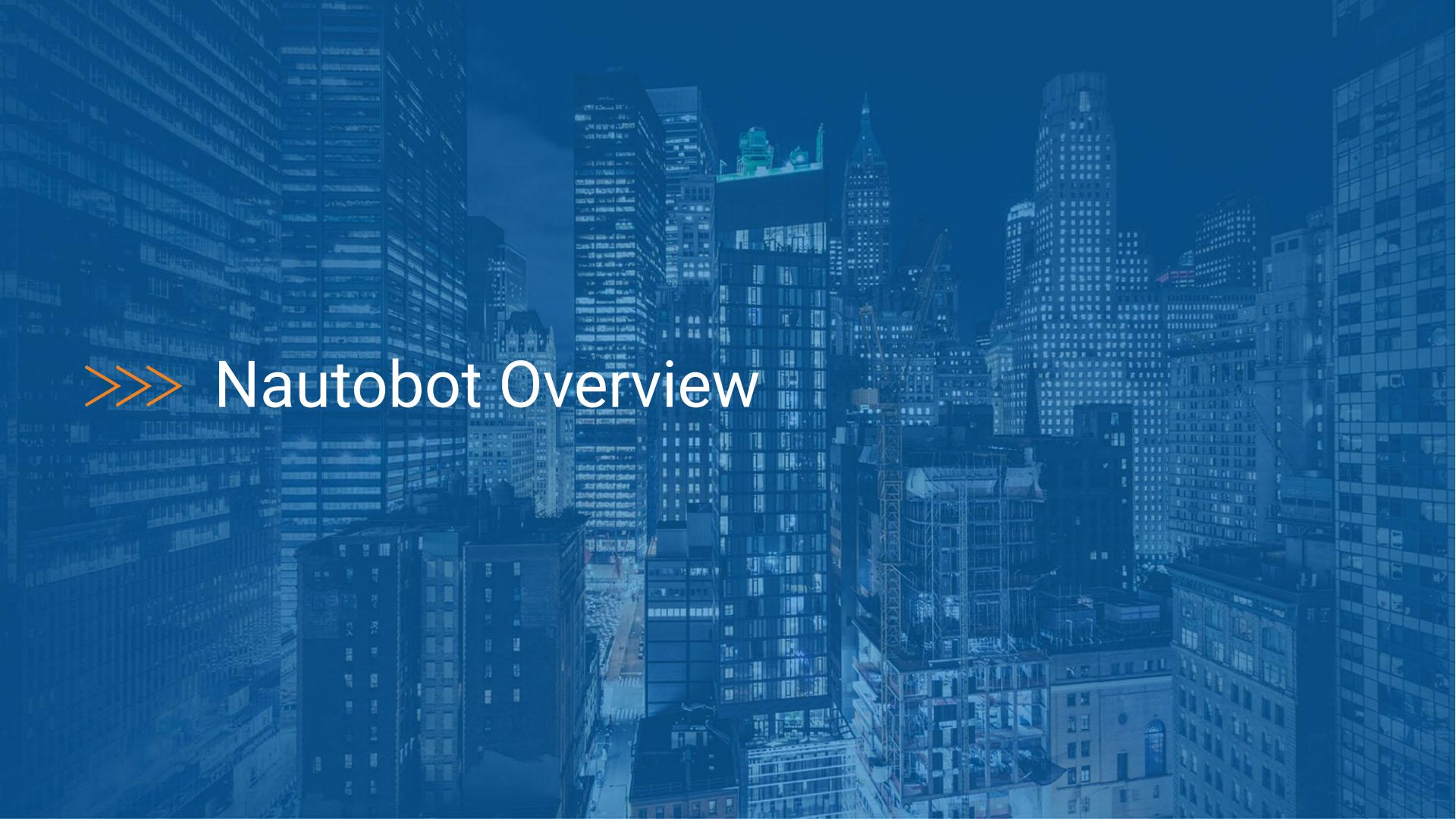

>>> network .toCode()
Nautobot Deep Dive
John Anderson
Principal Consultant & Nautobot Product Owner
>>> nautobot

