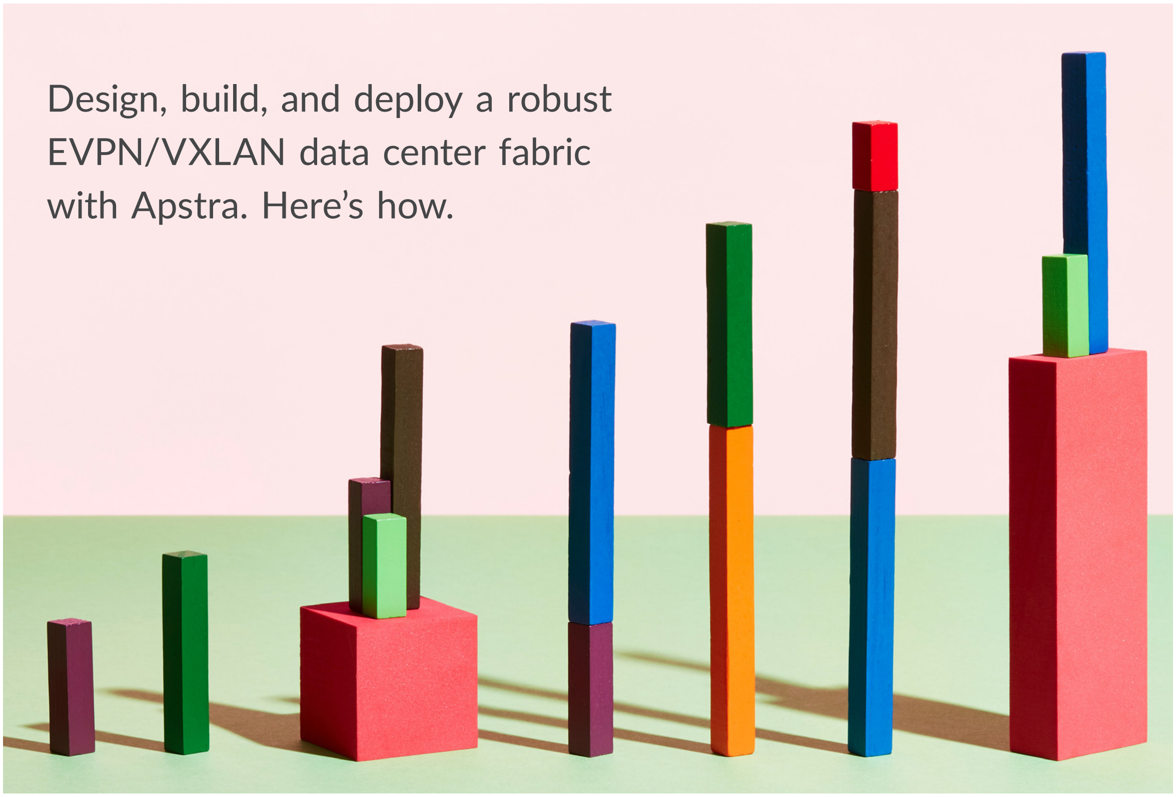
DAY ONE: DEPLOY A DATA CENTER FABRIC WITH APSTRA AND JUNOS®



Design, build, and deploy a robust EVPN/VXLAN data center fabric with Apstra. Here's how.

By Adam Jarvis and Cantemir Olaru

DAY ONE: DEPLOY A DATA CENTER FABRIC WITH APSTRA AND JUNOS®

This book focuses on Apstra, Juniper's intent-based data center networking solution. Apstra takes high-level user objectives (intent) and coverts them into Juniper validated, best practice designs. Device configurations are automatically rendered from the design with no user input required. Once deployed, Apstra leverages closed-loop automation and assurance to provide a complete fabric management solution.

The authors provide a step-by-step design, deploy, and maintenance guide to Apstra. You will build a data center fabric and learn the highlights of intent-based networking. Filled with over 150 illustrations, as well as integration examples with Junos devices and external servers, this *Day One* book takes you right into the lab.

"This marvelous little book fills a much-needed niche for Juniper Apstra: A deep dive into how Apstra works without having to either attend formal training or work your way through the 1000+ page user manual. Presenting a single use case, Adam and Cantemir take you through Apstra from the basics to the subtle details in clear, easy-to-follow examples. If you are considering Apstra, this book gives you a concise first look. If you are a new Apstra user, this book is a perfect guide for coming up to speed."

Jeff Doyle, Member of Apstra Technical Staff

"This book is the go-to resource for deploying fabrics with Juniper Apstra. It covers all the basics that you need to know about this new fabric management tool and its many examples are easy to follow with step-by-step screen captures. If you are new to Apstra you should definitely give this Day One *book a read but it's also a great reference book for the experienced user. I, for one, learned a great deal."*

Christian Scholz, Senior Consultant, Axians

IT'S DAY ONE AND YOU HAVE A JOB TO DO:

- Understand Asptra's design methodology.
- Get to know the Apstra UI.
- Design, build, and deploy your own EVPN/VXLAN data center fabric.
- Perform basic administration tasks on a EVPN/VXLAN fabric via Apstra.

DAY ONE: DEPLOY A DATA CENTER FABRIC WITH APSTRA AND JUNOS

Jarvis, Olaru

Day One: Deploy a Data Center Fabric with Apstra and Junos®

by Adam Jarvis and Cantemir Olaru

JUNIPER NETWORKS

Published by Juniper Networks Books
Authors: Adam Jarvis and Cantemir Olaru
Technical Reviewers: Jeff Doyle, Andy Ford, Jay Wilson, Scott Alexander, Ryan Booth, and Colin Wrightson
Editor in Chief: Patrick Ames
Copyeditor: Nancy Koerbel

Version History: v1, November 2021
2 3 4 5 6 7 8 9 10

About the Authors
Adam Jarvis is a Juniper Networks Systems Engineer based in the United Kingdom. He is new to the technical pre-sales world but despite this is focused on leading network architecture and design initiatives for enterprise customers across the EMEA region. His main interest lies in the data center and network automation space.

Cantemir Olaru is a Consulting Sales Engineer with over 10 years of experience across Juniper Networks TAC, Professional Services, and Technical Pre-sales. He holds a JNCIE-SP certification and for most of his career has been designing and deploying data center networks worldwide for some of Juniper's largest Enterprise and Financial Services customers.

Authors' Acknowledgments
The authors would like to thank the following people, in no particular order: Our former manager Phil Thomas for his support and encouragement throughout, the EMAM Loans Team for providing us with the lab equipment, and Patrick Ames for giving us the opportunity to write this book and his guidance from the beginning. Finally, we'd like to thank our reviewers: Jeff Doyle, Andy Ford, Jay Wilson, Scott Alexander, Ryan Booth, and Colin Wrightson for their valuable contributions, content, input, comments, and support.

Welcome to Day One

This book is part of the *Day One* library, produced and published by Juniper Networks Books. *Day One* books cover Juniper Networks network administration with straightforward explanations, step-by-step instructions, and practical examples that are easy to follow.

- Download a free PDF edition at https://www.juniper.net/dayone.
- Purchase the paper edition at Vervante Corporation (www.vervante.com).

Key Apstra Resources

The authors of this book highly recommend the following Apstra resources, especially the Juniper TechLibrary and its up-to-date information and specifications:

- Apstra Documentation: https://www.juniper.net/documentation/product/us/en/apstra.
- Apstera Product Page: https://www.juniper.net/us/en/products/network-automation/apstra/apstra-system.html.
- *Intent-based Networking for Dummies*: https://www.juniper.net/content/dam/www/assets/ebooks/us/en/intent-based-networking-for-dummies.pdf.

What You Need to Know Before Reading This Book

- The installation of the Apstra virtual machine is out of scope for this book but full deployment details can be found in the Apstra documentation: https://www.juniper.net/documentation/product/us/en/apstra.
- The authors assume that you have used the latest Apstra documentation for hypervisor support and installation instructions.
- You should have a general understanding of EVPN/VXLAN data center designs.
- You should be open to new ideas and approaches to data center design and management.

What You Will Be Able To Do After Reading This Book

- Understand Asptra’s design methodology
- Get to know the Apstra UI
- Design, build, and deploy your own EVPN/VXLAN data center
- Perform basic administration tasks on an EVPN/VXLAN fabric via Apstra

Preface

Data centers have been evolving significantly over the past two decades. Today the evolution is occurring rapidly. As virtualization continues and the use of bare metal servers decreases, many organizations have taken advantage of cloud computing and highly scalable, IP-native applications have emerged.

The industry has also seen a big change in the underlying data center networking technology; the focus has been on moving away from legacy Layer 2 technologies to more scalable Layer 3 IP fabrics. As a result, many organizations still rely on applications and virtualization technology that require Layer 2 connectivity in order to function as designed. This requirement has led to data center networks using underlying IP fabrics with virtual overlay techniques being used to provide Layer 2 connectivity across the network.

To solve this challenge the industry developed both Ethernet VPN (EVPN) and Virtual Extensible LAN (VXLAN), which, when combined, allow for Layer 2 packets to traverse an IP fabric and for MAC learning to be done efficiently using BGP, significantly different from the traditional flood-and-learn techniques. Juniper pioneered this approach, starting with MVPN, which became EVPN and QFabric, and which was pre-EVPN-VXLAN. In the last seven years Juniper has contributed to and embraced the industry's shift to the modern IP open standard architectures.

These advances in the data center mean that networks have become more scalable and efficient, but this comes with a trade-off: complexity. This complexity is three-fold, affecting deployment, operations, and monitoring of a data center network. However, complexity can be tamed using automation and Juniper has been an advocate for automation, employing off-the-shelf automation tools and contributing to open-source projects such as Ansible and Terraform.

This book focuses on Juniper's intent-based data center networking solution, Apstra. The intent-based networking tool takes high-level user objectives (intent) and converts them into Juniper-validated, best practice designs. Device configurations are automatically rendered from the design with no user input required. Once deployed, Apstra leverages closed-loop automation and assurance to provide a complete fabric management solution.

By employing Apstra early in this book users embark on the path of best practice using Juniper pretested prescriptive designs. These three or five-stage Clos fabric designs are constantly reviewed and tested against the newest Juniper hardware and software releases, enabling users to fast-track the technical validation and ensure successful deployments every time. During this design, before deployment, Apstra provides network teams with comprehensive cable plans and the ability to pre-stage the network before receiving the hardware, further expediting the deployment process.

Day 2 operations in the data center are arguably the most important part of the network lifecycle. This is because in the past the network was often seen as a blocker to innovation due to its inability to quickly adapt to the needs of business. Apstra solves this by allowing users to make everyday changes, such as editing hundreds of access ports, in a matter of minutes with a few simple clicks. Apstra can further help streamline networking changes by allowing users to leverage its comprehensive API, meaning it can be worked into existing workflows, tools, and automation processes.

The final part of Apstra's fabric management is monitoring and intent-based analytics. As Apstra understands the user's intent, it understands what the user is trying to achieve and in turn knows how the network should look once deployed. Apstra uses the network intent defined by the user, allowing the system to set up comprehensive monitoring that is self-aware and that self-updates as the fabric's intent changes. This means the monitoring never falls behind the network deployment over time, like in many traditional deployments. Apstra continuously validates changes, alerting teams when changes that might cause brownouts or deviations from the original intent are initiated or detected.

By understanding the user's intent from the outset, Apstra continually self-monitors and self-assures that the data center is functioning as desired, allowing operators to focus on business initiatives that matter, not day-to-day firefighting.

While design, deployment, and Day 2 operations are covered in this book, intent-base analytics are not. The authors recommend "Intent Based Networking for Dummies" by Jeff Doyle, for great insight into Apstra's analytics functions.

Adam Jarvis and Cademir Olaru, November 2021

Chapter 1

Use Cases, Terminology, Testbed

This *Day One* book demonstrates how to design, build, and deploy a complex data center fabric using Juniper's QFX switches and Apstra. The book works through several stages, each one building on the one before. To provide context the book is based on the following example scenario:

- You have been asked to design, build, and deploy a multi-tenant data center that will house a collection of application and database servers, some in a production tenant, some in a backup tenant.
- All hosts in the same tenant and different tenants should be able to communicate with each other. This means both intra- and inter-tenant communication needs to be operational.

Use Cases

This book employs the following use cases:

- *Phase 1: IP Clos Fabric*. The first phase is designing, building, and deploying an IP Clos fabric using Apstra. This is foundational for the whole deployment.
- *Phase 2: Adding Tenants and Virtual Overlays*. Phase 2 introduces a single tenant to the network and adds multiple virtual network overlays to the previously built IP Clos fabric. Although many overlay technologies exist, this book focuses on the one supported by Apstra, VXLAN, with VXLAN being the data plane overlay and EVPN as the control plane protocol. The virtual networks created enable L3 services, allowing communication between the virtual networks in the same tenant.

- *Phase 3: Inter-tenant Routing*. The final phase adds additional tenants and demonstrates routing between tenants using a router external to Apstra. This is achieved by peering the fabric with an external router using border gateway protocol (BGP).

The end goal of this book is to have a fully functioning fabric that contains multiple tenants, each with multiple virtual networks. The fabric will be designed and deployed using Apstra. Figure 1.1 shows a logical diagram of the result: a production database server can communicate with an app server in the backup tenant via the external router.

