Jesse White

OPENNMS INTEGRATION PATTERNS

OpenNMS®

Jesse White
CTO

ABOUT ME

- Using OpenNMS since 2012
- Participated in Google Summer of Code in 2013
- Joined The OpenNMS Group Inc. in 2014
- Started The OpenNMS Group Canada Inc. in 2017

INTEGRATION PATTERNS

1 ARCHITECTURE

An architectural overview of the components we'll be looking at

2 EVENTS

Sending and receiving events

3 ALARMS

Reacting to alarms

4 INVENTORY

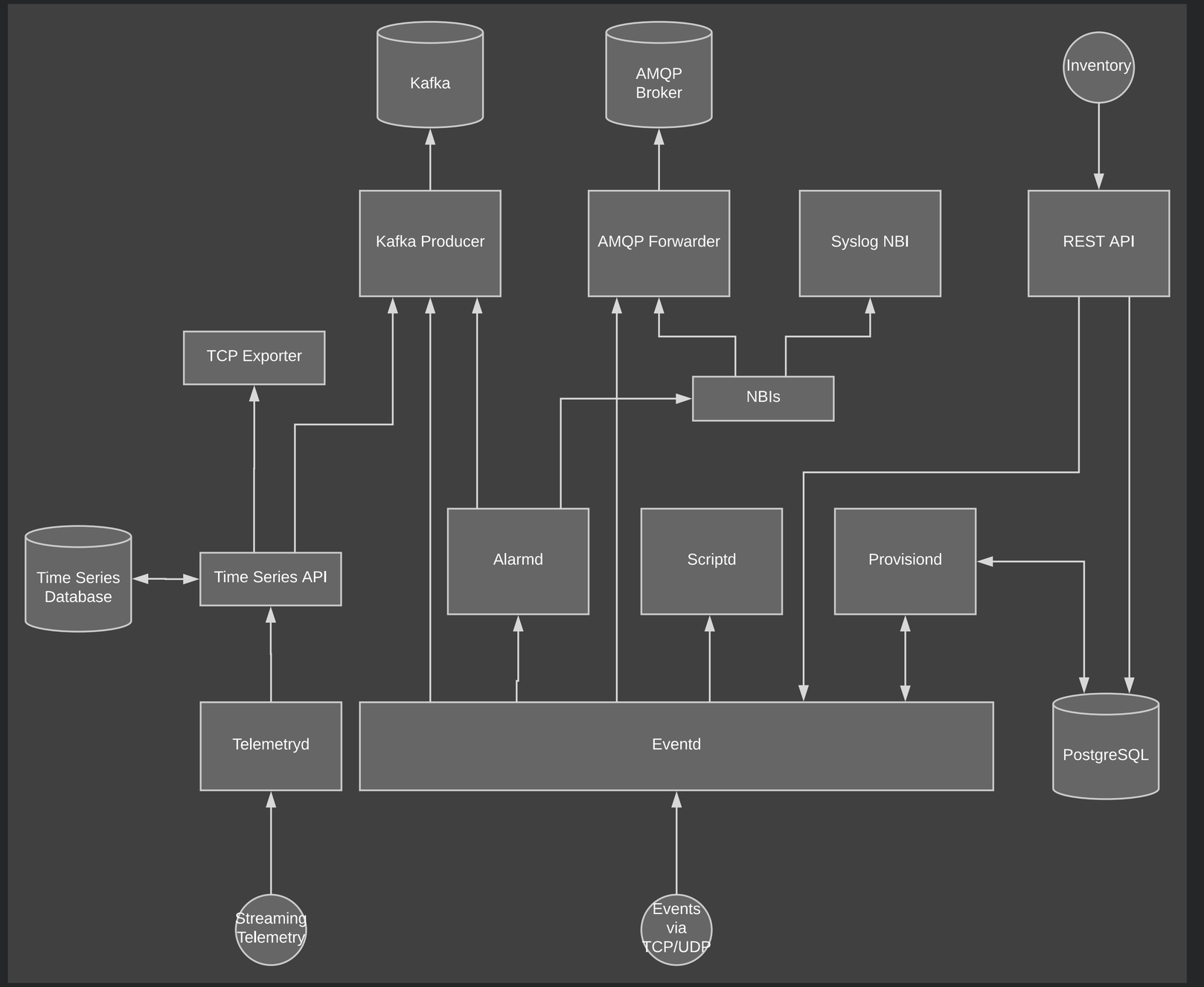
Managing elements

5 PERFORMANCE DATA

Metrics in and out

6 KAFKA

Stream processing


Kafka
AMQP Broker
Inventory
Kafka Producer
AMQP Forwarder
Syslog NBI
REST API
TCP Exporter
NBIs
Time Series Database
Time Series API
Alarmd
Scriptd
Provisiond
Telemetryd
Eventd
PostgreSQL
