lieter_
@lieter@mastodon.lieter.nl
pieterlexis

YANG and NETCONF

Model-based configuration management for networks

Pieter Lexis

cfgmgmtcamp 2020 Ghent, February 3

1. Hello, welcome

Lack of YAML

Mentions of XML

ASCII diagrams

DNS

Some bad memes

1. First off, some content warnings

Agenda

- Introduction
- The solution
  - Configuring devices: NETCONF
  - Modeling data: YANG
  - YANG and NETCONF servers
- YANG and NETCONF on *NIX?
- Wrap up

Agenda
Introduction
The solution
Configuring devices: NETCONF
Modeling data: YANG
YANG and NETCONF servers
YANG and NETCONF on *NIX?
Wrap up

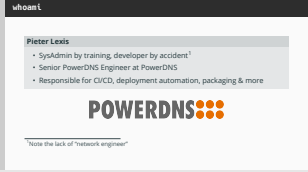
Introduction

Pieter Lexis

- SysAdmin by training, developer by accident[1]
- Senior PowerDNS Engineer at PowerDNS
- Responsible for CI/CD, deployment automation, packaging & more

POWERDNS

[1] Note the lack of “network engineer”

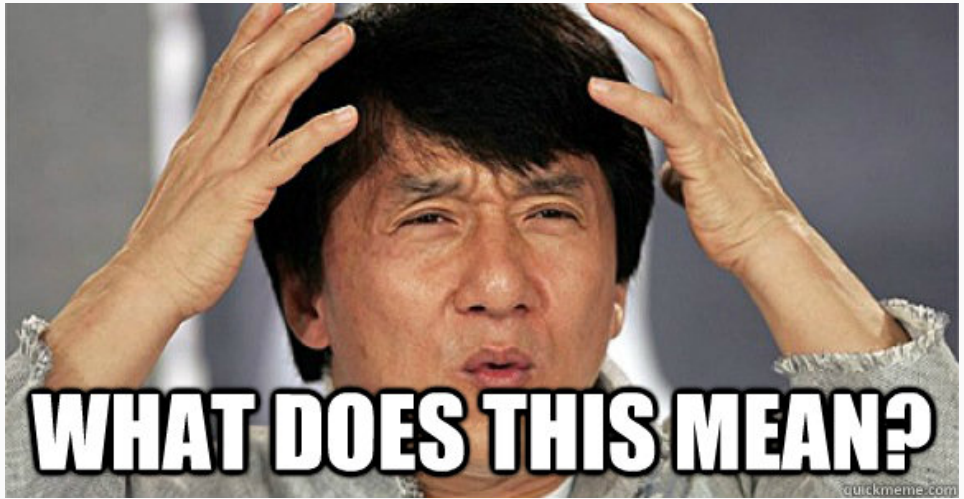


How did I get here?

“Please YANG-ify the PowerDNS Authoritative Server”
— $Customer’s research department

How did I get here?
“Please YANG-ify the PowerDNS Authoritative Server”
— $Customer’s research department

How did I get here?



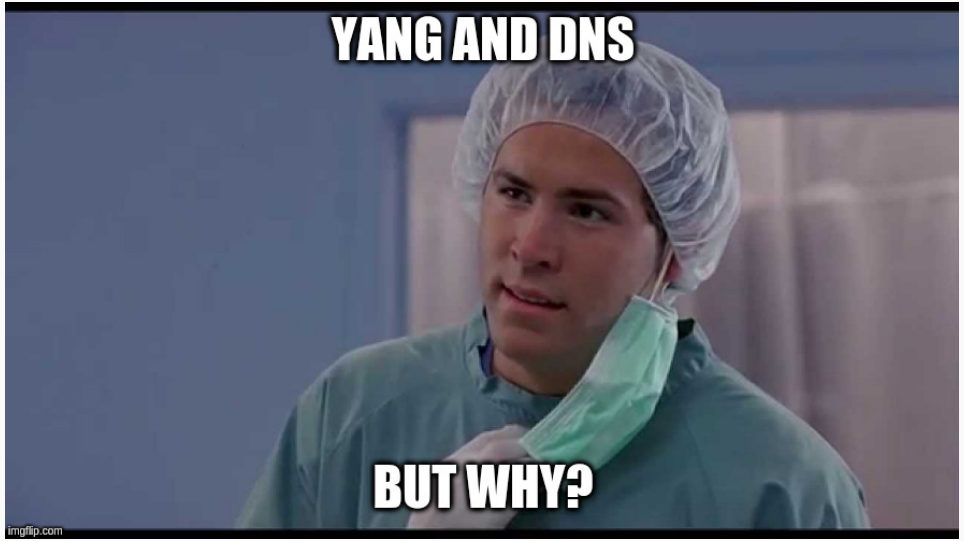
How did I get here?
WHAT DOES THIS MEAN?

