Model-driven automation

Anees Shaikh (Google), Rob Shakir (Jive Communications)

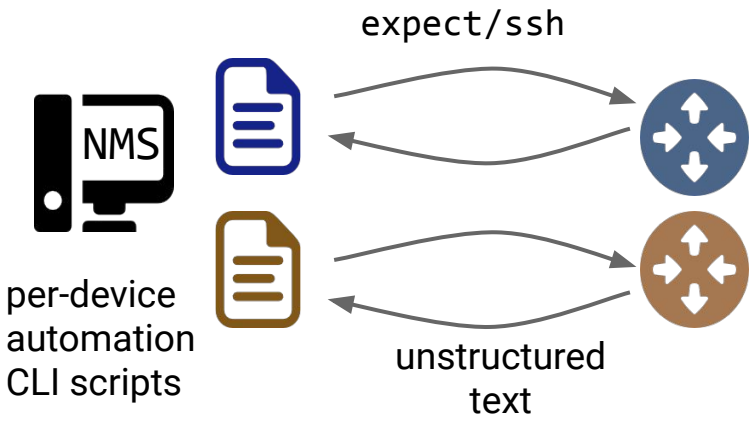
On behalf of 

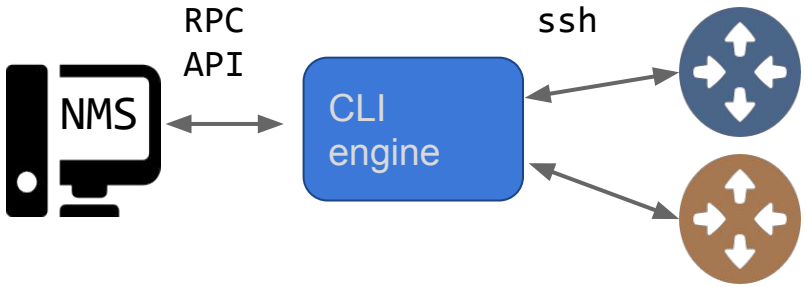
www.openconfig.net

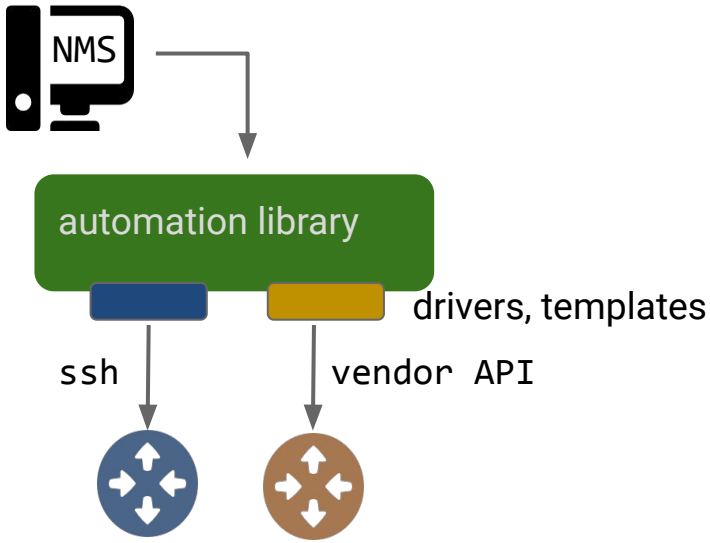
Goals for this talk …

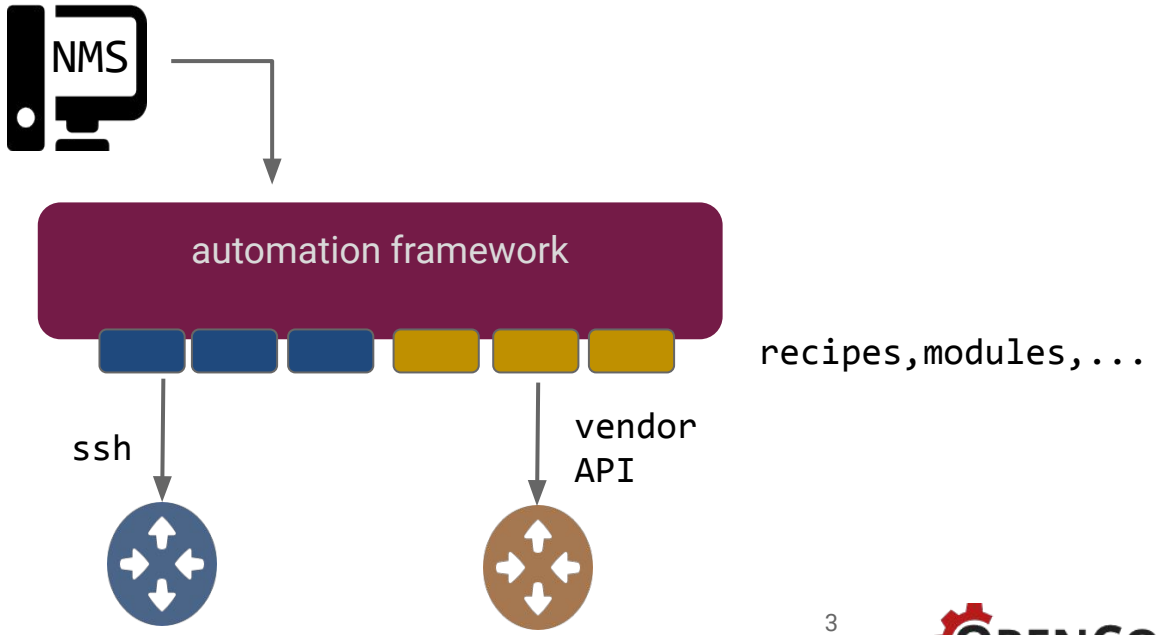
- Motivation
  - what's the value of model-based automation?
- Overview of the OpenConfig working group
  - come work with us!
- OpenConfig in a real network management stack
  - leveraging open source tools with OpenConfig

Network automation has come a long way...









3

....but there's still work to do

- automation frameworks are great, but mostly just type CLI commands faster
  - fundamental issue of multiple incompatible interfaces for the same things remains
- significant amount of proprietary integration to write and maintain
  - true even if vendors provide the "drivers", modules, recipes, playbooks, etc.
- introducing new platforms is considerable amount of new development work
- what about visibility and monitoring ?
  - SNMP still the state of the art -- despite scaling limitations, proprietary MIBs, legacy implementations, rigid structure, poor support for discovery, ...

4

Model-based automation

- durable APIs for managing and monitoring the network
- management abstraction layer (insulate from lower level details)
- forward compatibility with new platforms and technologies
- establishes a contract between NMS and infrastructure

5

OpenConfig: user-defined APIs

