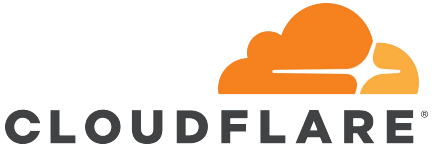


Event-driven network automation and orchestration

Mircea Ulinic
Cloudflare, London

London Network Automation Meetup
September 2017

Mircea Ulinic

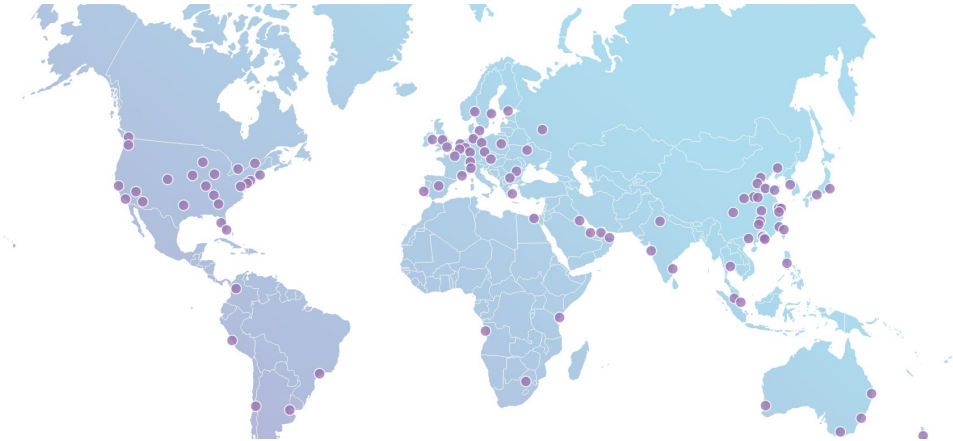
- Network engineer at Cloudflare
- Prev research and teaching assistant at EPFL, Switzerland
- Member and maintainer at NAPALM Automation
- Integrated NAPALM in Salt
- OpenConfig representative
- https://mirceaulinic.net/

 mirceaulinic

 @mirceaulinic

Cloudflare

- How big?
  - Four+ million zones/domains
  - Authoritative for ~40% of Alexa top 1 million
  - 43+ billion DNS queries/day
    - Second only to Verisign
- 100+ anycast locations globally
  - 50 countries (and growing)
  - Many hundreds of network devices



To automate, I have to learn Python or another programming language.

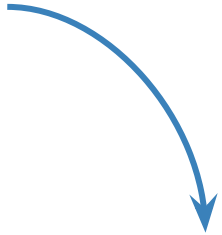
To automate, I have to learn Python or another programming language.

WRONG!

Do not jump into implementation.
Design first!

What’s the best tool?

Wrong question.



~~What's the best tool?~~

What’s the best tool for my network?