How did I get here?

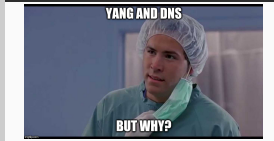
“You model the server config, including the zone data”
— $Customer’s research department

How did I get here?
“You model the server config, including the zone data”
— $Customer’s research department

How did I get here?



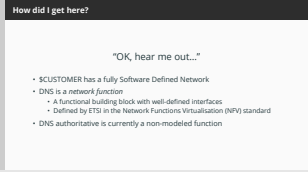
How did I get here?



How did I get here?

"OK, hear me out..."

- $CUSTOMER has a fully Software Defined Network
- DNS is a *network function*
  - A functional building block with well-defined interfaces
  - Defined by ETSI in the Network Functions Virtualisation (NFV) standard
- DNS authoritative is currently a non-modeled function



1. Nation-wide consumer access network, fully with YANG and NETCONF. Even the Modems are configured in this way
2. Most Telcos and ISPs are moving towards this, especially for 5g

The problem

- CLI differs between vendors
- Vendors have different configuration APIs
- SNMP is unreliable
- MIBs don’t distinguish between “state” and “configuration”
- Little standardized MIBs, no “common” MIBs

RFC 3535, §3 “Operator Requirements”

The problem

- CLI differs between vendors
- Vendors have different configuration APIs
- SNMP is unreliable
- MIBs don't distinguish between "state" and "configuration"
- Little standardized MIBs, no "common" MIBs

RFC 3535, §3 "Operator Requirements"

1. CLI input and output is different, making scripting and parsing hard
2. SNMP uses UDP
3. State is e.g. a counter or the load on a port
4. RFC 3535 §3 is 12 requirements, which boil down to “automation”

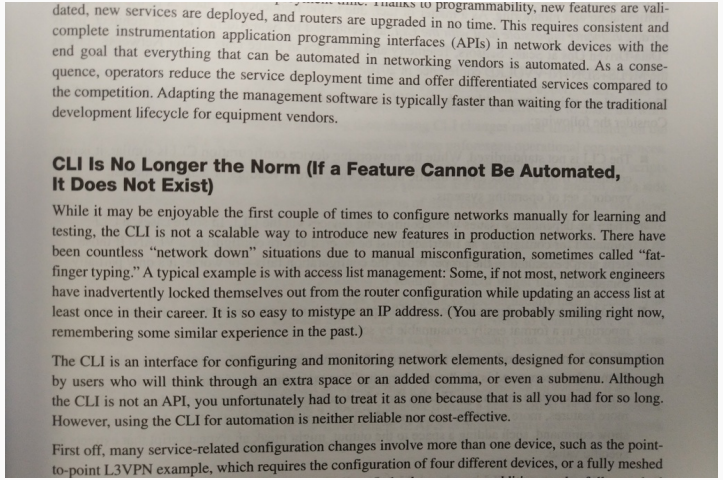
The problem — In short

- Heterogeneous environments are painful to configure
- Need for standardized configuration
- Need for data *and* configuration

The problem — In short

- Heterogeneous environments are painful to configure
- Need for standardized configuration
- Need for data *and* configuration

The problem



dated, new services are deployed, and routers are upgraded in no time. This requires consistent and complete instrumentation application programming interfaces (APIs) in network devices with the end goal that everything that can be automated in networking vendors is automated. As a consequence, operators reduce the service deployment time and offer differentiated services compared to the competition. Adapting the management software is typically faster than waiting for the traditional development lifecycle for equipment vendors.

