

NetBox

Zero to Hero Live

Fundamentals

Training



Agenda

- General introduction
- Introduction to NetBox
- The NetBox model
- Break
- Working with NetBox
- Config Contexts
- Config Templates

TL;DR - Populate a NetBox instance from scratch, and render device config from the data!

Introduction



About your Trainer

- I'm Rick Donato
 - o Founder of Packet Coders (@packetcoders)
 - o Network automation architect
- Automating networks for ~15 years
- Background: Network Security, ADC, DC Networking, Automation, SDN/NFV
- Symantec, Rackspace, Nokia, Network to Code



About You

- Quick round the table
 - Name
 - NetBox areas of interest

Training Resources and Lab

Hands-on Environment

- NetBox Cloud - Details will be provided

Exercises

- GitHub Repo – <https://github.com/packetcoders/netbox-live-fundamentals>

Introduction to NetBox



Agenda

- What is NetBox?
- What NetBox is not
- Source of Truth 101
- NetBox key features
- NetBox within the network

What is NetBox?

- A popular open-source IPAM/DCIM.
- Originally developed by Jeremy Stretch.
- Built on Python (Django web framework).
- Allows you to model and document your infrastructure.
- Acts as a **Source of Truth** for your infrastructure/network.
- A key enabler for network automation.
- Ideal for infrastructure visibility throughout organizations and teams.

The screenshot shows the NetBox web interface with the 'Prefixes' list page. The left sidebar navigation includes 'Organization', 'Devices', 'Connections', 'Wireless', and 'IPAM'. Under 'IPAM', there are sections for 'IP ADDRESSES' (IP Addresses and IP Ranges), 'PREFIXES' (Prefixes and Prefix & VLAN Roles), 'ASNS' (ASN Ranges and ASNs), 'AGGREGATES' (Aggregates), 'RIRs', 'VRFs', 'Route Targets', 'VLANS' (VLANS and VLAN Groups), and 'OTHER' (FHRP Groups). The main content area displays a table titled 'Prefixes' with 14 results. The columns are: Prefix, Status, Children, VRF, Utilization, Tenant, Site, VLAN, Role, and Description. The table lists various IP prefixes such as 10.0.0.0/8, 10.1.1.0/30, 10.1.2.0/30, 10.1.3.0/30, 10.1.4.0/30, 10.1.5.0/30, 10.1.6.0/30, 10.2.1.0/30, 10.2.2.0/30, 10.2.3.0/30, 10.2.4.0/30, 10.2.5.0/30, 10.2.6.0/30, and 172.29.151.0/24. Each row includes a checkbox, status (Active), number of children, VRF (Global or Toronto Data Center), utilization (0.0% to 100.0%), tenant (Global Tech or —), site (Toronto Data Center or —), and VLAN, Role, and Description fields with dropdown menus. Buttons for 'Add', 'Import', and 'Export' are at the top right, along with search and filter options. The bottom right shows 'Per Page' and 'Showing 1-14 of 14'.

Prefix	Status	Children	VRF	Utilization	Tenant	Site	VLAN	Role	Description
10.0.0.0/8	Active	12	Global	0.0%	Global Tech	—	—	—	
+ 10.1.1.0/30	Active	0	Global	100.0%	—	Toronto Data Center	—	—	
+ 10.1.2.0/30	Active	0	Global	100.0%	—	Toronto Data Center	—	—	
+ 10.1.3.0/30	Active	0	Global	100.0%	—	Toronto Data Center	—	—	
+ 10.1.4.0/30	Active	0	Global	100.0%	—	Toronto Data Center	—	—	
+ 10.1.5.0/30	Active	0	Global	100.0%	—	Toronto Data Center	—	—	
+ 10.1.6.0/30	Active	0	Global	100.0%	—	Toronto Data Center	—	—	
+ 10.2.1.0/30	Active	0	Global	100.0%	—	Toronto Data Center	—	—	
+ 10.2.2.0/30	Active	0	Global	100.0%	—	Toronto Data Center	—	—	
+ 10.2.3.0/30	Active	0	Global	100.0%	—	Toronto Data Center	—	—	
+ 10.2.4.0/30	Active	0	Global	100.0%	—	Toronto Data Center	—	—	
+ 10.2.5.0/30	Active	0	Global	100.0%	—	Toronto Data Center	—	—	
+ 10.2.6.0/30	Active	0	Global	100.0%	—	Toronto Data Center	—	—	
172.29.151.0/24	Active	0	Global	3.1%	Global Tech	—	—	—	

