>>> network.toCode()

Nautobot 1.2.0 Key Features

Tim Fiola

Developer Advocate

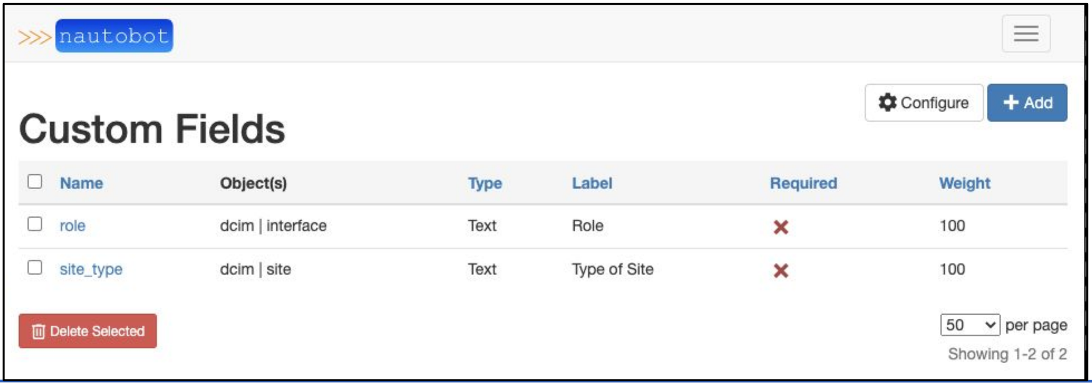
Table of Contents

View for Managing *Custom Fields* Outside of Django Admin (GH229)

Nautobot users can now manage *Custom Fields* outside of the Django Admin Panel

Custom Fields can now be accessed from the top-level Extensibility dropdown

Non-admin users with appropriate permissions can now manage Custom Fields

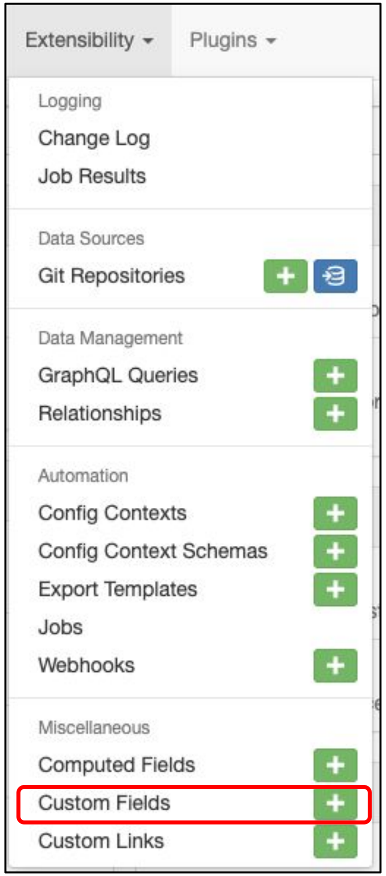


Custom Fields

	Name	Object(s)	Type	Label	Required	Weight
☐	role	dcim | interface	Text	Role	✗	100
☐	site_type	dcim | site	Text	Type of Site	✗	100

50 per page

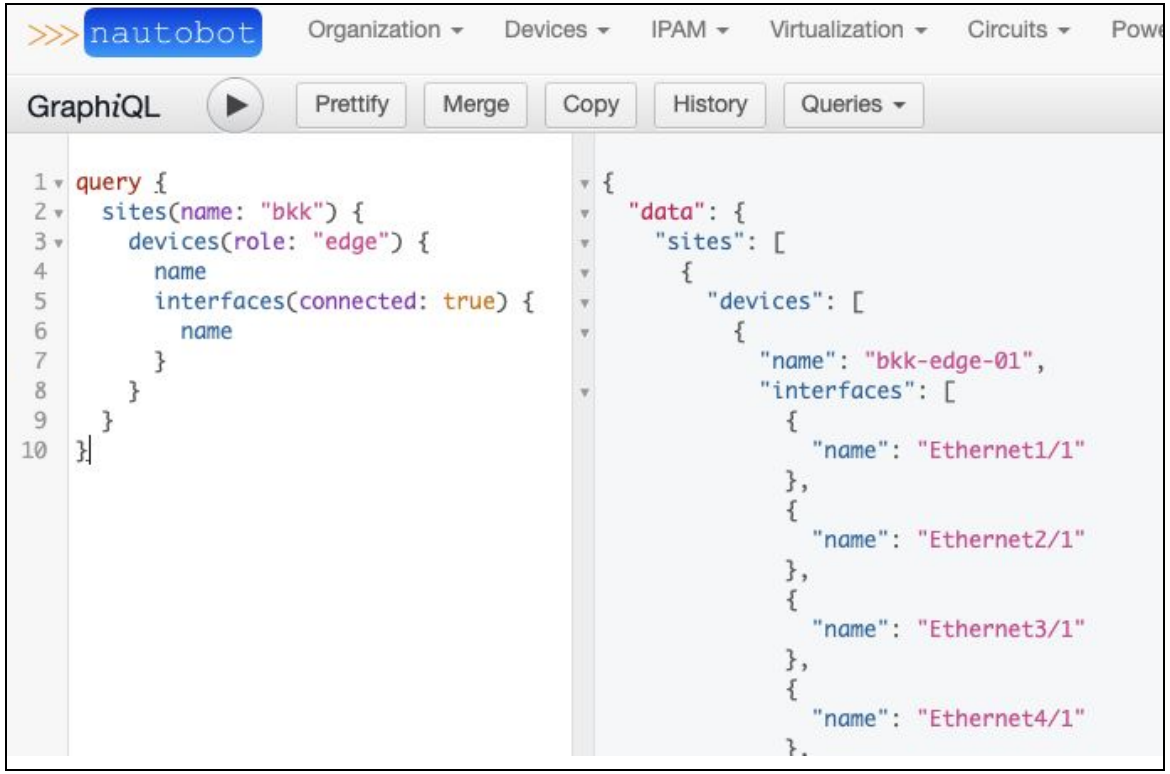
Showing 1-2 of 2



GraphQL Filters at All Levels (GH248)

The GraphQL API now supports query filter parameters at any level within a query

This greatly expands the types of queries Nautobot's GraphQL can handle



```
query {
  sites(name: "bkk") {
    devices(role: "edge") {
      name
      interfaces(connected: true) {
        name
      }
    }
  }
}
```

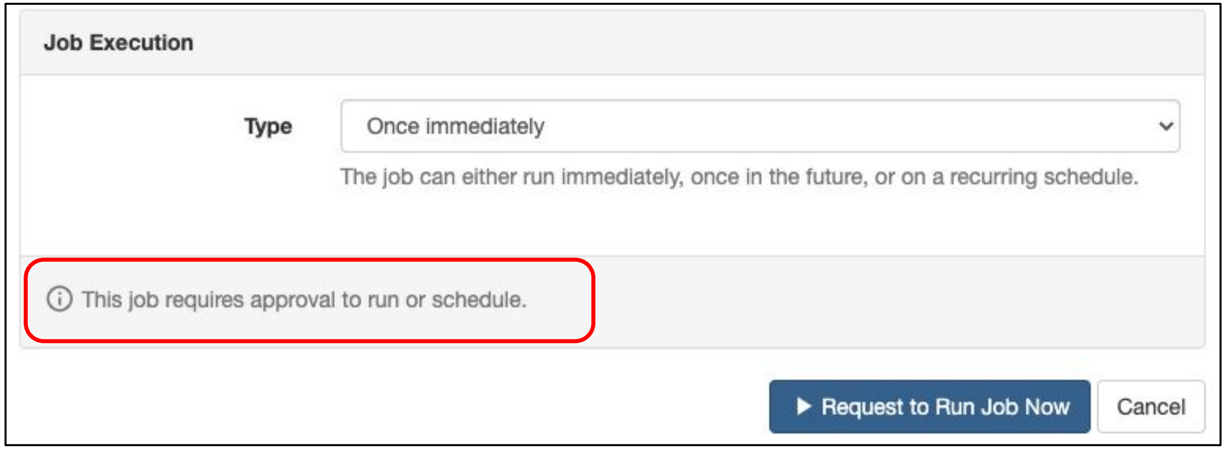
```
{
  "data": {
    "sites": [
      {
        "devices": [
          {
            "name": "bkk-edge-01",
            "interfaces": [
              {
                "name": "Ethernet1/1"
              },
              {
                "name": "Ethernet2/1"
              },
              {
                "name": "Ethernet3/1"
              },
              {
                "name": "Ethernet4/1"
              },
```

>>> Job Approval (GH125)

Jobs can now be optionally set to `approval_required = True` on their `Meta` object

Jobs with this attribute set to True

- Will have a marker, indicating approval is required
- Will be placed into an approval queue
- Require any other user other than the submitter to approve


Job Execution

Type Once immediately

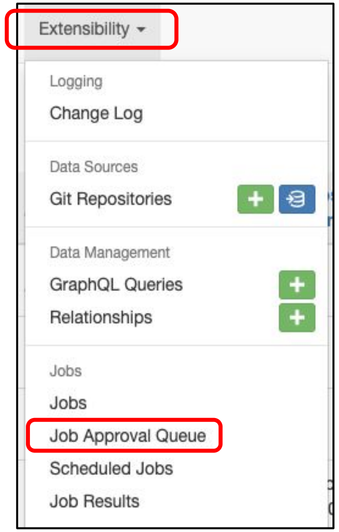
The job can either run immediately, once in the future, or on a recurring schedule.

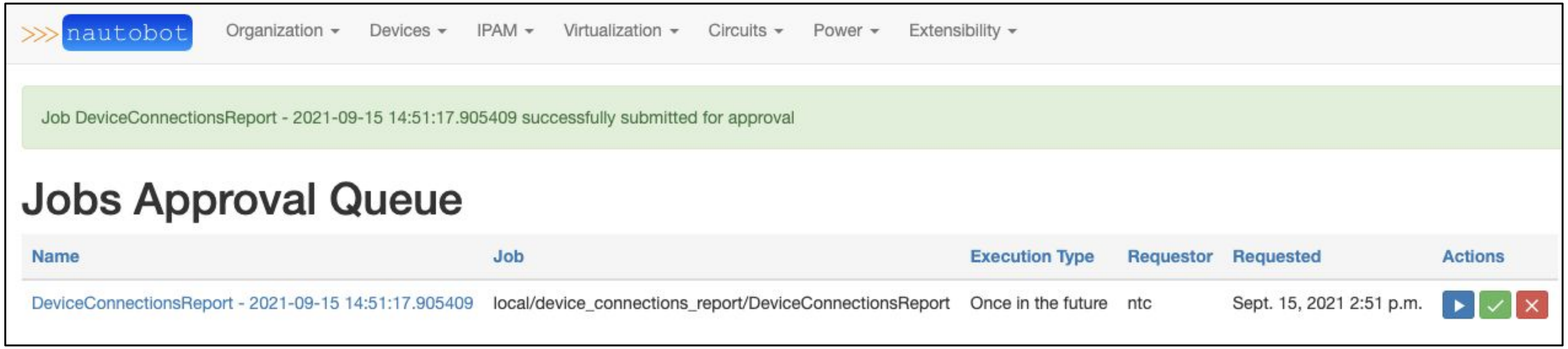
ⓘ This job requires approval to run or schedule.

