Juniper Apstra Freeform概要 & 簡易オペレーションガイド

2023/8/4 updated

JUNIPER NETWORKS | Driven by Experience™

CONFIDENTIALITY AND LEGAL NOTICE

This material contains information that is confidential and proprietary to Juniper Networks, Inc. Recipient may not distribute, copy, or repeat information in the document.

This statement of product direction sets forth Juniper Networks' current intention and is subject to change at any time without notice. No purchases are contingent upon Juniper Networks delivering any feature or functionality depicted in this presentation.

subject to a license agreement that describes program terms and conditions.

本資料は融資でベストエフォートで記載している資料となります。
内容に不備がある場合はご了承ください。
最新の状況などは公式のマニュアルをご確認ください。
また、内容は予告なしに変更になる場合があります。

はじめに

本資料はJuniper ApstraのFreeformのネットワーク構築手順を概要をまとめたものです。

内容は予告なく変更する可能性もあり、また都度修正させていただく可能性があります。
本資料のオペレーション部分は参考資料としてご活用ください。

不明点はJuniper Networks、またはパートナー様にご連絡いただくか、
Apstraのマニュアルを参照下さい。

Juniper Apstra 4.2 User Guide
Freeform Introduction
https://www.juniper.net/documentation/us/en/software/apstra4.2/apstra-user-guide/topics/concept/freeform.html

- Apstra Freeform概要
- Apstra Freeformオペレーション概要
  - 初期セットアップ手順
  - Config Templates
  - Apstra Telemetry
  - オプション機能
- 参考リンク

APSTRAの2つのデザインパターン

Juniper Apstra

Best of Everything

DC Reference Architecture

容易な自動化を
ベストプラクティスリファレンスと共に

Freeform Architecture

New

すべてのDC環境の自動化に

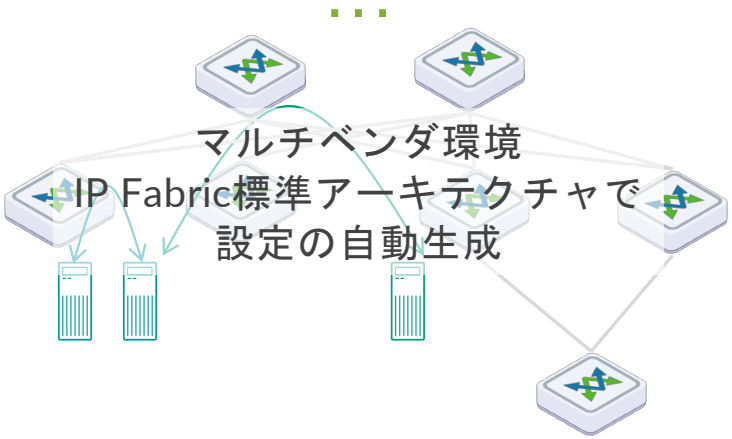
APSTRAの2つのデザインパターン

Juniper Apstra

Best of Everything

DC Reference Architecture

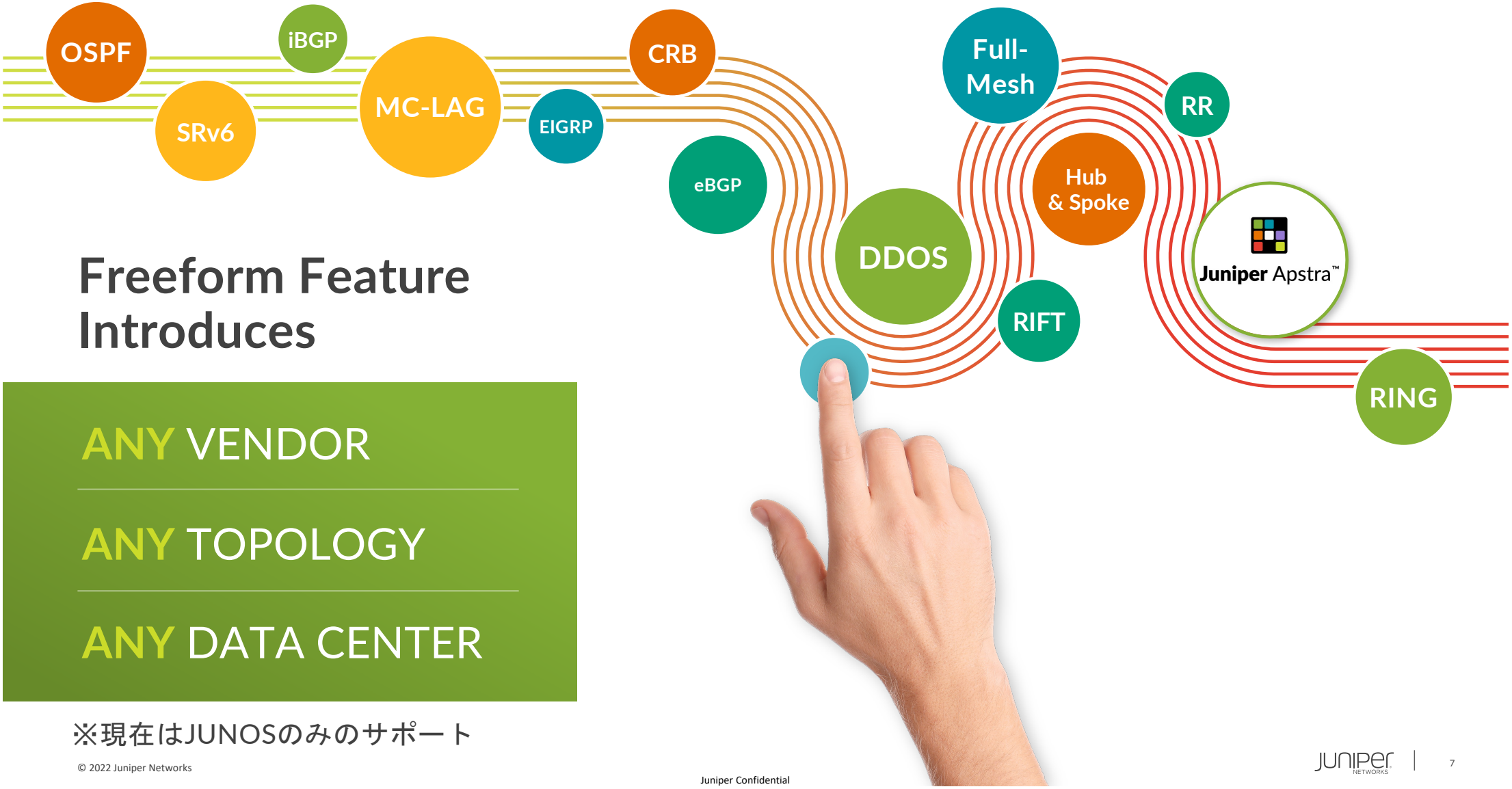
データセンタネットワーク向けの
ベストプラクティスアーキテクチャと
トポロジーを評価済み設定で自動化・可視化



New Freeform Architecture

すべてのトポロジーとフィーチャで
どのような設定でも
自動化・可視化基盤として利用可能

※現在はJUNOSのみ



Freeform Feature Introduces

ANY VENDOR

ANY TOPOLOGY

ANY DATA CENTER

※現在はJUNOSのみのサポート

Freeform Overview

- Network Designer & Config Templatesにより柔軟に設定を自動作成
- 可視化、ロールバック、OSアップグレード、など運用に必要な付加価値機能も利用

Network Designer

Config Templates & device context

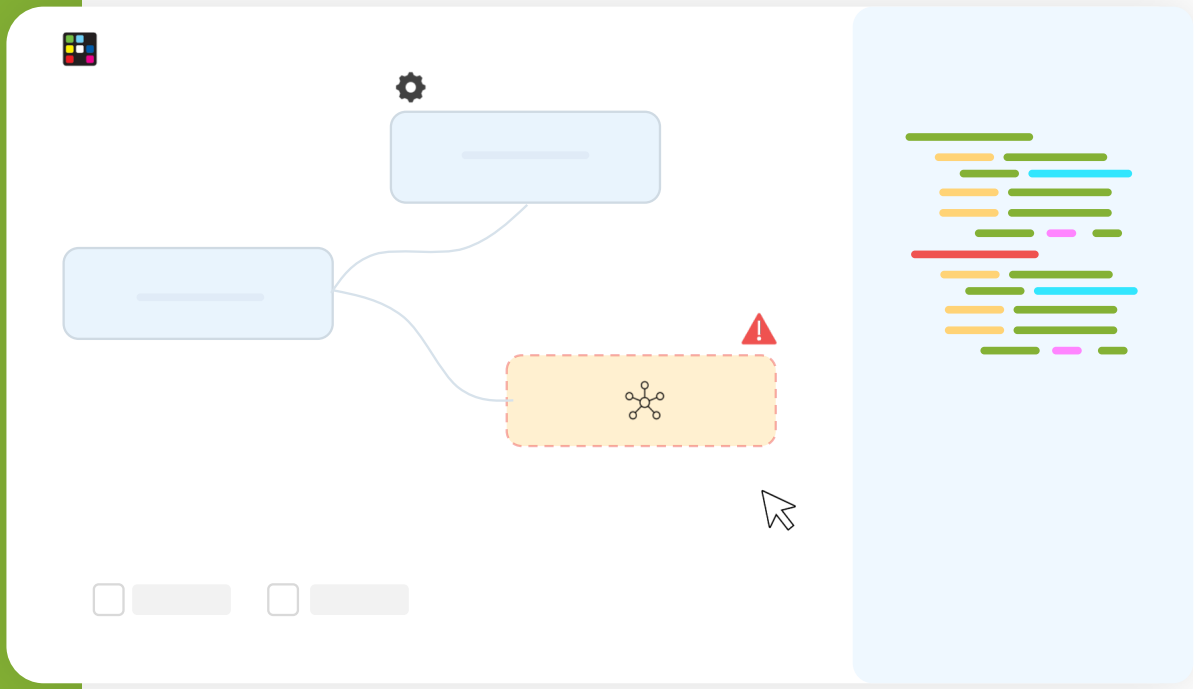
Property Sets

Closed-Loop Validation

Telemetry, IBA

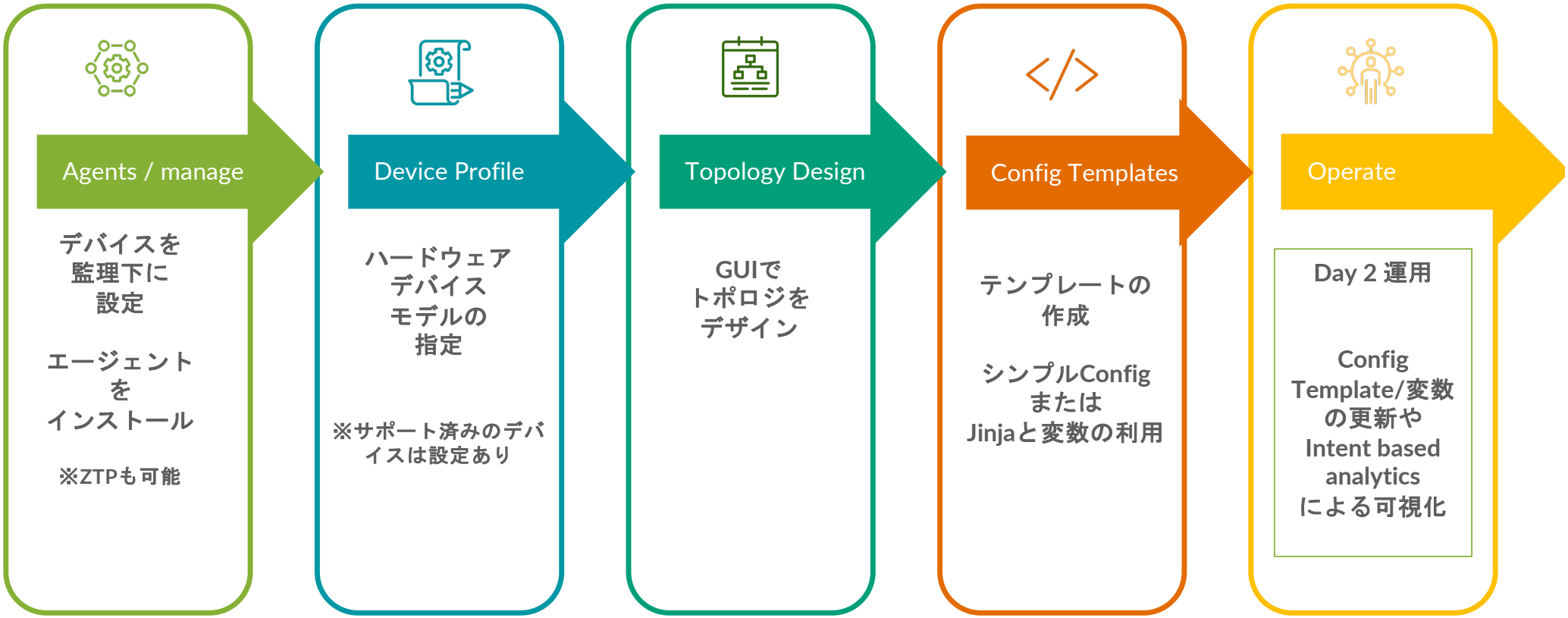
Time Voyager (Roll Back)

NOS Upgrades, ZTP,



Apstra Operation Overview - 4.1.1 – 4 min video

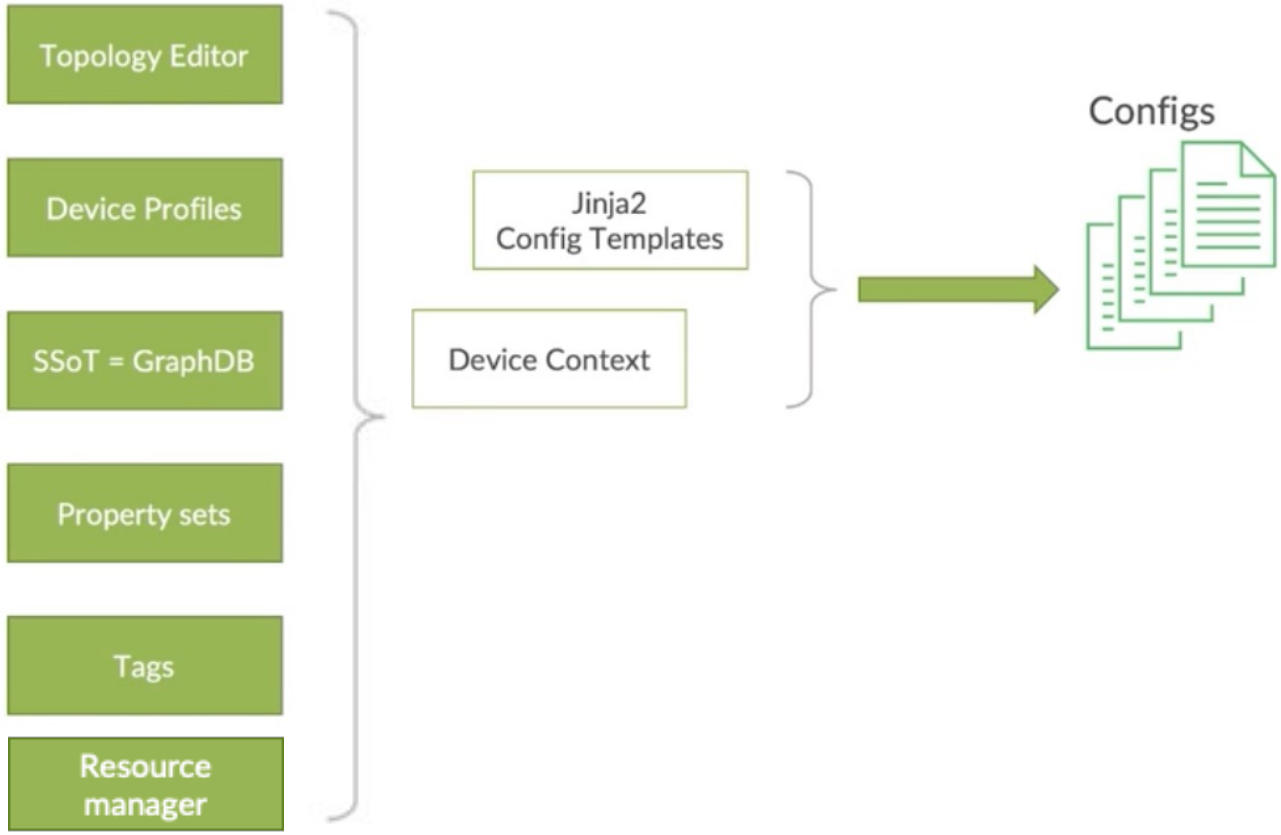
Building a Fabric with Freeform



DC ref.と同様

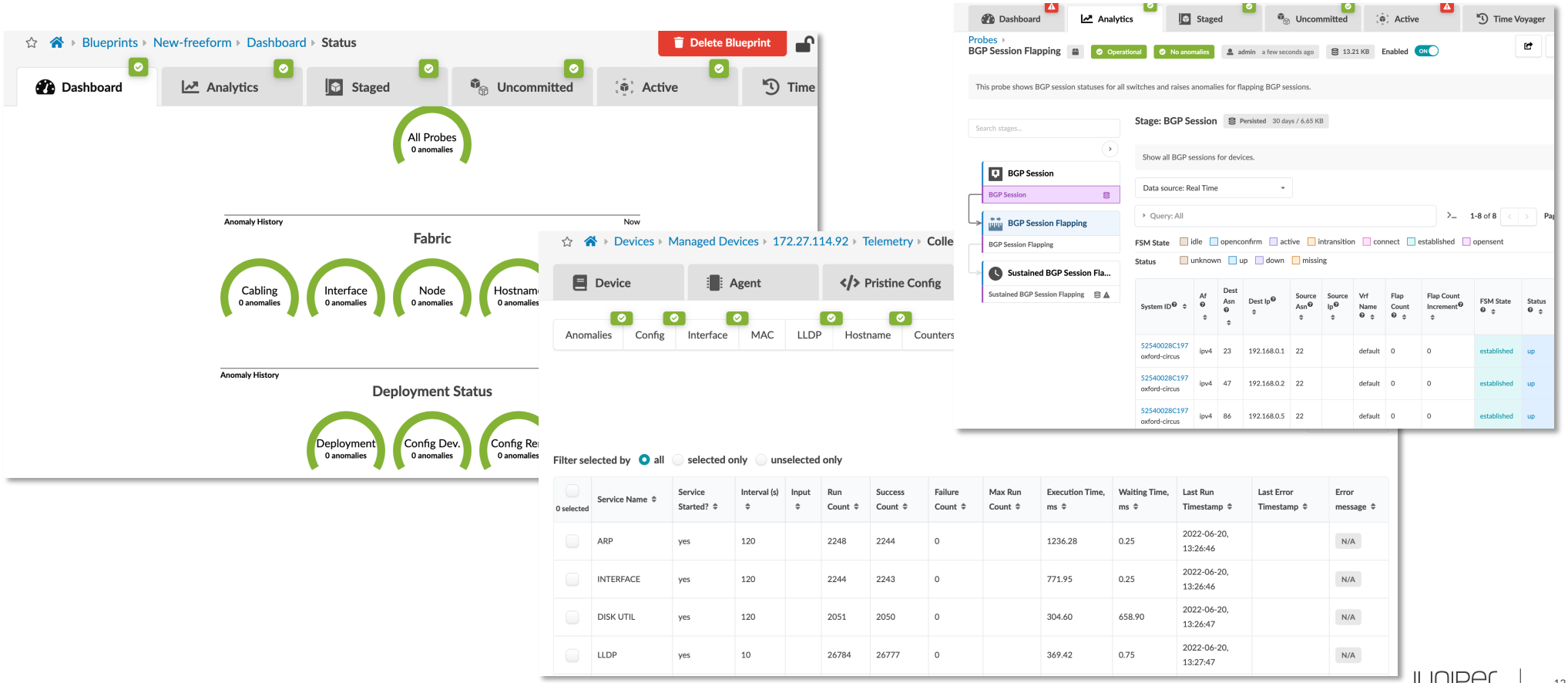
Device Context & Config Templates

- Network Designer & Config Templatesにより柔軟に設定を自動作成
- Config TemplatesはJinja 2 ベースのテンプレートを準備し、設定の自動生成を行う



