>>> network.toCode()

Nautobot Fundamentals - Day 1 of 2

Source of Truth and Automation Platform

David Cates

Network Automation Instructor

Introductions

To Network to Code and our courses

Instructor Introduction

David Cates

- Network Automation Instructor
- Nautobot App Developer

Who is Network to Code?

Network Automation Solutions Provider

Founded in 2014, we help companies transform the way their networks are deployed, managed, and consumed using network automation and DevOps technologies.

A Diverse Team, with Deep Expertise

Engineers and developers in network automation, software and security, with leadership from vendors, integrators, and top tier consulting firms - all drive value to our clients.

Vendor Neutral Community

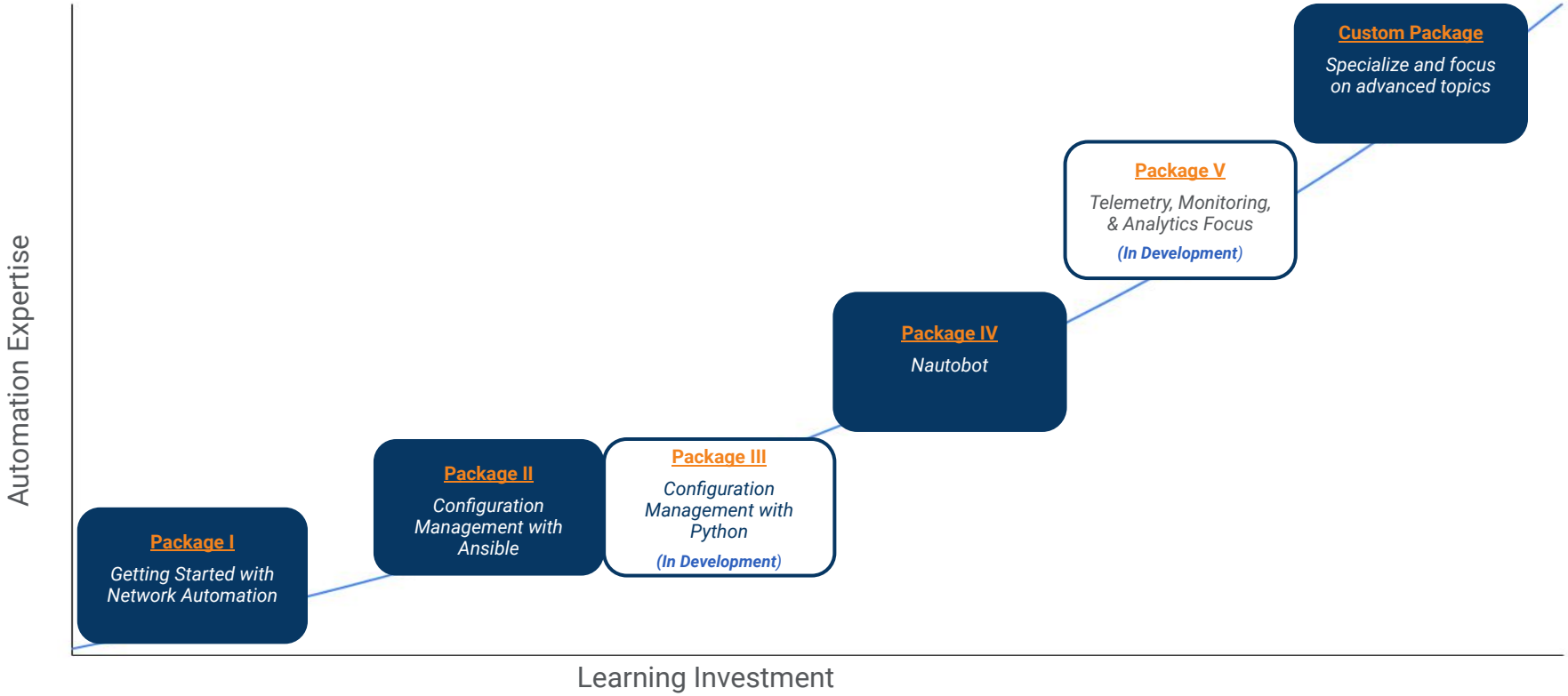
Partner with all OEMs, develop solutions with commercial and open source components. Host 19,000+ members and 300+ channels at slack.networktocode.com

Industry Recognized Thought Leaders

Working with clients across all industries and geographies, we promote a vendor and tool neutral approach, making automation a reality for any network.

>>> The Network Automation Journey: Enablement Packages

Pre-Built Packages from Start to Finish



>>> The Network Automation Journey: Enablement Milestones

Pre-Built Packages from Start to Finish

Package I

Getting Started with Network Automation

Kickstart into Network Automation *(2 hours of optional recordings)*

Network Programming & Automation *(5 days)*

Automation Journey: Next Steps *(2 hours)*

Package II

Configuration Management with Ansible

Kick-off Session, Choice of 3 Bite-sized Learning Sessions* *(4 hours)*

Network Configuration Testing with Batfish *(4 hours)*

Automating Networks with Ansible II *(3 days)*

Collaborative Workflows w/ Git & GitHub *(1 day)*

Package III

Configuration Management with Python

Kick-off Session, Choice of 3 Bite-sized Learning Sessions* *(4 hours)*

Network Configuration Testing with Batfish *(4 hours)*

Automating Networks with Python II *(3 days)*

Collaborative Workflows w/ Git & GitHub *(1 day)*

Package IV

Nautobot

Source of Truth and Nautobot Fundamentals *(2 days)*

Nautobot Extensibility *(1 day)*

Automating Nautobot with Python and Ansible *(1 day)*

Creating Nautobot Apps (plugins) *(3 days)*

Package V

Telemetry, Monitoring, & Analytics

Introduction to Telemetry: Data Collection, Dashboarding & Alerts *(2.5 days)*

Choose your speciality:

Prometheus Stack *(1 day)*

Influx Stack *(1 day)*

Elastic Stack *(1 day)*

Custom Package

Guided a la carte;

Choose min. 3 days up to 10 of workshops, bootcamps, and Bite-sized Learning

Bootcamps
Network Programming & Automation
Network Automation with Python
Network Automation with Ansible
Intermediate Ansible
NetBox Plugins
Workshops
Network Configuration Testing with Batfish
Collaborative Workflows w/ Git & GitHub
Using Jinja within Ansible
Jinja2 Deep Dive
Extending Ansible
SoT: NetBox Fundamentals
Nornir
Telemetry Deep Dive
Cisco NSO
… and more
Bite-sized Learning Sessions

>>> Enablement Milestones: Package IV

Nautobot (4 or 7 days)

Package IV

Nautobot

Source of Truth and Nautobot Fundamentals *(2 days)*

- *Understand the fundamental design philosophy of Source of Truth, Systems of Record and the benefits of tracking both the desired state and the operational state of enterprise networks*
- *Learn how to leverage Nautobot to define your network's intended state*

Nautobot Extensibility *(1 day)*

- *Learn how Nautobot extensibility features can be used to tailor Nautobot to integrate into almost any environment*

Automating Nautobot with Python and Ansible *(1 day)*

- *Take the next step and become familiar with Nautobot APIs and how to consume them*
- *Deep dive into the Nautobot REST API, GraphQL API, pynautobot, and the Nautobot Ansible collection*

Prerequisites: Package I

Creating Nautobot Apps (plugins) *(Optional Add-on: 3 days) *Advanced**

- *Introduces how Nautobot Apps (or plugins) extend the existing functionality of Nautobot*
- *Possess the knowledge and skills to write your own application (plugin) to extend Nautobot functionality*

>>> Nautobot Cloud & Managed Services

NTC Nautobot Services	Nautobot Support	Cloud*	NAaaS
Expert Support Backed by Nautobot Developers	✔	✔	✔
Access to Nautobot Support Knowledge Base	✔	✔	✔
24x7 Online Ticketing System	✔	✔	✔
Contribute Bug Fixes for Nautobot & Nautobot Apps	✔	✔	✔
Up to three (3) Nautobot Instances	✔	✔	✔
Scheduled Nautobot Support Maintenance Windows - 24x7	✔	✔	✔
Monthly Usage Report	✔	✔	✔
Quarterly Business Review with NTC	✔	✔	✔
Nautobot Deployed with HA Redundancy (App, Storage)		✔	✔
Nautobot Upgrades & Monitoring		✔	✔
Quarterly Strategic Roadmap Sessions		✔	✔
Access to NTC Network Automation Architects		✔	✔
Network Automation Tasks			✔
Automated Network Configuration Compliance			✔
Automated Network OS Upgrades with Pre/Post Check Validation			✔
NTC Fast Track Solutions			Add-On

**Deployed and managed by NTC in a client-provided AWS VPC. On-prem coming soon.*

Attendee Introduction

>>> Student Introductions

- *Name*
- *Job/Role*
- *Experience*
- *Programming/Automation/NetEng Experience*

Course Overview

>>> Housekeeping

- *Length & Time*
- *Breaks*
- *Feedback: a survey link will be sent on the last day.*
  Please take the time to fill this out.
  We value your feedback!

Course Objectives

- Understand what a Source of Truth is and the types of data that can be stored in a Source of Truth
- Understand concepts and terms such as Systems of Record, Authoritative Source of Data, and Single Source of Truth
- Articulate the difference between operational state (observed) and intended state (desired) data
- Understand the role a Source of Truth plays within a network automation platform
- Learn how to Deploy Nautobot, its dependencies, and the Welcome Wizard Application
- Navigate the Nautobot UI and populate Nautobot with Source of Truth data

>>> Course Agenda

1. Why Network Automation?
2. Nautobot Overview and Installation
3. Nautobot Organization and Devices
4. Nautobot IPAM to Circuits
5. Nautobot Apps
6. Nautobot Programming
7. Nautobot Extensibility

>>> Lab and Lecture Material

Access to Lecture and Lab course materials

- You need to be logged in to your free GitHub.com account to access the course materials

- If you gave NTC your public Github username, *please accept the invitation in your email* (the one connected to your Github account) to the NTC Training Org, which will give you access to the Nautobot Fundamentals repository.
  - https://github.com/ntc-training/nautobot-fundamentals

- If you need extra time to complete your labs, we can extend your pod access for one day. Please email training-pods@networktocode.com *prior to noon on the last day of training* to request the extra day and include your pod number.

Lab 0

Accessing the Lab Environment

Course Agenda

1. Why Network Automation?
2. Nautobot Overview and Installation
3. Nautobot Organization and Devices
4. Nautobot IPAM to Circuits
5. Nautobot Apps
6. Nautobot Programming
7. Nautobot Extensibility

Why Network Automation?

>>> Positive Outcomes

How to measure the value of network automation

The following Key Performance Indicators (KPIs) provide the data needed to show the benefits of network automation including: increased reliability, operational efficiency, acceleration of delivering new products, reduced operational cost, risk mitigation, compliance improvements, and overall increased business agility.

% of total network changes that are automated	**% of emergency changes**	**% of backed out changes**	**% of incidents caused by changes**	**Number of Network devices per network administrator**
Average time per phase for new network service	**% of successful changes**	**Average time of roll-back**	**% of out-of-hours changes**	**Time to detect network failure**
Average time per phase for network change	**% of compliance errors**	**Number of changes in the backlog**	**Resources used and funds spent on changes**	**Time to repair network failure**

>>> Positive Outcomes

Predict these benefits from network automation

Lower Opex

- Simplify your underlying infrastructure
- Consume fewer hours configuring, provisioning, and managing network services

Improve Reliability and Security

- Reduce the chance for human errors
- Deliver a higher level of services
- Drive consistency across the network

Increase Innovation

- Increase productivity by allowing humans to do higher level work
- Allocate more time to drive business strategy and innovation

Greater Insight and Network Control

- Gain more visibility into the network
- Allow IT operations to become more responsive to change through analytics

Increase Business Agility

- Develop new network operational models
- Improve time-to-market
- Build, test, deploy, and optimize new network services quickly and continuously

Systems of Record

QUIZ

Which spreadsheet System of Record could an automated system detect as correct (without human intervention)?

a. File with the most recent date

b. The largest file

c. File created by the most senior, technical person

d. Any spreadsheet

e. All of the spreadsheets

f. None of the spreadsheets

QUIZ

Which spreadsheet System of Record could an automated system detect as correct (without human intervention)?

a. File with the most recent date

b. The largest file

c. File created by the most senior, technical person

d. Any spreadsheet

e. All of the spreadsheets

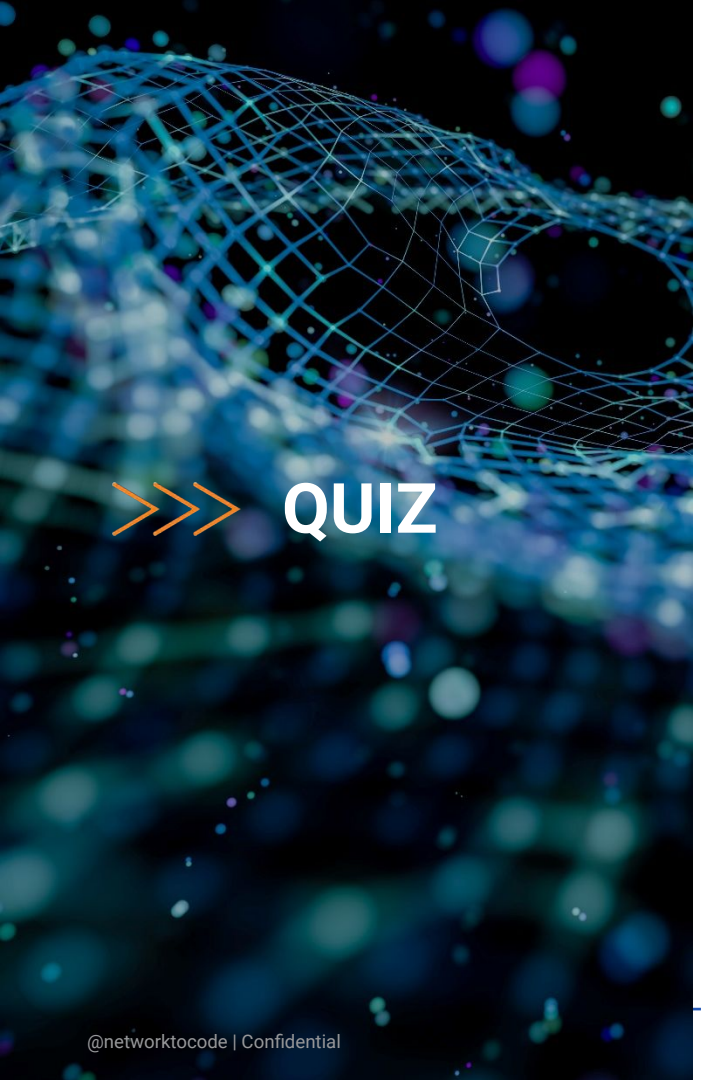
f. None of the spreadsheets

>>> What is a System of Record (SoR) vs a Source of Truth (SoT)?