Job Approval (GH125) (continued)

The Job Approval Queue is accessed from *Extensibility→ Jobs → Job Approval Queue*

Jobs can be approved via the UI or API





Job DeviceConnectionsReport - 2021-09-15 14:51:17.905409 successfully submitted for approval

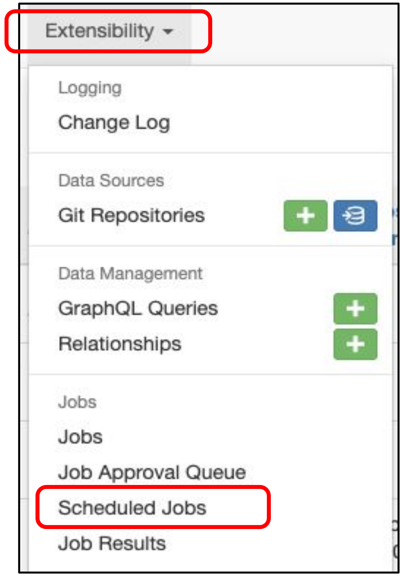
Jobs Approval Queue

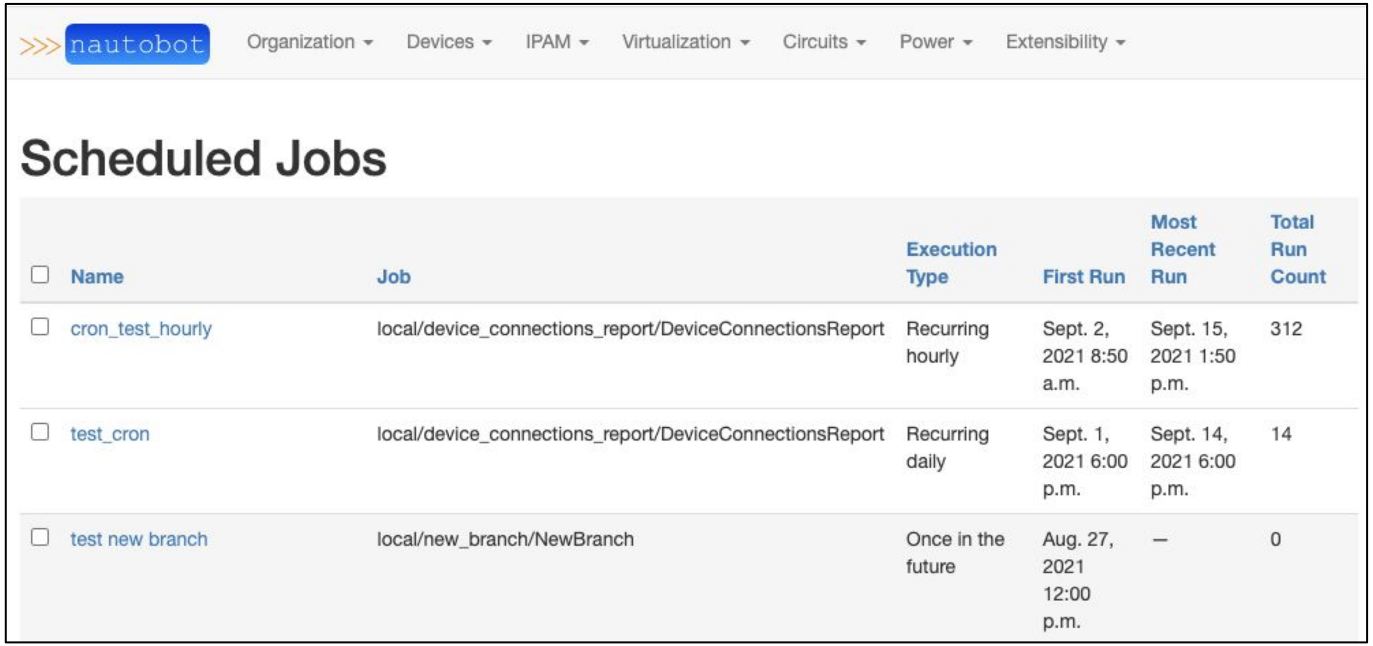
Name	Job	Execution Type	Requestor	Requested	Actions
DeviceConnectionsReport - 2021-09-15 14:51:17.905409	local/device_connections_report/DeviceConnectionsReport	Once in the future	ntc	Sept. 15, 2021 2:51 p.m.	▶ ✓ ✕

Job Scheduling (GH374)

Jobs can now be:

- Scheduled for execution at a future date and time
- Scheduled for repeated execution on an hourly, daily, or weekly recurring cadence





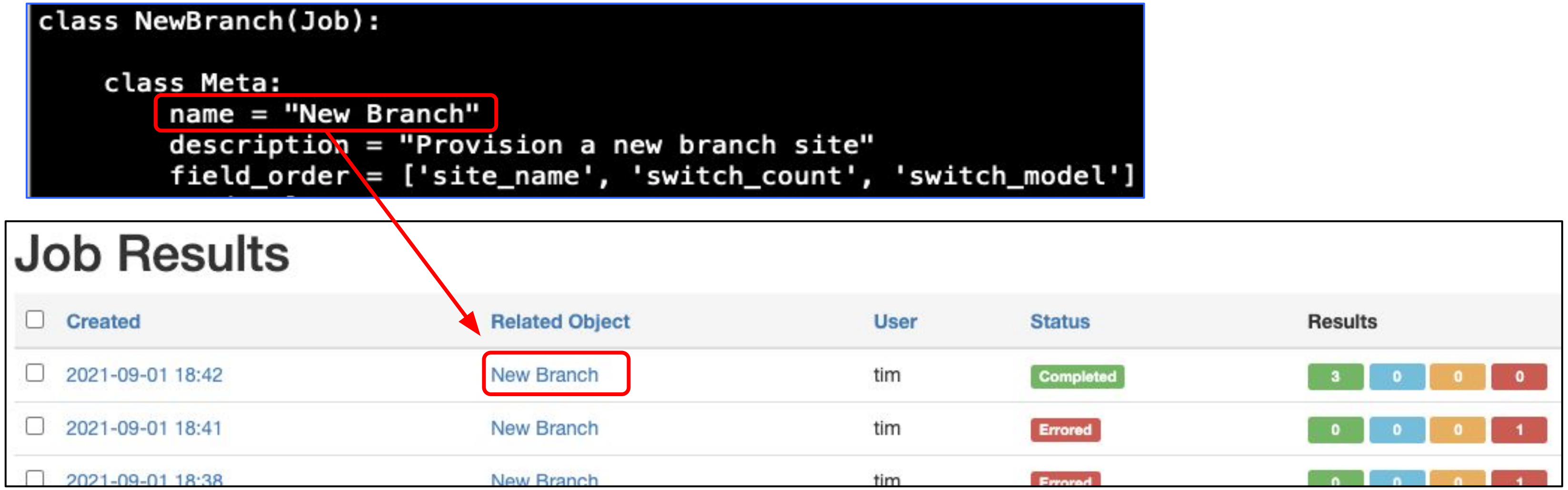
Scheduled Jobs

Name	Job	Execution Type	First Run	Most Recent Run	Total Run Count
cron_test_hourly	local/device_connections_report/DeviceConnectionsReport	Recurring hourly	Sept. 2, 2021 8:50 a.m.	Sept. 15, 2021 1:50 p.m.	312
test_cron	local/device_connections_report/DeviceConnectionsReport	Recurring daily	Sept. 1, 2021 6:00 p.m.	Sept. 14, 2021 6:00 p.m.	14
test new branch	local/new_branch/NewBranch	Once in the future	Aug. 27, 2021 12:00 p.m.	—	0

Job Results Page - Use Meta Name Attribute (GH472)

Job Results page can now show user-friendly names for the jobs

- Users can specify a user-friendly job name by specifying the *name* in the Meta class for the job
- If no Meta *name* attribute is specified, the displayed name reverts to `/path/to/job`
- The job name appears in the **Related Object** column



```
class NewBranch(Job):

    class Meta:
        name = "New Branch"
        description = "Provision a new branch site"
        field_order = ['site_name', 'switch_count', 'switch_model']
```

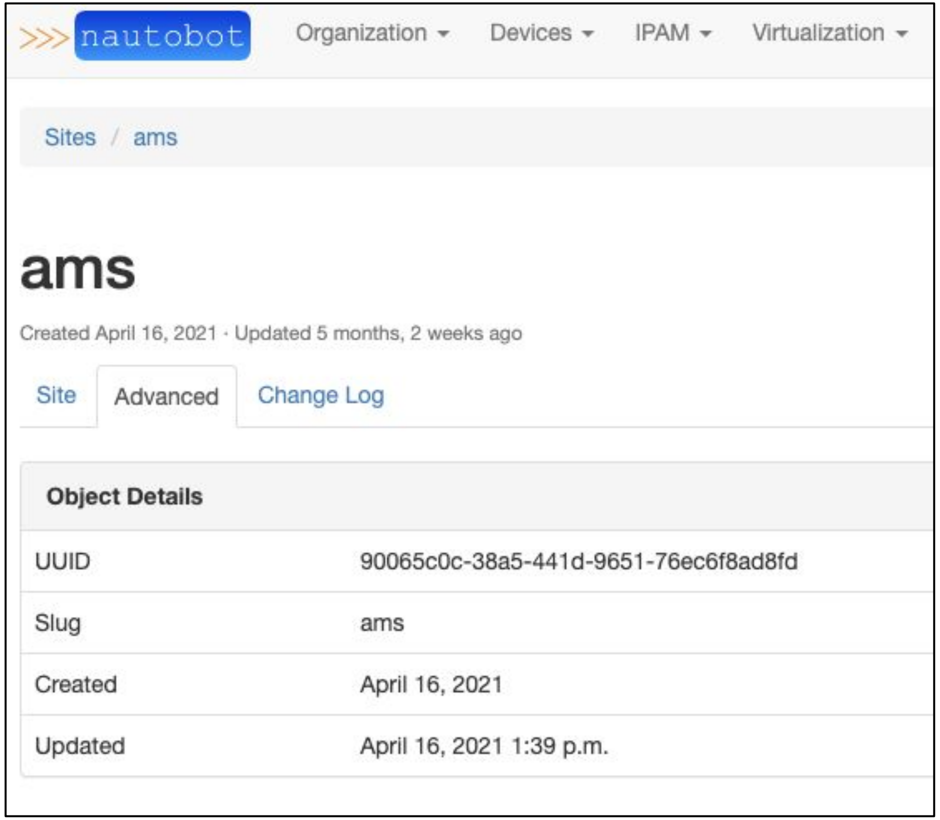
Job Results

☐	Created	Related Object	User	Status	Results			
☐	2021-09-01 18:42	New Branch	tim	Completed	3	0	0	0
☐	2021-09-01 18:41	New Branch	tim	Errored	0	0	0	1
☐	2021-09-01 18:38	New Branch	tim	Errored	0	0	0	1

Advanced Tab in Object Detail Views (GH585)

Added *Advanced* tab to object detail views including UUID and slug information

Helps in constructing REST API calls and GraphQL queries against specific objects because it quickly provides the specific slug and UUID information



ams

Created April 16, 2021 · Updated 5 months, 2 weeks ago

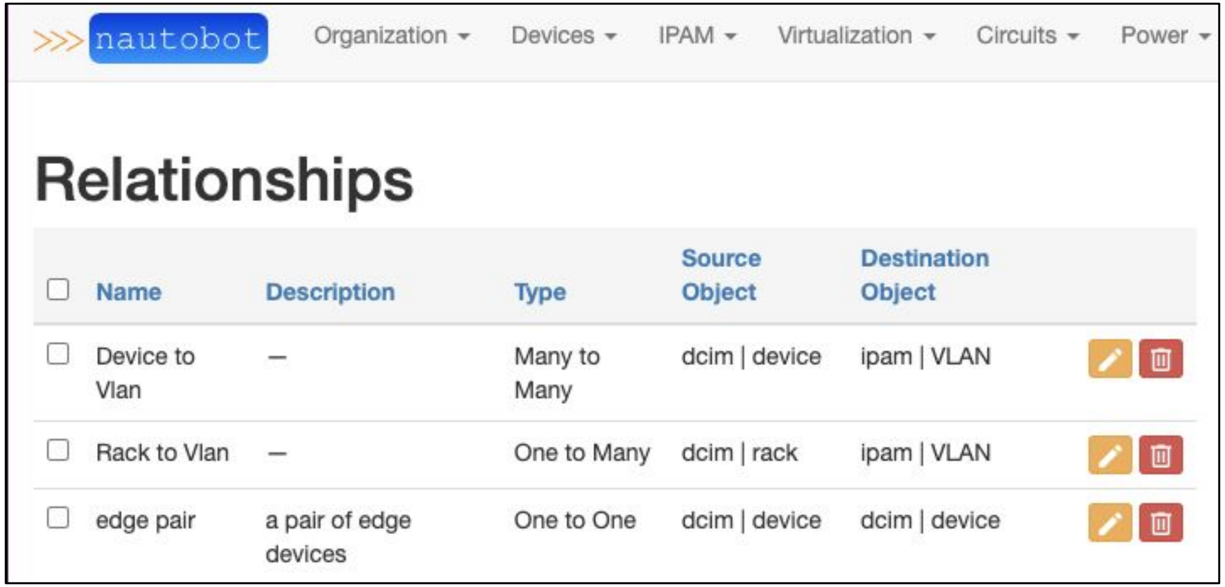
Object Details

UUID	90065c0c-38a5-441d-9651-76ec6f8ad8fd
Slug	ams
Created	April 16, 2021
Updated	April 16, 2021 1:39 p.m.