- industry collaboration among network operators
- data models for configuration **and** operational state, code written in YANG
- organizational model: informal, structured like an open source project
- development priorities driven by operator requirements
- engagements with major equipment vendors to drive **native** implementations
- engagement with standards (IETF) and OSS (OSR - quagga, ODL, ONOS, goBGP)

OPENCONFIG

OpenConfig's progress

Published OpenConfig models

- BGP
- routing policy
- locally generated routes
- interfaces, IP, VLANs
- MPLS, RSVP / TE
- networking / forwarding instances, VRFs
- RIB contents
- terminal optics
- streaming telemetry configuration
- top-level device structure
- system inventory / hardware

Models in development / review

- system management
- ACLs
- line optics (EDFAs and ROADMs)
- IS-IS
- QoS
- tunnels / encapsulation
- LLDP
- FIB / LFIB

OPENCONFIG

OpenConfig and telemetry support on vendor boxes

- initial versions of streaming telemetry available in 2016
  - Cisco Streaming Telemetry (IOS-XR)
  - Juniper JUNOS Telemetry Interface
  - additional vendors in progress
- OpenConfig BGP+policy model configuration native support
  - Cisco IOS-XR
  - Juniper JUNOS
  - Arista EOS
  - additional vendors with implementations underway
- implementations in progress for interfaces, MPLS / TE models
- operational state models being delivered as part of streaming telemetry

We have (many, many) models...now what?

Models are the (relatively) easy part -- how do users consume them?

