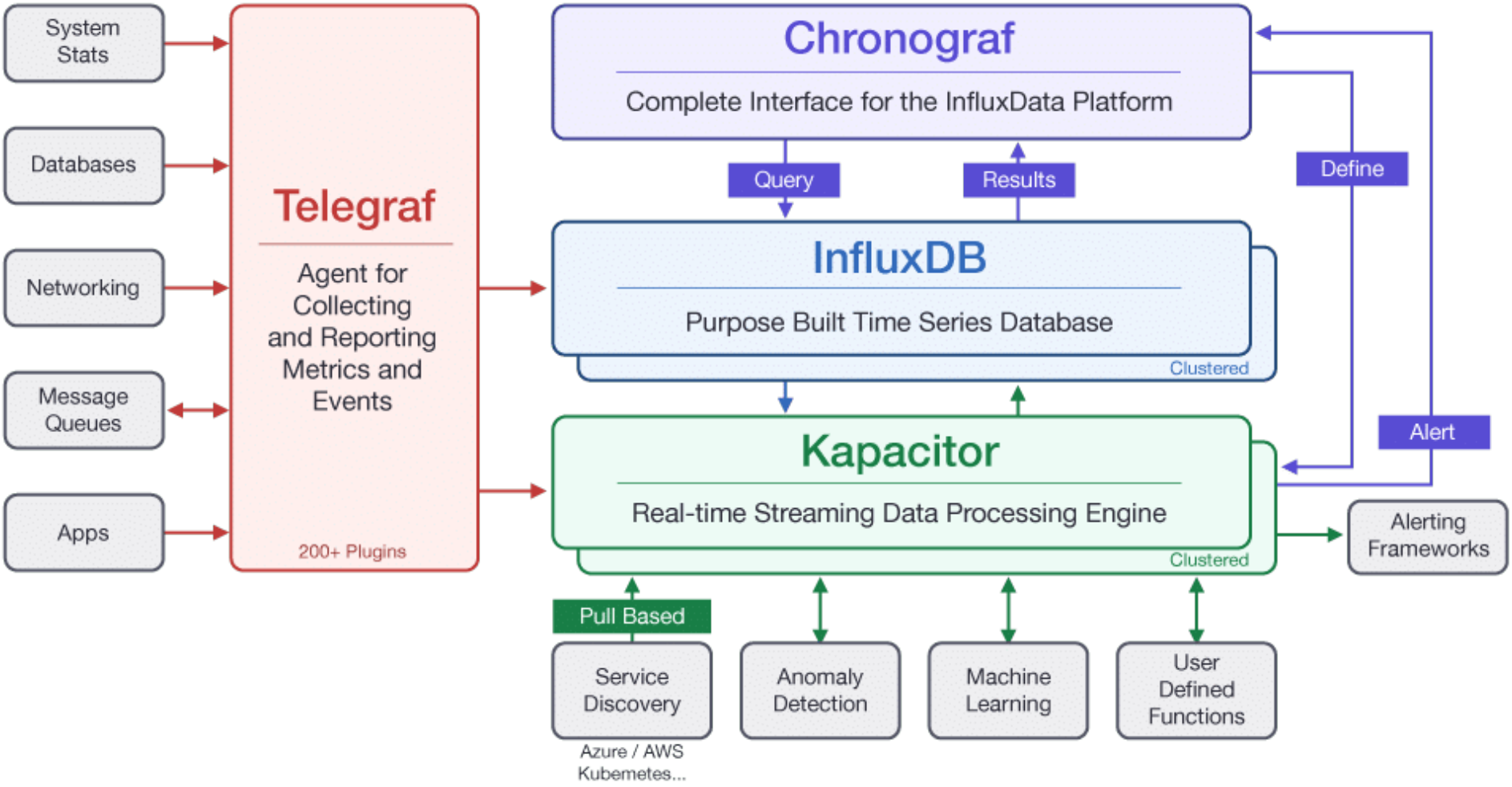
Analysis & visualization

ELK vs. TICK
TICK Stack

TICK:

- **Telegraph**: Metrics collection. Sends to InfluxDB
- **InfluxDB**: Time series, NoSQL database
- **Chronograf**: UI layer. Connects to InfluxDB and Kapacitor.
- **Kapacitor**: Metrics and events processing and alerting engine.

ELK vs. TICK


System Stats
Databases
Networking
Message Queues
Apps
Telegraf
Agent for Collecting and Reporting Metrics and Events
200+ Plugins
Chronograf
Complete Interface for the InfluxData Platform
Query
Results
Define
InfluxDB
Purpose Built Time Series Database
Clustered
Alert
Kapacitor
Real-time Streaming Data Processing Engine
Clustered
Alerting Frameworks
Pull Based
Service Discovery
Azure / AWS Kubernetes...
Anomaly Detection
Machine Learning
User Defined Functions

ELK and TICK

Comparing the Stacks – Over generalizing!

- ELK for log-based information
- TICK for network and log with alerting

Logstash	← →	Telegraph
Elasticsearch	← →	InfluxDB
Kibana	← →	Chronograf
Plugin to Elasticsearch	← →	Kapacitor
Beats	← →	*Standards like Google Protocol Buffers (GPB)*

Many other tools…

- **Kafka and KSQL**
  - Logstash and Elasticsearch (ELK)
  - Telegraph and InfluxDB (TICK)
- **Prometheus**
  - Like ELK and Tick but uses http pull to build time-series (NoSQL) data.
- **Grafana**
  - Like Kibana, but used for metrics analysis vs. exploring log data.

Many other tools…

- **Graphite**
  - Time-series, NoSQL DB and logger/grapher
- **Splunk**
  - Captures, indexes, correlates real-time date to generate graphs, reports, alerts, dashboards, and visualizations.
- **TimescaleDB**
  - Postgres-based, time-series DB, no stack but plugs in

Graphite

What Graphite is and is not.

Graphite does two things:

1. **Store numeric time-series data**
2. **Render graphs of this data on demand**

Graphite is **not** a collection agent, but it offers the simplest path for getting your measurements into a time-series database. **Feeding your metrics** into Graphite couldn't be any easier.

```
$ echo "foo.bar 1 `date +%s`" | nc localhost 2003
```

Need a collection agent or language bindings? Graphite has one of the **largest ecosystems** of data integrations and third-party tools.