Streaming Telemetry
Events via TCP/UDP

Events

Sending and receiving events

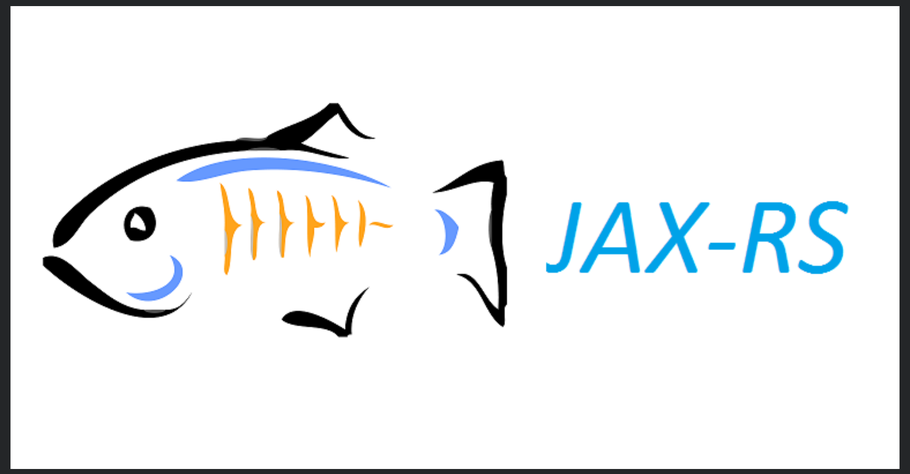
IN: Event TCP/UDP Listeners

- Netty based TCP and UDP listeners
- Listeners accept an “event log” in XML format

FACTS

- Available since: 1.0
- Authentication/Authorization: None
- Performance: Single log per socket/message, unlimited number of events per log, async processing
- Schema: Stable XML XSD

IN: POST events via REST



- Post JSON or XML to /rest/events

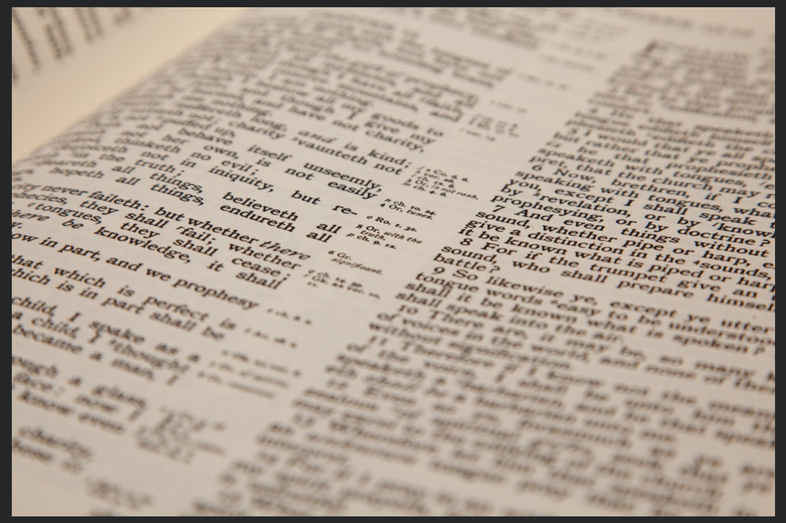
FACTS

- Available since: Horizon 17.1.1, Meridian-2016.1.0
- Introduced in: NMS-6404
- Authentication/Authorization: Valid user w/ role
- Performance: Single event per POST, async processing
- Schema: Stable XML XSD

