

Juniper Event Driven Infrastructure OpenConfig telemetry demo

Khelil Sator
ksator@juniper.net
November 2017

OVERVIEW OF THE DEMO

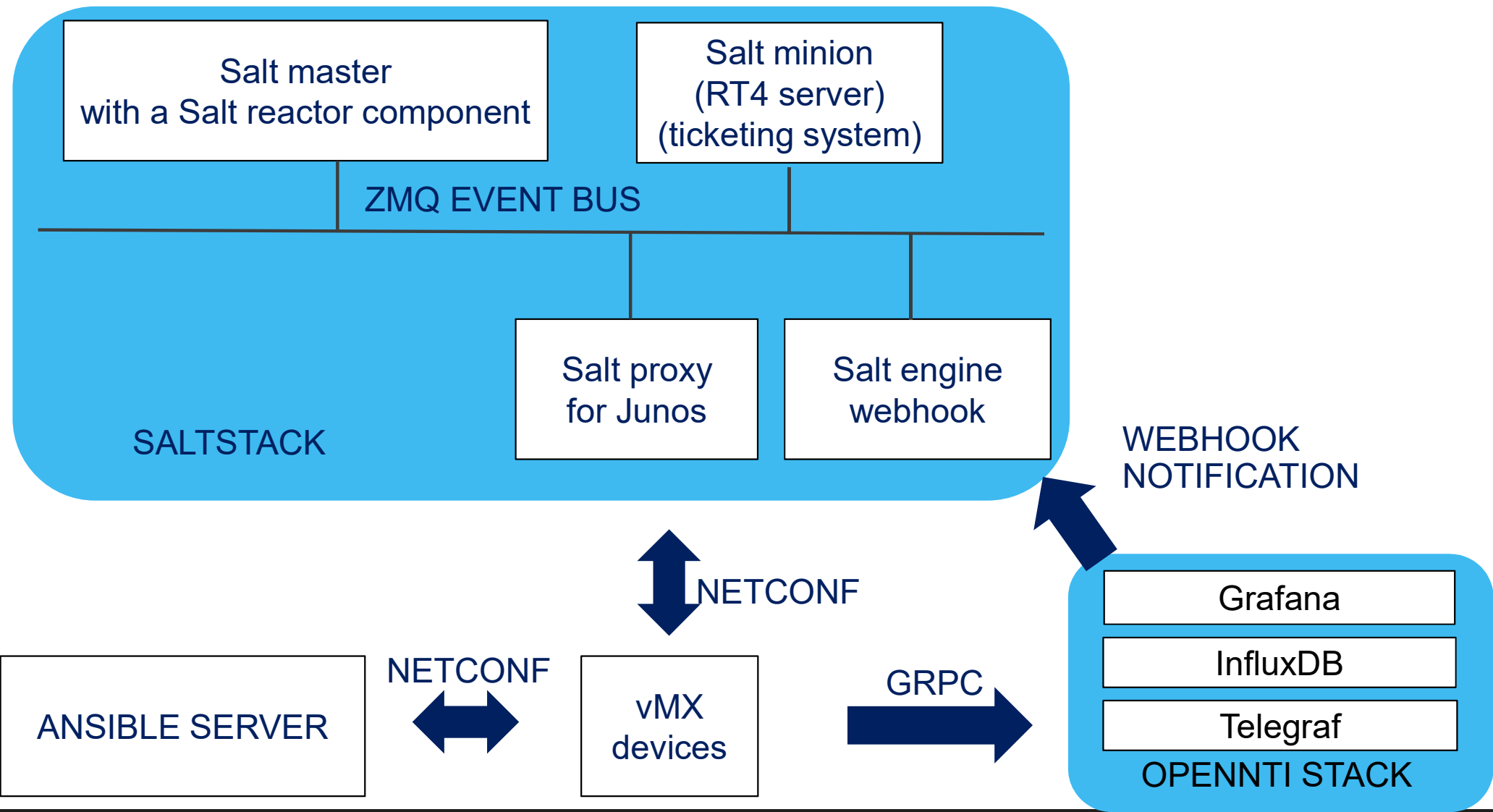
- OpenConfig configuration on vMX devices using Ansible
 - Configuration generation and deployment
 - Operational states audit
- OpenConfig streaming telemetry using gPRC with vMX devices and OPENNTI
- Webhook notifications from OPENNTI to SaltStack
- SaltStack orchestration to create automatically an RT4 ticket