CLI Is No Longer the Norm (If a Feature Cannot Be Automated, It Does Not Exist)

While it may be enjoyable the first couple of times to configure networks manually for learning and testing, the CLI is not a scalable way to introduce new features in production networks. There have been countless "network down" situations due to manual misconfiguration, sometimes called "fat-finger typing." A typical example is with access list management: Some, if not most, network engineers have inadvertently locked themselves out from the router configuration while updating an access list at least once in their career. It is so easy to mistype an IP address. (You are probably smiling right now, remembering some similar experience in the past.)

The CLI is an interface for configuring and monitoring network elements, designed for consumption by users who will think through an extra space or an added comma, or even a submenu. Although the CLI is not an API, you unfortunately had to treat it as one because that is all you had for so long. However, using the CLI for automation is neither reliable nor cost-effective.

First off, many service-related configuration changes involve more than one device, such as the point-to-point L3VPN example, which requires the configuration of four different devices, or a fully meshed

Figure 1: From "Network Programming with YANG", by Claise, Clarke, and Lindblad

The problem

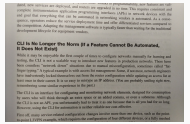


Figure 1: From "Network Programming with YANG", by Claise, Clarke, and Lindblad

The solution

The solution

The solution — NETCONF and YANG documents

RFC 4741 – “NETCONF Configuration Protocol”, December 2006
RFC 6020 – “YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)”, October 2010
RFC 6241 – “Network Configuration Protocol (NETCONF)”, June 2011
RFC 6244 – “An Architecture for Network Management Using NETCONF and YANG”, June 2011
RFC 7950 – “The YANG 1.1 Data Modeling Language”, August 2016
RFC 7951 – “JSON Encoding of Data Modeled with YANG”, August 2016
RFC 8040 – “RESTCONF Protocol”, January 2017
RFC 8342 – “Network Management Datastore Architecture (NMDA)”, March 2018

The solution — NETCONF and YANG documents

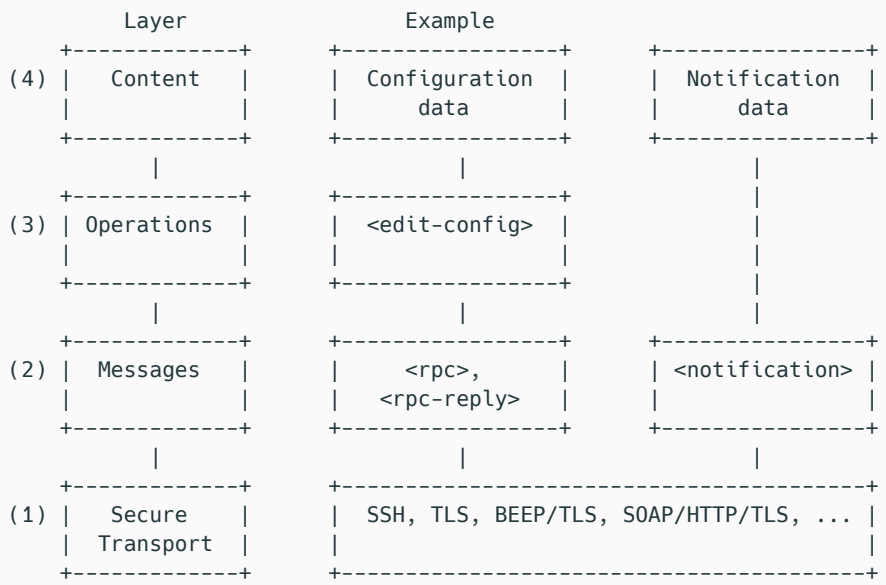
RFC 4741 – “NETCONF Configuration Protocol”, December 2006
RFC 6020 – “YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)”, October 2010
RFC 6241 – “Network Configuration Protocol (NETCONF)”, June 2011
RFC 6244 – “An Architecture for Network Management Using NETCONF and YANG”, June 2011
RFC 7950 – “The YANG 1.1 Data Modeling Language”, August 2016
RFC 7951 – “JSON Encoding of Data Modeled with YANG”, August 2016
RFC 8040 – “RESTCONF Protocol”, January 2017
RFC 8342 – “Network Management Datastore Architecture (NMDA)”, March 2018