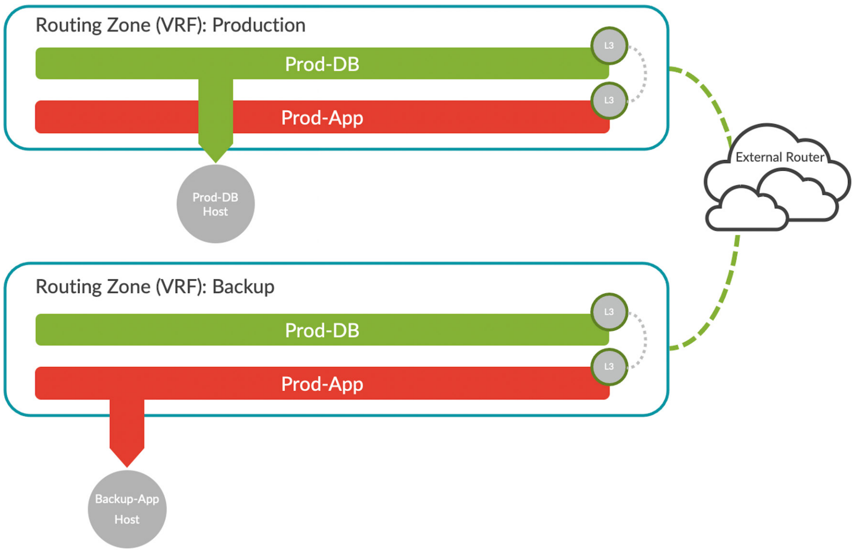


Figure 1.1 Logical Network Diagram Two Tenants Communicating via an External Router

The phases are covered in the book, as are a look at designing the fabric, onboarding switches, and adding the relevant tenants. The lab setup is detailed for those readers following along in a lab. First, however, let's sync our basic data center concepts.

Data Center Terminology

Apstra abstracts much of the complexity from building a multi-tenant data center. However, it is vital to understand the basic concepts and technologies used behind the scenes. For the sake of brevity, these technologies and concepts are not covered in-depth but rather defined at a high level.

MORE? Check out the *Day One* library for books that dive into these concepts at a much deeper level: https://www.juniper.net/dayone.

IP Clos Fabric

The Clos design is a type of non-blocking switching architecture that reduces the number of ports required in an interconnected fabric. A traditional three-tiered network model consists of *core* switches, *aggregation/distribution* switches, and *access* switches. These devices are interconnected for redundancy, which can create loops in the network. As part of the design, a protocol (Spanning Tree) that prevents looped paths is implemented. However, doing so deactivates all but the primary path. A backup path is then only brought up and utilized when the active path experiences an outage.

Network vendors in recent years have tried to overcome the limitations of Layer 2 STP by implementing proprietary solutions to limit the effect spanning tree had on link utilization. These were often hard to troubleshoot, had limited scalability, and usually provided a single point of failure.

With a Clos leaf-spine design, all devices are the same number of hops away and contain a predictable and consistent amount of latency. This is possible because of the new topology design that has only two layers, the leaf layer and spine layer. The leaf layer consists of switches that connect to devices, like servers, firewalls, load balancers, and edge routers. The spine layer interconnects the leaf layer switches to form a network, where every leaf switch is interconnected with each and every spine switch.

To allow for the predictable distance between devices in this two-layered design, dynamic Layer 3 routing is used to interconnect the layers. Dynamic routing allows the best path to be determined and adjusted based on network change. This design allows the use of equal cost multipathing (ECMP), meaning traffic can be load-balanced across all links and STP is eliminated altogether. By design, all paths have an equal path length, so it's the switches responsibility to determine which application session uses which path, by implementing hashing.

Using an all-IP fabric allows for open standards, eliminating vendor lock into a particular solution and a disaggregated architecture where no one box is a single point of failure. Troubleshooting is also made easier as there are no propriety protocols being used. The leaf-spine network design focuses on both east-west network traffic (machine to machine), as well as north-south traffic patterns, enhancing performance for both types of traffic. This new approach is a solution to the intrinsic limitations of the Spanning Tree Protocol and the complexity L2 brings.

As shown in Figure 1.2, the leaf and spine devices are interconnected using high-speed interfaces that are either single links or aggregated Ethernet interfaces.

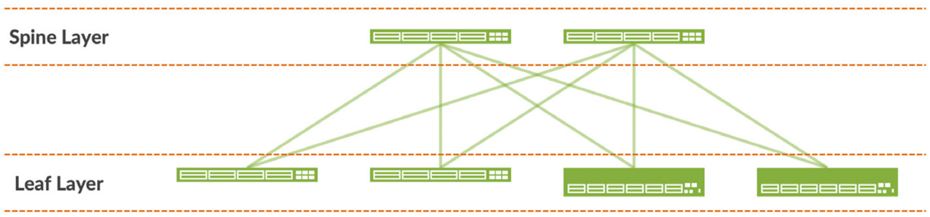


Figure 1.2 Example of Spine and Leaf Switches in a Clos Fabric

Spines, Access Leaf, Border Leaf

Spine/leaf switches in a Clos fabric have some important roles that need to be understood.

Spine Switches

A spine switch does not connect to end-host devices within a data center, it is a centrally located device in an IP fabric topology that has a connection to every leaf device. The role of the spine is to essentially forward IP packets.

Leaf Switches

With an access level network device in an IP fabric topology, end systems connect to the leaf devices. In this blueprint architecture a leaf switch connects to every spine switch within the fabric, allowing for traffic to be load-balanced across the entire fabric. Typically leaf switches are TOR switches in a data center.

Border Leaf Switches

This device typically has the role of providing a connection to one or more external devices, for example, multicast gateways or data center gateways, that provide additional functionality to the IP fabric. The border leaf switch is the same as an ordinary leaf switch, however, it takes on the border leaf role to provide external connectivity. For example, border leaf switches provide connectivity to data center gateways for DCI, firewall services, and a DHCP server as shown in Figure 1.3.

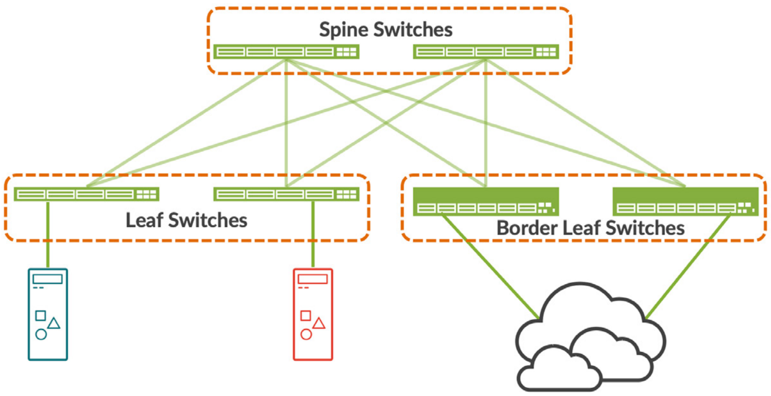


Figure 1.3 Switches in a Clos Fabric

BGP

Border Gateway Protocol (BGP) is an Exterior Gateway Protocol (EGP) used to exchange routing information among routers in different autonomous systems (ASs). BGP routing information includes the complete route to each destination and uses the routing information to maintain a database of network reachability information, which it exchanges with other BGP systems. Most modern data center fabrics use BGP to interconnect the fabric.

Multiprotocol BGP (MP-BGP) is an extension to the original BGP standard and enables BGP to carry routing information for multiple network layers and address families such as EVPN.

EVPN

EVPN (Ethernet VPN) is a control plane protocol for distributing MAC reachability, while reducing the traditional *flood-and-learn* mechanism observed in Ethernet networks. EVPN is based on BGP and inherits its vast scalability and multi-tenancy features. EVPN is a standards-based technology that provides virtual multipoint bridged connectivity between Layer 2 domains over an IP fabric.

VXLAN

VXLAN (Virtual Extensible LAN) provides connectivity for legacy applications that rely on L2 connectivity over the IP fabric, providing a bridge to connect heterogeneous workloads (bare metal or virtualized) that require L2 connectivity. VXLAN introduced an overlay scheme that expands the Layer 2 network address space from 4000 VLANs to theoretical maximum of 16 million virtual networks, largely solving the scaling issues seen in VLAN-based environments. VXLAN tunnelling is a data plane protocol that encapsulates Layer 2 Ethernet frames in Layer 3 UDP packets. This encapsulation enables Layer 2 subnets or segments to span physical Layer 3 networks.

Why Use EVPN and VXLAN In the Data Center?

An EVPN/VXLAN DC provides the ability to run one or more L2 overlays across a L3 IP network. It provides the control plane required to route between virtual segments while maintaining independence from the physical underlay network. Within that virtual network, applications can be separated, effectively creating a multi-tenant domain managed through the administration of distributed network policy. In short, the benefits of these technologies are as follows:

- EVPN reduces the number of protocols needed to operate the data center network, making it simpler and more reliable, lowering operating costs, and improving service availability.
- EVPN removes the need to flood broadcast, unknown unicast, and multicast (BUM) traffic through stretched Ethernet segments. Flood-and-learn is a long-established traditional mechanism used for learning the whereabouts of a required destination host. Flood-and-learn works but is generally not efficient at scale. EVPN suppresses ARP, eliminating unnecessary traffic replication, improving scalability, and reducing traffic noise for more efficient monitoring and troubleshooting, leading to higher service availability.
- EVPN provides support for multi-homed attachment at both L2 and L3 with fast-failure recovery, allowing more efficient use of resources by eliminating standby links. This removes the requirement for the proprietary MC-LAG redundant host connectivity solution, provided by all vendors.
- VXLAN can be utilized to move L2 frames across an IP fabric. This means traffic can use every link in the DC and L2 protocols such as STP can be eliminated.

What is a Tenant?

A tenant is a user community (such as a business, department, or application) that contains groups of endpoints. Groups may communicate with other groups in the same tenancy and tenants may communicate with other tenants if permitted by network policies. A group is typically expressed as a subnet (VLAN/VXLAN) that can communicate with other devices in the same subnet and reach external groups and endpoints by way of routing. Later in the book tenants will be created. It is important to note that a tenant in this context is a Virtual Routing and Forwarding instance (VRF).

Which Juniper Routing Models Does Apstra use?

In the context of using Apstra to deploy a data center there are two routing models to be aware of, Edge-Routed Bridging (BO, ERB) and external routers.

BO

A bridged overlay (BO) provides Ethernet bridging between leaf devices in an EVPN network. This overlay type simply extends VLANs between the leaf devices across VXLAN tunnels and does not provide any inter VXLAN routing.

ERB

ERB is a routing model that allows leaf switches to perform routing, permitting different virtual networks in the same tenant to communicate.

External Routers

External routers are used when hosts in different tenants need to communicate. Because tenants exist in different VRFs in the fabric, an external router is required to allow this communication. An external router is external to the data center fabric.

MORE? Dive deeper into these technologies with Juniper's Design and Architecture Guides. Data Center EVPN-VXLAN Fabric Architecture Guide: https://www.juniper.net/documentation/en_US/release-independent/solutions/information-products/pathway-pages/sg-005-data-center-fabric.pdf. Juniper Apstra Architecture Guide: https://www.juniper.net/content/dam/www/assets/white-papers/us/en/2021/juniper-apstra-architecture.pdf.

Testbed Overview

The testbed used to create the targeted multi-tenant DC fabric consists of two spine switches and four leaf switches, all connected in a folded Clos topology. Figure 1.4 shows the physical topology including the out of band network. It is comprised of:

Hardware:

- 2 x Juniper QFX5120-32C – Acting as spine switches
- 2 x Juniper QFX5120-48Y – Acting as leaf switches
- 2 x Juniper QFX10003-36Q – Acting as leaf switches
- 1 x Juniper SRX1500 – Acting as an external router

Software:

- Apstra Server – Release 4.0.0
- Apstra ZTP Server – Release 2.0.0
- Juniper QFX5120-32C – Junos Release 19.1R1.6
- Juniper QFX5120-48Y – Junos Release 20.4R2.7
- Juniper QFX10003-36Q – Junos Release 20.4R2.7
- Juniper SRX1500 – Junos Release 15.1X49

NOTE Apstra is not a physical appliance, it is a software-only package that can run on either VMware's ESXi, Hyper-V, or KVM as a virtual machine.

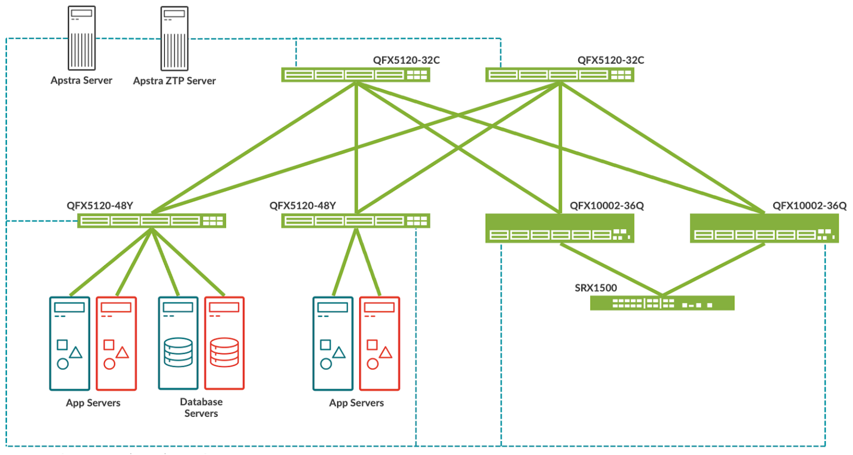


Figure 1.4 This Book's Physical Network Topology

Please note that every switch in this topology is connected to an out of band management (OOBM) network in the same VLAN / L2 broadcast domain. The out of band network connects the Apstra server, the Apstra ZTP server, and the management ports on each switch. Apstra will use this OOBM network to communicate with the devices through their entire lifecycle.

NOTE When using the ZTP server to on-board devices, the devices are not required to be in the same L2 domain. However a device that can replay DHCP requests to the ZTP server will be required in that scenario.