- You will likely have multiple **Systems of Record (SoR)** for different parts of the infrastructure.
  - e.g. IPs will be in one database, Servers in another.
  - Use the right data store - one size does NOT fit all
  - Ensure all systems have an API, versioning, changelogs/audit-trails

- A **Source of Truth (SoT)** acts as an aggregation layer that combines the necessary input data for your workflows.
  - Reduce data duplication to improve accuracy and reliability
  - It is a requirement for short-term wins and longer-term progress
  - Example: Infoblox is the SoR for IPs, but you have automated replication to Nautobot for integration with its DCIM inventory

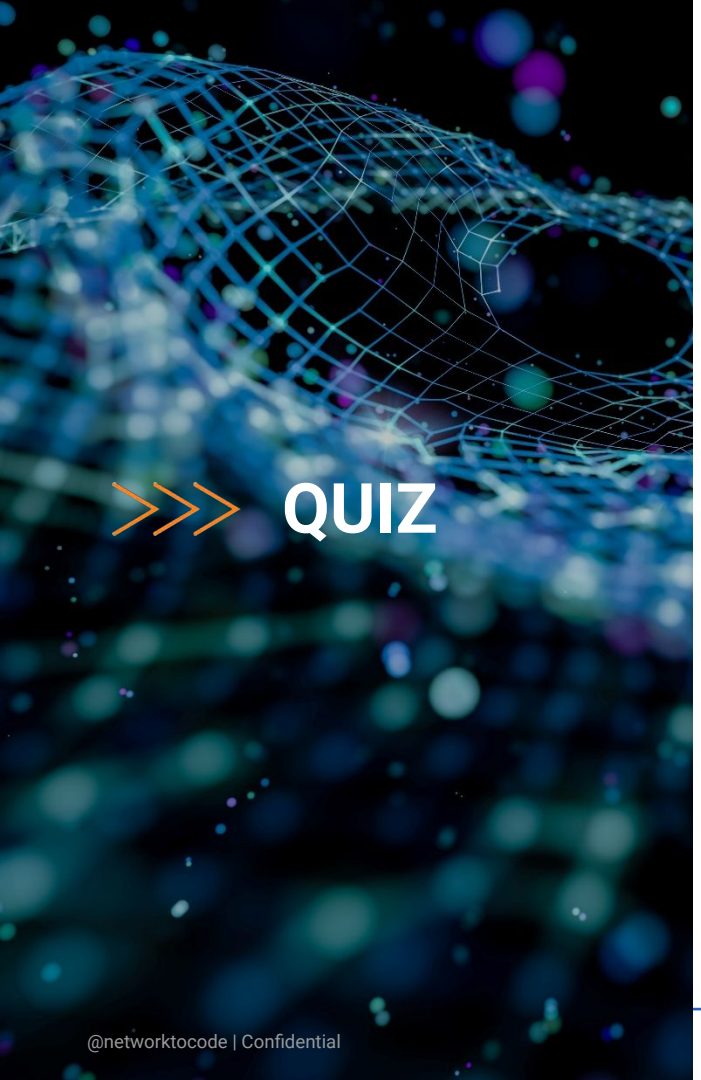
Intended versus Operational State



Should the network be your Source of Truth?

a. Yes

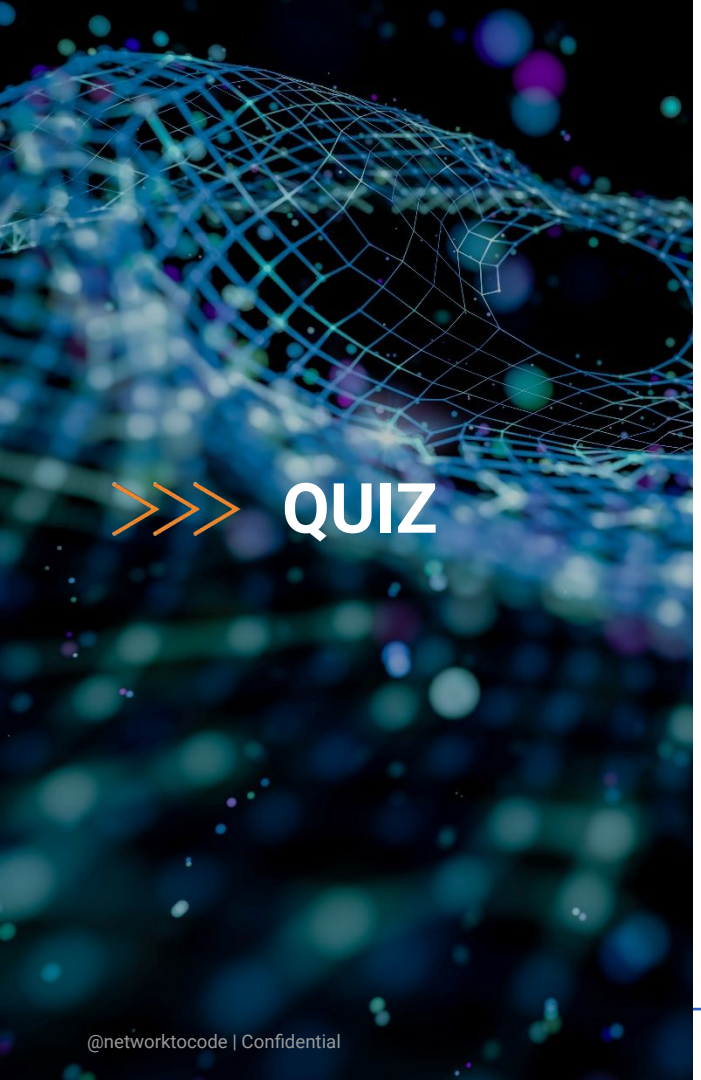
b. No



Should the network be your Source of Truth?

a. Yes

b. No



Which of the following are generally considered Sources of Truth data?

a. CRC errors

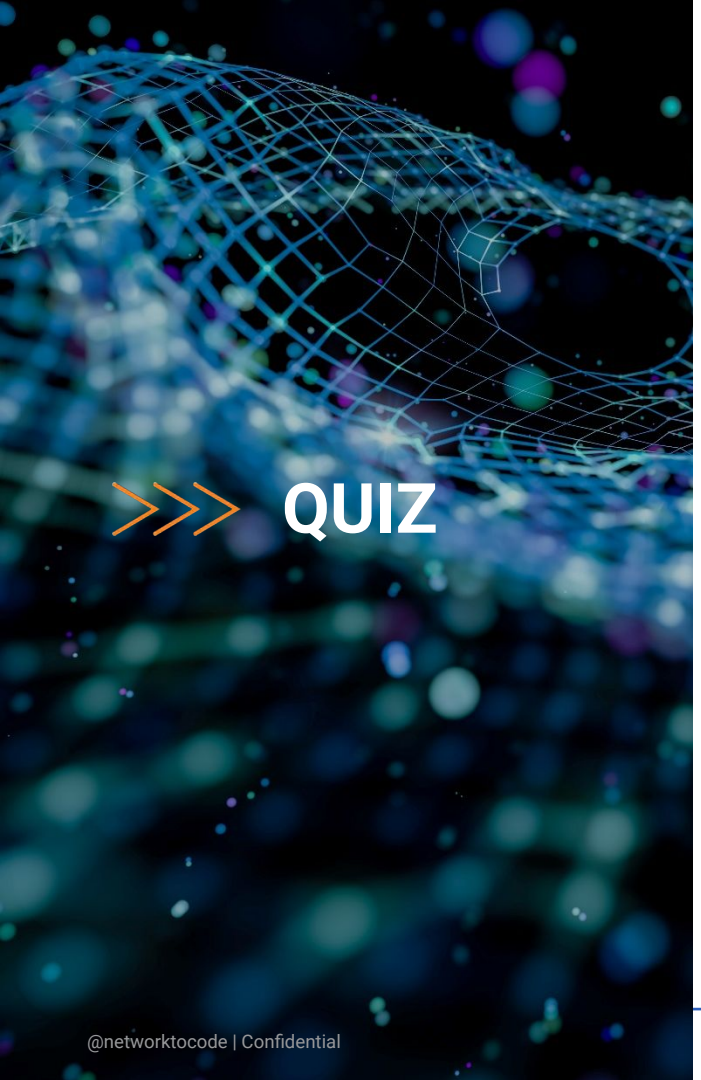
b. Interface admin status (Up/Down)

c. IP addresses

d. Light levels

e. Planning states for devices, racks, sites, etc (Active, Planned, Decommissioned, etc)

f. VLAN assignments



Which of the following are generally considered Sources of Truth data?

a. CRC errors

b. Interface admin status (Up/Down)

c. IP addresses

d. Light levels

e. Planning states for devices, racks, sites, etc (Active, Planned, Decommissioned, etc)

f. VLAN assignments

>>> Discussion

Can the network be my source of truth?

>>> Discussion

No. The network is your source of *reality*.

>>> Discussion

No. The network is your source of *reality*.

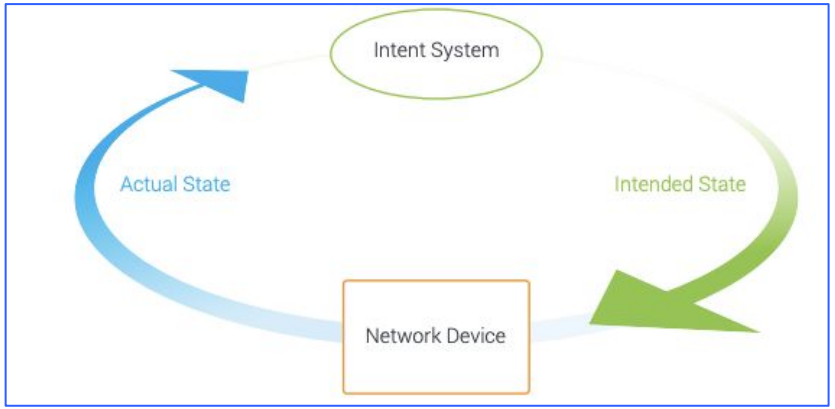
Truth != *reality*

>>> Networks and Intent

- Example: Network architecture documents describe the intended network architecture
  - Those docs are intentionally crafted to allow the network to accomplish the business objectives
  - If the network architecture does not match those documents, the network is wrong!

>>> Networks and Intent

- The Source of Truth (SoT) holds **data** for the *intended* network *state*
  - Like an architecture document, the data in the SoT is ***intentionally*** crafted to reflect intended state
  - If the network does not reflect the intended state, the network is wrong!



What is a Network Source of Truth?

>>> Source of Truth Data

- Source of Truth Data is any data that reflects the *intended state* of the network

- Examples:
  - IP addresses
  - VLAN assignments
  - Planning states for devices, racks, sites, etc (Active, Planned, Decommissioned, etc)
  - Admin status for any given interface (Up/Down)
  - etc.

>>> Source of Truth Data

What is NOT Source of Truth Data?

- Transient or performance data:
  - CRC errors
  - Light levels
  - Network throughput at some point in time, etc.

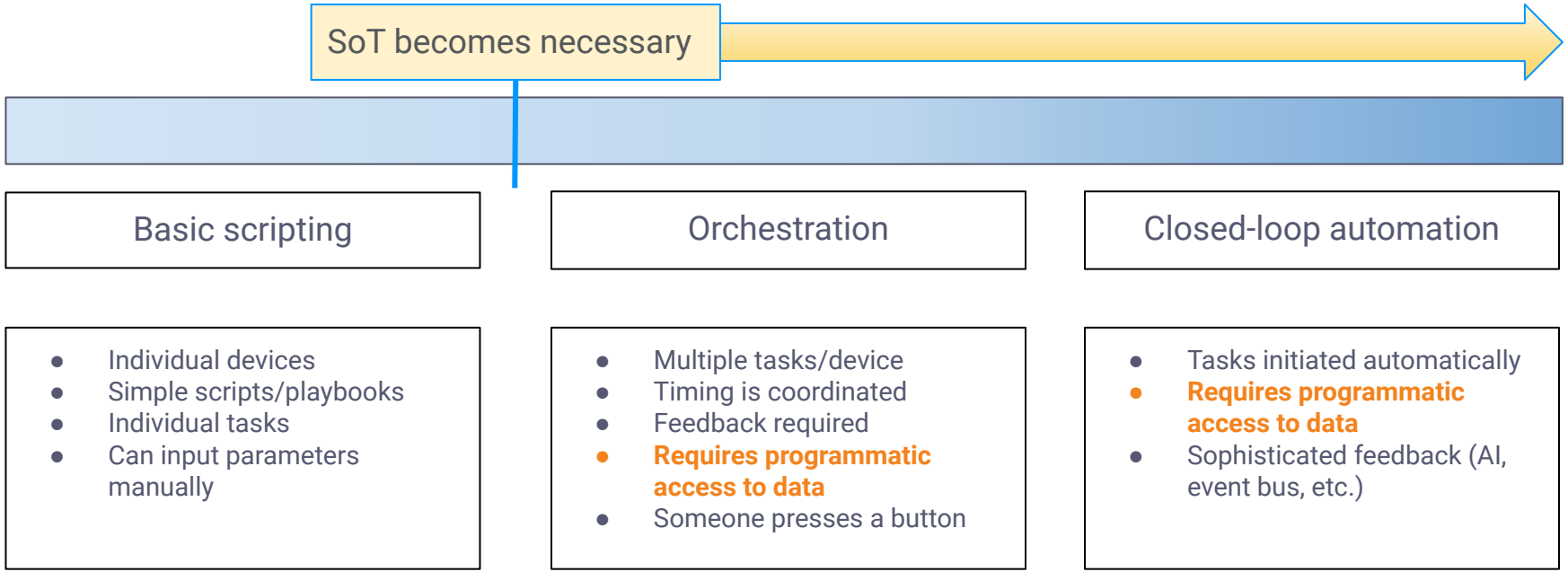
>>> A Source of Truth is Necessary For Automation

- There is a spectrum of capabilities
- Moving to the right on the spectrum requires more sophistication

Basic scripting	Orchestration	Closed-loop automation
• Individual devices • Simple scripts/playbooks • Individual tasks • Can input parameters manually	• Multiple tasks/device • Timing is coordinated • Feedback required • **Requires programmatic access to data** • Someone presses a button	• Tasks initiated automatically • **Requires programmatic access to data** • Sophisticated feedback (AI, event bus, etc.)

>>> A Source of Truth is Necessary For Automation

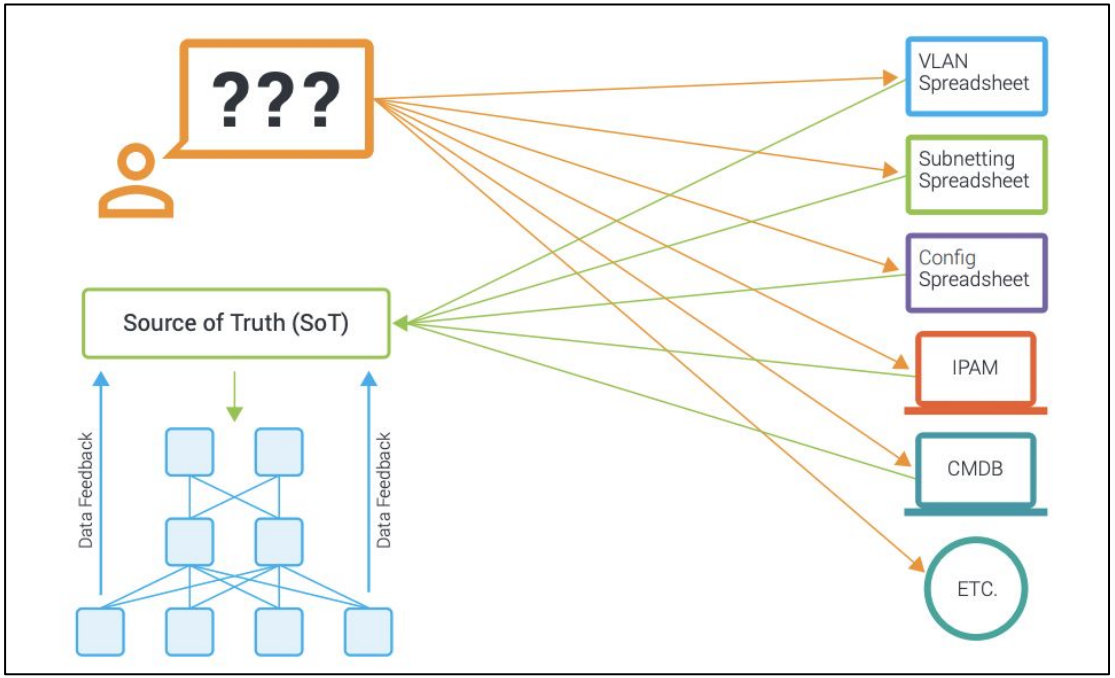
- There is a spectrum of capabilities
- Moving to the right on the spectrum requires more sophistication



>>> Centralization Matters

- SoT interacts with multiple authoritative sources
- Automation interacts with a single SoT

The fewer the variables there are in automation, the better the automation scales



>>> Centralization Matters

- A centralized SoT ensures that the data/intent questions are asked and answered in the planning phase, not the execution phase

- From a business perspective the network is often considered to be a single resource (“the network”)
  - A single SoT should reflect the desired state of the resource

Why is Source of Truth Data Important?

>>> Why is SoT Data Important?

- Configurations do not provide **tracking**
  - Who created the configuration?
  - Why did they make the change?
  - When did they make the change?

```
interface GigabitEthernet1/0/2
 description CIO Port
 switchport mode access
 switchport access vlan 101
 storm-control broadcast level 8.00
 storm-control action trap
```

When changed?

Why not 100?

Who added?

>>> Why is SoT Data Important?

- Configurations do not provide **tracking**
  - Who created the configuration?
  - Why did they make the change?
  - When did they make the change?

- Ensures **consistency**
  - Disaggregate data from the configurations

```
interface GigabitEthernet1/0/2
 description CIO Port
 switchport mode access
 switchport access vlan 101
 storm-control broadcast level 8.00
 storm-control action trap
```

When changed?

Why not 100?