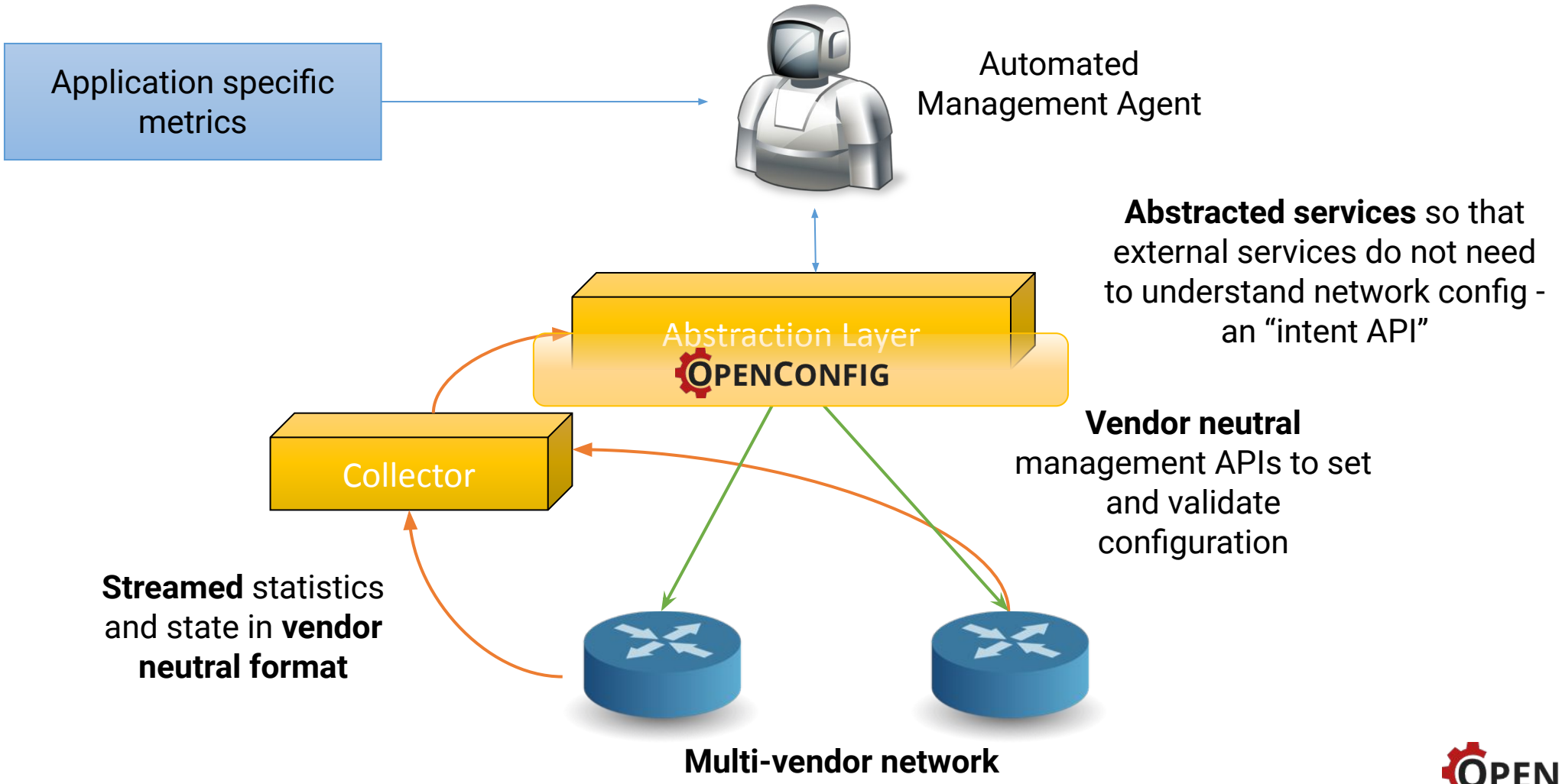
Option 1: use a commercial management stack (very few options)

- ISV-developed models, proprietary integrations with devices
- device support, vendor-neutrality is dependent on ISV

Option 2: build a toolchain and operational model around open source models

- engage with vendors on native support for open APIs and streaming telemetry
- develop tooling for generating and validating configuration data