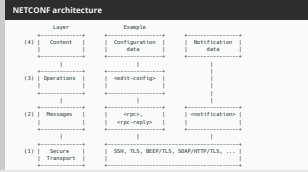
1. NETCONF and RESTCONF are the protocols that are used for communication with the device
2. YANG is the language that describes the device
3. YANG stands for Yet Another Next Generation
4. RESTCONF is NETCONF over HTTP in a REST, using the URI for the tree

The solution

Configuring devices: NETCONF

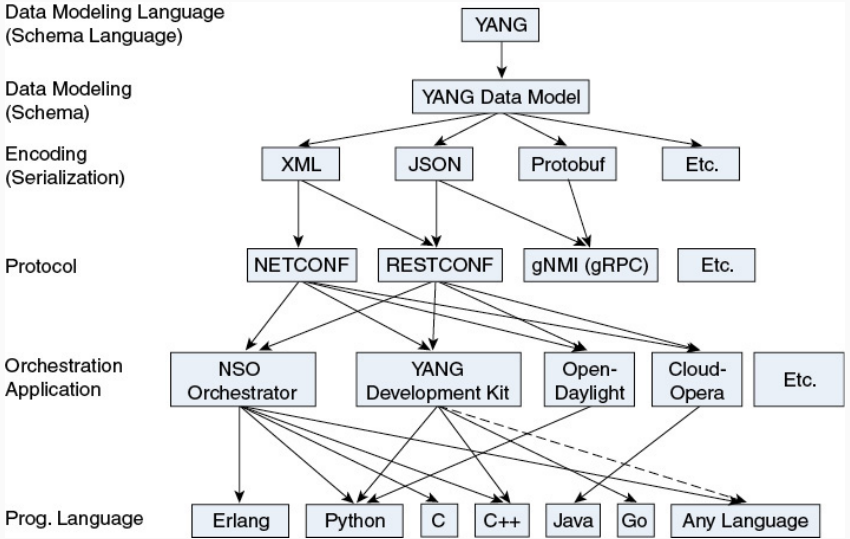
NETCONF architecture

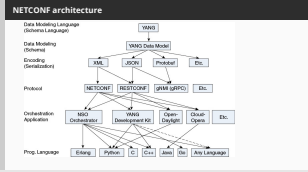




1. Content is the instantiated YANG model
2. Only a few operations exists: lock/unlock

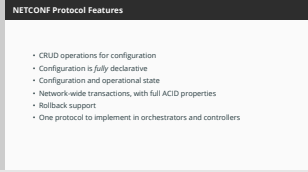
NETCONF architecture





NETCONF Protocol Features

- CRUD operations for configuration
- Configuration is *fully* declarative
- Configuration and operational state
- Network-wide transactions, with full ACID properties
- Rollback support
- One protocol to implement in orchestrators and controllers



1. State and config is human readable
2. Operations are applied fully or not, no “half” state
3. Protocol also features 2-phase commit/confirmed commit support with rollback
4. Rollback to a previously store config, allowed because config is declared
5. Additionally: because of the models, it is easy to go from state A to state B, as it is only a diff.

Who uses NETCONF

- Routers and Switches of the big vendors
- Orchestration frameworks
- Network Management Systems (NMS)
- Several *nix applications

Who uses NETCONF

- Routers and Switches of the big vendors
- Orchestration frameworks
- Network Management Systems (NMS)
- Several *nix applications

vendors

Cisco, Brocade, Juniper, HP, Alcatel Lucent, Arista

orchestration frameworks

ONAP, ETSI's NFV standards

NMS

Cisco NSO

1. More on *nix applications later

The solution

Modeling data: YANG

Yet Another Next Generation

- The *schema* defines the data
- The NETCONF server has the *instantiated data*
- Schema describes a tree

Yet Another Next Generation

- The *schema* defines the data
- The NETCONF server has the *instantiated data*
- Schema describes a tree

1. Think of it as a DTD
2. A string cannot be in a node that must contain an integer
3. When referencing another leaf, it must exists

YANG models — Example i

my-example-model.yang

```
 9  grouping endpoint {
10    description
11      "An IP endpoint, including the port";
12    leaf ip-address {
13      type inet:ip-address-no-zone;
14      mandatory true;
15    }
16    leaf port {
17      type inet:port-number;
18    }
19  }
20
21  container listen-addresses {
22    list listen-address {
23      key "name";
24      leaf name {
25        type string;
26      }
27      unique "ip-address port";
28      uses endpoint {
```