DEMO

Send events via TCP and REST - see https://github.com/j-white/ouce2018-oip

OUT: Trigger scripts with events in scriptd



- Trigger JSR-223 compatible scripts with events
- Supported languages include:
  - Beanshell
  - Groovy
  - Javascript
  - Python (Jython)
  - Ruby (JRuby)

FACTS

- Available since: 1.0
- Authentication/Authorization: Up to the script
- Performance: Single threaded
- Schema: Stable event bean

DEMO

Perform HTTP post on event via Jython - see https://github.com/j-white/ouce2018-oip

OUT: Events via AMQP

FACTS

- Available since: 17.1.0
- Introduced in: HZN-537
- Authentication/Authorization: Broker based
- Performance: Good for low/medium volumes of events
- Schema: Stable event bean

- Forwards events to a AMQP (Advanced Message Queuing Protocol) compatible broker
- Support for custom processors to mangle events before forwarding
- Requires AMQP 1-0 which is supported in:
  - ActiveMQ
  - QPID
  - RabbitMQ (w/ plugin)

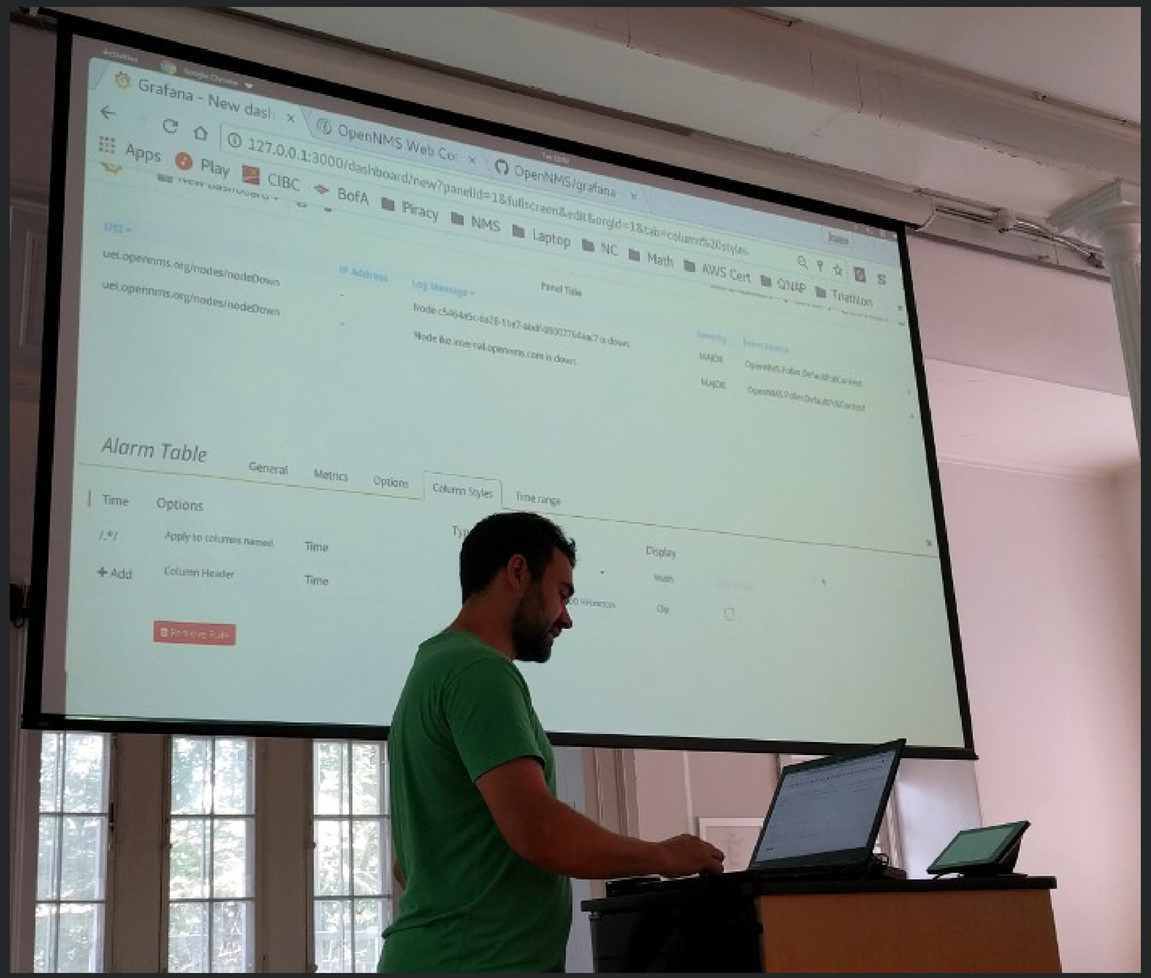
DEMO

Pub/sub events with AMQP and RabbitMQ – see https://github.com/j-white/ouce2018-oip

Alarms

Reacting to alarms

IN: Events trigger alarms



OUT: Alarms via Northbounder Interfaces (NBIs)

- Forward alarms via Syslog, SNMP (trap), JMS, AMQP, etc...

FACTS

- Available since: Depends on NBI
- Authentication/Authorization: Depends on NBI
- Performance: Single threaded
- Schema: Stable northbound alarm bean
- Limitations: Not aware of all updates to alarms

DEMO

Trigger syslog messages on alarm - see https://github.com/j-white/ouce2018-oip

Inventory

Managing elements

IN: Inventory via REST

- Manage provisioning requisitions via REST

FACTS

- Available since: 1.8?
- Authentication/Authorization: Valid user w/ role
- Performance: Async handling, needs turning for large env.
- Schema: Stable requisition schema

DEMO

Provision a node using provision.pl – see https://github.com/j-white/ouce2018-oip

OUT: Inventory via REST

- Query nodes via REST
- Flexible criteria support in the v2 API