NetBox is Not

However, NetBox does not provide:

- Network monitoring - active polling of services across devices
- DNS server - DNS translation services
- RADIUS server - Centralized AAA
- Configuration management - Configuration deployment
- Facilities management - Physical facilities management of cooling, power etc.



Source of Truth 101

What is an SoT?

- A trusted, centralized system that stores the intended state of the network.
- It is the most accurate, up-to-date representation of the network state.
- Ideally the network should not be used as a source of truth.
- Multiple SoR'd (system of records) can be unified into a SSoT.

SoT Benefits

- Drives automation workflows, through integrations and API's.
- Centralized form of documentation of the network.
- Reduces errors and dependencies (think schema enforcement).

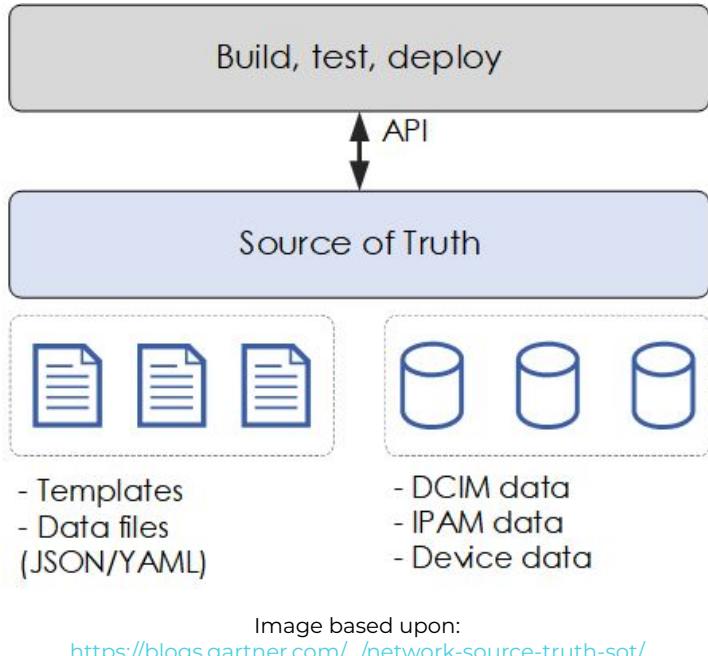


Image based upon:
<https://blogs.gartner.com/.../network-source-truth-sot/>

NetBox Key Features

- **Network modelling** (e.g regions, racks, VLANs, interfaces etc.).
- **API support**
 - REST (shown right) and GraphQL.
 - Webhooks - send an API request based on an event.
 - Pynetbox - Python library
- **Custom scripts** - provides the ability to perform custom actions using Python.
- **Custom reports** - provides ability to easily write reports using Python.
- **Job Scheduling** of custom scripts and reports.
- **Rack elevations** - view layout of racks, including image exports.

The screenshot shows the NetBox REST API documentation for the 'circuits' endpoint. At the top, it says 'NetBox REST API 0.5.1 (0.5) OAS3'. Below that is a 'Servers' dropdown set to 'NetBox' and an 'Authorize' button. The main area is titled 'circuits' and lists various HTTP methods and their corresponding URLs:

Method	URL
GET	/api/circuits/circuit-terminations/
POST	/api/circuits/circuit-terminations/
PUT	/api/circuits/circuit-terminations/
PATCH	/api/circuits/circuit-terminations/
DELETE	/api/circuits/circuit-terminations/
GET	/api/circuits/circuit-terminations/{id}/
PUT	/api/circuits/circuit-terminations/{id}/
PATCH	/api/circuits/circuit-terminations/{id}/
DELETE	/api/circuits/circuit-terminations/{id}/

NetBox Key Features (continued)

- **Custom fields** - extend what can be placed into NetBox (shown right).
- **Plugin support** - self developed or via plugin community.
- **Config Contexts** - add adhoc structured data to objects, with weighting.
- **Config Templates** - ability to provide templates and render directly within NetBox.
- **Synchronization** of data from remote sources (Git, S3).
- **NetBox Cloud** via NetBox Labs, to ease deployment.
- **Ansible module/plugin** support for reading and updating NetBox.