Who added?

```
interface {{ int['name'] }}
 description {{ int['desc'] }}
 switchport access vlan {{ int['vlan'] }}
 switchport mode access
 logging event link-status
 speed Auto
 duplex Auto
```

+

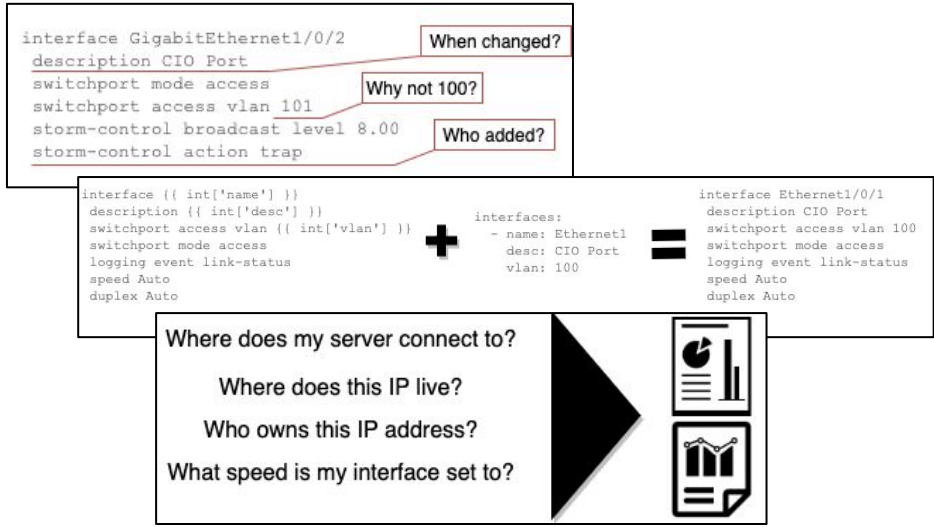
```
interfaces:
  - name: Ethernet1
    desc: CIO Port
    vlan: 100
```

=

```
interface Ethernet1/0/1
 description CIO Port
 switchport access vlan 100
 switchport mode access
 logging event link-status
 speed Auto
 duplex Auto
```

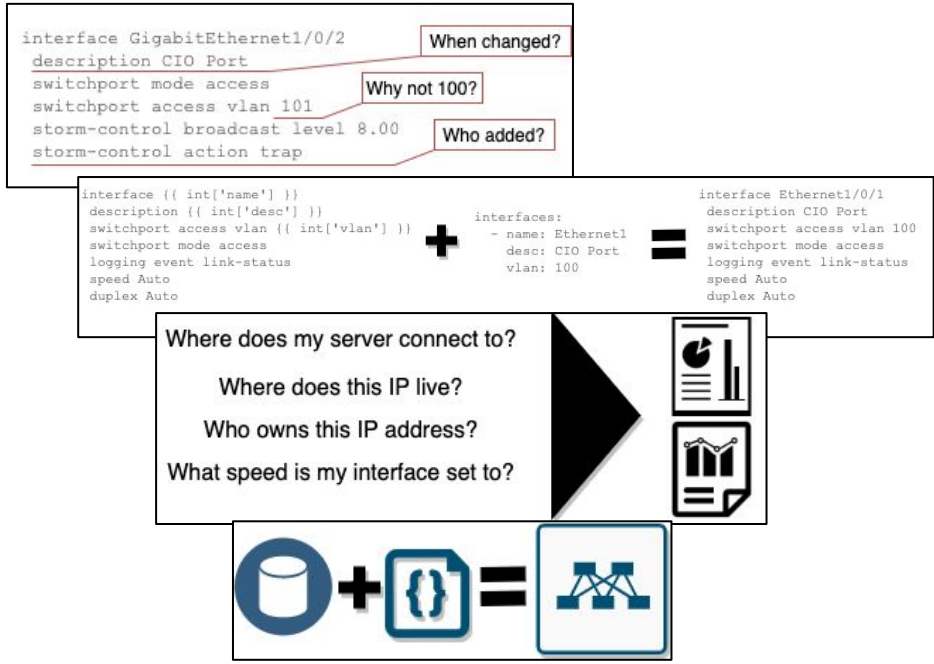
>>> Why is SoT Data Important?

- Configurations do not provide **tracking**
  - Who created the configuration?
  - Why did they make the change?
  - When did they make the change?
- Ensures **consistency**, if data cannot fit in, can't be added
  - Disaggregate data from the configurations
- Configurations are **not "queryable",** but data is
  - The data is valuable outside of the network engineering teams
  - Must take valuable engineering time to review
  - Data becomes auditable, rather than configuration



>>> Why is SoT Data Important?

- Configurations do not provide **tracking**
  - Who created the configuration?
  - Why did they make the change?
  - When did they make the change?
- Ensures **consistency**, if data cannot fit in, can't be added
  - Disaggregate data from the configurations
- Configurations are **not "queryable",** but data is
  - The data is valuable outside of the network engineering teams
  - Must take valuable engineering time to review
  - Data becomes auditable, rather than configuration
- Provides ability to document network
  - Documentation can be auto-generated with visuals and reports



>>> Why is SoT Data Important?

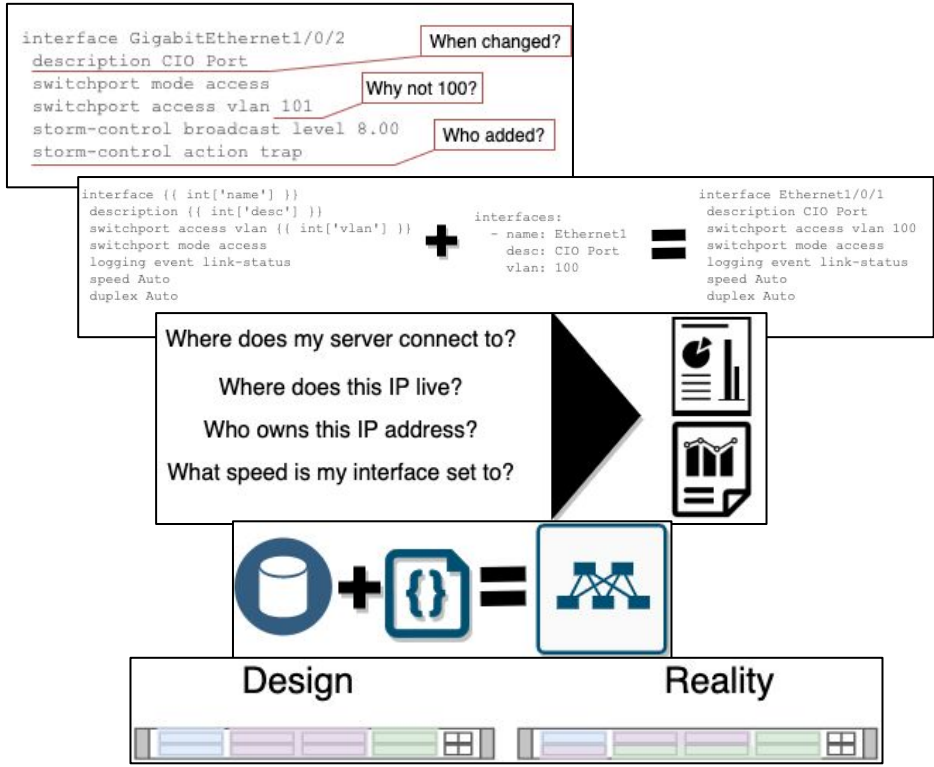
- Configurations do not provide **tracking**
  - Who created the configuration?
  - Why did they make the change?
  - When did they make the change?

- Ensures **consistency**, if data cannot fit in, can't be added
  - Disaggregate data from the configurations

- Configurations are **not "queryable"**, data is
  - The data is valuable outside of the network engineering teams
  - Only method is to take valuable engineering time to review
  - Data becomes auditable, rather than configuration

- Provides ability to document network
  - Documentation can be auto-generated visuals or reports

- Designs are intended to be consistent
  - Data allows a design to be flexible



>>> Summary

- The role of a Source of Truth is to capture the intended (desired) state of the infrastructure.

- SoT ensures data is accurate, reliable, traceable, and easy to access.

- Traditionally, network devices have been the SoT:
  - Intent and Operational result in one place (Configuration)
  - Documentation is hard to track, quick to become outdated
  - Designs, Deployment Sheets, Project Inventories
  - Infra life cycle is iterative, documentation usually trails behind
  - Update config first, maybe get to diagrams later
  - Config backups, Manual changes, Manual discovery
  - Makes automation difficult
    - Hard to abstract the original intent from the end result

>>> Conclusion

A centralized, authoritative Source of Truth is necessary for automation.

Course Agenda

1. Why Network Automation?
2. Nautobot Overview and Installation
3. Nautobot Organization and Devices
4. Nautobot IPAM to Circuits
5. Nautobot Apps
6. Nautobot Programming
7. Nautobot Extensibility

What is Nautobot?

What is Nautobot?

- *Source of Truth (SoT)*
- *Automation platform*
- *Application platform*

>>> Nautobot Use Cases

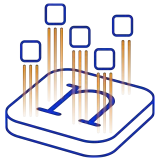
Network Source of Truth

- Devices
- IP Addresses
- VLANs
- ASN
- …
- Custom

- User-Defined Relationships
- Custom Fields
- Data Validation
- Git as a Data Source

Network Automation Platform

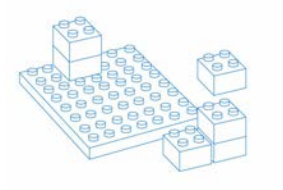
- Use Open Source Apps
- Build Custom Apps
- Save 70% development time using the platform



Powered by APIs and NetDevOps extensibility & integrations

{ REST } GraphQL

webhooks git



Extensible Plugin System

>>> Nautobot Use Cases

Network Source of Truth

- Devices
- IP Addresses
- VLANs
- ASN
- …
- Custom

- User-Defined Relationships
- Custom Fields
- Data Validation
- Git as a Data Source

Network Automation Platform

- Use Open Source Apps
- Build Custom Apps
- Save 70% development time using the platform

Powered by APIs and NetDevOps extensibility & integrations

{ REST } GraphQL

webhooks git

Extensible Plugin System

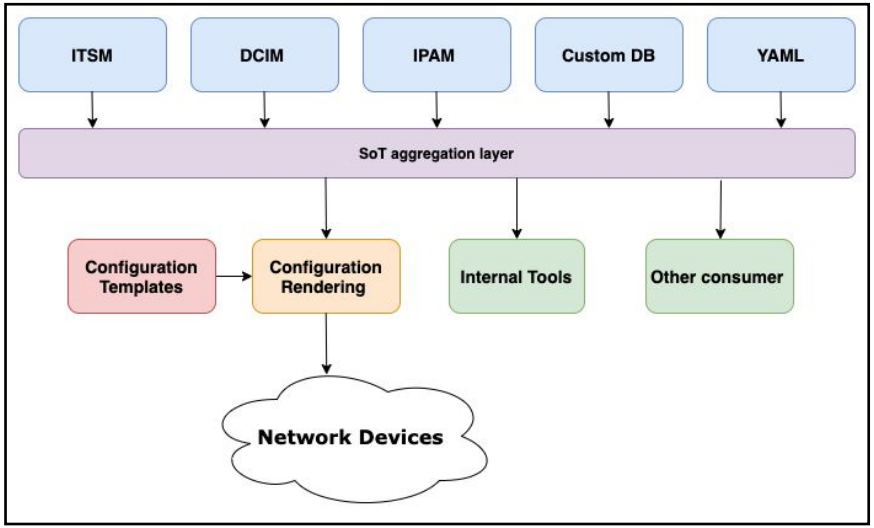
>>> What is Nautobot?

*Nautobot is a **centralized** source for authoritative data on intended state.*

>>> What is Nautobot?

Nautobot can **consolidate data** from disparate sources of record into a single, centralized Source of Truth

- Example: Access an external IPAM system to import authoritative data



>>> What is Nautobot?

- Nautobot Can Push Authoritative Data to Secondary Sources
  - If other orgs need to access data from their own systems, Nautobot can update those systems

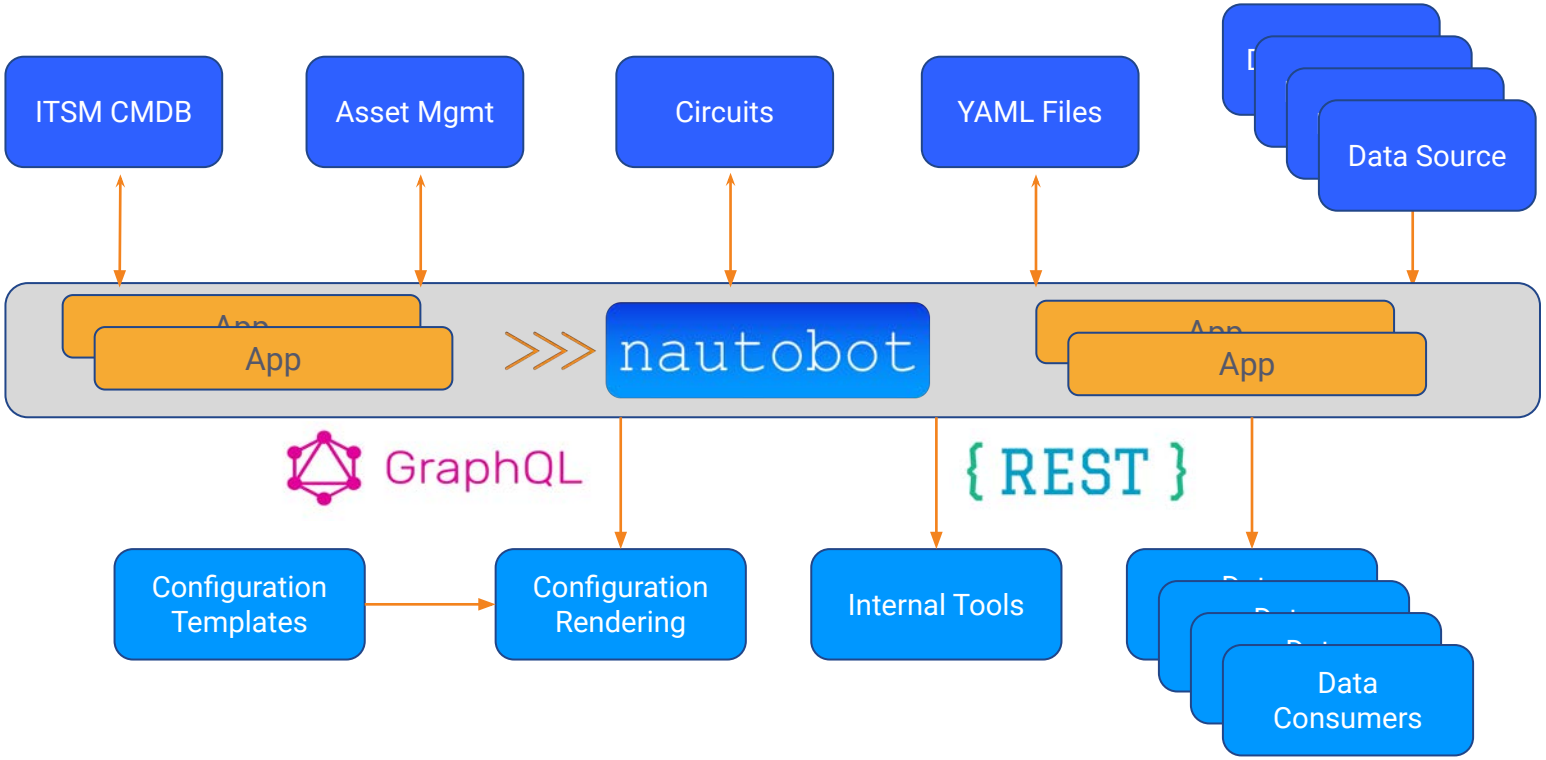
>>> Unified Source of Truth

- There will always be other tools that store network data
- Nautobot has core models and is extensible, but it is common to have CMDBs, IPAM systems, contract databases, asset management systems, etc.
- The Nautobot plugin system allows organizations to create apps and integrations to unify data creating a Single Source of Truth

Guiding Principles

- Ensure single System of Record (per data component)
- Replicate data between systems
- Use Nautobot and its APIs for a unified view into all data
  - Take advantage of GraphQL and REST APIs

Unified Source of Truth Design



>>> Nautobot Use Cases

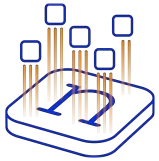
Network Source of Truth

- Devices
- IP Addresses
- VLANs
- ASN
- ...
- Custom

- User-Defined Relationships
- Custom Fields
- Data Validation
- Git as a Data Source

Network Automation Platform

- Use Open Source Apps
- Build Custom Apps
- Save 70% development time using the platform



Powered by APIs and NetDevOps extensibility & integrations

{ REST } GraphQL

webhooks git

Extensible Plugin System

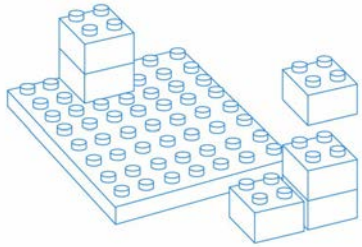
>>> Nautobot is an Application Platform

The “phone” is an enabler and delivery mechanism for high-value apps

Nautobot is an enabler and delivery mechanism for high-value apps

>>> Nautobot is an Application Platform

- Organizations can leverage Nautobot's App Platform to host automation applications!
  - Use Open Source Apps
  - Build Custom Apps
  - Save 70% development time using the platform



- Example:
  - ChatOps framework supports chatbots for multiple chat platforms

>>> Nautobot is a Network Automation Application Platform

The Nautobot App Platform provides an architecture that enables automation application development

Save substantial development time

Nautobot-Specific Use Cases

- Customize Nautobot UI
- Create APIs
- Extend data models
- Integrate external tools (like ServiceNow CMDB)
- Build automation processes

Automation Platform Use Cases

- Apps are not restricted to Nautobot-specific use cases
- Ex: the Nautobot platform can host various chatbots (Ansible AWX, CRM, CMDB, etc), perform backups, or execute any workflow