https://graphiteapp.org/

Details

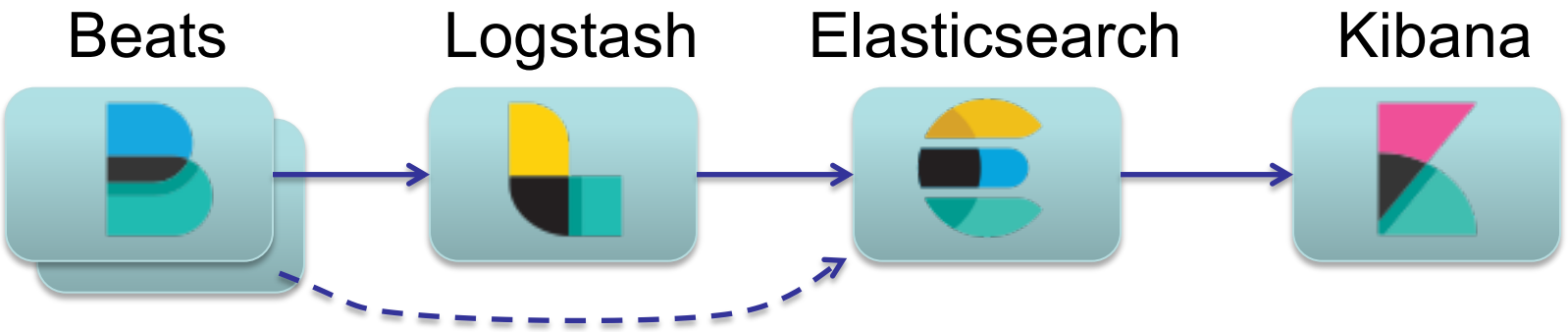
Requirements

- Timeseries storage
- EVENT LOGS and METRICS
  - Events => syslog, snmp traps, netflow
  - Metrics => counters, gauges
- Searching and Visualisation
- Alerting
- Scale to large volumes of data
- APIs and integration options

What's a timeseries?

- A set of *(timestamp, value)* points
- Need a way to identify all the points belonging to the same timeseries
  - Usually this is done by unique set of "tags"
  - `{metric="ifHCInOctets", device="1.2.3.4", ifDescr="Gi0/1"}`
- Timeseries constantly growing; eventually need to expire old data

The Elastic Stack (ELK)



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

Architecture

- Elasticsearch: JSON document database
  - An Elasticsearch database is called an "index"
  - Built on Lucene free-text search engine
  - Supports replicated and sharded clusters
- Logstash ingests and processes logs
- Kibana provides search UI and graphing
- Beats are lightweight, standalone data collectors

ELK Pros

- Long-standing and widely deployed
  - cloud-hosted services available, or DIY
- Fast free-text searching
- Can be scaled horizontally
- Rich data model, including first-class support for IP addresses
  - Per-field indexing if your data is structured (JSON)

ELK Pros

- Wide range of useful Beats, inc. winlog
- Logstash and Filebeat support Netflow
- Elasticsearch is also a component of many platforms including Wazuh, Security Onion, Graylog, SIEMonster etc.
- Pre-built Kibana dashboards

ELK Cons

- Many features, including alerting, require a commercial licence*
- Huge resource requirements
  - SSD recommended; but index typically 10 times larger than the ingested data
  - Large RAM requirements
  - It's all Java (apart from Beats)

** Check out Open Distro for Elasticsearch (by AWS), and X-pack alternatives such as elastalert and sentinl*

ELK Cons

- Not well suited to metrics
  - Expensive to scale up
  - Logstash finally released SNMP input plugin in October 2018
- Elasticsearch: questionable reliability as a primary data store?

ElastiFlow

ElastiFlow™ provides network flow data collection and visualization using the Elastic Stack (Elasticsearch, Logstash and Kibana). It supports Netflow v5/v9, sFlow and IPFIX flow types (1.x versions support only Netflow v5/v9).

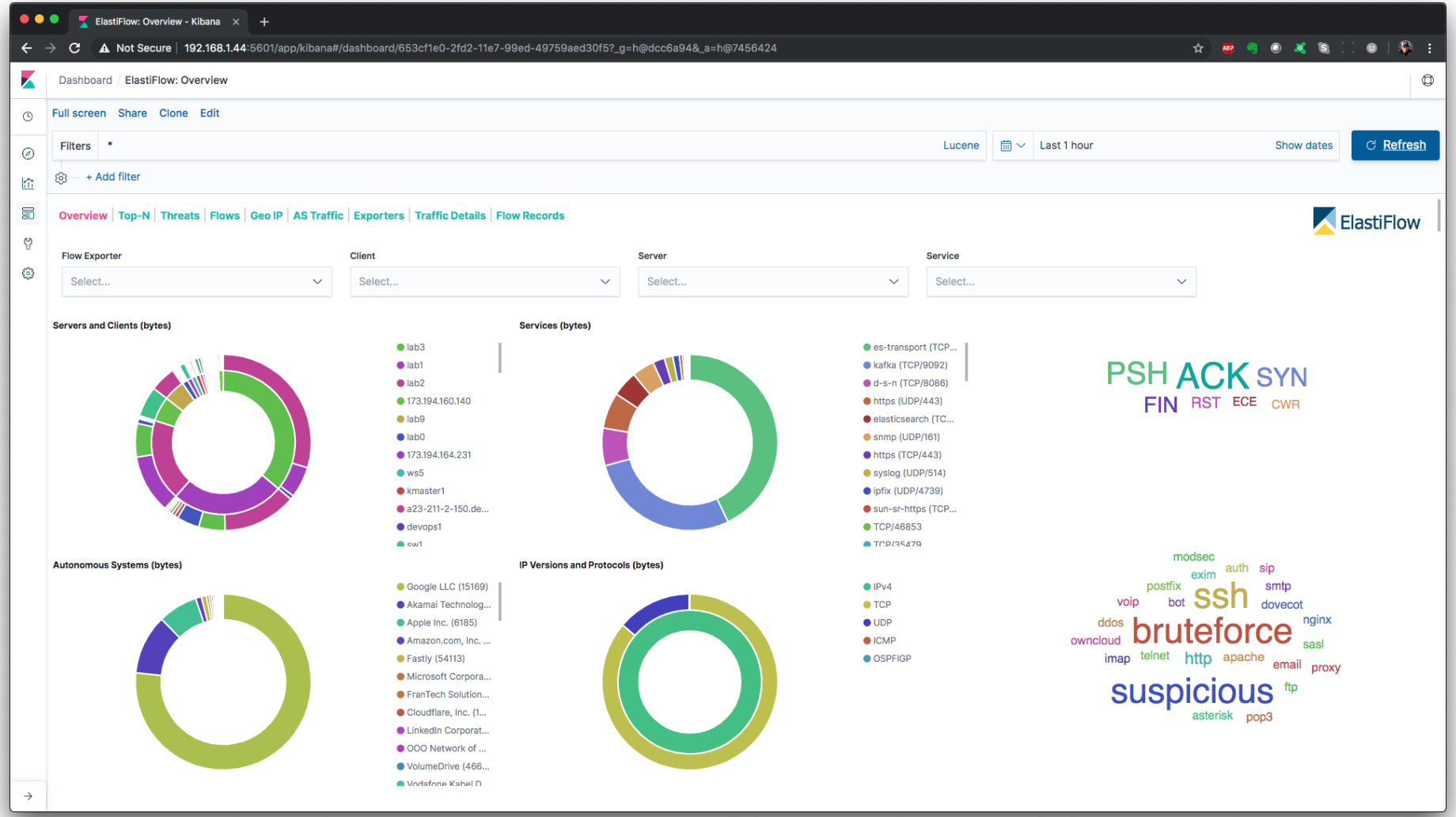
https://github.com/robcowart/elastiflow

Complete instructions for Ubuntu 18.04, Elastic Stack and ElastiFlow:

https://www.catapultsystems.com/blogs/install-elastiflow-on-ubuntu-18-04-part-1/

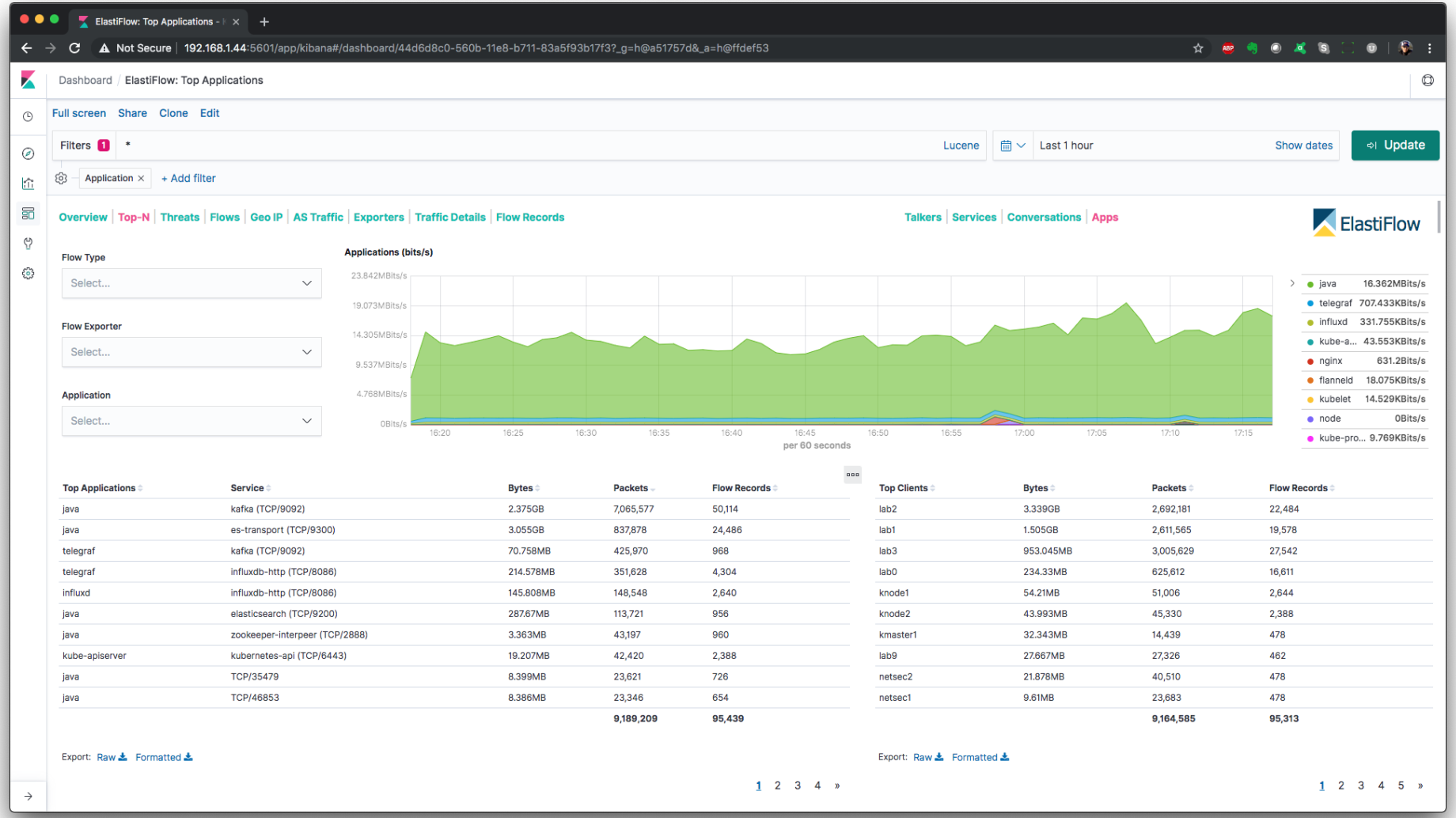
ElastiFlow Dashboards

Overview


ElastiFlow: Overview - Kibana
Not Secure 192.168.1.44:5601/app/kibana#/dashboard/653cf1e0-2fd2-11e7-99ed-49759aed30f5?_g=h@dcc6a94&_a=h@7456424
Dashboard / ElastiFlow: Overview
Full screen Share Clone Edit
Filters * Lucene Last 1 hour Show dates Refresh
+ Add filter
Overview Top-N Threats Flows Geo IP AS Traffic Exporters Traffic Details Flow Records
ElastiFlow
Flow Exporter
Client
Server
Service
Select...
Servers and Clients (bytes)
lab3
lab1
lab2
173.194.160.140
lab9
lab0
173.194.164.231
ws5
kmaster1
a23-211-2-150.de...
devops1
Services (bytes)
es-transport (TCP...
kafka (TCP/9092)
d-s-n (TCP/8086)
https (UDP/443)
elasticsearch (TC...
snmp (UDP/161)
https (TCP/443)
syslog (UDP/514)
ipfix (UDP/4739)
sun-sr-https (TCP...
TCP/46853
PSH ACK SYN
FIN RST ECE CWR
Autonomous Systems (bytes)
Google LLC (15169)
Akamai Technolog...
Apple Inc. (6185)
Amazon.com, Inc. ...
Fastly (54113)
Microsoft Corpora...
FranTech Solution...
Cloudflare, Inc. (1...
LinkedIn Corporat...
OOO Network of ...
VolumeDrive (466...
IP Versions and Protocols (bytes)
IPv4
TCP
UDP
ICMP
OSPFIGP
modsec
auth
sip
exim
postfix
ssh
smtp
voip
bot
dovecot
ddos
bruteforce
nginx
owncloud
sasl
imap
telnet
http
apache
email
proxy
suspicious
ftp
asterisk
pop3