Telemetry Realtime Analytics for Freeform

“Dashboard”-ネットワーク全体の健康状態の把握
”Analitics”-BGPセッション、memory, cpuなど各種各種パラメーターを取得可能



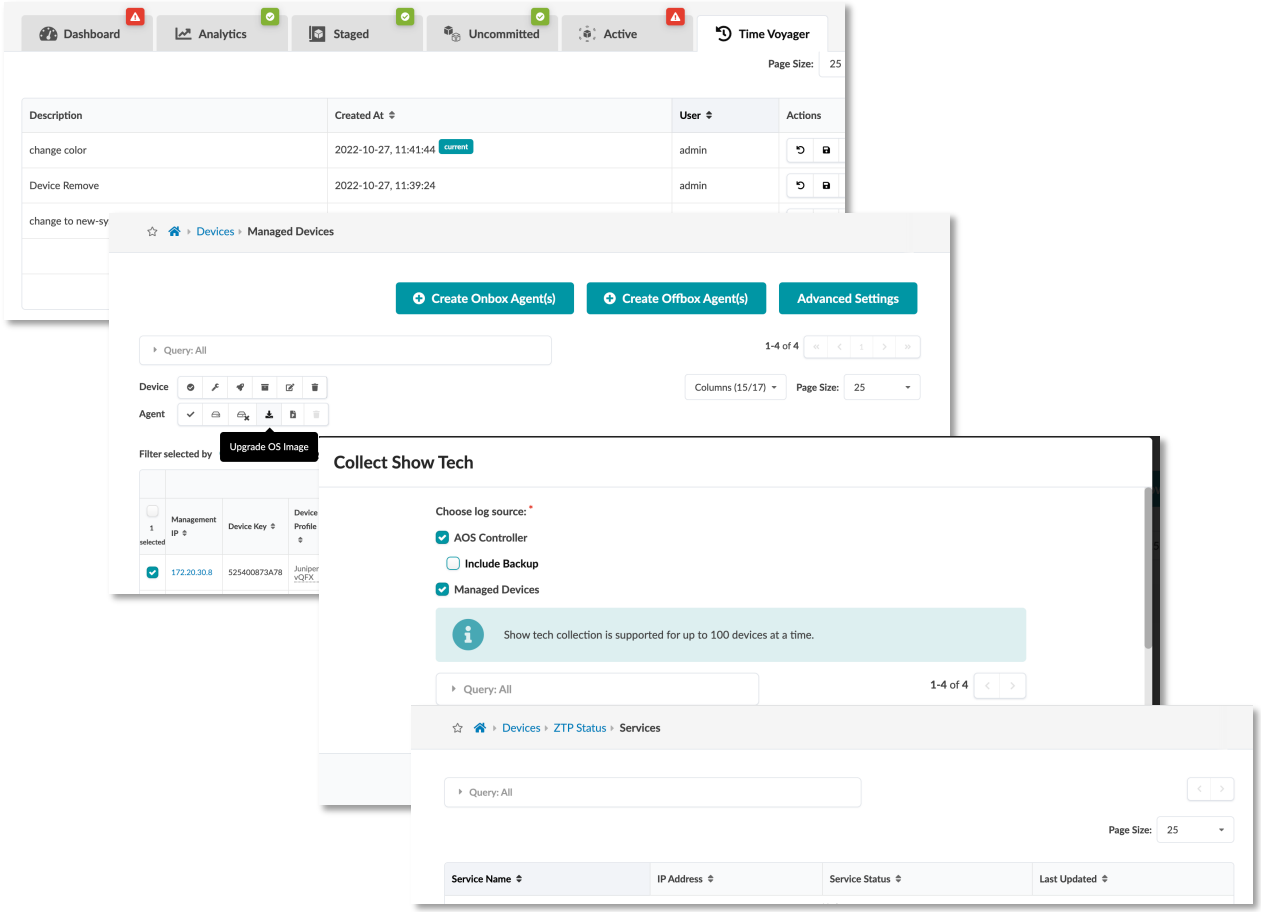
多様な運用オプション機能

Time Voyager (Roll Back)

Device OS Upgrade

Collect Show Tech

ZTP



Capabilities Across All of Apstra

1 SWITCH OR 100'S

豊富な機能	シンプル運用	革新的なオペレーション
○ Zero Touch Provisioning	○ Single software solution	○ Digital Network Designer
○ OS Upgrades	○ Vendor Agnostic ※	○ Time Voyager
○ RBAC	○ Optimized UI experience	○ Golden Config Validation(DC)
○ Metadata Tags	○ Full API programmability	○ Analytics and IBA
○ Cable Maps Exports	○ Turn-Key and Custom Designs	○ Real-Time validations

DC Reference DesignとFreeformの比較

	DC Ref. Design	Freeform
デザイン	Logical Device/Racks/Templatesから作成	トポロジーとプロトコルの制限なし（デバイス依存）
コンフィグ	Juniper/Apstraで動作保証済みの設定を自動生成	必要な設定を検討・検証し、テンプレートを作成した後に自動生成
監視	フルセット	DC Ref Designの一部 + IBA
CLI syntaxの知識	基本不要だがConfigletsには必要	必要
Jinja2の知識	Configletsには必要なことがある	必要(またはjinjaを使わず静的な設定を流し込みも可能)
Pythonの知識	APIには必要	必要な場合がある
Time Voyager	使える	使える
NOS Upgrade	できる	できる
ベンダーサポート	Juniper(JUNOS/EVO)/SONiC/NXOS/EOS	Juniperのサポート済みデバイスのみ(他はroadmap)
Brownfieldサポート	要PS相談	要PS相談(但し既存設定も利用可能)

- Apstra Freeform概要
- Apstra Freeformオペレーション概要
  - 初期セットアップ手順
  - Config Templates
  - Apstra Telemetry
  - オプション機能
- 参考リンク

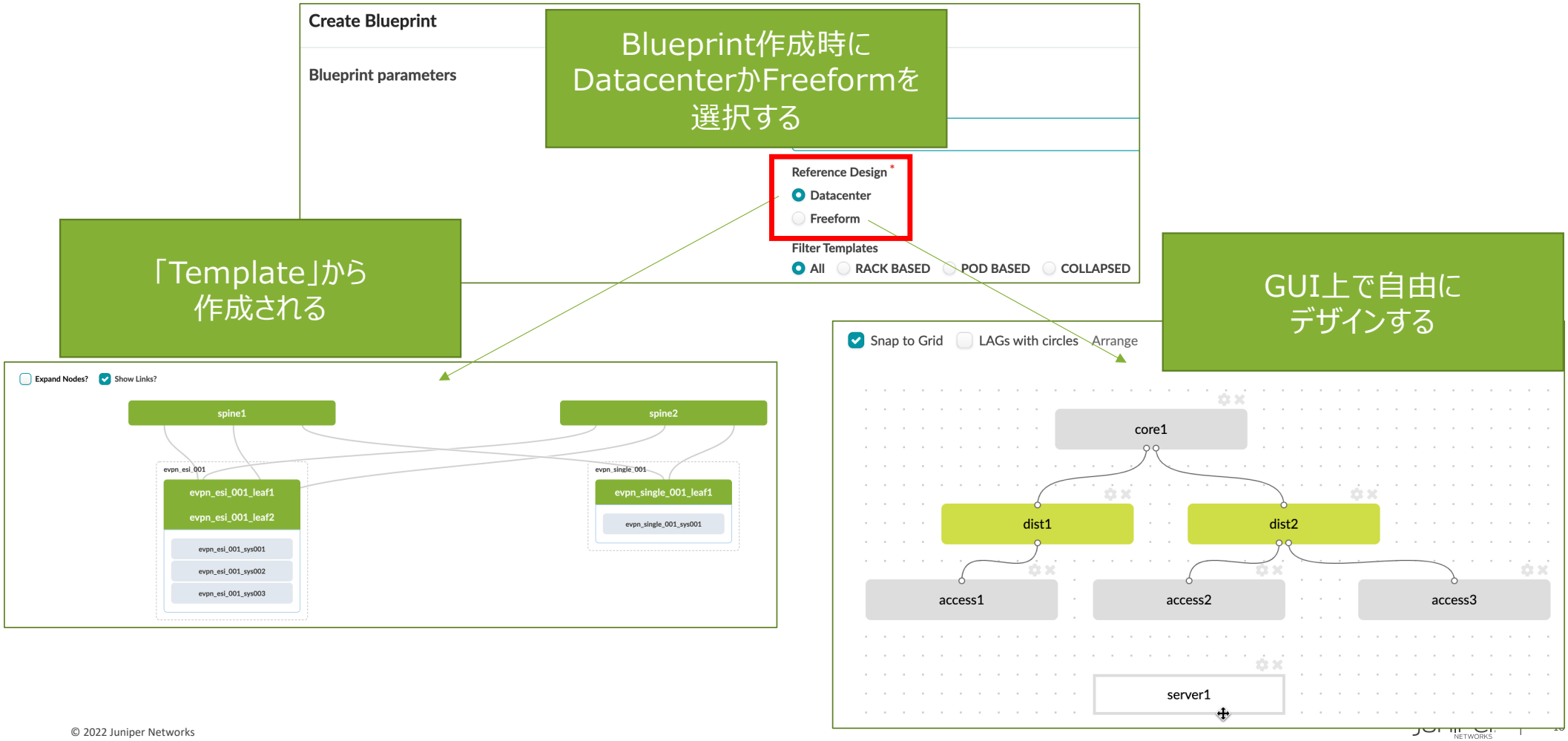
Freeform初期セットアップの流れ

■初期セットアップ

※Apstraの起動はDC Referenceと同様のため省略

- Blueprint作成時にFreeformを選択
- Managed Device登録
- System作成
- Topology作成
- Topology Link作成
- Device Profiles & System(Agent)アサイン
- Config Templatesアサイン
- Commitでデバイスに反映

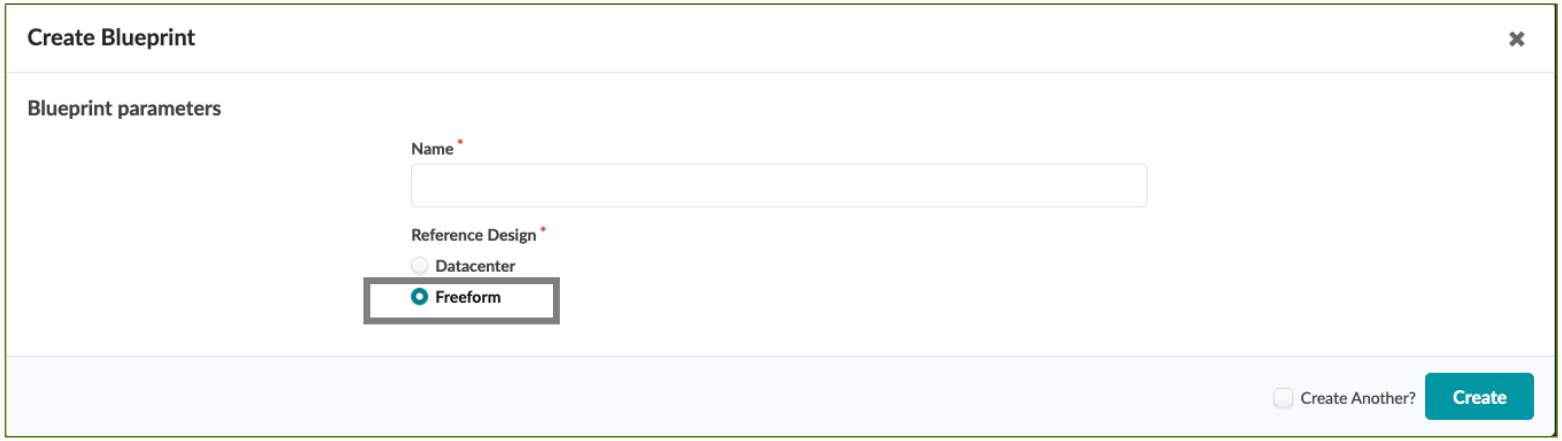
Freeform作成方法



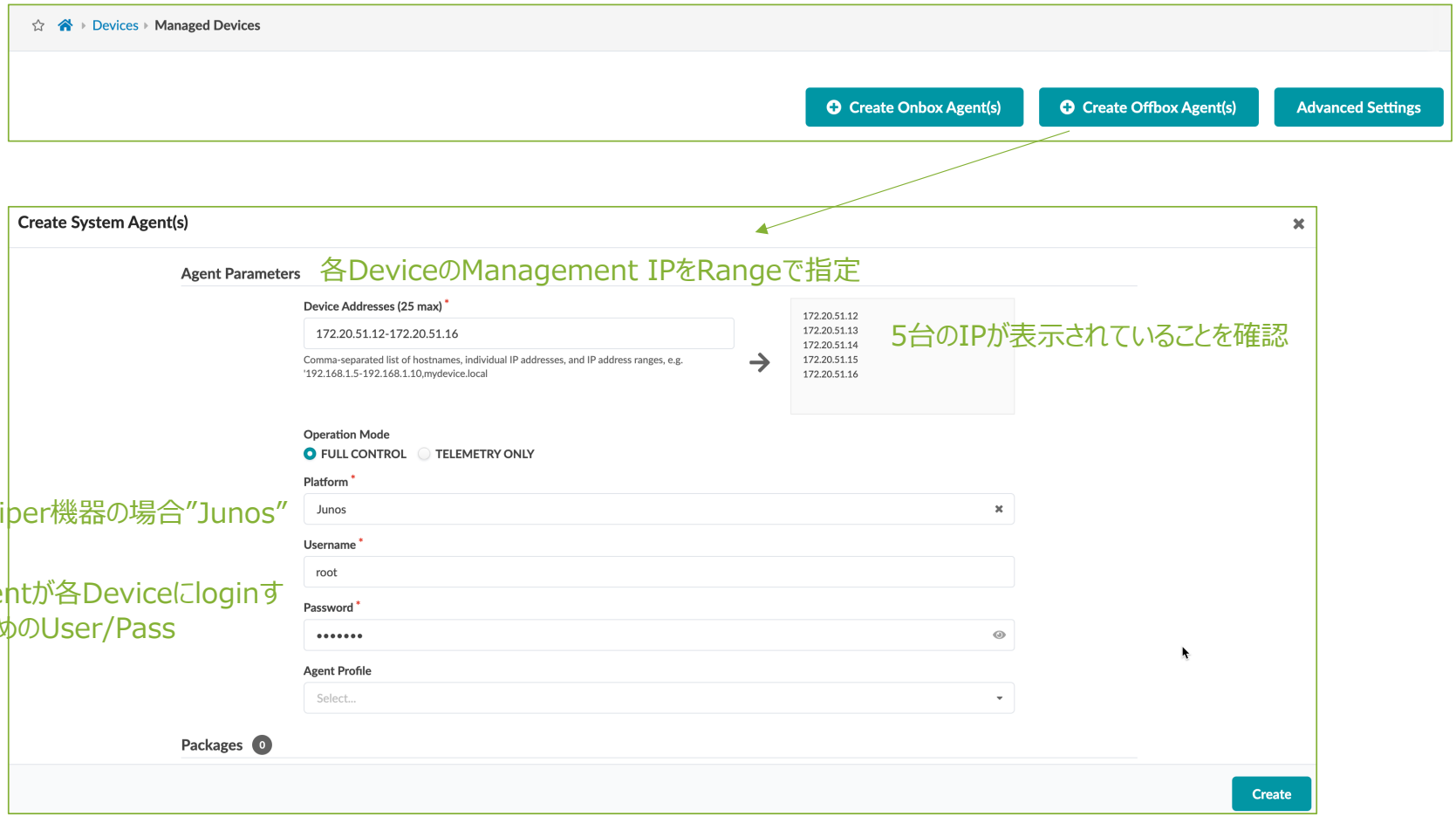
Blueprint作成時にFreeformを選択

Blueprintを作成するときにDatacenter or Freeformを選択する。

・Blueprint作成時にFreeformを選択



Managed Device登録（DC ref/freeform同様手順）



Juniper機器の場合”Junos”

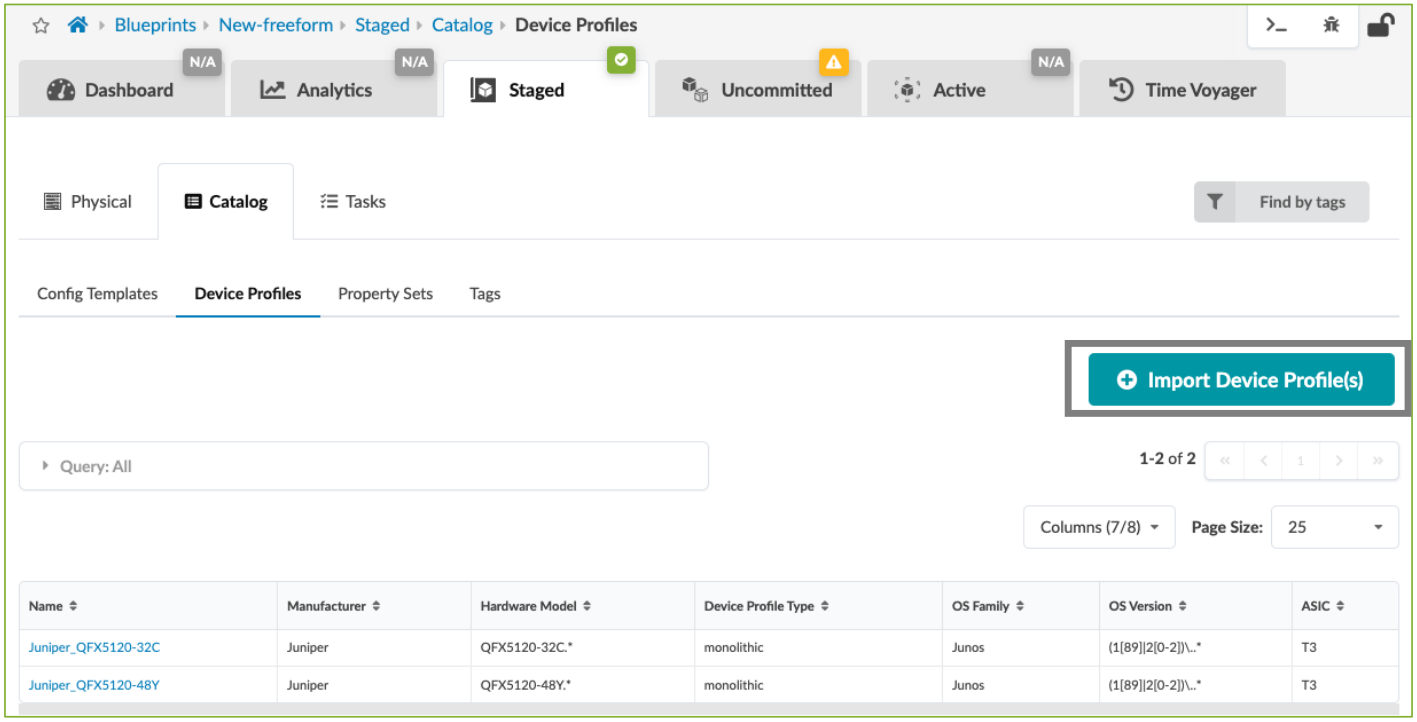
Agentが各Deviceにloginするためのuser/pass

入力完了後、Create

Device Profilesインポート

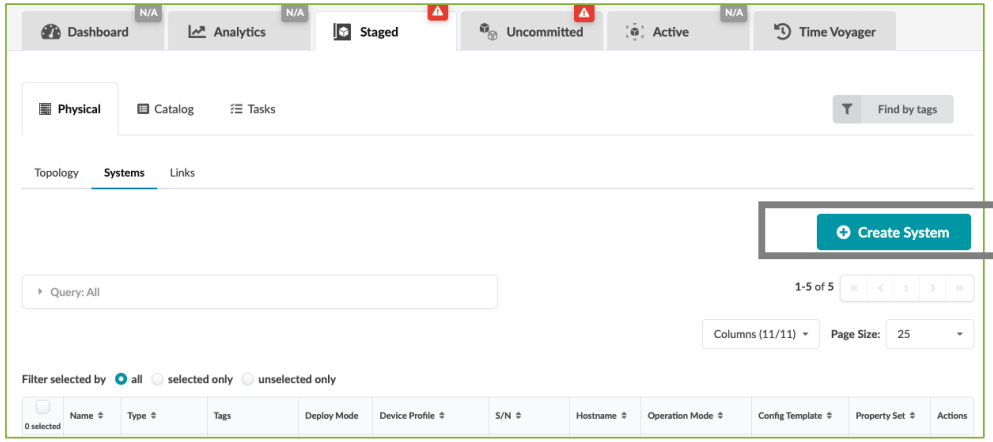
Managed Deviceとして登録していない場合、
利用するデバイスのDevice Profilesをインポート

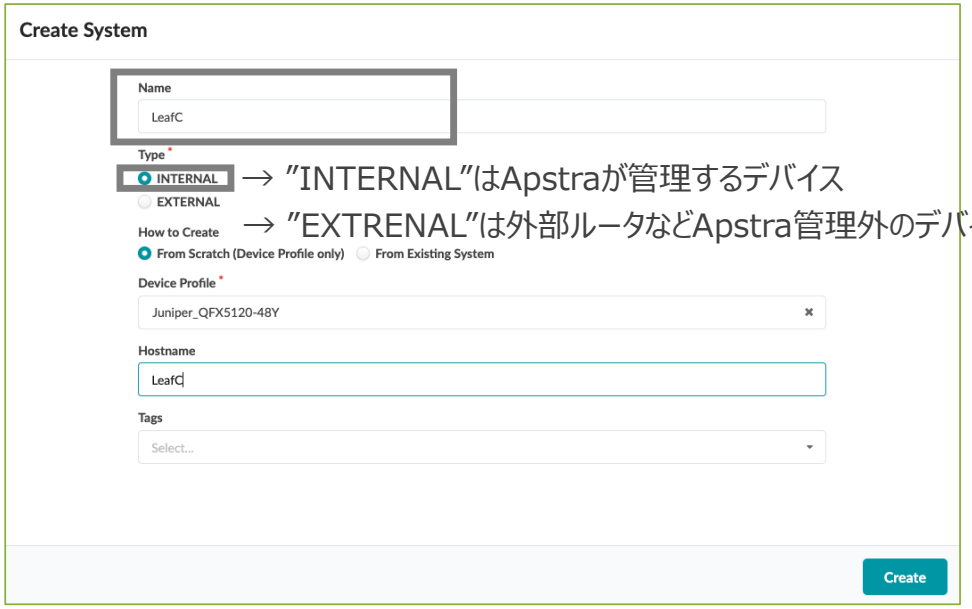
Catalog -> Device Profilesで利用するDevice Profilesをimportする