YANG models — Example ii

```
29          refine port {
30            default 25;
31          }
32        }
33      }
34    }
35
36    container counters {
37      config false;
38      leaf connection-count {
39        type uint32;
40      }
41    }
42  }
```


YANG models — Example ii

YANG model as a tree

```
module: my-example-model
  +--rw listen-addresses
  |  +--rw listen-address* [name]
  |     +--rw name           string
  |     +--rw ip-address     inet:ip-address
  |     +--rw port?          inet:port-number
  +--ro counters
     +--ro connection-count?   uint32
```

The end-nodes are called *leaf*s.

YANG Models — Tree elements

- *Grouping* — Set of nodes for re-use
- *Container* — A set of related nodes
- *List* — A keyed set of nodes
- *Leaf-list* — List of a single item

YANG Models — Tree elements

- *Grouping* — Set of nodes for re-use
- *Container* — A set of related nodes
- *List* — A keyed set of nodes
- *Leaf-list* — List of a single item

1. A list can be thought of as a hashmap, the key must be unique

YANG Models — Built-in types

- (u)int8, (u)int16, (u)int32, (u)int64
- decimal64
- string
- bits
- boolean
- enumeration
- union

YANG Models — Built-in types

- (u)int8, (u)int16, (u)int32, (u)int64
- decimal64
- string
- bits
- boolean
- enumeration
- union

1. A union can be e.g. an int or a string

YANG Models — Other modeling tools

- Import: Enables re-use of models
- Augment: Add new nodes to previously defined nodes
- Grouping: Set of nodes for re-use
- Container: Group of related nodes
- Feature: Allows marking part of the tree as optional

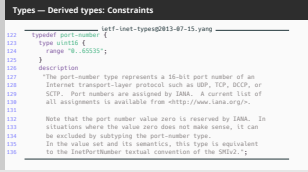
YANG Models — Other modeling tools
• Import: Enables re-use of models
• Augment: Add new nodes to previously defined nodes
• Grouping: Set of nodes for re-use
• Container: Group of related nodes
• Feature: Allows marking part of the tree as optional

1. import: e.g. IETF and ETSI have models for standards like IP addresses, SSH servers, TLS servers etc
2. import: e.g. openconfig has models for base routers and switches
3. feature: “When this feature is supported/enabled this part of the tree is instantiated

Types — Derived types: Constraints

ietf-inet-types@2013-07-15.yang

```
122   typedef port-number {
123     type uint16 {
124       range "0..65535";
125     }
126     description
127      "The port-number type represents a 16-bit port number of an
128       Internet transport-layer protocol such as UDP, TCP, DCCP, or
129       SCTP.  Port numbers are assigned by IANA.  A current list of
130       all assignments is available from <http://www.iana.org/>.
131
132       Note that the port number value zero is reserved by IANA.  In
133       situations where the value zero does not make sense, it can
134       be excluded by subtyping the port-number type.
135       In the value set and its semantics, this type is equivalent
136       to the InetPortNumber textual convention of the SMIv2.";
```

Types — Derived types: Deriving further

ietf-inet-types@2013-07-15.yang

```
193   typedef ipv4-address {
194     type string {
195       pattern
196         '(([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.){3}'
197       +  '([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])'
198       + '(%[\p{N}\p{L}]+)?';
199     }
```

ietf-inet-types@2013-07-15.yang

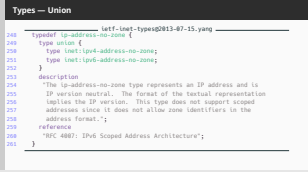
```
263   typedef ipv4-address-no-zone {
264     type inet:ipv4-address {
265       pattern '[0-9\.]*';
266     }
```


1. The patterns for derived types are AND-ed together

Types — Union

ietf-inet-types@2013-07-15.yang

```
248   typedef ip-address-no-zone {
249     type union {
250       type inet:ipv4-address-no-zone;
251       type inet:ipv6-address-no-zone;
252     }
253     description
254      "The ip-address-no-zone type represents an IP address and is
255       IP version neutral.  The format of the textual representation
256       implies the IP version.  This type does not support scoped
257       addresses since it does not allow zone identifiers in the
258       address format.";
259     reference
260      "RFC 4007: IPv6 Scoped Address Architecture";
261   }
```

