

Event-driven network automation and orchestration

Tom Strickx
Cloudflare, London

UKNOF 40
Manchester, April 2018

Tom Strickx

- Chaos Monkey at Cloudflare (Network software engineer)
- Contributor at [NAPALM Automation](#)
- <https://tom.strickx.com/>



Cloudflare

- How big?
 - 7+ million zones/domains
 - Authoritative for ~40% of Alexa top 1 million
 - 200 million Internet users served
 - 100+ billion DNS queries/day
 - Largest
 - Fastest
 - 35% of the Internet requests
 - Now also a resolver (1.1.1.1)
 - 10 trillion requests / month
 - 10% of the Internet traffic
- 150+ anycast locations globally
 - 74 countries (and growing)
 - Many hundreds of network devices

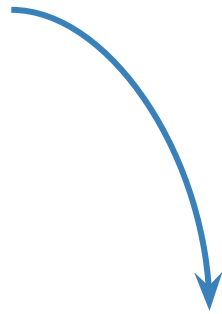


Agenda

- Vendor-agnostic automation
- napalm-logs
- Using napalm-logs for event-driven network automation

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**

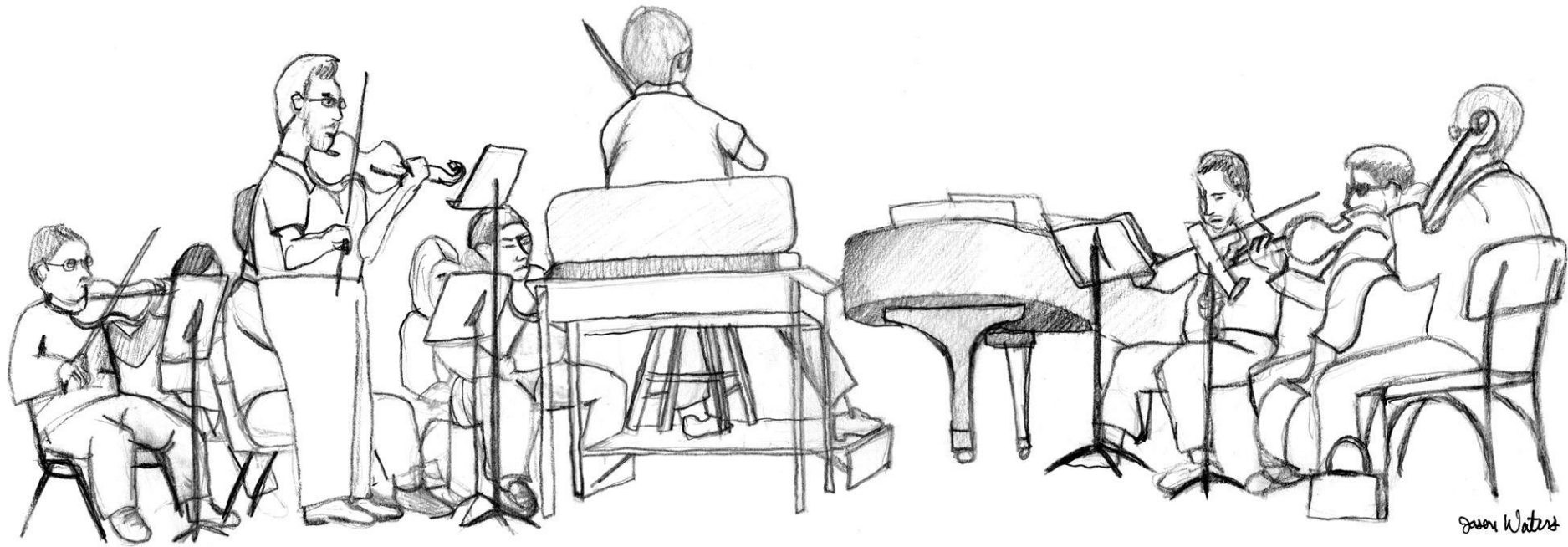
Frameworks used in networking



Why Salt

- Very scalable
 - e.g., [LinkedIn 70,000 servers](#)
- Event-driven orchestrator
- Easily configurable & customizable
- Native caching and drivers for useful tools
- One of the friendliest communities
- Vendor neutral
- 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



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 included 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](#)

NAPALM integrated in Salt: Nitrogen

Introduced in 2016.11, the modules for cross-vendor network automation have been improved, enhanced and widened 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 specifying 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 contains 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.

