Network Device Properties
As Code

NANOG 75, San Francisco

Damien Garros, Network Reliability Engineer @ Roblox

Twitter @damgarros
Github @dgarros
Agenda

1. Introduction / Roblox
2. Network device properties definition
3. How are we managing device properties today?
4. How to manage Network device properties as code
5. Questions
1

Introduction / Roblox
What is Roblox?

- Educational platform for young software developers
- Gaming and Social platform
- Core audience for player is kids ages 9-12
- 2 Million Active developers
- 80+ Million monthly active users
- AS 22697
Rebuilt everything in 2018

- Couldn’t have done it without a proper Source of Truth
- The Source Of Truth is the Network Property Store
Start with the Source of Truth

- Integrate everything with it
- Your Source Of Truth is only as good as the quality of the data it contains
Vendors don’t tell you to deploy a Source of Truth because it introduces dependencies and requirements.

But it’s the most important part.
Automation Stack @ Roblox

Ansible / Jinja
Netbox Extensions

Netbox
Git

Configuration Generation

Network Source of Truth

Monitoring Alerting

Populate SOT

Observium
Influxdb / Grafana
Custom Collector
(REST / Netconf)
Custom Alert Manager
2

Network device properties
Network device properties

- Name
- IP addresses
- Cabling information / Peer properties
- Vlans
- BGP Peering
- Device specific info (ASN, etc.. )
Each device has a unique set of properties

- 1 template per role
- Unique set of properties per device
Your properties reflect your network design

- Network Design
- Naming Convention
- Cabling Convention
- Datacenter Layout
- Vendor Specific Information
People are failing to automate their network because they simplify the problem and assume that everything is homogeneous.
Example

SFO

NYC

Console server
Be prepared to manage MANY version of your properties

Network Design
Naming Convention
Cabling Convention
Datacenter Layout
Vendor Specific Information
For every rule, there is an exception. So you always follow the rule, except when there is an exception. In which case you follow a new rule based on that exception.
Be prepared to manage MANY version of your properties

Network Design
Naming Convention
Cabling Convention
Datacenter Layout
Vendor Specific Information
Properties @ Roblox

In 12 months we had to manage

- 42 different design revision just for the network
- Up to 9 versions for a given network device role

We also added

- 1 000 Network Devices
- 26 000 IP addresses
- 4 500 Prefixes
How are organizations managing Network device properties today?
What are the ways to generate these properties?

By Hand

By Script / Code
Pros / Cons with Script / Code approach

**Pros**
- Can generate large number of properties quickly
- Very flexible

**Cons**
- Hard to support multiple version of properties
- Need to “Write code” to adapt the design
- Hard to maintain
What are the ways to store these properties?

Source of Truth
Database
Git
All of the above

Network devices configuration
How to manage network device properties as Code?
Infrastructure as Code principles

- **Idempotent** > Always the same results

- **Version Control Friendly** > Input as text file, peer review

- **Safe & Predictable** > Plan everything before, know what changes will be made before you run it.
High level workflow

Network Builder

Source of Truth

Generate

Read
High level workflow - Plan & Apply

**PLAN**
- User
- Plan & Apply
- Network Builder
- Source of Truth

**APPLY**
- User
- Plan & Apply
- Network Builder
- Source of Truth
- Generate Diff

Read

Diff
Infrastructure as Code principles

- **Idempotent** > Always the same results
- **Version Control Friendly** > Input as text file, peer review
- **Safe & Predictable** > Plan everything before, know what changes will be made before you run it.
How to capture your design for a rack switch?

<table>
<thead>
<tr>
<th>Name</th>
<th>rsw, id of the cluster, name of the site separated by dashes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loopback</td>
<td>Any IP from the management network of the site</td>
</tr>
<tr>
<td>Uplinks</td>
<td>1x100G interface connected to each aggregation device</td>
</tr>
<tr>
<td></td>
<td>1 /31 allocated per interface from the /22 block reserved for point to point links</td>
</tr>
<tr>
<td>Console port</td>
<td>Any port on the the nearest console server</td>
</tr>
<tr>
<td>Server ports</td>
<td>A /24 network allocated from the /16 block reserved for server</td>
</tr>
</tbody>
</table>
Network Builder - Building blocks

- Resource Manager
- Create Properties
- Manage Diff

Source of Truth
Resource Manager

Manage all your resources as you manage IPs with DHCP

1. Possible to reserve resource in advance
2. Each resource allocated is associated with an ID
3. Same ID always get the same response.
Resource Manager - Example

- Create pools of resource identifiable with name or roles

- Query resources by defining
  - WHAT type of resource
  - From WHICH pool
  - WHO is requesting

```
asn: [ 65100, 65200 ]
prefixes:
  loopback: 10.10.10.0/24
  point-to-point: 10.128.0.0/22
```

```
WHAT: Loopback (/32)
WHICH: Loopback in SFO
WHO: device1
```

```
10.10.10.1/32
```

```
Query
```

```
Resp
```