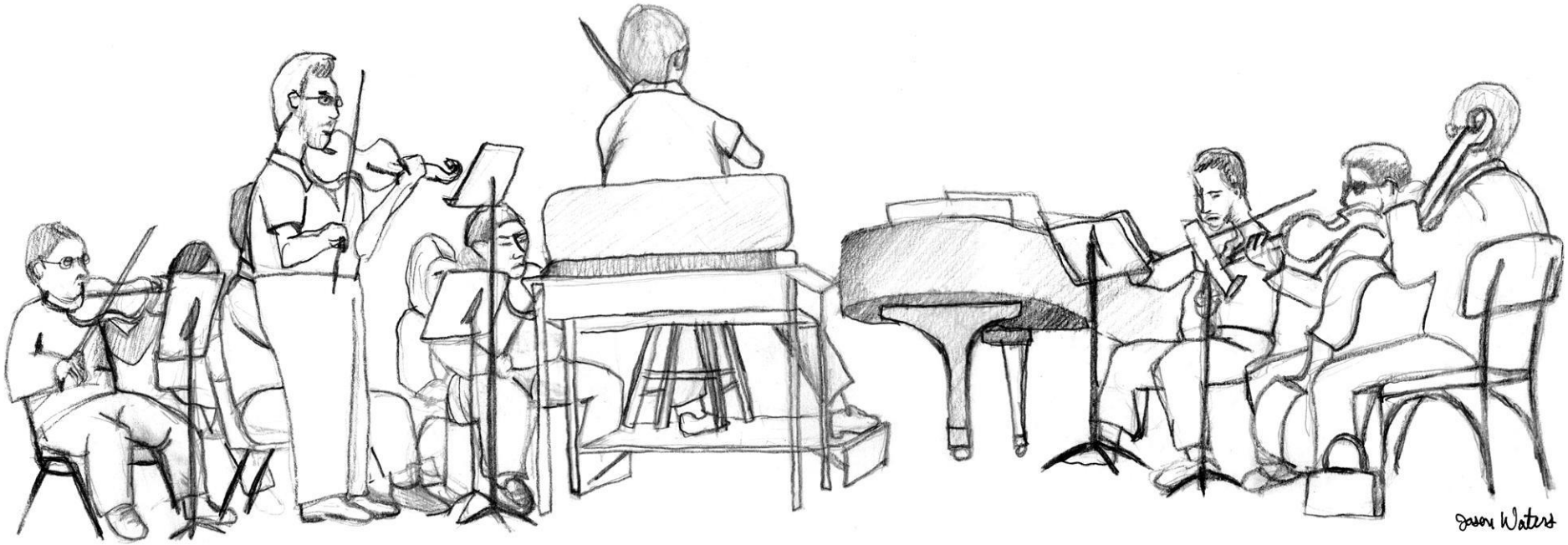
What's the best tool for my network?

- How large is your network?
- How many platforms / operating systems?
- How dynamic?
- External sources of truth? e.g. IPAM
- Do you need native caching? REST API?
- Event-driven automation?
- **Community**

Why Salt

- Very scalable
- Concurrency
- Event-driven automation
- Easily configurable & customizable
- Native caching and drivers for useful tools
- One of the friendliest communities
- Great documentation

Why Salt
Orchestration vs. Automation



Why Salt

"

In SaltStack, speed isn't a byproduct, it is a design goal. SaltStack was created as an extremely fast, lightweight communication bus to provide the foundation for a remote execution engine.

SaltStack now provides orchestration, configuration management, event reactors, cloud provisioning, and more, all built around the SaltStack high-speed communication bus.

... + cross-vendor network automation from 2016.11 (Carbon) "