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

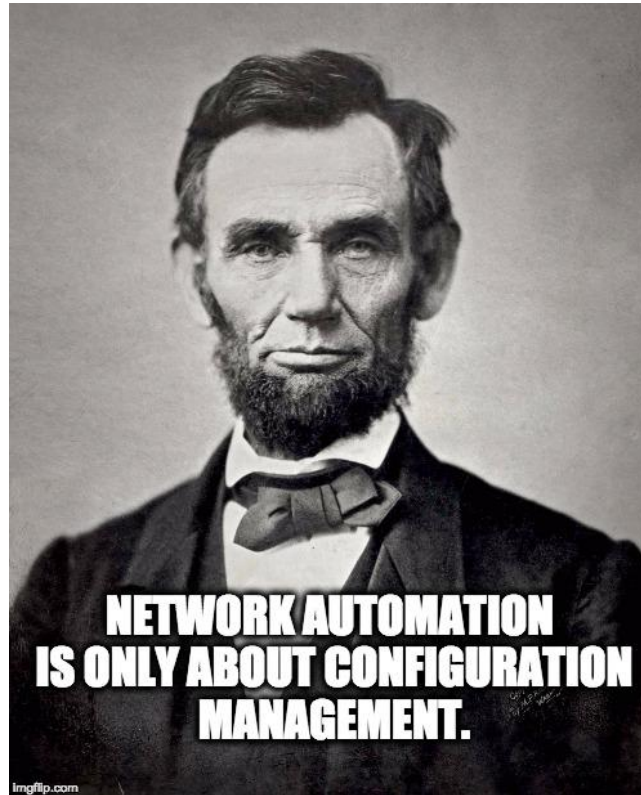
- [Salt in 10 minutes](#)
- [Salt fundamentals](#)
- [Configuration management](#)
- [Network Automation official Salt docs](#)
- [Step-by-step tutorial](#) -- up and running in 60 minutes
- [Using Salt at Scale](#)

Vendor-agnostic automation: how to

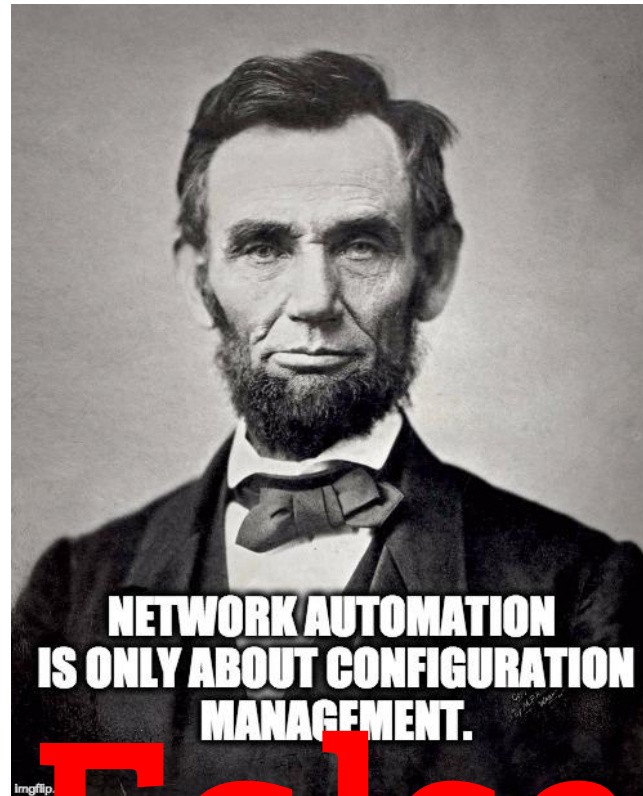
Read more, do more, reinvent less.

Event-driven automation

Event-driven network automation (1)



Event-driven network automation (1)



