OpenNMS

Integration Patterns in OpenNMS Horizon

Horizon Ecosystem Developer Training • 28 Mar 2022

Presented by Jeff Gehlbach `<jeffg@opennms.com>` / Mattermost: `@jeffg`

Important acknowledgment

This talk is a retread of Jesse's one from OUCE 2018. See `j-white/ouce2018-oip` on GH.

Who is this person?

Jeff Gehlbach
Product Manager

- Network management practitioner since 1999
- Started using OpenNMS in 2005
- Joined The OpenNMS Group in 2007 as support engineer
- Later: consultant / solutions engineer / sales engineer
- Product Manager since 2021

Integration Patterns in Horizon – Agenda

1. Architecture

- A high-level overview of what we'll be getting into

2. Events

- Sending and receiving *events*

3. Alarms

- Working with *alarms*

4. Inventory

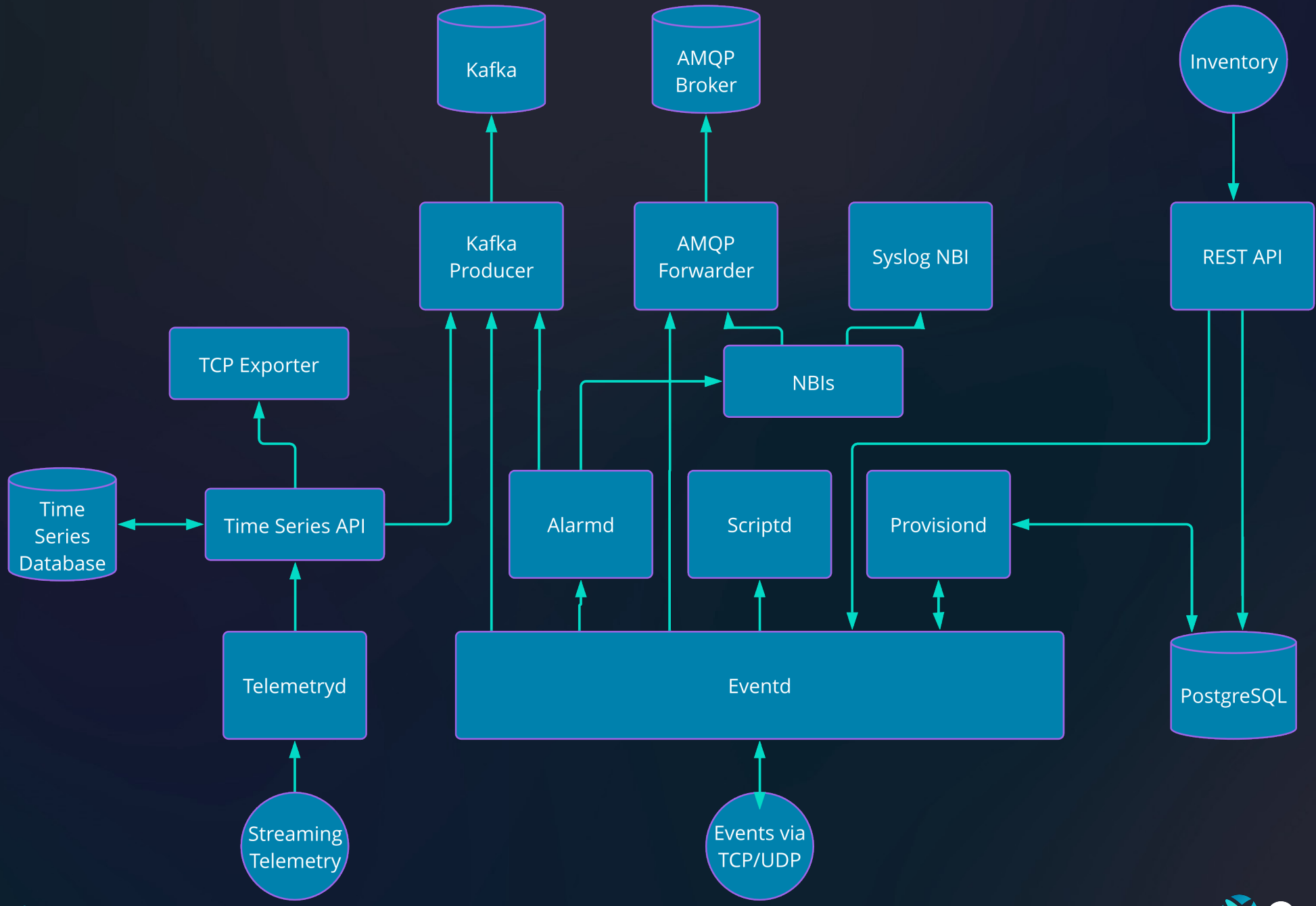
- Managing *nodes*, *interfaces*, and *services*