Same-Type and Symmetric Relationships (GH157)

The Relationships feature has been extended in two ways:

- Relationships between the same object type (e.g. device-to-device) are now permitted
- For same-object-type relationships specifically, symmetric (peer-to-peer rather than source-to-destination) relationships are now an option



Relationships

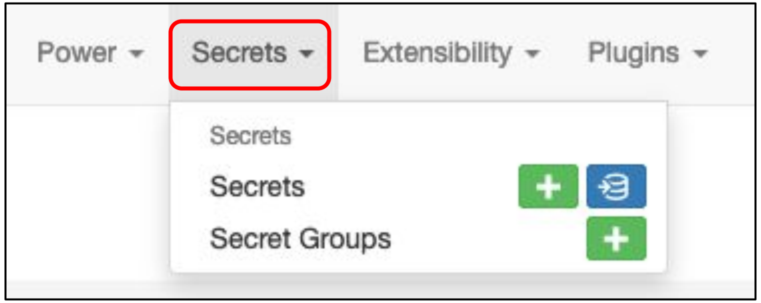
	Name	Description	Type	Source Object	Destination Object
☐	Device to Vlan	—	Many to Many	dcim | device	ipam | VLAN
☐	Rack to Vlan	—	One to Many	dcim | rack	ipam | VLAN
☐	edge pair	a pair of edge devices	One to One	dcim | device	dcim | device

Secrets Integration (GH541)

This feature allows secure retrieval and usage of arbitrary secrets

Allows for industry standard secrets storage tooling

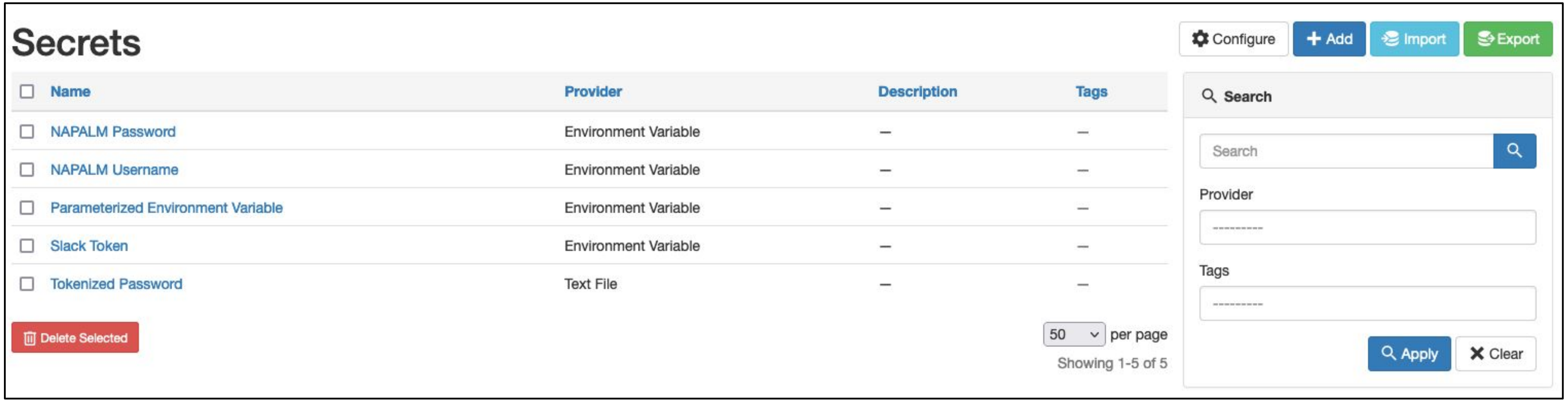
Secrets are not stored directly in Nautobot



>>> Secrets Integration (GH541) (continued)

Introduces *Secret* model

- Defines how Nautobot can **retrieve** secret value and use when needed
- Does **not** store secret value



Name	Provider	Description	Tags
NAPALM Password	Environment Variable	—	—
NAPALM Username	Environment Variable	—	—
Parameterized Environment Variable	Environment Variable	—	—
Slack Token	Environment Variable	—	—
Tokenized Password	Text File	—	—

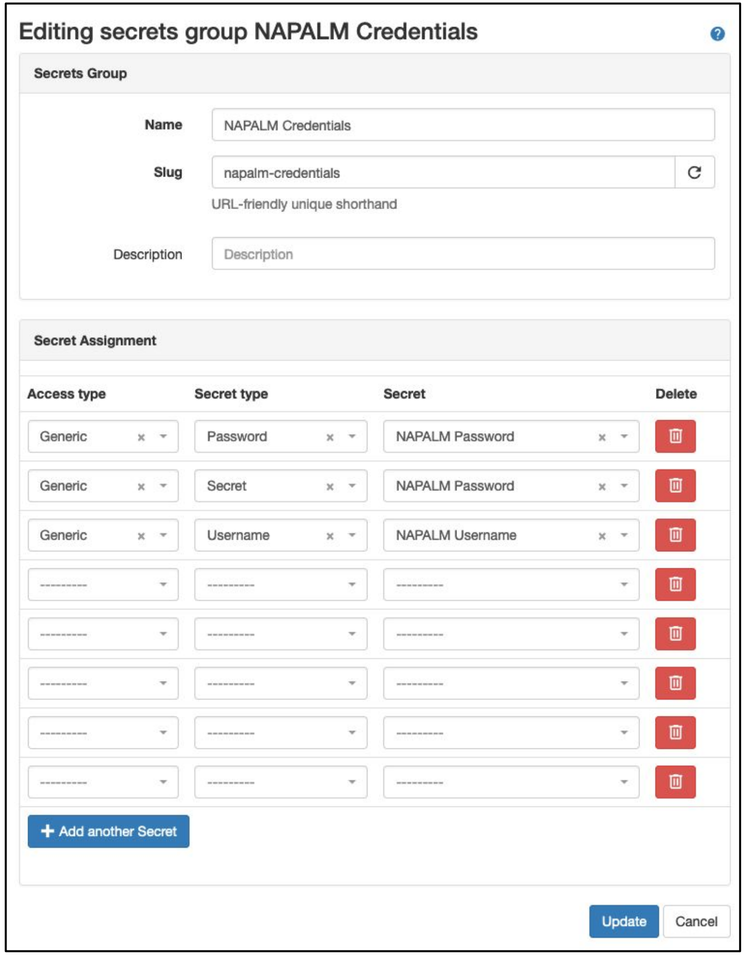
Secrets Integration (GH541) (continued)

Introduces *Secrets Group* model

- Collects and assigns meaning to secrets

The example to the right shows a Secrets Group consisting of three NAPALM credentials needed to access and configure a device

- Username
- Password
- Secret



Editing secrets group NAPALM Credentials

Secrets Group

Name	NAPALM Credentials
Slug	napalm-credentials
	URL-friendly unique shorthand
Description	Description

Secret Assignment

Access type	Secret type	Secret	Delete
Generic	Password	NAPALM Password	
Generic	Secret	NAPALM Password	
Generic	Username	NAPALM Username	
---------	---------	---------	
---------	---------	---------	
---------	---------	---------	
---------	---------	---------	
---------	---------	---------	

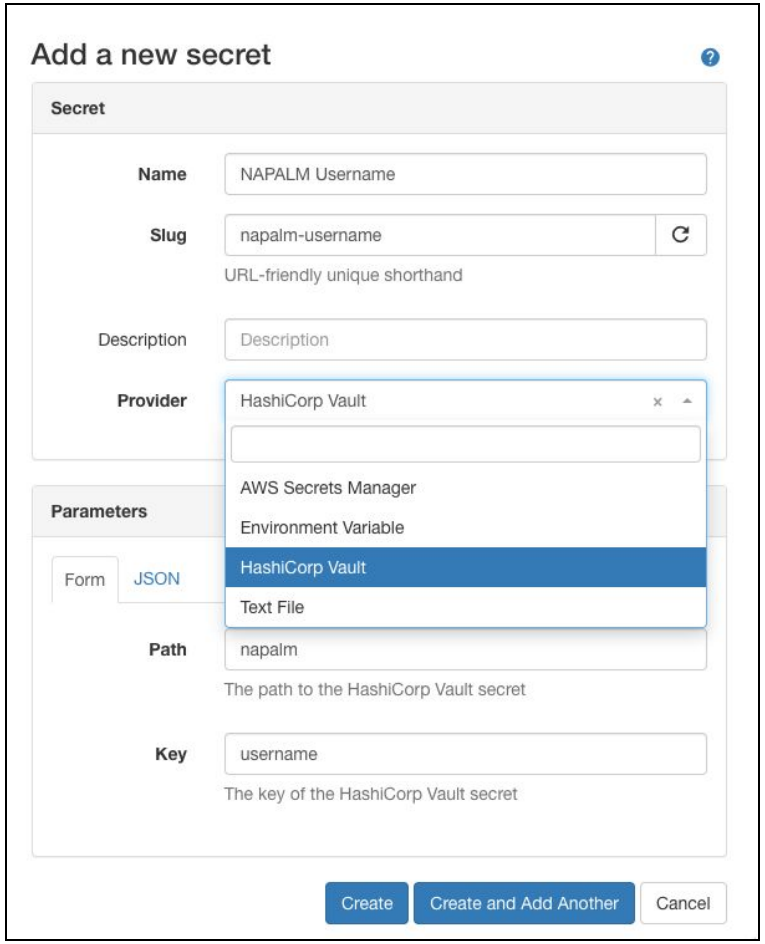
Secrets Integration (GH541) (continued)

The feature's core code enables these two secret sources:

- Environment variable
- Text file

This feature also enables the *Nautobot Secrets Providers* app:

- Defines additional secret providers
- This app initially includes AWS Secrets Manager and HashiCorp Vault



Add a new secret

Secret

Name	NAPALM Username
Slug	napalm-username
	URL-friendly unique shorthand
Description	Description
Provider	HashiCorp Vault

Parameters

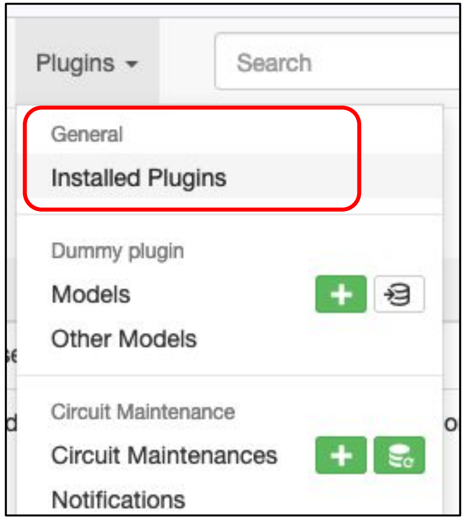
Path	napalm
	The path to the HashiCorp Vault secret
Key	username
	The key of the HashiCorp Vault secret

>>> Installed Plugins (Apps) View and Homepage (GH935)

Provides *Installed Plugins* view that is accessible by all authenticated users

- Replaces the *Installed Plugins* view that was accessible only to *admin* users