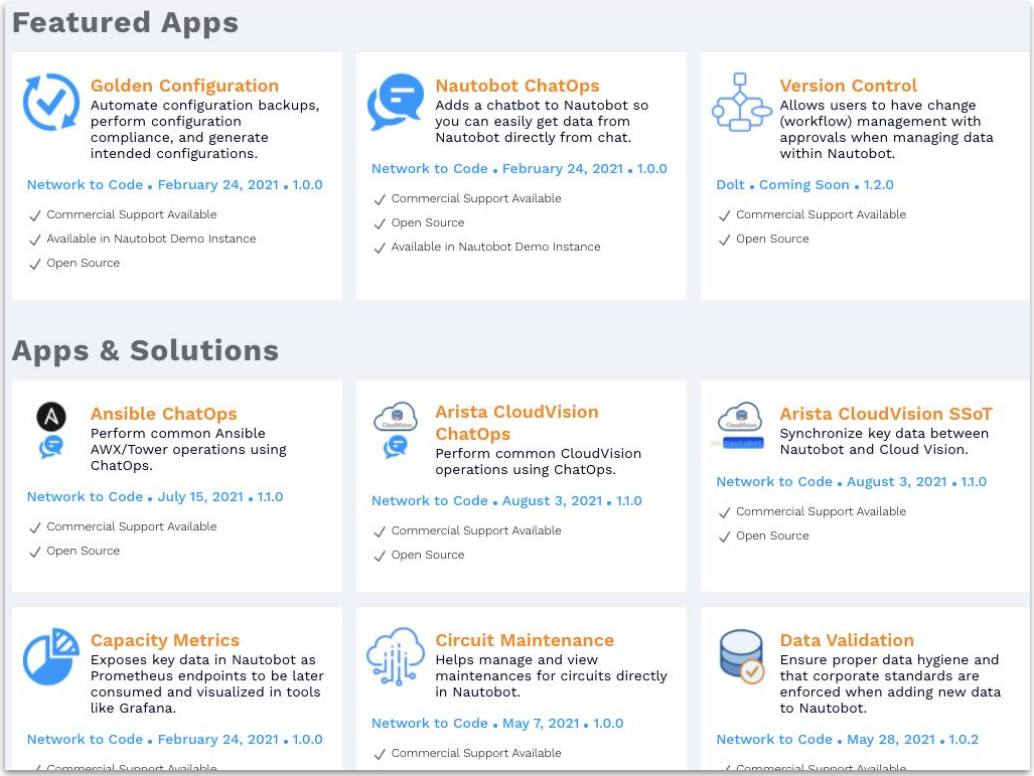
Advantages

- Flexible
- Allows for lightweight or complex apps
- Leveraging the architecture saves development time
- Leverage SoT data

>>> The Nautobot App Ecosystem

go.nautobot.com/apps

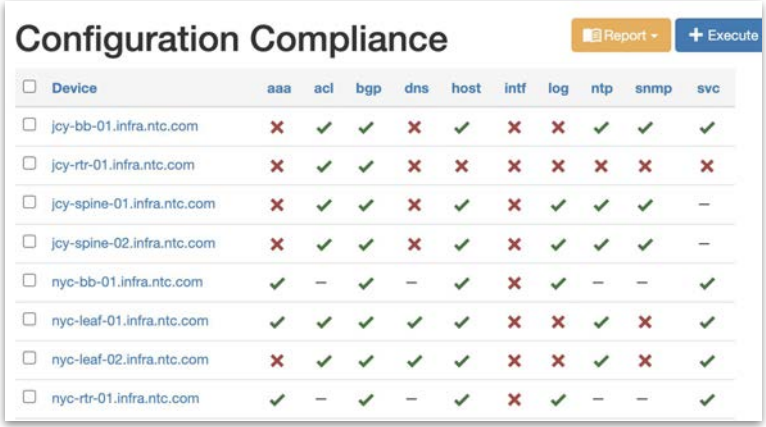
- Nautobot apps
  - Provide value as stand-alone units
  - Can leverage the rich **SoT data** already in Nautobot
- Many apps are complementary to each other, enabling more **sophisticated functionality** and workflows



>>> Nautobot App Ecosystem - Complementary Capabilities

Onboarding a device into Nautobot

The **Device Onboarding** app can quickly get basic network device info into Nautobot



Configuration Compliance

Report | + Execute

Device	aaa	acl	bgp	dns	host	intf	log	ntp	snmp	svc
jcy-bb-01.infra.ntc.com	✗	✓	✓	✗	✓	✗	✗	✓	✓	✓
jcy-rtr-01.infra.ntc.com	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗
jcy-spine-01.infra.ntc.com	✗	✓	✓	✗	✓	✗	✓	✓	✓	–
jcy-spine-02.infra.ntc.com	✗	✓	✓	✗	✓	✗	✓	✓	✓	–
nyc-bb-01.infra.ntc.com	✓	–	✓	–	✓	✗	✓	–	–	✓
nyc-leaf-01.infra.ntc.com	✓	✓	✓	✓	✓	✗	✗	✓	✗	✓
nyc-leaf-02.infra.ntc.com	✗	✓	✓	✓	✓	✗	✗	✓	✗	✓
nyc-rtr-01.infra.ntc.com	✓	–	✓	–	✓	✗	✓	–	–	✓

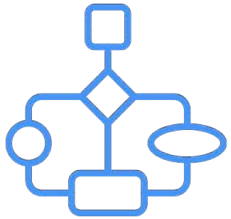
The **Golden Config** app can then:

- Automate configuration backups for the device
- Generate intended configuration for the device
- Audit the device's actual configuration against the intended config

>>> Nautobot App Ecosystem - Complementary Capabilities

*Clean** data

- The pending **Version Control** app brings git-like version control to Nautobot's database
  - Allows for GitHub-like workflows on Nautobot's data
  - These workflows help protect the data's veracity



- The **Data Validation Engine** app allows users to define rules that enforce business constraints on data formats in Nautobot

* *clean* data is correct (error-free) and properly formatted to be usable by the automation

>>> Nautobot App Ecosystem - Complementary Capabilities

- Nautobot's **Jobs** capability allows custom, arbitrary python code to run
  - Can use these jobs to further validate data formatting, content, etc.

Jobs

Module / Job	Description
Data Quality	
Verify Circuit Termination	Verify a circuit has termination and an IP address
Verify Device Rack	Verify a device is inside a rack
Verify Hostnames	Verify device hostnames match corporate standards

  - Jobs are an **Extensibility** feature, not an app, but this illustrates how Nautobot apps and Nautobot's extensibility features complement each other.

>>> Nautobot Use Cases

Network Source of Truth

- Devices
- IP Addresses
- VLANs
- ASN
- ...
- Custom

- User-Defined Relationships
- Custom Fields
- Data Validation
- Git as a Data Source

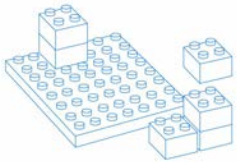
Network Automation Platform

- Use Open Source Apps
- Build Custom Apps
- Save 70% development time using the platform

Powered by APIs and NetDevOps extensibility & integrations

{ REST } GraphQL

webhooks git



Extensible Plugin System

>>> Nautobot is an Automation Platform

- Can run user-defined scripts as jobs
- Data-sharing via webhooks
- Git support
- Supports Jinja templating
- GraphQL and REST APIs

Nautobot Origins

>>> Origin Story of Nautobot

Nautobot was forked from **Netbox** in January 2021, a **Source of Truth** (SoT) to manage Data Center infrastructure, specifically for NetDevOps.

Netbox was created in 2016 to address various shortcomings of IPAM/DCIM tools available at the time:

- No/limited API
- Quantity-based commercial licensing
- Lack of support
- Missing features
  - Extensibility via Network Automation Apps (plugins)
  - Seamless integration with existing network automation solutions

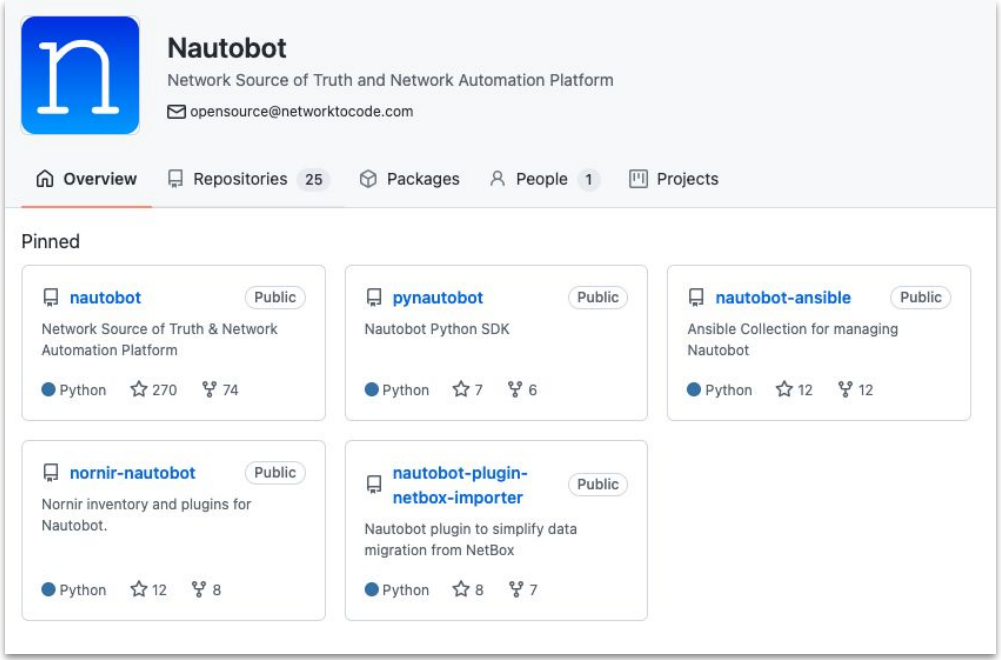
>>> Introducing Nautobot

Open Source Project - hosted on GitHub

https://github.com/nautobot/

Design Philosophy:

- **Replicate the real world**: Data model is tightly coupled to real-world constraints
- **Function as a source of truth** for the network: Use Nautobot to provision devices, not vice versa
- **Keep things simple**: High value and ease of maintenance are preferred over 100% complete solutions

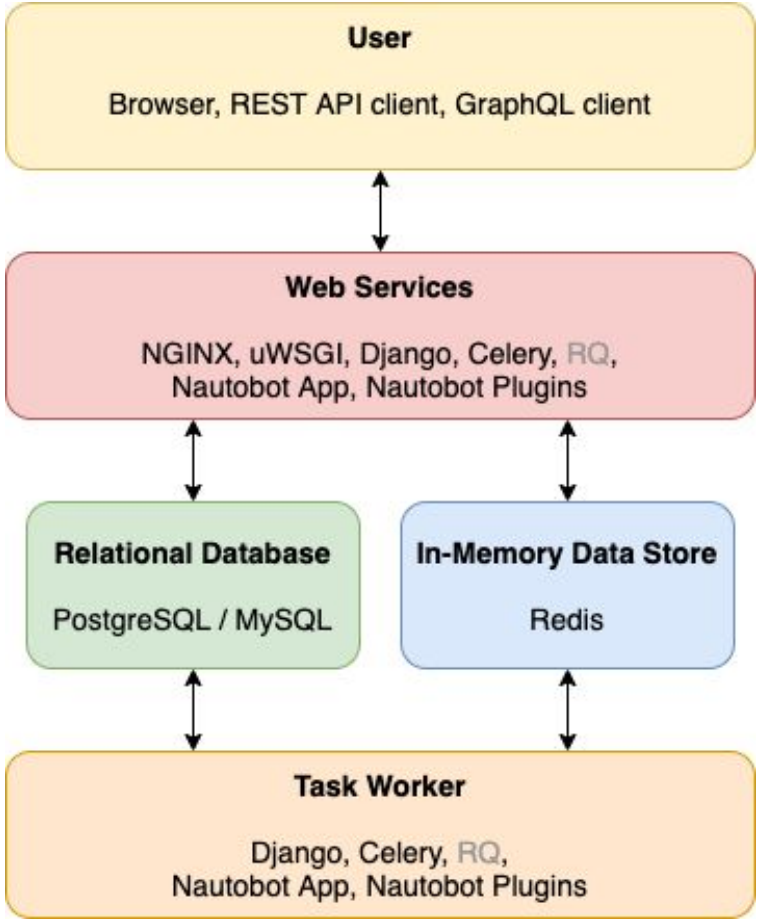


>>> Introducing Nautobot

Use Nautobot to:

- **Model** network resources: regions, buildings, racks, devices, connections, power, VMs, clusters, etc.
- **Manage** IP prefixes and address allocation
- Conveniently **integrate** with other applications
  - API driven interaction for automated workflows
- Host an **authoritative source of truth** for your infrastructure
  - instead of spreadsheets, sharepoint, emails, text files
- Build a **platform** for network automation
  - Extend (plugins), interact, discover the network

>>> The Nautobot Stack



Nautobot Web UI Overview

>>> Nautobot Home Page

https://159.203.71.114

nautobot | Organization | Devices | IPAM | Virtualization | Circuits | Power | Secrets | Extensibility | Plugins | Search | patn

The Nautobot Welcome Wizard can help you get started with Nautobot!

Search | All Objects | Search

Organization

- Sites — Geographic location — 6
- Tenants — Customers or departments — 1

DCIM

- Racks — Equipment racks, optionally organized by group — 33
- Device Types — Physical hardware models by manufacturer — 3
- Devices — Rack-mounted network equipment, servers, and other devices — 50
- Virtual Chassis — Represents a set of devices which share a common control plane — 0

Connections

- Cables — 0
- Interfaces — 1502
- Console — 50
- Power — 0

Power

- Power Feeds — Electrical circuits delivering power from panels — 0
- Power Panels — Electrical panels receiving utility power — 0

IPAM

- VRFs — Virtual routing and forwarding tables — 0
- Aggregates — Top-level IP allocations — 0
- Prefixes — IPv4 and IPv6 network assignments — 0
- IP Addresses — IPv4 and IPv6 network assignments — 156
- VLAN — Layer two domains, identified by VLAN ID — 64

Circuits

- Providers — Organizations which provide circuit connectivity — 16
- Circuits — Communication links for Internet transit, peering, and other services — 24

Virtualization

- Clusters — Clusters of physical hosts in which VMs reside — 0
- Virtual Machines — Virtual compute instances running inside clusters — 0

Data Sources

- Git Repositories — Collections of data and/or job files — 1

Job History

- Git repository - Devicetype-library — patn - 2022-02-25 22:18 — Completed
- Git repository - Devicetype-library — patn - 2022-02-25 21:54 — Completed
- Git repository - Devicetype-library — patn - 2022-02-25 21:54 — Completed
- Git repository - Devicetype-library — patn - 2022-02-25 21:53 — Completed
- Git repository - Devicetype-library — patn - 2022-02-25 21:52 — Completed
- Git repository - Devicetype-library — patn - 2022-02-25 21:35 — Completed

View All History

Change Log

- Modified Device hou01-leaf-01 — patn - 2022-03-15 17:24
- Created IP Address 10.35.192.16/32 — patn - 2022-03-14 14:10
- Created IP Address 10.35.192.12/32 — patn - 2022-03-14 14:10
- Created IP Address 10.35.192.8/32 — patn - 2022-03-14 14:10
- Created IP Address 10.35.192.7/32 — patn - 2022-03-14 14:10
- Created IP Address 10.35.192.5/32 — patn - 2022-03-14 14:10
- Created IP Address 10.35.192.4/32

nto-nautobot-patkelly-temp (v1.2.5) | 2022-03-16 01:15:24 UTC | Docs · API · GraphQL · Code · Help

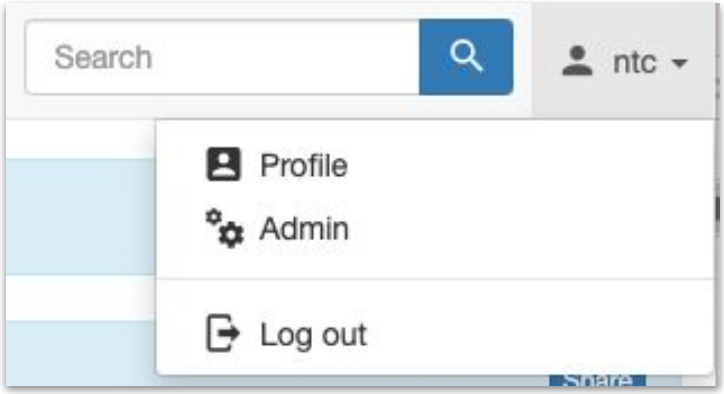
>>> Nautobot Web UI Overview

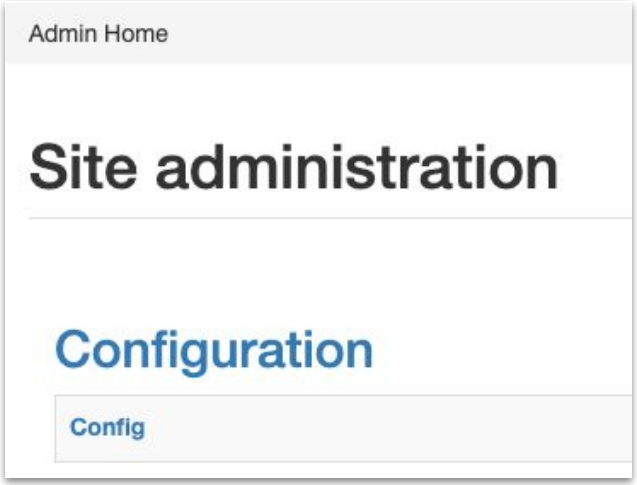
- Major Areas of All Nautobot Web Pages:
  - Nautobot Logo - top left - click here to return to the Nautobot home page
  - Menu Tabs - across the top - click to get to the relevant topic and options