NOTE Apstra uses the management VRF on Junos devices, which is not supported in older Junos software versions. Please refer to the latest Apstra documentation for supported Junos software versions.

The overall goal of this book is to create two separate tenants in the fabric, *production* and *backup*. These two tenants will have two virtual networks assigned, one for database servers (DBs) and the other for application servers (Apps).

Intra-tenant communication is enabled by default if virtual networks are assigned IP gateways when deploying this fabric using Apstra. *Inter*-tenant routing, on the other hand, is not allowed and an external system such as a firewall or router is needed to allow communication between tenants. Figure 1.5 illustrates a logical representation of the Juniper SRX1500 being used as an external router with the fabric and the SRX1500 peer using BGP and exchange routes. The topology permits tenants to communicate with each other even though they run in separate routing instances within the fabric.

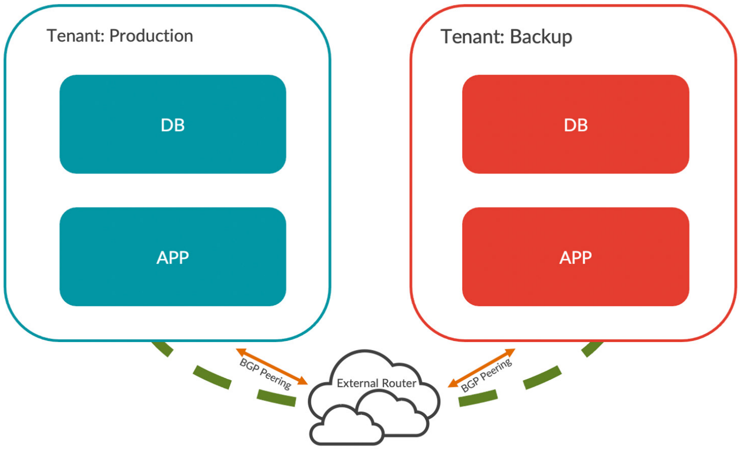


Figure 1.5 Testbed Logical Diagram

Okay, it's day one. Let's start to deploy an Apstra multi-tenant data center beginning with its unique design.

Chapter 2

Apstra Data Center Design

Building an EVPN/VXLAN data center requires several steps that need to happen in a carefully planned sequence, especially when the devices are being managed via the CLI. As networks scale and change over time, ensuring the network documentation, device configs, and monitoring are all in sync can present many challenges to network administrators.

Apstra's approach to running the data center is different from how most data centers are run today. Apstra looks at the DC fabrics as a single or set of systems and not a collection of boxes managed in isolation. This is fundamentally different from many currently deployed DC fabrics.

Apstra abstracts much of the deployment and *Day 2* complexities of a DC fabric, validating, and modeling the user's intent so businesses have a repeatable template that can be used to deploy a single DC fabric or even multiple DC fabrics in the exact same way.

High Level

Designing a DC fabric in Apstra starts from one mandatory point, the user's intent. These general expressions of intent are captured by Apstra and converted into a fully-fledged fabric design based on one of Apstra's built-in reference architectures. These inputs from the designer must be structured and abide by certain rules. A reference design is the underpinning foundation of the overall DC fabric design, enforcing rules and best practices throughout every stage.

NOTE *Intent-Based Networking* (IBN) is a term originally used by Apstra to describe a software system to produce a desired outcome (a complex 3- or 5-stage EVPN-VXLAN DC) rather than using the CLI syntax to produce that outcome. It's essentially a single system that provides an abstraction that can be used to design, build, deploy, and operate your DC fabric or fabrics.

This book focuses on a L3 Clos fabric with an EVPN/VXLAN overlay and the refence design being used by Apstra reflects this. The workflow steps to design such an IBN fabric with Apstra, shown in Figure 2.1, are:

- The initial step of the design phase models the type of switches that will be used. These models created are *logical devices*; they dictate port count, port speed, and how the ports can be used. This is an important concept, as it allows users to define switches as abstractions of what are required in a network. Apstra then allows users to select among a list of multiple vendors' switches that comply with the defined specifications. Thus, from the beginning a network is designed to the businesses needs and technical requirements. Vendor/model selections can be completed later.
- Once logical devices are defined, racks are designed to house leaf switches and the associated compute, storage, and even network devices external to the Clos fabric.
- Individual rack types are modeled according to their functional role and the various racks that are part of the DC design are bound together through a DC template. It is common in a data center to have a common rack type through-out, meaning in Apstra you design a rack once and that design can be used over and over again. The DC template is where the number of spine devices are assigned, the protocols of choice are defined, and the type of network is chosen.

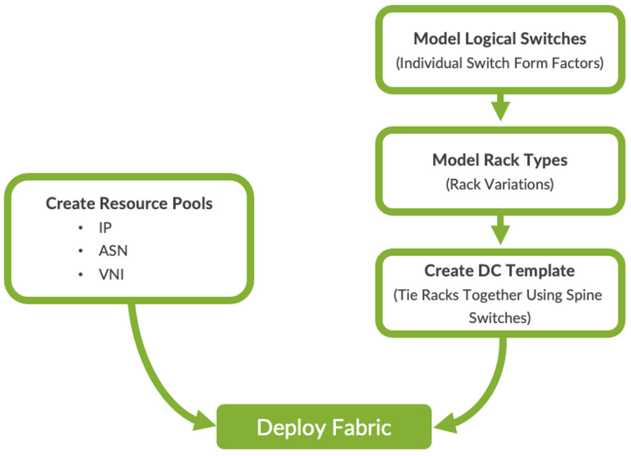


Figure 2.1 Logical Workflow for Designing a Fabric with Apstra

The rest of this chapter dives deeper into each stage of the design process, and it concludes by discussing resource pools and how they are created. While not strictly part of the design process, they are needed when it comes to deploying a blueprint. Blueprints are the instantiation of a template that a user creates and represents a DC fabric; they are discussed in subsequent chapters. It's important to point out that templates become user-defined standardized designs in Apstra, meaning the same templates can be used to deploy multiple, identical data center fabrics.

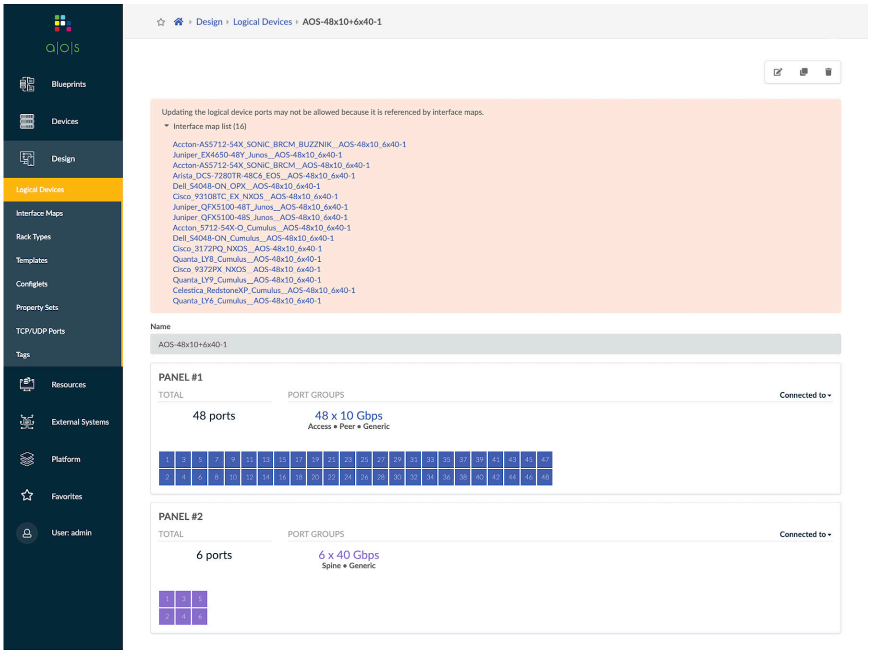
IMPORTANT Once again the authors assume you have Apstra installed, are familiar with the user interface, and have read the documentation, which is quite excellent: https://www.juniper.net/documentation/product/us/en/apstra.

Logical Devices (LDs)

Let's begin with logical devices. It is the first concept that needs to be understood in the Apstra design phase. A logical device represents the abstraction of a physical switch, regardless of its network vendor. Thus, when logical devices are used it means DC fabrics can be modeled in a vendor-neutral way, defining port count, speeds, and how ports on a switch are consumed. It permits the DC Fabric to be modeled before making a final hardware choice. Apstra ships with many predefined logical device models, and additional logical devices can be added by the user.

An example of a logical device in Apstra is a 48x10GE switch with 6x40GE uplinks. This LD ships with Apstra by default and is called *AOS-48x10+6x40-1*, as displayed in Figure 2.2. This particular logical device model maps to many different hardware types (device profiles) such as the Juniper QFX5100-48S, QFX5120-48Y, and even the EX4650-48Y, as well as other vendor's devices.

Logical devices can be thought of as placeholders during the design phase. Using the selected logical device models in the design phase allows for the remainder of the DC to be designed, and then during actual deployment (discussed in later chapters) they can be mapped to physical hardware platforms.



Updating the logical device ports may not be allowed because it is referenced by interface maps.

Interface map list (16)

Accton-AS5712-54X_SONiC_BRCM_BUZZNIK__AOS-48x10_6x40-1
Juniper_EX4650-48Y_Junos__AOS-48x10_6x40-1
Accton-AS5712-54X_SONiC_BRCM__AOS-48x10_6x40-1
Arista_DCS-7280TR-48C6_EOS__AOS-48x10_6x40-1
Dell_S4048-ON_OPX__AOS-48x10_6x40-1
Cisco_93108TC_EX_NXOS__AOS-48x10_6x40-1
Juniper_QFX5100-48T_Junos__AOS-48x10_6x40-1
Juniper_QFX5100-48S_Junos__AOS-48x10_6x40-1
Accton_5712-54X-O_Cumulus__AOS-48x10_6x40-1
Dell_S4048-ON_Cumulus__AOS-48x10_6x40-1
Cisco_3172PQ_NXOS__AOS-48x10_6x40-1
Quanta_LY8_Cumulus__AOS-48x10_6x40-1
Cisco_9372PX_NXOS__AOS-48x10_6x40-1
Quanta_LY9_Cumulus__AOS-48x10_6x40-1
Celestica_RedstoneXP_Cumulus__AOS-48x10_6x40-1
Quanta_LY6_Cumulus__AOS-48x10_6x40-1

Name: AOS-48x10+6x40-1

PANEL #1

TOTAL	PORT GROUPS	Connected to
48 ports	48 x 10 Gbps Access • Peer • Generic	

PANEL #2

TOTAL	PORT GROUPS	Connected to
6 ports	6 x 40 Gbps Spine • Generic	

Figure 2.2 Logical Device Example

It is common for different ports on a switch to be designated for specific roles such as ports that are only designated to be connected to spines. Apstra allows for this strict port designation to be enforced. Table 2.1 lists the different port roles available in Apstra's LDs.

Table 2.1 Logical Device Port Roles

Port Roles	
Spine	Port is configured to face a spine device.
Super Spine	Port is configured to face a super spine device. Applies to 5-stage Clos data center fabric only.
Leaf	Port is configured to face a leaf device.
Access	Port is configured to face an access switch.

Generic System	A port where a generic device can be attached. A few examples of what a generic device could be include a firewall, an external router, a bare metal server, and a load balancer. In other words, a Generic System is anything that is directly attached to an Apstra-controlled fabric but is not itself controlled by Apstra. Within the Apstra UI, a Generic System will be rendered within a rack, and an external Generic System will be rendered outside of a rack.
Peer	Port is configured as a peer link between two leaf devices. Please note that the "Peer" port role is not supported on Junos Leaf switches.
Unused	Apstra does not attempt to render configuration on a port with the unused role (a dead port).

When modeling a LD, those ports that are similar in role and speed are grouped together into *port groups*. Apstra allows for port groups to be assigned multiple port roles, which can be seen in Table 2.1. The 10GE ports are assigned access, peer, and generic system.

It's important to understand that this LD simply represents the port count, port speed, and how each port can be used. Later in the book interface maps (IMs) and device profiles (DPs) are discussed. Figure 2.3 illustrates, at a high level, the association between logical devices, interface maps, and device profiles. A single LD can be mapped to a number of DPs through their respective IMs.

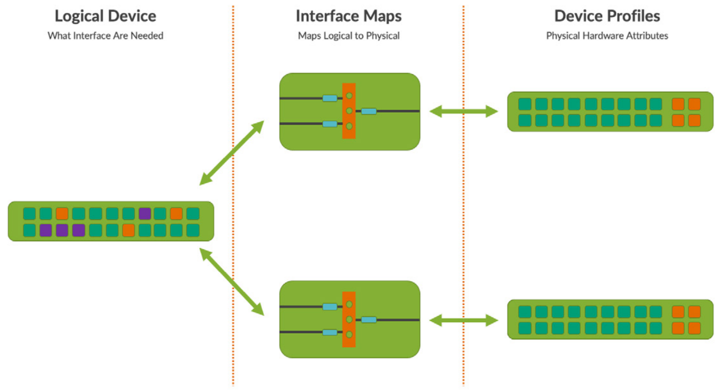


Figure 2.3 LD / IM / DP Relationship

So, to create the intended DC Fabric design for this book (See Figure 1.4) different logical device models are required to complete the design. One of these ships with Apstra, the other two need to be created. Table 2.2 lists the two logical devices that need to be created.