Types — Grouping

my-example-model.yang

```
9   grouping endpoint {
10    description
11      "An IP endpoint, including the port";
12    leaf ip-address {
13      type inet:ip-address-no-zone;
14      mandatory true;
15    }
16    leaf port {
17      type inet:port-number;
18    }
19  }
```


Types — Using groupings

my-example-model.yang

```
21  container listen-addresses {
22    list listen-address {
23      key "name";
24      leaf name {
25        type string;
26      }
27      unique "ip-address port";
28      uses endpoint {
29        refine port {
30          default 25;
31        }
32      }
33    }
34  }
```

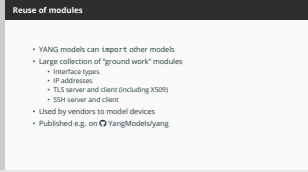

Types — Non-config

my-example-model.yang

```
36  container counters {
37    config false;
38    leaf connection-count {
39      type uint32;
40    }
41  }
```


Reuse of modules

- YANG models can `import` other models
- Large collection of “ground work” modules
  - Interface types
  - IP addresses
  - TLS server and client (including X509)
  - SSH server and client
- Used by vendors to model devices
- Published e.g. on YangModels/yang



1. Modules are published by IETF, ETSI, OpenDaylight, IEEE, Broadband Forum

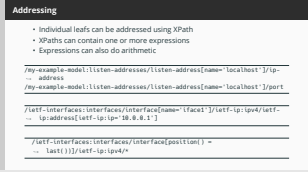
Addressing

- Individual leafs can be addressed using XPath
- XPaths can contain one or more expressions
- Expressions can also do arithmetic

```
/my-example-model:listen-addresses/listen-address[name='localhost']/ip-
↪  address
/my-example-model:listen-addresses/listen-address[name='localhost']/port
```

```
/ietf-interfaces:interfaces/interface[name='iface1']/ietf-ip:ipv4/ietf-
↪  ip:address[ietf-ip:ip='10.0.0.1']
```

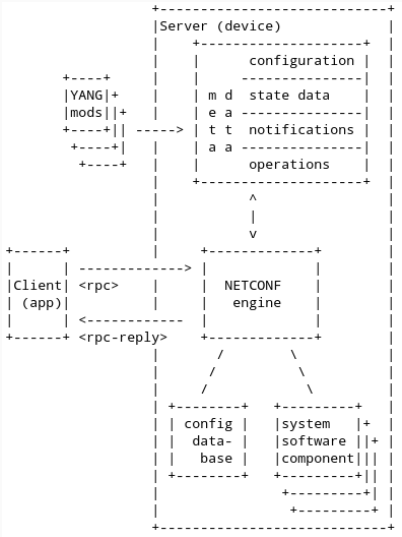
```
  /ietf-interfaces:interfaces/interface[position() =
  ↪  last())]/ietf-ip:ipv4/*
```

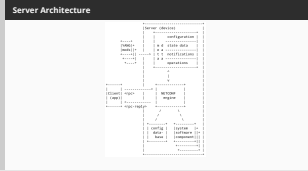



The solution

YANG and NETCONF servers

Server Architecture





1. Server, aka “a device” or “network element”

Datastores

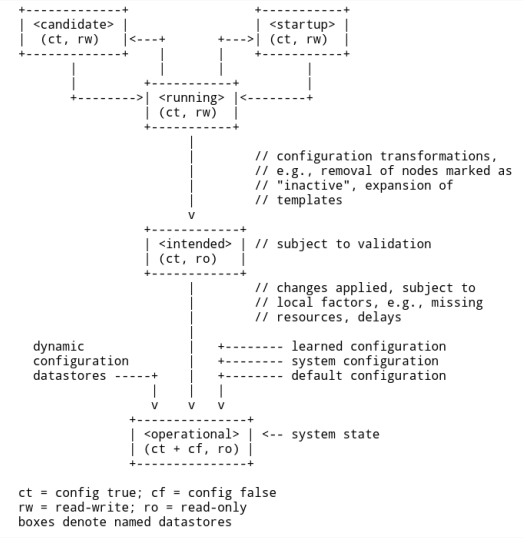
- Startup — Config to use upon boot
- Running — Current configuration
- Candidate — Used for staging config changes
- Operational – Contains the config and state of the system