About Juniper Event Driven Infrastructure (J-EDI)

- Uses regular/native SaltStack building blocks
 - Salt master, minions, event bus, reactor, ...
 - Salt proxies for Junos (Juniper contribution to SaltStack)
 - Salt sls and execution modules for Junos (Juniper contribution to SaltStack)
 - Salt junos_syslog engines (Juniper contribution to SaltStack)
- Uses the event bus as an automation backplane
- Loosely couples a growing collection of open-source and Juniper maintained tools
- Is developed by Juniper. Is installed, configured, and supported by Juniper Professional Services.

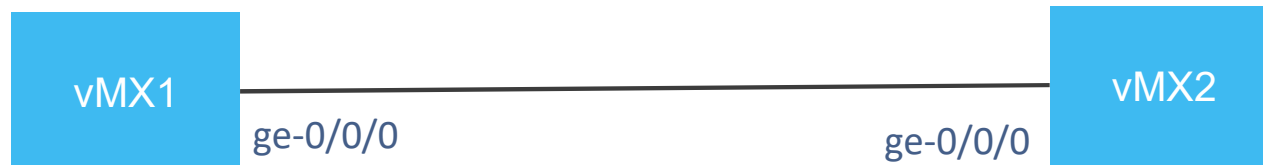
J-EDI PLUGINS USED FOR THIS DEMO

- Request Tracker 4 plugin
 - RT4 is a popular ticketing system.
 - It has an API that can be used for the CRUD operations against tickets.
 - J-EDI interacts with RT4 using its API to automate tickets manipulation (tickets creation and update)



LAB TOPOLOGY

- The 2 vMX devices are connected between them



- JUNOS has:
 - Netconf server
 - OpenConfig support
 - gRPC server

```
lab@dc-vmx-2> show configuration system services netconf | display set
set system services netconf ssh
lab@dc-vmx-2> show version | match "openconfig|telemetry"
JUNOS na telemetry [17.2R1-S2.1-C1]
JUNOS Openconfig [0.0.0.4]
```

OPENCONFIG CONFIGURATION USING ANSIBLE

- Generates the OpenConfig configuration for each Junos device
 - Rendering a Jinja template
- Deploys the OpenConfig on Junos devices
 - using the Ansible module `junos_template`
- Audits the operational states on Junos devices
 - using the Ansible module `junos_command`
- Automation content <https://github.com/ksator/openconfig-demo-with-juniper-devices>
- Documentation <https://github.com/ksator/openconfig-demo-with-juniper-devices/wiki>

OPENCONFIG CONFIGURATION USING ANSIBLE

- Get the remote repository content locally

```
git clone https://github.com/ksator/openconfig-demo-with-juniper-devices.git
cd openconfig-demo-with-juniper-devices/
```

- Run the playbook in dry-run to see which devices would change

```
ansible-playbook pb.conf.bgp.oc.yaml --check --tag 'configuration'
```

- Add the flag diff to see which configuration changes would happen on a device

```
ansible-playbook pb.conf.bgp.oc.yaml --check --diff --limit dc-vmx-1 --tag 'configuration'
```

- Execute the ansible playbook

- to generate and deploy the openconfig configuration on junos device.
- to audit the operational states
 - compare the actual state against the desired state (session state should be established)

```
ansible-playbook pb.conf.bgp.oc.yaml
```


MANUAL VERIFICATIONS ON JUNOS DEVICES

- Display the commit history

```
lab@dc-vmx-2> show system commit
0    2017-11-24 12:37:27 UTC by lab via netconf
    OC BGP configuration from Ansible
```

- Print the changes between the two last commit

```
lab@dc-vmx-2> show configuration | compare rollback 1
```