Table 2.2 Logical Devices Information for Creation

Logical Devices		
Name	Port Groups	Role(s) – Connects to
DO_48x1_8x40		
	48 x 1GE	Generic System
	8 x 40GE Ports	Spines
DO_72x10_18_40		
	72 x 10GE Ports	Generic System
	14 x 40GE	Generic System
	4 x 10GE	Spines

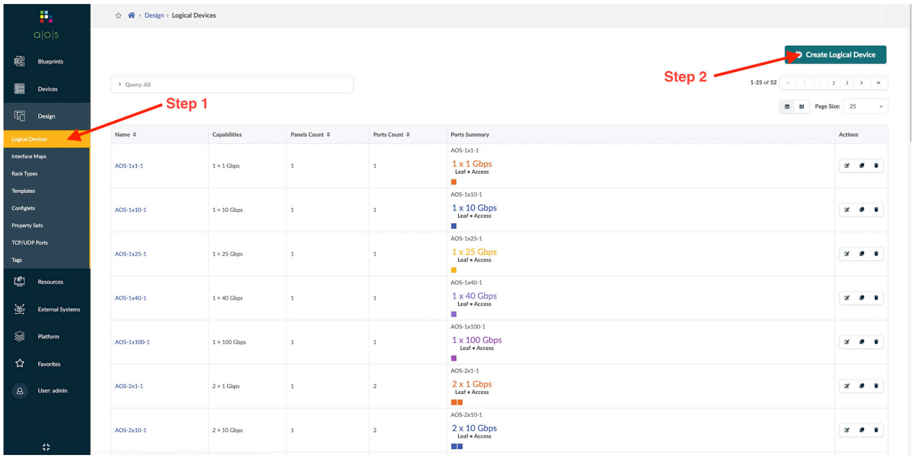
The remainder of this section shows you how the Logical Device DO_48x1_8x40 is created to demonstrate the process. Fire up Apstra and let's get into the lab.

Create a Logical Device

Creating logical devices is a relatively straightforward process. Again, we're assuming you have read the documentation (https://www.juniper.net/documentation/product/us/en/apstra) and understand the Apstra UI.

Step 1: Go to Design > Logical Devices. (Figure 2.4)

Step 2: Click Create Logical Device.



Name	Capabilities	Panels Count	Ports Count	Ports Summary	Actions
AOS-1x1-1	1 × 1 Gbps	1	1	AOS-1x1-1 1 x 1 Gbps Leaf • Access	
AOS-1x10-1	1 × 10 Gbps	1	1	AOS-1x10-1 1 x 10 Gbps Leaf • Access	
AOS-1x25-1	1 × 25 Gbps	1	1	AOS-1x25-1 1 x 25 Gbps Leaf • Access	
AOS-1x40-1	1 × 40 Gbps	1	1	AOS-1x40-1 1 x 40 Gbps Leaf • Access	
AOS-1x100-1	1 × 100 Gbps	1	1	AOS-1x100-1 1 x 100 Gbps Leaf • Access	
AOS-2x1-1	2 × 1 Gbps	1	2	AOS-2x1-1 2 x 1 Gbps Leaf • Access	
AOS-2x10-1	2 × 10 Gbps	1	2	AOS-2x10-1 2 x 10 Gbps Leaf • Access	

Figure 2.4 Creating Logical Devices

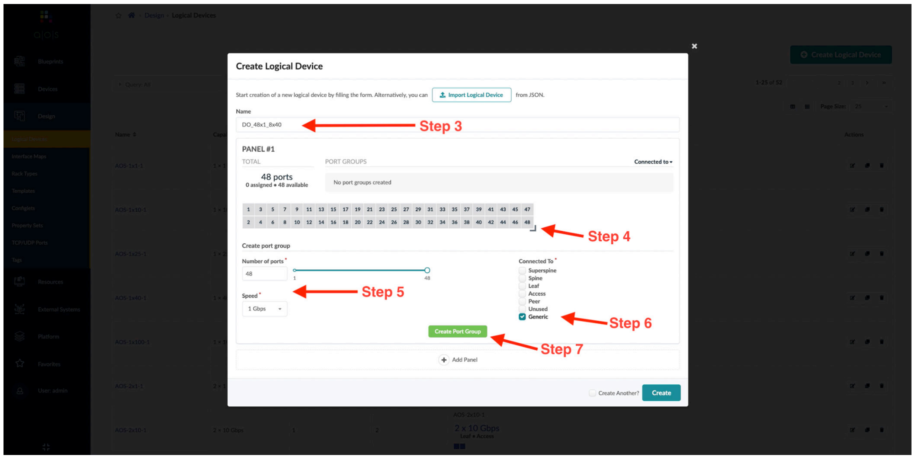
Step 3: Name the Logical Device. (Figure 2.5)

Step 4: Click on a port and while holding the mouse button down slide/drag the cursor until the number of ports required are highlighted and then release the mouse button; this will be a port group (a port group is a group if ports that share the same characteristics).

Step 5: Select the port speed required.

Step 6: Select the port role (what these ports will connect to on the other end of the link).

Step 7: Click Create Port Group.



Start creation of a new logical device by filling the form. Alternatively, you can Import Logical Device from JSON.

Name: DO_48x1_8x40

PANEL #1

TOTAL: 48 ports, 0 assigned • 48 available

PORT GROUPS: No port groups created

Create port group

Number of ports: 48

Speed: 1 Gbps

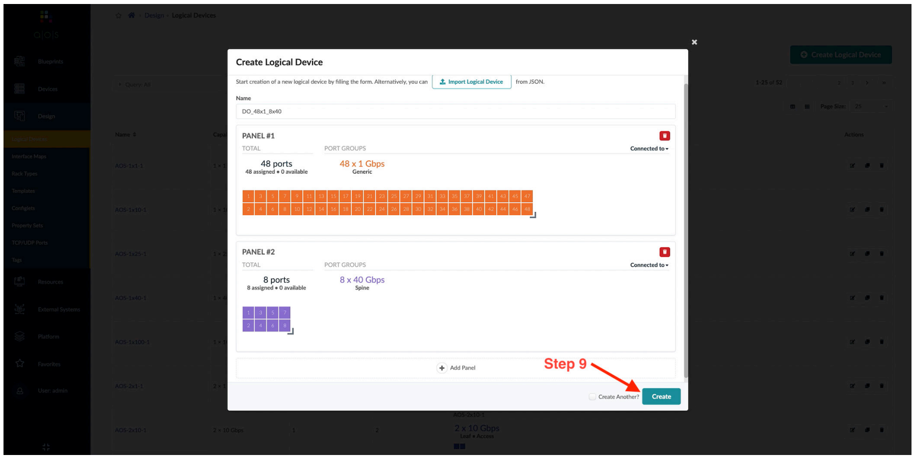
Connected To: Superspine, Spine, Leaf, Access, Peer, Unused, [✓ Generic]

Create Another?

Figure 2.5 *Naming Logical Devices*

Step 8: Add additional panels if required, as per this example one additional panel needs to be added (Figure 2.6).

Step 9: Once all additional panels are added click Create.


Create Logical Device

Start creation of a new logical device by filling the form. Alternatively, you can Import Logical Device from JSON.

Name: DO_48x1_8x40

PANEL #1
TOTAL: 48 ports, 48 assigned • 0 available
PORT GROUPS: 48 x 1 Gbps, Generic

PANEL #2
TOTAL: 8 ports, 8 assigned • 0 available
PORT GROUPS: 8 x 40 Gbps, Spine

Figure 2.6 Adding Additional Panels

As you can see, the steps for creating a logical device are relatively simple. Also remember that, by default, Apstra ships with many industry standard device formats.

NOTE This process is made a little easier by finding, cloning, modifying, and renaming an LD that's similar to the one needed in your *Blueprint* (to be discussed shortly).

Racks

Okay, we just finished learning how to model individual switch variants, the basic building block for the entire design phase. In a conventional DC those leaf switches would sit within a rack and would be connected to various pieces of infrastructure such as compute, storage, or other networking devices, as well as connecting to spine devices via the uplinks of the switch.

Rack types define multiple entities including leaf switches and generic systems. They could, for example, be connecting servers. Initially, the connectivity design must be chosen so Apstra knows the target architecture of the fabric. These options are listed in Table 2.3.

NOTE This book only focuses on L3 Clos designs. To find out about more fabric design options please visit the latest Apstra documentation: https://www.juniper.net/documentation/product/us/en/apstra.

Table 2.3 Fabric Design Types

Fabric Connectivity Design	Description
L3 Clos	Used for either a 3 or 5 stage Clos fabric
L3 Collapsed Model	Collapsed fabric

Once the fabric design option is chosen the leaf switches and generic server connectivity can be defined.

Leaf Switches

Leaf switches, also referred to as Top of Rack (ToR) switches, provide connectivity down to end hosts and up to the fabric spines. Logical devices are used to represent individual leaf switches during the design phase. Once a rack type has been modelled, Apstra performs some sense checking to validate that the connectivity required matches what can be provided by the logical device, such as ensuring that there are enough switch ports to connect all the specified servers, and that the port speeds are correct. This is the very first validation that Apstra performs.

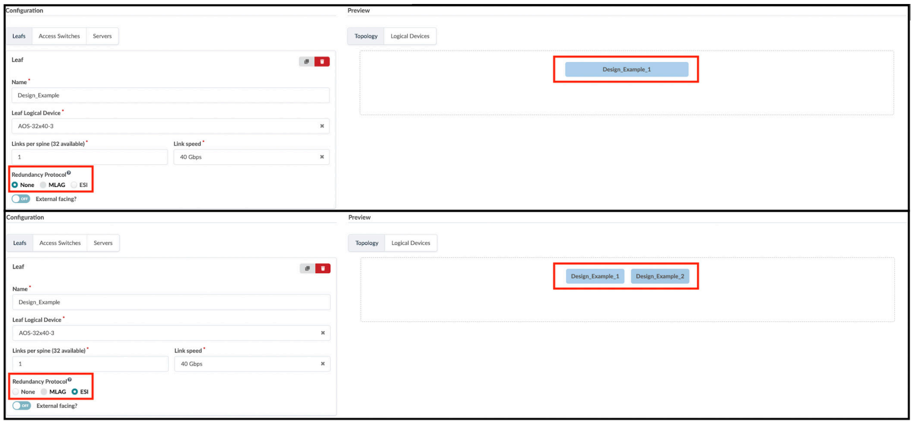
When creating a leaf switch design inside a rack, the main design element that needs to be addressed, other than choosing the correct logical device, is the redundancy requirements. Apstra provides multiple redundancy options to enable the fabric to suit the needs of the network as shown in Table 2.4.

NOTE When designing racks, make sure the intended platform supports the chosen redundancy protocol, otherwise the fabric deployment will fail. Access switch models will not be covered in this book; please refer to the Juniper Apstra documentation for more information: https://www.juniper.net/documentation/product/us/en/apstra.

Table 2.4 Leaf Switch Redundancy Options

Redundancy Protocol Options	Description
None	No redundancy protocol is used, and end devices will only connect to one leaf.
MC-LAG **Currently not supported by the Junos OS.	For dual-homed server connections where both switches act as the same logical device.
ESI **Currently ESI is only supported by the Junos OS.	Uses EVPN Ethernet Segment IDs which are assigned to interfaces on leaf switches that face the same end host. To the end host, much like in MC-LAG, it believes it is speaking to a single upstream device.

The choice in redundancy protocol will dictate how many leaf switches are required within an Apstra rack model. For example, using no redundancy options means only a single leaf is required, however ESI will require two leaf switches. When the user selects the desired redundancy model Apstra automatically adds the correct number of switches required to fulfill this. Figure 2.7 shows an example of this in Apstra.



Leaf

Name *: Design_Example

Leaf Logical Device *: AOS-32x40-3

Links per spine (32 available) *: 1

Link speed *: 40 Gbps

Redundancy Protocol: [✓ None] MLAG ESI

External facing? OFF

Design_Example_1

Leaf

Name *: Design_Example

Leaf Logical Device *: AOS-32x40-3

Links per spine (32 available) *: 1

Link speed *: 40 Gbps

Redundancy Protocol: None MLAG [✓ ESI]

External facing? OFF

Design_Example_1 Design_Example_2

Figure 2.7 No Redundancy Versus ESI Lag

It's important to note that multiple leaf switches can be added to a single rack design if required. However, adding multiple redundant leaf pairs to the same rack does NOT allow for N-way ESI-LAG.

Server Connectivity

The other task when designing a rack is defining server connectivity to the leaf switches and ultimately the DC fabric.

Server connectivity is fairly simple, and a rack may have several types of servers, all connected in different ways. Similar to leaf switches, when adding servers, the base component is selecting a logical device that models the server connectivity. For example, Apstra models several different server type devices that have varying interface speeds and ports. If the particular variant isn't available by default, you can create one.

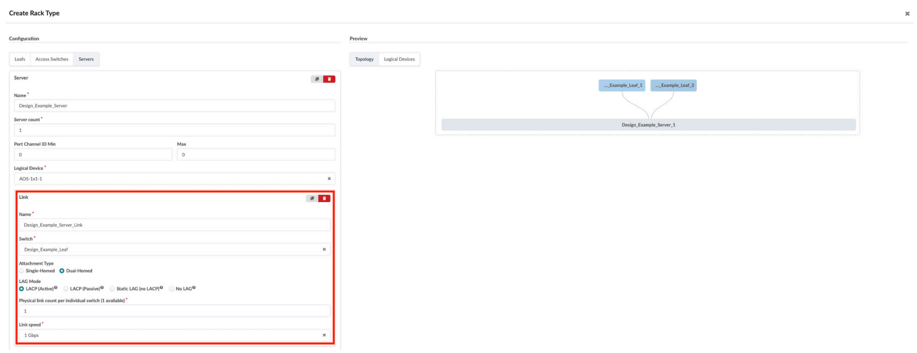
Once the LD for a group of servers has been chosen, the number of servers, as well as the desired connectivity, needs to be defined. The connectivity options will vary depending on the leaf redundancy options chosen as discussed previously. Table 2.5 lists example connectivity choices based on the leaf switch redundancy type chosen and Figure 2.8 shows how a server would be dual homed when using ESI as the redundancy choice.