Editing custom field Ospf router id

Custom Field

Content types* DCIM > Device

Name* ospf_router_id Internal field name

Label Label Name of the field as displayed to users (if not provided, the field's name will be used)

Group name Group name Custom fields within the same group will be displayed together

Type* Text The type of data stored in this field. For object/multi-object fields, select the related object type below.

Object type ----- Type of the related object (for object/multi-object fields only)

Required If true, this field is required when creating new objects or editing an existing object.

Description Description

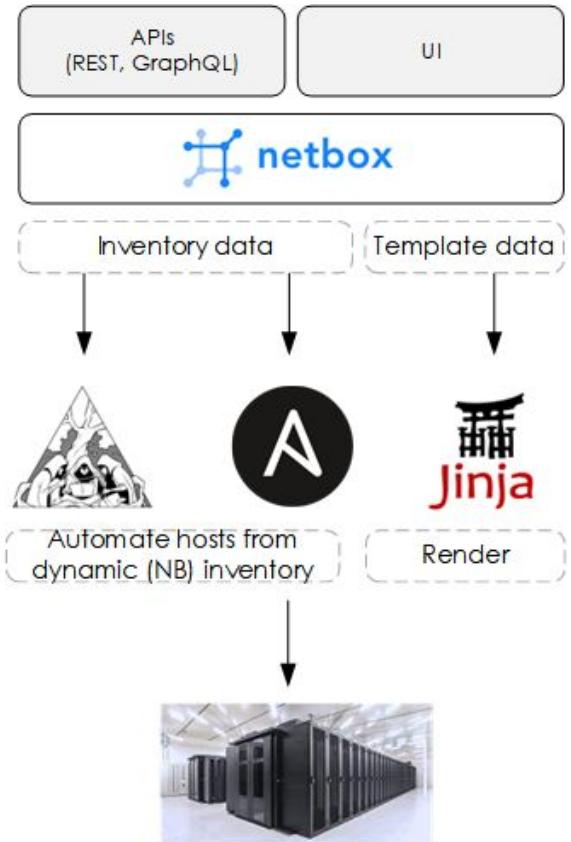
Behavior

Search weight* 1000 Weighting for search. Lower values are considered more important. Fields with a search weight of zero will be ignored.

NetBox within the Network

Use Cases

- Dynamic inventory for backups, audits of certain devices.
- Configuration rendering using NetBox data.
- Data export (used for config generation or testing)
- Documentation and reporting via the UI or custom reports.



Users and Groups

- Permissions can be assigned:
 - **User**
 - **Group**
- Supports **granular** permission assignment
- Permissions can be assigned at a:
 - **CRUD** level (Create, Update etc.)
 - **object** level (Device, IP Prefix etc.)
- Supports low-level restrictions at the Django ORM level.

Add permission

Name:

Description:

Enabled

Actions

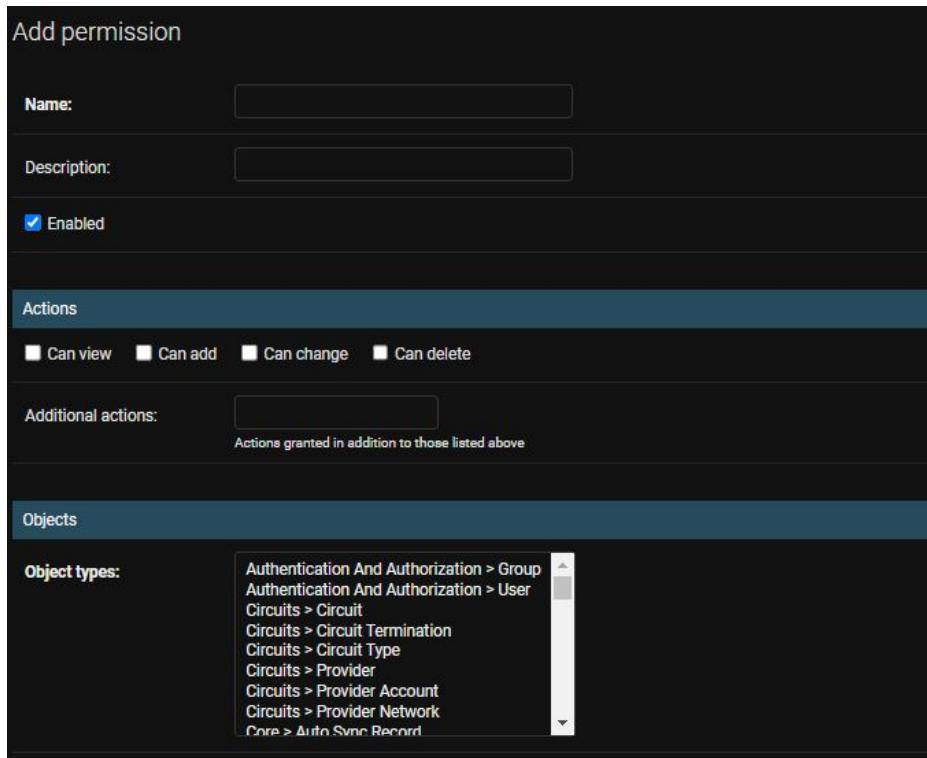
Can view Can add Can change Can delete

Additional actions:
Actions granted in addition to those listed above

Objects

Object types:

Authentication And Authorization > Group
Authentication And Authorization > User
Circuits > Circuit
Circuits > Circuit Termination
Circuits > Circuit Type
Circuits > Provider
Circuits > Provider Account
Circuits > Provider Network
Core > Auto Sync Record



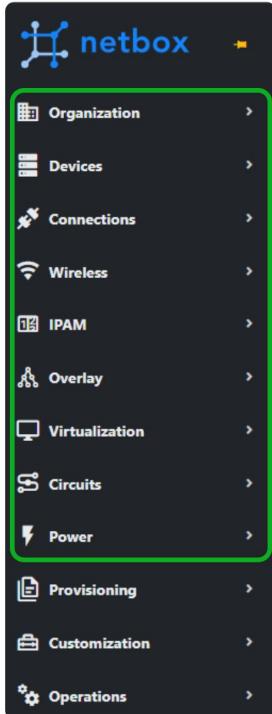
The NetBox Model



Agenda

- Introduction to NetBox Modelling
- Introduction to Global Tech
- Organization modelling
- Device modelling
- Connection modelling
- IPAM modelling

An Introduction to NetBox Modelling



NetBox can model an entire infrastructure (port level to regional building level)!

What can we model?

- **Organization** - Regions, Sites, Tenants, Racks
- **Devices** - Device types, components, platforms
- **Connections** - Cabling, interface connections, power connections
- **IPAM** - Prefixes, IP addresses, ASNs, VRFS, VLANs
- **Also:** Wireless, Circuits, Virtual Machines and Power

Introduction to Global Tech

Example company - Global Tech

- 2 parent regions
- 5 regions
- 5 sites (3 site groups)
- 1 x rack
- 5 x network devices (multi-vendor)

Note: Your Global Tech examples in your exercises will be slightly smaller



Organization

Regions - geographical grouping inc. parent regions.