System作成

Topologyで使用する各デバイスをSystemとして作成



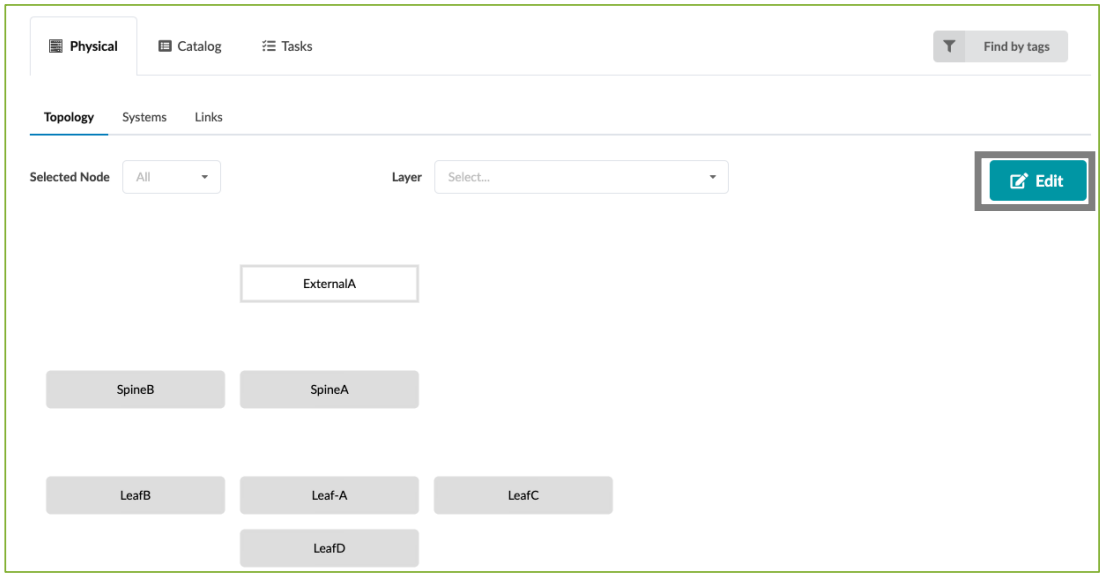


→ ”INTERNAL”はApstraが管理するデバイス

→ ”EXTRENAL”は外部ルータなどApstra管理外のデバイス

Topology 作成

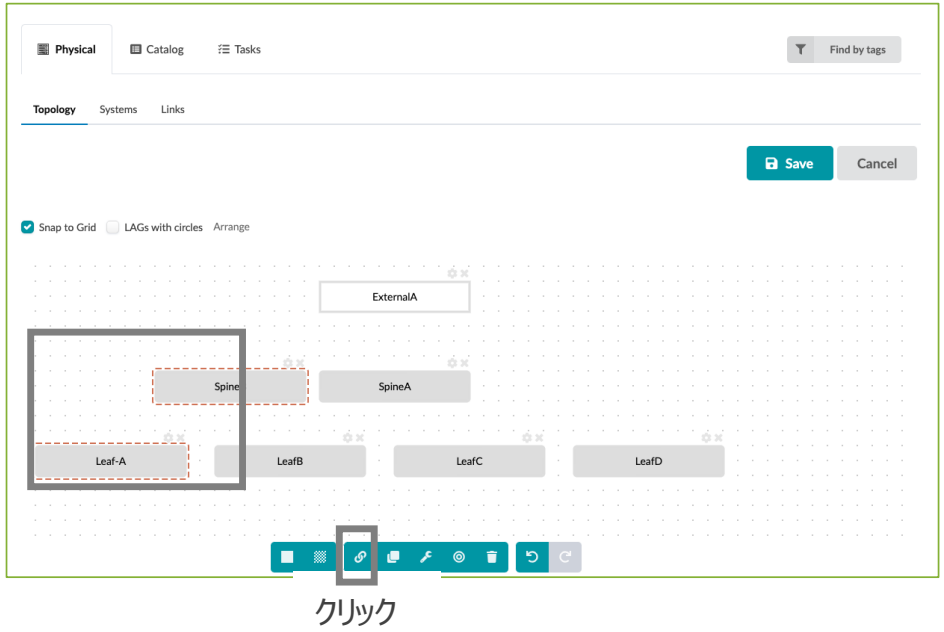
Physical -> Topology画面のEditからノードやリンクを作成可能



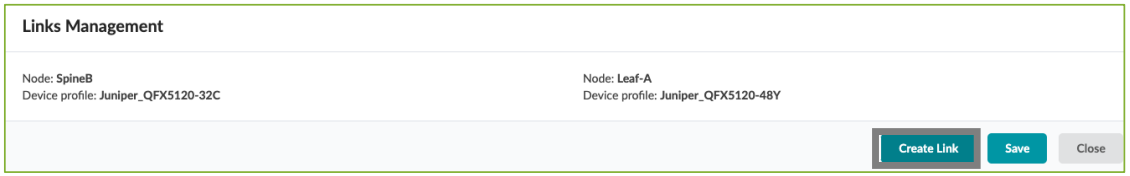
Topology Link作成

Link接続したい2つのスイッチを選択(Alt/Commandを押しながら選択するか、クリックをしながらグラブして対象を選択)、

Physical -> Topology



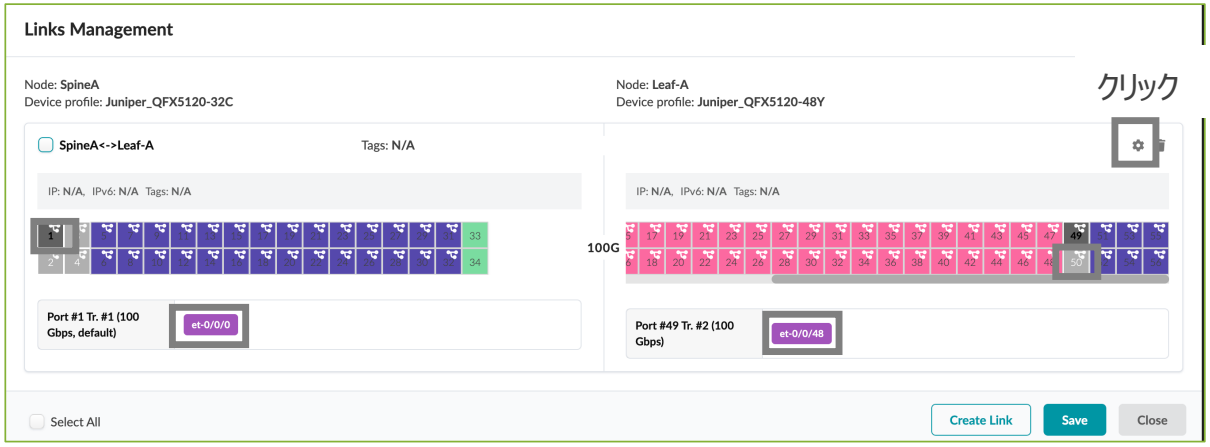
クリック

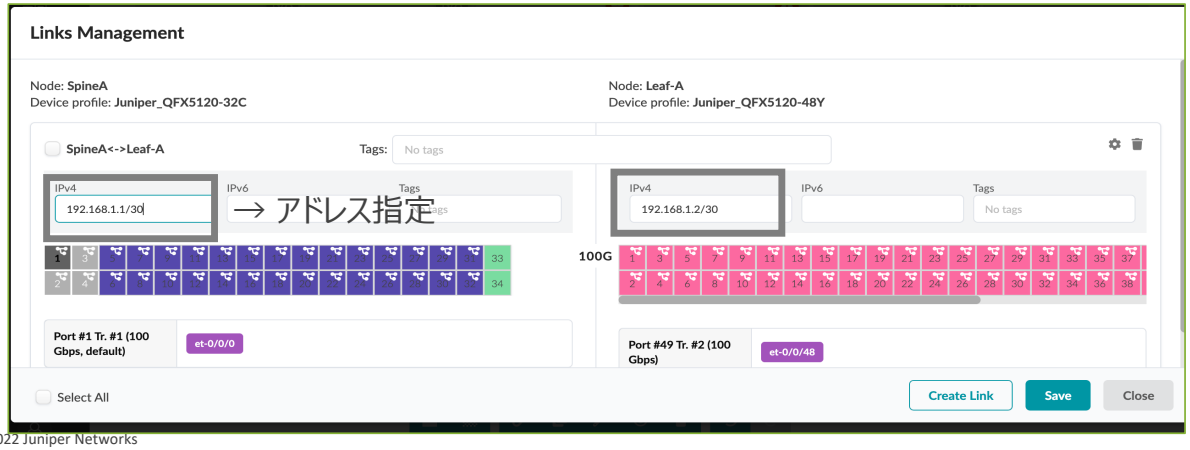


クリック

Topology Link作成

接続するポートとインタフェース名(スピード)を選択し、Create Link

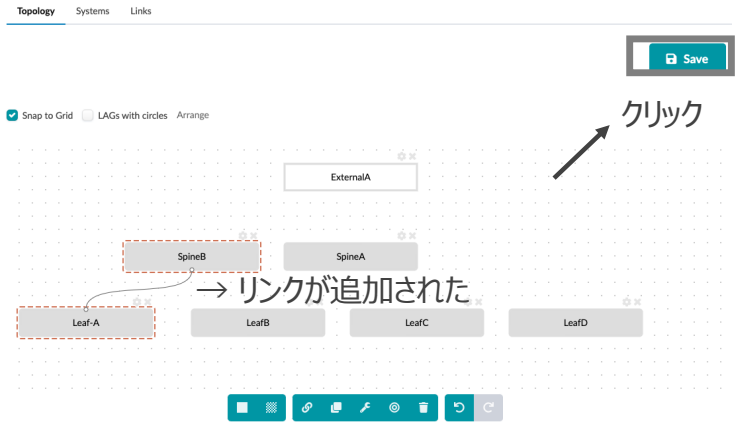




(Option)リンクのアドレス設定
これらのインタフェース名や
アドレス情報は後で利用するDevice
Contextとしてconfigテンプレートの変
数値として利用することが可能となる。

Topology Link作成

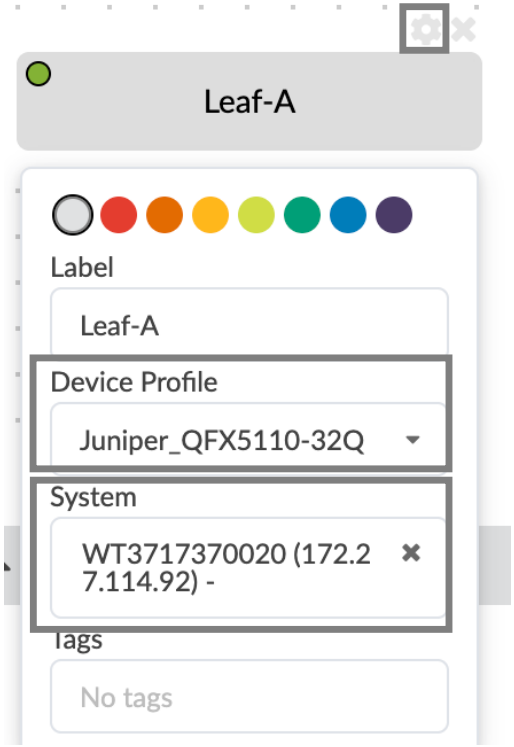
Saveで保存



同様に物理接続する各デバイスのリンクとアドレスをそれぞれ設定する

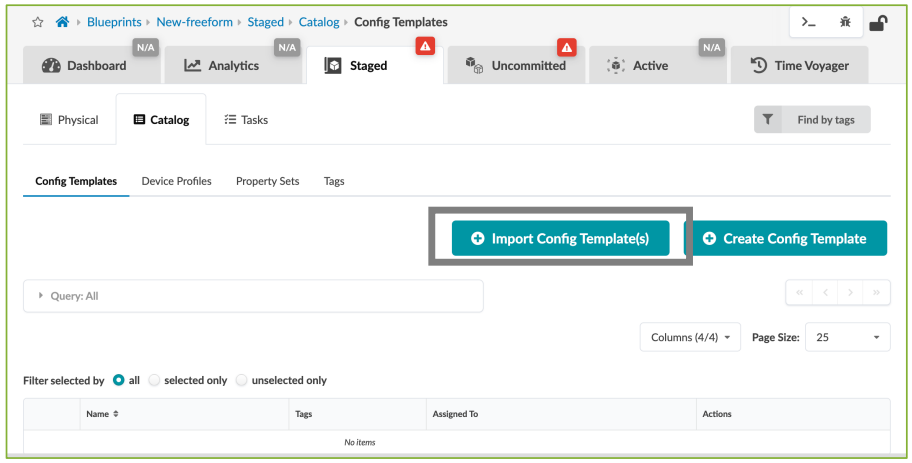
Device Profiles & System(Agent)アサイン

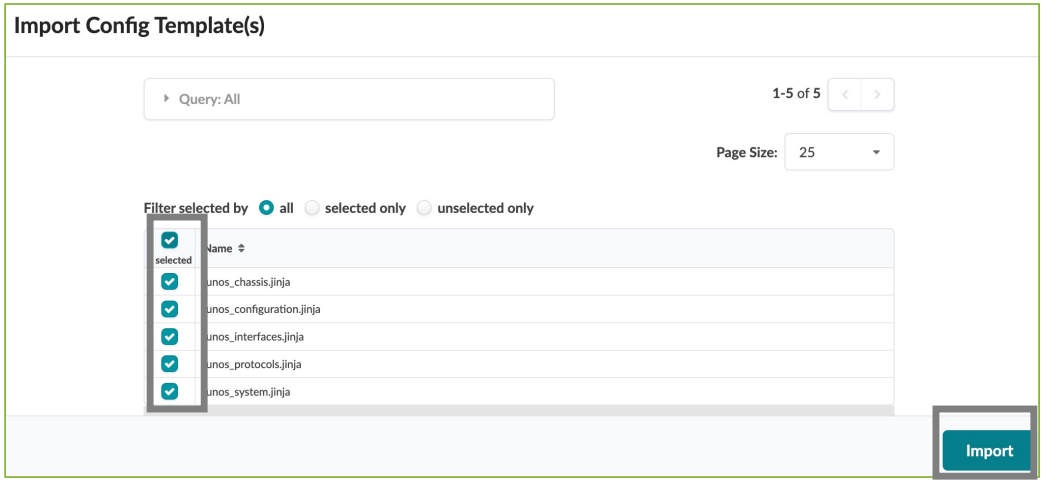
各システムにDevice ProfileとSystem(Managed Device)をアサインする。
ただし、Managed Deviceを登録していない場合は、System(Managed Device)をアサインせずに動作確認することは可能。



Config templateアサイン (import)

各デバイスに既存のconfigテンプレートをアサインする場合は、既存のテンプレートをインポートしアサインする。

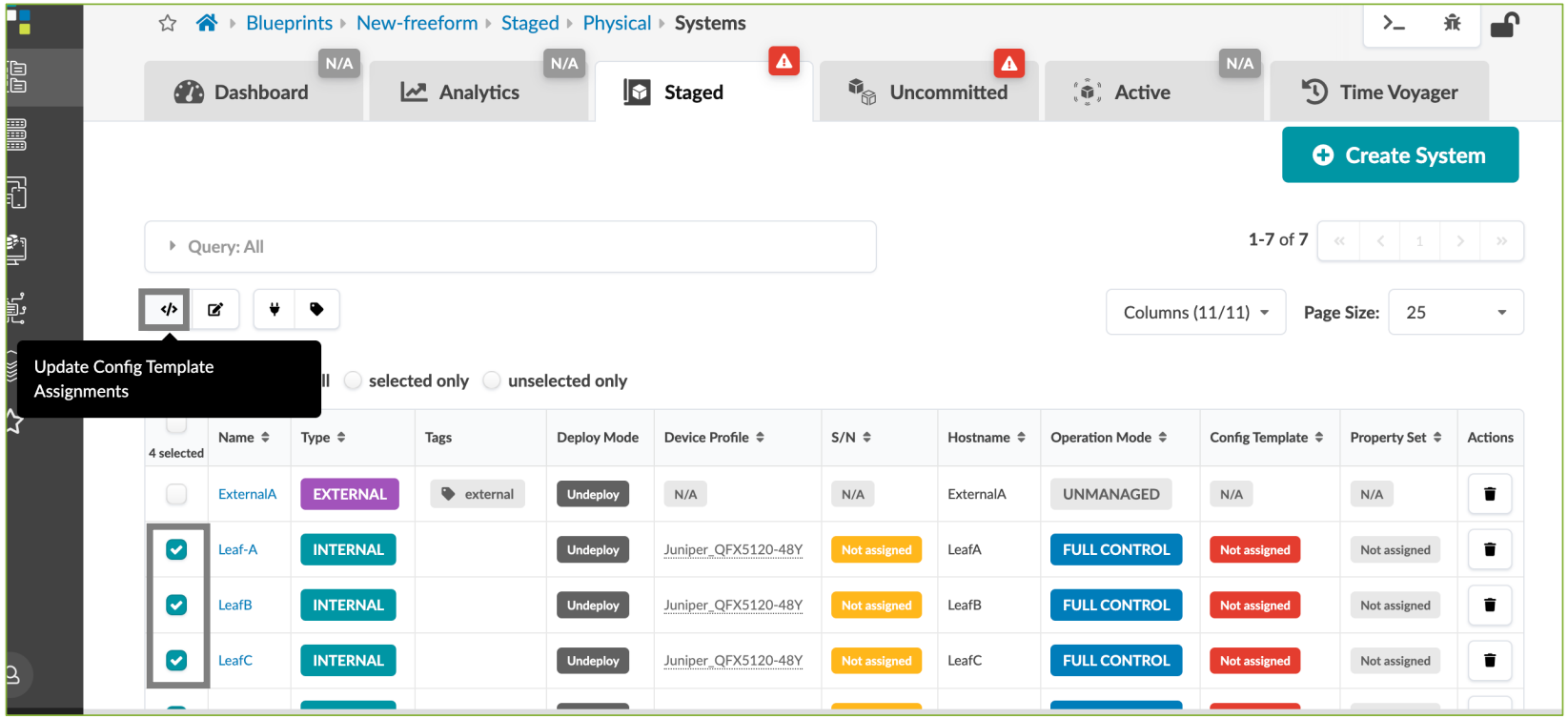




※Config templateについては別の章で解説

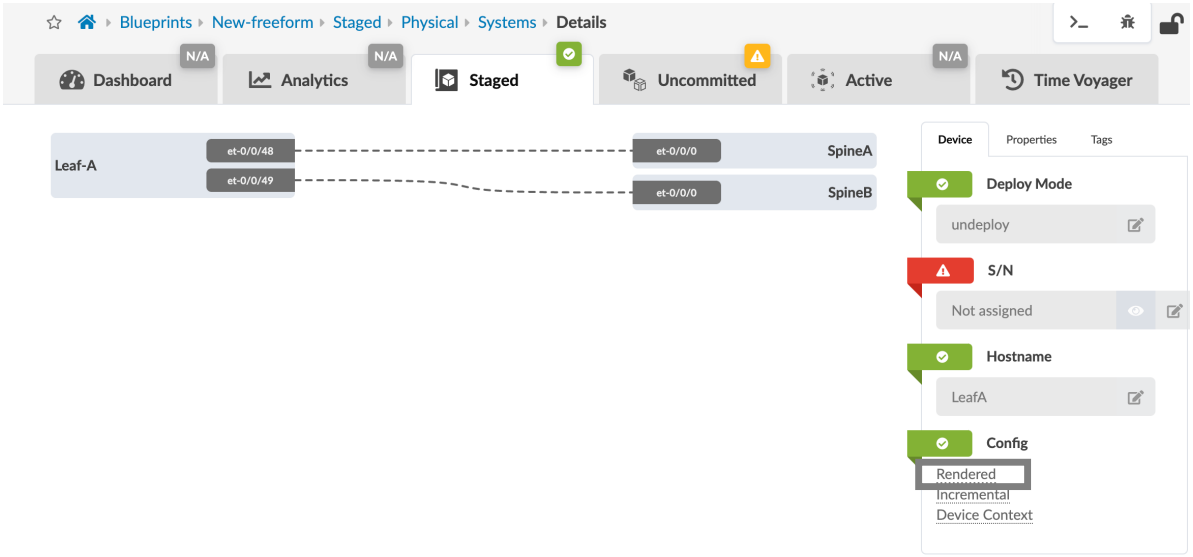
Config templateアサイン (assignment)

各デバイスにconfigテンプレートをインポートまたは作成し、Jintaテンプレートをアサインする。



Configuration 確認

各デバイスのRendered configを確認すると、テンプレートで定義した変数(interfaceやアドレス等)が補完され適用されていることがわかる。

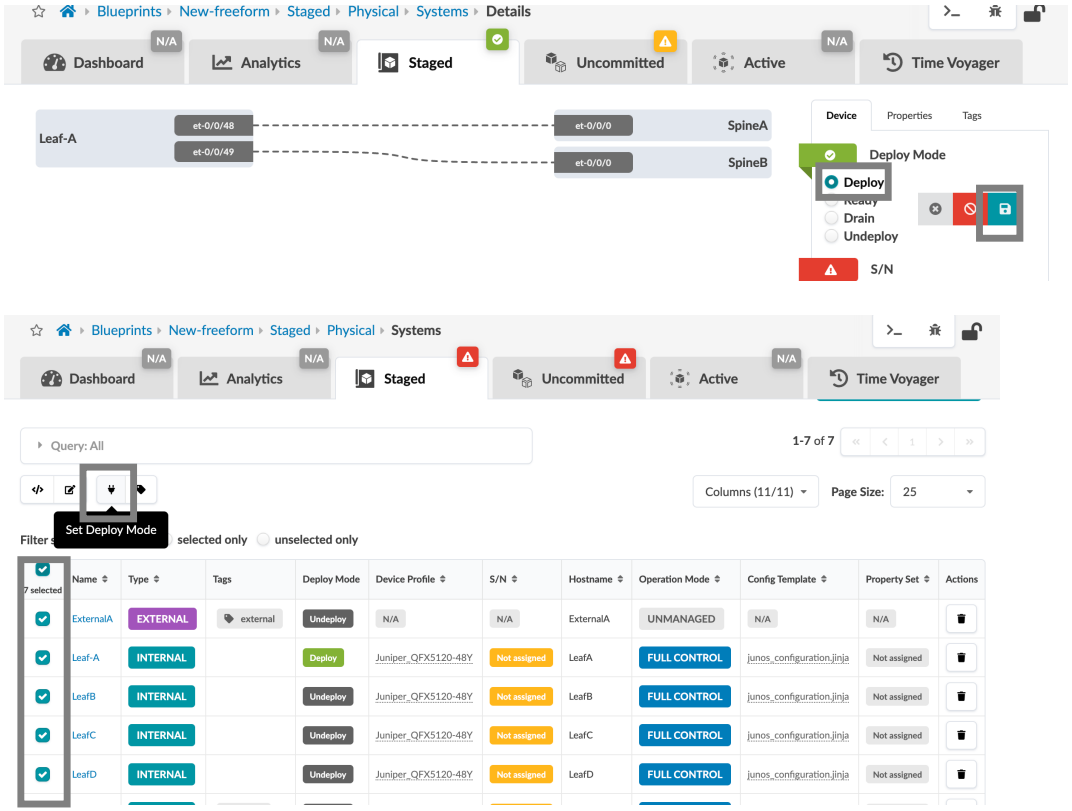


Leaf-A Rendered Config Preview

```
system {
    host-name LeafA;
}
interfaces {
    replace: et-0/0/48 {
        description "facing_spinea:et-0/0/0";
        mtu 9216;
        unit 0 {
            family inet {
                address 192.168.1.2/30;
            }
        }
    }
    replace: et-0/0/49 {
        description "facing_spineb:et-0/0/0";
        mtu 9216;
        unit 0 {
            family inet {
                address 192.168.2.1/30;
```