5. Performance Data and Flows

- Metrics in and out
- Flows in and out

6. Kafka

- Stream all the things


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

OpenNMS

Events

Sending and receiving events

OpenNMS

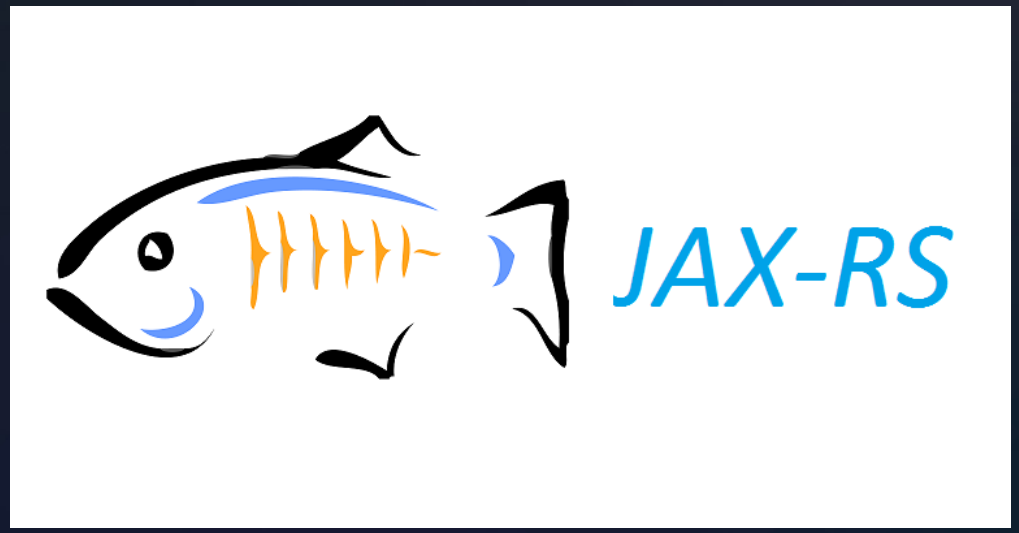
IN: Event TCP/UDP Listeners

- Netty-based TCP and UDP listeners
- Accept an “event log” XML document
  - No JSON support

Facts

- Available since: 1.0
- Authn / Authz: None
- Performance:
  - One log per socket per message
  - Unlimited 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
  - Also Meridian 2016.1.0
- Introduced in: NMS-6404
- Authn / Authz: Valid user with role
- Performance:
  - Single event per POST
  - Async processing
- Schema: Stable XML XSD

IN: Send event via Karaf shell

- Minimum spec: `send-event <uei>`
- Get `--help` if you need it

Facts

- Available since: 25.0.0
- Introduced in: NMS-12078
- Authn / Authz: Valid user with role
- Performance: Useful mainly for quick testing
- Schema: N/A – deals in high-level concepts

Demo – Events IN

- Send event via TCP – `01_in_tcp.sh`
- Send event via REST – `02_in_rest.xml`, `02_in_rest.sh`
- Send event via `send-event` Karaf shell command

OUT: Trigger scripts with events in Scriptd

```
<?xml version="1.0"?>
<scriptd-configuration>
        <engine language="beanshell" className="bsh.util.BeanShellBSFEngine" extensions="bsh"/>

        <start-script language="beanshell">

                import org.opennms.core.utils.InetAddressUtils;
                import org.opennms.netmgt.snmp.SnmpTrapBuilder;
                import org.opennms.netmgt.config.DataSourceFactory;
                import org.opennms.netmgt.utils.SingleResultQuerier;
        import org.opennms.netmgt.events.api.model.IEvent;

                log = bsf.lookupBean("log");
                snmpTrapHelper = new org.opennms.netmgt.scriptd.helper.SnmpTrapHelper();

        server(12345);
        source("/opt/opennms/etc/scriptd-event-forwarder.bsh");

        </start-script>

        <stop-script language="beanshell">
```