https://docs.saltstack.com/en/getstarted/speed.html

Who's Salty


LinkedIn


SAP


linode


Adobe


CLOUDFLARE


COMCAST

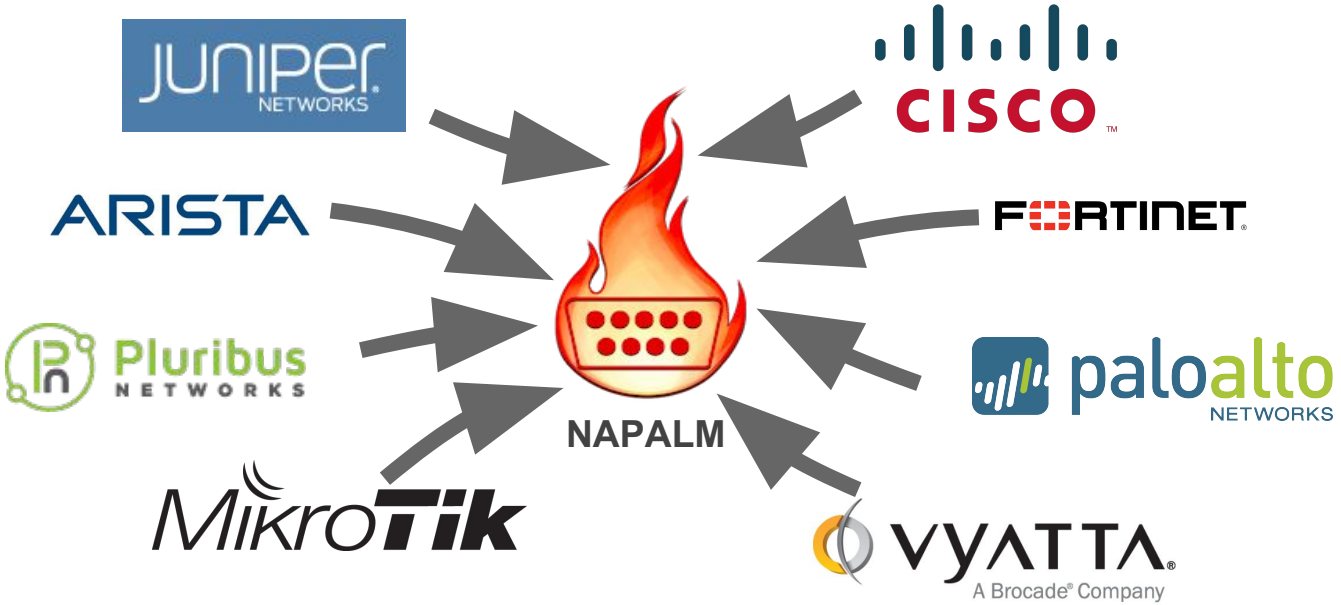

DigitalOcean


KPMG


Bloomberg

Vendor-agnostic API: NAPALM

(Network Automation and Programmability Abstraction Layer with Multivendor support)



https://github.com/napalm-automation

NAPALM integrated in Salt: Carbon

NETWORK AUTOMATION: NAPALM

Beginning with 2016.11.0, network automation is inclued by default in the core of Salt. It is based on the NAPALM library and provides facilities to manage the configuration and retrieve data from network devices running widely used operating systems such as: JunOS, IOS-XR, eOS, IOS, NX-OS etc. - see the complete list of supported devices.

The connection is established via the `NAPALM proxy`.

In the current release, the following modules were included:

- `NAPALM grains` - Select network devices based on their characteristics
- `NET execution module` - Networking basic features
- `NTP execution module`
- `BGP execution module`
- `Routes execution module`
- `SNMP execution module`
- `Users execution module`
- `Probes execution module`
- `NTP peers management state`
- `SNMP configuration management state`
- `Users management state`

https://docs.saltstack.com/en/develop/topics/releases/2016.11.0.html

NAPALM integrated in Salt: Nitrogen

Introduced in 2016.11, the modules for cross-vendor network automation have been improved, enhanced and widenened in scope:

- Manage network devices like servers: the NAPALM modules have been transformed so they can run in both proxy and regular minions. That means, if the operating system allows, the salt-minion package can be installed directly on the network gear. Examples of such devices (also covered by NAPALM) include: Arista, Cumulus, Cisco IOS-XR or Cisco Nexus.
- Not always alive: in certain less dynamic environments, maintaining the remote connection permanently open with the network device is not always beneficial. In those particular cases, the user can select to initialize the connection only when needed, by specifying the field `always_alive: false` in the `proxy configuration` or using the `proxy_always_alive` option.
- Proxy keepalive: due to external factors, the connection with the remote device can be dropped, e.g.: packet loss, idle time (no commands issued within a couple of minutes or seconds), or simply the device decides to kill the process. In Nitrogen we have introduced the functionality to re-establish the connection. One can disable this feature through the `proxy_keep_alive` option and adjust the polling frequency speciying a custom value for `proxy_keep_alive_interval`, in minutes.