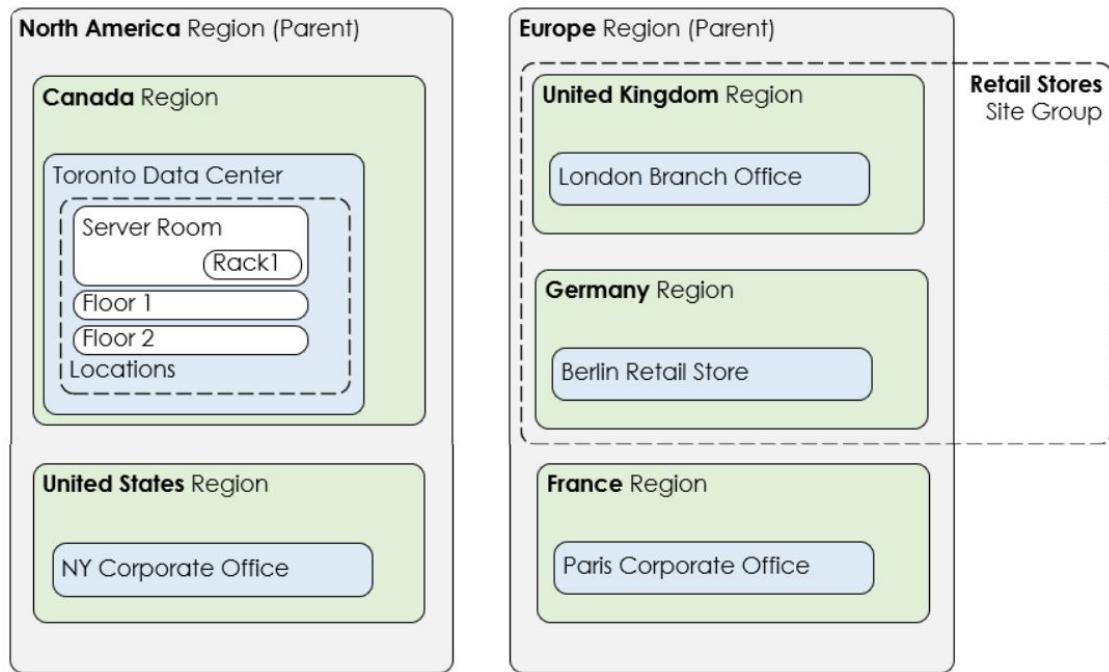
Sites - building or campus.

Site groups - collection of sites.

Locations - physical location within site.

Racks - elevation, space utilization.

Contacts - administrative association to objects



Over to NetBox

- Organizational modelling
- Regions
- Sites
- Locations

Devices

Device Components

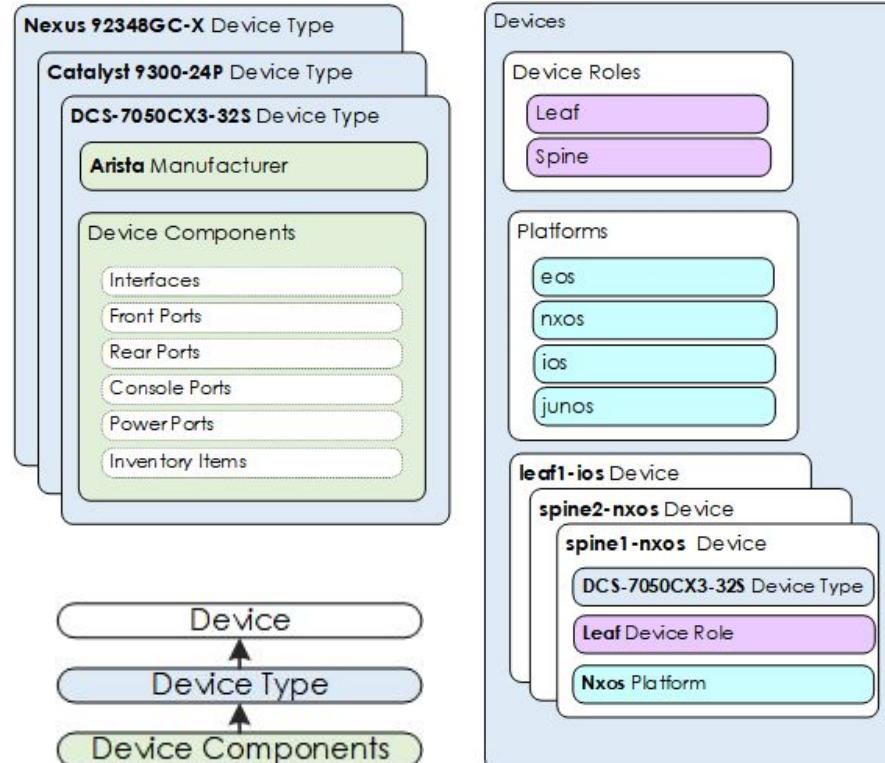
- Interfaces - MTU, speed/duplex, 802.1Q mode/tagging
- Modules - field-replaceable component.
- Front/rear ports - modelling of patching panelling.
- Inventory (PSU, CPU, line card, modules)