Site SFO
Represent a property in a compact way

<LO4::sfo/loopback>

Variable Type
ASN   LO4
NET_IP
INT   VLAN

Pool Name / Path
Can be different per Variable Type

WHAT
WHICH

WHO is determined based on when this query is invoked
Define your device properties with variables

```python
name: "rsw1-1-sfo"
elevation: 30
type: qfx5100
role: rack-switch
ASN: "<ASN::sfo/private>"
network:
  lo0.0:
    ips:
      - addr: "<LO4::sfo/internal-loopbacks>"
      - addr: "<LO6::external-loopbacks>"
p2p:
  et-0/0/48:
    peer: "<DEV_INT::psw1-sfo/rack-switch>"
    ips:
      - addr: "<NET_IP4::sfo/point-to-point/31>"
  et-0/0/49:
    peer: "<DEV_INT::psw1-sfo/rack-switch>"
    ips:
      - addr: "<NET_IP4::sfo/point-to-point/31>"
```
Define your device properties with variables

```yaml
name: "rsw1-1-sfo"
elevation: 30
type: qfx5100
role: rack-switch
ASN: 65100
network:
  lo0.0:
    ips:
      - addr: 10.10.10.1/32
      - addr: 2020:1234:beef::756/128
p2p:
  et-0/0/48:
    peer: psw1-slo::et-0/0/1
    ips:
      - addr: 10.128.195.124/31
  et-0/0/49:
    peer: psw2-slo::et-0/0/1
    ips:
      - addr: 10.128.195.126/31
```
Create template of design

rack_switch_v1:
    name: "rsw{{id}}-1-sfo"
    elevation: 30
    type: qfx5100
    role: rack-switch
    ASN: "<ASN::sfo/private>"
    network:
        lo0.0:
            ips:
                - addr: "<LO4::sfo/internal-loopbacks>"
                - addr: "<LO6::external-loopbacks>"
    p2p:
        et-0/0/48:
            peer: "<DEV_INT::psw1-sfo/rack-switch>"
            ips:
                - addr: "<NET_IP4::sfo/point-to-point/31>"
Reuse templates across site and rack

```yaml
site: sfo
racks:
  101:
    racks: rack_switch_v1
    id: 1
  102:
    racks: rack_switch_v1
    id: 2
  103:
    racks: rack_switch_v1
    id: 3
```

```yaml
site: sfo
racks:
  101:
    name: rsw1-1-sfo
    ASN: 62100
    [ .. ]
  102:
    name: rsw2-1-sfo
    ASN: 62101
    [ .. ]
  103:
    name: rsw3-1-sfo
    ASN: 62102
    [ .. ]
```
Design is often dependent on the location

- Which Console Server should I connect to?
- Which Cluster / Pod is this rack part of?
- Which out-of-band device should I connect to?
- What is the out-of-band network for this rack?
- ...

ROBLOX
Context Resolution

Input File
text / yaml

Convert
Back to
Txt

Import
extract
Site
and
Rack info

Calculate / Resolve
Rack Specific info

Jinja

YML

TXT

YML

Site and Rack info

YML

TXT

import
site:
  name: sfo
  elevation: 30
  pod_id: 1
  rack:
    console: col-sfo
    mgmt_nwk: 10.0.0.0/16
Use Jinja to add contextual information

name: "rsw{{ id }}-{{rack.pod_id}}-{{site.name}}"
elevation: 30
rack_face: front
type: qfx5100
nb_role: rack-switch
ASN: "<ASN::{{site.name}}/private>"
network:
  lo0.0:
    ips:
      - addr: "<LO4::{{site.name}}/internal-loopbacks>"
      - addr: "<LO6::external-loopbacks>"
p2p:
et-0/0/48:
  peer: "<DEV_INT::psw1-{{site.name}}/rack-switch>"
  ips:
    - addr: "<NET_IP4::{{site.name}}/point-to-point/31>"
et-0/0/49:
  peer: "<DEV_INT::psw1-{{site.name}}/rack-switch>"
  ips:
    - addr: "<NET_IP4::{{site.name}}/point-to-point/31>"
### Network Builder - 3 main components

<table>
<thead>
<tr>
<th>Context Resolution</th>
<th>Variables Resolution</th>
<th>Apply / Create</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pull information specific per site and rack</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pod and Cluster Info</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Console Server</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OOB devices, lps…</td>
<td>Resolve/generate properties using the resource manager</td>
<td>Understand what already exist what needs to be created</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Apply the diff</td>
</tr>
</tbody>
</table>

**PLAN**

**APPLY**
Next steps

- Get feedback on this approach
- Open Source the resource manager
- Open Source the network builder
Thank You