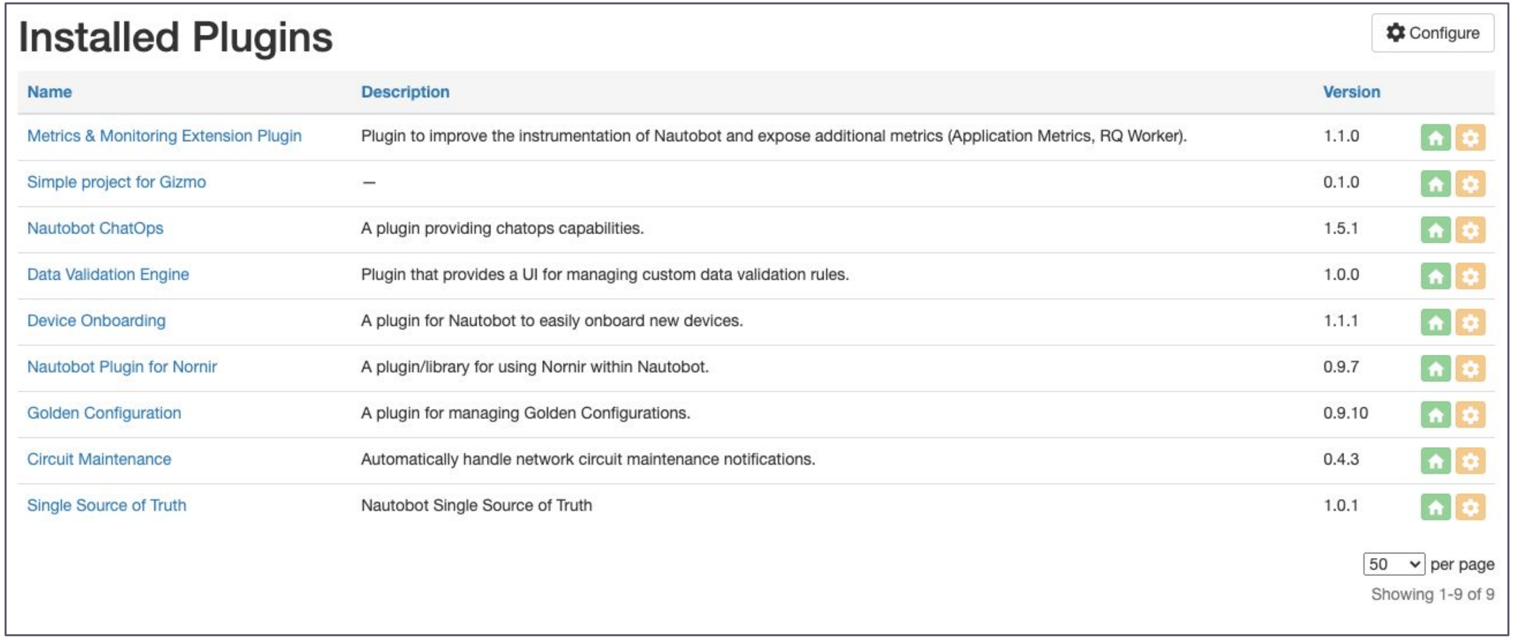
Access the view by navigating to *Plugins→Installed Plugins*



Installed Plugins (Apps) View and Homepage (GH935) (continued)

Users are taken to the *Installed Plugins* list view

- Provides list view of installed and enabled plugins (apps)
- Similar to former administrator-only view



Installed Plugins

Configure

Name	Description	Version	
Metrics & Monitoring Extension Plugin	Plugin to improve the instrumentation of Nautobot and expose additional metrics (Application Metrics, RQ Worker).	1.1.0	
Simple project for Gizmo	—	0.1.0	
Nautobot ChatOps	A plugin providing chatops capabilities.	1.5.1	
Data Validation Engine	Plugin that provides a UI for managing custom data validation rules.	1.0.0	
Device Onboarding	A plugin for Nautobot to easily onboard new devices.	1.1.1	
Nautobot Plugin for Nornir	A plugin/library for using Nornir within Nautobot.	0.9.7	
Golden Configuration	A plugin for managing Golden Configurations.	0.9.10	
Circuit Maintenance	Automatically handle network circuit maintenance notifications.	0.4.3	
Single Source of Truth	Nautobot Single Source of Truth	1.0.1	

50 per page

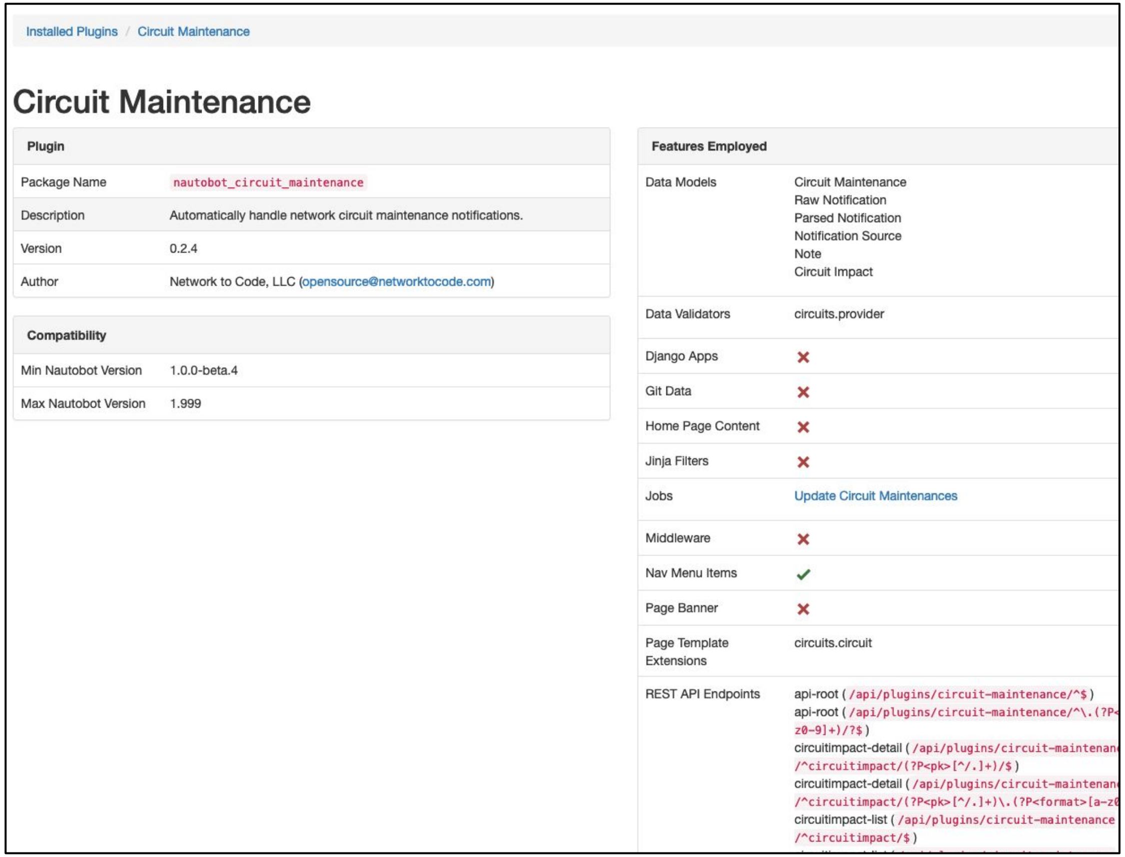
Showing 1-9 of 9

Installed Plugins (Apps) View and Homepage (GH935) (continued)

Clicking on a plugin/app from the list view takes the user to a detail view of the plugin/app

This page includes an in-depth look at the capabilities of the plugin

- Includes info on which Nautobot plugin features it uses



Installed Plugins / Circuit Maintenance

Circuit Maintenance

Plugin	
Package Name	`nautobot_circuit_maintenance`
Description	Automatically handle network circuit maintenance notifications.
Version	0.2.4
Author	Network to Code, LLC (opensource@networktocode.com)

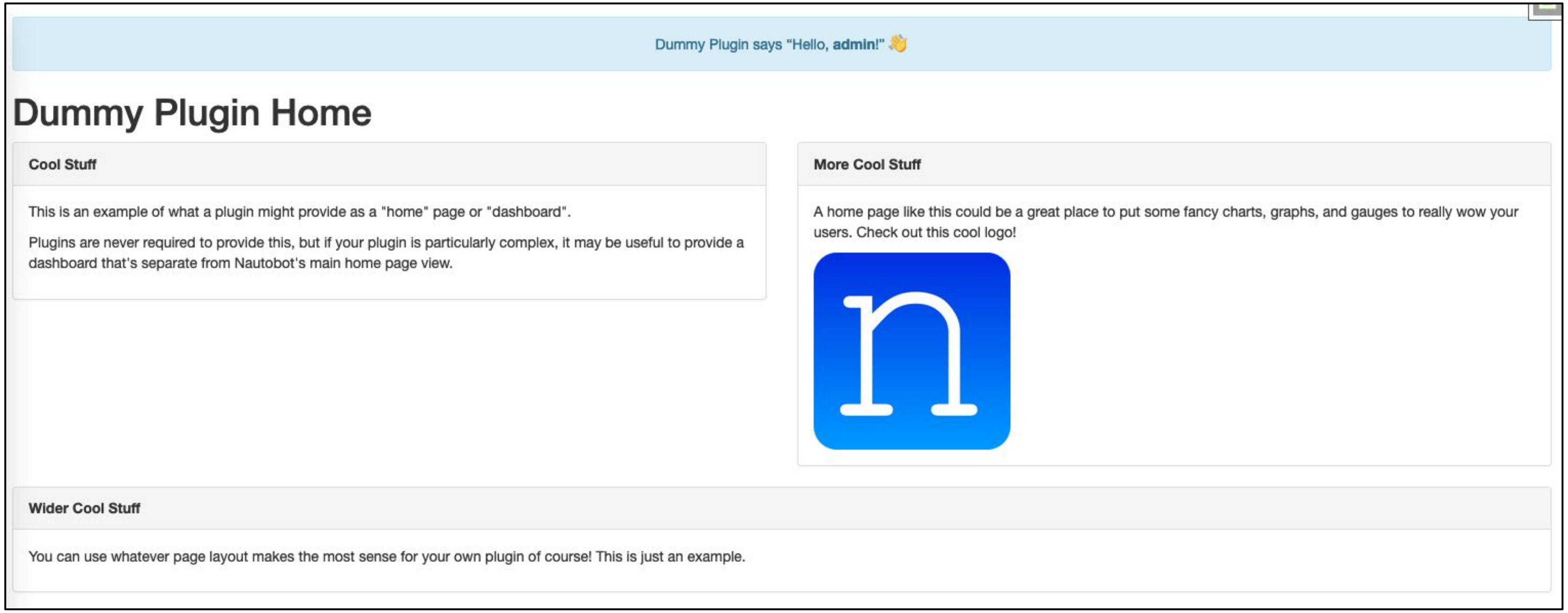
Compatibility	
Min Nautobot Version	1.0.0-beta.4
Max Nautobot Version	1.999

Features Employed	
Data Models	Circuit Maintenance Raw Notification Parsed Notification Notification Source Note Circuit Impact
Data Validators	circuits.provider
Django Apps	✗
Git Data	✗
Home Page Content	✗
Jinja Filters	✗
Jobs	Update Circuit Maintenances
Middleware	✗
Nav Menu Items	✓
Page Banner	✗
Page Template Extensions	circuits.circuit
REST API Endpoints	api-root (`/api/plugins/circuit-maintenance/^$`) api-root (`/api/plugins/circuit-maintenance/^\.(?P` `z0-9]+)/?$`) circuitimpact-detail (`/api/plugins/circuit-maintenan` `/^circuitimpact/(?P<pk>[^/.]+)/$`) circuitimpact-detail (`/api/plugins/circuit-maintenan` `/^circuitimpact/(?P<pk>[^/.]+)\.(?P<format>[a-z0` circuitimpact-list (`/api/plugins/circuit-maintenance` `/^circuitimpact/$`)

>>> Installed Plugins (Apps) View and Homepage (GH935) (continued)

This feature also allows for a home page for the app, accessible when the user clicks on the *home* icon for the app on the *Installed Plugins* page

The app home page is highly customizable



Dummy Plugin says "Hello, **admin**!" 👋

Dummy Plugin Home

Cool Stuff

This is an example of what a plugin might provide as a "home" page or "dashboard".