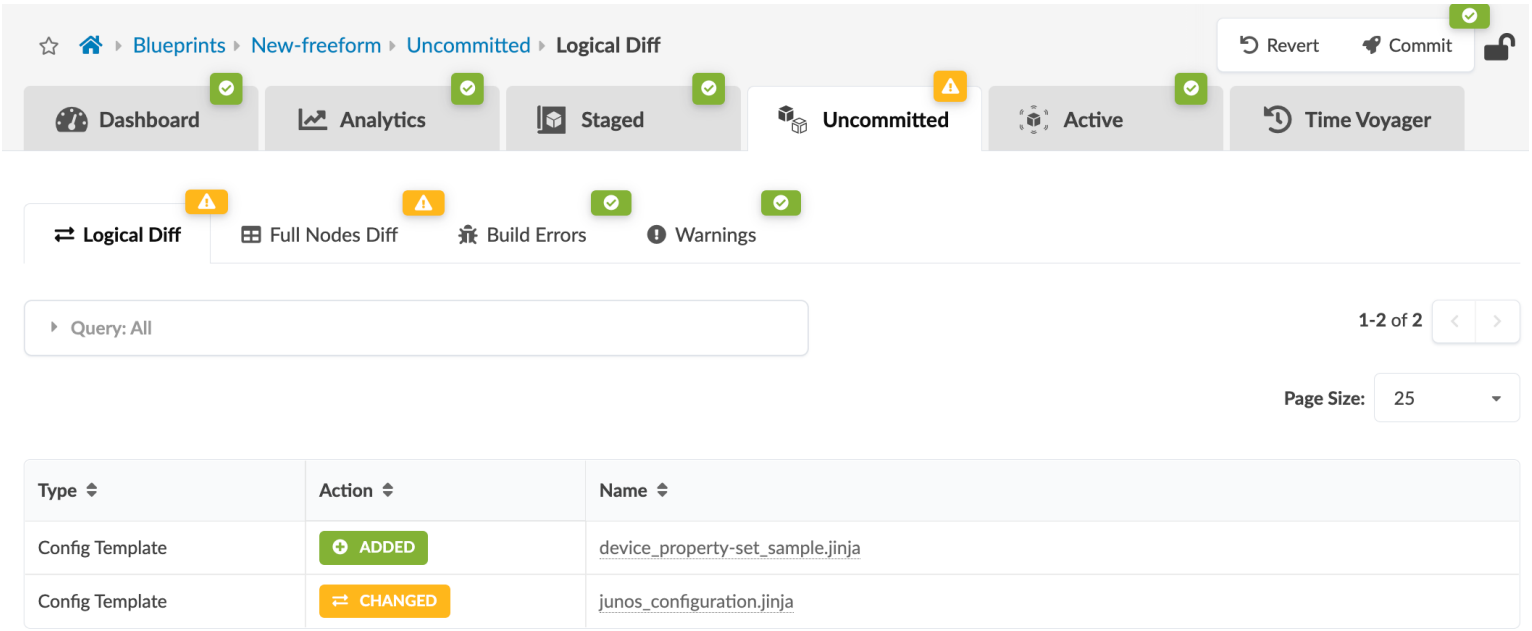
Deploy Mode “Deploy”

各デバイスのDeploy Modeを”Deploy”に変更。
複数選択し、Deploy Modeに変更することも可能。



Commit

Commitにより設定をデバイスに反映。



- Apstra Freeform
  - 初期セットアップ手順
  - **Config Templates**
  - Apstra Telemetry
  - オプション機能
- 参考リンク

Config Templatesオペレーション記載内容

Config Templates概要

- Config Templateサンプル

新規Config Template作成 Day-0

- Jinja reference document
- プリインストールのデフォルトサンプル
- 複数Config Templatesの利用方法

変数値の利用

- Device Context概要
- Device Context変数値確認方法
- Property Sets概要 – 静的変数値
- Property Sets利用方法
- Property Setsサンプル
- Resource Management – 動的変数値利用方法

Config Template 編集方法 Day-2/Advanced編

- Config Template編集
- クローン方法/デバイス毎の異なるConfig Template
- Day2オペレーションサンプル – vlan追加

Advanced編

- Day2オペレーションサンプル – Drain Modeサンプル
- タグベースの動的なコンフィグ生成

Config Templates with Jinja2 Templates概要

Direct Config

```
system {
    host-name piccadilly-circus;
}
interfaces {
    replace: xe-0/0/0 {
        unit 0 {
            description "facing_oxford-circus:xe-0/0/0";
            family inet {
                address 192.168.0.1/31;
            }
        }
    }
    replace: xe-0/0/1 {
        unit 0 {
            description "facing_green-park:xe-0/0/2";
            family inet {
                address 192.168.0.9/31;
            }
        }
    }
    replace: xe-0/0/2 {
        unit 0 {
```

Jinja2 Template

```
system {
    host-name {{hostname}};
}
```

Nesting - Composable

複数jinjaテンプレート利用

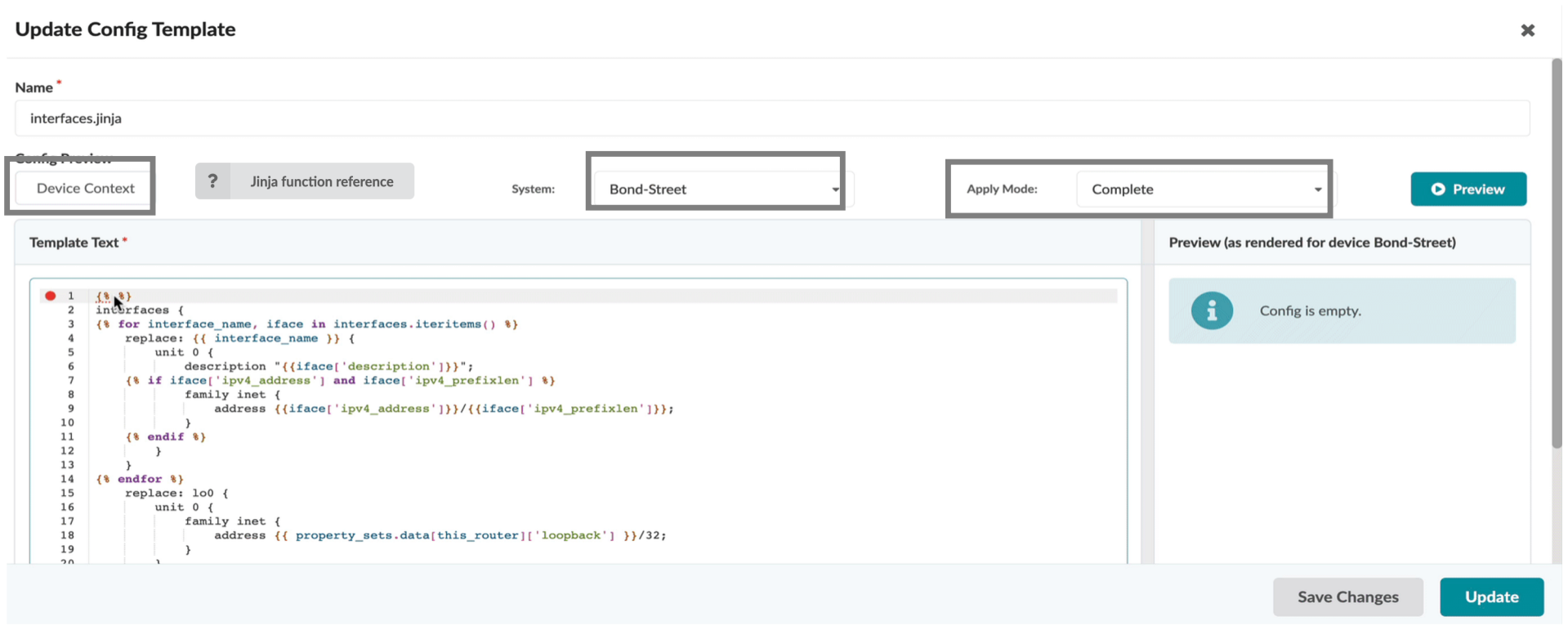
```
{% include "system.jinja" %}

{% include "interfaces.jinja" %}

{% include "protocols.jinja" %}
```

※ Jinja2 Templates形式でなくJUNOS CLIをそのまま貼り付けてデバイスに適用することも可能

Config Templates with Jinja2 プレビュー

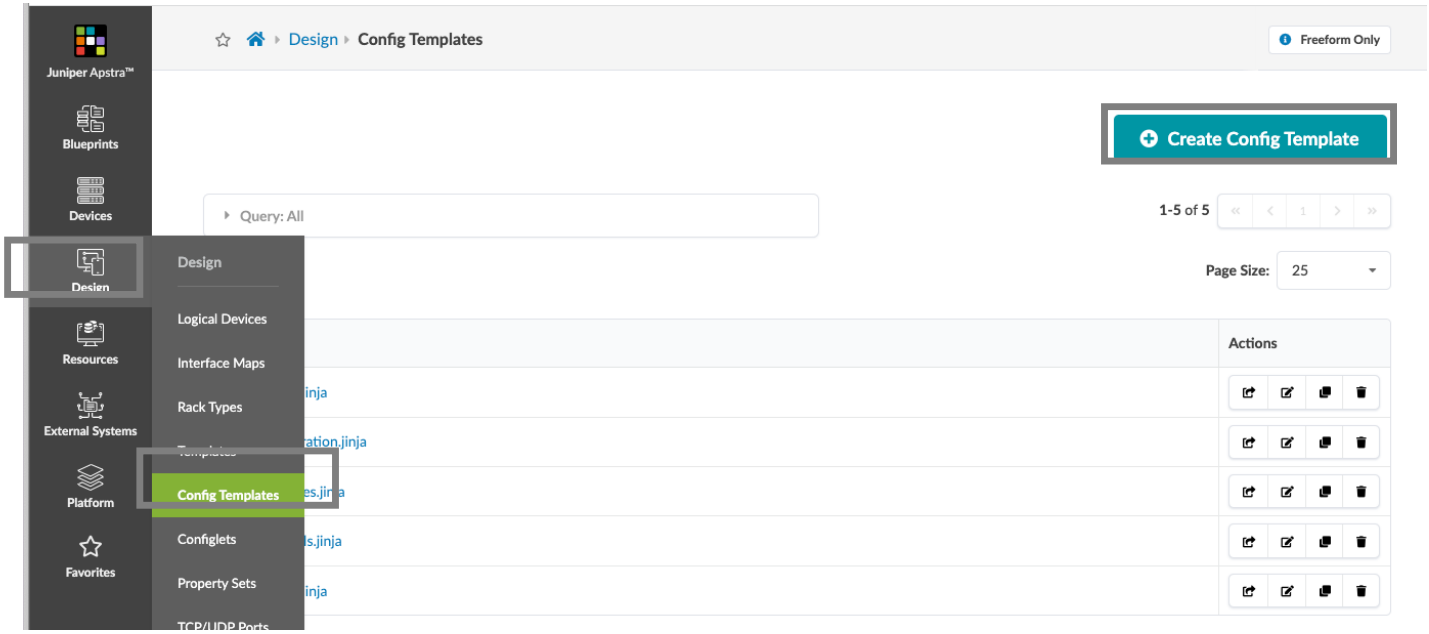
Blueprint作成後に、Templateから各デバイスにどのようなconfigが生成されるか動的に確認可能
Blueprint > Staged > Catalog > Config Templates >

Innovative Interactive Editor



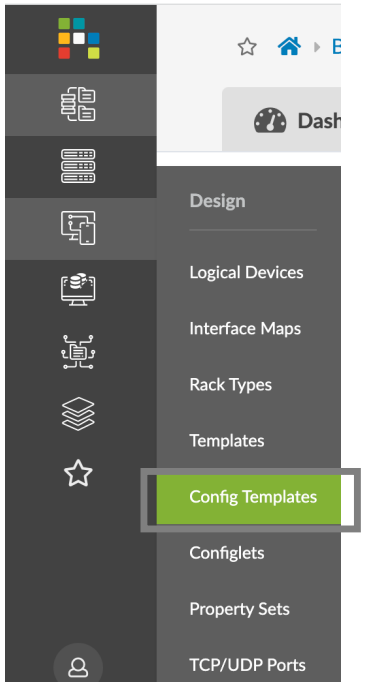
新規Config template作成

テンプレートを新規作成する場合は”Create Config Template”で作成。



Config Templates プリインストールデフォルト サンプル

予めプリロードされたConfig Templatesを活用・参照し、Templates作成の利活用可能。



既存テンプレートの一覧
デフォルトでjunos_configuration.jinjaなどいくつかのテンプレートが用意されているため
それらを複製し作成することも可能。

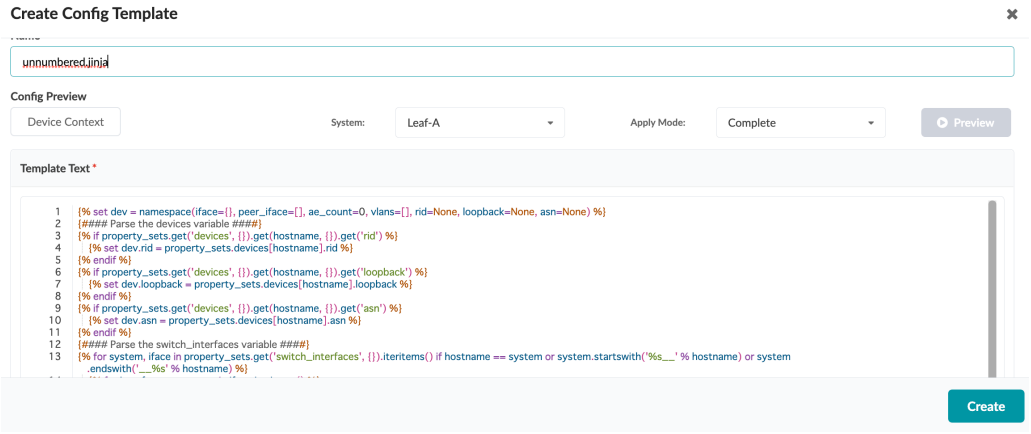
Name	Actions
junos_chassis.jinja	
junos_configuration.jinja	
junos_interfaces.jinja	
junos_ospf-static.jinja	
junos_protocols.jinja	
junos_system.jinja	
ospf.jinja	

Config Templateの利用パターン

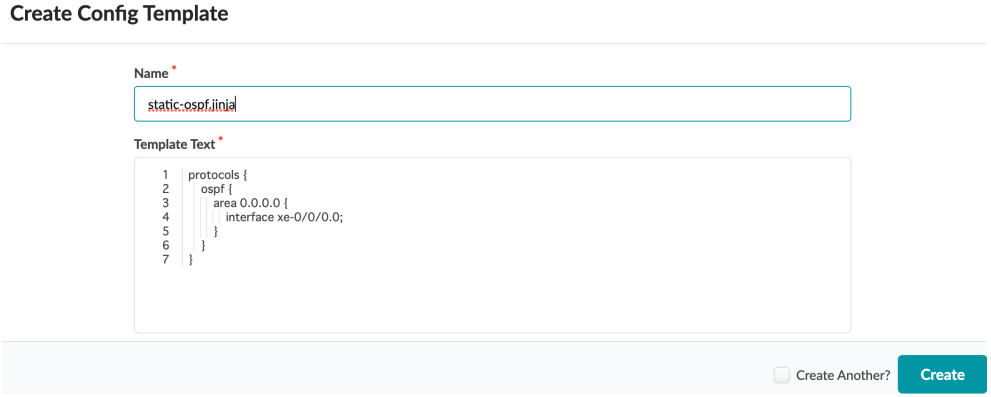
Jinja2テンプレートで構文を利用する方法と、シンプルにJUNOS設定をそのまま利用するケースがあり。

1.Jinja2構文活用して様々な変数や繰返処理を使ってテンプレートを作ることも、
2.変数を使用せず静的にJUNOS設定をそのまま定義することでもテンプレートを作成することも可能。

1.Jinja構文活用 (advanced)



2.静的なJUNOS設定(変数は最小限) (no-code)



複数のConfig Template 利用

- 複数のテンプレートを１台のデバイスに適用したい場合、ひとつ親となるテンプレートを作成し、そこに複数のテンプレートを指定することで適用。
- デフォルトサンプルのjunos_configuration.jinjaの中身を確認すると他のテンプレートをネステッドし、テンプレートを作成していることが確認できる。

※{{変数}}については次ページにで記載。

Expanded View | Compact View

Common Parameters

親となる、
デビスに適用する
テンプレート

Name	junos_configuration.jinja
Tags	
Assigned To	0 systems

Text

```
{% block system %}
{%- include "junos_system.jinja" %}
{% endblock system %}

{% block chassis %}
{%- include "junos_chassis.jinja" %}
{% endblock chassis %}

{% block interfaces %}
{%- include "junos_interfaces.jinja" %}
{% endblock interfaces %}

{% block protocols %}
{%- include "junos_protocols.jinja" %}
{% endblock protocols %}
```

複数のjinjaテンプレートを利用する指定をすることで、
ひとつのコンフィグレーションとして生成可能になる

Name	junos_interfaces.jinja
Tags	
Assigned To	0 systems

```
{% for interface_name, iface in interfaces.iteritems() %}
  {% if loop.first %}
interfaces {
  {% endif %}{# loop.first #}
  {% set is_l2 = function.is_l2_interface(iface) %}
  {% set is_part_of_parent_intf = iface.get('part_of') %}
  {#  Ensure any Layer 3 interface, interfaces with bond members are rendered in configuration. #}
  {% if iface.get('description') or iface.get('ipv4_address') or iface.get('ipv6_address') or iface.get
      ('part_of') or iface.get('composed_of') %}
  replace: {{interface_name}} {
    {% if iface.get('operation_state') == 'down' %}
    disable;
    {% endif %}{# iface.get('operation_state') == 'down' #}
    {% if iface.get('description') %}
    description "{{iface['description']}}";
    {% endif %}{# iface.get('description') #}
    {% if portSetting is defined and portSetting.get(interface_name, {}).get('interface', {}).get('speed') %}
    speed {{ portSetting[interface_name]['interface']['speed'] }};
    {% endif %}{# portSetting is defined and portSetting.get(interface_name, {}).get('interface', {}).get
        ('speed') #}
    {% if portSetting is defined and portSetting.get(interface_name, {}).get('interface', {}).get('link_mode'
        ) %}
    link-mode {{ portSetting[interface_name]['interface']['link_mode'] }};
    {% endif %}{# portSetting is defined and portSetting.get(interface_name, {}).get('interface', {}).get
        ('link_mode') #}
```

Config Templatesオペレーション記載内容

Config Templates概要

- Config Templateサンプル

新規Config Template作成 Day-0

- Jinja reference document
- プリインストールのデフォルトサンプル
- 複数Config Templatesの利用方法

変数値の利用

- **Device Context概要**
- **Device Context変数値確認方法**
- **Property Sets概要 – 静的変数値**
- **Property Sets利用方法**
- **Property Setsサンプル**
- **Resource Management 利用方法**

Config Template 編集方法 Day-2/Advanced編

- Config Template編集
- クローン方法/デバイス毎の異なるConfig Template
- Day2オペレーションサンプル – vlan追加

Advanced編

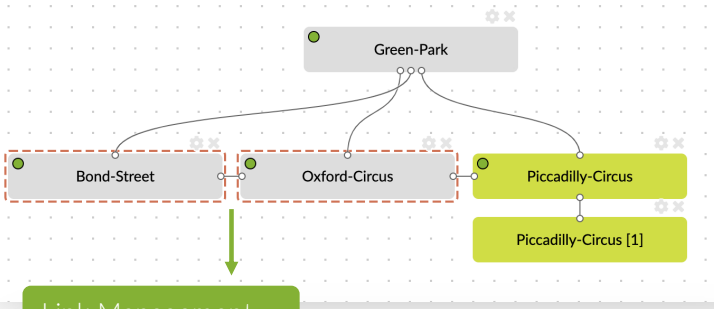
- Day2オペレーションサンプル – Drain Modeサンプル
- タグベースの動的なコンフィグ生成

Device Context 変数値の利用の概要

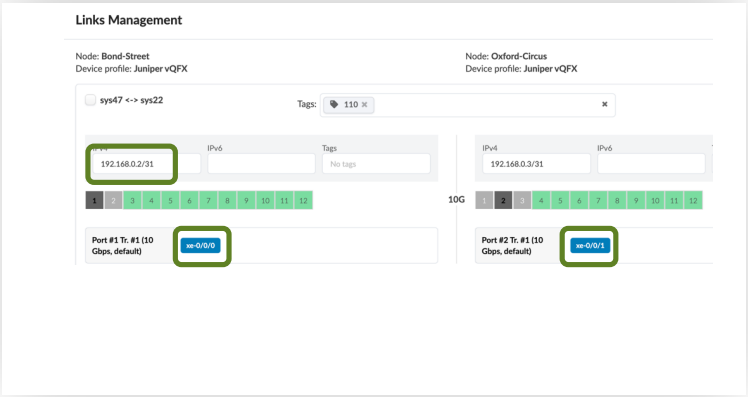
Network DesignerやDevice Profile の情報を変数値として利用しコンフィグの自動生成