Top-N

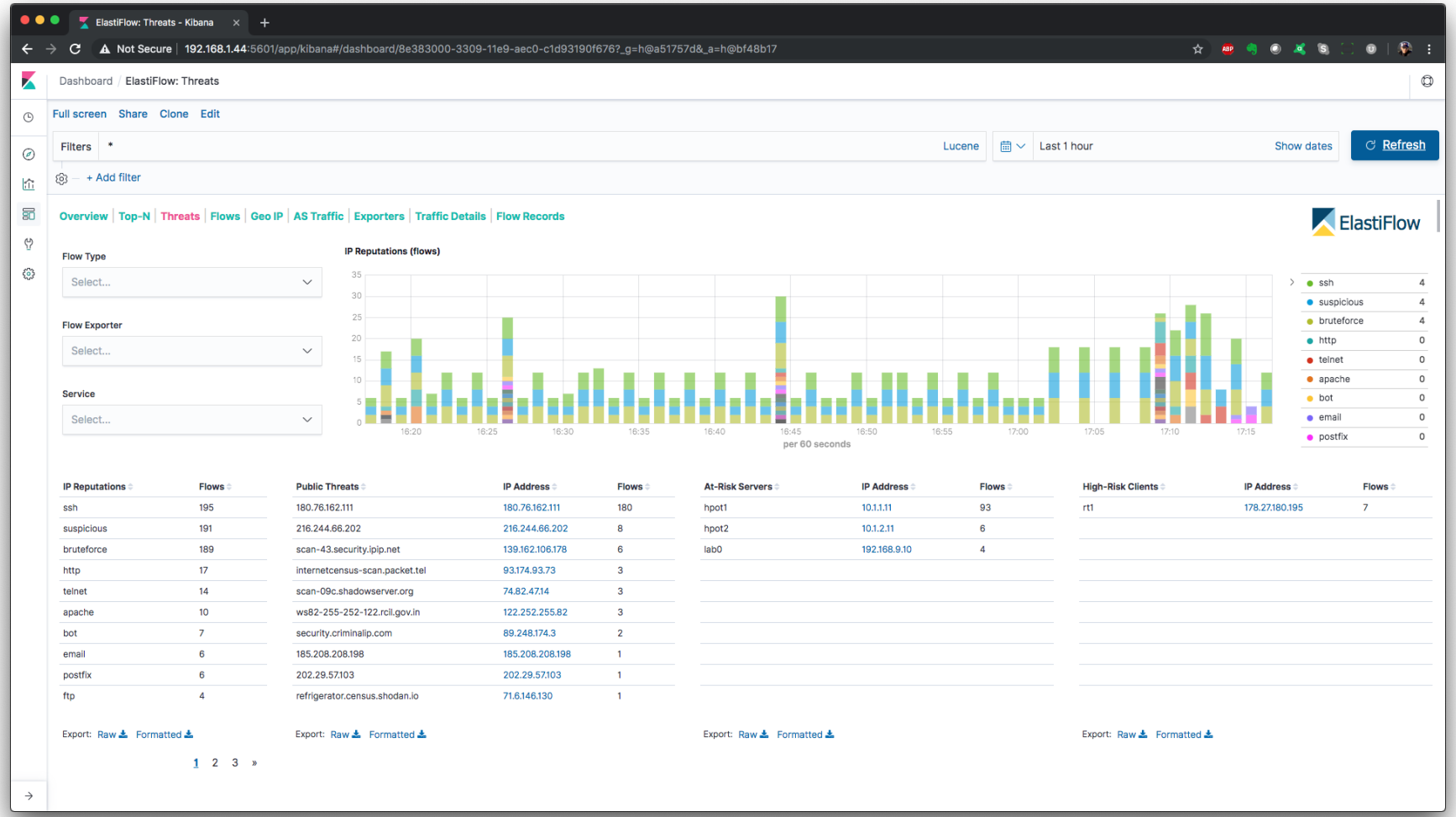
There are separate Top-N dashboards for Top Talkers, Services, Conversations and Applications.



Threats

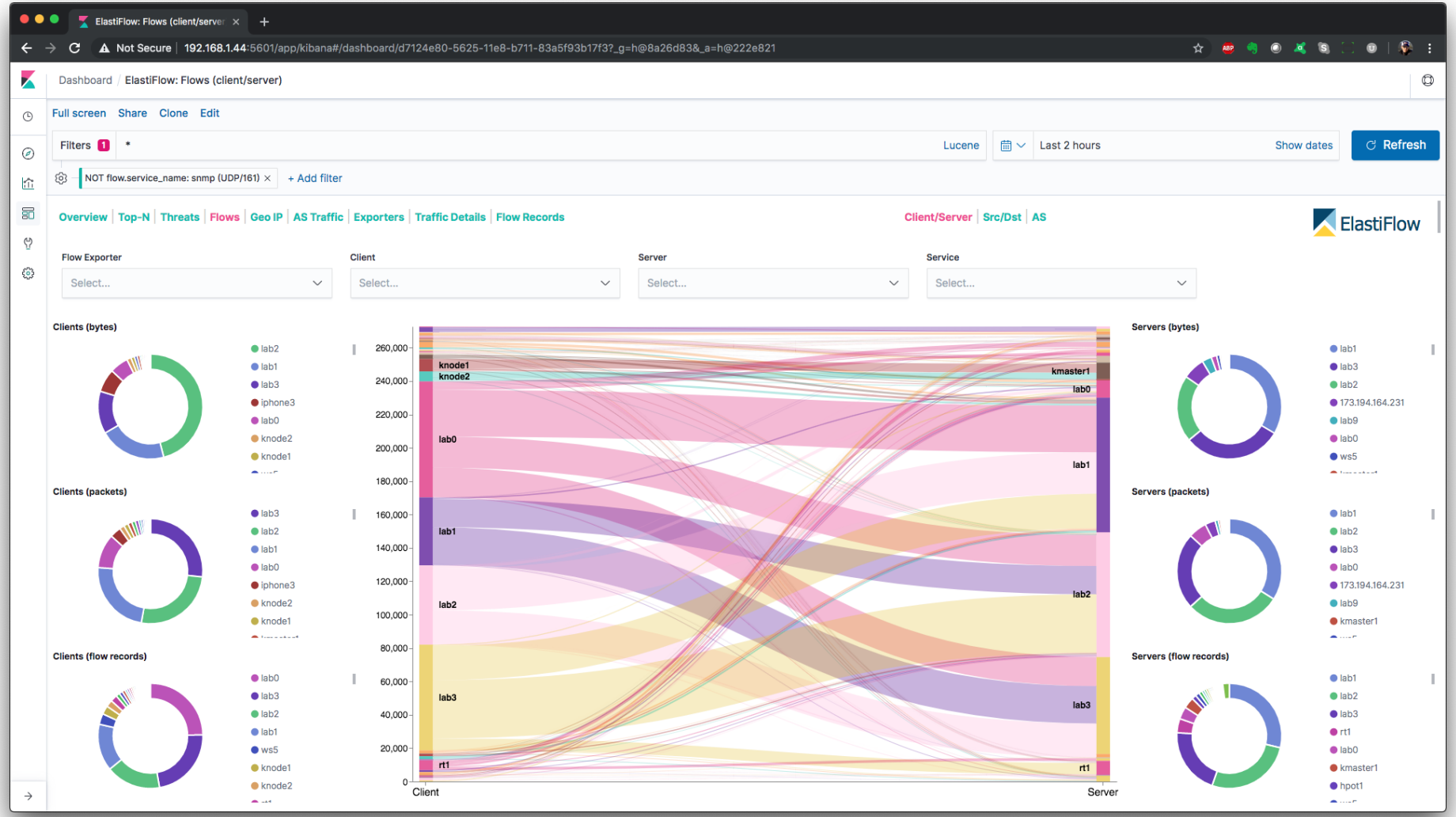
ElastiFlow™ includes a dictionary of public IP addresses that are known to have a poor reputation. This dictionary is built from many OSINT data sources, normalized to a common taxonomy. The Threats dashboard uses this IP reputation information to highlight three threat/risk types.

- **Public Threats** - Public clients with a poor IP reputation that are reaching private addresses.
- **At-Risk Servers** - Private Servers that are being reached by clients with a poor IP reputation.
- **High-Risk Clients** - Private clients that are accessing public servers which have a poor reputation.