Plugins are never required to provide this, but if your plugin is particularly complex, it may be useful to provide a dashboard that's separate from Nautobot's main home page view.

More Cool Stuff

A home page like this could be a great place to put some fancy charts, graphs, and gauges to really wow your users. Check out this cool logo!

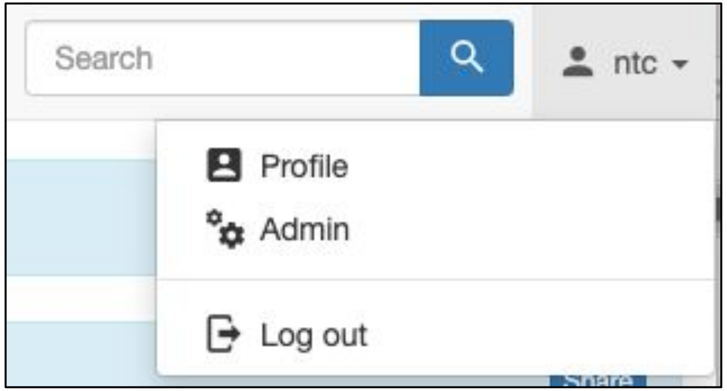
Wider Cool Stuff

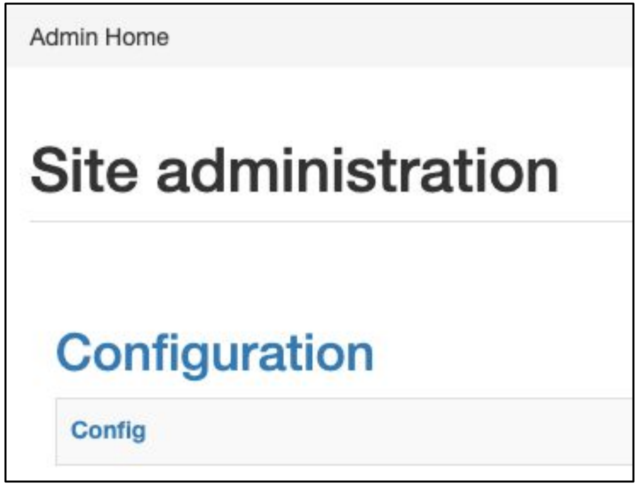
You can use whatever page layout makes the most sense for your own plugin of course! This is just an example.

Admin Configuration UI (GH370)

The Nautobot Admin UI now includes a *Configuration* page

- Navigate to *Admin* → *Configuration* → *Config*

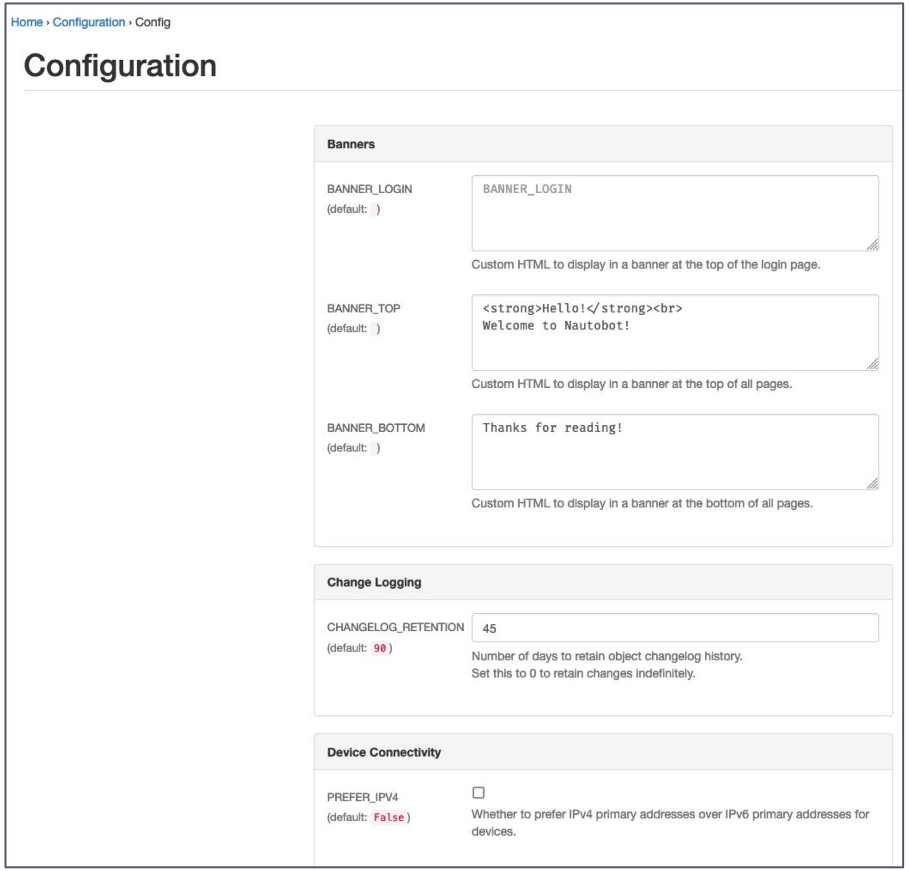




>>> Admin Configuration UI (GH370) (continued)

Features

- Admin user can dynamically customize a number of optional settings
- It's an alternative to editing **nautobot_config.py** and restarting the Nautobot processes
- Specific settings in *nautobot_config.py* will override corresponding sections on this page
- The settings available here do not affect the operation of the Nautobot service itself



Home › Configuration › Config

Configuration

Banners

BANNER_LOGIN (default:)	BANNER_LOGIN Custom HTML to display in a banner at the top of the login page.
BANNER_TOP (default:)	`<strong>Hello!</strong> ` `Welcome to Nautobot!` Custom HTML to display in a banner at the top of all pages.
BANNER_BOTTOM (default:)	`Thanks for reading!` Custom HTML to display in a banner at the bottom of all pages.

Change Logging

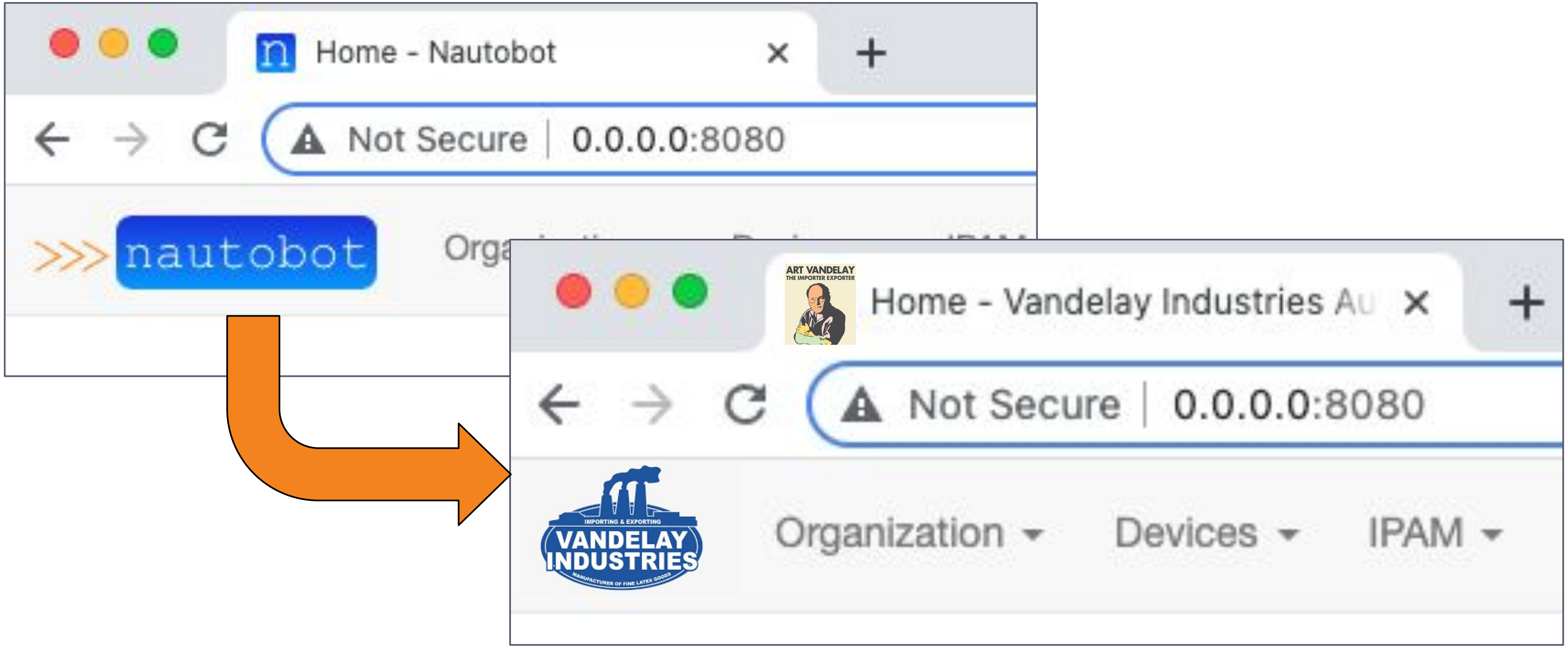
CHANGELOG_RETENTION (default: `90`)	45 Number of days to retain object changelog history. Set this to 0 to retain changes indefinitely.

Device Connectivity

PREFER_IPV4 (default: `False`)	☐ Whether to prefer IPv4 primary addresses over IPv6 primary addresses for devices.

>>> Custom Branding (GH859)

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

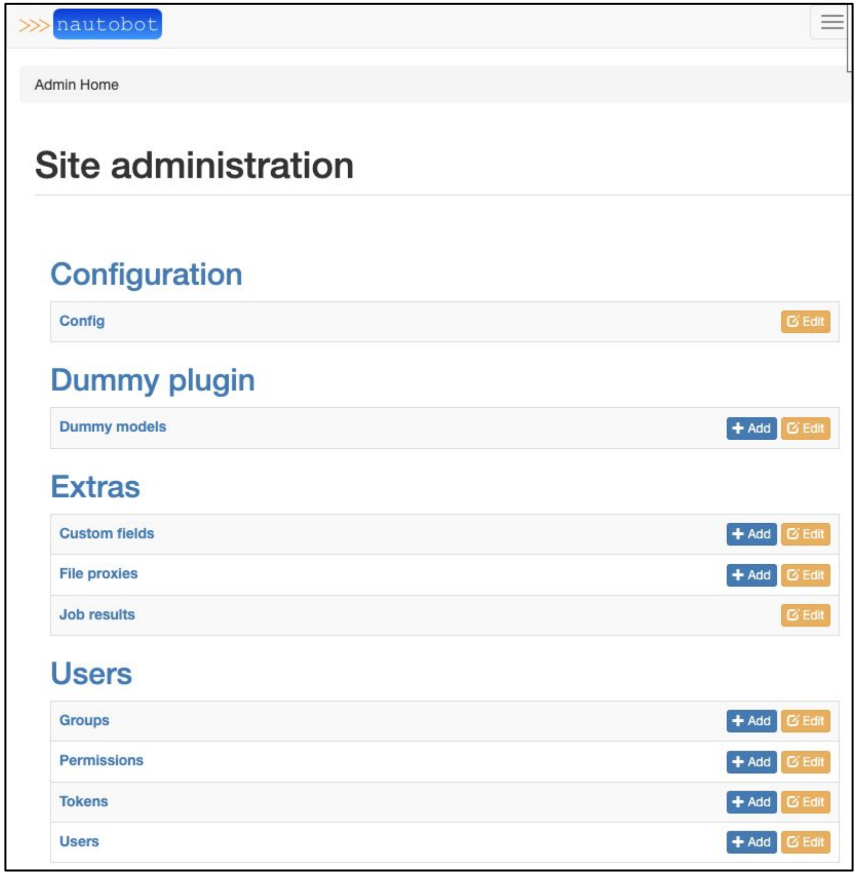


New Skin For Django Admin Page (GH900)