Device Types

- Manufacturers.
- Device type library.

Devices

- Platforms.
- Device Roles.
- “inherits” from device type.
- pulled by inventory plugins.



Over to NetBox

- Device modelling
- Device components
- Device types
- Devices

Connections

Cables

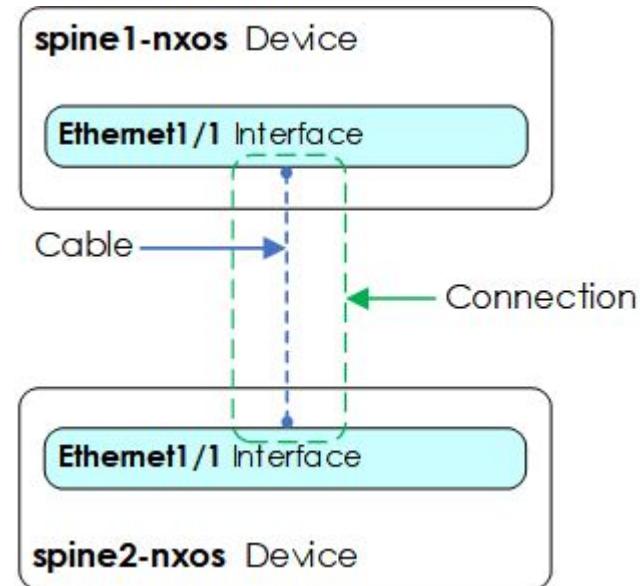
- cabling characteristics.
- type.
- length.

Wireless Links

- point-to-point wireless connections.

Connections

- interface connections.
- console connections.
- power connections.

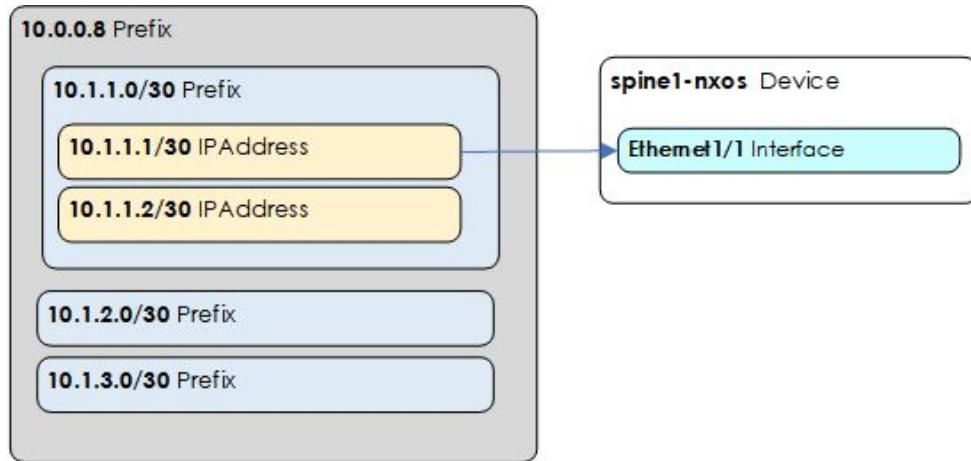


Over to NetBox

- Connection modelling
- Cables
- Interfaces

IPAM

- **Prefixes**
 - auto-hierarch`ing of prefixes.
- **IP assignment**
 - range or address
 - IPv4/6 support
- **Aggregates and RIR's**
 - assignment to prefixes.
 - utilization visibility
- **VLANs**
 - groups (site, regions)
 - roles (voice, data).
- **VRFs** - RT/RD support
- **ASNs**