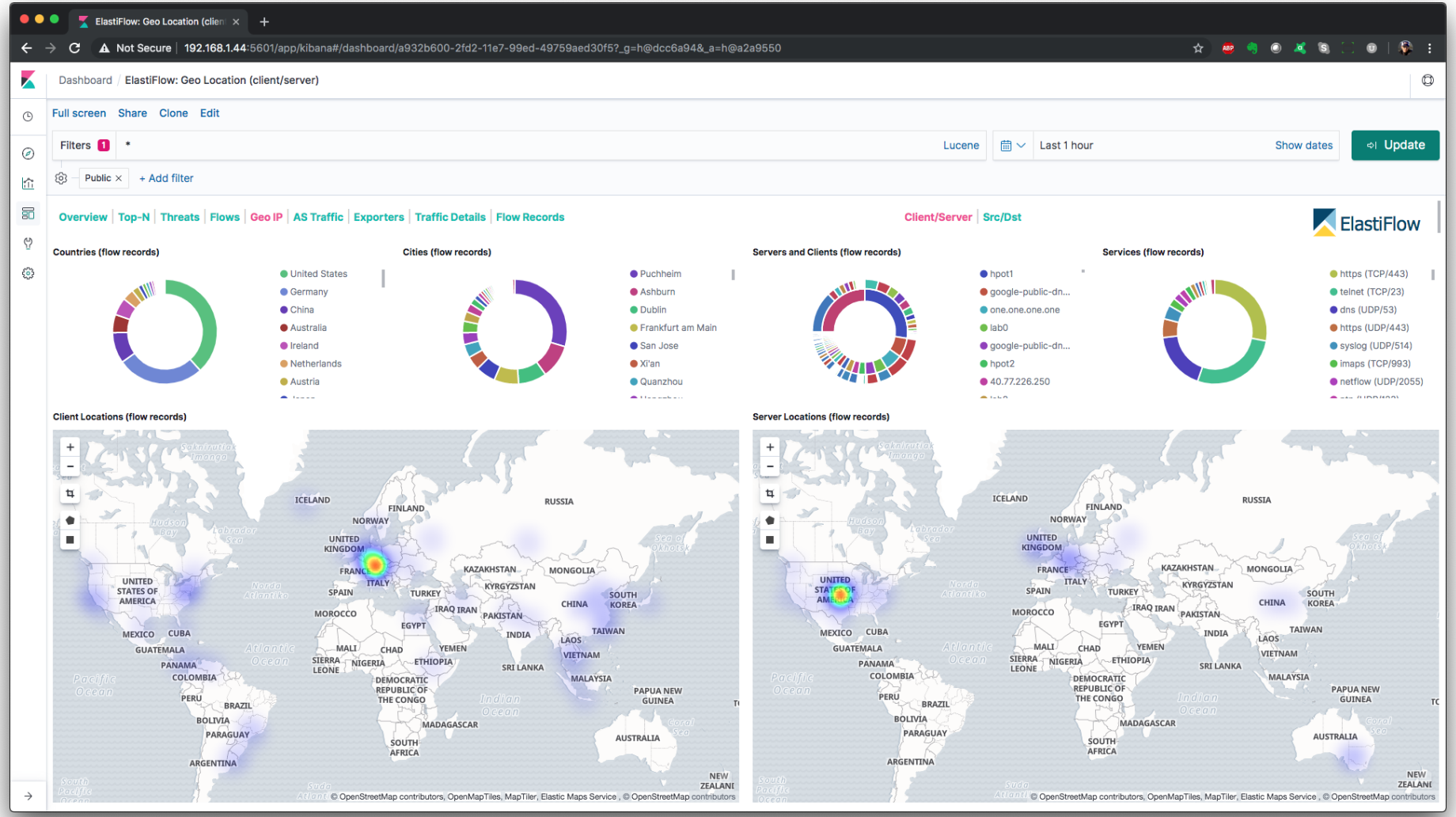
Flows

There are separate Sankey dashboards for Client/Server, Source/Destination and Autonomous System perspectives. The sankey visualizations are built using the new Vega visualization plugin.



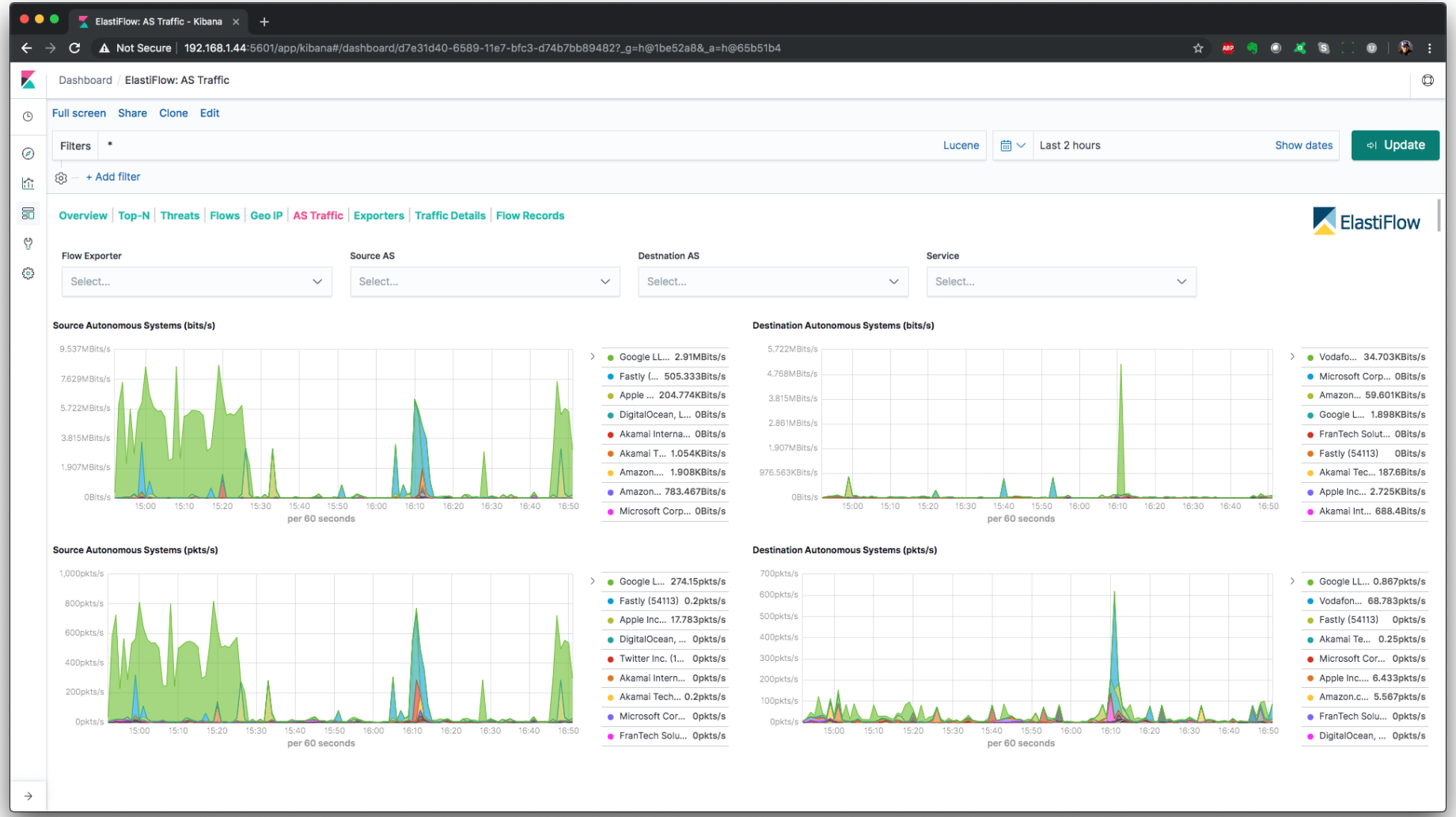
Geo IP

There are separate Geo Location dashboards for Client/Server and Source/Destination perspectives.