例:Network Designerでアサインしたアドレスから動的にコンフィグ作成

Network Desighner



Link Management



Device Context

```
interfaces
{
  xe-0/0/0
  {
    link_tags [ ... ]
    neighbor_interface { ... }
    port_settings { ... }
  composed_of: []
  description: "facing_oxford-circus:xe-0/0/1"
  id: "fwXE7OAvM9HfzuocY04"
  if_type: "ethernet"
  index: 0
  ipv4_address: "192.168.0.2"
  ipv4_prefixlen: 31
  ipv6_address: null
  ipv6_prefixlen: null
  label: null
  lag_mode: null
  name: "xe-0/0/0"
  part_of: null
  port_channel_id: null
  port_id: 1
  role: "internal"
  tags: []
  transformation_id: 1
  }
  xe-0/0/1 { ... }
  xe-0/0/10 { ... }
```

Jinja2 Template

```
1  {% set this_router=hostname %}
2  interfaces {
3  {% for interface_name, iface in interfaces.iteritems() %}
4     replace: {{ interface_name }} {
5        unit 0 {
6           description "{{iface['description']}}";
7     {% if iface['ipv4_address'] and iface['ipv4_prefixlen'] %}
8           family inet {
9              address {{iface['ipv4_address']}}/{{iface['ipv4_prefixlen']}};
10          }
11    {% endif %}
```

Config

Bond-Street Rendered Config Preview

```
7   interfaces {
8      replace: xe-0/0/0 {
9         unit 0 {
10           description "facing_oxford-circus:xe-0/0/1";
11        family inet {
12           address 192.168.0.2/31;
13        }
14     }
15  }
```

KB:[Juniper Apstra] Jinja Configlets using dynamic data from deviceModel

https://juniper.lightning.force.com/lightning/r/Knowledge__kav/ka03c000000rSslAAC/view

Device Context 変数値の利用サンプル

例として以下のように{{hostname}}としてConfig Templateに設定した変数が、
“LeafA”の値として設定に反映されていることが確認できる。

Name	junos_system.jinja
Tags	
Assigned To	0 systems

```
{% if hostname is defined and hostname%}
system {
    host-name {{hostname}}
}
{% endif %}
```

Device Context

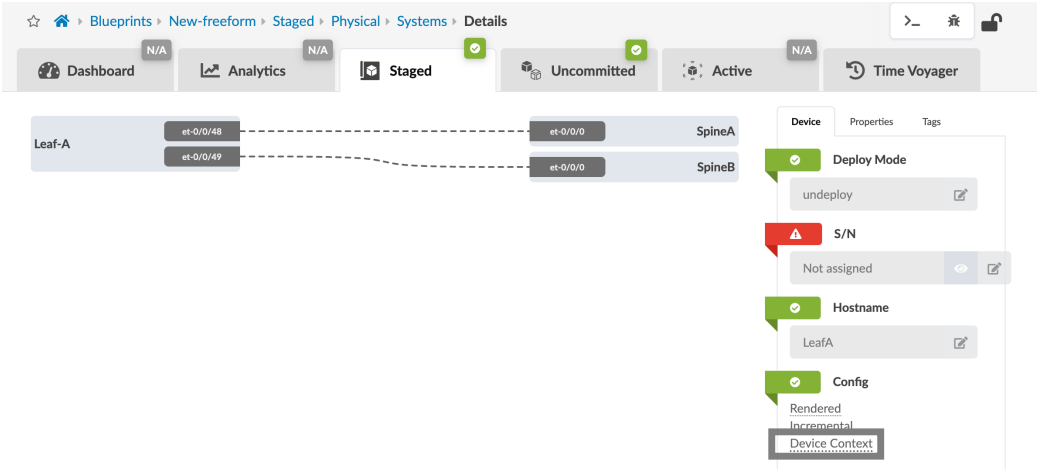
```
▸ all_resources { ... }
▸ interfaces { ... }
  aos_version: "4.1.1"
  configured_system_type: "internal"
  deploy_mode: "deploy"
  hostname: "LeafA"
  id: "sUfrSty6fNKtBCumWG8"
  management_ip: null
  model: "Juniper_QFX5120-48Y.*"
  name: "Leaf-A"
  os_family: "Junos"
  property_sets: {}
  reference_architecture: "freeform"
  resources: []
  routing_instance_supported: true
  system_tags: []
  system_type: "internal"
```

Leaf-A Rendered Config Preview

```
system {
    host-name LeafA;
}
interfaces {
    replace: et-0/0/48 {
        description "facing_spinea:et-0/0/0";
        mtu 9216;
        unit 0 {
            family inet {
                address 192.168.1.2/30;
            }
        }
    }
    replace: et-0/0/49 {
        description "facing_spineb:et-0/0/0";
        mtu 9216;
        unit 0 {
            family inet {
                address 192.168.2.1/30;
```

Device Context確認 (デバイス変数値)

各デバイスの変数/パラーメーターは、Device Contextから確認可能。
このインタフェースやアドレスはTopologyで設定したものを変数値として利用可能。



Device Context


```
▸ all_resources { ... }
▾ interfaces
{
  ▾ et-0/0/48
  {
    ▸ neighbor_interface { ... }
    composed_of: [ ]
    description: "facing_spinea:et-0/0/0"
    id: "3N7xaLPB3c2g16Tt-HY"
    if_type: "ethernet"
    index: 0
    ipv4_address: "192.168.1.2"
    ipv4_prefixlen: 30
    ipv6_address: null
    ipv6_prefixlen: null
    label: null
    lag_mode: null
    link_tags: [ ]
```

Property Sets 概要

各デバイス固有の変数値を設定し、Jinji2で動的にアサインしのコンフィグの自動生成

Property Sets

例:各デバイス毎のas番号をproperty setとして定義し、動的にコンフィグの生成

Name	data
Values	Properties { 4 items bond-street : { 2 items asn : 47 loopback : "10.0.0.2" } piccadilly-circus : { 2 item asn : 23 loopback : "10.0.0.1" } green-park : { 2 items asn : 86 loopback : "10.0.0.3" } oxford-circus { 2 items asn : 22 loopback : "10.0.0.0" } }

Jinja2 Template

```
46    bgp {
47        group external-peers {
48            type external;
49            export send-direct;
50    {% for interface_name, iface in interfaces.iteritems() if iface.get('ipv4_address') and iface.get
          ('neighbor_interface', {}).get('ipv4_address') %}
51        {% set peer_hostname=iface.neighbor_interface.system_hostname %}
52            neighbor {{ iface.neighbor_interface.ipv4_address }} {
53                peer-as {{ property_sets.data[peer_hostname]['asn'] }};
54                export add-med-{{ iface.link_tags[0] }};
55            }
56    {% endfor %}
57        }
58    }
59    }
60
61    routing-options {
62        autonomous-system {{ property_sets.data[this_router]['asn'] }};
63    }
```

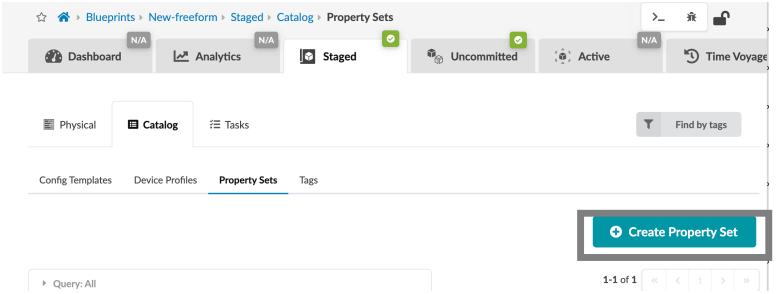
Config

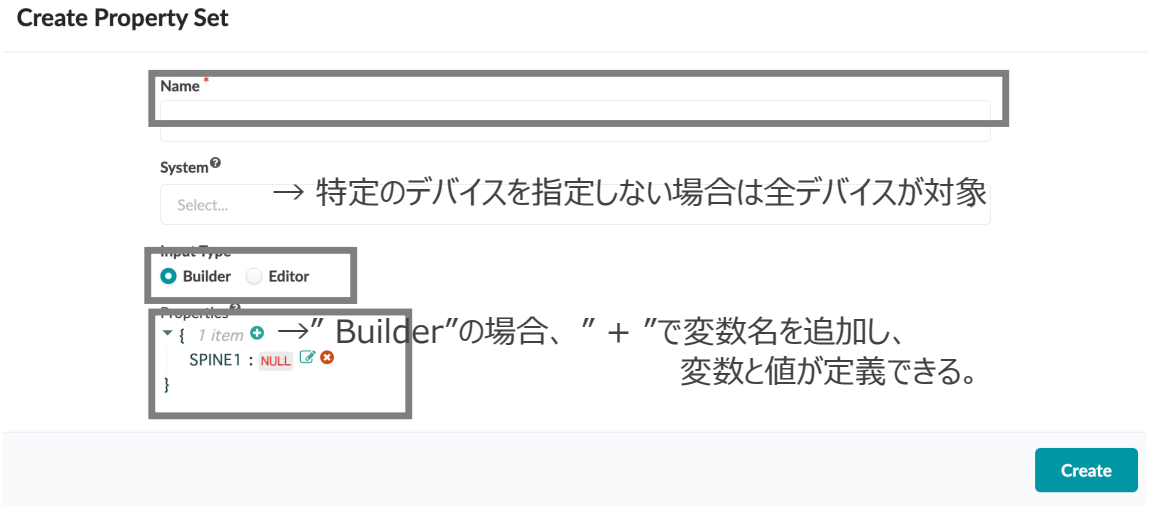
Bond-Street Rendered Config Preview

```
123    bgp {
124        group external-peers {
125            type external;
126            export send-direct;
127            neighbor 192.168.0.3 {
128                peer-as 22;
129                export add-med-110;
130            }
131            neighbor 192.168.0.7 {
132                peer-as 86;
133                export add-med-177;
134            }
135        }
136    }
137 }
138 routing-options {
139     autonomous-system 47;
140 }
```

Property Sets利用方法

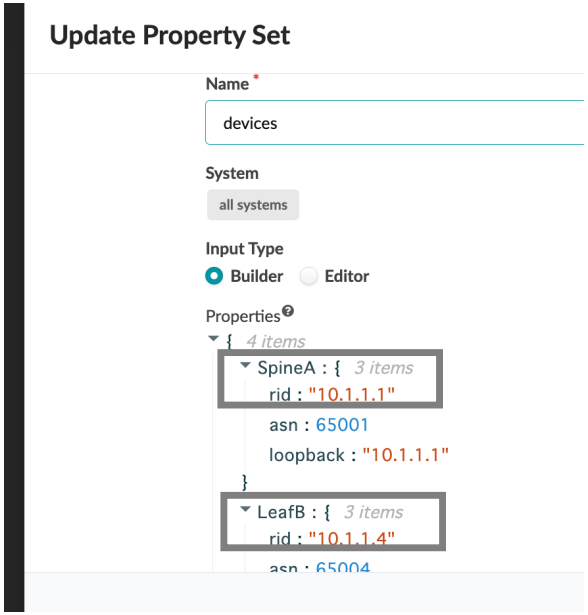
Config Template内の変数は、Device Contextから利用できる変数の他に
Blueprint内のProperty Setsとして変数定義も可能。

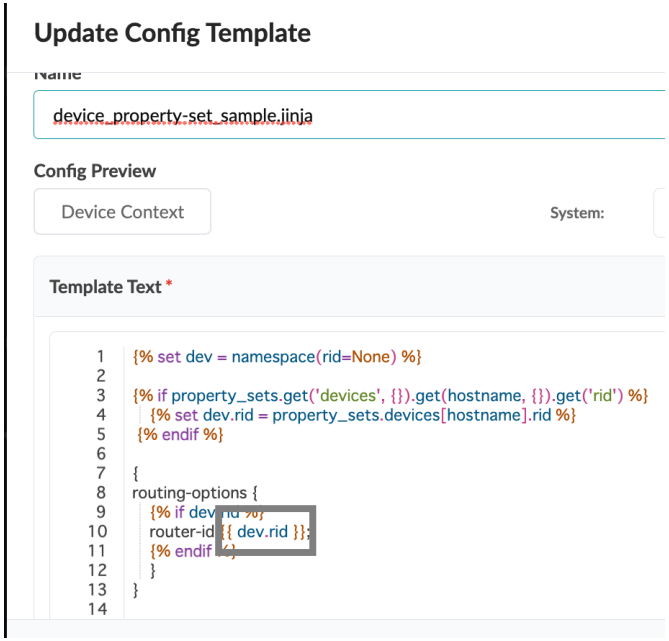


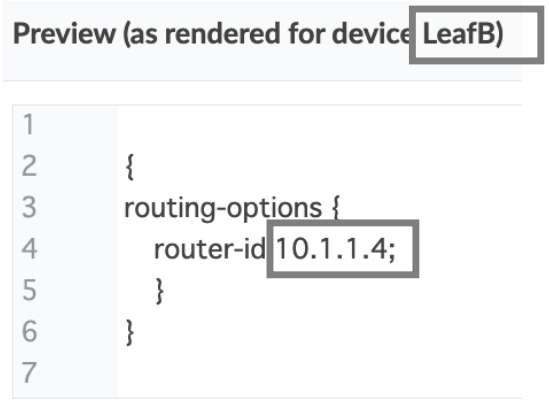


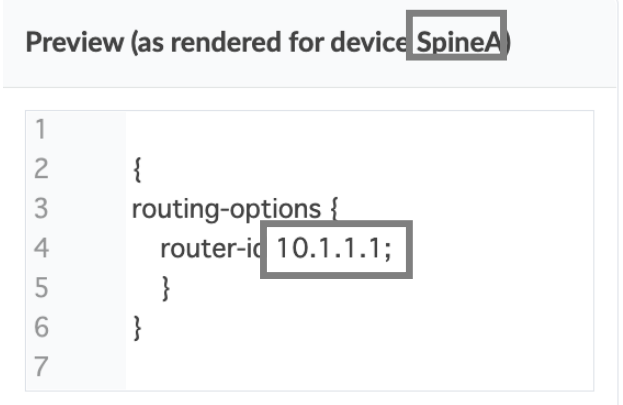
Property Setサンプル

例として以下のように各デバイスに対して変数と値を定義し、ridの値を各デバイスに適用することも可能。
これらを活用し、junos設定全体のテンプレート定義を行うこともできる。









v4.1.2より

Resource Management in Freeform

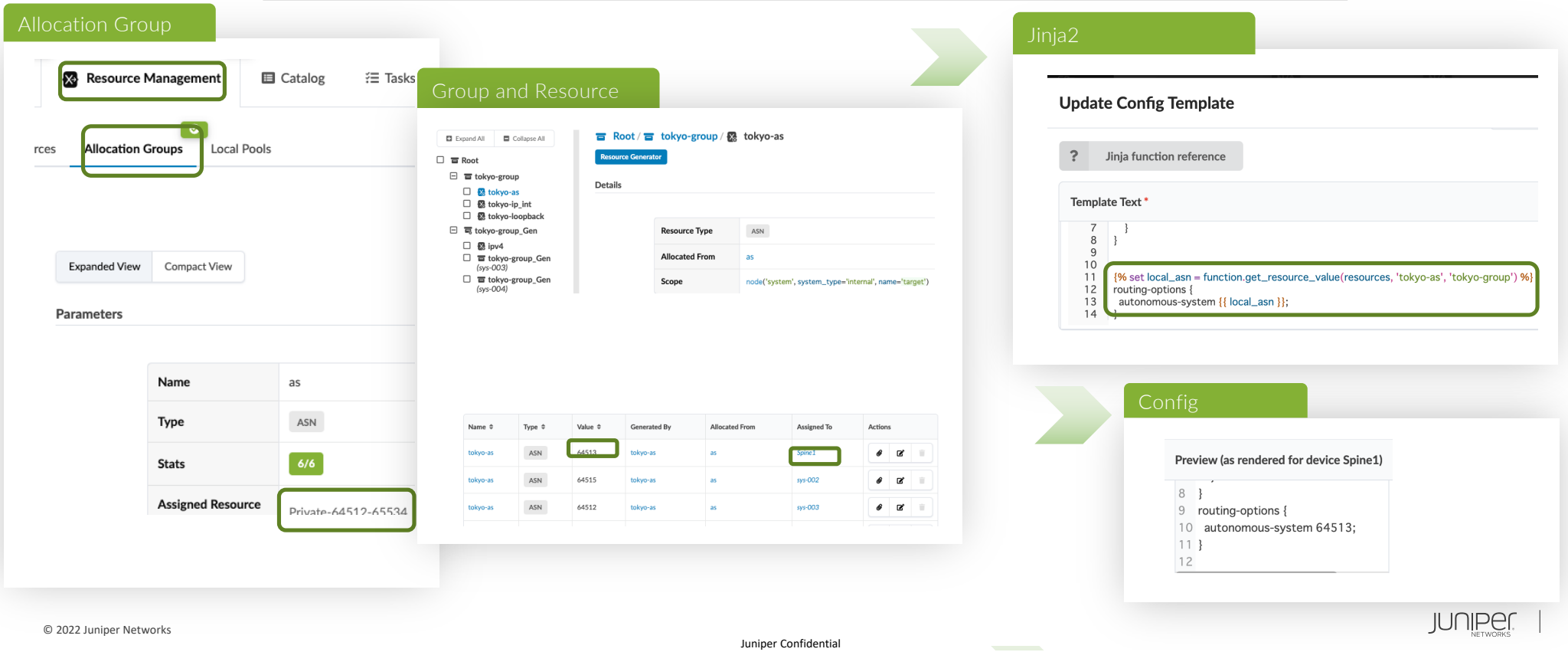
4.1.2以前は変数に対する値はproperty setまたはDevice Contextから
静的にアサインする必要があったが、Freeformのリソースアロケーションにより、
リソースプールから動的な割当が可能となりました。

v4.1.2より

Resource Management概要 ～動的変数値割当～

リソースのレンジから動的に各デバイスシステムやリンクにアドレスを割り当てjinjaからアサイン。

例:リソースレンジから各デバイス毎のas番号を動的にアサインし、動的にコンフィグの生成



v4.1.2より

Resource Management in Freeform

変数の生成と割当は作成は以下の手順で行う。それぞれの手順を次頁から記載する。

1. 値のレンジ設定
   [Resource]

2.上記Resourceを該当Blueprintで有効化 - Allocation Group
   [Blueprint] -> [Staged] -> [Create Allocation Group]

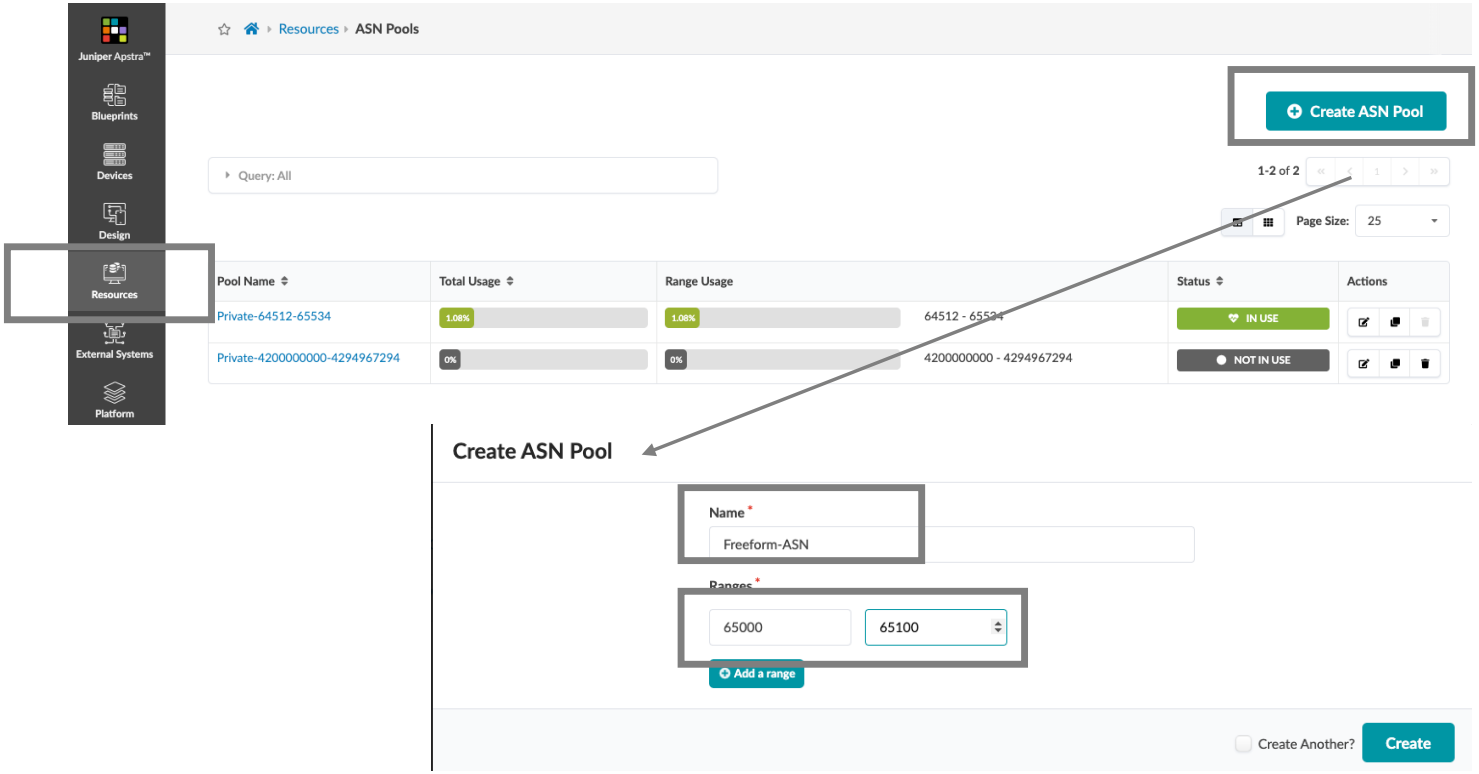
3.[Group]を作成し値を割り当てる対象を生成 - Group / Group Generator
[Blueprint] -> [Staged] -> [Resource Management] → [Group / Group Generator]
 ※各デバイス単位や、各リンク単位等で割当ることが可能。
    3-1:デバイス単位の割当、3-2:リンク単位での割当の例を示す。