False

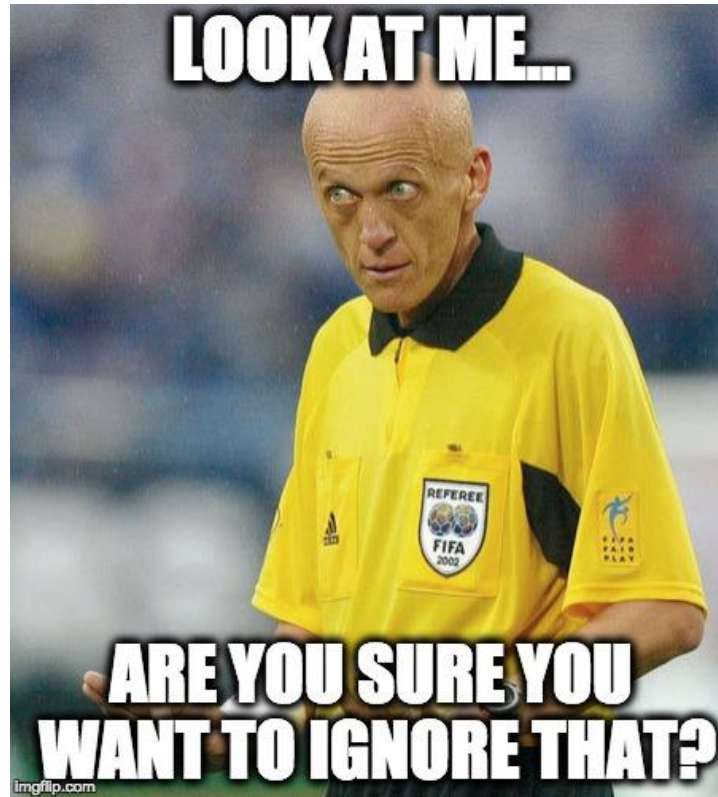
Event-driven network automation (2)

- Several 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)



Event-driven network automation

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 $\geq 6.1.1$
 - Junos ≥ 15.1 (depending on the platform)

Event-driven network automation

Syslog messages

- Junos

```
<99>Jul 13 22:53:14 device1 xntpd[16015]: NTP Server 172.17.17.1 is Unreachable
```

- IOS-XR

```
<99>2647599: device3 RP/0/RSP0/CPU0:Aug 21 09:39:14.747 UTC: ntpd[262]: %IP-IP_NTP-5-SYNC_LOSS : Synchronization lost :  
172.17.17.1 :The association was removed
```

Event-driven network automation

Syslog messages: napalm-logs (1)