AS Traffic

Provides a view of traffic to and from Autonomous Systems (public IP ranges)



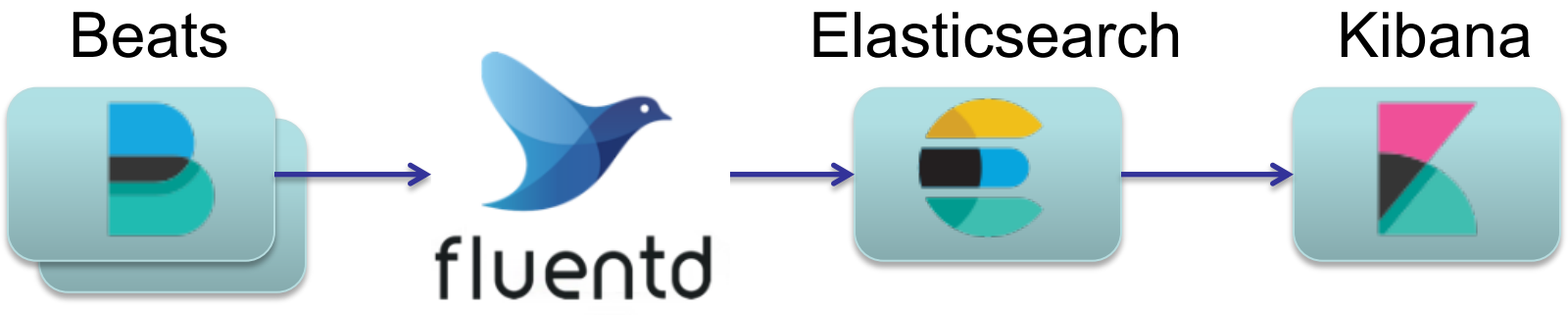
Resources for ElastiFlow

What amount of resources are required to run ElastiFlow:

flows/sec	(v)CPUs	Memory	Disk (30-days)	ES JVM Heap	LS JVM Heap
250	4	24 GB	305 GB	8 GB	4 GB
1000	8	32 GB	1.22 TB	12 GB	4 GB
2500	12	64 GB	3.05 TB	24 GB	6 GB

https://github.com/robcowart/elastiflow/blob/master/INSTALL.md

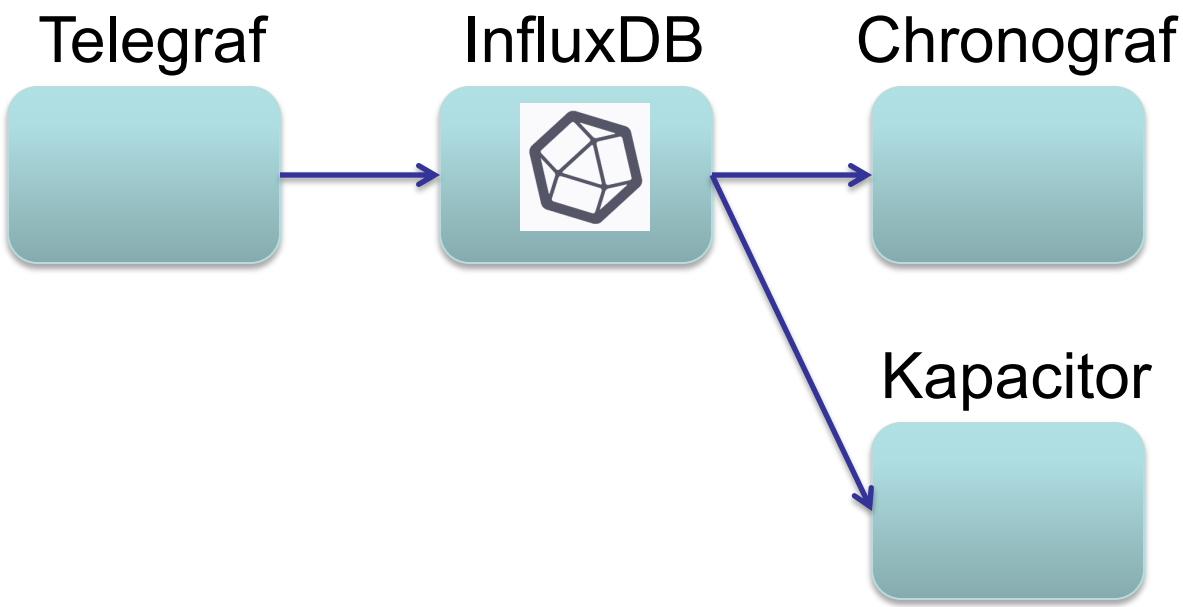
Modification: the EFK stack



Replace logstash with fluentd and fluent-plugin-beats
- avoids a large Java dependency

https://www.slideshare.net/repeatedly/fluentpluginbeats-at-elasticsearch-meetup-14

The TICK Stack



Architecture

- Telegraf is data collector and processor
  - SNMP polling, system and application metrics, syslog
- InfluxDB is custom time series database
- Chronograf is admin UI and visualisation
  - includes basic syslog browser
- Kapacitor is stream processing and alerting

TICK Pros

- Lightweight binaries (Go); easy to install
- Excellent metric support
  - Columnar storage with high compression
- Stores int, float, text and bool natively
- "SQL-inspired" query language is easy to get started with
- Inline data processing via "continuous queries" and "subscriptions"