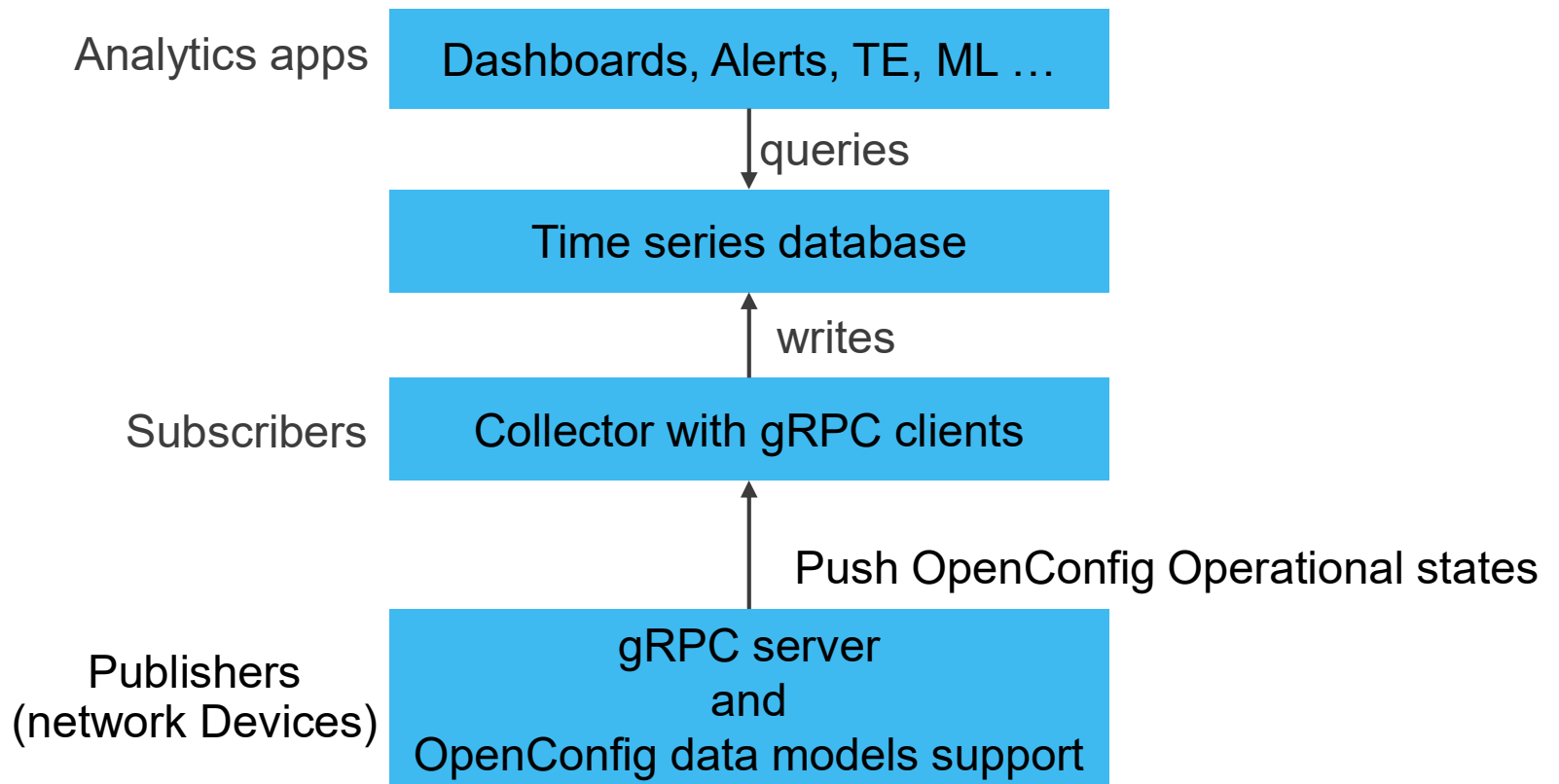
- Print the OpenConfig BGP running configuration on the Junos device

```
lab@dc-vmx-2> show configuration openconfig-bgp:bgp | display json
```