- Major Areas of the Nautobot Home Page:
  - Login - top right - authentication and, if applicable, access to higher-privilege Admin menus
  - Body - center full width - loosely corresponds to some of the Menu Tab topics
  - Version - lower left - this describes the current version
  - Date and Time - lower center
  - Programming - lower right - Docs, ReST API, GraphQL, Code, and Help

>>> Admin Configuration UI (GH370)

- The Nautobot Admin UI now includes a *Configuration* page
  - Navigate to *Admin* → *Configuration* → *Config*

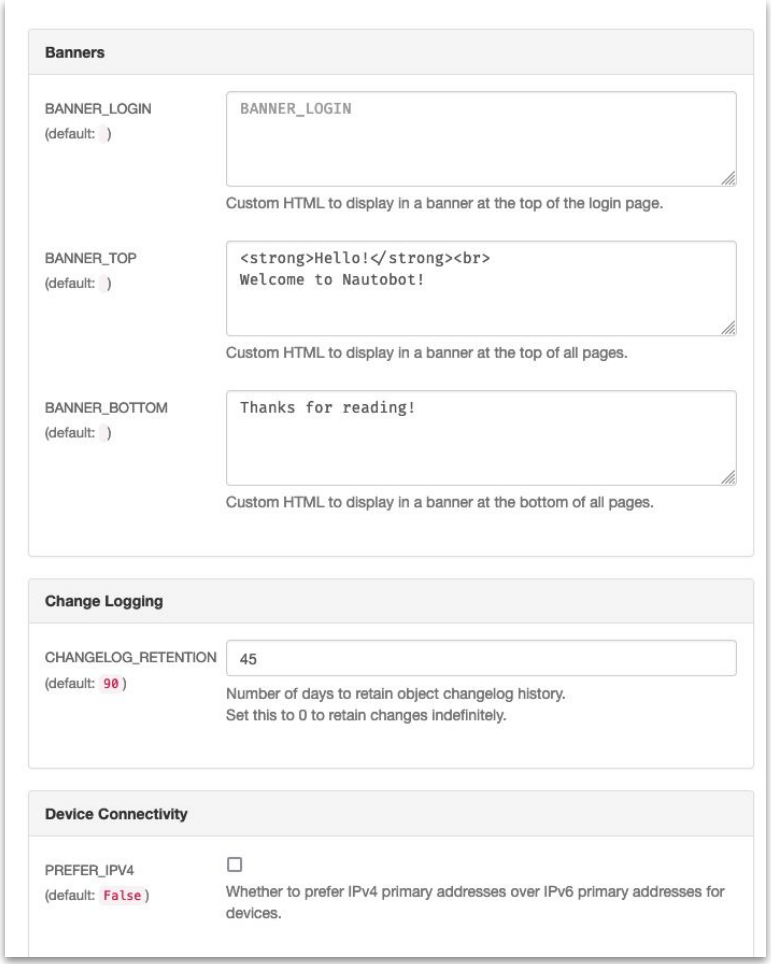




>>> Admin Configuration UI

- Admin users can dynamically customize a number of optional settings via the UI
  - BANNER_BOTTOM
  - BANNER_TOP
  - BANNER_LOGIN
  - CHANGELOG_RETENTION
  - HIDE_RESTRICTED_UI
  - MAX_PAGE_SIZE
  - PAGINATE_COUNT
  - PER_PAGE_DEFAULTS
  - PREFER_IPV4
  - RACK_ELEVATION_DEFAULT_UNIT_HEIGHT
  - RACK_ELEVATION_DEFAULT_UNIT_WIDTH
  - RELEASE_CHECK_TIMEOUT
  - RELEASE_CHECK_URL

- Settings configuration file will override these settings



>>> Admin Configuration

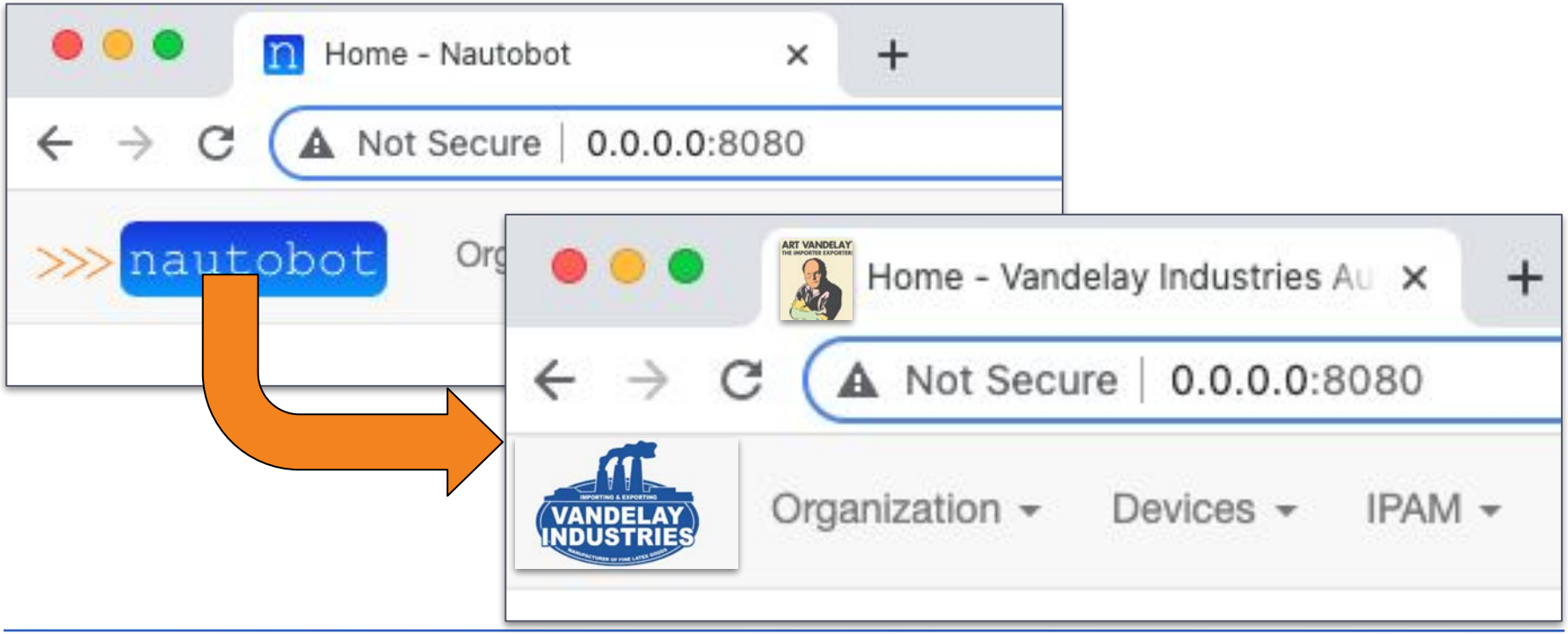
- Additional optional settings can be specified in the settings configuration file

- A few examples include:
  - BRANDING_TITLE
  - BRANDING_URLS
  - BRANDING_PREPENDED_FILENAME
  - BRANDING_FILEPATHS
    - Customize nautobot!

- link to full branding and Optional Configuration settings:
  - https://docs.nautobot.com/projects/core/en/stable/configuration/optional-settings/

>>> Custom Branding

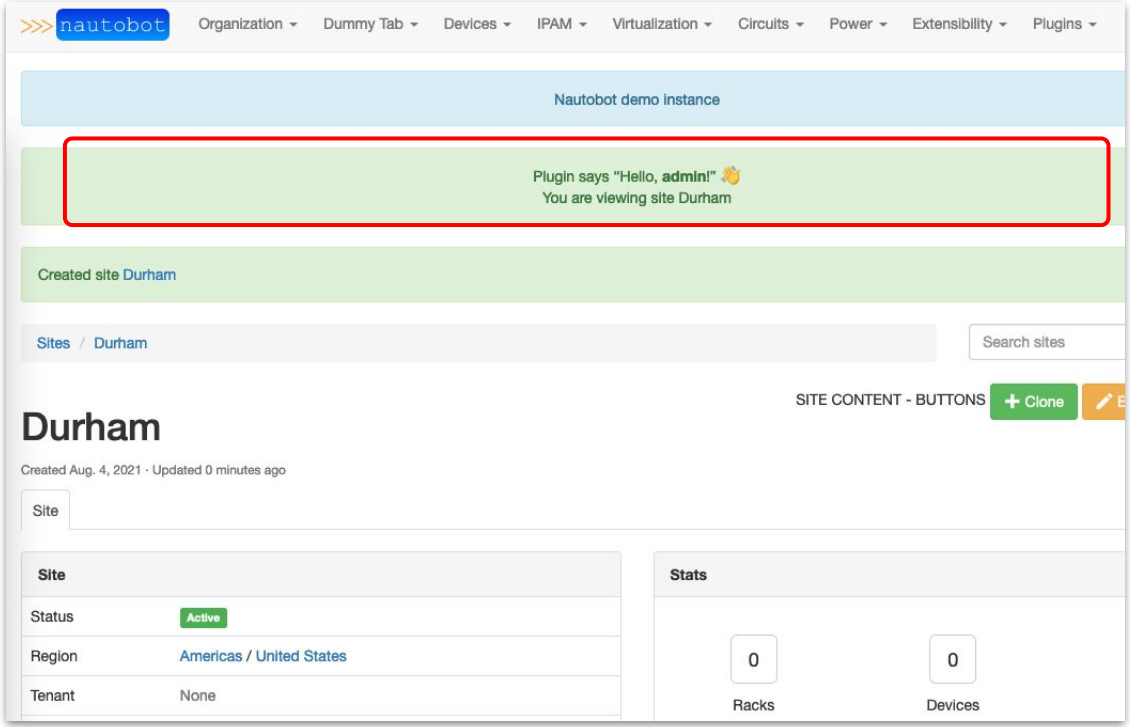
Organizations may provide custom branding assets to change the logo, icons, favicon, and footer URLs to help Nautobot fit within their environments and user communities



>>> Banners

- Apps can add banners to display specific and/or important information to the user.

- Examples include:
  - Reminders
  - Change window information
  - Maintenance Notifications
  - Last known network automation task or execution



>>> Demo 1

Introducing the Nautobot Web UI

Demo 2

Installing Nautobot Lab

Permissions

>>> Permissions

- Nautobot provides an object-based permissions framework
- Replace's Django's built-in permissions model.
- Object-based permissions enable an administrator to grant users or groups the ability to perform an action on arbitrary subsets of objects in Nautobot, rather than all objects of a certain type.
    - For example, It is possible to grant a user permission to view only sites within a particular region, or to modify only VLANs with a numeric ID within a certain range.

>>> Permissions

Object Permissions

- A permission in Nautobot represents a relationship shared by several components:
  - Object type(s) - One or more types of object in Nautobot
  - User(s)/Group(s) - One or more users or groups of users
  - Action(s) - The action(s) that can be performed on an object - View, Add, Change, or Delete
  - Constraints - An arbitrary filter used to limit the granted action(s) to a specific subset of objects

>>> Permissions

Object Permissions

- At a minimum, a permission assignment must specify at least:

One Object Type

One User or Group

One Action

- The specification of constraints is optional:
  - A permission without any constraints specified will apply to all instances of the selected model(s)

>>> Permissions

Actions

- Four core actions that can be permitted for each type of object within Nautobot Roughly analogous to the CRUD convention (create, read, update, and delete):
  - **View** - Retrieve an object from the database
  - **Add** - Create a new object
  - **Change** - Modify an existing object
  - **Delete** - Delete an existing object

- Permissions can also grant custom actions that may be required by a specific model or plugin.
  - For example, the napalm_read permission on the device model allows a user to execute NAPALM queries on a device via Nautobot's REST API.
  - These can be specified when granting a permission in the "additional actions" field.

Constraints

Expressed as a JSON object or list representing a Django query filter. This is the same syntax that you would pass to the QuerySet filter() method when performing a query using the Django ORM. As with query filters, double underscores can be used to traverse related objects or invoke lookup expressions. Some example queries and their corresponding definitions are shown below.

All attributes defined within a single JSON object are applied with a logical AND. For example, suppose you assign a permission for the site model with the following constraints: { “status”: “active”, “region_name”: “Americas” }

The permission will grant access only to sites which have a status of "active" **and** which are assigned to the "Americas" region.

To achieve a logical OR with a different set of constraints, define multiple objects within a list. For example, if you want to constrain the permission to VLANs with an ID between 100 and 199 *or* a status of "reserved," do the following:

[{ “vid__gte”: 100, “vid__lt: 200}, { “status”: “reserved” }]

Additionally, where multiple permissions have been assigned for an object type, their collective constraints will be merged using a logical "OR" operation.

>>> Example Constraint Definitions for Permissions

Constraints	Descriptions
{ "status": "active"}	Status is active
{ "status__in": ["planned", "reserved"] }	Status is planned OR reserved
{ "status": "active", "role": "testing" }	Status is active OR role is testing
{ "name__startswith": "Foo" }	Name starts with "Foo" (case-sensitive)
{ "name__iendswith": "bar" }	Name ends with "bar" (case-insensitive)
{ "vid__gte": 100, "vid__lt": 200 }	VLAN ID is greater than or equal to 100 AND less than 200
[{"vid__lt": 200}, {"status": "reserved"}]	VLAN ID is less than 200 OR status is reserved

Data Models

>>> What is a Model?

- A model is a partial representation of something real.
- Nautobot models seek to model a real network as closely as possible.
- Models have fields to describe an instance of a Model:
  - Some fields are required: Device requires a role, type, and site
  - Other fields are optional
  - Custom fields may be added
- Models can have relationships to other models
  - For instance, Device to interfaces or Interfaces to cables
  - Custom relationships may be added
- The implementation will determine how much or how little information to put into the model.

>>> Data Models

- Data models **describe** data
  - Use well-defined **parameters** to represent the data
  - Define **constraints** for the data

Data Description

- VLAN ID
- VLAN Name

Data Model

- `vlan_id`
  - Type: Integer
  - Range: 1-4096
- `vlan_name`
  - Type: String
  - Length: 2-32 characters
  - Constraints: cannot contain spaces or special characters

Data

YAML

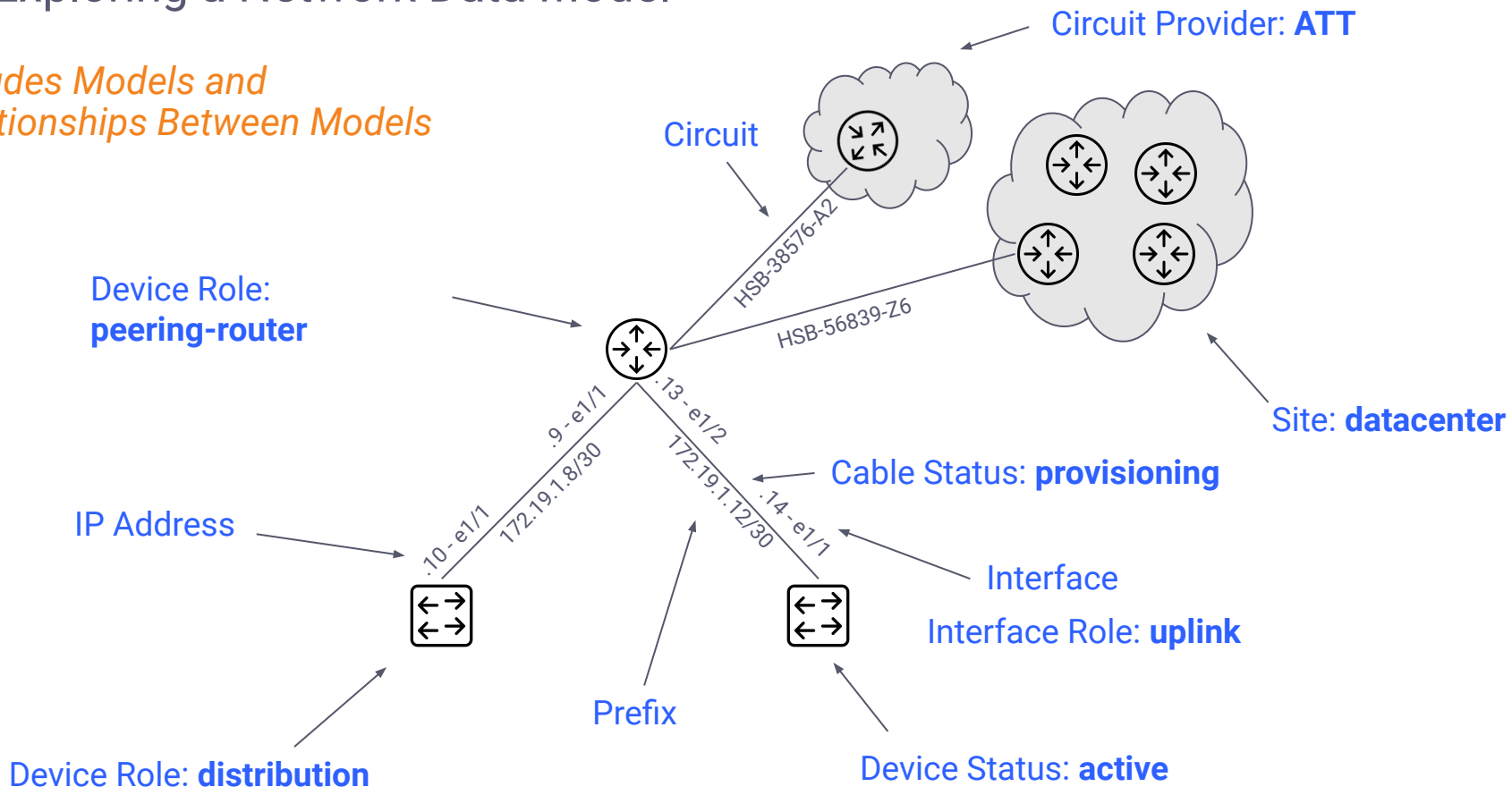
```
vlan_id: 100
vlan_name:
"web_vlan"
```