Datastores

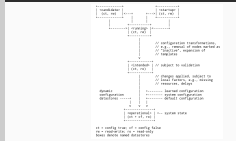
- Startup — Config to use upon boot
- Running — Current configuration
- Candidate — Used for staging config changes
- Operational – Contains the config and state of the system

1. Only Operational and Running are mandatory
2. On boot, startup is copied to the running
3. Running is copied to startup either implicitly or via a NETCONF rpc
4. Running config *must* be a valid tree
5. Candidate starts as a copy of running
6. Upon commit, candidate is copied to running
7. Operational is a “view”

Datastores



Datastores



YANG and NETCONF on *NIX?

YANG and NETCONF on *NIX?

Why should I care?

- Many applications could be “network functions”
- With the right orchestrator, have “versioned infra”
- Even without NETCONF, YANG is a powerful config language
  - Typed and constrained configuration items
  - Dhall (for Haskell) is similar, sans the tree
- Integrates into telco environments

Why should I care?

- Many applications could be "network functions"
- With the right orchestrator, have "versioned infra"
- Even without NETCONF, YANG is a powerful config language
  - Typed and constrained configuration items
  - Dhall (for Haskell) is similar, sans the tree
- Integrates into telco environments

1. cfgmgmt in telcos is usually not ansible or puppet, except for small departments

Software Support

- `libyang` – YANG parser and toolkit
- `sysrepo` – YANG Datastore
- `Netopeer2` – NETCONF server and client
- `pyang` – Python YANG validator, transformer and code generator
- YDK – YANG Development Kit by Cisco

Software Support

- `libyang` – YANG parser and toolkit
- `sysrepo` – YANG Datastore
- `Netopeer2` – NETCONF server and client
- `pyang` – Python YANG validator, transformer and code generator
- YDK – YANG Development Kit by Cisco

1. All these programs seem to be mostly written by one person
2. None of these are packaged for the major OS's
3. libyang has C++, java, python and javascript language-bindings, and includes validators
4. sysrepo – Implements all the datastores, can do subscriptions on paths for changes
5. sysrepo – Based on libyang
6. Netopeer uses libyang, sysrepo, libnetconf2 and libssh
7. The YDK can create APIs for different languages from YANG schemas to control network elements

pdns-sysrepo

- Uses `libyang` and `sysrepo`
- Configures PowerDNS Authoritative Server
- Stores zone-data in sysrepo
- Exposes a Remote Backend endpoint for PowerDNS
- PowerDNS/pdns-sysrepo

pdns-sysrepo
- Uses libyang and sysrepo
- Configures PowerDNS Authoritative Server
- Stores zone-data in sysrepo
- Exposes a Remote Backend endpoint for PowerDNS
- PowerDNS/pdns-sysrepo

1. It restarts the service when needed
2. Zone data is “just configuration” for the operator’s perspective
3. To the operator, a VM with pdns-sysrepo acts as a single DNS Server that is configured by NETCONF

Wrap up

Wrap up

In conclusion

- The networking world is not so different
- YANG and NETCONF are the industry standard
- It is a viable technology for *nix service configuration

In conclusion

- The networking world is not so different
- YANG and NETCONF are the industry standard
- It is a viable technology for *nix service configuration

Thank you!

Any questions?

CC-BY

1. **Thanks cfgmgtcamp for having me**

Thank you!

Any questions?

CC-BY

References and further reading

- http://www.netconfcentral.org/modulelist
- https://www.fir3net.com/Networking/Protocols/an-introduction-to-netconf-yang.html
- https://www.sysrepo.org/static/doc/html/
- https://www.slideshare.net/Cisco/software-defined-networking-and-network-programmability
- "Network Programming with YANG", Claise, Clarke, and Lindblad6

- http://www.netconfcentral.org/modulelist
- https://www.fir3net.com/Networking/Protocols/an-introduction-to-netconf-yang.html
- https://www.sysrepo.org/static/doc/html/
- https://www.slideshare.net/Cisco/software-defined-networking-and-network-programmability
- "Network Programming with YANG", Claise, Clarke, and Lindblad6