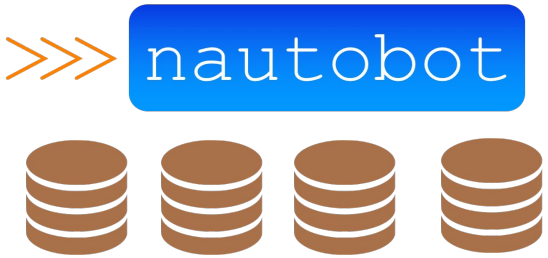


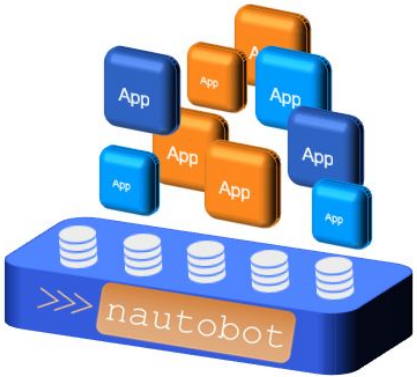
Nautobot Overview

>>> About Nautobot

Source of Truth and Network Automation Platform

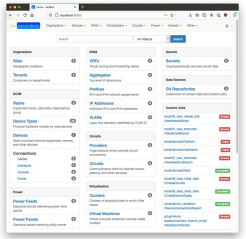
- Open source community project created in 2021
  - *Apache 2 License*
- Sponsored by Network to Code
- Forked from existing NetBox project





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

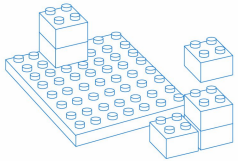
Network Automation Platform

- 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

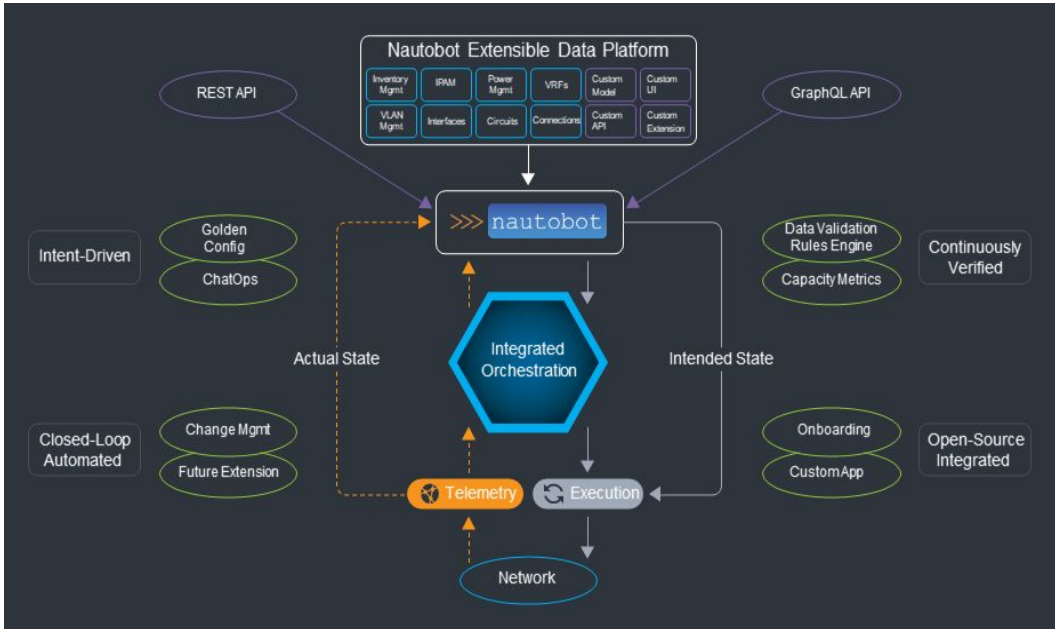


Extensible Plugin System

>>> Intent-Based Networking with Nautobot

Creating a closed-loop automation system

- Nautobot focuses on the intended & desired state
- Nautobot unifies disparate data sources creating a Single Source of Truth
- Nautobot exposes all intended state data and systems through its REST and GraphQL
- Nautobot Apps are used to tailor designs and build integrations with external systems
- External systems are still often used for orchestration, execution, and telemetry