4.[Resource]の値を、作成した[Group]に割り当て - Resource Generator
   [Blueprint] -> [Staged] -> [Resource Management] →[Resource Generator]

5.ConfigTemplateで変数の指定

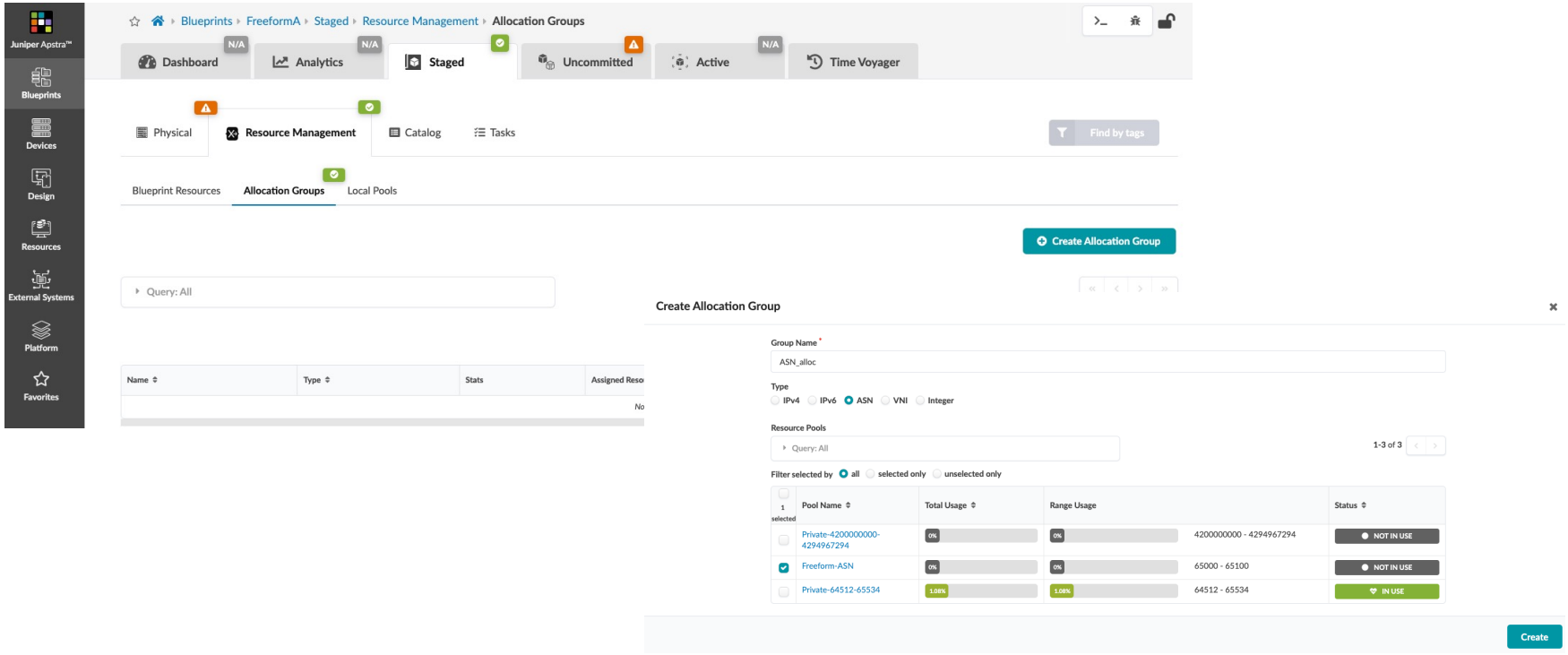
1.Resource – 変数値の指定

DC. Referenceと同様に[Resource]にて利用する値のレンジを設定。
ここでは例として、1.ASNを指定するサンプルを示す。



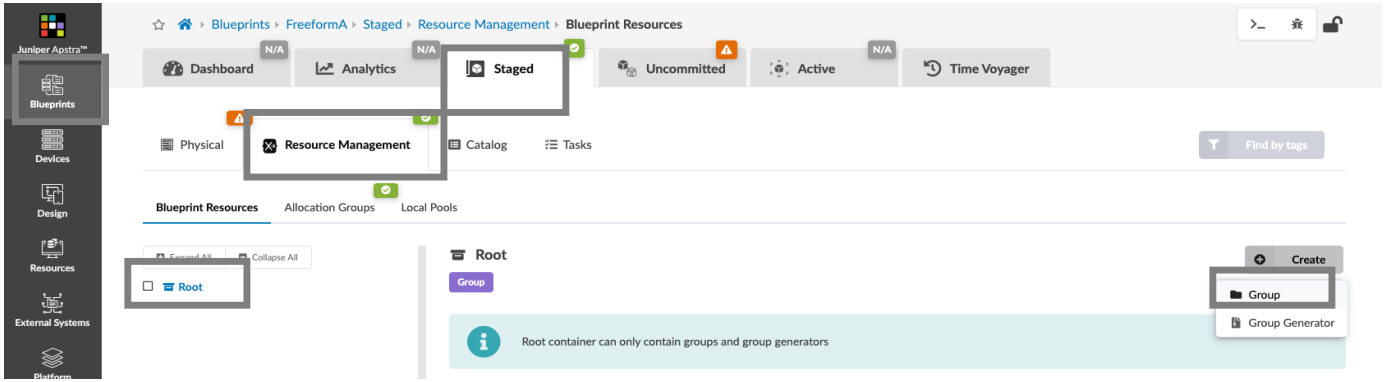
2.Allocation Group – Blueprint内での有効化

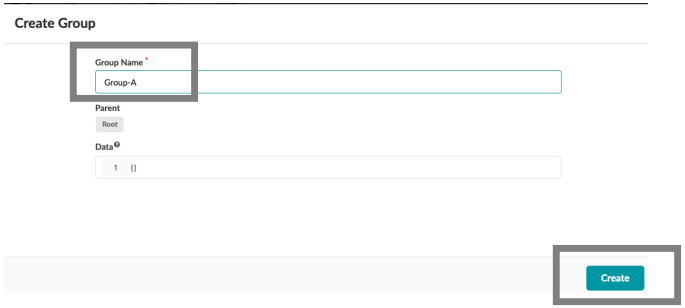
[Resource]で作成した値を、該当するBlueprintで利用できるように割り当て。



3.Resource Management Group

それぞれの値を割り当てるためのグループ分けとして、Resource Groupを作成。
Resource Groupとは、フォルダ構造のような階層型のグループになり、リソースを保存する





※Rootグループに直接リソースを作成できないため、
リソースを割り当てるためのグループを作成。
Device用のGroupや、VRF毎のGroup等を分割する
ような使い方も可能。

3. Resource Management Group Generator

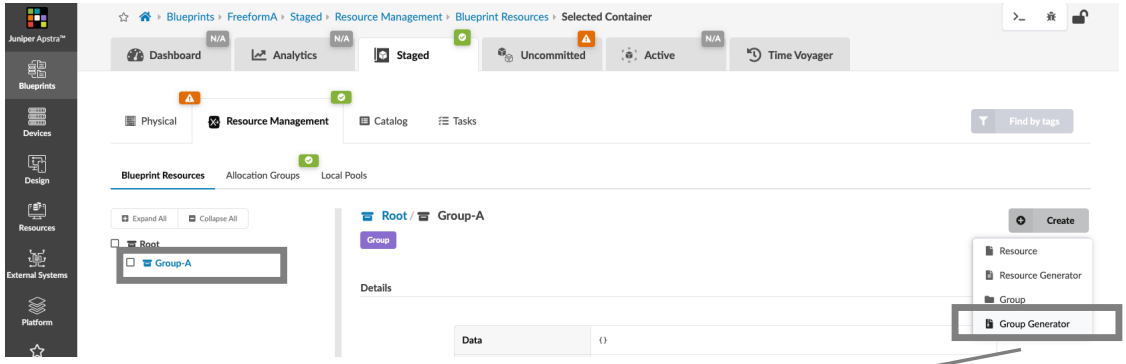
リソースから動的に割り当てる方法としては、いくつかの割当方法があるが、
ここでは以下の2つの例を示します。

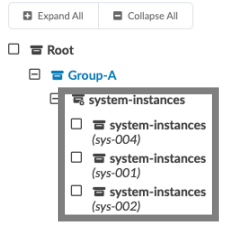
1.ASN,loopbackのように各デバイスに変数をアサイン
- ・各デバイス用のResource Groupを作成
- ・Resource Generatorで各GroupにASNまたはアドレスを割り当てる

2.各リンクにアドレスをアサインする
- ・ Resource Generatorでリンクにアドレスを割り当てる

3-1. Resource Management Group Generator
- デバイス/ノードGroup作成

例として、”ASNやloopbackのような各デバイスに固有の値を割り当てるような例”を示す。
各デバイスに値を割り当てるためのGroupを作成する場合は、“Group Generator”を利用。
※Groupを作成せずにリソースを割り当てることも可能だが、管理を容易にするために作成することを推奨。





3つデバイスのそれぞれに
対してGroupが自動作成される。

※以下をScopeとしてGroup Generatorを行うことで、各ノード/デバイス毎のResourceが生成される。

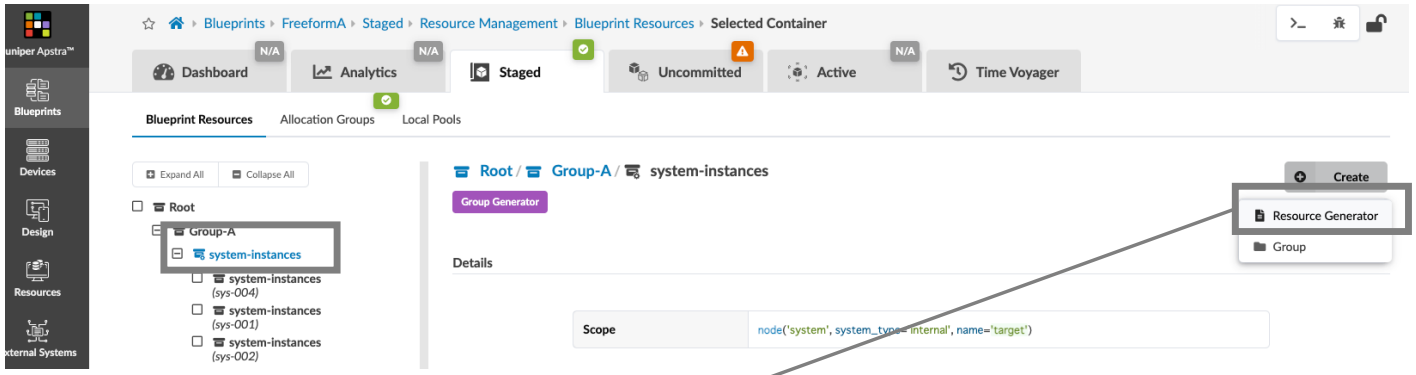
```
node('system',system_type='internal',name='target')
```

この指定方法はApstraが持っているGraph DBから対象とするGroupを生成しており、指定の方法によって様々な対象でGroupが作成できる。

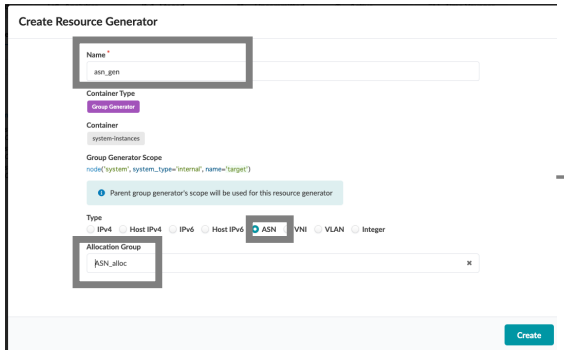
Scopeの記述方法はここでは割愛。

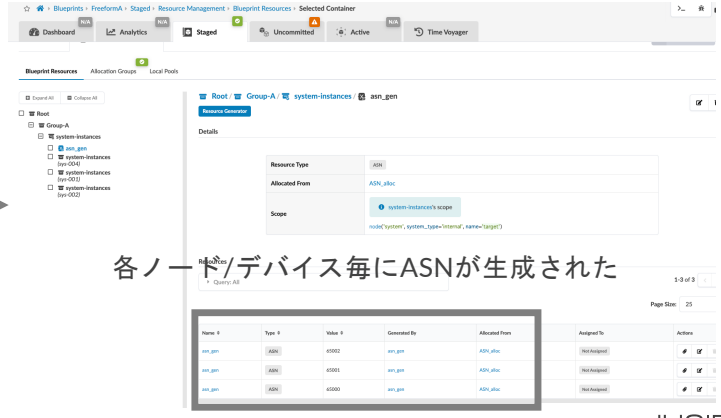
4-1.Resource Generator – ASN割当

事前に設定していたAllocation Groupのリソースレンジから各デバイスに値を割り当てる。
以下では各デバイスにAS番号を割り当てる例を示す。



割り当てる各デバイスの
上位のGroupを選択

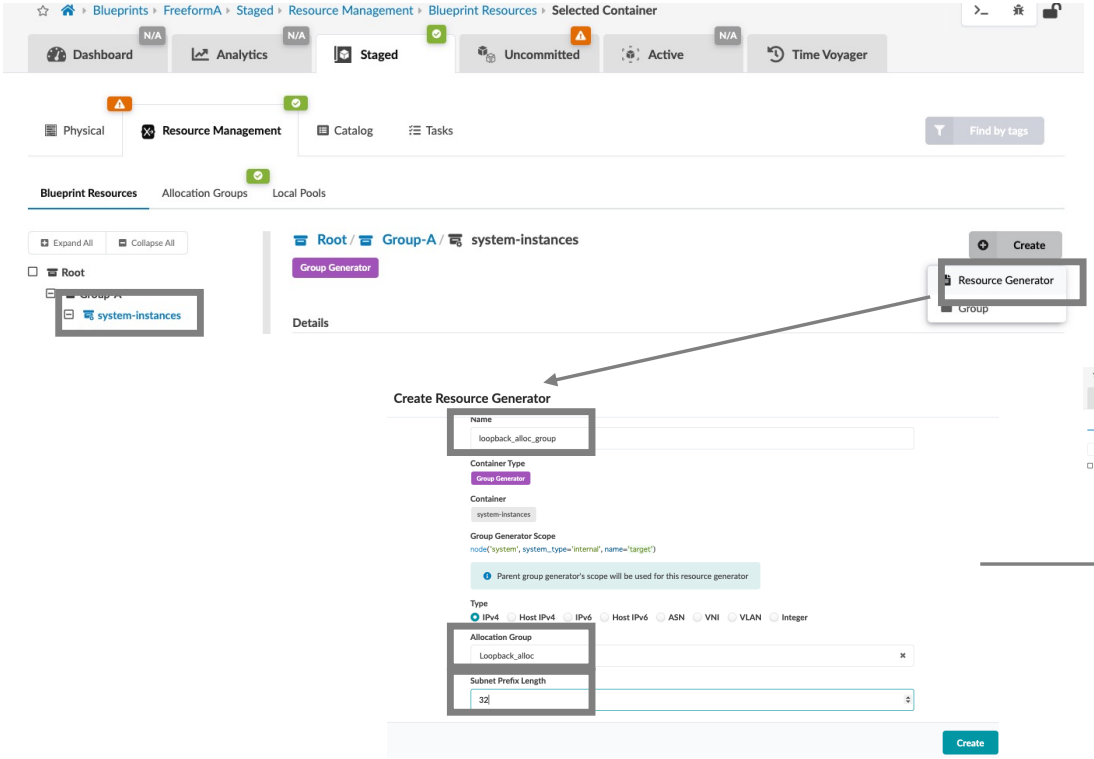


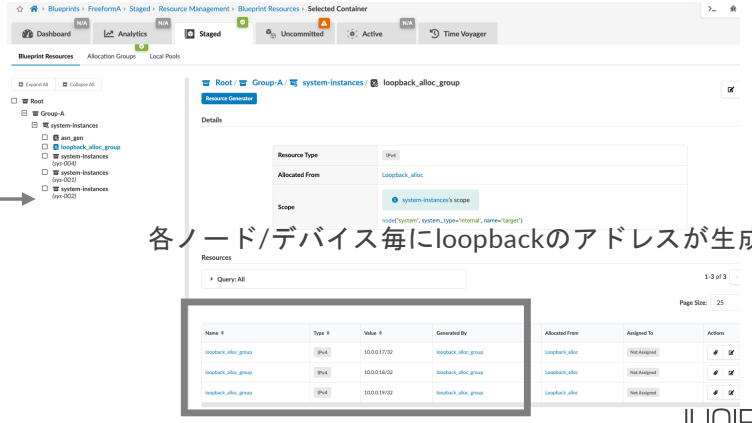


各ノード/デバイス毎にASNが生成された

4-1.Resource Generator – loopback アドレス割当

ASNと同じ用にloopback用の[Resource]と[Allocation Groups]をまずは作成する。
その後、リソースレンジから各デバイスに値を割り当てる。

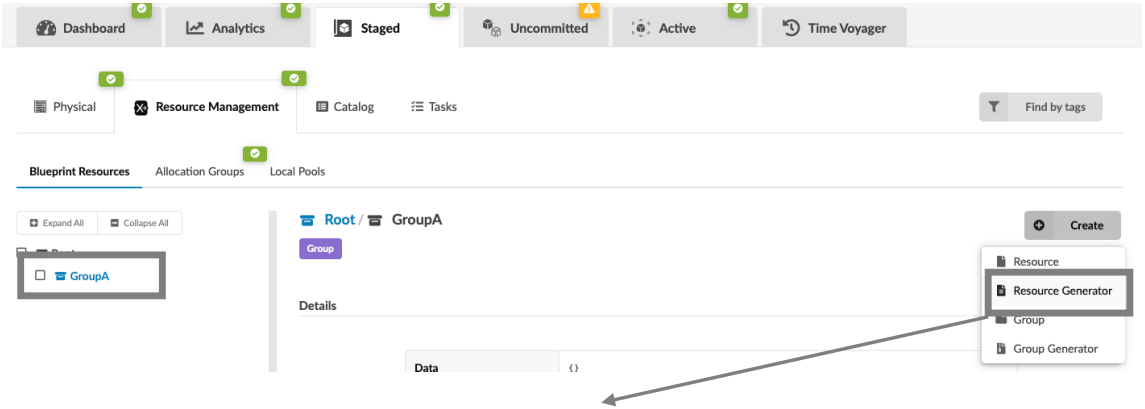


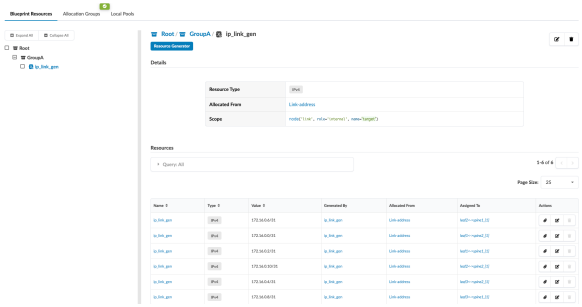


各ノード/デバイス毎にloopbackのアドレスが生成された

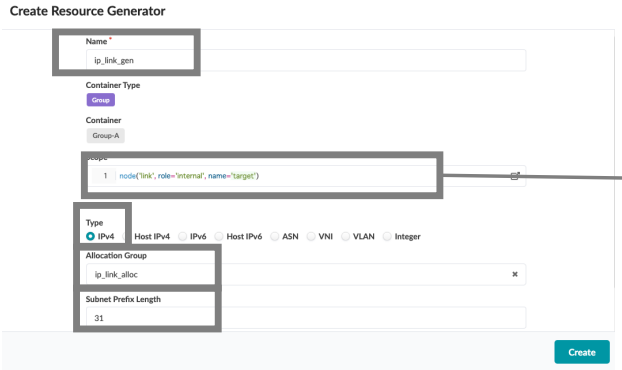
4-2.Resource Generator –物理リンクアドレス割当

次にデバイス毎でなく、デバイス間のリンクにアドレスを割り当てる方法を示す
作成していた全体のGroup を指定して [Resource Generator] でリソースを割り当てる。





リンクにアドレスが割り振りされた



※以下をScopeとしてScope Generatorを行うことで、各Linkに毎にIPv4アドレスが生成される。

```
node('link', system_type='internal', name='target')
```

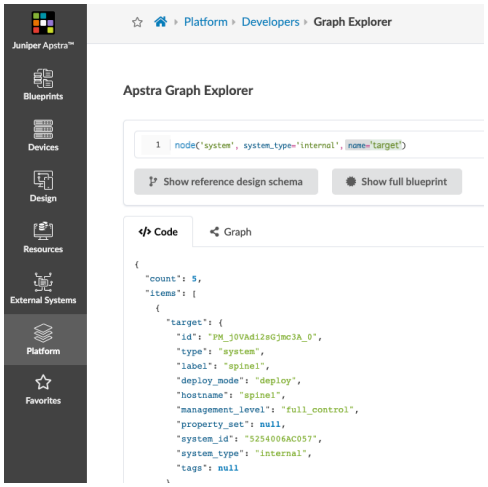
この指定方法はApstraが持っているGraph DBから対象とするGroupを生成しており、指定の方法によって様々な対象でGroupが作成できる。

Scopeの記述方法はここでは割愛。

補足： Graph model Query

Scopeで利用したGraph model部分の詳細はドキュメントを参照のこと。
https://www.juniper.net/documentation/us/en/software/apstra4.1/apstra-user-guide/topics/topic-map/graph.html

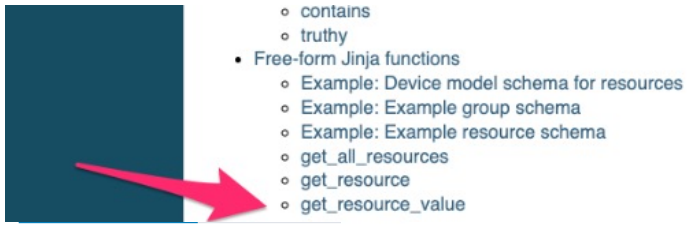
例として、
node('link', role='internal', name='target')
これにより、すべての内部(ファブリックに面する) 'internal' リンクのみが選択される。