Typically, all servers in a rack are of the same type. This means users only have to specify the server details once, and then tell Apstra how many are required in the rack. If multiple server types are needed in a single rack, such as some dual homed and some connected to only one ToR, you can specify the multiple types and again just tell Apstra how many of each you want in the rack.

NOTE Defining servers at this stage of the design is not a requirement. Once a blueprint has been created the user can add servers one at a time and remove any that were created during this design phase.

Table 2.5 Leaf Redundancy Types

Leaf Redundancy Type	Server Link Options
None	• LAG Mode Active Passive Static (no LACP) None • No. of Uplinks • Uplink Speed
ESI	•Attachment Type Dual Homed Single Homed • LAG Mode Active Passive Static (no LACP) None • No. of Uplinks • Uplink Speed


Create Rack Type

Configuration

Server

Name*: Design_Example_Server

Server count*: 1

Port Channel ID Min: 0 Max: 0

Logical Device*: AOS-1x1-1

Link

Name*: Design_Example_Server_Link

Switch*: Design_Example_Leaf

Attachment Type: Single-Homed [✓ Dual-Homed]

LAG Mode: [✓ LACP (Active)] LACP (Passive) Static LAG (no LACP) No LAG

Physical link count per individual switch (1 available)*: 1

Link speed*: 1 Gbps

Figure 2.8 Dual-Homed Server Example

The Apstra global catalog (in the Design menu) includes numerous predefined rack types based on common rack designs which may accommodate your own deployments. However, it is possible to create new rack types from scratch or clone, edit, and delete existing ones. For the purposes of this book, let's create three new rack types defined to fit the intended design. The three rack types will accommodate 1GE, 10GE, and external connectivity, whose specifications are listed in Tables 2.6, 2.7, and 2.8.

Table 2.6 1GE Rack Type

DO_Rack_1GE		
Connectivity Type		
Fabric Connectivity Design	L3 Clos	
Leaf(s)		
Name	DO_Rack_1GE_Leaf	
Logical Device	DO_48x1_8x40	
Uplinks to Spine	1 x 40GE	
Redundancy Protocols	None	
External facing	No	

Servers		
Server Group 1		
	Name	DO_Rack_1GE_Single_Server
	Server Count	4
	Logical Device	Apstra_1x1-1
	Link Name	DO_Rack_1GE_Single_Server_Link
	Upstream Leaf Name	DO_Rack_1GE_Leaf
	LAG Mode	No LAG
	Link Count Per Leaf	1
	Link Speed	1 Gbps
Server Group 2		
	Name	DO_Rack_1GE_Dual_Server
	Server Count	1
	Logical Device	Apstra_2x1-1
	Link Name	DO_Rack_1GE_Dual_Server_Link
	Upstream Leaf Name	DO_Rack_1GE_Leaf
	LAG Mode	LACP (Active)
	Link Count Per Leaf	2
	Link Speed	1 Gbps

Table 2.7 10GE Rack Type

DO_Rack_10GE		
Connectivity Type		
Fabric Connectivity Design	L3 Clos	
Leaf(s)		

Name	DO_Rack_10GE_Leaf	
Logical Device	Apstra-48x10+8x40-1	
Uplinks to Spine	1 x 40GE	
Redundancy Protocols	None	
External facing	No	
Servers		
Server Group 1		
	Name	DO_Rack_10GE_Single_Server
	Server Count	2
	Logical Device	Apstra_1x10-1
	Link Name	DO_Rack_10GE_Single_Server_Link
	Upstream Leaf Name	DO_Rack_10GE_Leaf
	LAG Mode	No LAG
	Link Count Per Leaf	1
	Link Speed	10 Gbps

Table 2.8 External Connectivity Rack Type

DO_Border_Rack		
Connectivity Type		
Fabric Connectivity Design	L3 Clos	
Leaf(s)		
Name	DO_Border_Rack_Leaf	
Logical Device	DO-_72x10_18x40	

Uplinks to Spine	1 x 40GE	
Redundancy Protocols	ESI	
External facing	Yes – 1 x 10GE Per Leaf	

Create Rack Types

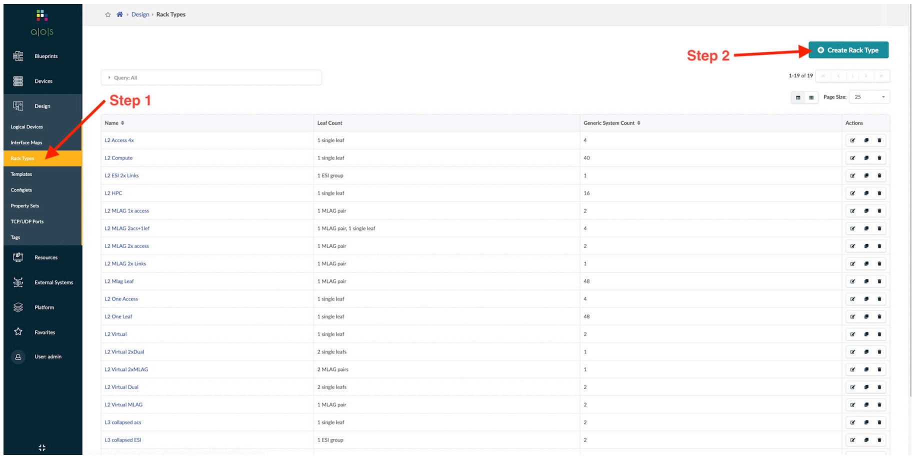
Creating rack types is not as straightforward as creating LDs in Chapter 1, mostly because they can become complex depending on DC requirements.

Whilst designing racks is more complex, most data centers use many multiples of the same rack design. If users need, for example, eight identical racks they will only have to specify the rack design once. Then in the template design (discussed later, when tying everything together) users simply tell Apstra that eight instances of that rack type are required.

Let's demonstrate how to create the DO_Rack_10GE with the specifications listed in Table 2.7.

Step 1: Go to Design > Rack Types.

Step 2: Click Create Rack Type.



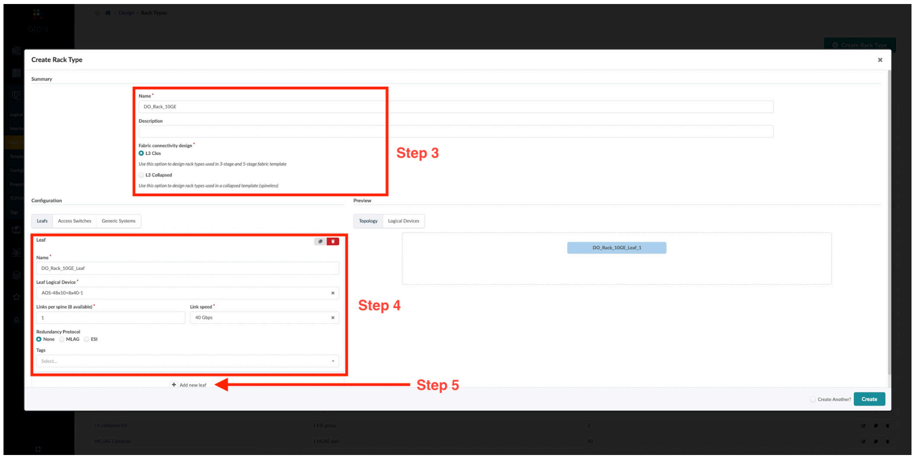
Name	Leaf Count	Generic System Count	Actions
L2 Access 4x	1 single leaf	4	
L2 Compute	1 single leaf	40	
L2 ESI 2x Links	1 ESI group	1	
L2 HPC	1 single leaf	16	
L2 MLAG 1x access	1 MLAG pair	2	
L2 MLAG 2acs+1lef	1 MLAG pair, 1 single leaf	4	
L2 MLAG 2x access	1 MLAG pair	2	
L2 MLAG 2x Links	1 MLAG pair	1	
L2 Mlag Leaf	1 MLAG pair	48	
L2 One Access	1 single leaf	4	
L2 One Leaf	1 single leaf	48	
L2 Virtual	1 single leaf	2	
L2 Virtual 2xDual	2 single leafs	1	
L2 Virtual 2xMLAG	2 MLAG pairs	1	
L2 Virtual Dual	2 single leafs	2	
L2 Virtual MLAG	1 MLAG pair	2	
L3 collapsed acs	1 single leaf	2	
L3 collapsed ESI	1 ESI group	2	

Figure 2.9 Create Rack Types

Step 3: Add a name, description (optional) and chose the fabric connectivity type

Step 4: Chose the leaf switches details such as the logical device model, the number of uplinks to the spine switches and the redundancy option.

Step 5: Add another leaf if required.



Create Rack Type

Summary

Name *
DO_Rack_10GE

Description

Fabric connectivity design *
[✓ L3 Clos]
Use this option to design rack types used in 3-stage and 5-stage fabric template
L3 Collapsed
Use this option to design rack types used in a collapsed template (spineless)

Configuration: Leafs | Access Switches | Generic Systems

Preview: Topology | Logical Devices

DO_Rack_10GE_Leaf_1

Leaf

Name *
DO_Rack_10GE_Leaf

Leaf Logical Device *
AOS-48x10+8x40-1

Links per spine (8 available) *
1

Link speed *
40 Gbps

Redundancy Protocol
[✓ None] MLAG ESI

Tags
Select...

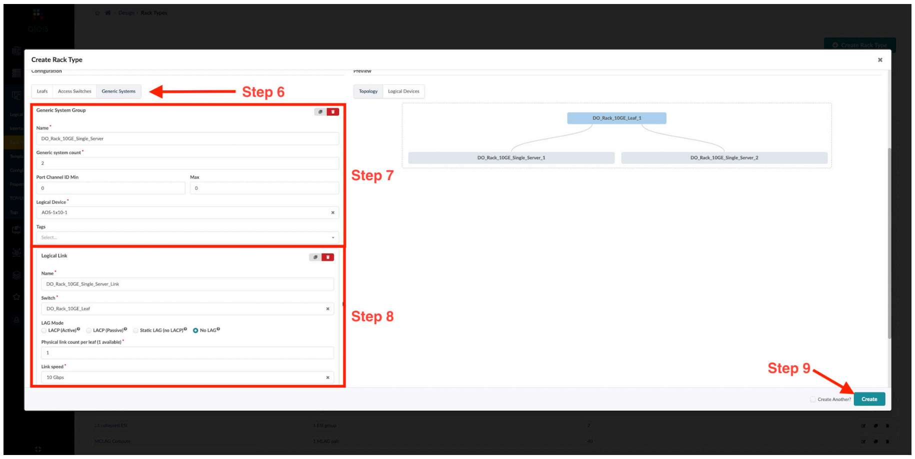
Figure 2.10 *Create Rack Types 2*

Step 6: Click on the Generic Systems tab.

Step 7: Add a group of logical servers and specify the group name, server quantity and Logical device model.

Step 8: Add a link, specify a link name, the target leaf switch, link aggregation method, link speed and link count. Multiple server groups can be added if required.

Step 9: Click Create.



Create Rack Type

Configuration: Leafs | Access Switches | Generic Systems

Preview: Topology | Logical Devices

DO_Rack_10GE_Leaf_1
DO_Rack_10GE_Single_Server_1
DO_Rack_10GE_Single_Server_2

Generic System Group

Name *
DO_Rack_10GE_Single_Server

Generic system count *
2

Port Channel ID Min
0

Max
0

Logical Device *
AOS-1x10-1

Tags
Select...

Logical Link

Name *
DO_Rack_10GE_Single_Server_Link

Switch *
DO_Rack_10GE_Leaf

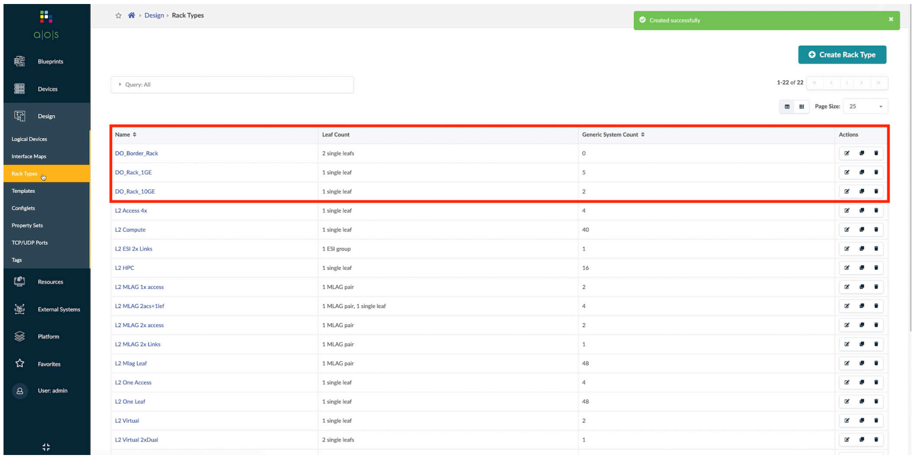
LAG Mode
LACP (Active) LACP (Passive) Static LAG (no LACP) [✓ No LAG]

Physical link count per leaf (1 available) *
1

Link speed *
10 Gbps

Figure 2.11 *Create Rack Types 3*

For our example DC fabric, once all three rack types are created, the finished result should look like Figure 2.12. Now that the individual racks have been modeled, a DC template can be created.