- Trigger JSR-223-compatible scripts with events
- Supported languages include:
  - Beanshell
  - Groovy
  - Javascript
  - Python 2 (Jython – BYOJAR)
  - Ruby (JRuby – BYOJAR)

Facts

- Available since: 1.0
- Authn / Authz: Dealer's choice
- Performance: Single-threaded
- Schema: Stable Event bean

OpenNMS

Demo – Events OUT with Scriptd

- Copy scriptd-configuration.xml into place from `03_scriptd-configuration.xml`
  - Groovy BSF engine configured
  - Event script fires on `uei.jeffg.org/training/ecosystem/eventsOutScriptd`
  - Append to a file and `flush()`
- Reload Scriptd configuration
- Tail-follow the appender file
- Send the triggering event

OUT: Events via AMQP

- Forward events to an 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 (via plugin)

Facts

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

Alarms

Reacting to Alarms

IN: (Some) Events trigger Alarms

Facts

- Available since: 1.3
- Introduced in: NMS-????
- Authn / Authz: Event channel-based
- Performance: High
- Schema: Stable Alarm bean

- Alarms are “events that matter”
- Unlike events, they’re mutable
- Alarms anchor many workflows, integrations
- Presence of `<alarm-data>` element is key

Demo – Alarms IN

- Show `<alarm-data>` element in event config
- Show `uei.opennms.org/alarms/trigger` ; Helm / Grafana

OUT: Alarms via Northbound Interfaces (NBIs)

- Forward alarms via various protocols
  - Syslog
  - SNMP traps
  - JMS
  - AMQP
  - Others

Facts

- Available since: Varies by NBI
- Authn / Authz: Varies by NBI
- Performance: Single threaded
- Schema: Stable northbound alarm bean
- Limitations: Not aware of all alarm updates

OpenNMS

Demo: Alarms OUT via Syslog Northbounder

`04_listen_for_syslog.sh`

- Enable syslog northbounder in `/opt/opennms/etc/syslog-northbounder-configuration.xml` and restart OpenNMS
- Use netcat to listen on 514/udp:
  - `sudo nc -v -u -l -p 514`
- Trigger alarms as in previous example

Inventory

Managing nodes, interfaces, and services

IN: Inventory via REST

Facts

- Available since: 1.8.0 (Provisiond debut)
- Authn / Authz: Valid user with role
- Performance: Async handling, needs tuning for large environments
- Schema: Stable requisition schema

- Manage requisitions via REST

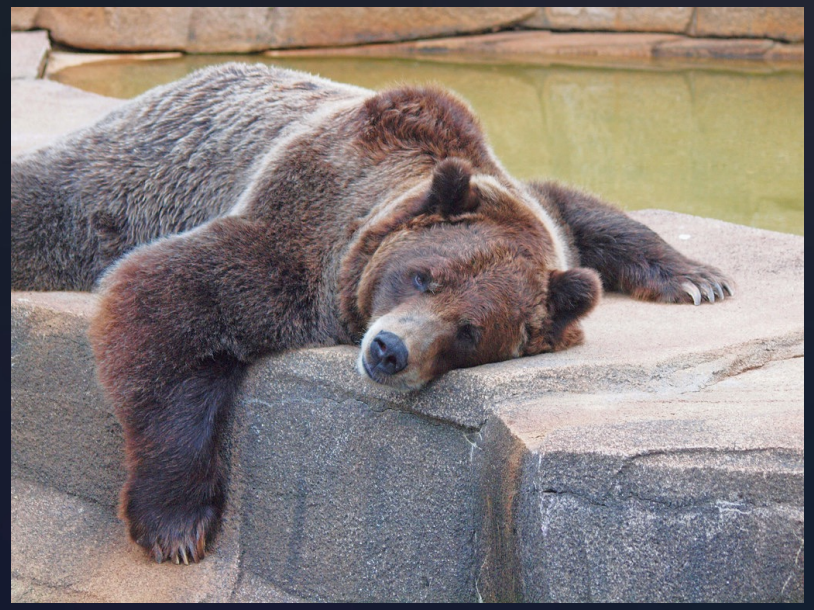
OpenNMS

Demo: Inventory IN via `onmsctl`

`05_requisitions_via_onmsctl.sh`

- Install Golang
- Clone and build **`onmsctl`** from GitHub – **`agalue/onmsctl`**
- Do the provisioning
  - `./onmsctl provision requisition list`
  - `./onmsctl provision requisition add DevTraining`
  - `./onmsctl provision node add DevTraining node001 --label test-node`
  - `./onmsctl provision interface add DevTraining node001 192.168.115.1`
  - `./onmsctl provision service add DevTraining node001 192.168.115.1 ICMP`
  - `./onmsctl provision requisition list DevTraining`
  - `./onmsctl provision requisition import DevTraining`
  - `./onmsctl provision requisition list DevTraining`
- Peep the web UI