Over to NetBox

- IP addresses
- IP ranges
- Prefixes

Break



Working with NetBox



Workbook 1 – Working with NetBox (Part 1)

Exercise 1: Working with a NetBox Regions and Sites

Task 1 – Creating a tenant

Task 2 – Creating regions

Task 3 - Creating sites

Exercise 2: Working with NetBox Devices

Task 1 – Create a device type via library

Task 2 - Create a device

Workbook 1 – Working with NetBox (Part 2)

Exercise 3: Working with NetBox Racks

Task 1 - Add devices to a rack

Task 2 - View the rack elevation

Exercise 4: Working with the NetBox IPAM

Task 1 - Create a prefix

Task 2 - Assign an IP from prefix

Task 3 - View prefix population

Exercise 5: Working with Custom fields

Task 1 - Creating a custom field

Task 2 - Populating a custom field

Config Contexts + Templates



Agenda

- What are Config Contexts?
- Config Context example
- Demo
- What are Config Templates?
- Demo

Config Contexts

What are Config Contexts?

- Also referred to as **context data**.
- **Supplemental data**.
- Applied to devices and/or VMs.
- Data can be defined via
 - **config contexts** (Global)
 - **local config contexts**. (Local)
- Flexible assignment.
- Supports **weights**
 - Higher weight takes precedence.
- Data can be consumed via **Config Templates** or via **API**

Config Context

Name* Syslog 1

Weight* 900

Description Description

Data

```
{  
  "syslog_servers": [  
    "192.168.220.1",  
    "192.168.210.2"  
  ]  
}
```

Enter context data in [JSON](#) format.

Is active

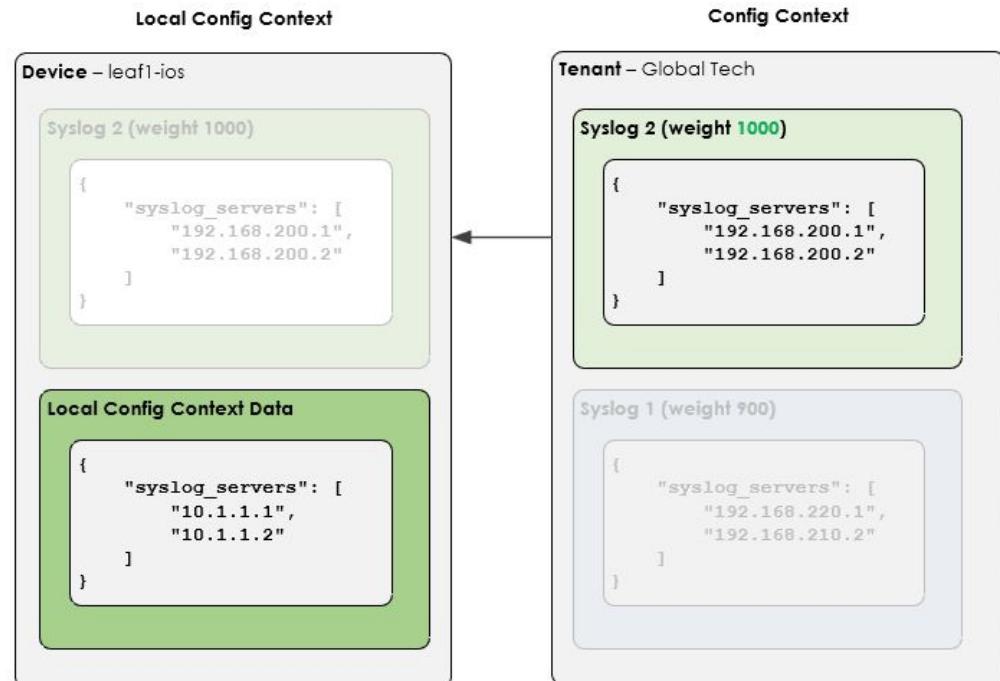
Config Contexts Example

Config Context

- **syslog_servers** (weight 1000)
- **syslog_servers** (weight 900)

Local Config Context

- **syslog_servers**



Over to NetBox

- Creating context data
- Creating local context data
- Utilizing context weights

Config Templates

What are Config Templates?