New modules:

- `Netconfig state` - Manage the configuration of network devices using arbitrary templates and the Salt-specific advanced templating methodologies.
- `Network ACL execution module` - Generate and load ACL (firewall) configuration on network devices.
- `Network ACL state` - Manage the firewall configuration. It only requires writing the pillar structure correctly!
- `NAPALM YANG execution module` - Parse, generate and load native device configuration in a standard way, using the OpenConfig/IETF models. This module cotains also helpers for the states.
- `NET finder` - Runner to find details easily and fast. It's smart enough to know what you are looking for. It will search in the details of the network interfaces, IP addresses, MAC address tables, ARP tables and LLDP neighbors.
- `BGP finder` - Runner to search BGP neighbors details.
- `NAPALM syslog` - Engine to import events from the napalm-logs library into the Salt event bus. The events are based on the syslog messages from the network devices and structured following the OpenConfig/IETF YANG models.

https://docs.saltstack.com/en/develop/topics/releases/nitrogen.html

Vendor-agnostic automation (1)

```
$ sudo salt junos-router net.arp
junos-router:
    ----------
    out:
        |_
          ----------
          age:
              129.0
          interface:
              ae2.100
          ip:
              10.0.0.1
          mac:
              84:B5:9C:CD:09:73
        |_
          ----------
          age:
              1101.0
```

```
$ sudo salt iosxr-router net.arp
iosxr-router:
    ----------
    out:
        |_
          ----------
          age:
              1620.0
          interface:
              Bundle-Ether4
          ip:
              10.0.0.2
          mac:
              00:25:90:20:46:B5
        |_
          ----------
          age:
              8570.0
```

Vendor-agnostic automation (2)

```
$ sudo salt junos-router state.sls ntp
junos-router:
----------
          ID: oc_ntp_netconfig
    Function: netconfig.managed
      Result: True
     Comment: Configuration changed!
     Started: 10:53:25.624396
    Duration: 3494.153 ms
     Changes:
              ----------
              diff:
                  [edit system ntp]
                  -     peer 172.17.17.2;
                  [edit system ntp]
                  +     server 10.10.10.1 prefer;
                  +     server 10.10.10.2;
                  -     server 172.17.17.1 version 2 prefer;
```

```
$ sudo salt iosxr-router state.sls ntp
iosxr-router:
----------
          ID: oc_ntp_netconfig
    Function: netconfig.managed
      Result: True
     Comment: Configuration changed!
     Started: 11:02:39.162423
    Duration: 3478.683 ms
     Changes:
              ----------
              diff:
                  ---
                  +++
                  @@ -1,4 +1,10 @@
                  +ntp
                  + server 10.10.10.1 prefer
                  + server 10.10.10.2
                   !
```