OUT: Inventory via REST



Him doin a REST

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

Facts

- Available since: 1.8
  - v2 API since 21.0.0
- Authn / Authz: Valid user with role
- Performance: Database bound
- Schema: None

Demo: Inventory OUT via REST

`06_inventory_out_via_rest.sh`

- `curl -u admin:admin http://127.0.0.1:8980/opennms/rest/nodes`

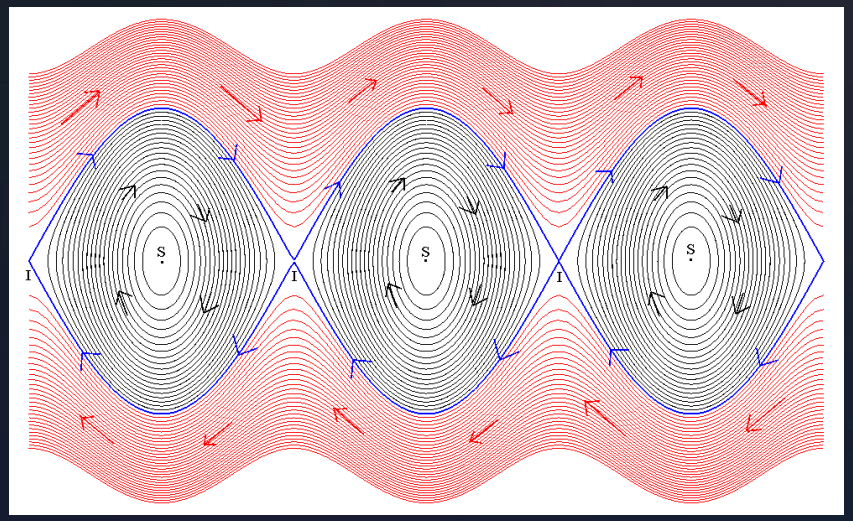
IN: Streaming Telemetry

- Support for multiple streaming protocols
  - NX-OS (Cisco)
  - JTI (Juniper)
  - sFlow telemetry
  - Graphite
- Scalable processing added to Horizon 23+ (Sentinel + Newts)

Facts

- Available since: 21.0.0
- Authn / Authz: None
- Performance: Fast!
- Schema: Varies by protocol

IN: Flow Ingest



- Supports multiple flow protocols:
  - NetFlow v5, v9
  - IPFIX
  - sFlow (sampled flows)
- Scalable processing with Sentinel

Facts

- Available since: 24.0.0
- Authn / Authz: None
- Performance: Fast!
- Schema: Varies by protocol

Sorry, no demo for Streaming Telemetry or Flows

We're already short on time.

OpenNMS

OUT: TCP Exporter

PerformanceData.proto — ~/git/horizon-work/opennms-rrd/opennms-rrd-tcp/src/main/proto

PerformanceData.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
```

Facts

- Available since: 1.7.9
- Authn / Authz: None
- Performance: Fast!
- Schema: Protobuf

- Send RRD updates over a TCP socket

Sorry, no demo for TCP Export

I’m amazed we made it this far.

OpenNMS

OUT: Kafka Producer

- Stream all the data
  - Consistent interface for events, alarms, inventory, and performance data
- Stable API and model thanks to Protobuf
  - Enables compact transmission
  - We can add fields without breaking apps
- Supports many consumers
  - Many applications can subscribe same topics
- Scale
  - Scale up your Kafka cluster as needed

Facts

- Available since: ~23.0.0 (OIA plugin)
- Authn / Authz: Broker-based
- Performance: Fast as your Kafka can take
- Schema: Stable beans

- Everybody loves Kafka

Sorry, no demo for Kafka Producer

There's no way we got this far without skipping a bunch.

Q&A

Find me on Mattermost `@jeffg`.