Nautobot runs on the Django framework

Nautobot's Django *admin* section used the generic, default layout

In 1.2.0 the *admin* site has been revised and re-skinned to more closely match the core Nautobot UI


nautobot

Admin Home

Site administration

Configuration

Config	Edit

Dummy plugin

Dummy models	+ Add Edit

Extras

Custom fields	+ Add Edit
File proxies	+ Add Edit
Job results	Edit

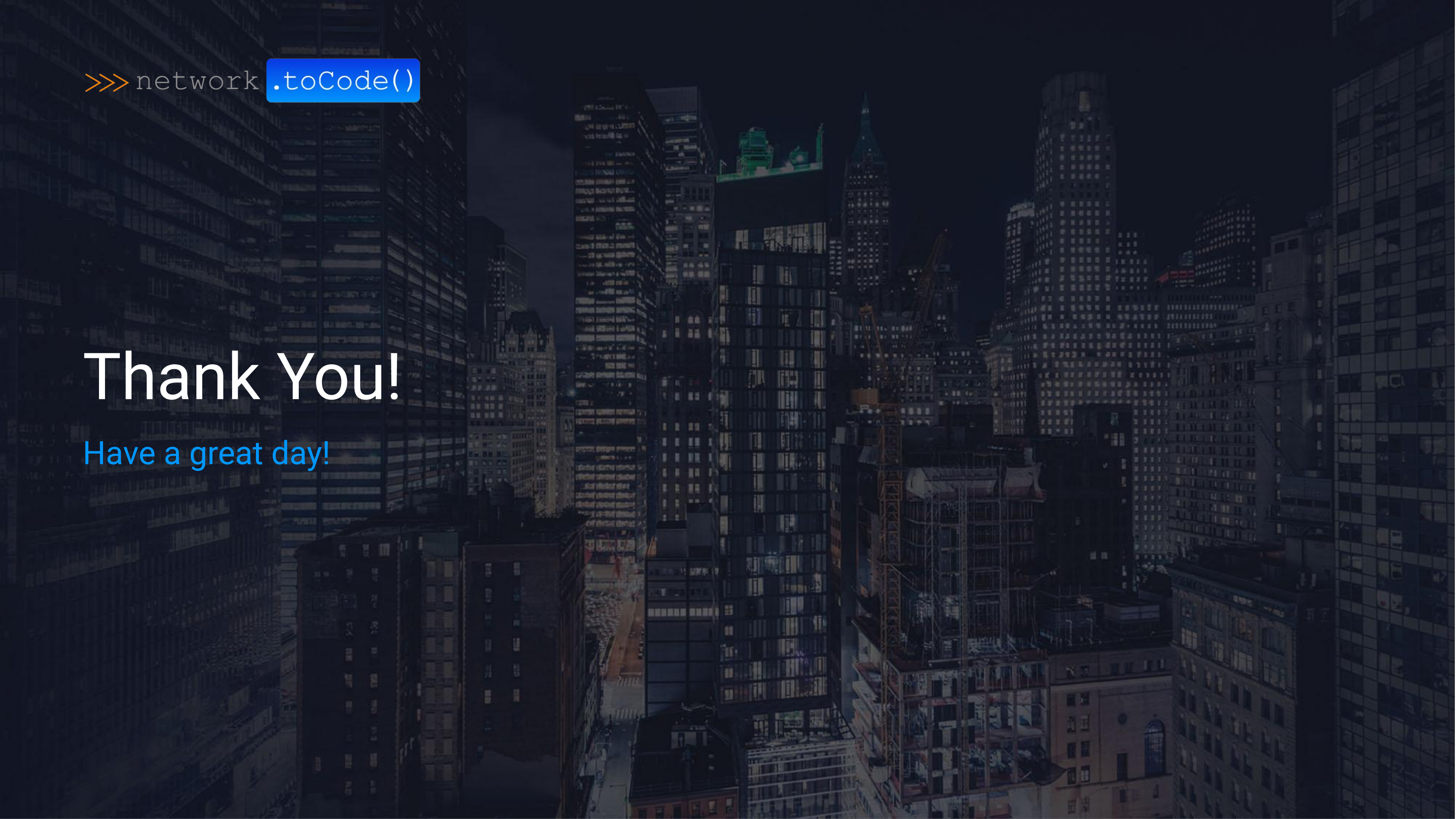
Users

Groups	+ Add Edit
Permissions	+ Add Edit
Tokens	+ Add Edit
Users	+ Add Edit

>>> network .toCode()

Thank You!

Have a great day!



>>> network.toCode()

Nautobot 1.2.0 Key Features

Special *developer-facing* overview

Tim Fiola

Developer Advocate

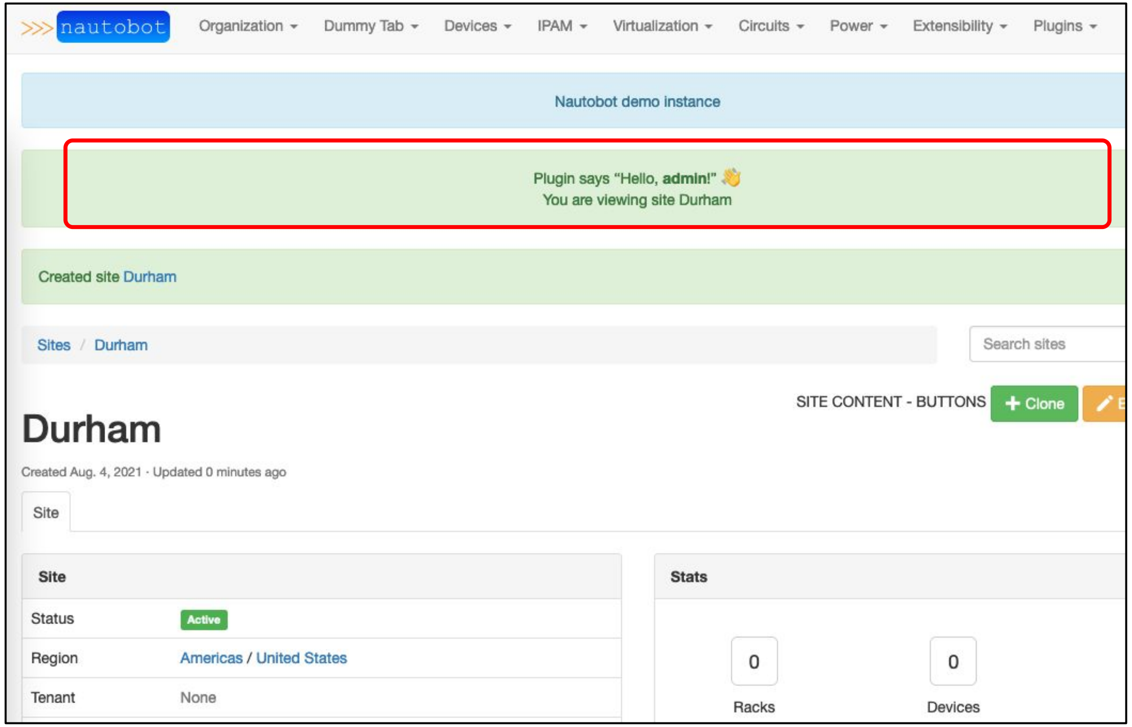
Table of Contents

Apps Can Add Banners (GH534)

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



Nautobot demo instance

Plugin says "Hello, **admin!**" 👋
You are viewing site Durham

Created site Durham

Sites / Durham

Durham

Created Aug. 4, 2021 · Updated 0 minutes ago

Site	
Status	Active
Region	Americas / United States
Tenant	None

Stats	
0	0
Racks	Devices

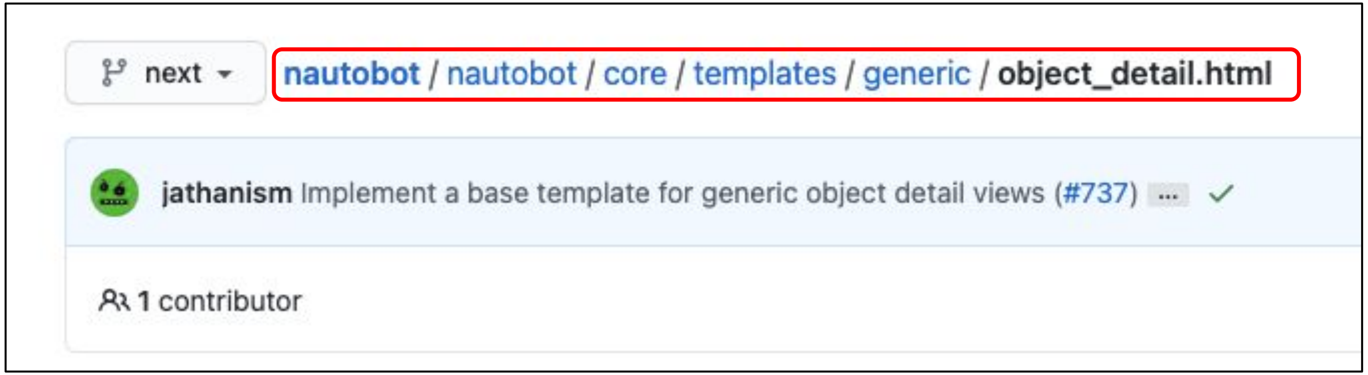
Generic Base Template File for Object Detail Views (GH479)

Almost every detail page template has the same format

Nautobot 1.2.0 creates a generic detail view template

Nautobot app developers can use this consolidated, common detail view template as well

Additionally, adding this common template simplifies the code base, replacing multiple, similar detail view templates



next

nautobot / nautobot / core / templates / generic / object_detail.html

jathanism Implement a base template for generic object detail views (#737) … ✓

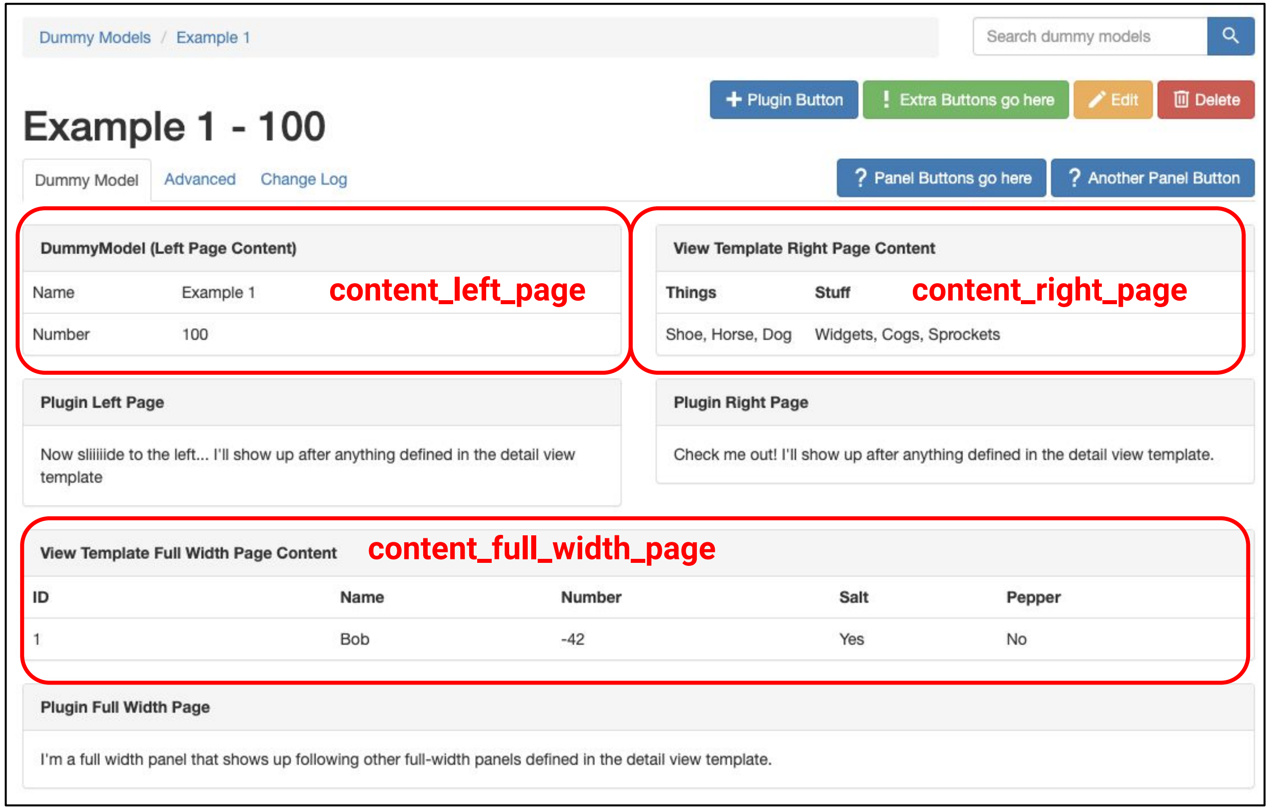
1 contributor

>>> Generic Base Template File for Object Detail Views (GH479)