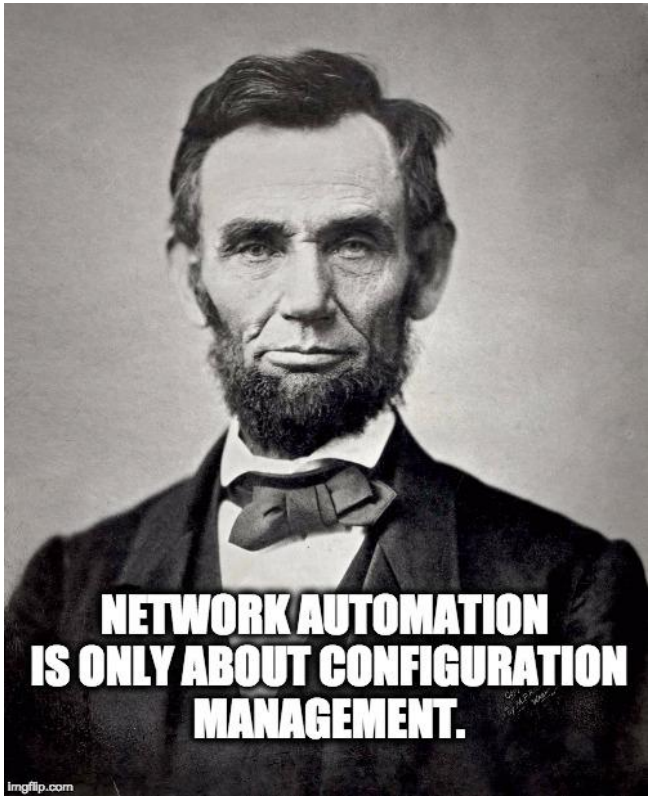
Vendor-agnostic automation: how to

- Salt in 10 minutes
- Salt fudamentals
- Configuration management
- Network Automation official Salt docs
- Step-by-step tutorial -- up and running in 60 minutes
- Using Salt at Scale

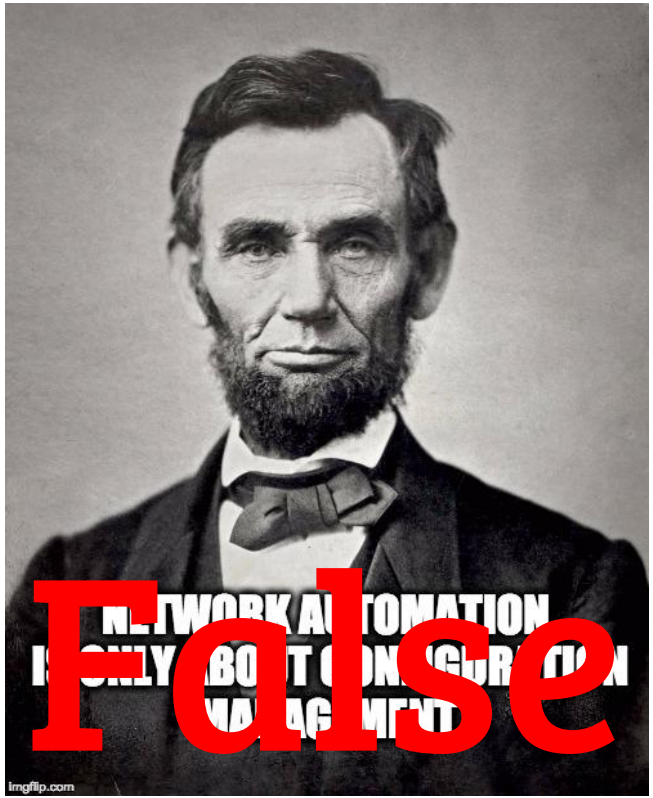
Vendor-agnostic automation: how to

Read more, do more, reinvent less.

Event-driven network automation (1)



Event-driven network automation (1)



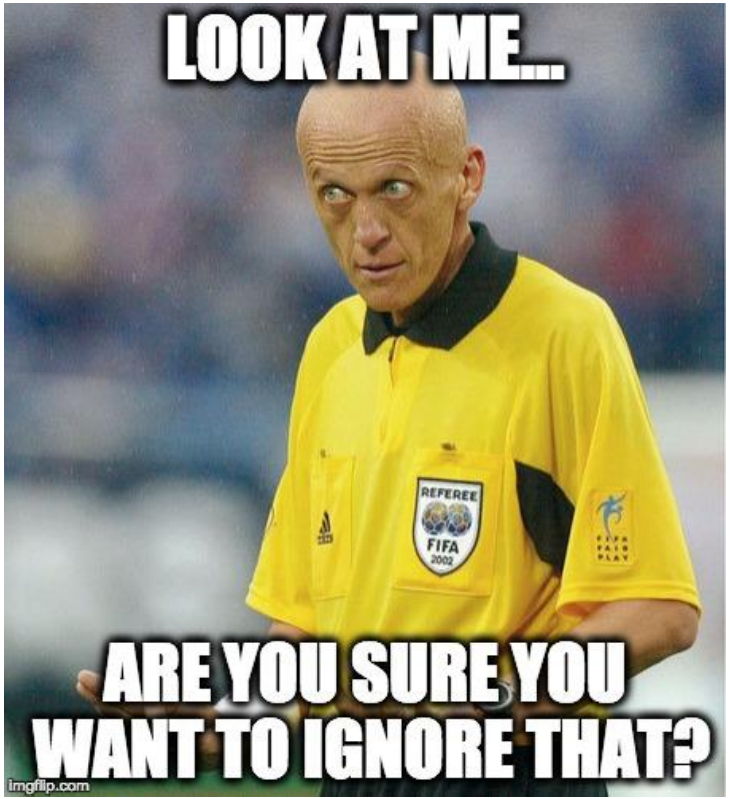
Event-driven network automation (2)