Flexible Source of Truth for Networking

Use Cases & Feature Deep Dive

>>> Flexible Source of Truth for Networking

- Understanding Data Models
- Navigating the Nautobot UI
- Nautobot Core Data Models
- Extending Core Models
  - Custom Fields
  - Config Contexts
  - Git as a Data source
  - Relationships
  - Data Validation
  - Tags
  - Plugins
- Export Templates
- Custom Links
- Changelog

>>> Understanding Data Models

Data models ***describe*** data

- Use well-defined **parameters** to represent of 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

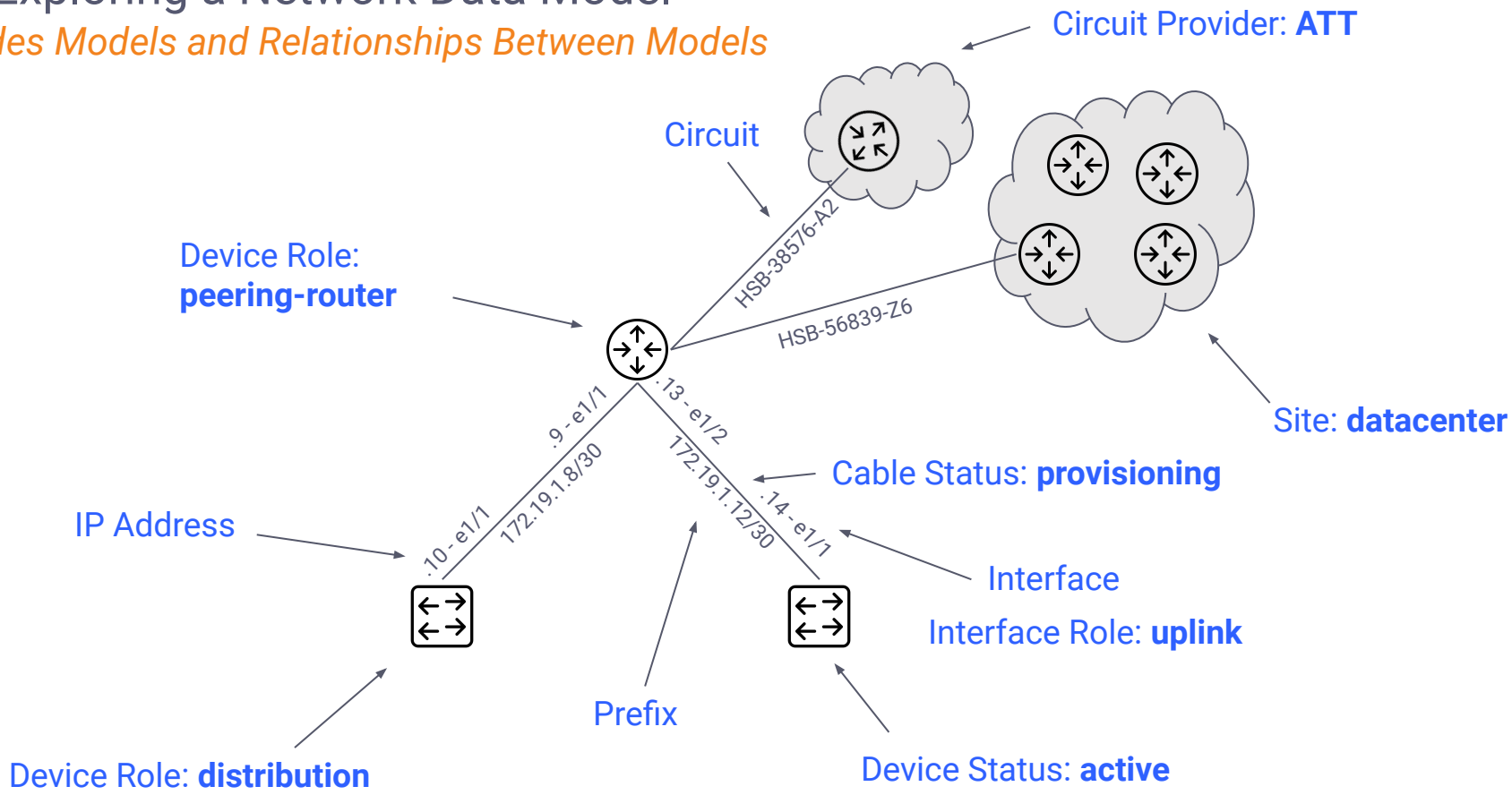
```
vlan_id: 100
vlan_name:
"web_vlan"
```