- Configuration templates based upon **Jinja**.
- Templates can consume data from:
 - **Config Contexts**
 - **NetBox model classes**
 - Supplemental **API data**
- Assigned to devices
- Templates can be **rendered** via the **UI**, or **API**.
- Support Jinja **environment parameters** (trim, lstrip).

Config Template

Name * Arista EOS

Description

Environment params {"trim_blocks": true, "lstrip_blocks": true}

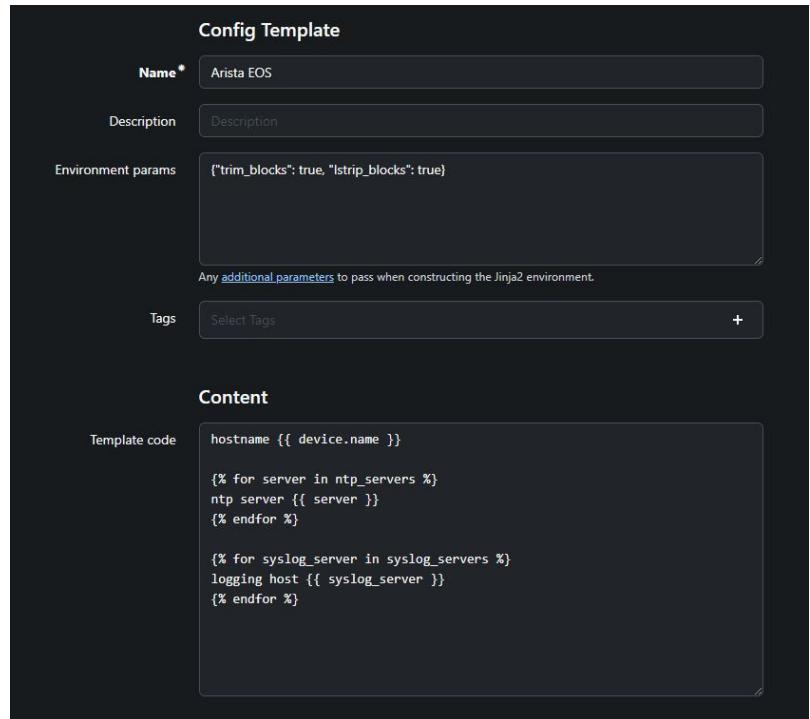
Any additional parameters to pass when constructing the Jinja2 environment.

Tags Select Tags +

Content

Template code

```
hostname {{ device.name }}  
  
{% for server in ntp_servers %}  
ntp server {{ server }}  
{% endfor %}  
  
{% for syslog_server in syslog_servers %}  
logging host {{ syslog_server }}  
{% endfor %}
```



Workbook 2 – Config Contexts

Exercise 1: Creating Config Contexts

Task 1 – Create Config Contexts

Task 2 – Validate Config Context Data

Exercise 2: Altering Config Context Weight

Task 1 – Alter Weight

Task 2 – Observe Outcome

Exercise 3: Create Local Config Contexts

Task 1 – Create Local Config Context

Task 2 – Validate the Rendered Config Context Data

Workbook 3 – Config Templates

Exercise 1: Creating a Config Template

Task 1 - Observe Available Context Data

Task 2 - Create a Template

Exercise 2: Render Config Template

Task 1 - Assign Template to Devices

Task 2 - Validate Rendered Config

Task 3 - Download Rendered Config

Exercise 3: Update Template Whitespace

Task 1 - Validate Rendered Config

Task 2 - Apply Whitespace Control (Bonus)

Task 3 - Validate Improvement (Bonus)

THANK YOU!



Enjoyed the Training?

- <https://forms.gle/4R8umJGCnEWNoYXX6>