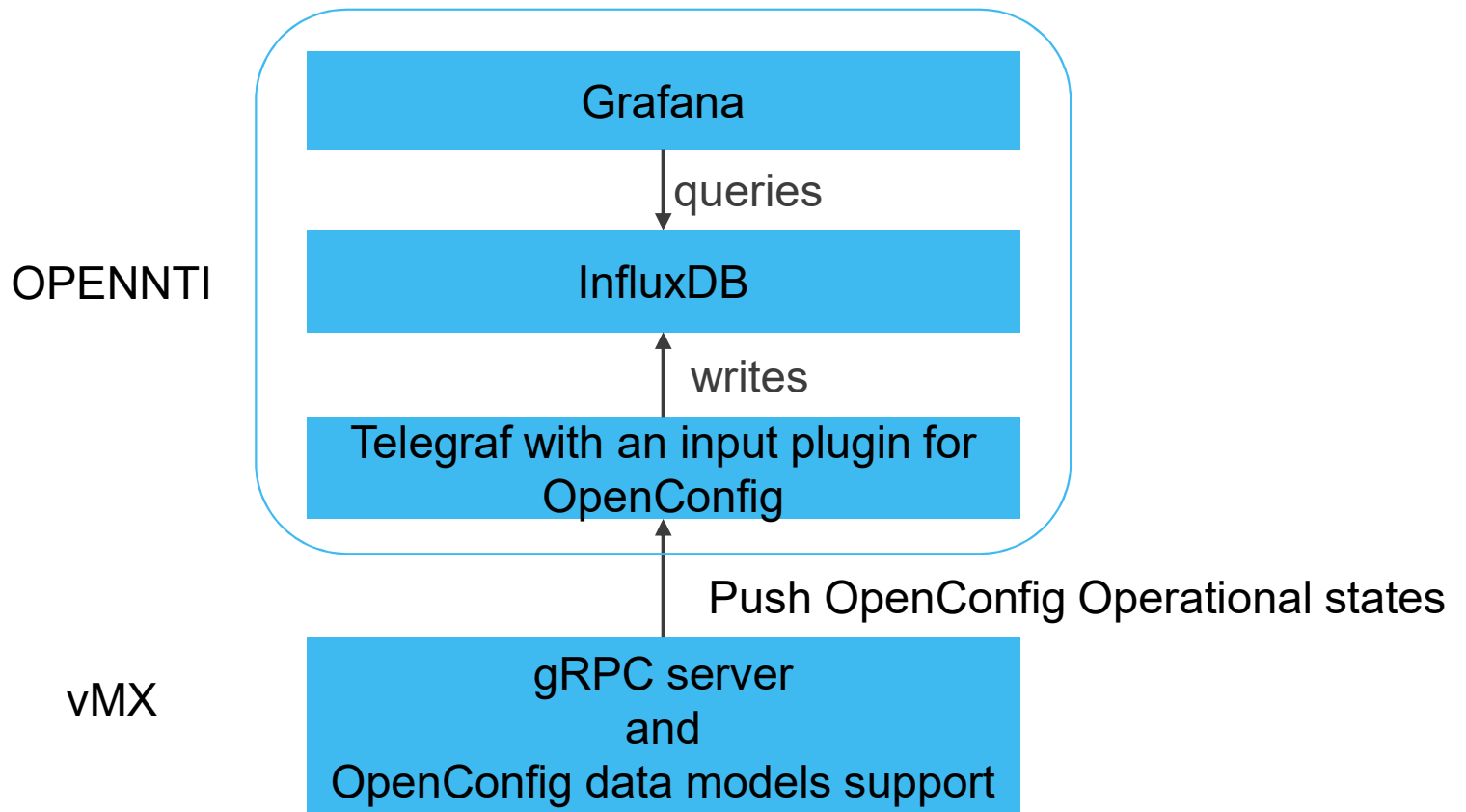
- Validate the BGP operational state

```
lab@dc-vmx-2> show bgp neighbor
```