JSON

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

>>> Exploring a Network Data Model

Includes Models and Relationships Between Models



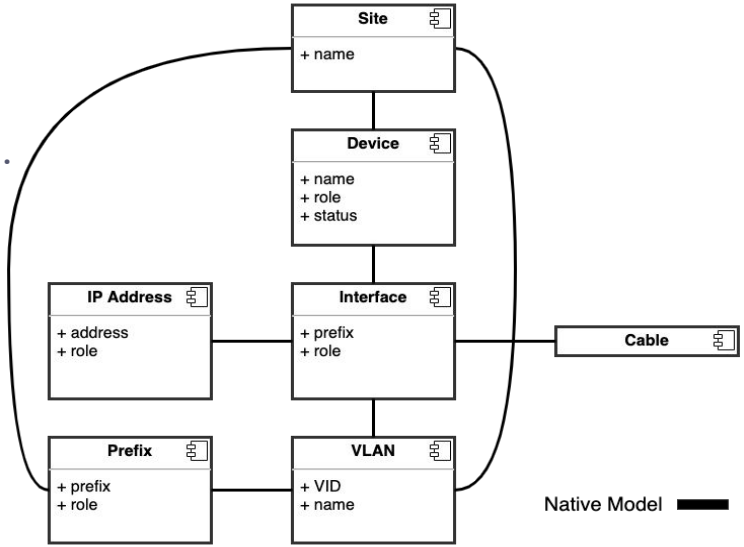
About the Nautobot Data Model

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

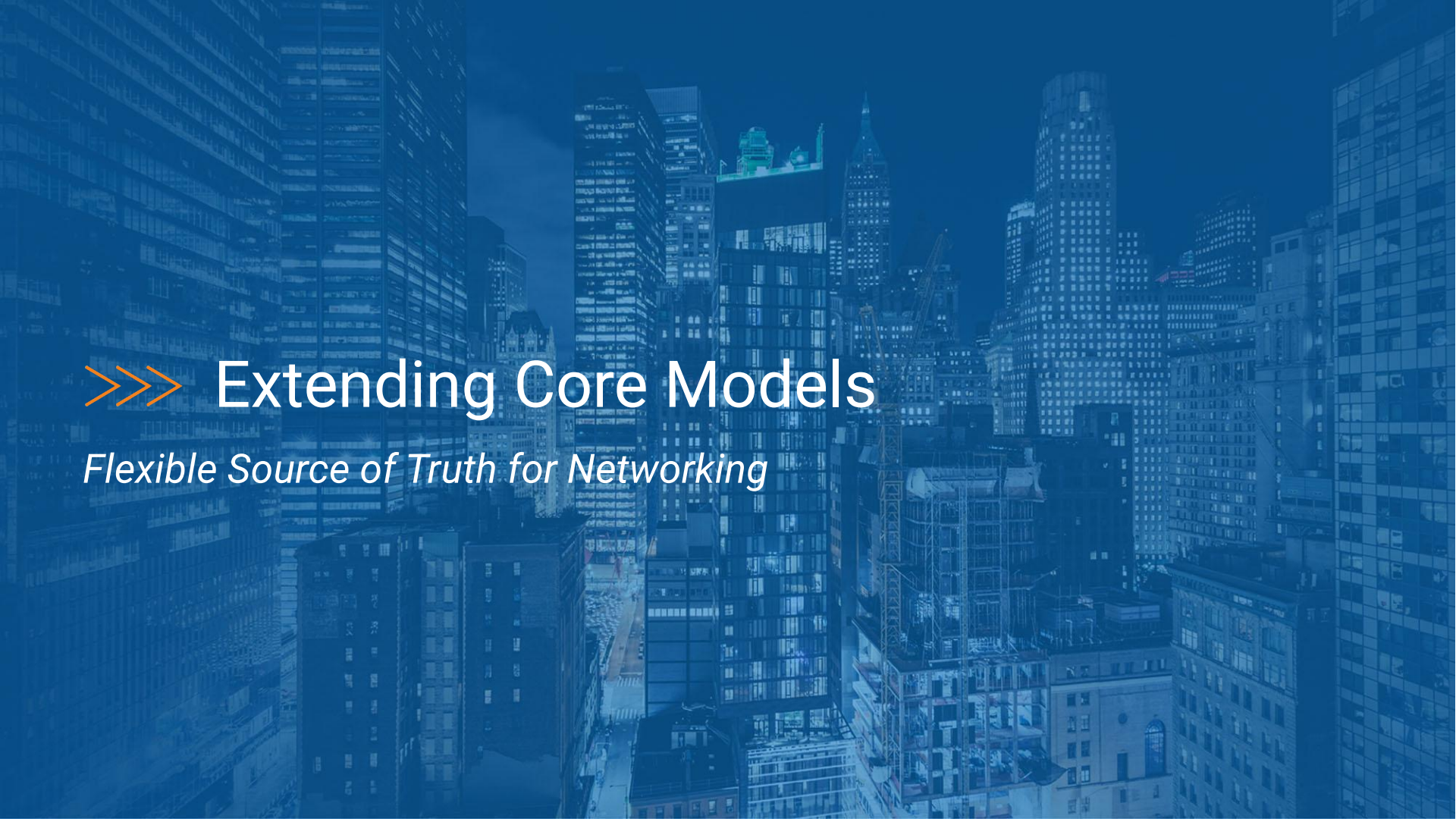
Nautobot is extensible and flexible to meet the unique requirements and use cases for all organizations.

- Nautobot offers flexibility in the data model