JSON

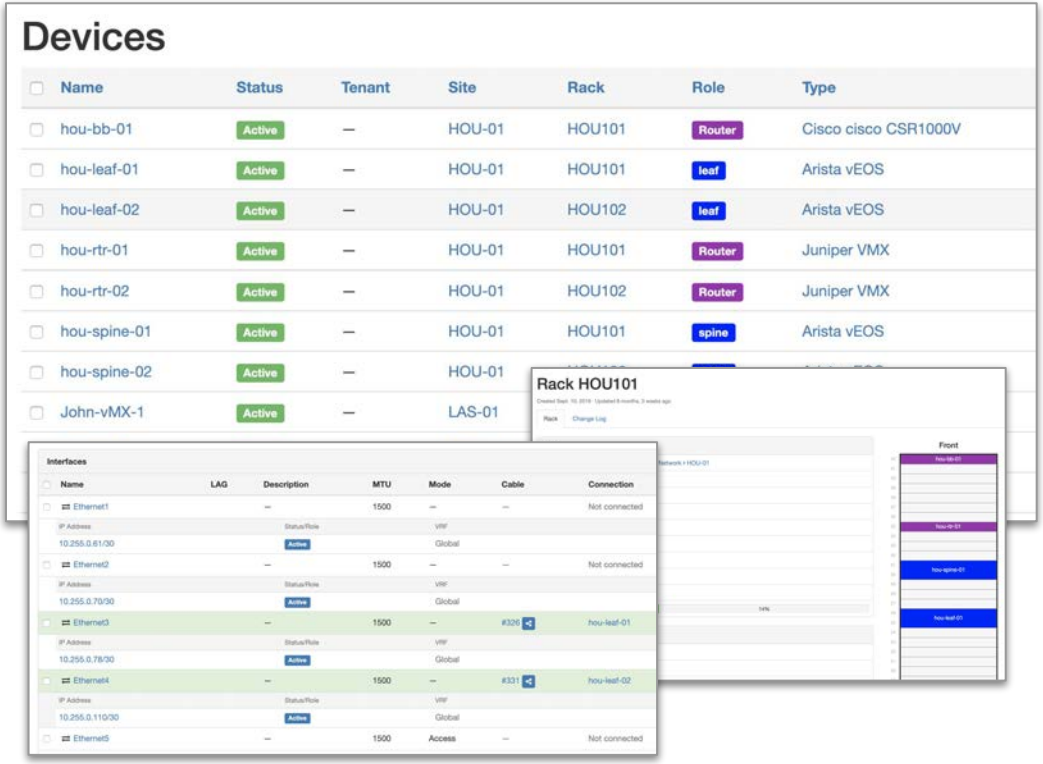
```
{
  "vlan_id": 100,
  "vlan_name": "web_vlan"
}
```

>>> Exploring a Network Data Model

Includes Models and Relationships Between Models



>>> Source of Truth Population



- Create a central SoT
- Extract the state of your existing network
- Populate the Source of Truth

>>> Recommendations

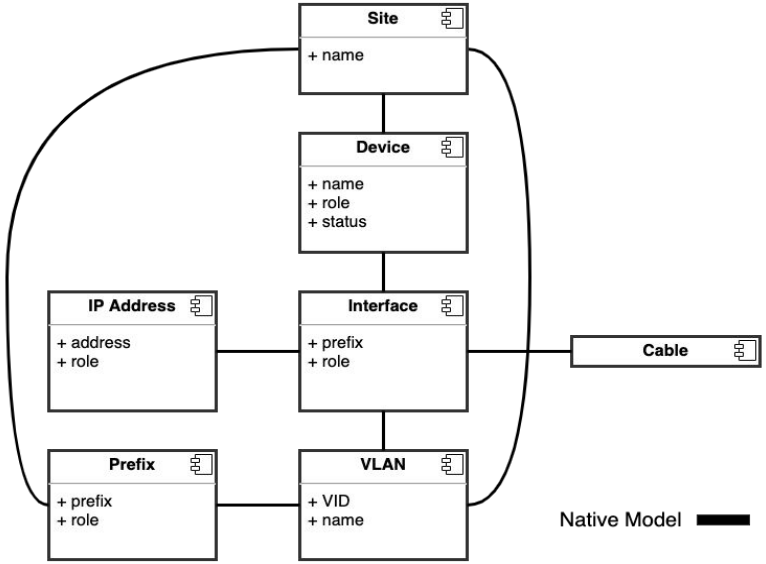
- Build a strategy for your data
  - Where will the data be stored? How will it be accessed?
  - Define the system of record (SoR) for each component
  - Consider how current and future automation will use the data
  - Identify gaps
  - Document it!
- Start small
  - Focus on the system of record(s) that provides the most value for your immediate automation requirements
  - Identify a device role that has good config standardization and large impact
  - Model global device properties before attempting more complicated configurations/designs
  - Avoid trying to describe the entire design in data (i.e. if the config is standard for the design, it may belong in the template)
- Iterate
  - Accept that the first attempt at modeling won't cover all use cases
  - Improve on the existing data model as needed

Nautobot Core Data Models

>>> The Nautobot Data Model

- Nautobot defines a network data model that represents core network assets and resources.
  - Serves as the Source of Truth (SoT) for Network Automation efforts
  - Model intends to reflect the real world

- Nautobot is extensible and flexible to meet the unique requirements and use cases for all organizations.
  - Nautobot offers flexibility in the data model
  - Several targeted features and plugin ecosystem



>>> Nautobot Core Data Models

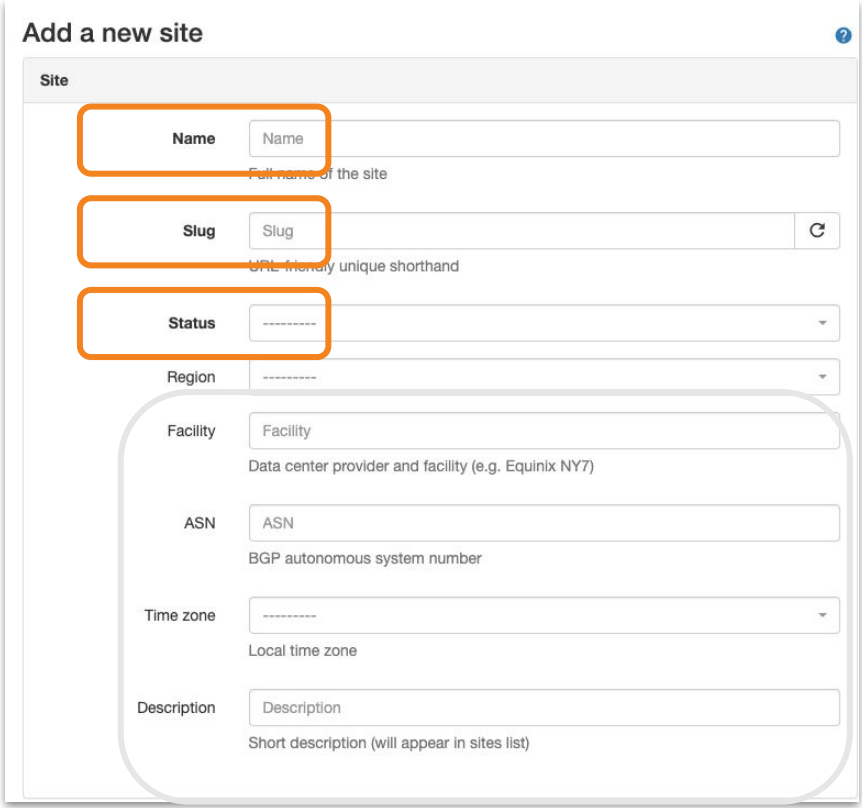
- circuits > provider
- circuits > circuittype
- circuits > circuit
- circuits > circuittermination
- dcim > consoleport
- dcim > consoleserverport
- dcim > powerport
- dcim > poweroutlet
- dcim > interface
- dcim > frontport
- dcim > rearport
- dcim > devicebay
- dcim > inventoryitem
- dcim > manufacturer
- dcim > devicetype
- dcim > devicerole
- dcim > platform
- dcim > device
- dcim > virtualchassis
- dcim > cable
- dcim > consoleporttemplate
- dcim > consoleserverporttemplate
- dcim > powerporttemplate
- dcim > poweroutlettemplate
- dcim > interfacetemplate
- dcim > frontporttemplate
- dcim > rearporttemplate
- dcim > devicebaytemplate
- dcim > powerpanel
- dcim > powerfeed
- dcim > rackgroup
- dcim > rackrole
- dcim > rack
- dcim > rackreservation
- dcim > region
- dcim > site
- ipam > vrf
- ipam > routetarget
- ipam > rir
- ipam > aggregate
- ipam > role
- ipam > prefix
- ipam > ipaddress
- ipam > vlangroup
- ipam > vlan
- ipam > service
- extras > tag
- tenancy > tenantgroup
- tenancy > tenant
- virtualization > clustertype
- virtualization > clustergroup
- virtualization > cluster
- virtualization > virtualmachine
- virtualization > vminterface

>>> The Nautobot Data Model

Nautobot contains “Django apps” (e.g. DCIM, IPAM) which house the various models (e.g. devices, virtual-machines, racks).

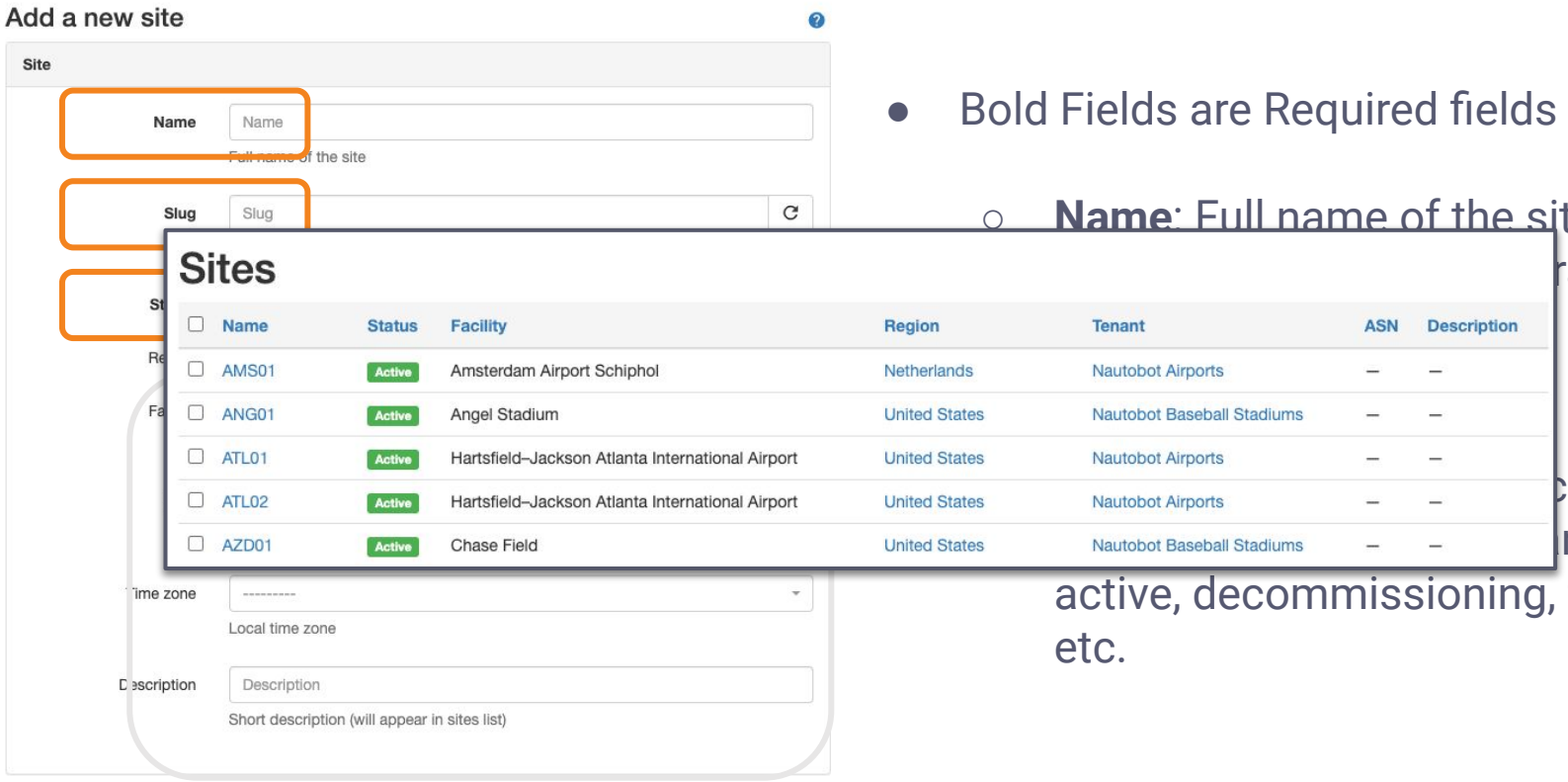
- **DCIM** - Data Center Infrastructure Management
  - Organizational and Device specific
  - Regions, Sites, Rack Groups, Racks,
  - Devices, Cables, Interfaces
- **Circuits** - Service Provider communication circuits, not electrical
- **Power** - Power feeds, power cables, and usage
- **IPAM** - IP Address Management
  - VRFs, Prefixes, Addresses, AS numbers
    - Full feature parity for IPv4 and IPv6
  - VLANs and VLAN Groups
- **Virtualization** - Hypervisor clusters, VMs
- **Tenancy, Secrets, Apps / Plugins**

>>> Model Input Patterns



- Bold Fields are Required fields
  - **Name**: Full name of the site, may have spaces or other characters
  - **Slug**: API friendly field
  - **Status**: reflects the lifecycle stage or state of the object - planned, active, decommissioning, failed, etc.

>>> Model Input Patterns



- Bold Fields are Required fields
  - **Name**: Full name of the site, may racters
  - cle stage nned, active, decommissioning, failed, etc.