OpenConfig streaming telemetry overview



OPEN-NTI STACK USED FOR THIS DEMO



OPEN-NTI STACK USED FOR THIS DEMO

- OpenNTI is a multi-containers application to collect and visualize time series data from network devices.
 - It is available <https://github.com/Juniper/open-nti>
- Telegraf
 - Plugin-driven collector
 - OpenConfig telemetry input plugin (gRPC collector subscribes to OpenConfig operational states on vMX devices)
 - InfluxDB output plugin (Telegraf writes data to InfluxDB)
- InfluxDB
 - time series database
- Grafana
 - Dashboards application
 - Queries InfluxDB to get the data
 - generates graphs
 - Triggers webhook notifications when an alert change state
 - HTTP POST with JSON body to SaltStack

GRPC CONFIGURATION

- grpc servers
 - The grpc service is configured on vMX
- grpc client
 - The telegraf input plugin for OpenConfig is configured to subscribe:
 - To the sensor BGP
 - To grpc servers (the vMX devices).
 - With a frequency of 3000 ms

=> The vMX will stream BGP operational states using the OpenConfig data model to telegraf every 3000 ms

START OPEN-NTI STACK

- Print the running containers
 - OPEN-NTI containers are not running

```
# docker ps
```

- Set the environment variables from a file

```
# source open-nti.params
```

- Run a multi-container applications

```
# docker-compose -f docker-compose.yml up -d
```

- Print the running containers
 - OPEN-NTI containers are running

```
# docker ps
```

InfluxDB queries from web interface

- InfluxDB has API, CLI and web interface.
- You can make queries for interacting with data in InfluxDB.
- Examples to get data using InfluxDB web interface:

```
SHOW MEASUREMENTS
SHOW TAG VALUES FROM "/bgp" WITH KEY = "device"
SHOW TAG VALUES FROM "/bgp" WITH KEY = "/bgp/neighbors/neighbor/@neighbor-address"
SHOW TAG VALUES FROM "/bgp" WITH KEY = "/bgp/neighbors/neighbor/@neighbor-address"
WHERE device='172.30.52.86'
SHOW TAG VALUES FROM "/bgp" WITH KEY = "/bgp/peer-groups/peer-group/@peer-group-
name"
SELECT * FROM "/bgp" WHERE device='172.30.52.86' limit 10
SELECT * FROM "/bgp" WHERE "/bgp/neighbors/neighbor/@neighbor-address"
='192.168.1.1' limit 10
```



Query: SHOW TAG VALUES FROM "/bgp" WITH KEY = "/bgp/neighbors/neighbor/@neighbor-address" WHERE device='172.30.52.86'

Generate Query URL

Query Templates ▾

/bgp

key	value
/bgp/neighbors/neighbor/@neighbor-address	"192.168.1.1"

Grafana

- Grafana uses dashboards composed of individual graphs.
- Each graph queries data from the configured Grafana Data Source
- This demo has a dashboard with 4 graphs.
 - Each graph make queries to InfluxDB to get OpenConfig states streamed by Junos devices to telegraf

devices

172.30.52.85

commit



/bgp/global/state/total-prefixes



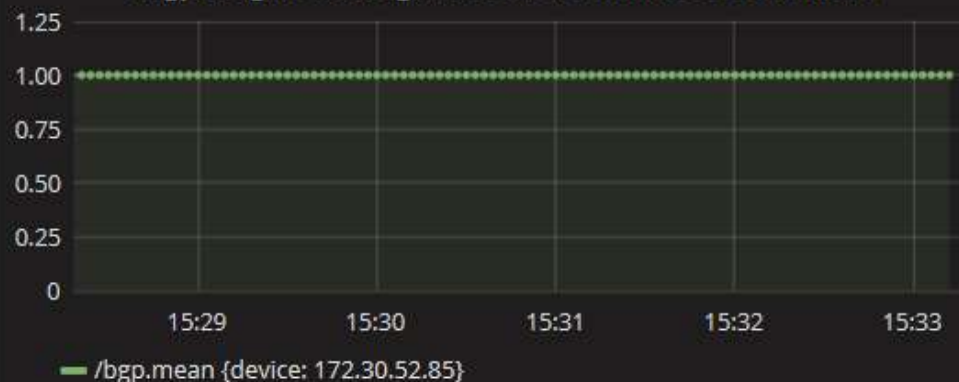
/bgp/neighbors/neighbor/state/session-state ESTABLISHED