- Several targeted features and plugin ecosystem



>>> Nautobot Core Data Models

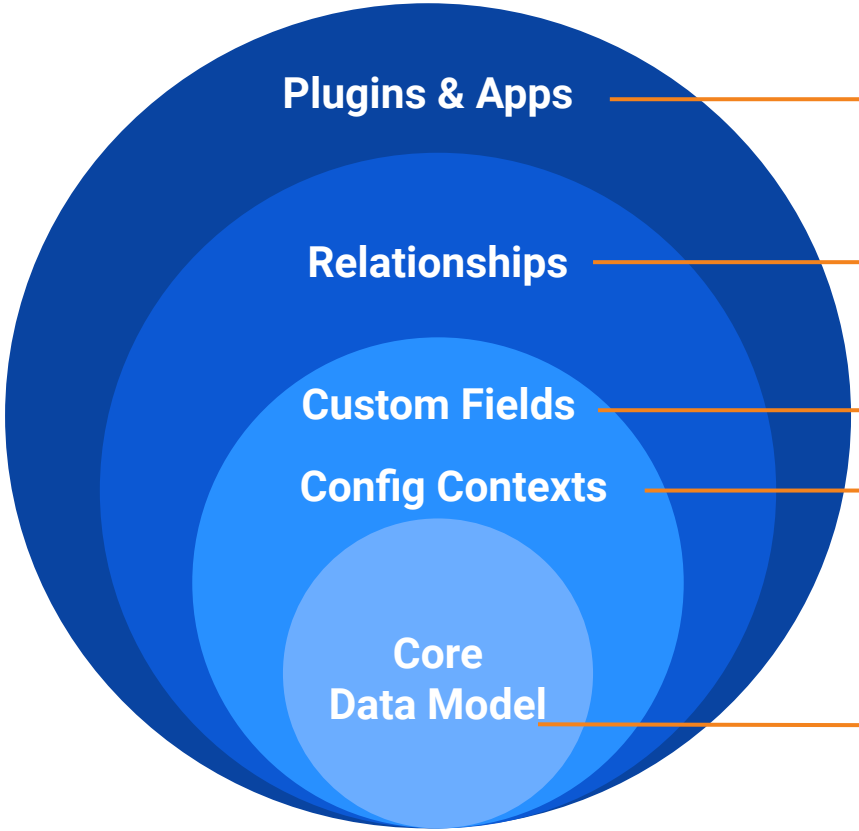
- circuits > provider
- circuits > circuittype
- circuits > circuit
- circuits > circuittermination
- dcim > consoleport
- dcim > consoleserverport
- dcim > powerport
- dcim > poweroutlet
- dcim > interface
- dcim > frontport
- dcim > rearport
- dcim > devicebay
- dcim > inventoryitem
- dcim > manufacturer
- 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