FACTS

- Available since: 1.8?, v2 API since 21.0.0
- Authentication/Authorization: Valid user w/ role
- Performance: Database bound
- Schema: None

DEMO

Query nodes using the v2 REST API - see https://github.com/j-white/ouce2018-oip

Performance Data

Metrics in and out

IN: Streaming telemetry

- Support for NXOS (Cisco), JTI (Juniper), and sFlow protocols
- Extensible framework for adding new protocols
- Scalable processing added to Horizon 23.0.0 (Sentinel+Newts)

FACTS

- Available since: 21.0.0
- Authentication/Authorization: None
- Performance: Fast!
- Schema: Depends on protocol

DEMO

Stream JTI payloads - see https://github.com/j-white/ouce2018-oip

OUT: TCP exporter

PerformanceData_Horizon_23.0.0.proto

```
1  option java_package = "org.opennms.netmgt.rrd.tcp";
2  option java_outer_classname = "PerformanceDataProtos";
3
4  message PerformanceDataReading {
5    required string path = 1;
6    required string owner = 2;
7    required uint64 timestamp = 3;
8    repeated double dblValue = 4;
9    repeated string strValue = 5;
10 }
11
12 message PerformanceDataReadings {
13   repeated PerformanceDataReading message = 1;
14 }
15
```

- Send RRD updates over a TCP socket

FACTS

- Available since: 1.7.9
- Authentication/Authorization: None
- Performance: Fast!
- Schema: Protobuf

DEMO

Python TCP listener for performance data – see https://github.com/j-white/ouce2018-oip

A

Stream all the data

Consistent interface for events, alarms, inventory & performance data

B

Stable API & Model

Objects are modeled in Protobuf which allows for compact transmission and allows us to add fields without breaking existing applications

C

Support Many Consumers

Many applications can subscribe to the same topics

D

Scale

Scale up your clusters as needed to support the required data rates or retention periods

DEMO

Python Kafka Consumer – see https://github.com/j-white/ouce2018-oip

SAY HI

- jesse on chat.opennms.com
- @jesse_white_ on Twitter
- jesse@opennms.ca

openNMS®