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

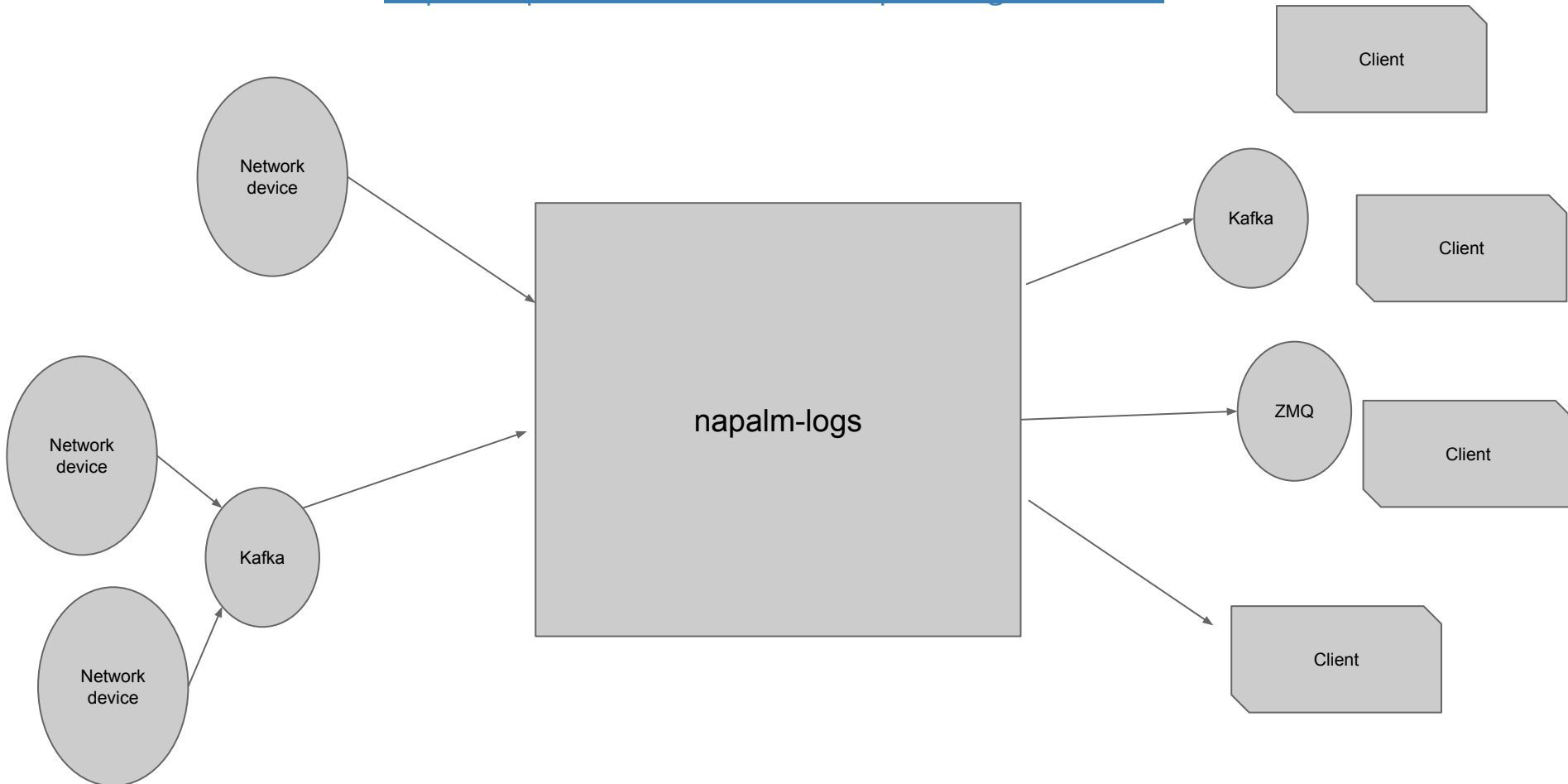


- Listen for syslog messages
 - Directly from the network devices, via UDP or TCP
 - Other systems: Apache Kafka, ZeroMQ, etc.
- Publish encrypted messages
 - Structured documents, using the YANG standards
 - OpenConfig
 - IETF
 - Over various channels: ZeroMQ, Kafka, etc.

Event-driven network automation

Syslog messages: napalm-logs (2)

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



Event-driven network automation

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>

Event-driven network automation

Syslog messages (again)

- Junos

```
<99>Jul 13 22:53:14 device1 xntpd[16015]: NTP Server 172.17.17.1 is Unreachable
```

- IOS-XR

```
<99>2647599: device3 RP/0/RSP0/CPU0:Aug 21 09:39:14.747 UTC: ntpd[262]: %IP-IP_NTP-5-SYNC_LOSS : Synchronization lost :  
172.17.17.1 :The association was removed
```


Event-driven network automation Syslog messages: *napalm-logs* structured objects

```
{
  "error": "NTP_SERVER_UNREACHABLE",
  "facility": 12,
  "host": "device1",
  "ip": "127.0.0.1",
  "os": "junos",
  "severity": 4,
  "timestamp": 1499986394,
  "yang_message": {
    "system": {
      "ntp": {
        "servers": {
          "server": {
            "172.17.17.1": {
              "state": {
                "stratum": 16,
                "association-type": "SERVER"
              }
            }
          }
        }
      }
    }
  },
  "yang_model": "openconfig-system"
}
```

Event-driven network automation

Other raw syslog message example

- 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
```

Event-driven network automation 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"
}
```

Event-driven network automation

napalm-logs key facts to remember

- Continuously listening to syslog messages
- Continuously publishing structured data
 - Structure following the YANG standards
 - OpenConfig
 - IETF

Event-driven network automation

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



Event-driven network automation

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
```

Event-driven network automation Syslog messages: *napalm-logs* structured objects

(from slide #33)

```
{
  "error": "NTP_SERVER_UNREACHABLE",
  "facility": 12,
  "host": "device1",
  "ip": "127.0.0.1",
  "os": "junos",
  "severity": 4,
  "timestamp": 1499986394,
  "yang_message": {
    "system": {
      "ntp": {
        "servers": {
          "server": {
            "172.17.17.1": {
              "state": {
                "stratum": 16,
                "association-type": "SERVER"
              }
            }
          }
        }
      }
    }
  },
  "yang_model": "openconfig-system"
}
```

Event-driven network automation

Salt event bus

Using the *napalm-syslog* Salt engine you can inject *napalm-logs* events into the Salt event bus.

See <https://napalm-automation.net/napalm-logs-released/> and <https://mirceaulinic.net/2017-10-19-event-driven-network-automation/> For more examples

```
napalm/syslog/junos/NTP_SERVER_UNREACHABLE/edge01.bjm01 {
  "error": "NTP_SERVER_UNREACHABLE",
  "facility": 12,
  "host": "edge01.bjm01",
  "ip": "10.10.0.1",
  "os": "junos",
  "timestamp": 1499986394,
  "yang_message": {
    "system": {
      "ntp": {
        "servers": {
          "server": {
            "172.17.17.1": {
              "state": {
                "association-type": "SERVER",
                "stratum": 16
              }
            }
          }
        }
      }
    }
  },
  "yang_model": "openconfig-system"
}
```


Event-driven network automation

Fully automated configuration changes

/etc/salt/master