Extending Core Models

Flexible Source of Truth for Networking

>>> Nautobot Data Model Layers



Larger bodies of functionality and reusable in the ecosystem. BGP Peering, Firewalls, Load Balancers.

Defining custom associations between data. Ex: VLAN per rack, VLAN per circuit

Organizational specific attributes like interface role, device ASN, etc.

Structured data (YAML/JSON), optionally loaded from a Git repository

Data Validation

Tracking Devices, Sites, VLANs, Prefixes, Interfaces, Circuits, etc.


Extensible Data Platform for Network Automation

>>> REST API and GraphQL

{ REST }

Expose Create/Read/Update and Delete Endpoints for most objects in the database.

List of endpoints documented with OpenAPI / Swagger. Native support for pagination

- Manipulate one object type at a time (device, interface ..)
- Bulk query, create, update
- The resource is indicated in the url
- Optimized for large volume

GraphQL

Standard query language to query and traverse multiple objects in the database.

Schema of the database generated dynamically and exposed externally.

- Select exactly what should be returned.
- Query multiple type of objects at the same time.
- Optimized for complex query
- Not optimized for large volume

Webhooks

- Nautobot webhooks allow users to have Nautobot make outbound APIs on specific events: Create, Update, and Delete
- Notify 3rd party systems when the Source of Truth data changes
- Common use cases: Trigger automation jobs, Chat alerts, ITSM notifications

Nautobot triggers outbound API call (webhook)

Jobs

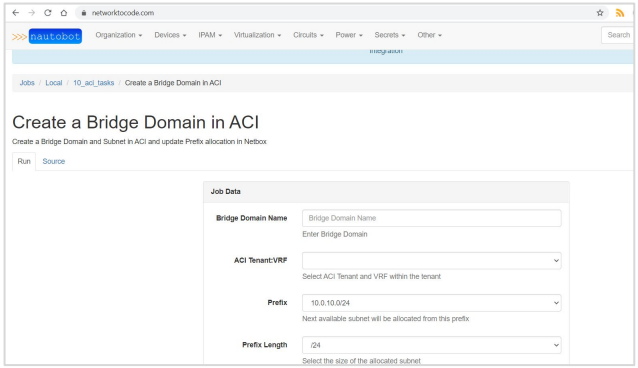
Jobs offer the ability for users to define Python scripts which take user input before running.

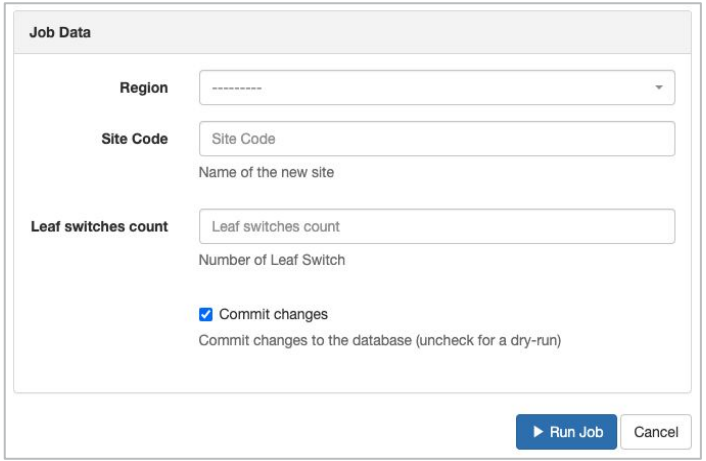
- Can be used to inspect and validate data over time
- Offer the ability to simply report, but can change data if desired