>>> Regions, Sites, Rack Groups, and Racks

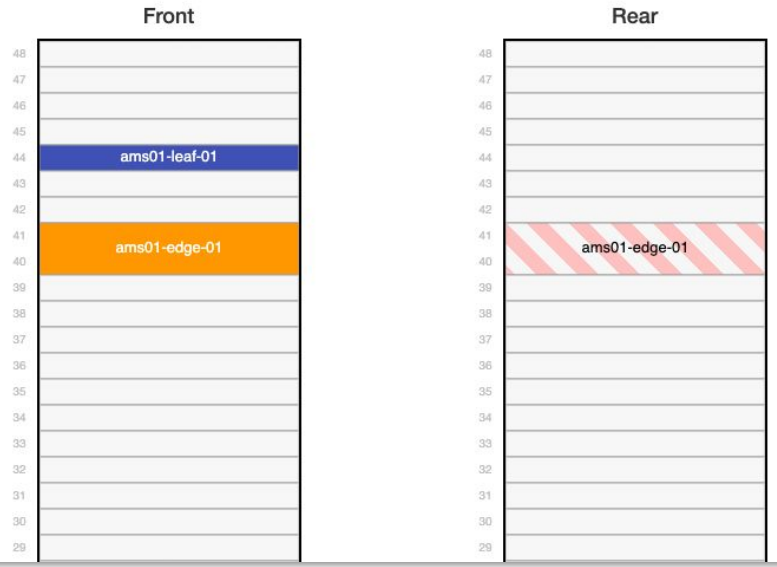
Previous Rack | Next Rack | + Clone | Edit | Delete

Rack ams01-101

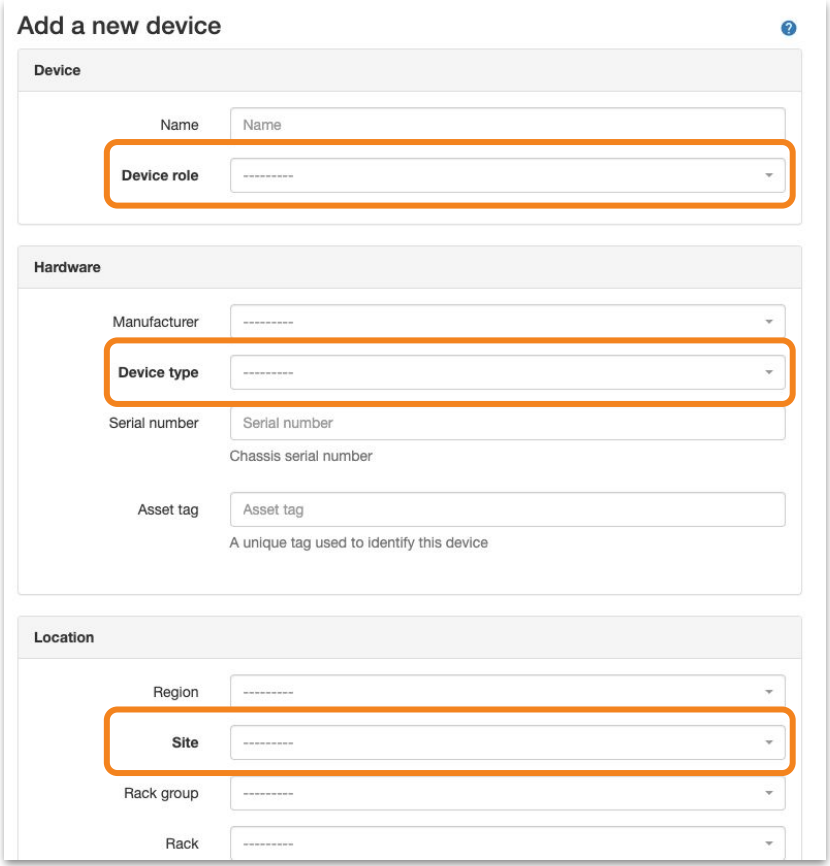
Created Sept. 7, 2021 · Updated 1 week, 2 days ago

Rack | Change Log | Show Images

Rack	
Site	Netherlands / AMS01
Group	None
Facility ID	—
Tenant	Nautobot Airports
Status	Active
Role	None
Serial Number	—
Asset Tag	—
Devices	2
Space Utilization	6%
Power Utilization	0%

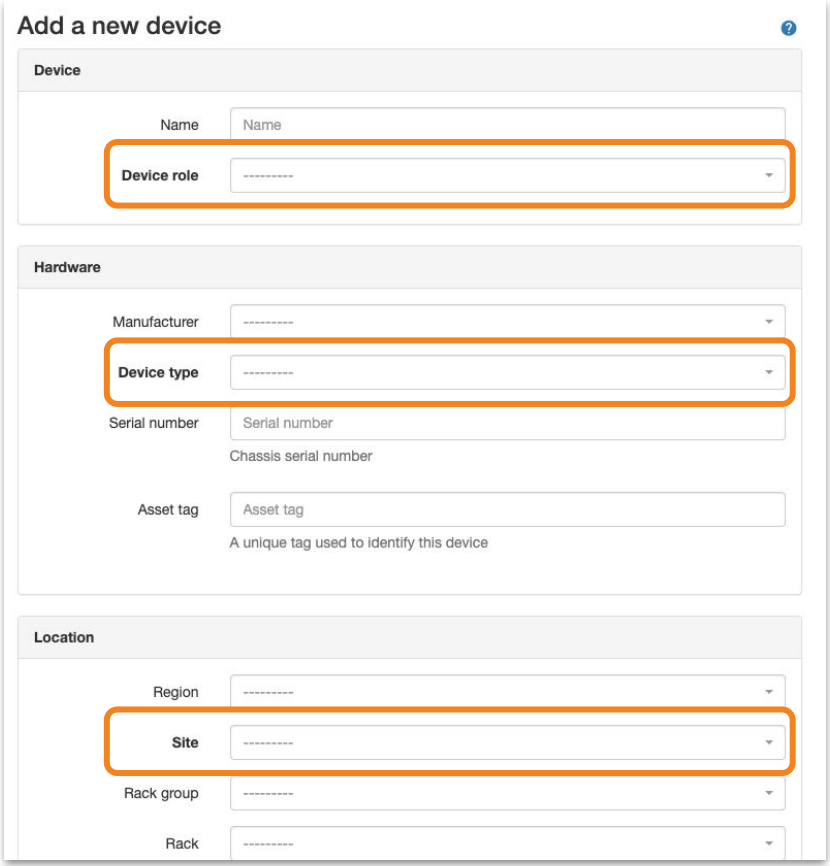


>>> Devices and Interfaces

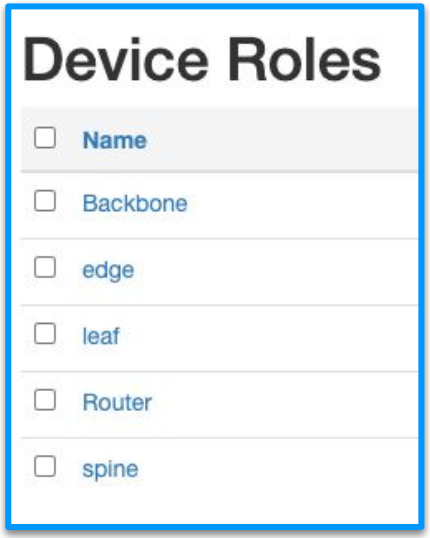


- Certain objects are required in order to create other objects relationships:
- **Site**, **Device Role,** and **Device Type** for Devices

>>> Devices and Interfaces



- Certain objects are required in order to create other objects relationships:
- **Site**, **Device Role,** and **Device Type** for Devices



>>> Devices and Interfaces

Add a new device

Device

Name: Name

Device role: ---------

Hardware

Manufacturer: ---------

Device type: ---------

Serial number: Serial number

Chassis serial number

Asset tag: Asset tag

A unique tag used to identify this device

Location

Region: ---------

Site: ---------

Rack group: ---------

Rack: ---------

Add a new device type

Device Type

Manufacturer: ---------

Model: Model

Slug: Slug

URL-friendly unique shorthand

Part number: Part number

Discrete part number (optional)

Height (U): 1

☑ Is full depth

Device consumes both front and rear rack faces

Parent/child status: ---------

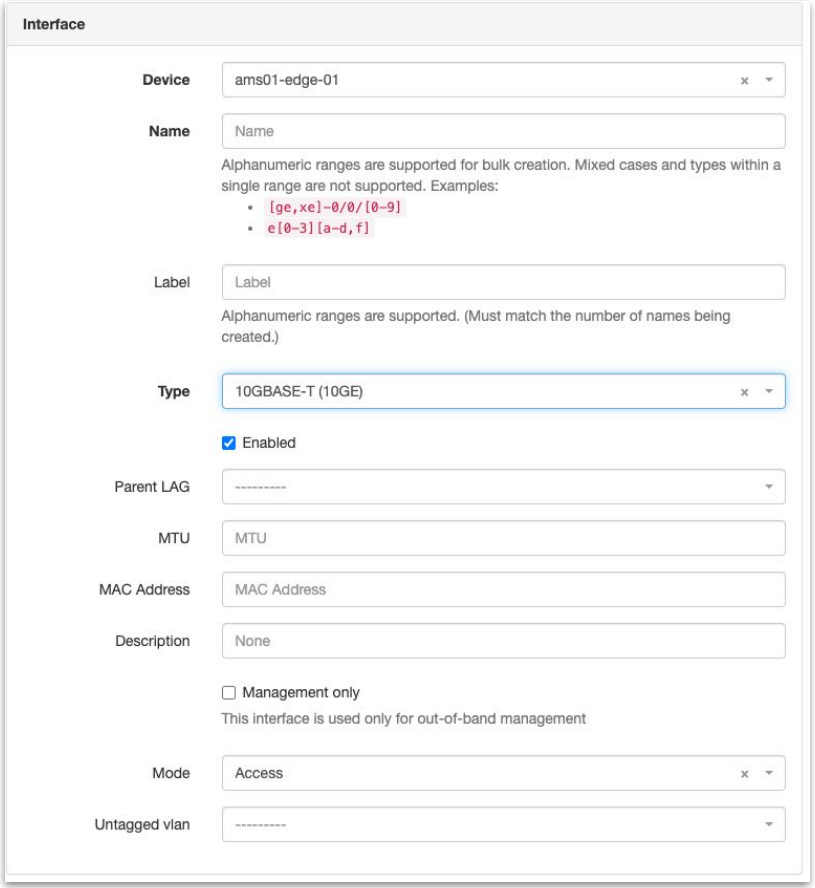
Parent devices house child devices in device bays. Leave blank if this device type is neither a parent nor a child.

Rack Images

Front image: Choose File No file chosen

Rear image: Choose File No file chosen

>>> Devices and Interfaces



- Interfaces can be added to **Devices**
- Interface Templates can be added to **Device Types**
  - Interface Templates define what interfaces are added to all child instances of the parent Device Type
- **Device**, **Interface Name**, **Type**, and **Status** are required fields for Interfaces

>>> Prefixes, IP addresses, and VLANs

Prefixes

Prefix	Status	Children	VRF	Utilization	Tenant
10.0.0.0/8	Container	1309	Global	13%	Nautobot Baseball Stadiums
10.0.0.0/8	Container	1309	Global	13%	Nautobot Airports
• • 10.0.0.0/16	Container	38	Global	100%	Nautobot Airports
• • • 10.0.0.0/18	Container	8	Global	12%	Nautobot Airports
• • • • 10.0.0.0/24	Active	0	Global	0%	Nautobot Airports
• • • • 10.0.1.0/24	Active	0	Global	0%	Nautobot Airports
• • • • 10.0.2.0/24	Active	0	Global	0%	Nautobot Airports

- Prefixes can be containers or subnets
- Organization of Prefixes and addresses is automatic
- IPs are assigned to interfaces which belong to devices

>>> Prefixes, IP addresses, and VLANs

VLANs

☐	ID	Site	Group	Name
☐	99	AMS01	—	ams01-108-mgmt
☐	99	AMS01	—	ams01-103-mgmt
☐	99	AMS01	—	ams01-106-mgmt
☐	99	AMS01	—	ams01-107-mgmt
☐	99	AMS01	—	ams01-101-mgmt
☐	99	AMS01	—	ams01-104-mgmt

- VLANs can be organized into VLAN Groups
- VLANs or VLAN groups can be assigned to sites

>>> Circuits

Circuits

☐	ID	Provider	Type	Status	Tenant	A Side	Z Side	Description
☐	ntt-104265404093023273	NTT	Transit	Active	Nautobot Airports	SIN01	—	—
☐	ntt-104265404093069929	NTT	Transit	Active	Nautobot Airports	SIN01	—	—
☐	ntt-104539051754046505	NTT	Transit	Active	Nautobot Baseball Stadiums	SLC01	—	—

- **Circuits** are objects representing some physical or virtual connectivity between two endpoints
- **Circuits** require a **Circuit Type** and a **Circuit Provider**

>>> Power

Power Feed

Rack: ---------

Name: Name

Status: ---------

Characteristics

Type: Primary

Supply: AC

Voltage: 120

Amperage: 20

Phase: Single phase

Max utilization: 80

Maximum permissible draw (percentage)

- Power includes **Power Feeds** and **Power Panels**
- They are used to keep track of power utilization

Welcome Wizard App

>>> Welcome Wizard Overview

- The Welcome Wizard is an open-source Nautobot Plugin with the goal to assist users with the necessary initial steps in populating data within Nautobot.
- The Welcome Wizard adds four (4) key features:
  - **Import Wizard**
    - Welcome Wizard uses the Import Wizard to allow ease of adding community defined Device Types and Manufacturers into Nautobot. This is built upon the git datasources feature of Nautobot.
  - **Quick-Start Settings**
    - Welcome Wizard includes by default the DeviceType-library, but this can be disabled and a custom library can be used instead.
  - **Helpful Middleware**
    - Welcome Wizard includes banners in forms to alert the user when required form fields have no associated resources in Nautobot.
  - **Welcome Wizard Dashboard**
    - The Welcome Wizard Dashboard contains a list of common Nautobot Data Models that many other Nautobot models require. This page allows ease of adding items to Nautobot or, if supported, importing them. This ties all of the features together.

>>> Welcome Wizard

https://159.203.71.114/plugins/welcome_wizard/dashboard/

>>> nautobot | Organization | Devices | IPAM | Virtualization | Circuits | Power | Secrets | Extensibility | Plugins

Dashboard

Welcome Wizard

Name	Completed	Ignored	Actions
Sites	☑	✗	+
Manufacturers	☑	✗	+
Device Types	☑	✗	+
Device Roles	☑	✗	+
Circuit Types	☑	✗	+
Circuit Providers	☑	✗	+
RIRs	☐	✗	+
VM Cluster Types	☐	✗	+

1000 per page

Showing 1-8 of 8

Demo 3

Taking the First Steps with Nautobot

>>> Lab 1

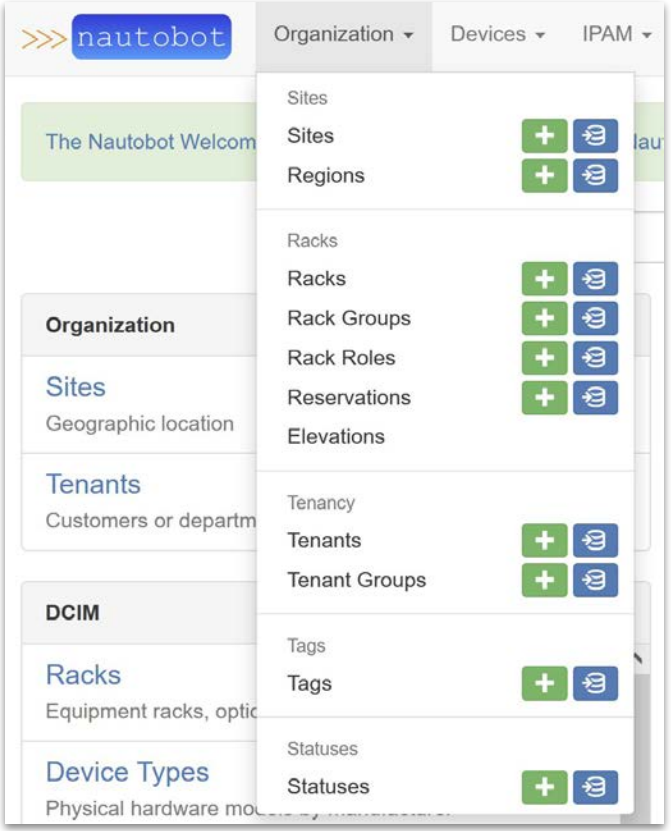
Taking the First Steps with Nautobot

Course Agenda

1. Why Network Automation?
2. Nautobot Overview and Installation
3. Nautobot Organization and Devices
4. Nautobot IPAM to Circuits
5. Nautobot Apps
6. Nautobot Programming
7. Nautobot Extensibility

Nautobot Organizations

>>> Nautobot Organizations



- Click on the Organization menu tab from almost any Nautobot web page
- Major subsections:
  - Sites
  - Racks
  - Tenancy
  - Tags
  - Statuses

>>> Organization Term Definitions

Tenants

- A tenant represents a discrete grouping of resources used for administrative purposes. Typically, tenants are used to represent individual customers or internal departments within an organization.
- The following objects can be assigned to tenants:
  - Sites
  - Racks
  - Rack reservations
  - Devices
  - VRFs
  - Prefixes
  - IP addresses
  - VLANs
  - Circuits
  - Clusters
  - Virtual machines

>>> Organization Term Definitions

Tenants

- Tenant assignment is used to signify the ownership of an object in Nautobot. As such,
  - Each object may only be owned by a single tenant.
  - Example:
    - A firewall dedicated to a particular customer, would be assigned to the tenant which represents that customer.
    - If the firewall serves multiple customers, it doesn't *belong* to any particular customer, and tenant assignment would not be appropriate.

>>> Organization Term Definitions

Sites

- How you choose to employ sites when modeling your network may vary depending on the nature of your organization,
  - Generally a site will equate to a building or campus.
  - Example
    - A chain of banks might create a site to represent each of its branches, a site for its corporate headquarters, and two additional sites for its presence in two colocation facilities.

Statuses

- Each site must be assigned a unique name and operational status and may optionally be assigned to a region and/or tenant.
- The following operational statuses are available by default:
  - Planned
  - Staging
  - Active
  - Decommissioning
  - Retired

>>> Organization Term Definitions

Regions

- Sites can be arranged geographically using regions.
  - A region might represent a continent, country, city, campus, or other area depending on your use case.
  - Regions can be nested recursively to construct a hierarchy.
  - Example
    - You might define several country regions, and within each of those several state or city regions to which sites are assigned.