Name	Leaf Count	Generic System Count	Actions
DO_Border_Rack	2 single leafs	0	
DO_Rack_1GE	1 single leaf	5	
DO_Rack_10GE	1 single leaf	2	
L2 Access 4x	1 single leaf	4	
L2 Compute	1 single leaf	40	
L2 ESI 2x Links	1 ESI group	1	
L2 HPC	1 single leaf	16	
L2 MLAG 1x access	1 MLAG pair	2	
L2 MLAG 2acs+1lef	1 MLAG pair, 1 single leaf	4	
L2 MLAG 2x access	1 MLAG pair	2	
L2 MLAG 2x Links	1 MLAG pair	1	
L2 Mlag Leaf	1 MLAG pair	48	
L2 One Access	1 single leaf	4	
L2 One Leaf	1 single leaf	48	
L2 Virtual	1 single leaf	2	
L2 Virtual 2xDual	2 single leafs	1	

Figure 2.12 Completed Rack Design

Templates

Once each physical rack type has been defined the final step can be undertaken, building a template. A template can represent a row, a pod, or the entire DC network.

Templates themselves are relatively simple but tie together the racks designed in the previous section and fabric routing policy information. There are three types of templates that can be created in Apstra: *rack-based*, *pod-based,* and *collapsed fabric*.

Rack-based

Rack-based templates are 3-stage Clos fabrics and permit multiple racks to be connected via spine switches as shown in Figure 2.13.

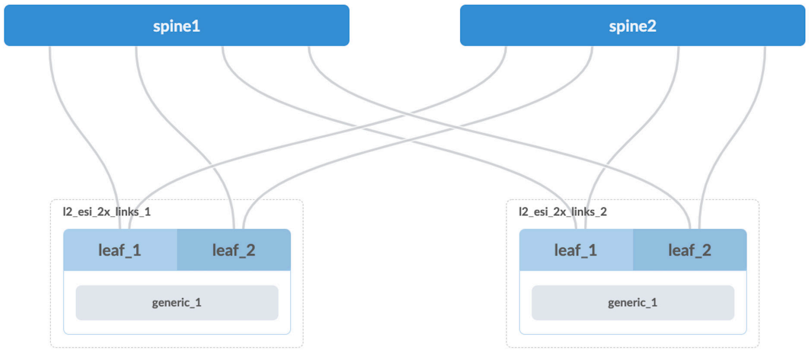


Figure 2.13 *Rack-Based Template*

Pod-based

Pod-based templates are used for 5-stage Clos fabrics. A 5-stage fabric uses super spines to connect smaller, 3-stage Clos fabrics, essentially connecting multiple rack-based templates together. Figure 2.14 shows a logical representation of a pod-based template.

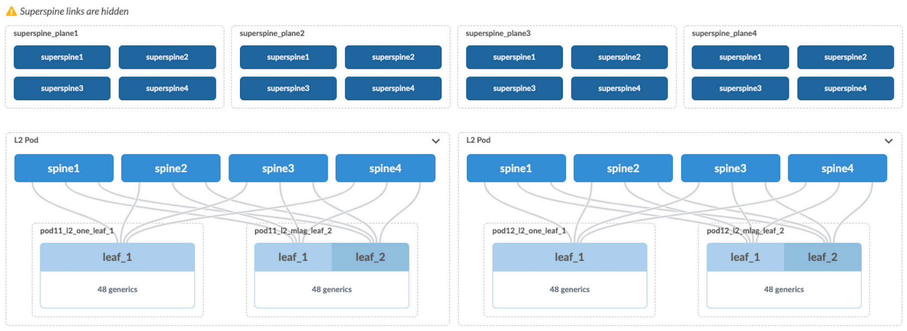


Figure 2.14 *Pod-Based Template*

NOTE Pod-based DCs will not be covered in this book. You can refer to the Apstra documentation for more details: https://www.juniper.net/documentation/product/us/en/apstra.

Collapsed

Collapsed fabric templates provide the ability to create an ESI-LAG design typically seen in small edge data centers or in the core of campus networks. See Figure 2.15.

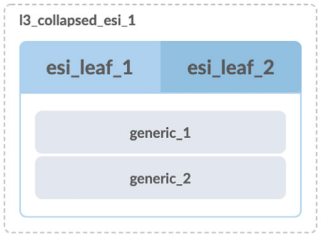


Figure 2.15 ESI-LAG Template

NOTE Collapsed fabric based DCs are not covered in this book, please refer to the Apstra documentation for more details: https://www.juniper.net/documentation/product/us/en/apstra.

Routing Policies

Once we've decided on the template type, we'll turn our attention to the routing polices that are implemented and enforced throughout the fabric. The options discussed are those available with rack-based templates. Please note that you have different options available when creating a pod-based or collapsed fabric templates. Check out the latest Apstra documentation: https://www.juniper.net/documentation/product/us/en/apstra.

Okay, a single rack-based template needs to be created. This template will consist of the items listed in Table 2.9. Details about the table's listed options follow.

Table 2.9 This Book's Template Design

Option Name	Choice	Notes
Name	DO_DC_Template	
Type	Rack Based	3 Stage Fabric Design
Policies		
ASN Allocation	Unique	3 Stage Fabric Design
Routing Policies	Default	0.0.0.0/0 Import Only
Overlay Protocol	MP-EBGP EVPN	
IP Allocation	IPv4	

Rack Types		
1GE Access Leaf	1 x DO_Rack_1GE	40GE Uplinks to Spines
10GE Access Leaf	1 x DO_Rack_10GE	40GE Uplinks to Spines
Border Leaf Rack	1 x DO_Border_Rack	40GE Uplinks to Spines
Spines		
Spine Logical Device(s)	2 x Apstra-32x40-3	

ASN Allocation Scheme

BGP ASN numbers are assigned to switches when building the underlay network. There are two ASN allocation schemes available with a rack-based template: *unique* and *single*. The unique allocation option is to be used with a 3-stage design. A different ASN number will be assigned to each spine in the fabric once deployed.

If the rack template being created is destined to be a pod in a 5-stage design, then the allocation scheme should be set to *single*. This means one ASN is assigned to all spines within a pod and another ASN is assigned to all super spines. It's important to note that the 3-stage Clos (representing a pod) will need to be assigned a 'single ASN allocation' before building the 5-stage Clos template. The 5-stage Clos template is essentially a template of multiple 3-stage templates.

Routing Policy (Import)

Routing policy allows an administrator to alter the default behavior of a routing protocol. Import policies control whether routes that that have been received are installed into the routing table and are controlled in the Apstra web UI with a simple toggle. This allows Apstra to validate the user's intent as well as whether the intended routes are being received correctly. For example, 'Am I actually receiving 0.0.0.0/0 as expected'. If not, Apstra will raise a routing anomaly accordingly to inform the user.

Default

Only accepts only a 0.0.0.0/0 BGP route.

ALL

Accepts all routes. Note that when accepting routes from an external device, the receiving device must be able to handle the number of incoming routes, or it may cause a crash or other undesirable behavior for the Apstra fabric network devices that are attached to external routers. *Verify that your network devices can accept the appropriate number of routes.*

Overlay Control Protocol

This defines the inter-rack virtual network overlay protocol used in the fabric. The Static VXLAN, also known as unicast VXLAN, uses statistically configured VX-LAN virtual tunnel endpoints (VTEPs) to route and distribute Layer 2 virtual network traffic between racks. Whereas the MP-EBGP EVPN uses EVPN family eBGP sessions between device loopbacks to exchange EVPN routes for hosts (Type 2) and networks (Type 5). MP-BGP is the preferred choice.

NOTE At the time of this writing only single-vendor EVPN fabrics are supported by Apstra, and VXLAN/EVPN capabilities for inter-rack virtual networks are dependent on the make and model of the network devices being used. Please check the latest Apstra documentation: https://www.juniper.net/documentation/product/us/en/apstra.

Spine to Leaf Links Type

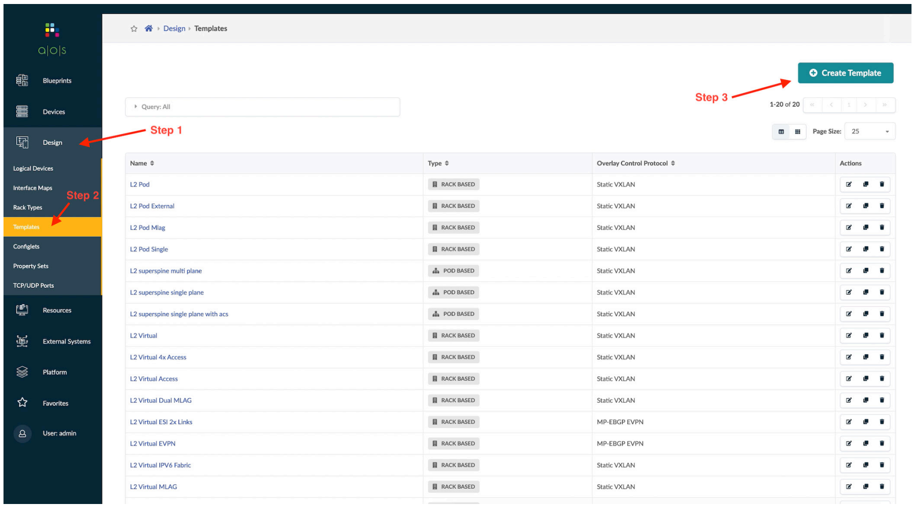
This option simply determines which IP protocol is used for connecting spines and leaf switches together. The options are either IPv4 or IPv6.

Creating Templates

Step 1: Navigate to the Design menu. See Figure 2.16.

Step 2: Navigate the Templates option under design menu.

Step 3: Click Create Template.



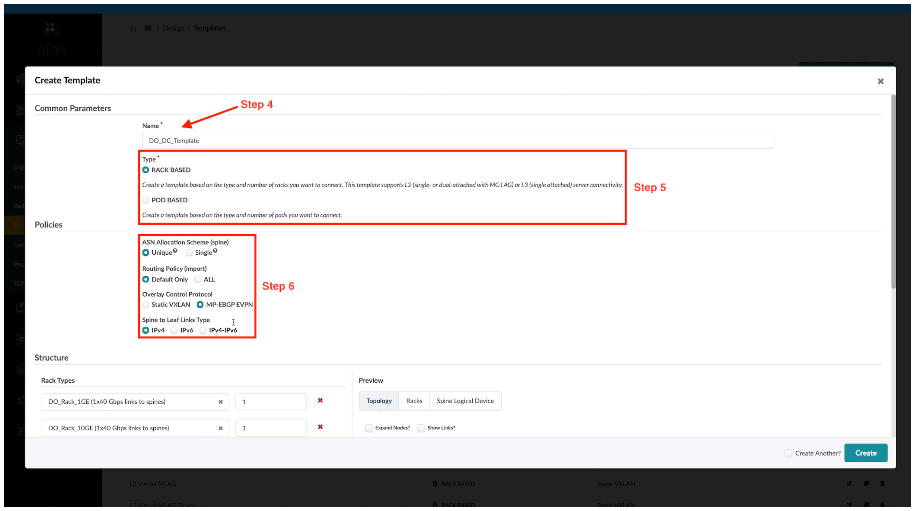
Name	Type	Overlay Control Protocol	Actions
L2 Pod	RACK BASED	Static VXLAN	
L2 Pod External	RACK BASED	Static VXLAN	
L2 Pod Mlag	RACK BASED	Static VXLAN	
L2 Pod Single	RACK BASED	Static VXLAN	
L2 superspine multi plane	POD BASED	Static VXLAN	
L2 superspine single plane	POD BASED	Static VXLAN	
L2 superspine single plane with acs	POD BASED	Static VXLAN	
L2 Virtual	RACK BASED	Static VXLAN	
L2 Virtual 4x Access	RACK BASED	Static VXLAN	
L2 Virtual Access	RACK BASED	Static VXLAN	
L2 Virtual Dual MLAG	RACK BASED	Static VXLAN	
L2 Virtual ESI 2x Links	RACK BASED	MP-EBGP EVPN	
L2 Virtual EVPN	RACK BASED	MP-EBGP EVPN	
L2 Virtual IPV6 Fabric	RACK BASED	Static VXLAN	
L2 Virtual MLAG	RACK BASED	Static VXLAN	

Figure 2.16 Creating DC Template

Step 4: Enter the name of the DC template. See Figure 2.17.

Step 5: Select the template type.

Step 6: Select the routing policy options.



Create Template

Common Parameters

Name *
DO_DC_Template

Type *
[✓ RACK BASED]
Create a template based on the type and number of racks you want to connect. This template supports L2 (single- or dual-attached with MC-LAG) or L3 (single attached) server connectivity.
POD BASED
Create a template based on the type and number of pods you want to connect.