Great way to provision data in a standard way

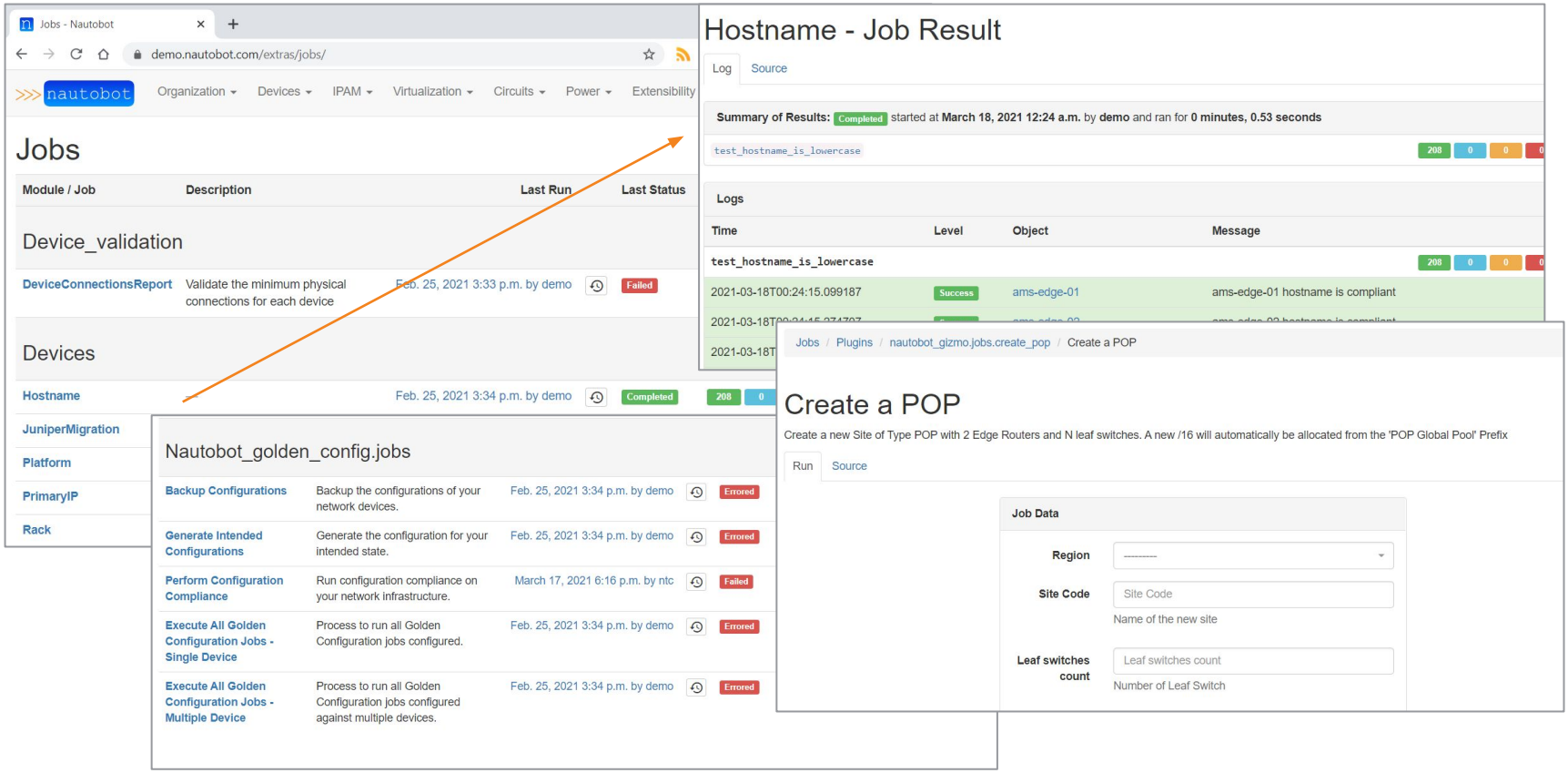
- Enforces patterns and conventions
- Circumvents some data issues