Example network management architecture…



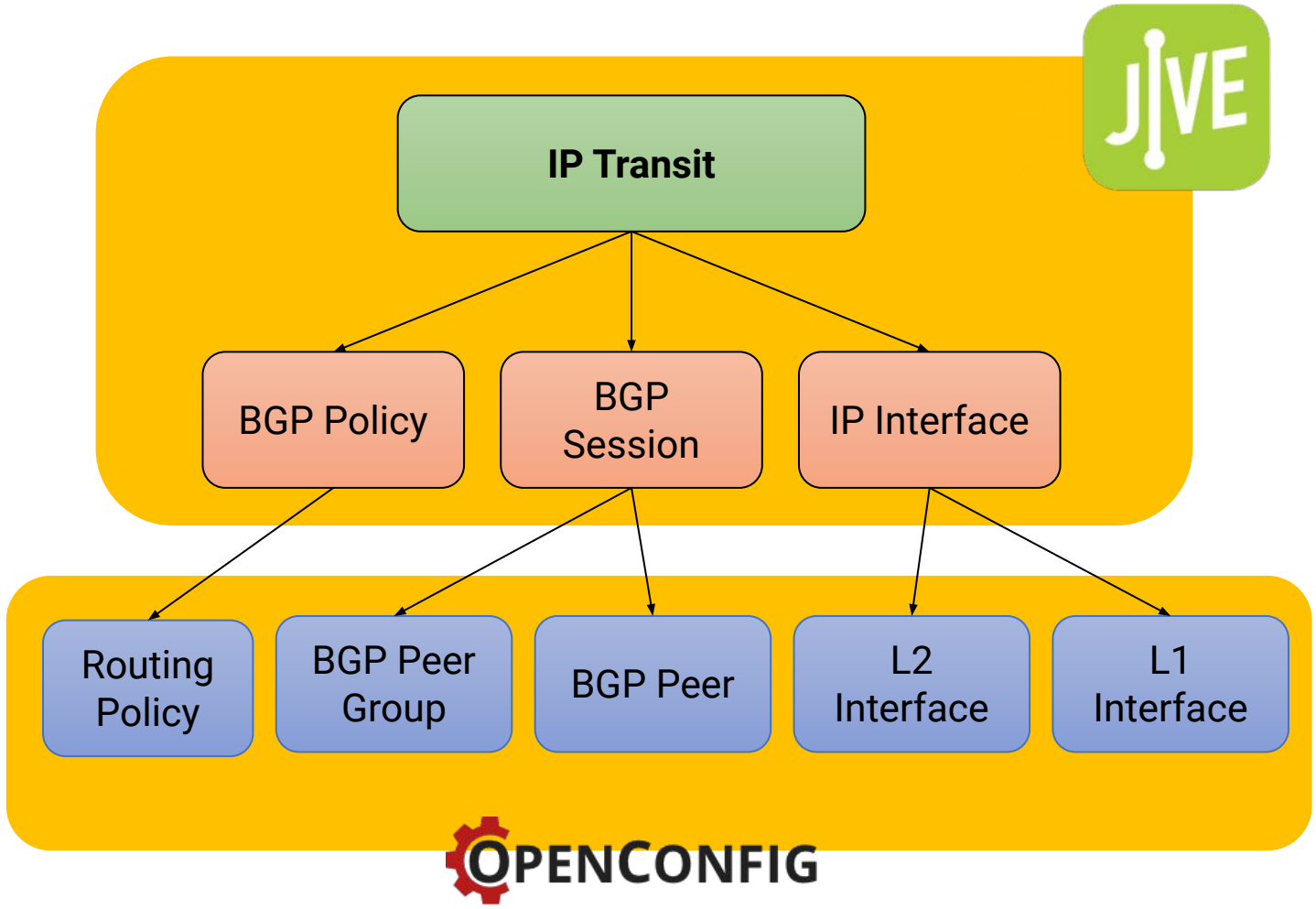
OPENCONFIG

Abstraction – using OpenConfig and YANG

Abstracted Service:
External ”application” provided by the network – addressed by operators and other infrastructure components.