- Several of ways your network is trying to communicate with you
- Millions of messages

Event-driven network automation (3)

- SNMP traps
- Syslog messages
- Streaming telemetry

Event-driven network automation (4)



Streaming Telemetry

- Push notifications
  - Vs. pull (SNMP)
- Structured data
  - Structured objects, using the YANG standards
    - OpenConfig
    - IETF
- Supported on very new operating systems
  - IOS-XR >= 6.1.1
  - Junos >= 15.1 (depending on the platform)

Syslog messages

- Junos

```
<149>Jun 21 14:03:12  vmx01 rpd[2902]: BGP_PREFIX_THRESH_EXCEEDED: 192.168.140.254 (External AS 4230): Configured maximum
prefix-limit threshold(140) exceeded for inet4-unicast nlri: 141 (instance master)
```

- IOS-XR

```
<149>2647599: xrv01 RP/0/RSP1/CPU0:Mar 28 15:08:30.941 UTC: bgp[1051]: %ROUTING-BGP-5-MAXPFX : No. of IPv4 Unicast prefixes
received from 192.168.140.254 has reached 94106, max 12500
```

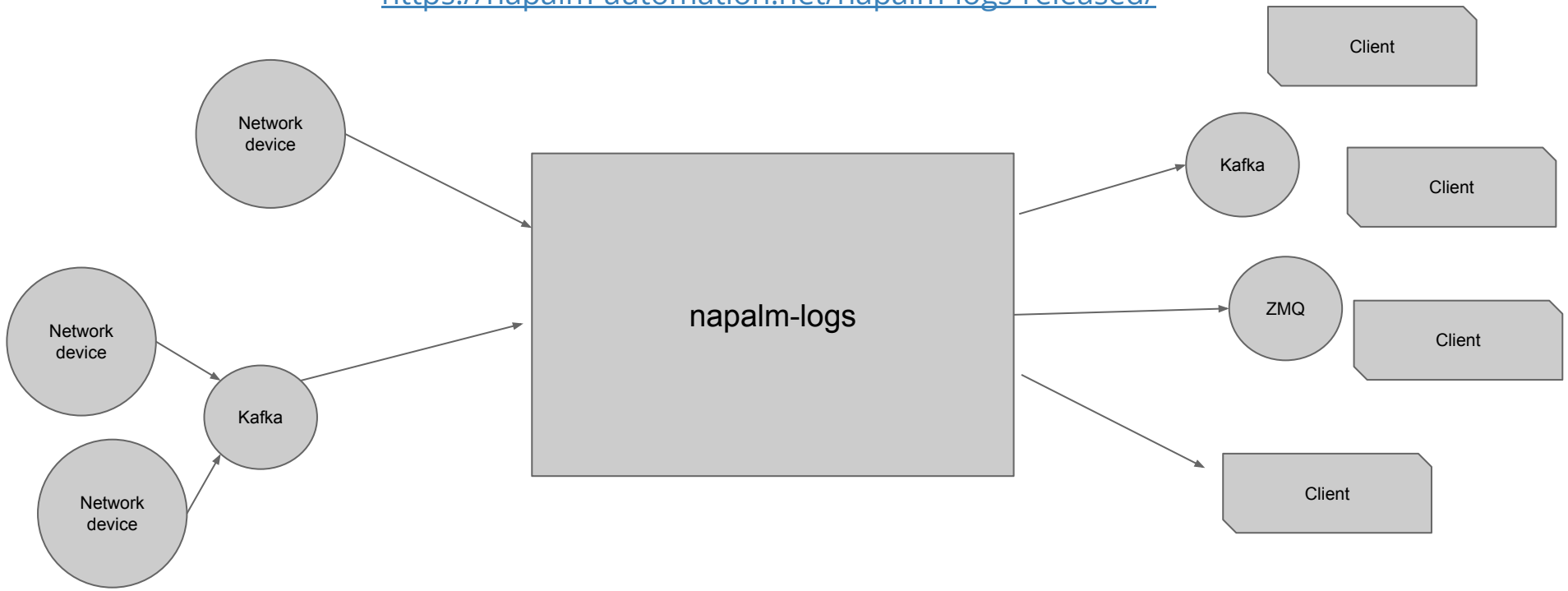
Syslog messages: napalm-logs (1)