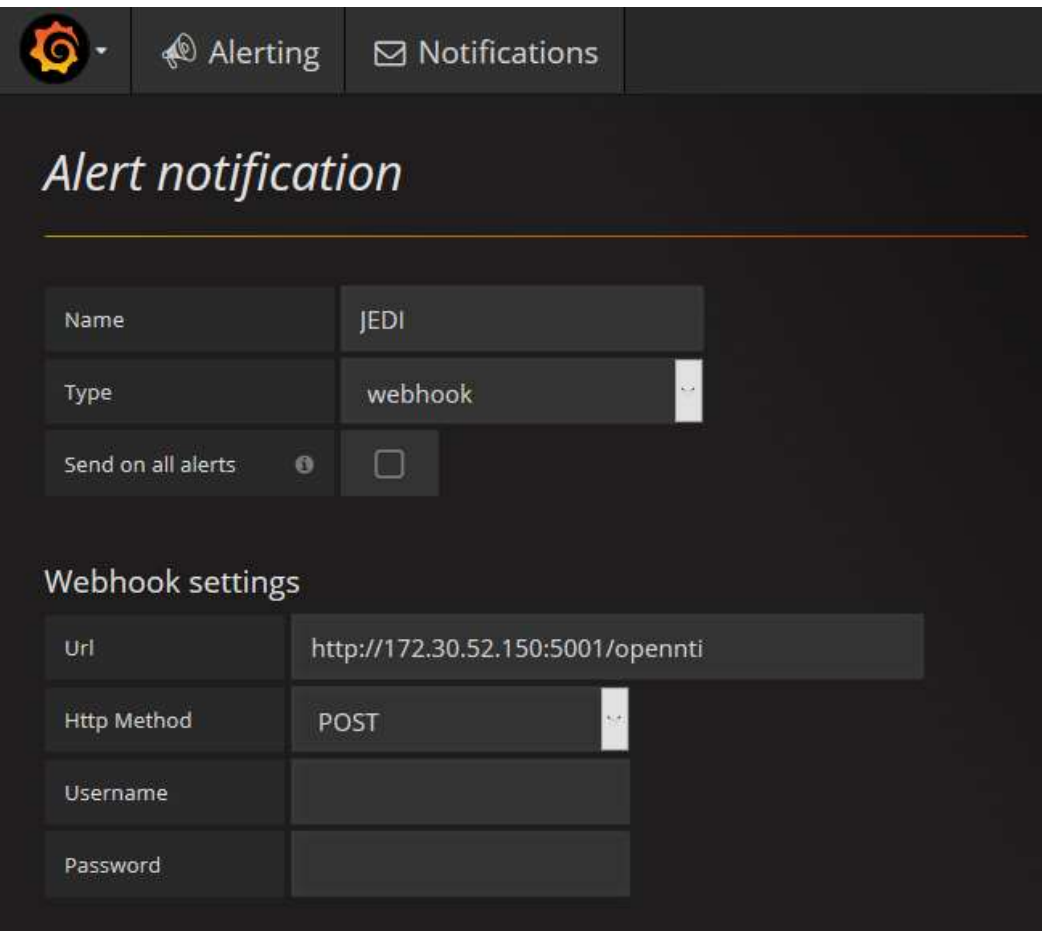
/bgp/neighbors/neighbor/state/peer-type EXTERNAL