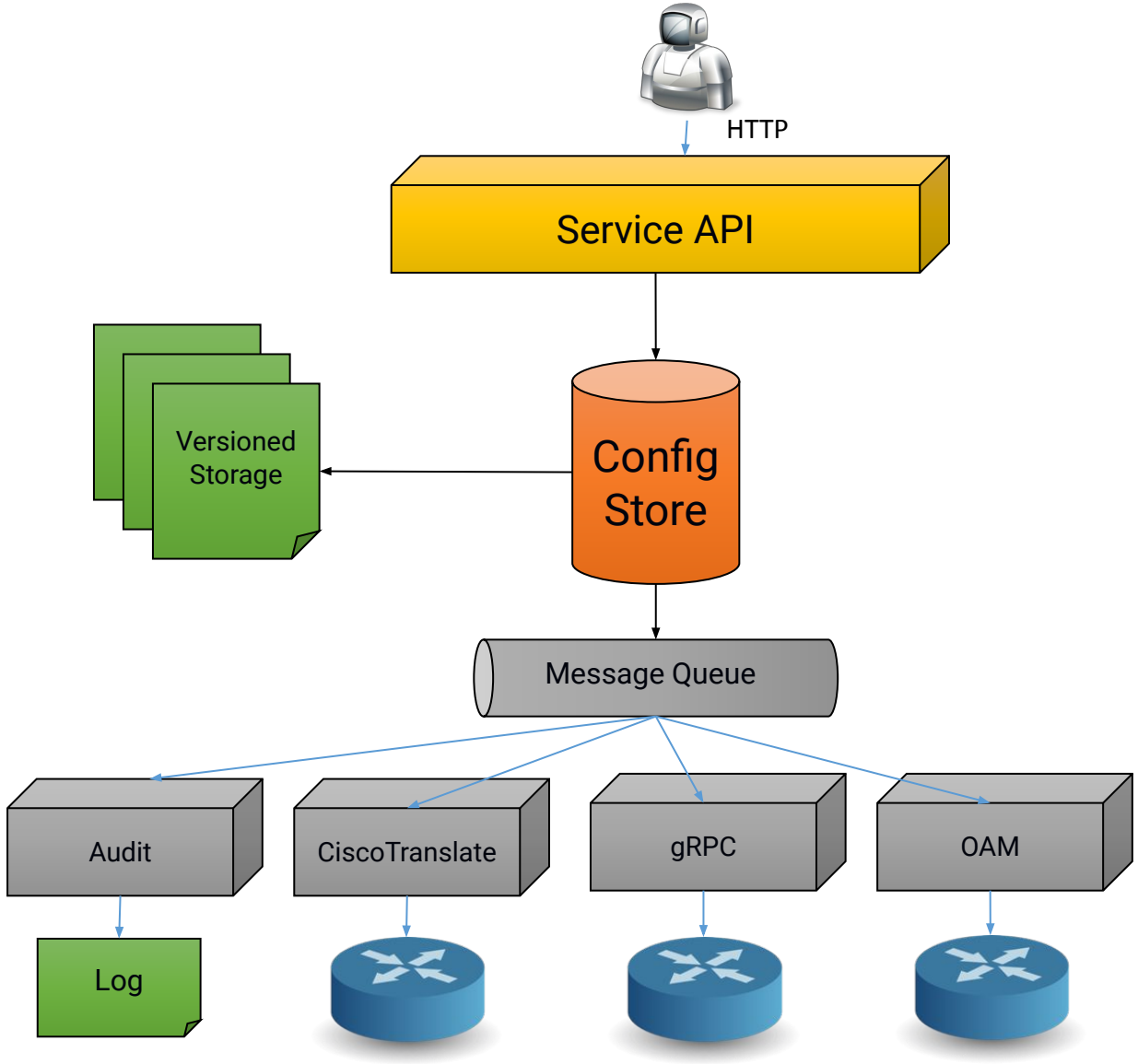
Service Components:
Re-usable network design elements – move network design from .docx -> .py

Config Primitives:
Vendor neutral blocks of configuration which are combined to make services



OPENCONFIG

Jive “Plumber” – using OpenConfig today!



Flask-RESTful bindings around **PyangBind models** - present REST API to update service models