Policies

ASN Allocation Scheme (spine)
[✓ Unique] Single

Routing Policy (import)
[✓ Default Only] ALL

Overlay Control Protocol
Static VXLAN [✓ MP-EBGP EVPN]

Spine to Leaf Links Type
[✓ IPv4] IPv6 IPv4-IPv6

Structure

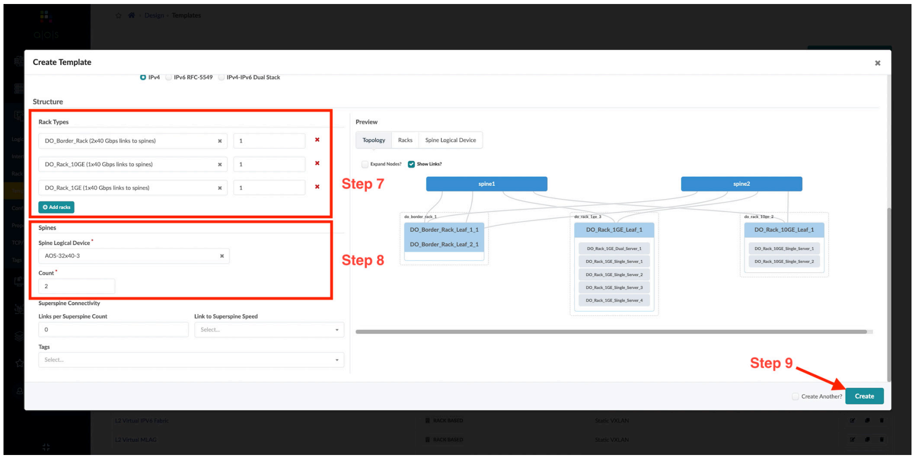
Rack Types	
DO_Rack_1GE (1x40 Gbps links to spines)	1
DO_Rack_10GE (1x40 Gbps links to spines)	1

Figure 2.17 Assigning Template Type and Policies

Step 7: Add the rack types required for the given design and then simply add the number of instances of the rack needed.

Step 8: Choose the spine switch logical device type and how many spine switches are desired.

Step 9: Click Create.



Create Template

[✓ IPv4] IPv6 RFC-5549 IPv4-IPv6 Dual Stack

Structure

Rack Types	
DO_Border_Rack (2x40 Gbps links to spines)	1
DO_Rack_10GE (1x40 Gbps links to spines)	1
DO_Rack_1GE (1x40 Gbps links to spines)	1

Spines

Spine Logical Device *
AOS-32x40-3

Count *
2

Superspine Connectivity

Links per Superspine Count: 0

Link to Superspine Speed: Select...

Tags: Select...

Figure 2.18 Selecting Template Racks and Spines

Once a complete DC template has been created it can be used to create a blueprint (these will be discussed in more depth in Chapter 4). A blueprint is an instance of this DC design where site specific information and configurations are applied and eventually deployed.

It is not mandatory to onboard physical devices to Apstra and have the underlying infrastructure ready before creating a blueprint. You can deploy a blueprint, select the vendor models you plan to use, start building out the blueprint's configuration independently of a deployed physical infrastructure, and then go back and bring the devices under Apstra management later on. This is recommended because Apstra provides a cabling map with a working blueprint that you can then use to cable up your physical infrastructure.

The next section discusses resource pools, although optional as Apstra ships with pre-defined defaults, they are strongly recommended for your own deployments.

Resource Pools

Let's step away from the DC design for a moment and examine an optional but important stage of pre-deployment: creating the necessary resource pools. Resource pools are logical elements that are used when deploying a physical DC. The workings are covered in depth in Chapter. Apstra has four types of resource pools: IPv4 addresses, IPv6 addresses, AS Numbers, and VXLAN VNIs. These pools are some of the fundamental pieces of information that Apstra needs to deploy a data center. The pools are assigned to devices during the deployment phase.

In Apstra resource pools are managed as a unit instead of micromanaging each individual IP, ASN, and so on. Apstra already ships with default resource pools that can be used out of the box, however it is recommended that users create their own based on their environment.

Every DC deployment will need at least two resource pools. For example, when creating a simple L3 only Clos design, IP address for the links between the leafs and spines, as well as ASNs for every switch.

It is important to remember that Apstra looks at the network as a single distributed system, not as multiple network devices. Apstra is looking to change the attitude of traditional DC deployments where IP addresses and ASN numbers would usually be allocated in advance. Apstra assigns these in an intelligent way, meaning no pre-allocation is needed. This means users do not need to worry about tracking these objects or verifying that they are unique within Apstra. Apstra does this automatically.

Three resource pool types are needed for this book's deployment. Table 2.10 defines those that must be created.

Table 2.10 Resource Pools to be Created for This Book

Name	Subnet	
DO_IP_Loopbacks	192.168.0.0/16	
DO_IP_Fabric_Links	172.16.0.0./16	
DO_IP_External_Links	172.17.0.0/16	
Name	First ASN	Last ASN
DO_ASN_Fabric	64499	64510
Name	First VNI	Last VNI
DO_VNI_Fabric	30000	50000

IP Address Pools

IP pools configured in Apstra are required when deploying a DC fabric (blueprint) and they are the fundamental to building any IP Clos fabric. IP addresses are assigned for the following reasons:

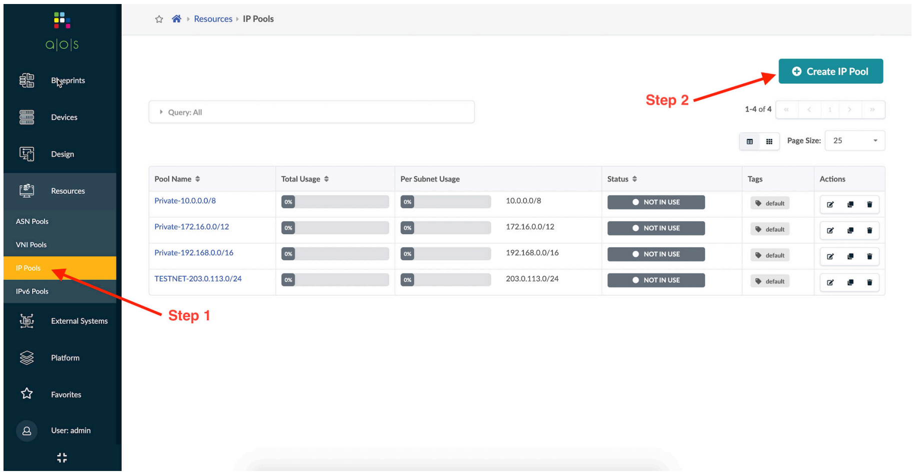
- Loopback Addresses: This interface, unlike traditional interfaces, does not rely on any physical connection being present. This makes these addresses more stable as they are not affected by things such as link flaps. Apstra, by default, assigns loopback addresses to both spine and leaf switches. These loopbacks will be used as the BGP router ID and when peering with external routers.
- Link Addresses: These are the IPs assigned to each end of a point-to-point connection. These addresses are used for creating a L3 Clos topology, BGP peering between spine and leaf switches, and as the next hop for traffic facing external routers.

By default, Apstra ships with several RFC1918 address pools that can be used. However, most users prefer to define their own pools as in real world deployments these addresses are likely to be used elsewhere in the network. This could accidently cause an overlap with fabric addresses.

Creating new IP pools is easy and the following steps create one of the needed IP Pools for the deployment this book is building. Once logged into Apstra complete the following steps to create an IP resource pool:

Step 1: Go to Resources > IP Pools. See Figure 2.19.

Step 2: Click Create IP Pool.



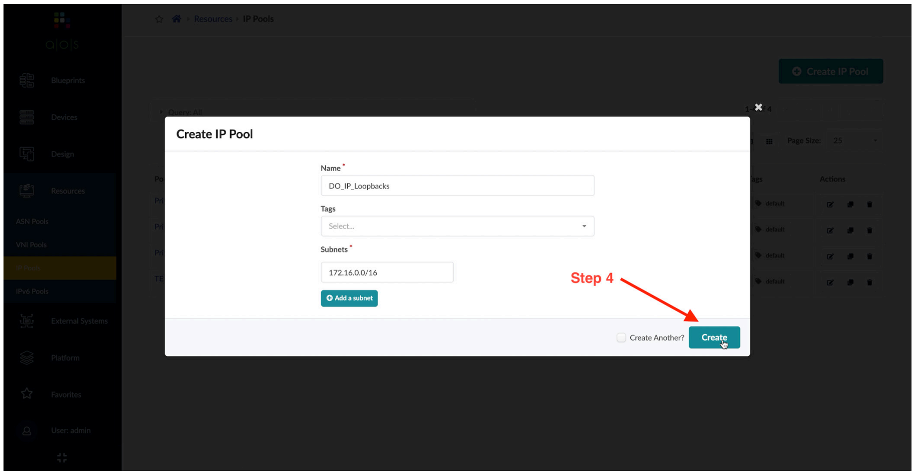
Pool Name	Total Usage	Per Subnet Usage		Status	Tags	Actions
Private-10.0.0.0/8	0%	0%	10.0.0.0/8	NOT IN USE	default	
Private-172.16.0.0/12	0%	0%	172.16.0.0/12	NOT IN USE	default	
Private-192.168.0.0/16	0%	0%	192.168.0.0/16	NOT IN USE	default	
TESTNET-203.0.113.0/24	0%	0%	203.0.113.0/24	NOT IN USE	default	

Figure 2.19 *Creating IP Resource Pool*

Step 3: Add the following information shown in Figure 2.20:

- Name (Required) – The Pool identification name.
- Tag (Optional) - To enable filtering by user-specified categories.
- Subnet (Required) – The Network address followed by the subnet mask (Note – Multiple networks can be added to a single IP Pool).

Step 4: Click Create.

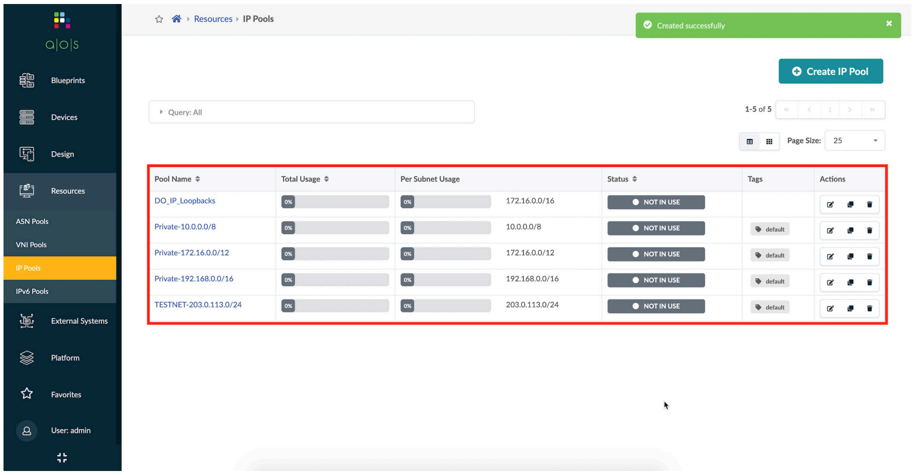


Name: DO_IP_Loopbacks
Tags: Select...
Subnets: 172.16.0.0/16
Create Another?

Figure 2.20 *Assigning IP Pool Parameters*

Once these pools have been created, they can be edited and deleted by going to Resources > IP Pools and selecting either Edit or Delete.

If you're following along in your own lab, the remaining IP pools need to be created at this time. Refer to the Table 2.10. Your final state should look something like Figure 2.21.



Pool Name	Total Usage	Per Subnet Usage		Status	Tags	Actions
DO_IP_Loopbacks	0%	0%	172.16.0.0/16	NOT IN USE		
Private-10.0.0.0/8	0%	0%	10.0.0.0/8	NOT IN USE	default	
Private-172.16.0.0/12	0%	0%	172.16.0.0/12	NOT IN USE	default	
Private-192.168.0.0/16	0%	0%	192.168.0.0/16	NOT IN USE	default	
TESTNET-203.0.113.0/24	0%	0%	203.0.113.0/24	NOT IN USE	default	

Figure 2.21 Resource Pools

ASN Pools

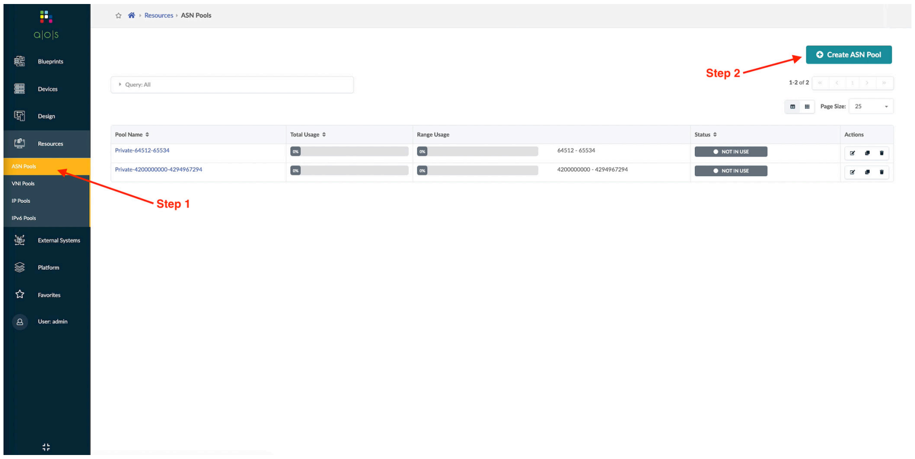
BGP is an inter-autonomous system routing protocol. An Autonomous System Number (ASN) is a globally unique identifier that allows its autonomous system to exchange routing information with other systems.

Within Apstra, super-spines, spines, leafs, and Layer 3 servers use BGP to exchange routing information. EBGP requires each device to be in an AS, which is designated by the ASN. This can be either from the same pool or from different pools. They can also be assigned individually if a specific resource must be assigned to a specific device.

Creating ASN pools is a simple process:

Step 1: Go to Resources > ASN Pools. See Figure 2.22.

Step 2: Click Create ASN Pool.