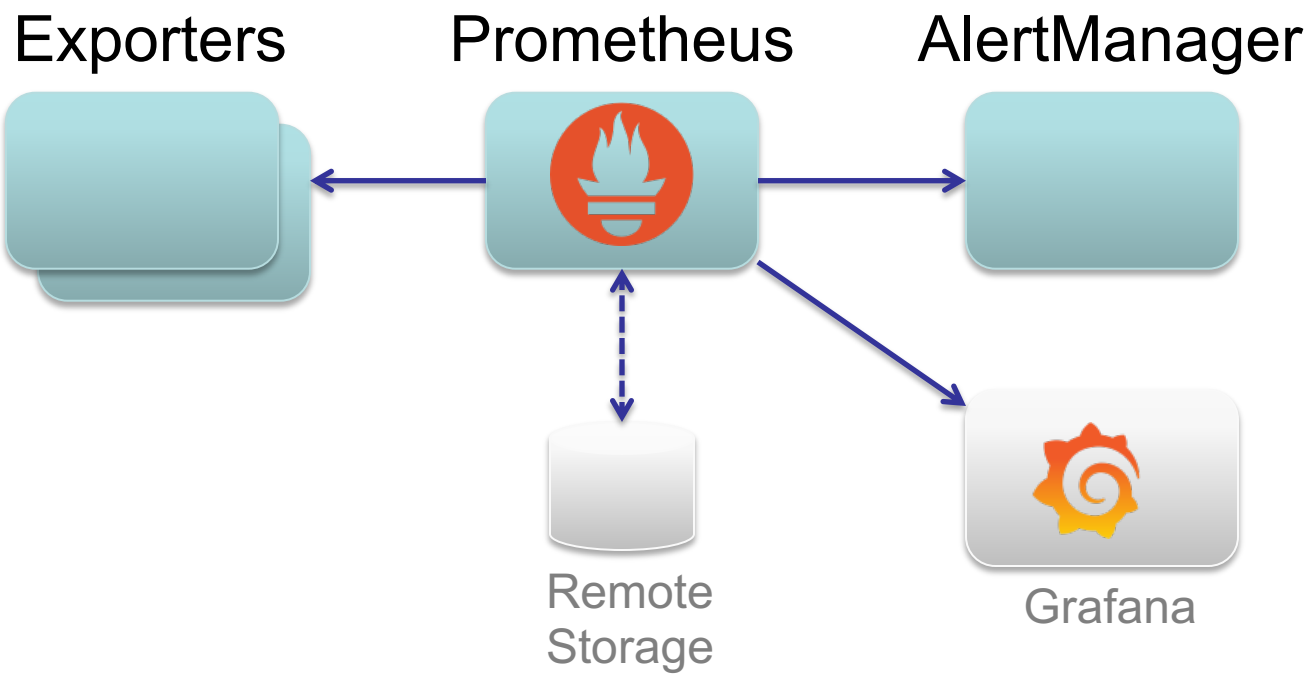
TICK Cons

- InfluxQL differences from SQL soon become apparent
  - single and double quotes are *very* different!
- Kapacitor has yet another language too
  - "TICKScript"
  - Plan to unify in InfluxDB 2.0 ("Flux")
- Alerting hard to set up, and not too intelligent

TICK Cons

- A columnar database isn't *really* suited to event logs
  - Cannot have two "rows" with the same timestamp (but has nanosecond resolution)
  - Text search is brute-force full column scan
- Maybe unpredictable RAM usage and database corruption (but much improved)
- Horizontal scaling only in commercial cluster product

Prometheus


Exporters
Prometheus
AlertManager
Remote
Storage
Grafana

Architecture

- "Exporters" are simple HTTP endpoints that return metrics when queried
- Prometheus contains polling engine, metric database, threshold detection, and rewriting/downsampling engine
- Alertmanager processes and delivers alerts
  - e.g. can delay and aggregate related alerts
- Basic query UI; use Grafana for dashboards

Architecture: "Pull" monitoring

- All polling is performed by prometheus
  - makes periodic outbound HTTP connections
  - This process is called "scraping" the target
- Exporters respond passively on request
  - easy to point additional prometheus servers at them, e.g. for redundancy or testing
  - monitored servers don't need to be told where the monitoring station is

Sample exporters

- *node_exporter:* local system metrics
- *snmp_exporter:* poll SNMP devices
- *grok_exporter* or *mtail:* generate metrics from parsing log files
- *blackbox_exporter:* nagios-like service checks
  - can also integrate with real nagios: e.g. *nagios_exporter*, *nrpe_exporter, nagitheus*

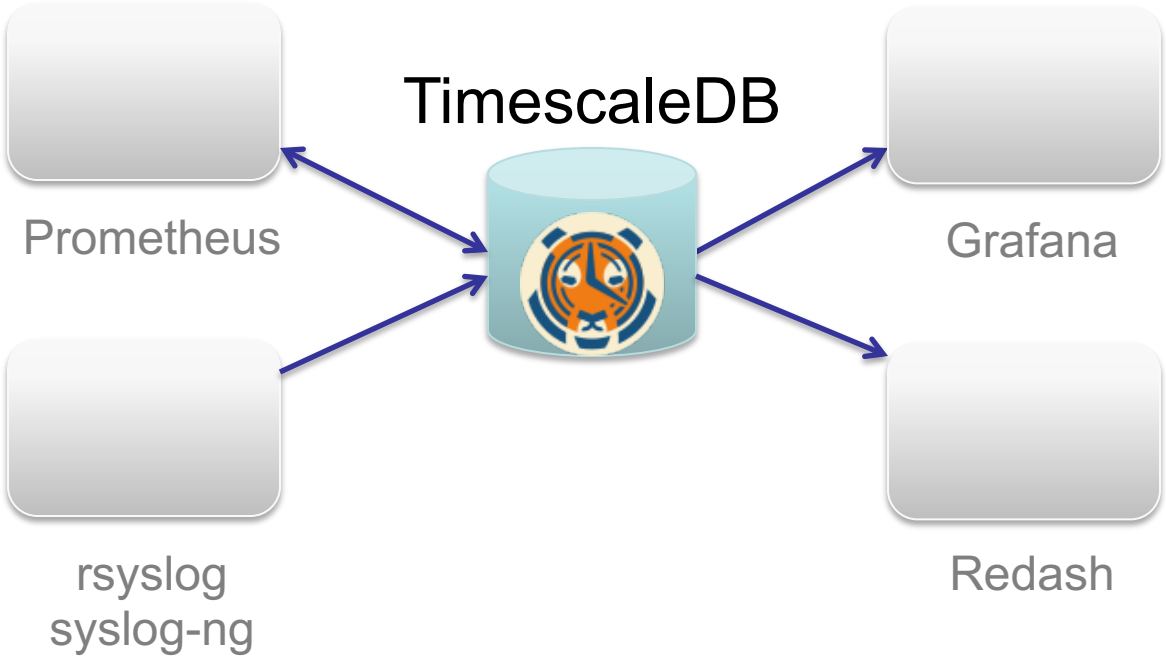
Prometheus Pros

- Massively scalable and efficient
  - handles millions of timeseries
  - typically less than 2 bytes per data point
- Pretty easy to install
  - static binaries with no dependencies
- Wide ecosystem, easy to extend
  - With *node_exporter*, just drop custom metrics into a file and you're done!