```
reactor:
```

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

/etc/salt/reactor/exec_ntp_state.sls

```
triggered NTP state:
```

```
  local.state.sls:
```

```
    - tgt: {{ data.host }}  
    - arg:  
      - ntp
```

Matches the event tag

```
napalm/syslog/junos/NTP_SERVER_UNREACHABLE/edge01.bjm01
```

CLI Equivalent:

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

Event-driven network automation

Fully automated configuration changes & more

/etc/salt/master

```
reactor:
```

```
- 'napalm/syslog/*/INTERFACE_DOWN/*':  
  - salt://reactor/if_down_shutdown.sls  
  - salt://reactor/if_down_send_mail.sls
```

Shutdown the interface

Send an email notification

Matches the event tag

```
napalm/syslog/junos/INTERFACE_DOWN/edge01.bjm01
```

(Event pushed when an interface is operationally down)

More details at:

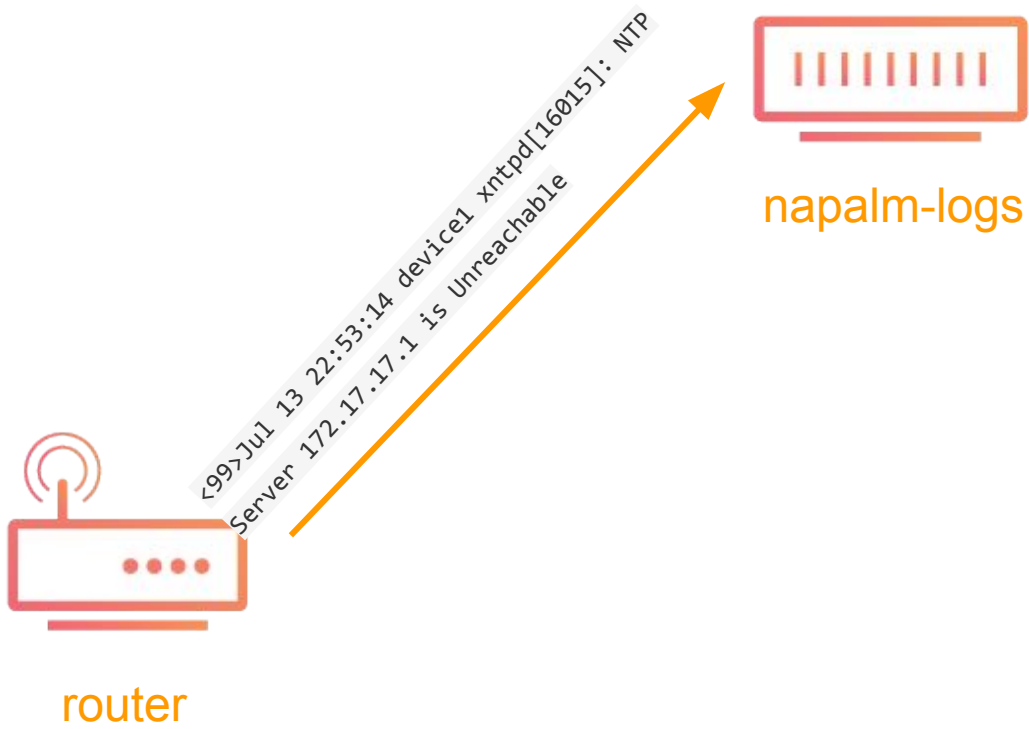
<https://mirceaulinic.net/2017-10-19-event-driven-network-automation/>

Conclusion

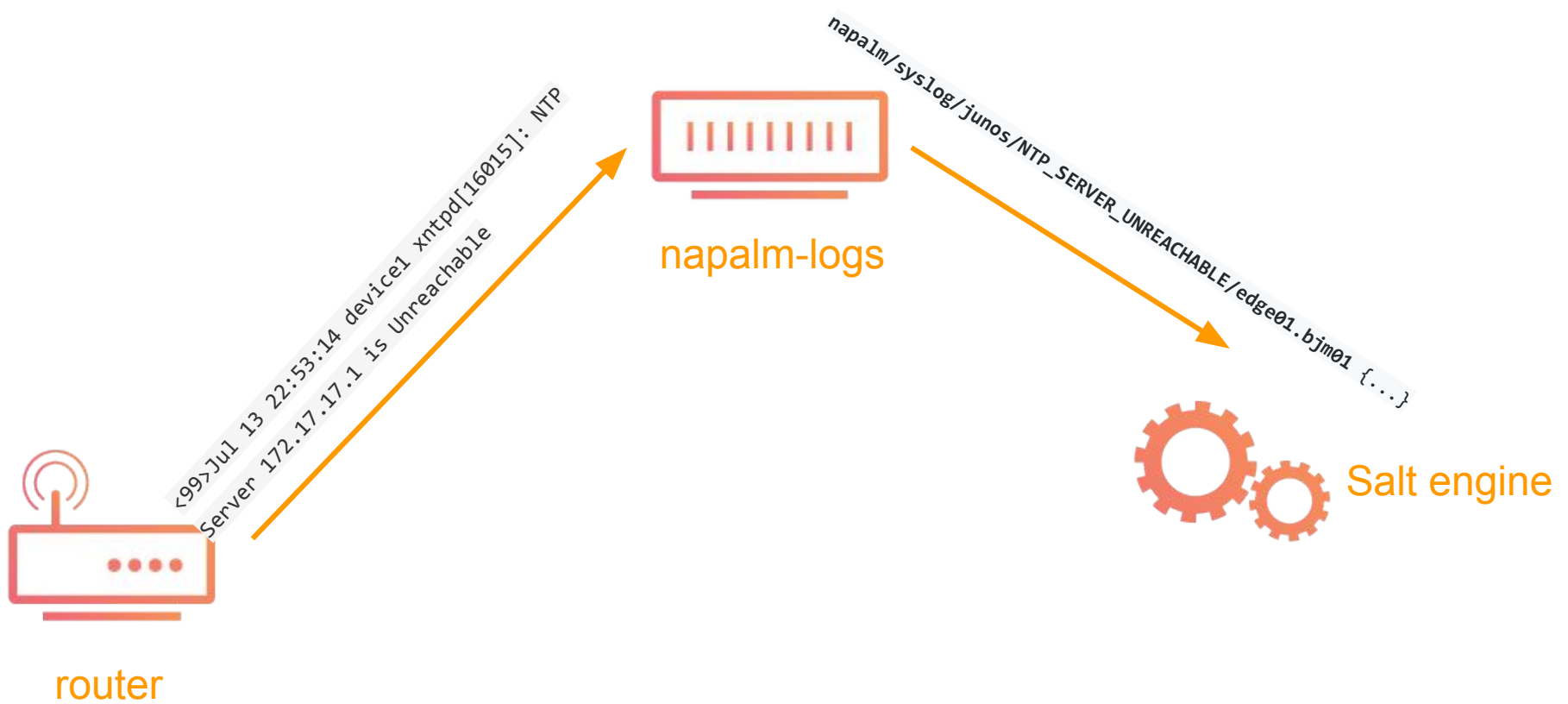


router

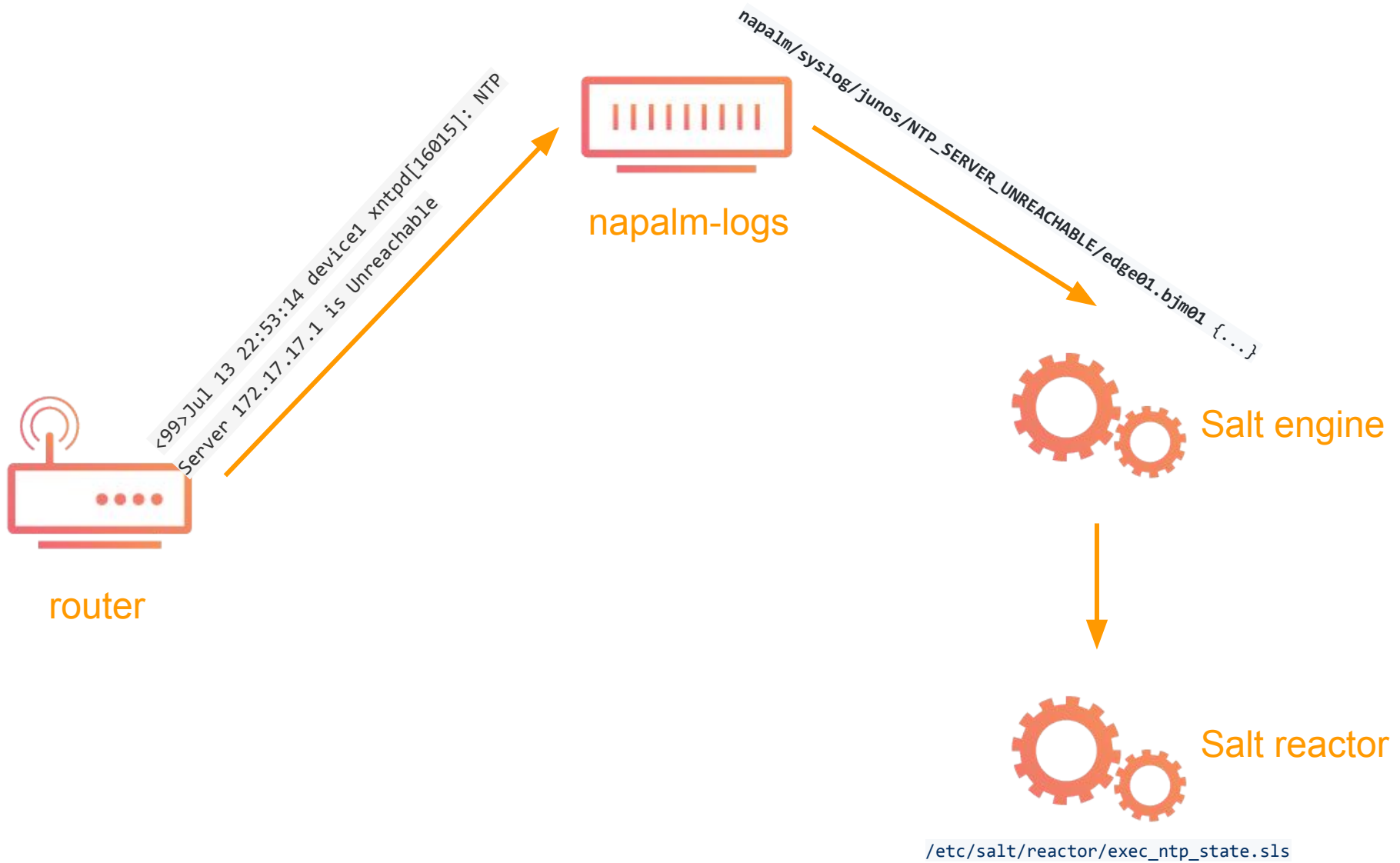
Conclusion



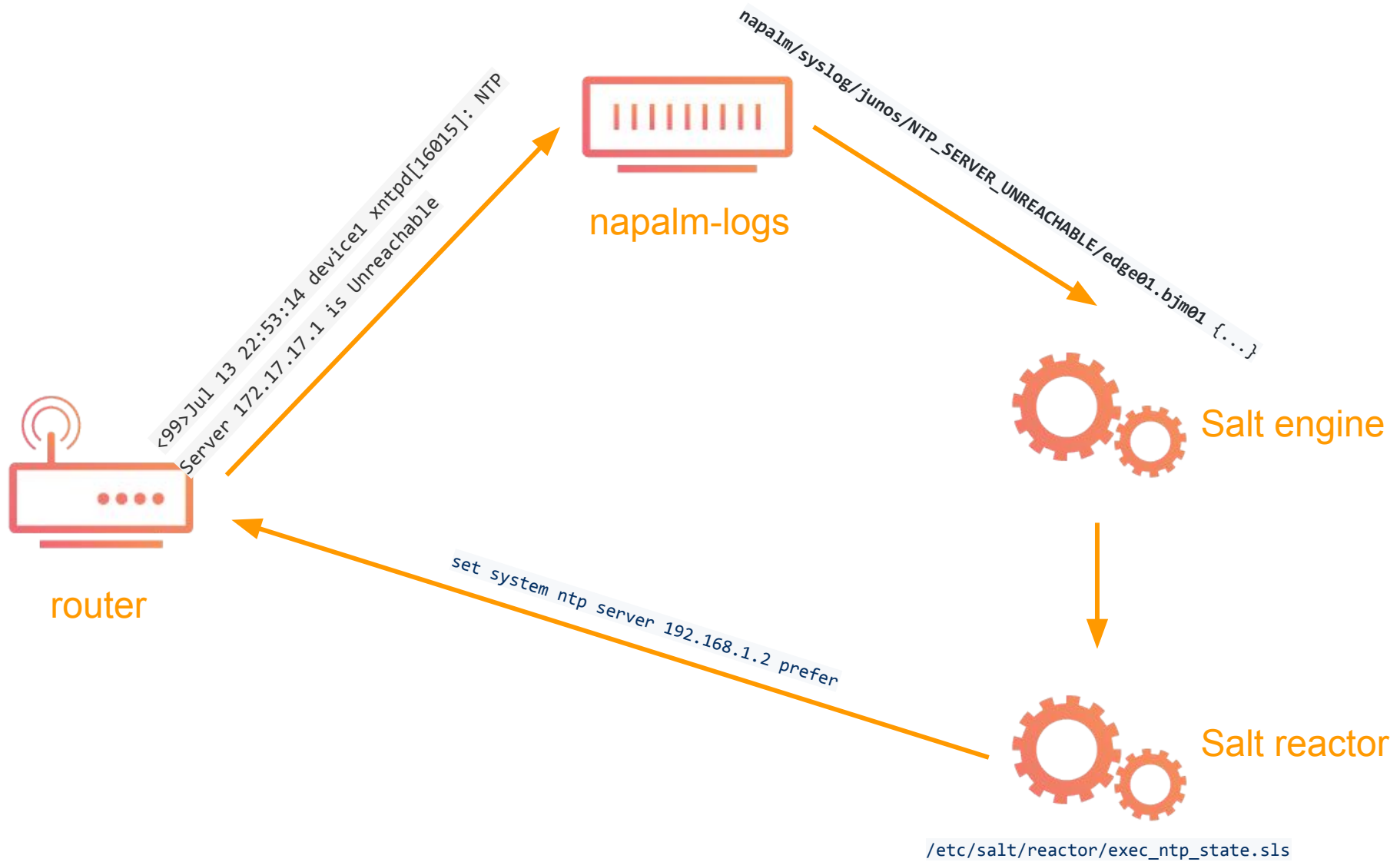
Conclusion



Conclusion



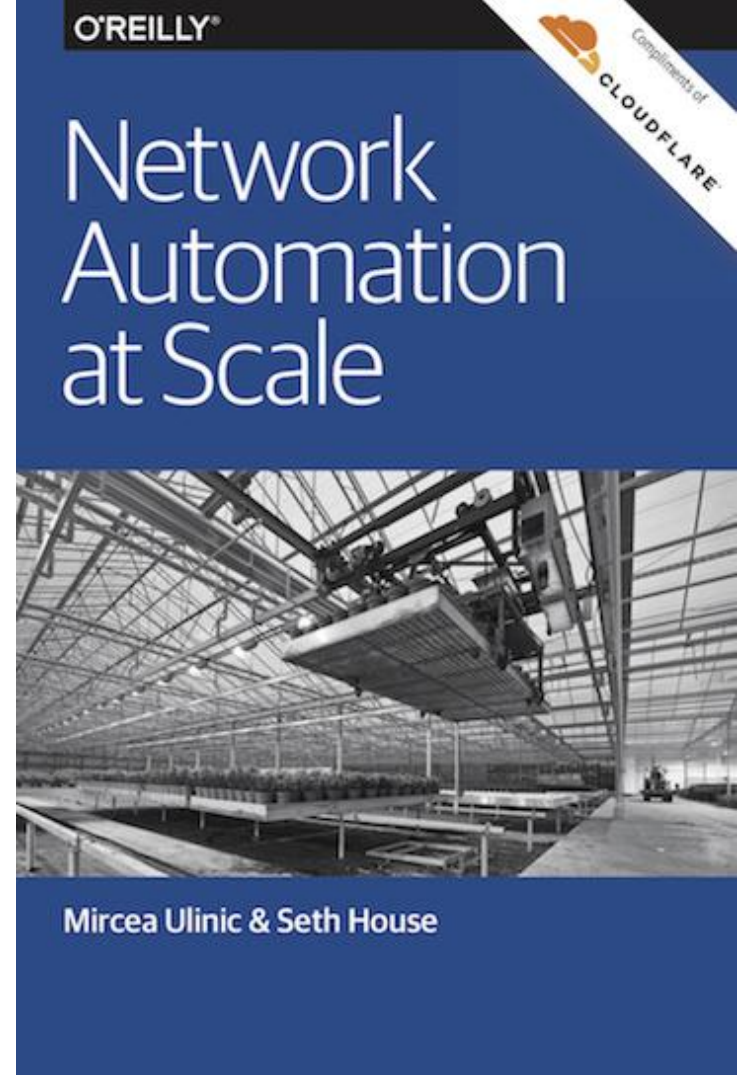
Conclusion



Network Automation at Scale: the book

Free download:

<https://www.cloudflare.com/network-automation-at-scale-ebook/>



Need help/advice?



Join <https://networktocode.slack.com/>
rooms: **#saltstack #napalm**

<https://saltstackcommunity.slack.com>
rooms: **#networks**

How can you contribute?

GitHub

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

Questions