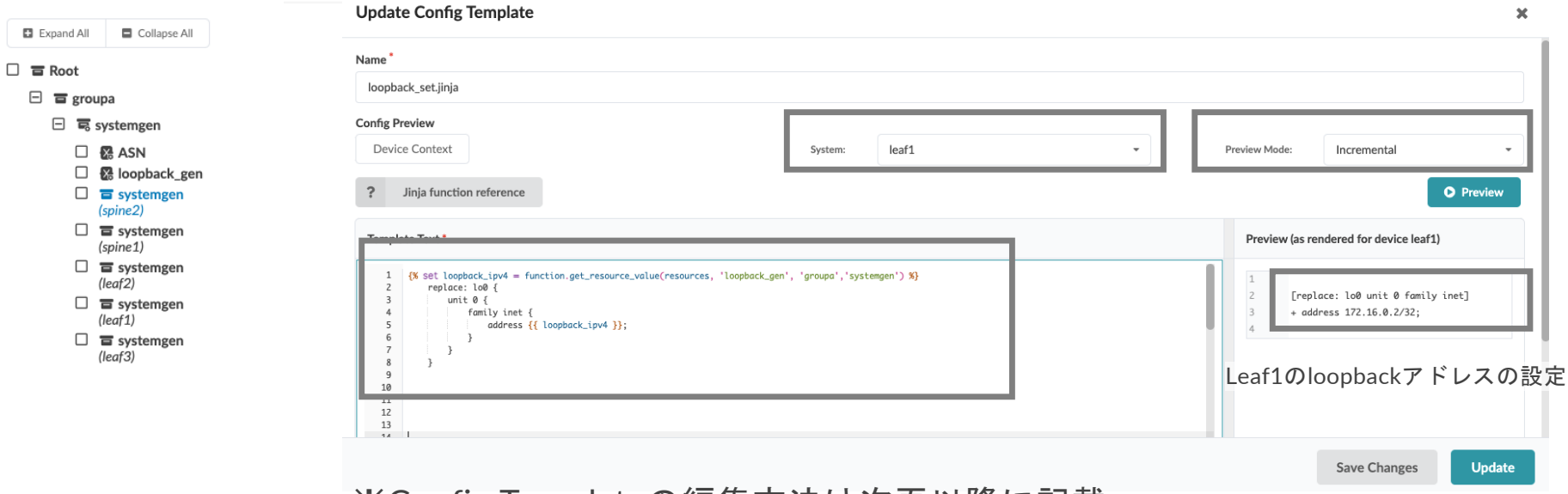


Graph Explorerにてどのような値が取得可能かを確認することも可能。

5.Config TemplateのResource割当

Resource Managerにて割り当てた変数値を割り当てる場合は、”get_resourece_value”を利用することで変数値を指定可能





Leaf1のloopbackアドレスの設定が可能となる

※Config Templateの編集方法は次頁以降に記載

Config Templatesオペレーション記載内容

Config Templates概要

- Config Templateサンプル

新規Config Template作成 Day-0

- Jinja reference document
- プリインストールのデフォルトサンプル
- 複数Config Templatesの利用方法

変数値の利用

- Device Context概要
- Device Context変数値確認方法
- Property Sets概要 – 静的変数値
- Property Sets利用方法
- Property Setsサンプル
- Resource Management – レンジ変数値

Config Template 編集方法 Day-2/Advanced編

- **Config Template編集**
- **クローン方法/デバイス毎の異なるConfig Template**
- **Day2オペレーションサンプル – vlan追加**

Advanced編

- Day2オペレーションサンプル – Drain Modeサンプル
- タグベースの動的なコンフィグ生成

Config template Jinja 参考ドキュメント

Blueprint作成後に、以下のリンクからFreeformのReferenceドキュメントが確認可能

Blueprint > Staged > Catalog > Config Templates > Create Config Template > ? Jinja function reference

Jinja function reference Document

Apstra Jinja Functions documentation »

Table of Contents

Apstra Jinja helper functions reference
Indices and tables

Next topic

Jinja Common Filters

Quick search

Go

Apstra Jinja helper functions reference

This document serves as a general reference against custom jinja filters, tests, and functions as related to configuration templates and configlets.

Consult https://jinja.palletsprojects.com/en/2.11.x/templates/ for general Jinja template designer details.

- Jinja Common Filters
  - bool
  - cidr_merge
  - intersect
  - json_query
  - to_cidr
  - to_ip
  - to_mac_format
  - to_netmask
  - to_network
  - to_prefixlen
- Jinja Common Functions
  - gen_acl_port_token
  - max_os_version
  - merge_vlans_to_list
  - min_os_version
  - raise_error
  - re_findall

Example: Extract two resource values from group underlay

Jinja for example *Extract two resource values from group underlay*

```
{% set loopback_ipv4 = function.get_resource_value(resources, 'ipv4_loopback', 'underlay') %}
{% set bgp_asn = function.get_resource_value(resources, 'bgp_asn', 'underlay') %}
Loopback ipv4: {{loopback_ipv4}} BGP ASN: {{bgp_asn}}
```

Device model context for example *Extract two resource values from group underlay*

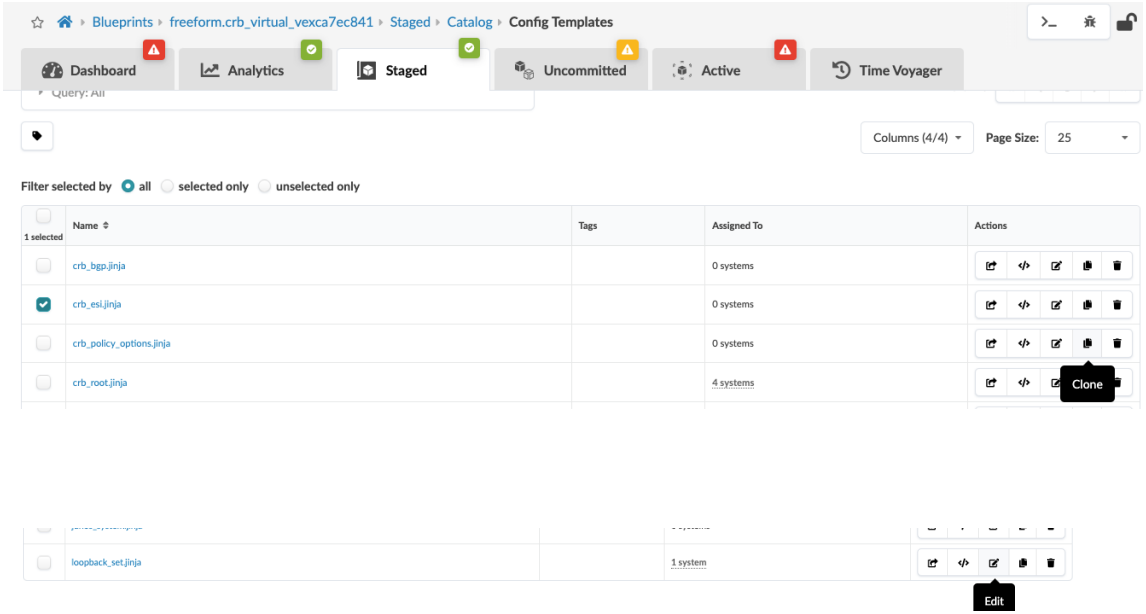

```
{
  "resources": [
    {
      "tags": [],
      "label": "underlay",
      "groups": [],
      "data": {},
      "id": "test_group_id-underlay",
      "resources": [
        {
          "id": "test_res_id-ipv4_loopback-10.0.0.1/32",
          "assigned_to": [],
          "resource_type": "ip",
          "value": "10.0.0.1/32",
          "label": "ipv4_loopback"
        },
        {
          "id": "test_res_id-bgp_asn-64512",
          "assigned_to": [],
          "resource_type": "asn",
          "value": "64512",
          "label": "bgp_asn"
        }
      ]
    }
  ]
}
```


Expected text output for example *Extract two resource values from group underlay*

```
Loopback ipv4: 10.0.0.1/32 BGP ASN: 64512
```

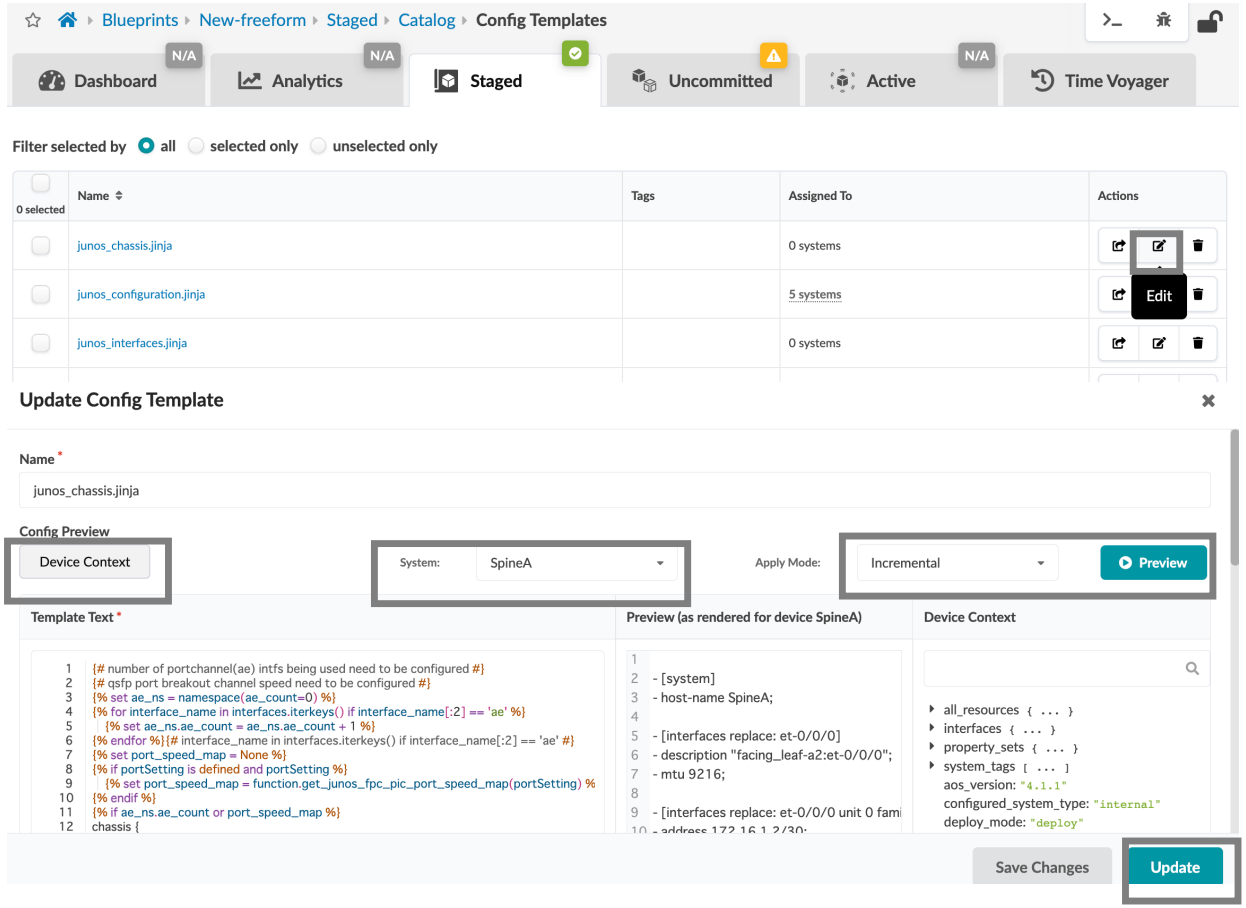
Config Templateの編集

特定のシステムのテンプレートのみ変更したい場合は、
クローンでコピーした後に、編集して特定のシステムに適用することも可能。



Config Templateの編集

Editで編集を行う際に、Systemを指定し、Device ContextやPreviewの確認をすることも可能。



Day2オペレーションサンプル : property set

例:vlan設定のJinja2テンプレートを用意し、Property Setで必要なvlan情報等を追加することで追加

vlans.jinja テンプレート

```
抜粋
{
   "vlans": {
      "<VLAN_ID>": {
         "vrf_name": <VRF_NAME>,
         "description": <DESCRIPTION, optional string>,
         "vxlan_id": <VXLAN_ID, optional integer>,
         "vlan_id": <VLAN_ID, required integer>,
         "ipv4_virtual_gateway_address": <IP Address without prefixlen, optional>
         "systems": {
            <SYSTEM HOSTNAME>: {
               "ipv4_address": <UNIQUE IRB IPV4 ADDRESS WITH PREFIXLEN>,
               "tagged_interfaces": [<LIST OF INTERFACE NAMES WITH VLAN
TAGGING, optional strings>],
               "native_interfaces": [<LIST OF INTERFACE NAMES THIS VLAN IS
NATIVE VLAN FOR, optional strings>]
            }
         }
      }
   }
```

Vlans.json(property-set) 変数

```
{
 "10": {
  "vxlan_id": null,
  "description": "default_vlan10_leaf3_singleton",
  "vrf_name": "default",
  "systems": {
   "leaf3": {
    "ipv4_address": "172.16.10.1/24",
    "tagged_interfaces": [],
    "untagged_interfaces": [
     "ge-0/0/2"
    ]
   }
  },
  "vlan_id": 10,
  "ipv4_virtual_gateway_address": null
 },
 "11": {
  "vxlan_id": 10011,
  "description": "blue_vxlan10011_leaf3_singleton",
  "vrf_name": "blue",
  "systems": {
   "spine1": {
    “ipv4_address”: “172.16.11.2/24”
抜粋,
```

Config Templatesオペレーション記載内容

Config Templates概要

- Config Templateサンプル

新規Config Template作成 Day-0

- Jinja reference document
- プリインストールのデフォルトサンプル
- 複数Config Templatesの利用方法

変数値の利用

- Device Context概要
- Device Context変数値確認方法
- Property Sets概要 – 静的変数値
- Property Sets利用方法
- Property Setsサンプル
- Resource Management 利用方法

Config Template 編集方法 Day-2/Advanced編

- Config Template編集
- クローン方法/デバイス毎の異なるConfig Template
- Day2オペレーションサンプル – vlan追加

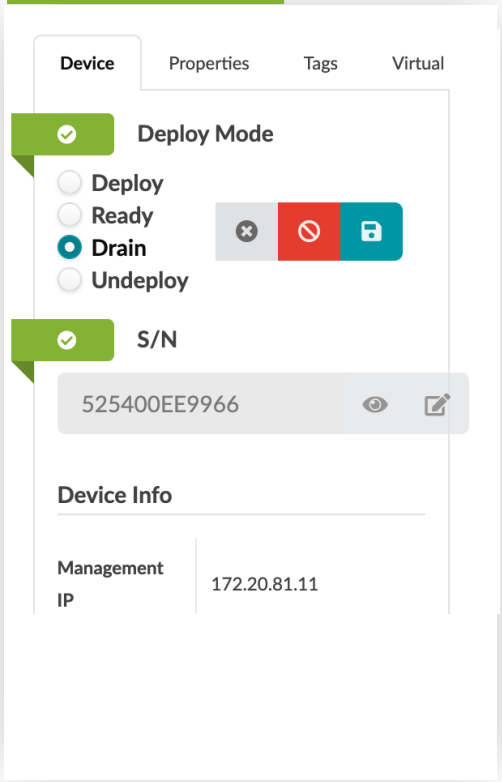
Advanced編

- **Day2オペレーションサンプル – Drain Modeサンプル**
- **タグベースの動的なコンフィグ生成**

Day2 オペレーションサンプル - トラフィック片寄せ

事前にトラフィック片寄のポリシーを作成し、Deploy ModeがDrainを選択することで有効化

Drain Mode



Config Template - Policy

```
policy-options {
    policy-statement AllPodNetworks {
        term AllPodNetworks-10 {
            from {
                family inet;
                protocol direct;
            }
            then accept;
        }
        term AllPodNetworks-100 {
            then reject;
```

抜粋

Config Template - BGP

抜粋

Generated Config

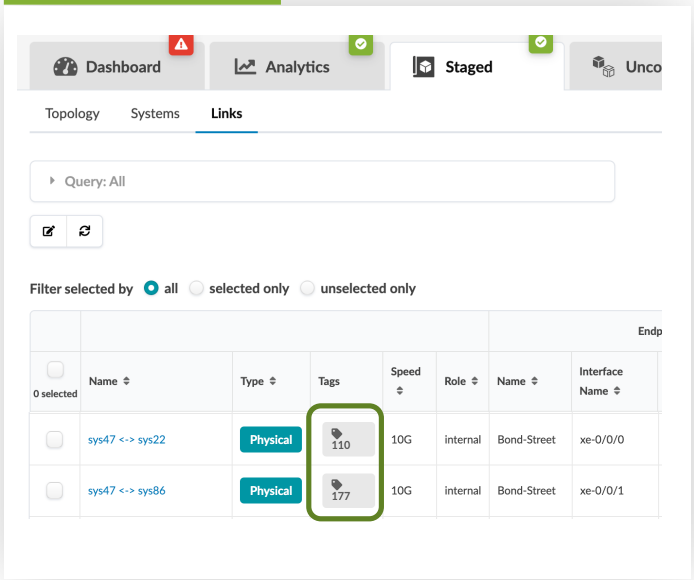
CRBサンプルポリシー : https://github.com/Juniper-SE/apstra-freeform/blob/main/CRB_stage3/crb_policy_options.jinja

タグベースの動的コンフィグ生成

タグはラベリングとして、整理、検索、参照などに活用でき、様々な構成要素(デバイス・リンク・)に設定。
Jinja でタグを使用して、コンフィグ生成時に動的変数を設定することができる。

例:リンクにタグ情報を付与し、タグ情報を元にBGPのMEDを設定

Tag



Config Template

```
{% set this_router = hostname %}
{% set med_tags = namespace(meds=[]) %}
policy-options {
    policy-statement send-direct {
        term 1 {
            from protocol direct;
            then accept;
        }
    }


{% for interface_name, iface in interfaces.iteritems() %}
{#
  First, building a list of link_meds so we can later deduplicate the
  add-med-<value> policy statements
#}
    {% do med_tags.meds.extend(iface.link_tags) %}
{% endfor %}

{% for link_med in med_tags.meds|sort|unique %}
    policy-statement add-med-{{ link_med }} {
        from {
            route-filter 0.0.0.0/0 longer;
        }
        then {
            bgp {
                group external-peers {
                    type external;
                    export send-direct;
{% for interface_name, iface in interfaces.iteritems() if iface.get('ipv4_address') and iface.get
    ('neighbor_interface', {}).get('ipv4_address') %}
{% set peer_hostname=iface.neighbor_interface.system_hostname %}
                    neighbor {{ iface.neighbor_interface.ipv4_address }} {
                        peer-as {{ prop
                        export add-med
```

Config

Bond-Street Rendered Config Preview

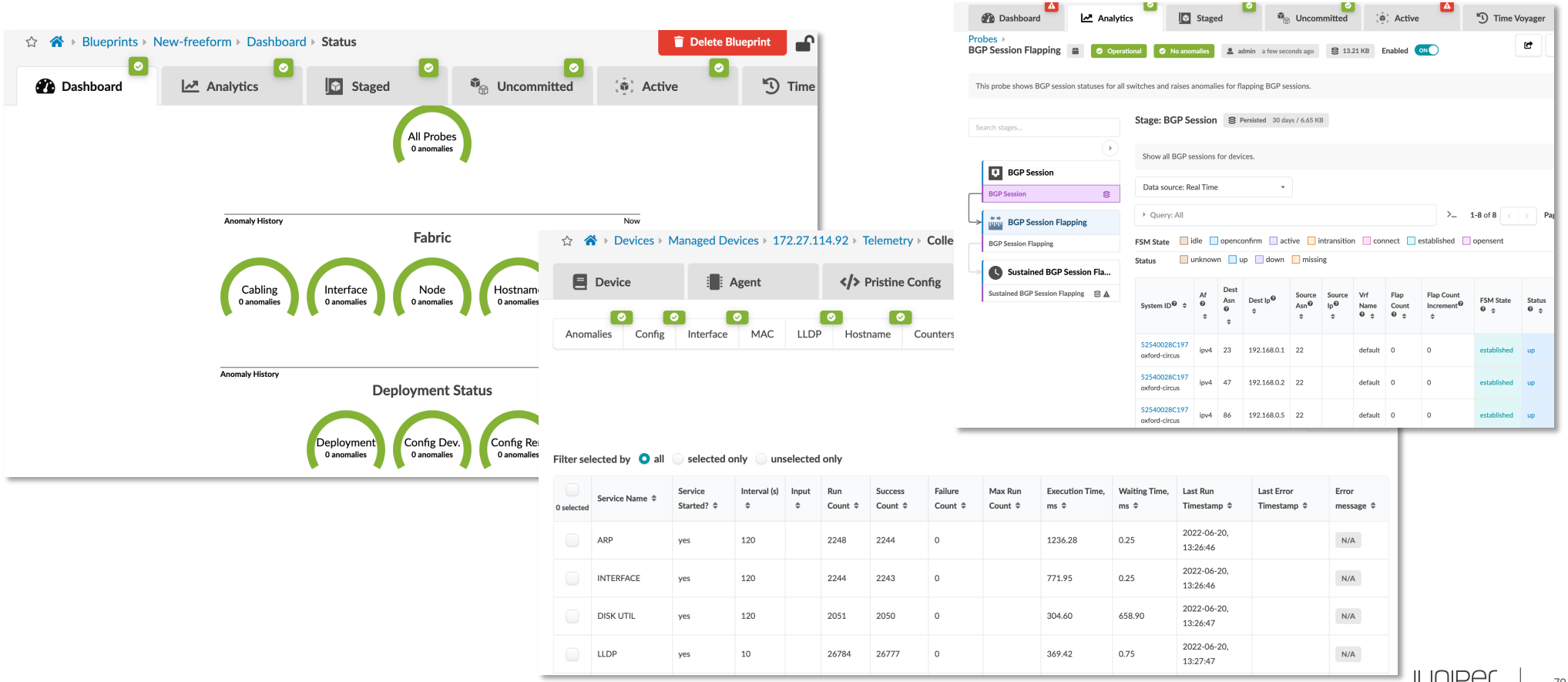
```
    }
    policy-statement add-med-110 {
        from {
            route-filter 0.0.0.0/0 longer;
        }
        then {
            metric ad 110;
        }
        then {
            accept
        }
    }
    policy-statement add-med-177 {
        from {
            route-filter 0.0.0.0/0 longer;
        }
        then {
            metric ad 177;
```

```
bgp {
    group external-peers {
        type external;
        export send-direct;
        neighbor 192.168.0.3 {
            peer-as 22;
            export add-med-110;
        }
        neighbor 192.168.0.7 {
            peer-as 86;
            export add-med-177;
```