>>> Organization Term Definitions

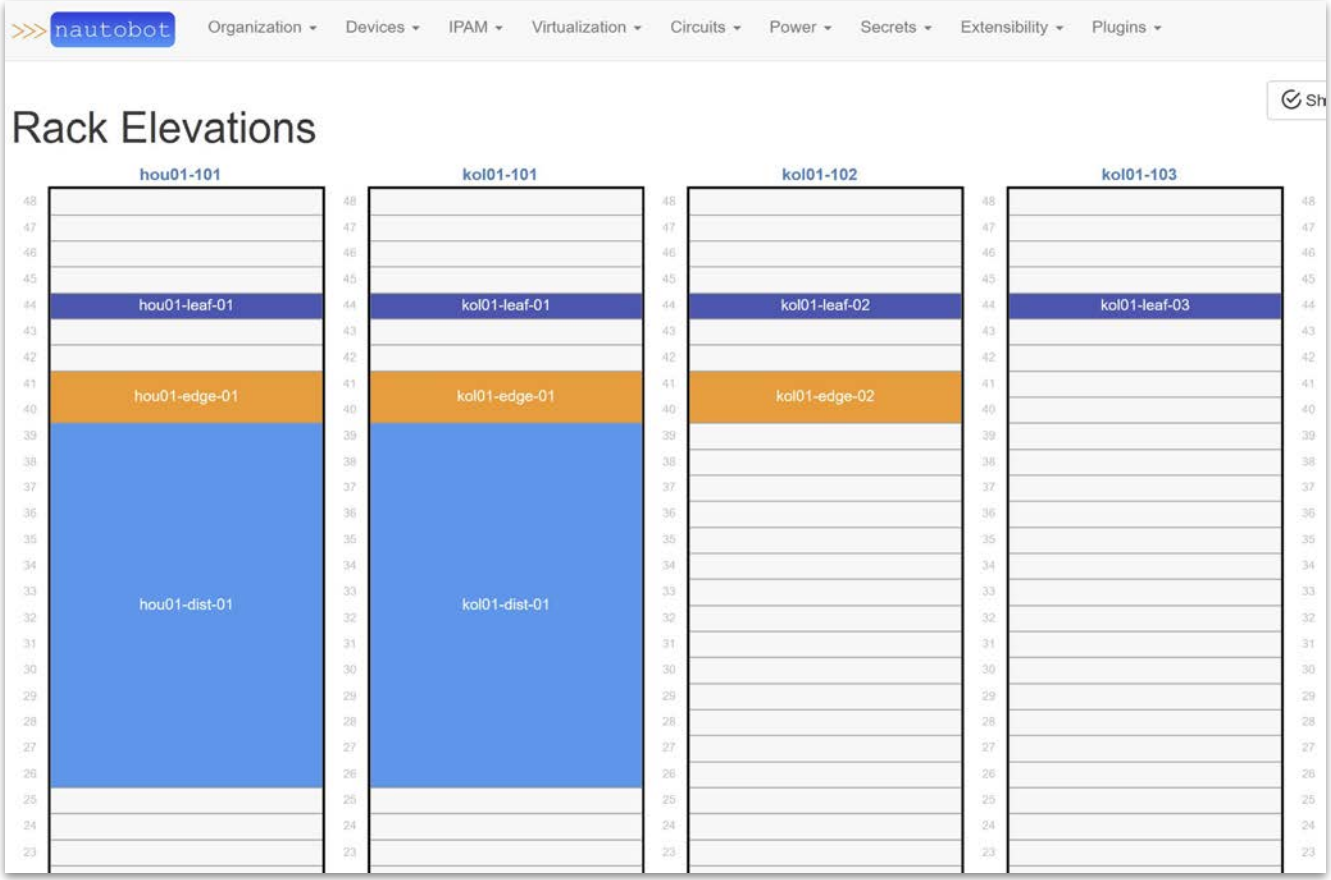
Racks

- The rack model represents a physical equipment rack in which devices can be installed.
- Must be assigned to a site
- May optionally be assigned to a rack group and/or tenant.
- Can be organized by user-defined functional roles.

Rack Groups

- How you choose to designate rack groups will depend on the nature of your organization.
- Racks can be organized into groups, which can be nested into themselves similar to regions.
- Each rack group must be assigned to a parent site
- May optionally be nested within a site to model a multi-level hierarchy
  - Examples
    - If each site represents a campus, each group might represent a building within a campus.
    - If each site represents a building, each rack group might equate to a floor or room.

>>> Rack Elevations



>>> Tags

Tags can be applied to nearly any object to further tailor organization and network design parameters.

AP00c8.8b6a.04c2

Created July 8, 2021 · Updated 1 month ago

Device | Interfaces 2 | Status | LLDP Neighbors | Configuration | Config Context | Change Log

Device

Site	United States / Round lake
Rack	None
Position	—
Tenant	None
Device Type	Cisco AIR-AP2802I-B-K9 (1U)
Serial Number	FDW2046D1RA
Asset Tag	—

Tags

Name	Items	Slug	Color	Description
onboarding-class=APOnboarding	4849	onboarding-classaponboarding		—
onboarding-class=AsaHaOnboarding	51	onboarding-classasahaonboarding		—
onboarding-class=FortiOSHAOnboarding	4	onboarding-classfortioshaonboarding		—
onboarding-class=NxosFexOnboarding	185	onboarding-classnxosfexonboarding		—
onboarding-class=StackOnboarding	3033	onboarding-classstackonboarding		—
onboarding-class=StandaloneOnboarding	1179	onboarding-classstandaloneonboarding		—
onboarding-class=VssOnboarding	94	onboarding-classvssonboarding		—
onboarding-class=WlcHaOnboarding	63	onboarding-classwlchaonboarding		—
onboarding-class=WLCHaOnboarding	45	onboarding-classwlchaonboarding_1		—
onboarding-class=WLCStandaloneOnboarding	179	onboarding-classwlcstandaloneonboarding		—
ucse_onboarding	8	ucse_onboarding		—

Edit Selected | Delete Selected

Tags

ap-name=AP00C8.8B6A.04C2 | onboarding-class=APOnboarding | wlc-name=zALUSrlcTest

>>> Demo 4

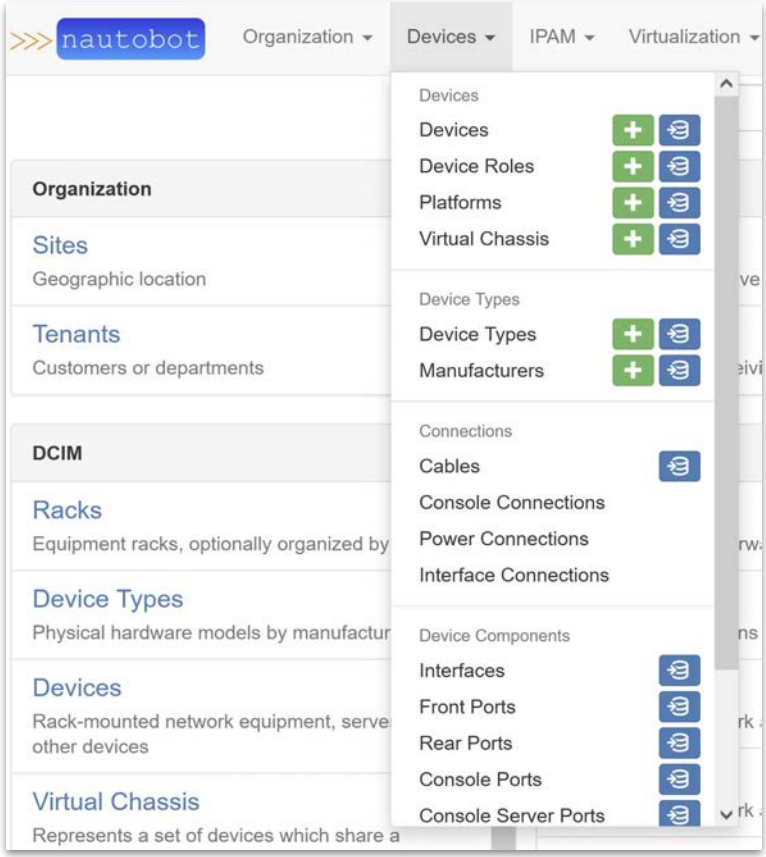
Learning the Nautobot Organization Model

>>> Lab 2

Learning the Nautobot Organization Model

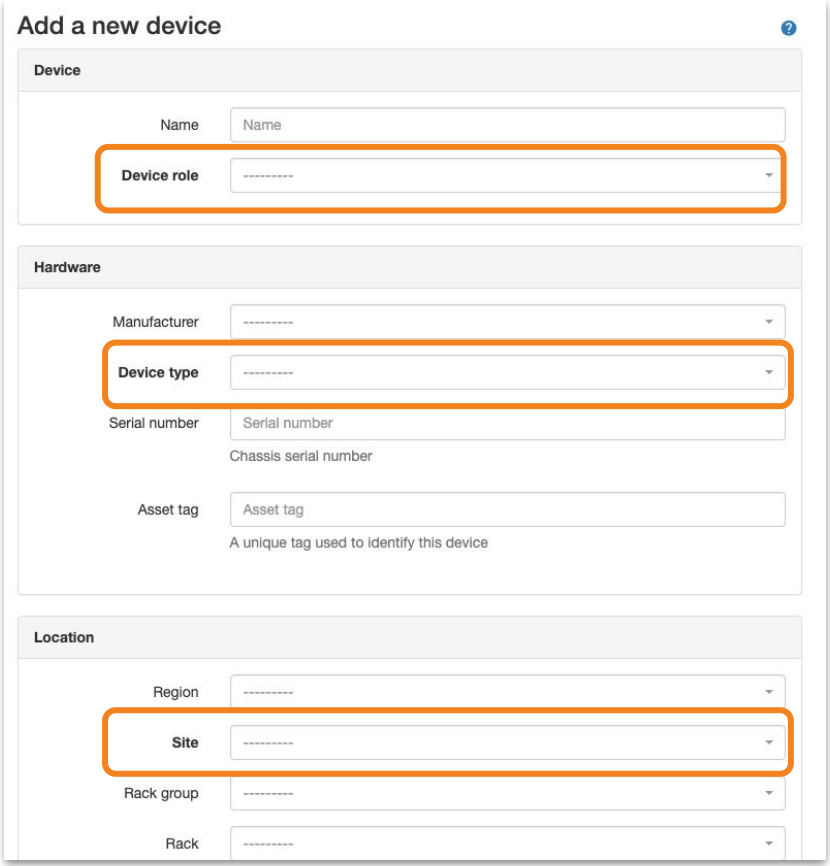
Nautobot Devices

>>> Nautobot Devices



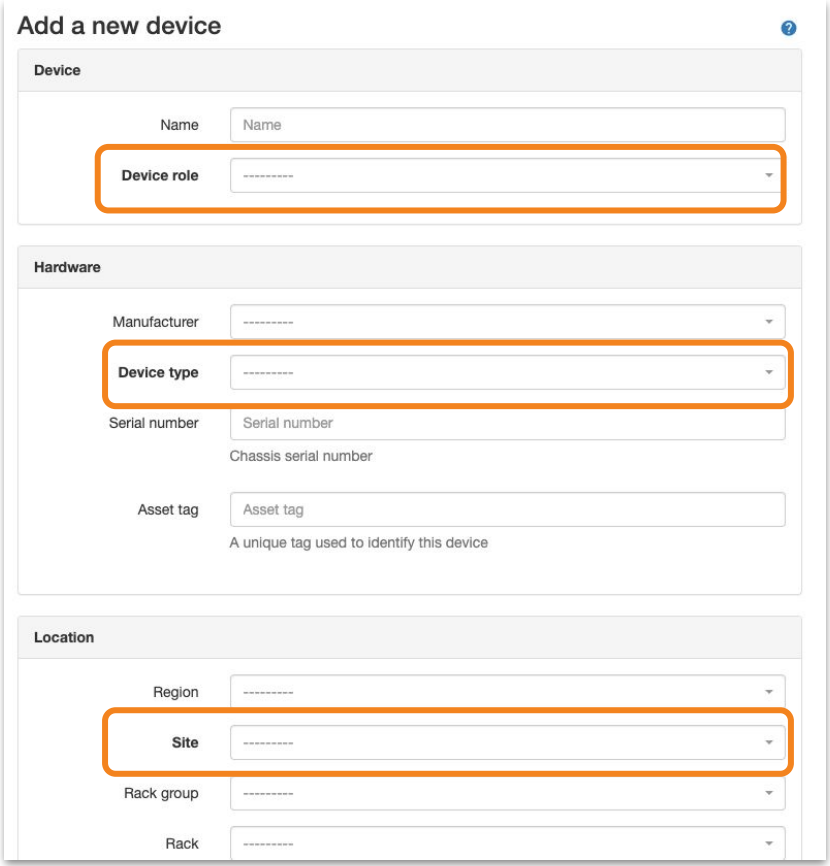
- Click on the Devices menu tab from almost any Nautobot web page
- Major subsections:
  - Devices
  - Device Types
  - Connections
  - Device Components

>>> Devices and Interfaces

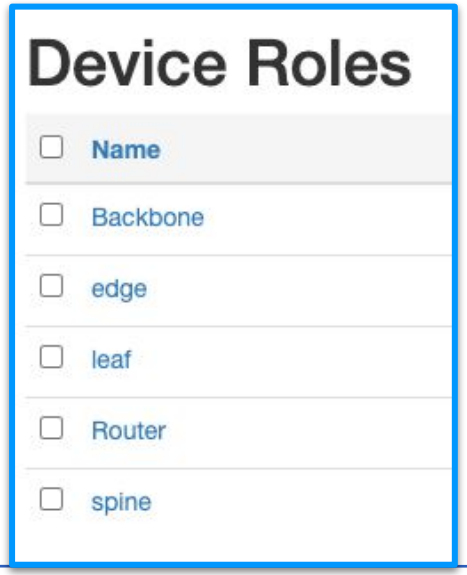


- Certain objects are required in order to create other objects relationships:
  - **Site**, **Device Role,** and **Device Type** for Devices

>>> Devices and Interfaces



- Certain objects are required in order to create other objects relationships:
  - **Site**, **Device Role,** and **Device Type** for Devices



>>> Devices and Interfaces

Add a new device

Device

Name: Name

Device role: ---------

Hardware

Manufacturer: ---------

Device type: ---------

Serial number: Serial number

Chassis serial number

Asset tag: Asset tag

A unique tag used to identify this device

Location

Region: ---------

Site: ---------

Rack group: ---------

Rack: ---------

Add a new device type

Device Type

Manufacturer: ---------

Model: Model

Slug: Slug

URL-friendly unique shorthand

Part number: Part number

Discrete part number (optional)

Height (U): 1

☑ Is full depth

Device consumes both front and rear rack faces

Parent/child status: ---------

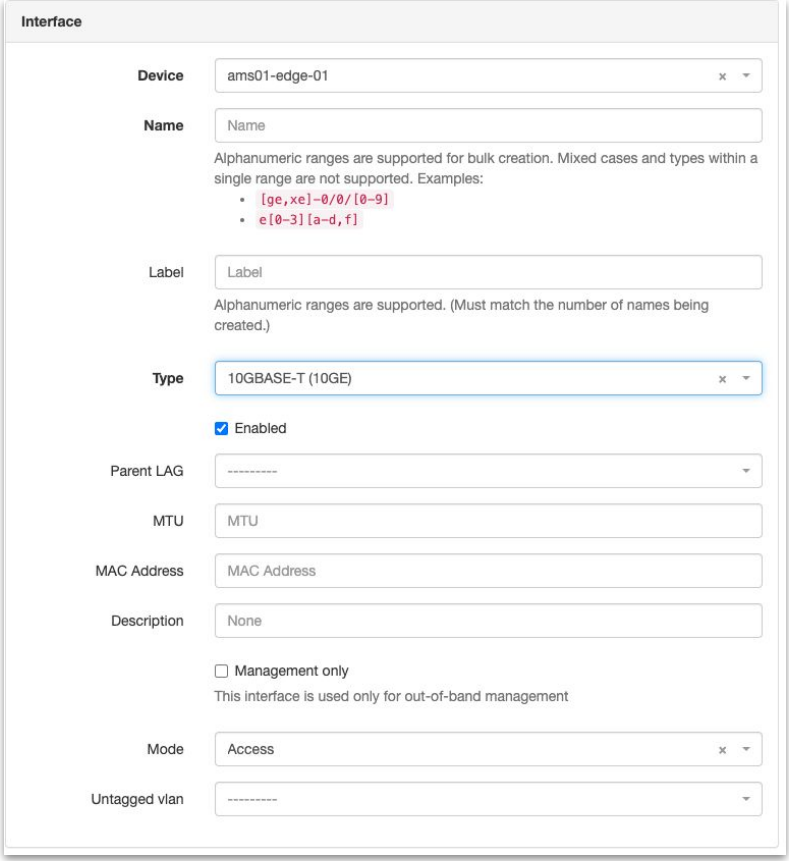
Parent devices house child devices in device bays. Leave blank if this device type is neither a parent nor a child.

Rack Images

Front image: Choose File No file chosen

Rear image: Choose File No file chosen

>>> Devices and Interfaces



- Interfaces can be created under **Devices** and **Device Types** (although they are different types of objects)
- **Device**, **Interface Name**, **Type**, and **Status** are required fields for Interfaces

Demo 5

Investigating Nautobot Devices

>>> Lab 3

Investigating Nautobot Devices