The object_detail.html template has multiple blocks that can be overloaded in the `block content` block, including

- `content_left_page`
- `content_right_page`
- `content_full_width_page`

This example shows where these panes would render relative to plugin-defined sections in a detail view for a plugin-defined model



Example 1 - 100

DummyModel (Left Page Content)

Name	Example 1
Number	100

View Template Right Page Content

Things	Stuff
Shoe, Horse, Dog	Widgets, Cogs, Sprockets

Plugin Left Page

Now sliiiiide to the left... I'll show up after anything defined in the detail view template

Plugin Right Page

Check me out! I'll show up after anything defined in the detail view template.

View Template Full Width Page Content

ID	Name	Number	Salt	Pepper
1	Bob	-42	Yes	No

Plugin Full Width Page

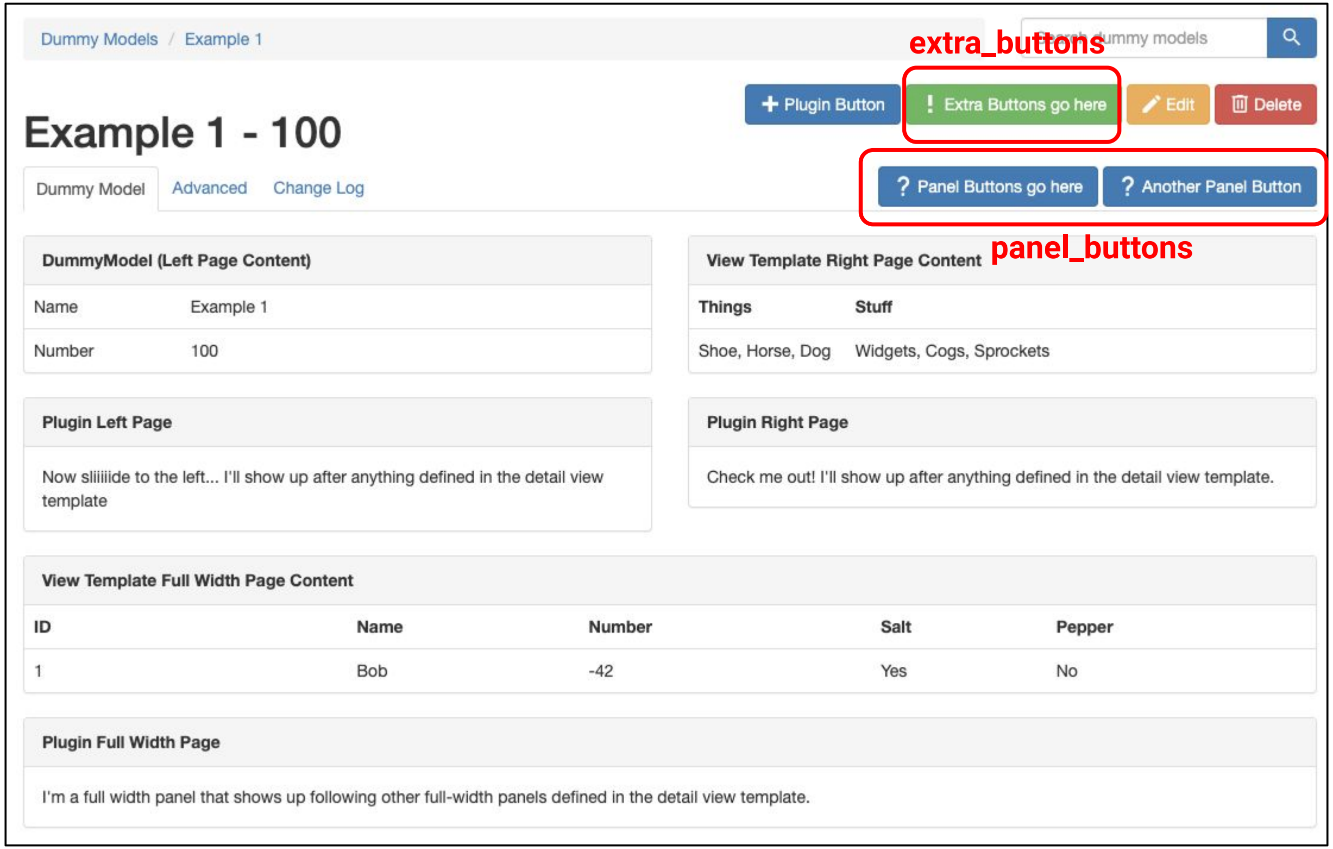
I'm a full width panel that shows up following other full-width panels defined in the detail view template.

>>> Generic Base Template File for Object Detail Views (GH479)

The object_detail.html template has multiple blocks that can be overloaded in the `block buttons` block, including

- `extra_buttons`
- `panel_buttons`

This example shows where the buttons defined in the `extra_buttons` and `panel_buttons` blocks would render relative to plugin-defined buttons and the `edit` and `delete` buttons defined in the object_detail template



Example 1 - 100

DummyModel (Left Page Content)

Name	Example 1
Number	100

View Template Right Page Content

Things	Stuff
Shoe, Horse, Dog	Widgets, Cogs, Sprockets

Plugin Left Page

Now sliiiiide to the left... I'll show up after anything defined in the detail view template

Plugin Right Page

Check me out! I'll show up after anything defined in the detail view template.

View Template Full Width Page Content

ID	Name	Number	Salt	Pepper
1	Bob	-42	Yes	No

Plugin Full Width Page

I'm a full width panel that shows up following other full-width panels defined in the detail view template.

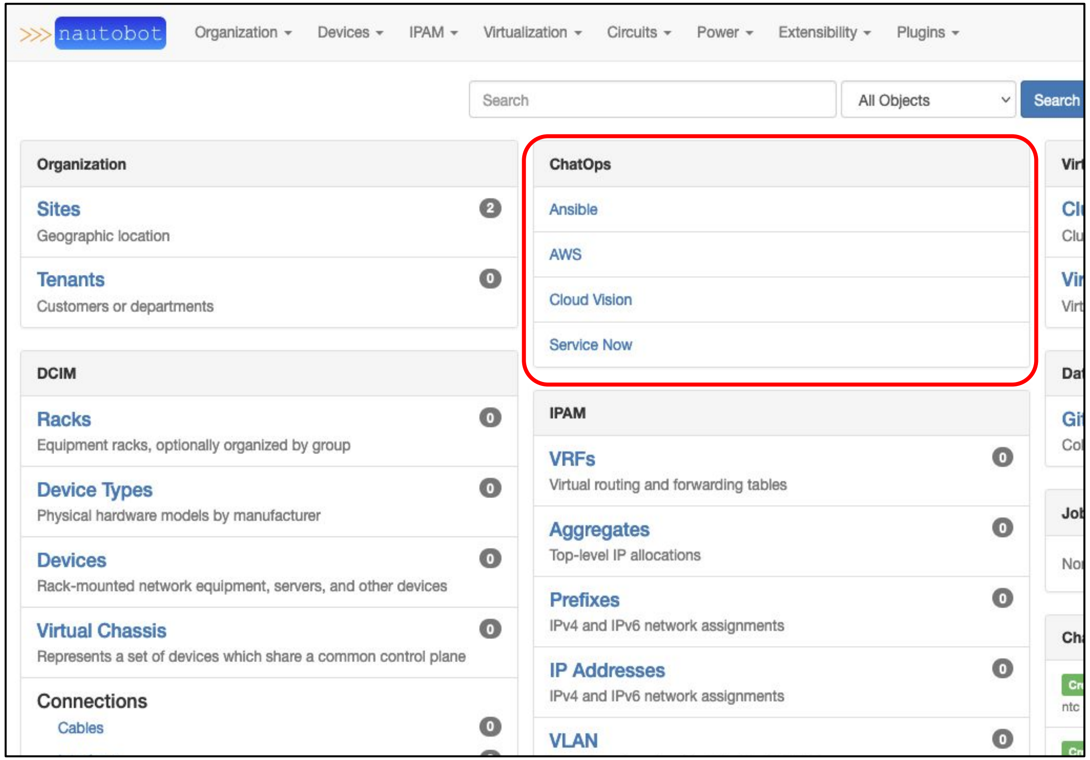
>>> Apps Can Add Content Panels to the Nautobot Home Page (GH546)

Nautobot apps can add custom content panels on the **Nautobot home page**

This allows users to prominently display important app capabilities

Examples

- Add ChatOps data to the homepage (shown)
- Add last devices automated
- Add devices that are failing to respond to tooling



Organization

- **Sites** 2 — Geographic location
- **Tenants** 0 — Customers or departments

DCIM

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

ChatOps

- Ansible
- AWS
- Cloud Vision
- Service Now

IPAM

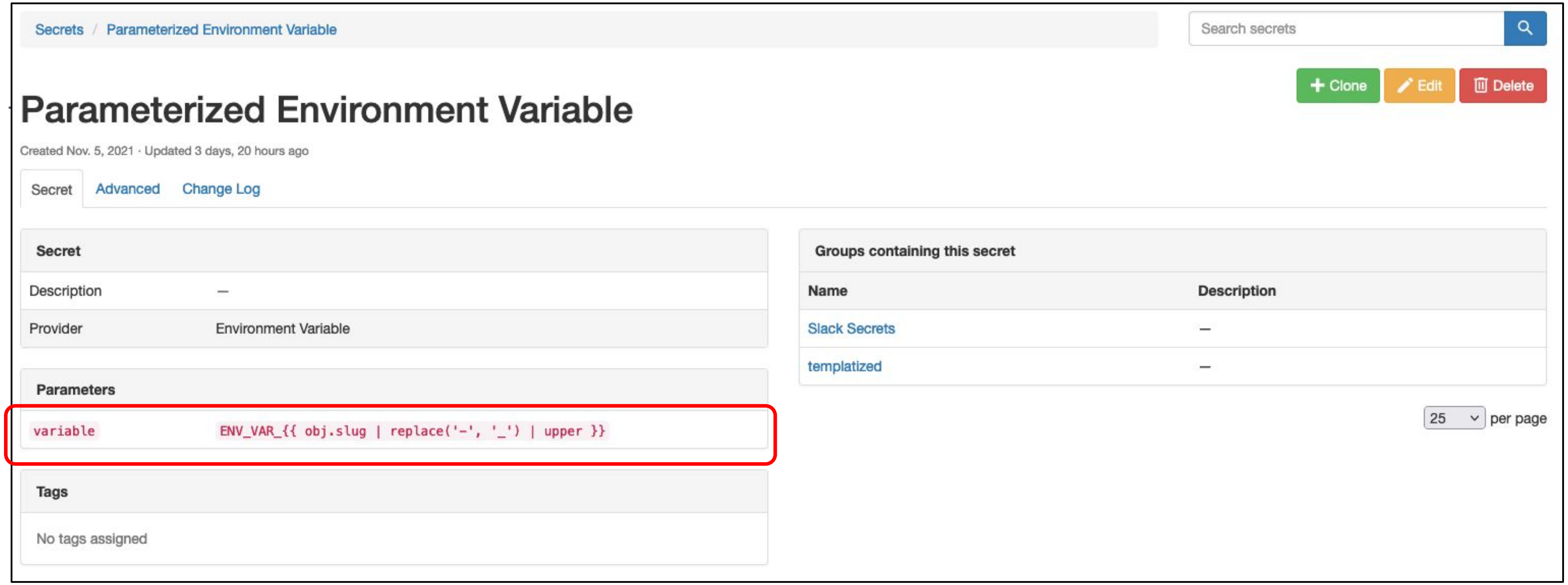
- **VRFs** 0 — Virtual routing and forwarding tables
- **Aggregates** 0 — Top-level IP allocations
- **Prefixes** 0 — IPv4 and IPv6 network assignments
- **IP Addresses** 0 — IPv4 and IPv6 network assignments
- **VLAN** 0