/bgp/neighbors/neighbor/state/established-transitions



Grafana: Notifications for alerts

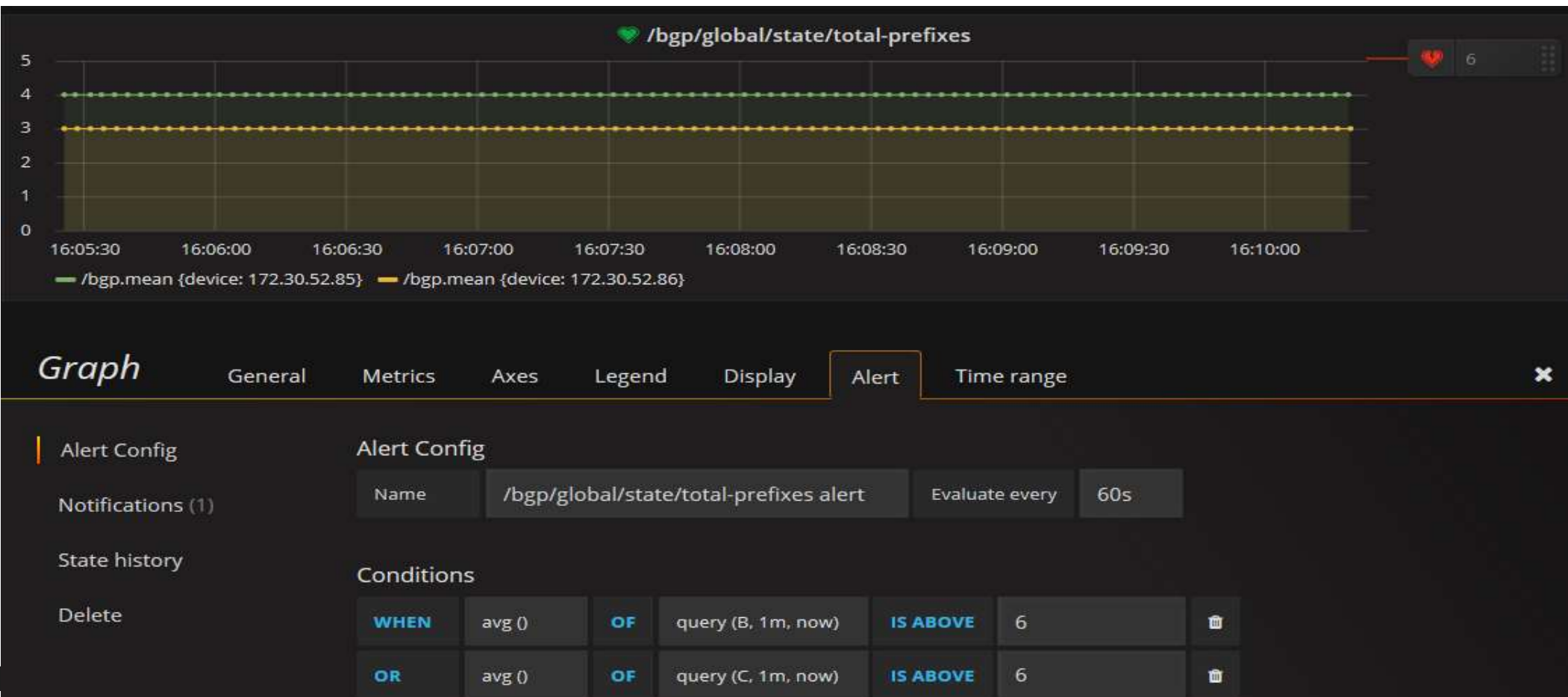


The screenshot shows the Grafana Alerting configuration interface. At the top, there are tabs for 'Alerting' and 'Notifications'. The 'Alert notification' section is active, showing details for a notification named 'JEDI'. The 'Type' is set to 'webhook'. There is a checkbox for 'Send on all alerts' which is currently unchecked. Below this, the 'Webhook settings' section is visible, containing fields for 'Url', 'Http Method', 'Username', and 'Password'. The 'Url' field is populated with 'http://172.30.52.150:5001/opennti' and the 'Http Method' is set to 'POST'.