- Apstra Freeform
  - 初期セットアップ手順
  - Config Templates
  - **Apstra Telemetry**
  - オプション機能
- 参考リンク

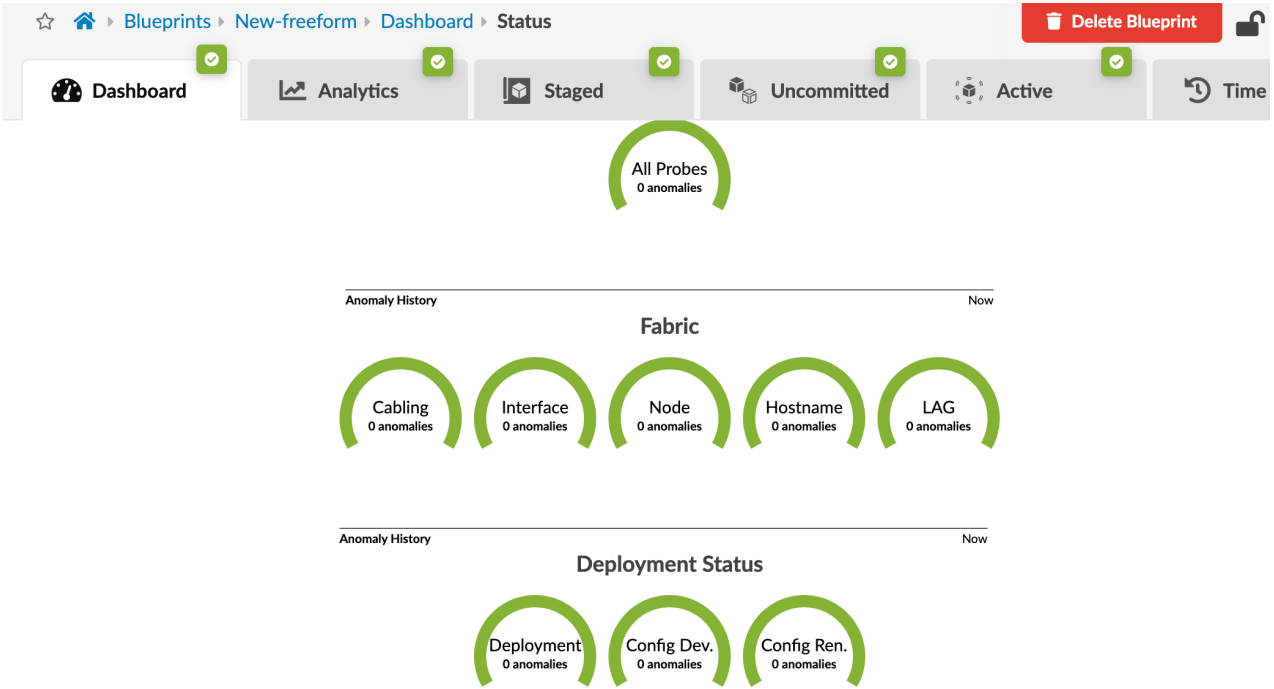
Telemetry Realtime Analytics for Freeform概要

“Dashboard”-ネットワーク全体の健康状態の把握
”Analitics”-BGPセッション、memory, cpuなど各種各種パラメーターを取得可能



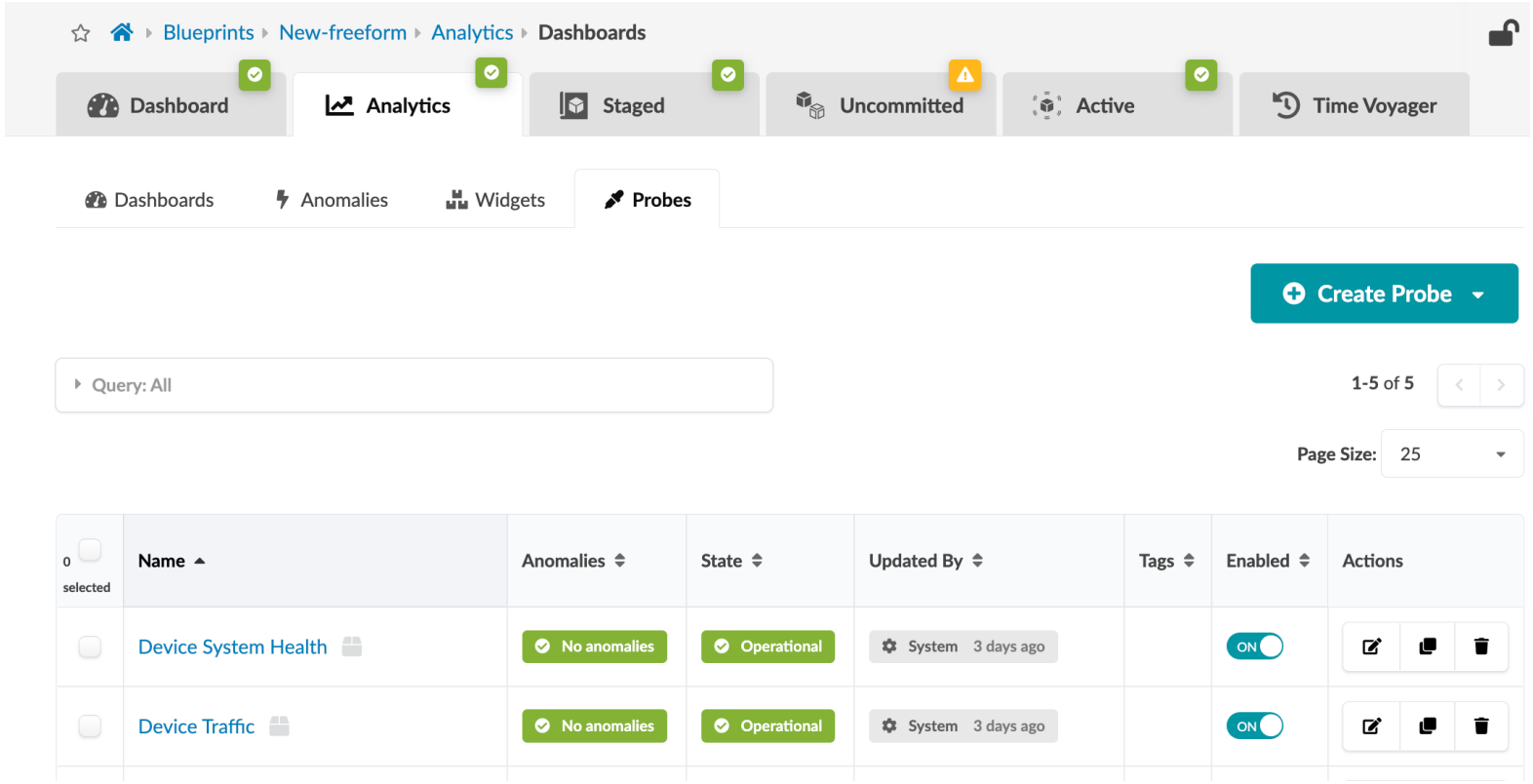
Default Telemetry

DC reference architectureと同じく“Dashboard”でネットワーク全体の健康状態を確認。
但し、デフォルトで用意されている監視項目(Generic System Connectivity, Liveness, Route Verification)についてはFree formではDefault Telemetryの対象外となる(4.1.1現在)



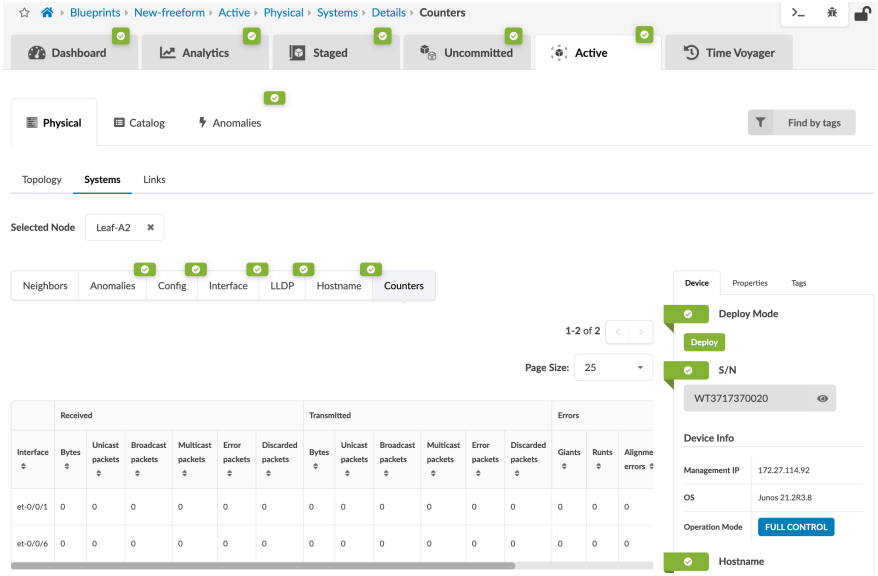
IBA Probes

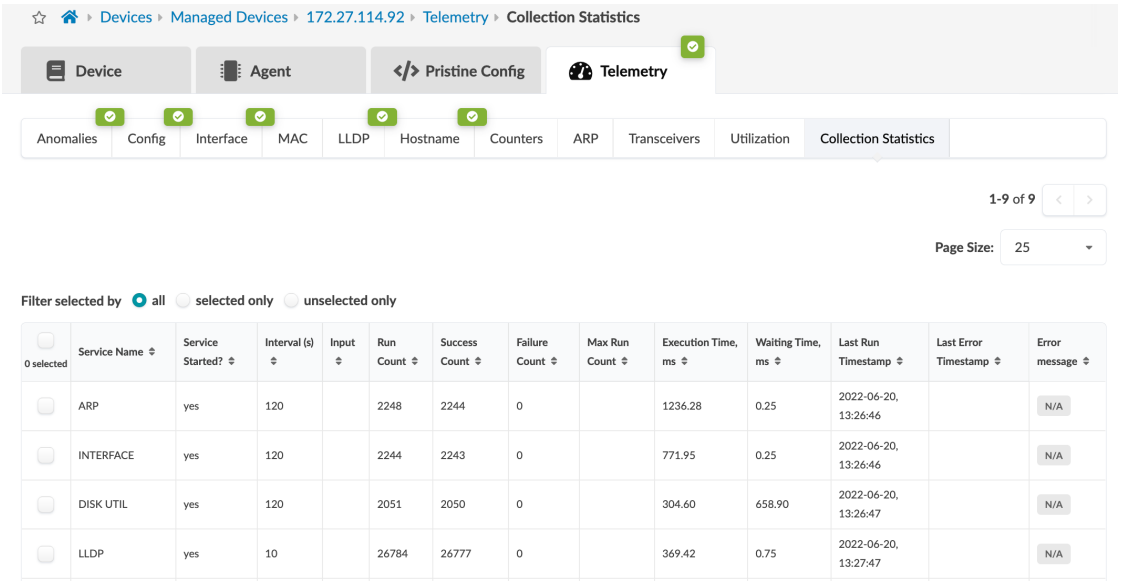
DC reference architectureと同じく、Analytics > Probesで取得必要な各ステータスの取得をすることも可能。但し、利用トポロジーなどによって利用できないProbesもあり。



Device Telemetry確認

Blueprint > Active > Physical 、Devices > Managed Devices > 該当デバイス > Telemetry で該当デバイスを選択することでConfig, Interface Counters等の状態確認可能。





- Apstra Freeform
  - 初期セットアップ手順
  - Config Templates
  - Apstra Telemetry
  - **オプション機能**
- 参考リンク

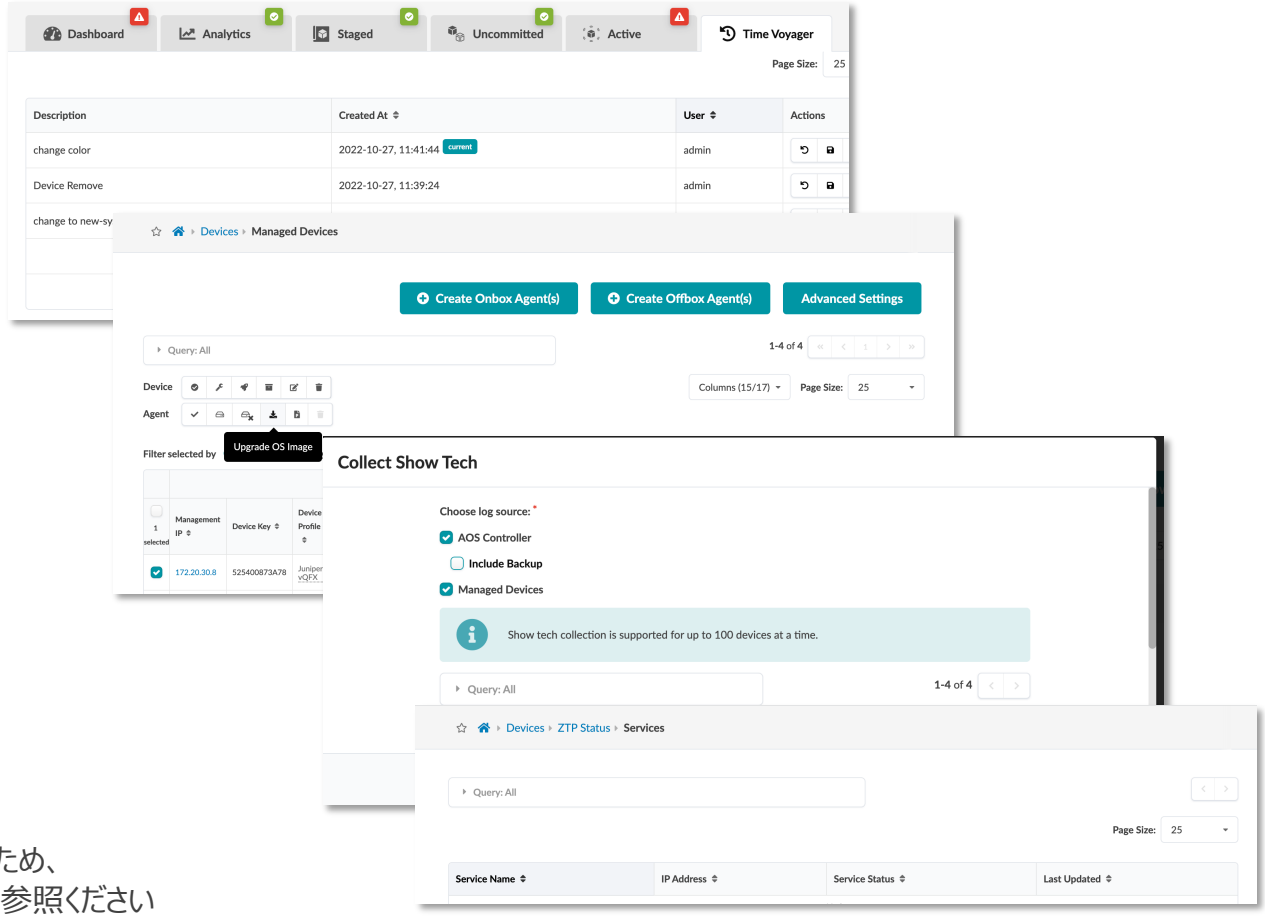
多様な運用オプション機能

Time Voyager (Roll Back)

Device OS Upgrade

Collect Show Tech

ZTP

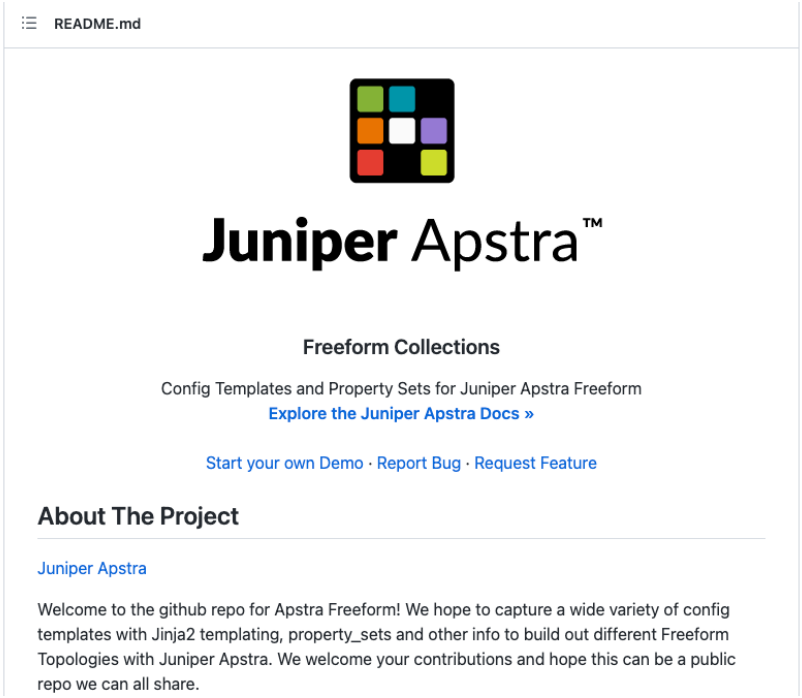


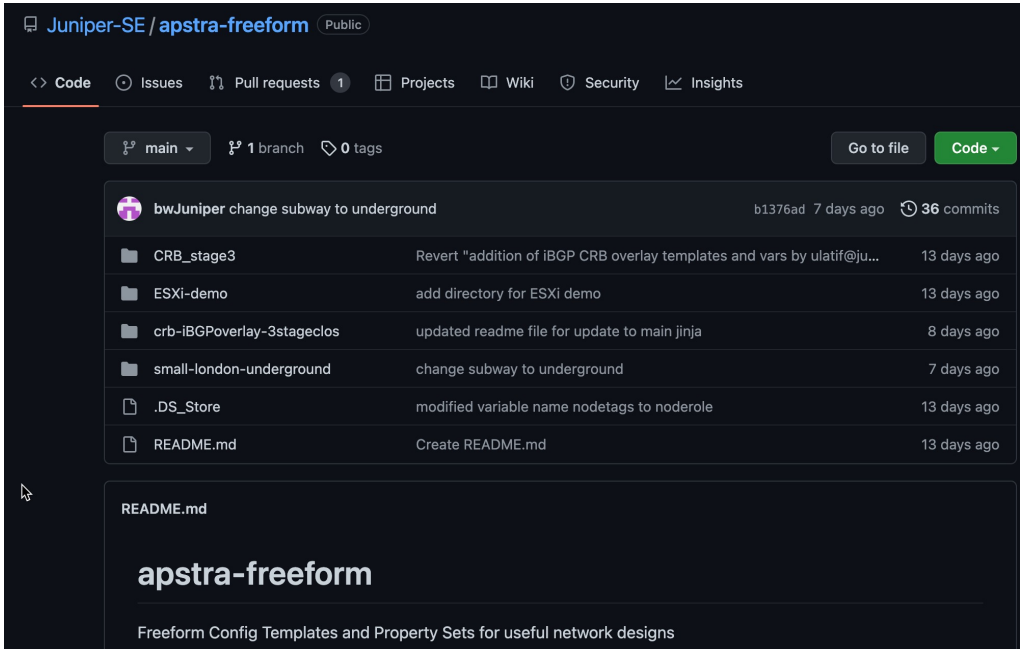
※これらのオペレーション方法はDC Referenceと同様のため、
オペレーション方法はApstra簡易オペレーションガイドを参照ください

- オペレーション概要
  - 初期セットアップ
  - Config templates
  - 可視化/Apstra Telemetry
  - オプション
- 参考リンク

Freeform サンプル on Github

以下にJuniperのSE作成のConfig TemplatesとProperty Setsのサンプル の公開
リンク：https://github.com/Juniper-SE/apstra-freeform





Let’s try Apstra Freeform on CloudLabs

Juniper Apstra CloudLabs™

Table of Contents

Lab Guide - Apstra Freeform

Product SE Team – Version 0.3

Juniper Apstra™

Estimated Time: 60 minutes.

Objective

※CloudLabsの利用希望がある場合は、Juniper社員にご連絡お願いします。
　現在の利用期限は2週間です。

参考リンク

- Apstra Documentation
  - https://www.juniper.net/documentation/product/us/en/apstra
- Freeform Jinjaテンプレートサンプル
  - https://github.com/Juniper-SE/apstra-freeform
- Apstra Freeform Technical Documentation In Focus 4.1.1
  - https://www.juniper.net/documentation/us/en/software/apstra4.1/infocus-apstra-freeform-4.1.1
- Cloud Lab Guide – Apstra Freeform
  - https://cloudlabs.apstra.com/labguides/en/labff.html
- 公式トレーニング - APSTRAFF
  - https://learningportal.juniper.net/juniper/user_activity_info.aspx?id=13119
- Jinja Template Designer Documentation
  - https://jinja.palletsprojects.com/en/3.1.x/templates/#base-template
  - https://jinja.palletsprojects.com/en/3.1.x/changes/#version-3-1-2
  - https://ttl255.com/jinja2-tutorial-part-1-introduction-and-variable-substitution/


Thank you
JUNIPER NETWORKS
Driven by Experience™