Secrets Integration (GH541)

Allows for use of dynamically-defined *parameterized* environment variables

Environment variable names can be defined using Jinja templates

- When secret is retrieved, context can be passed



Parameterized Environment Variable

Created Nov. 5, 2021 · Updated 3 days, 20 hours ago

Secret | Advanced | Change Log

Secret

Description	—
Provider	Environment Variable

Parameters

`variable`	`ENV_VAR_{{ obj.slug | replace('-', '_') | upper }}`

Tags

No tags assigned

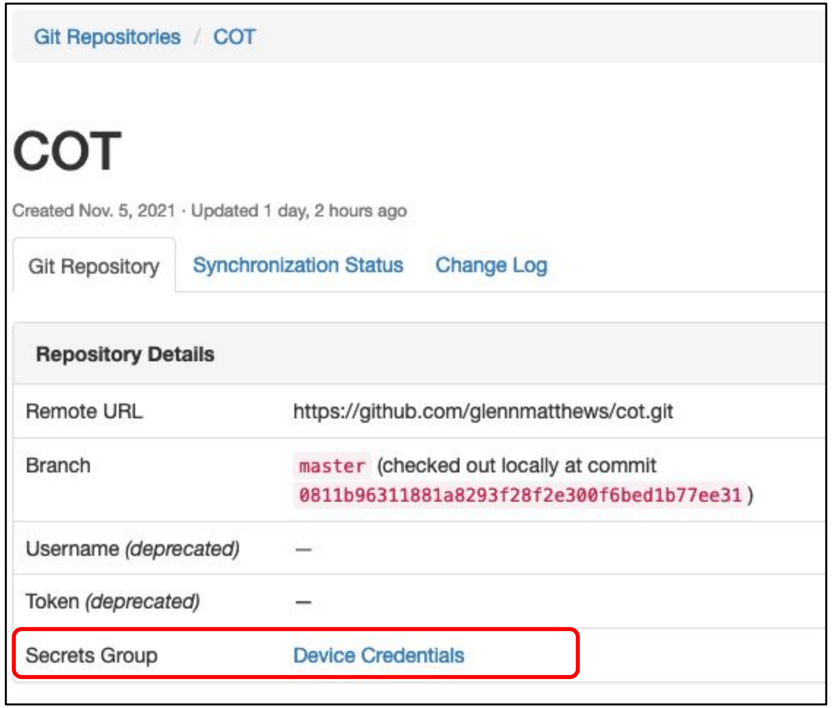
Groups containing this secret

Name	Description
Slack Secrets	—
templatized	—

25 per page

Secrets Integration (GH541)

Devices and Git Repositories can now use *Secrets Groups* models:


Git Repositories / COT

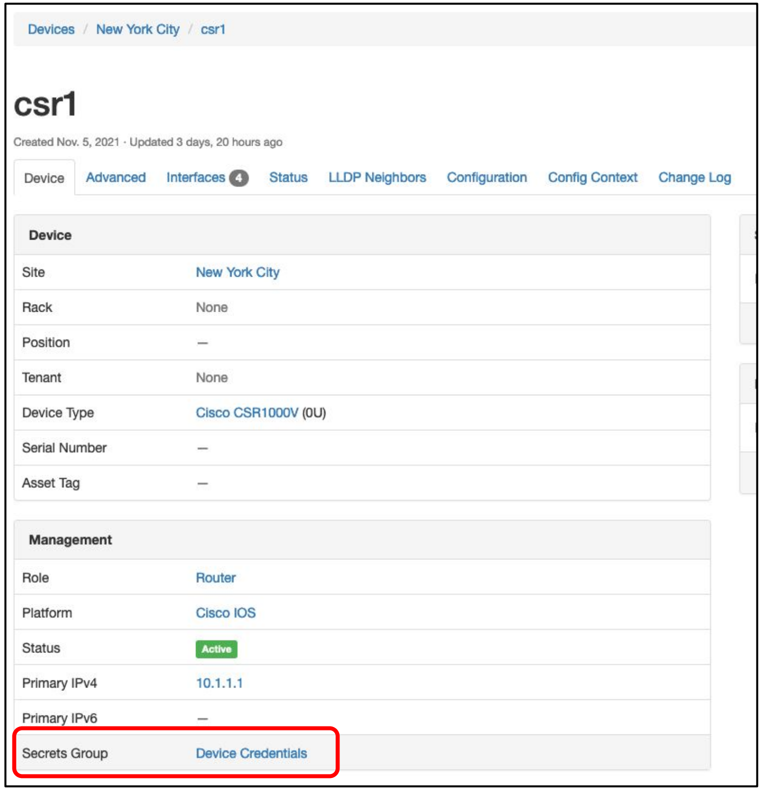
COT

Created Nov. 5, 2021 · Updated 1 day, 2 hours ago

Git Repository | Synchronization Status | Change Log

Repository Details

Remote URL	https://github.com/glennmatthews/cot.git
Branch	`master` (checked out locally at commit `0811b96311881a8293f28f2e300f6bed1b77ee31`)
Username *(deprecated)*	—
Token *(deprecated)*	—
Secrets Group	Device Credentials


Devices / New York City / csr1

csr1

Created Nov. 5, 2021 · Updated 3 days, 20 hours ago

Device | Advanced | Interfaces 4 | Status | LLDP Neighbors | Configuration | Config Context | Change Log

Device	
Site	New York City
Rack	None
Position	—
Tenant	None
Device Type	Cisco CSR1000V (0U)
Serial Number	—
Asset Tag	—

Management	
Role	Router
Platform	Cisco IOS
Status	Active
Primary IPv4	10.1.1.1
Primary IPv6	—
Secrets Group	Device Credentials

Database ready signal (GH13)

Introduces new `nautobot_database_ready` signal

- Triggered when running `nautobot-server migrate` or `nautobot-server post_upgrade`
- Upon a Nautobot install/upgrade, this signal is designed for apps to connect to, in order to perform automatic database population, including
    - Custom fields
    - Relationships
    - Webhooks
- Example: it can be useful for a plugin to automatically create a *custom field* or *relationship* as a result of being installed and activated

Complex GraphQL Query Optimization (GH171)

Complex GraphQL queries are optimized

- Performance improvement via integration with **`graphene-django-optimizer`**
- Greatly reduces number of SQL queries generated per GraphQL query
- Designed to improve user experience and improve performance of automated queries

>>> IPAM custom lookups for filtering on REST API Calls (GH982)

Nautobot again supports custom lookup filters on the *IPAddress*, *Prefix*, and *Aggregate* models, such as address__net_contained, network__net_contains_or_equals, etc

Examples:

- network__net_contained="192.0.0.0/8" would include 192.168.0.0/24 in the result
- network__net_contained_or_equal="192.0.0.0/8" would include 192.168.0.0/24 and 192.0.0.0/8 in the result
- network__net_contains="192.168.0.0/16" would include 192.0.0.0/8 in the result

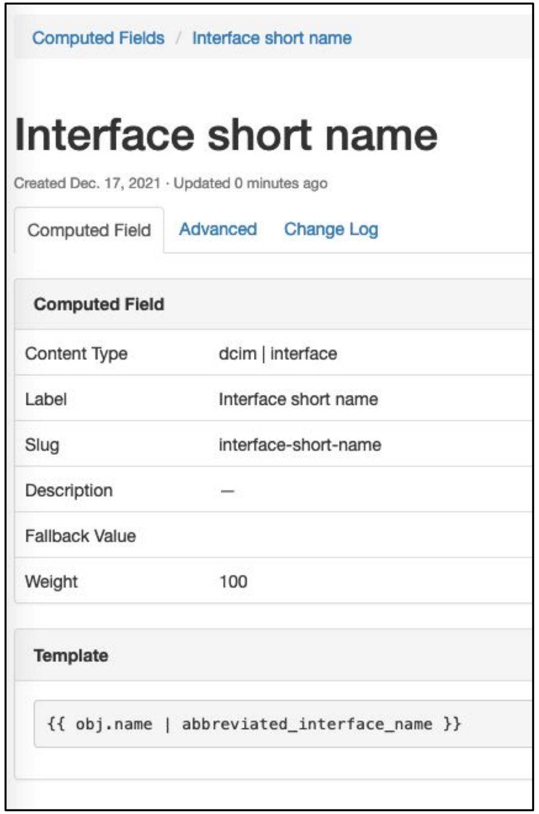
Many more examples @ https://nautobot.readthedocs.io/en/latest/rest-api/filtering/#network-and-host-fields

>>> Added netutils template filters for both Django and Jinja2 template rendering (GH1082)

netutils is NTC library of python functions to do common ops around networking (processing interface names, IP addresses, etc)

These functions are now available in any object that uses Jinja2 or Django templated output (computed fields, export templates, page templates)

This example shows a computed field using netutils' abbreviated_interface_name filter to make the interface's short name easily accessible in the UI or REST API



Interface short name

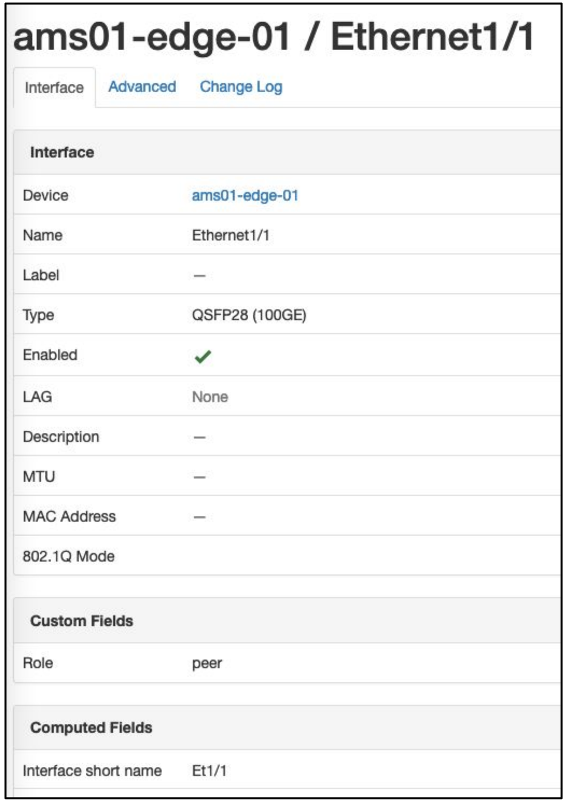
Created Dec. 17, 2021 · Updated 0 minutes ago

Computed Field

Content Type	dcim | interface
Label	Interface short name
Slug	interface-short-name
Description	—
Fallback Value	
Weight	100

Template

```
{{ obj.name | abbreviated_interface_name }}
```



ams01-edge-01 / Ethernet1/1

Interface

Device	ams01-edge-01
Name	Ethernet1/1
Label	—
Type	QSFP28 (100GE)
Enabled	✓
LAG	None
Description	—
MTU	—
MAC Address	—
802.1Q Mode	

Custom Fields

Role	peer

Computed Fields

Interface short name	Et1/1

>>> network .toCode()

Thank You!

Have a great day!