Field	Value
Name	JEDI
Type	webhook
Send on all alerts	<input type="checkbox"/>
Webhook settings	
Url	http://172.30.52.150:5001/opennti
Http Method	POST
Username	
Password	

- When an alert changes state, Grafana uses a notification.
- This notification uses a webhook to notify SaltStack
 - HTTP POST with a JSON body to <http://172.30.52.150:5001/opennti>

Grafana: Alert definition for a graph



LET'S TRIGGER A NOTIFICATION

- Lets use SaltStack to change the vMX2 configuration in order to have the vMX2 to advertise more BGP routes to vMX1
 - So the vMX1 will learn more BGP routes
 - This will change the alert state for the graph /bgp/global/state/total-prefixes
 - This will trigger the notification (webhook to SaltStack)

```
salt 'dc-vmx-2' state.apply junos.routes_to_propagate
```

```
lab@dc-vmx-2> show system commit
0    2017-11-24 15:32:39 UTC by SaltStack via netconf
    configured the model routes_to_propagate using SaltStack
```

♥ /bgp/global/state/total-prefixes



mgd

UI_COMMIT_COMPLETED: commit complete

dc-vmx-2

2017-11-24 16:32:39

/bgp/neighb

1.25

SALTSTACK API FOR WEBHOOK NOTIFICATIONS

- Grafana notifications use HTTP POST to <http://172.30.52.150:5001/opennti>
- SaltStack listens for webhook notifications on port 5001, and send an equivalent ZMQ

```
# more /etc/salt/master
...
engines:
  - webhook:
      port: 5001
...
```

- So the Salt engine 'webhook' generates and publishes to the event bus a ZMQ message
 - with the topic 'salt/engines/hook/opennti'
 - and a JSON body that has the same content as the webhook.

SALTSTACK REACTOR

```
# salt-run reactor.list
...
  salt/engines/hook/opennti:
    - /srv/reactor/create_opennti_ticket.sls
...
```

- The SaltStack reactor is subscribing to the topic 'salt/engines/hook/opennti'
- If a ZMQ message 'salt/engines/hook/opennti' is published on the event bus, SaltStack will execute the state file `create_opennti_ticket.sls`
 - This will create an RT4 ticket if it receives

RT4 TICKET CREATED AUTOMATICALLY BY J-EDI

172.30.52.150:9081

Search

Home

Search

Reports

Articles

Assets

Tools

Admin

Logged in as root

RT for example.com

REQUEST TRACKER

RT at a glance

New ticket in

General

Search...

Edit

10 highest priority tickets I own

Edit

10 newest unowned tickets

Edit

#	Subject	Queue	Status	Created	
20	Alert from Open-NTI for /bgp.mean { device: 172.30.52.85 }	General	new	38 minutes ago	Take

My reminders

Edit

Queue list

Edit

Queue	new	open	stalled
General	19	-	-

RT4 TICKET DETAILS

⬅️ ➡️ ↺️ 🏠

172.30.52.150:9081/Ticket/Display.html?id=20

📄 ⋮ 🛡️ ☆

🔍 Search

⬇️ 📑 📄

☰

Display History Basics People Dates Links Jumbo Reminders Actions ⌵ ☆ ⌚

^ History

Show all quoted text — Show full headers

Fri Nov 24 11:42:05 2017

root (Enoch Root) - Ticket created

Reply Comment Forward

From: root@localhost

Subject: Alert from Open-NTI for /bgp.mean { device: 172.30.52.85 }

Alert name is '/bgp/global/state/total-prefixes alert'. Metric is '/bgp.mean { device: 172.30.52.85 }'. Recorded value is '10'

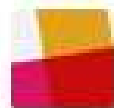
Download (untitled)

with headers

text/plain 126B

LETS RESTORE THE VMX2 PREVIOUS CONFIGURATION

- Lets use a chatbot to restore the vMX2 previous configuration
 - For more information about how to delegate junos automation tasks chatting to a bot with slack, you can visit this repository <https://github.com/ksator/junos-automation-with-chatops>



ksator 3:14 PM

@j-bot dev=dc-vmx-2 rollback 1



j-bot APP 3:14 PM

I'll take care of that right away!



Playbook pb.rollback.yml with rollback 1 APP 3:14 PM

configuration rolled back on device dc-vmx-2

💖 /bgp/global/state/total-prefixes



JUNOS AUTOMATION RESOURCES

If you are looking for more details about Junos automation, you can visit these repositories

<https://github.com/ksator?tab=repositories>

<https://gitlab.com/users/ksator/projects>

<https://gist.github.com/ksator/>

Thank you