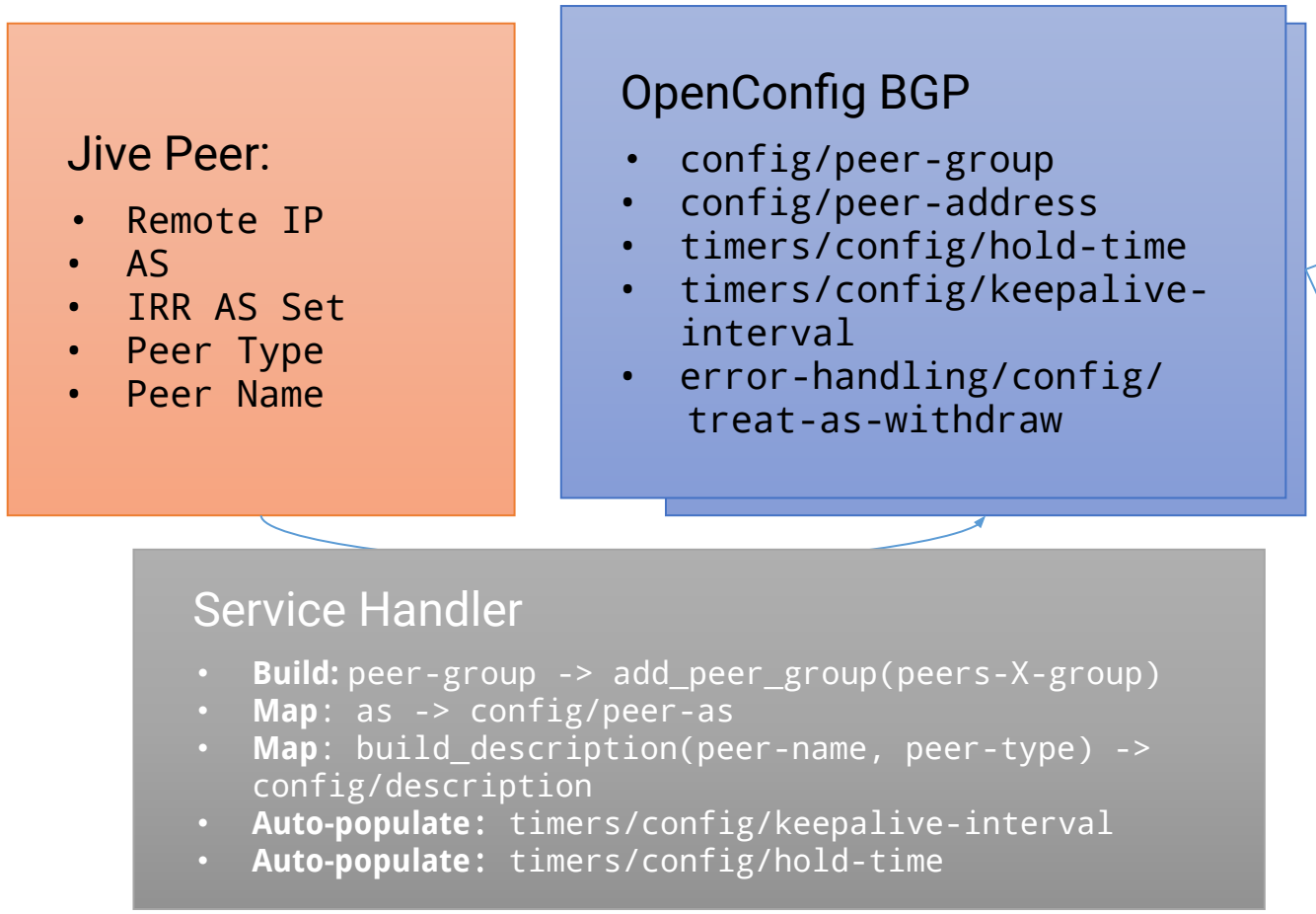
Mapping modules – take service and translate into components and config data instances.

API around **PyangBind** modules – serialised to JSON and stored in **git.**

Multiple consumers:
Audit Log – write changes to audit store
CiscoTranslate – take OpenConfig and translate to Cisco CLI via jinja2 or Python native
gRPC – apply configuration changes to gRPC endpoints natively
OAM – take tasks related to services (e.g., ping) triggered from YANG RPCs and run

OPENCONFIG

Simple Example – how does this work?



Done by manipulating PyangBind objects:

```
peer = openconfig.bgp.neighbors.neighbor.add(remote_ip)
peer.timers.config.hold_time = 90
peer.error_handling.config.treat_as_withdraw = True
```

Cisco Config

```
router bgp {{ bgp.global.as_ }}
 template peer-policy {{ peer_group_name }}
  timers {{ hold_time }} {{
keepalive_interval }}
 …
  exit
 address-family ipv4 unicast
  neighbor {{ peer_address }} activate
```