https://napalm-automation.net/napalm-logs-released/

- Listen for syslog messages
  - Directly from the network devices, via UDP or TCP
  - Other systems: Apache Kafka, ZeroMQ, etc.
- Publish encrypted messages
  - Structured documents, using the YANG standards
    - OpenConfig
    - IETF
  - Over various channels: ZeroMQ, Kafka, etc.

Syslog messages: napalm-logs (2)

https://napalm-automation.net/napalm-logs-released/



Syslog messages: napalm-logs startup

```
$ napalm-logs --listener udp --address 172.17.17.1 --port 5514  --publish-address 172.17.17.2 --publish-port 49017
              --publisher zmq --disable-security
```

More configuration options:
https://napalm-logs.readthedocs.io/en/latest/options/index.html

Syslog messages: napalm-logs clients

```
import zmq  # when using the ZeroMQ publisher
import napalm_logs.utils

server_address = '127.0.0.1'  # IP
server_port = 49017           # Port for the napalm-logs publisher interface

context = zmq.Context()
socket = context.socket(zmq.SUB)
socket.connect('tcp://{address}:{port}'.format(address=server_address,
                                               port=server_port))
socket.setsockopt(zmq.SUBSCRIBE, '')  # subscribe to the napalm-logs publisher

while True:
    raw_object = socket.recv()  # binary object
    print(napalm_logs.utils.unserialize(raw_object))  # deserialize
```

More complete example:
https://github.com/napalm-automation/napalm-logs/blob/master/examples/client_auth.py

Syslog messages: napalm-logs structured objects


```
"yang_message": {
      "bgp": {
          "neighbors": {
              "neighbor": {
                  "192.168.140.254": {
                      "afi_safis": {
                          "afi_safi": {
                              "inet4": {
                                  "ipv4_unicast": {
                                      "prefix_limit": {
                                          "state": {
                                              "max_prefixes": 140
                                          }
                                      }
                                  },
                                  "state": {
                                      "prefixes": {
                                          "received": 141
                                      }
                                  }
                              }
                          }
                      },
                      "state": {
                          "peer_as": "4230"
                      }
                  }
              }
          }
      }
  },
  "yang_model": "openconfig-bgp"
```

Salt event system

Salt is a data driven system. Each action (job) performed (manually from the CLI or automatically by the system) is uniquely identified and has an identification tag:

```
$ sudo salt junos-router net.arp
# output omitted
```

```
$ sudo salt-run state.event pretty=True
salt/job/20170110130619367337/new {
    "_stamp": "2017-01-10T13:06:19.367929",
    "arg": [],
    "fun": "net.arp",
    "jid": "20170110130619367337",
    "minions": [
        "junos-router"
    ],
    "tgt": "junos-router",
    "tgt_type": "glob",
    "user": "mircea"
}
```