Also great for continuous data reporting





Jobs Examples



App Ecosystem

>>> Nautobot Apps

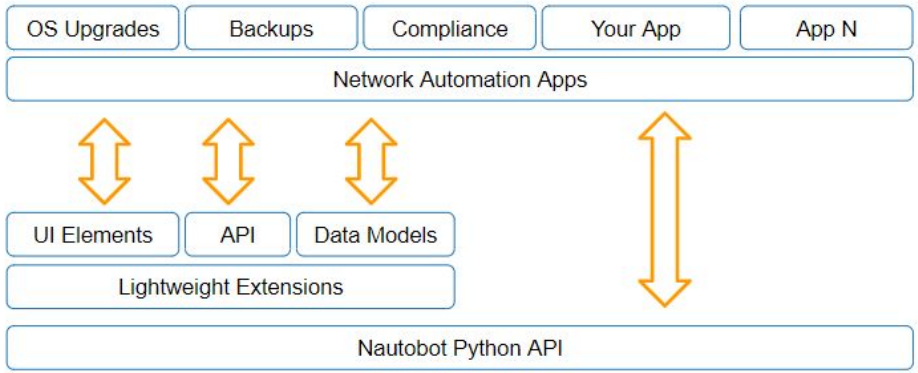
Apps allow developers to invent and implement entirely new functionality.

Based on a plugin architecture, that breaks into two levels of overall functionality, extensions and apps.

Can provide:

- Models (and integration with core)
- Views
- REST APIs
- Inject content into core pages
- Data Validation

Apps are Python packages which are installed by the Nautobot user.



>>> Nautobot App Capabilities

Extending & Customizing Nautobot

Customized Navigation & UI

- Add top-level navigation menu tabs
- Add menu groups beneath *Plugins* navigation menu tab
- Add custom URL navigation to new pages
- Add new pages with custom layouts

Customized Content & UI Presentation

- Add content panels to existing object detail view pages
- Add button(s) to existing object detail pages
- Add banners in Nautobot UI
- Add panels to the Nautobot home page
- Add Jinja2 filters for customized content rendering

Data Models

- Define new data Models
- Define custom validators (validation logic) for Existing Data Models

Create custom REST API Endpoints

Add Custom Content Handlers

Package and Install Jobs With Dependencies

Full GraphQL Integration

Add additional Django Middleware to alter Nautbot's input/output

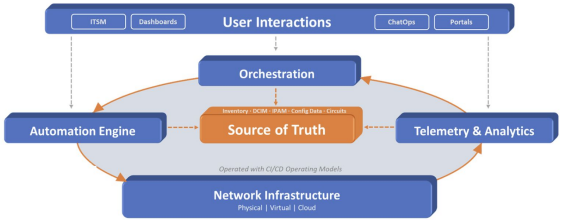
Create new management commands

Leverage Nautobot features inside Apps, including

- Custom fields
- Webhooks
- Relationships

Governance

Evolving Governance in 2023 - Enhancing Customer Experience



John Anderson
Nautobot

Christian Adell
Source of Truth

Ken Celenza
Automation Engine

David Flores
Telemetry (Observability)

Paddy Kelly
User Interactions

Josh VanDeraa
NetDevOps & Programmability

Bryan Culver
Dev Standards

Cristian Sirbu
OSS Governance

David Cole
OSRB

>>> Stakeholder Engagement

Direct product questions and requests to relevant Product Owners

Focus on directing inbound engagement through proper channels

- Ensure feedback and customer asks are getting to the right people
- Product Owner is able to properly plan and roadmap
- Core team remains focused on their committed work


>>> network .toCode()
Thank you!