Jinja2 templates, or Python mapping code to produce text output

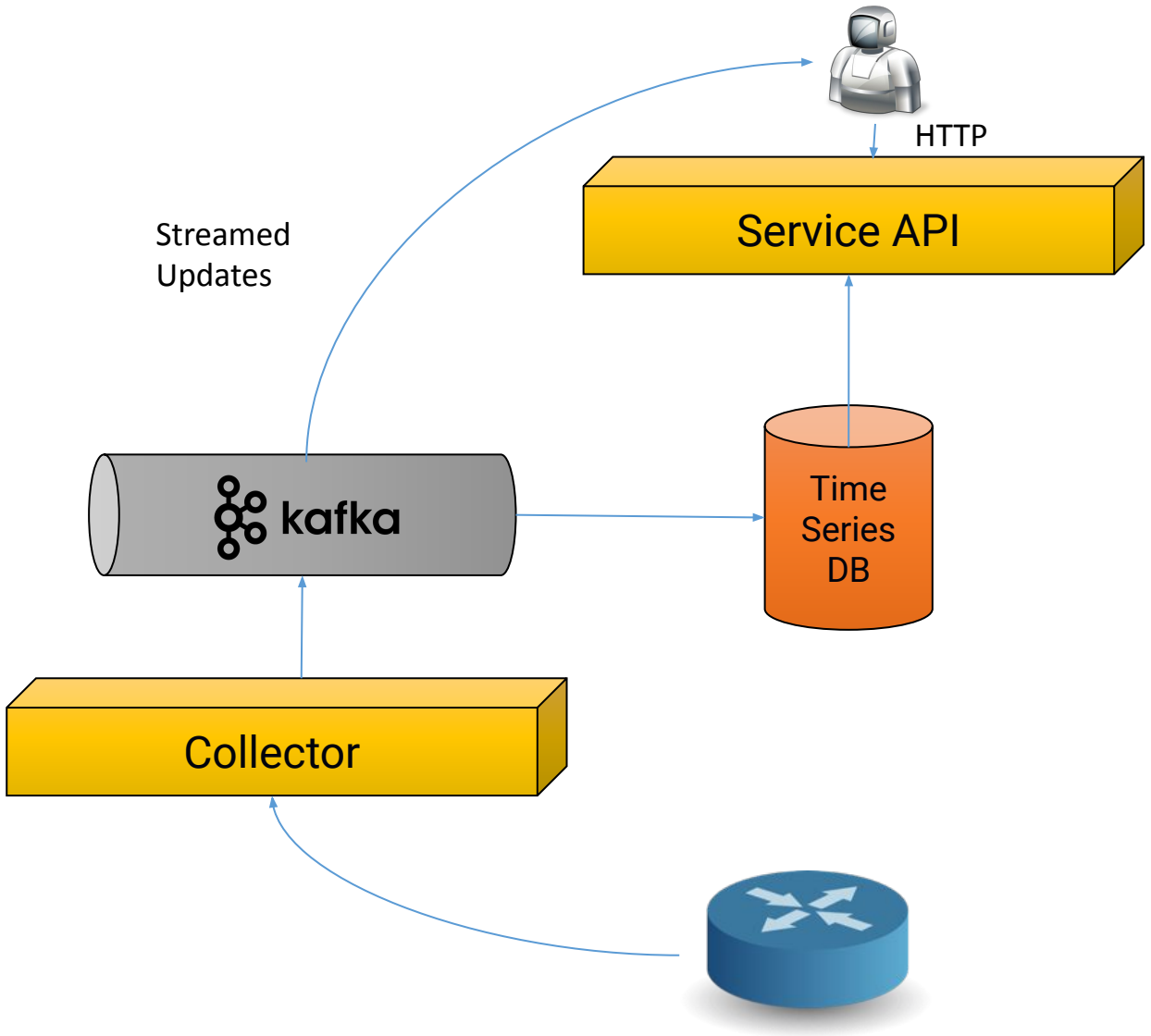
Apply JSON Directly

```
{
 “neighbors”: [
   { “neighbor”: … },
 ]
}
```

PyangBind serialise to relevant encoding (e.g., IETF JSON)