Prometheus Cons

- Metrics only (one data type: float64)
- Not designed for long-term storage
  - Default is 14 days. You can increase this but must be same for all metrics
  - Can remote-write to other types of database
  - *Thanos* extension offers long-term S3 storage
- Functional query language is powerful but hard to get to grips with

TimescaleDB



Architecture

- TimescaleDB is an *extension* to Postgres
- Transparently creates table chunks for different time ranges
  - Prevents indexes getting too large
  - Most activity is in most recent chunk
  - Very cheap to expire old chunks
- Adds some aggregation functions too

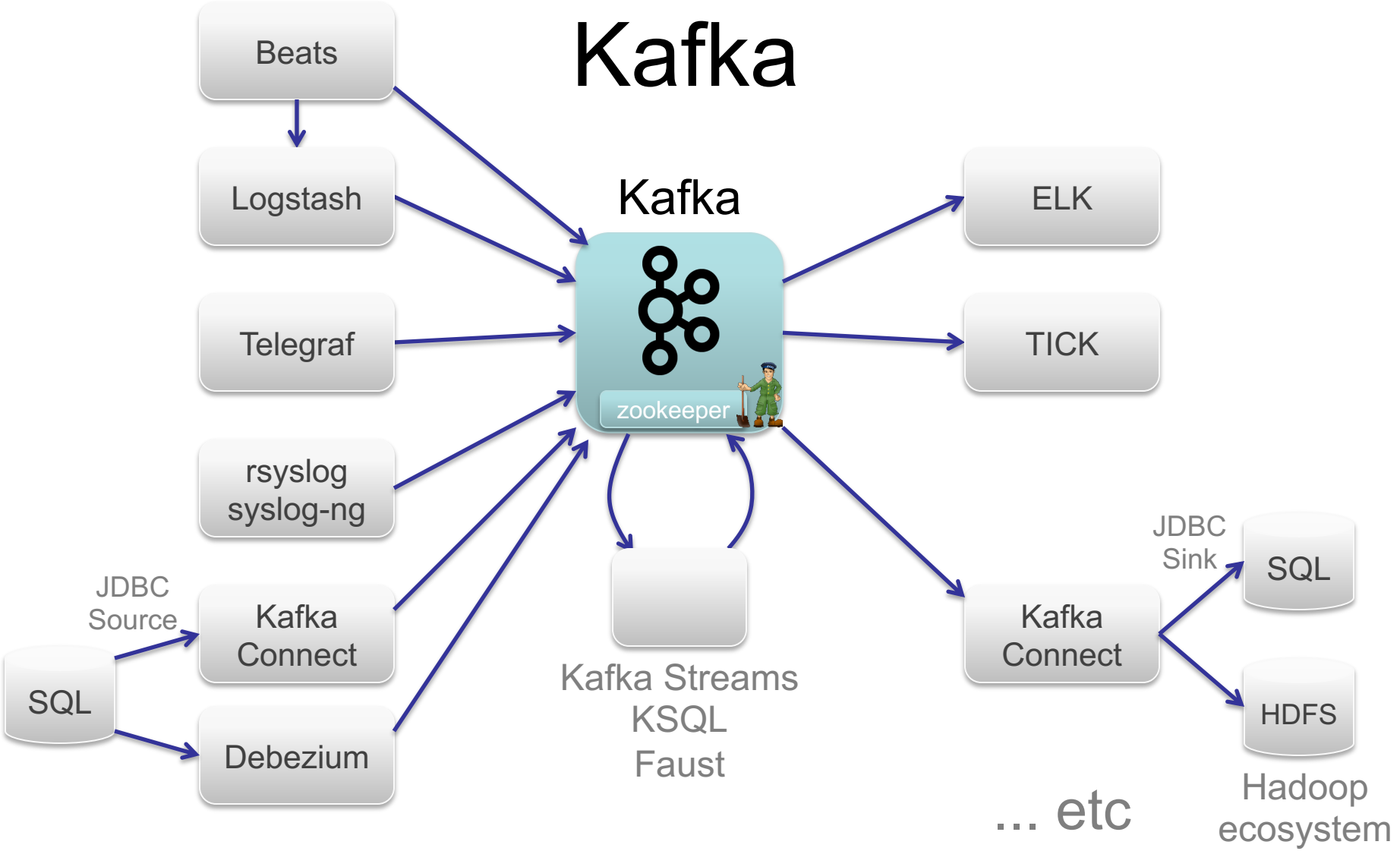
TimescaleDB Pros

- It's Postgres!
  - Rock-solid data storage
  - Reliable backup and restore, replication
  - Use existing DBA skills
  - Full power of SQL in queries

TimescaleDB Cons

- It's just a database – you need to build the stack yourself
  - Creating schemas
  - Getting data in and out
  - Alerting (maybe Grafana's alerting is OK?)
- Fits easily with Prometheus, but not TICK
- Uses substantially more storage space than InfluxDB or Prometheus

Kafka


Beats
Logstash
Telegraf
rsyslog
syslog-ng
JDBC
Source
SQL
Kafka
Connect
Debezium
Kafka
zookeeper
Kafka Streams
KSQL
Faust
ELK
TICK
Kafka
Connect
JDBC
Sink
SQL
HDFS
Hadoop
ecosystem
... etc

Architecture

- Kafka is really just a big queue
- Append messages to "topics"
- Subscribe in "consumer groups"
  - each consumer group receives a copy
- Messages aren't deleted until you say so
  - Consumers can rewind and replay
  - Potentially query in place (e.g. Apache Drill)

Kafka Pros

- System decoupling
  - e.g. use Beats outside of ELK
  - bolt on new alerting/analysis tools
  - write your own custom stream processing
- Massively scalable and reliable
  - Topics can be partitioned and replicated
- Can act as a long-term archive
  - Append-only model is efficient even with HDD

Kafka Cons

- Not a complete monitoring stack
- Not one but *two* Java services to maintain
- No integrated admin UI, several to choose
- You have to decide the message format
  - Telegraf JSON? Beats JSON? fluentd JSON? InfluxDB line protocol? CSV? Avro? ...
- Not well suited to Prometheus "pull" model

Honorable mentions: metrics

- netdata
  - awesome tool for performance debugging
  - configures itself out of the box
  - system metrics at *1 second resolution* by default, retained for one hour in RAM
  - plenty of predefined alerts

Honorable mentions: logs

- fluentd (td-agent)
  - easily extensible in ruby; worth looking at for custom requirements
  - EFK stack: replacing logstash with fluentd
  - fluent-bit for very lightweight install
- loki
  - new project: prometheus-inspired storage of logs, with grafana as user interface

Interesting notes

- Grafana can be used with Prometheus, InfluxDB *and* Elasticsearch
  - mixed dashboards are possible
- Grafana can do basic alerting by itself
  - maybe you find this easier
- Prometheus remote storage can read and write to InfluxDB *and* TimescaleDB

Consider when choosing

- All of these (apart from TimescaleDB) have esoteric query languages you'll have to learn
- Managing large Java apps can be difficult if you don't have the skillz
- How do you feel about separate stacks for metrics and logs?

References

- Kafka
  https://docs.confluent.io/current/streams-ksql.html
- Graphite
  https://graphiteapp.org/
- InfluxDB
  https://www.influxdata.com/
- Logz.io (Information on *Elastic Stack,* others)
  https://logz.io/
- Prometheus
  https://prometheus.io/

References

- Splunk
  https://www.splunk.com/
- 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
- 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/

The End!

Questions?