Unique job tag

Syslog messages: napalm-syslog Salt engine (1)

https://docs.saltstack.com/en/latest/ref/engines/all/salt.engines.napalm_syslog.html

Imports messages from *napalm-logs* into the Salt event bus

`/etc/salt/master`

```
engines:
  - napalm_syslog:
      transport: zmq
      address: 172.17.17.2
      port: 49017
      auth_address: 172.17.17.3
      auth_port: 49018
```

Syslog messages: Napalm-syslog Salt engine (2)

Salt event bus:

```
napalm/syslog/junos/NTP_SERVER_UNREACHABLE/edge01.bjm01 {
  "error": "NTP_SERVER_UNREACHABLE",
  "host": "edge01.bjm01",
  "ip": "10.10.0.1",
  "os": "junos",
  "timestamp": 1499986394,
  "yang_message": {
    "system": {
      "ntp": {
        "servers": {
          "server": {
            "172.17.17.1": {
              "state": {
                "stratum": 16,
                "association-type": "SERVER"
              }
            }
          }
        }
      }
    }
  },
  "yang_model": "openconfig-system"
}
```

Fully automated configuration changes

/etc/salt/master

```
reactor:
  - 'napalm/syslog/*/NTP_SERVER_UNREACHABLE/*':
    - salt://reactor/exec_ntp_state.sls
```

Matches the event tag

napalm/syslog/junos/NTP_SERVER_UNREACHABLE/edge01.bjm01

/etc/salt/reactor/exec_ntp_state.sls

```
triggered NTP state:
  cmd.state.sls:
    - tgt: {{ data.host }}
    - arg:
      - ntp
```

CLI Equivalent:

```
$ sudo salt edge01.bjm01 state.sls ntp
```

More advanced topics

- Orchestration: define complex workflows
  https://docs.saltstack.com/en/latest/topics/orchestrate/index.html
  See also: https://docs.saltstack.com/en/develop/ref/states/requisites.html
- Publish events to external services (e.g.: logstash, hipchat)
  https://docs.saltstack.com/en/develop/ref/engines/all/index.html
- Pillar: load data from external services, not just static
  https://docs.saltstack.com/en/develop/ref/pillar/all/
- Custom authentication methods for the minions
  https://docs.saltstack.com/en/develop/ref/auth/all/index.html
- Forward outputs in external data systems on runtime
  https://docs.saltstack.com/en/develop/ref/returners/all/index.html

How can you contribute?

GitHub

- NAPALM Automation: https://github.com/napalm-automation
- SaltStack: https://github.com/saltstack/salt

Need help/advice?

Join https://networktocode.herokuapp.com/
rooms: #**saltstack** #**napalm**

mircea@cloudflare.com

Questions

CLOUDFLARE

References

Arista Software download
Authentication system
Beacons
Engines
Event System
Grains
Jinja
load_template documentation
Master config file, default
Master config file, example
Master configuration options
Master systemd file
Mine
NAPALM
NAPALM BGP execution module functions
NAPALM Grains
NAPALM Installation
NAPALM network execution module functions
NAPALM NTP execution module functions
NAPALM Proxy
NAPALM route execution module functions
NAPALM SNMP execution module functions
NAPALM users execution module functions
Nested outputter
NETAPI Modules
Netconfig state
Node Groups
NTP state
Orchestration
Output modules
Pillar
Pillar modules
Proxy config file, default
Proxy config file, example
Proxy Minion
Proxy systemd file
Reactor
REST CherryPy

References

Returners
Runners
Salt 2016.11 (Carbon) release notes
Salt Get Started
Salt Installation
Salt Walkthrough
Salt-key
SaltStack Package Repo
SNMP state
States
Targeting minions
The Top file
Users state
Vagrant boxes, HashiCorp
Vagrant Installation
Vagrantfile example 1
Vagrantfile example 2
VirtualBox Installation
YAML