OPENCONFIG

Telemetry – streaming API with common data model



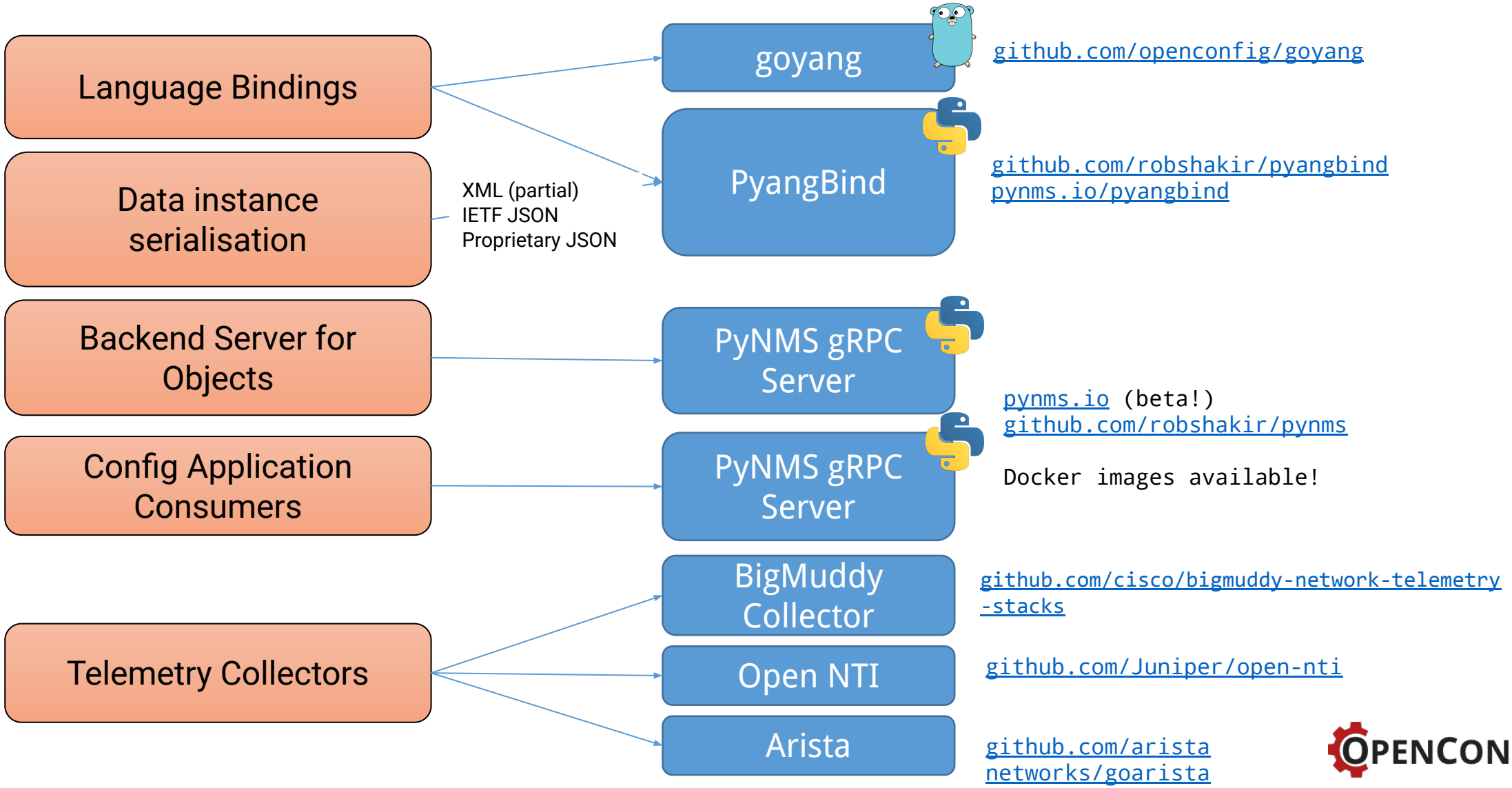
Plumber application – an external consumer can get data about *state* of service as well as *configuration* – without needing to understand the device data model

Leverage existing metrics distribution infrastructure
Kakfa – *streaming API is a producer, apps are consumers.*
Data logged into existing timeseries database (OpenTSDB, InfluxDB etc.)

Numerous options for collector – manages subscriptions to devices for timeseries data that is desired.
Cisco, Juniper, Arista all have open source collector projects.

OPENCONFIG

Open source tooling exists for OpenConfig today



Summary

Data Models

Good traction on modelling operationally important functions.

Vendor implementations emerging.

More operators involved in this effort means more vendor support, better models, and wider coverage – consider contributing!

Tooling

A range of open source tooling exists, allowing NMSes supporting OpenConfig to be put together.

Consider trying them out, and contributing or reporting issues where you come across them!

We believe we should target building modern management interfaces for the network, rather than emulate humans with automation tools.