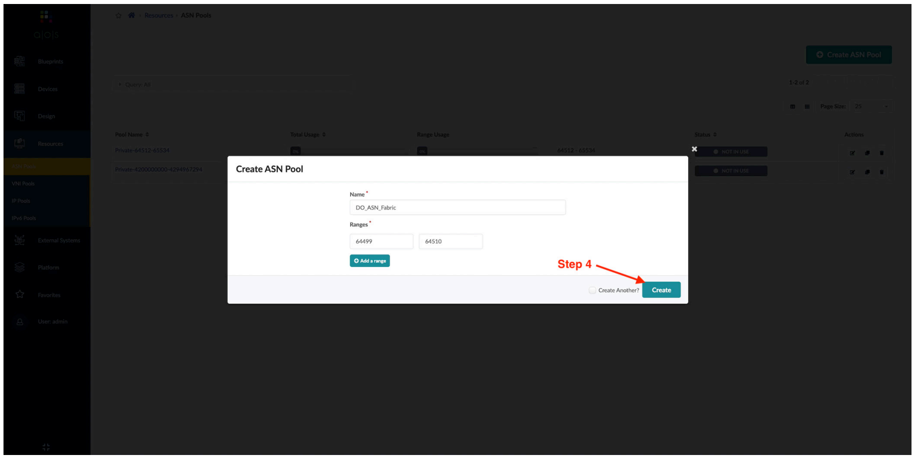
Pool Name	Total Usage	Range Usage		Status	Actions
Private-64512-65534	0%	0%	64512 - 65534	NOT IN USE	
Private-4200000000-4294967294	0%	0%	4200000000 - 4294967294	NOT IN USE	

Figure 2.22 Creating ASN Resource Pool

Step 3: Add the following information for the resource pool.

- Name (Required) – Name of the resource pool
- First ASN (Required) – First ASN number of the pool
- Last ASN (Required) – Last ASN number of the pool

Step 4: Click Create. See Figure 2.23.



Name: DO_ASN_Fabric
Ranges: 64499 64510

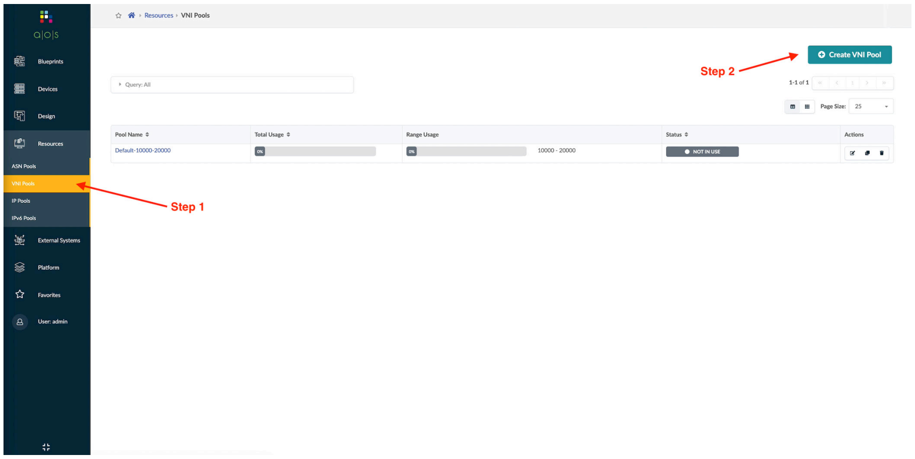
Figure 2.23 Assigning ANS Pool Parameters

VNI Pool

VXLAN is an encapsulation protocol that provides data center connectivity using tunneling or stretching Layer 2 connections over an underlying Layer 3 network. VXLAN uses VNIs to identify a LAN segment, similar to a VLAN ID. Unlike VLANs, VXLAN allows data centers to scale to 16 million virtual networks. Apstra assigns these VNIs from the pool when new VXLAN virtual networks are created.

Step 1: Go to Resources > VNI Pools. See Figure 2.24.

Step 2: Click Create VNI Pool.



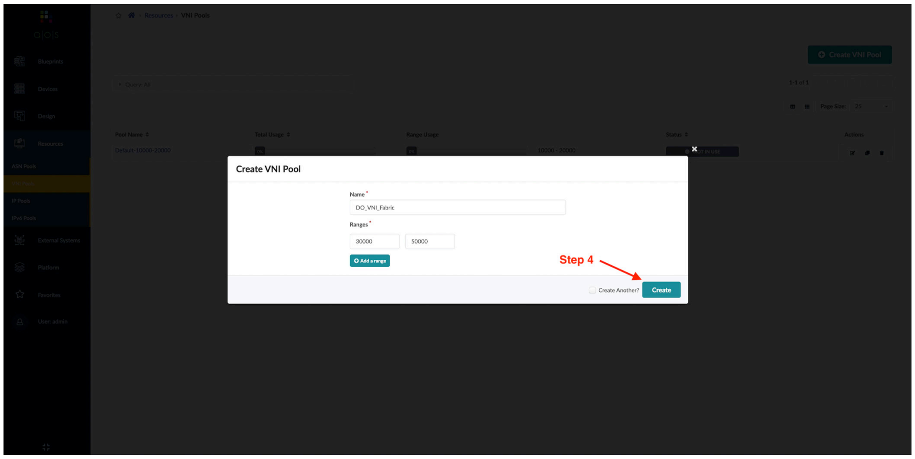
Pool Name	Total Usage	Range Usage		Status	Actions
Default-10000-20000	0%	0%	10000 - 20000	NOT IN USE	

Figure 2.24 *Creating VNI Resource Pool*

Step 3: Add the following information for the resource pool. See Figure 2.25.

- Name (Required) – Name of the resource pool
- First VNI (Required) – First VNI number of the pool
- Last VNI (Required) – Last VNI number of the pool

Step 4: Click Create.


Create VNI Pool

Name: DO_VNI_Fabric

Ranges: 30000 50000

Figure 2.25 Assigning VNI Pool Parameters

Summary

This chapter showed you how to design a simple DC using Apstra, starting with building logical devices, moving on to modeling switch variants, then using these to create rack types, and finally tying them all together using templates. The following resources must be created and will be used during the actual fabric deployment:

- Logical Devices
- Rack Types
- DC Templates
- Resource Pools

That's it! A deployment of a DC fabric using the template created in this chapter can now happen. Users can now go ahead and instantiate this template and begin the deployment of a data center. Apstra does not require users to onboard devices as a prerequisite. Being able to do this independent of hardware allows users to pre-stage the deployment and download comprehensive cable plans even before a hardware model or vendor is chosen. Once ready, users can bring actual devices under Apstra's management to fully complete the deployment.

The next chapter discusses different ways users can onboard devices to Apstra and transition them to a state where they can be allocated to a data center deployment.

Chapter 3

Device On-boarding and Management

This chapter will give you an overview of how Apstra discovers, onboards, and manages devices using software agents that are either deployed on a device or run on the Apstra server. It begins with a general overview of device discovery and then focuses on procedures to onboard and manage a Junos device from Apstra.

Devices connected to Apstra are managed with *device agents*. Apstra provides an integrated agent installer that automates installation and verification of Apstra agents for network devices. The agents handle configuration management, device-to-server communication, and telemetry collection:

- ON-BOX agents are installed on the device itself within the guest shell and require a user with full administration and root privileges in order for Apstra to configure the box.
- OFF-BOX agents are *not* installed on the network devices; they run on the Apstra server and communicate with the devices through their APIs to deploy configurations and collect telemetry data. This is currently the only supported model for Junos.

Juniper QFX-series switches can be onboarded into Apstra in one of two ways:

- For ‘New out of the Box’ (NooB) devices shipped from the factory you can use Zero-Touch Provisioning (ZTP) to discover the device, assign it an IP address, username, and password, as well as upgrade its software if required and install an Apstra system agent. Existing Junos devices can be returned to the NooB state using the CLI command ‘request system zeroize’.

- Devices that already have an IP address and management user configured can be onboarded by Apstra by simply deploying the system agent on the Apstra server and granting it access to the device.

Once onboarded, these devices will transition through many functional states during their lifecycle and their configuration will change to reflect functionality. This chapter discusses each of these methods, but first let's review the prerequisites for onboarding Juniper devices.

Apstra Device Modeling

Before onboarding devices and making them useable within Apstra, Device Profiles and an Interface Mapping need to be present.

Device Profiles (DP)

Logical devices (LDs) were discussed in Chapter 2; these are the abstractions that enabled you to model a DC deployment without specifying the exact hardware model. During the actual deployment phase (discussed in Chapter 4) a hardware model must be mapped to the DC design's logical device model. Part of the way this is achieved is using device profiles. However, DPs are not something you should edit, as they reflect the single source of truth for the physical switch.

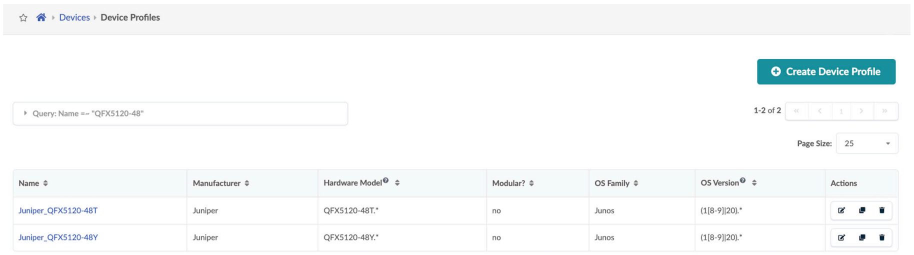
DPs allow a new device to be recognized and utilized by Apstra. They essentially capture much of the device-specific semantics (for example, hardware and software capabilities of the device) and need to be discovered by Apstra, as well as applied to the correct configuration for that device.

DPs store vendor-specific hardware information such as the model, OS family, port layout, port speeds, and even hardware information such as CPU, RAM, and SSD capacity. From Apstra's perspective, the form factor of the hardware is not essential; the network operating system (NOS) is what matters. As long as Apstra can communicate with the NOS, Apstra will only see a difference in the number of interfaces. The features will still be managed and configured the same way.

NOTE Between switch software versions, the NOS API needs to be validated if changes have occurred. Consult the latest Apstra documentation to view compatible NOS versions: https://www.juniper.net/documentation/product/us/en/apstra.

By default, Apstra ships with many different device profiles that define various hardware from a wide variety of vendors. These device profiles have been tested by Apstra and are fully supported. It's also possible, of course, to create a device profile from scratch in the web UI or through API requests, however, this approach is not recommended for first-time users. Also, if a DP is missing from Apstra the user should contact Juniper JTAC and seek further advice.

Several different types of Juniper switches that are components of the fabric are used in this book, and Apstra already comes with all of these device profiles included. You can see them by going to Devices > Device Profiles and filter based on model number. See Figure 3.1.



Devices › Device Profiles

Query: Name =~ "QFX5120-48"

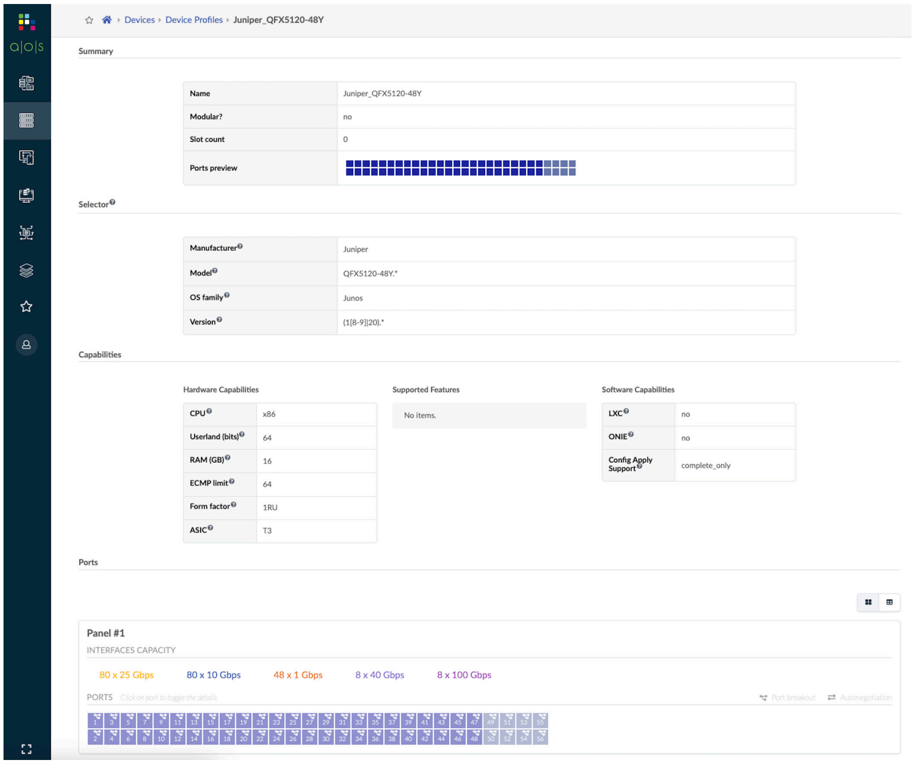
1-2 of 2

Page Size: 25

Name	Manufacturer	Hardware Model	Modular?	OS Family	OS Version	Actions
Juniper_QFX5120-48T	Juniper	QFX5120-48T.*	no	Junos	(1[8-9]|20).*	
Juniper_QFX5120-48Y	Juniper	QFX5120-48Y.*	no	Junos	(1[8-9]|20).*	

Figure 3.1 *Device Profiles Search*

Next is an example of a Juniper QFX5120-48Y switch in Figure 3.2.



Devices › Device Profiles › Juniper_QFX5120-48Y

Summary

Name	Juniper_QFX5120-48Y
Modular?	no
Slot count	0
Ports preview	

Selector

Manufacturer	Juniper
Model	QFX5120-48Y.*
OS family	Junos
Version	(1[8-9]|20).*

Capabilities

Hardware Capabilities

CPU	x86
Userland (bits)	64
RAM (GB)	16
ECMP limit	64
Form factor	1RU
ASIC	T3

Supported Features

No items.

Software Capabilities

LXC	no
ONIE	no
Config Apply Support	complete_only

Ports

Panel #1

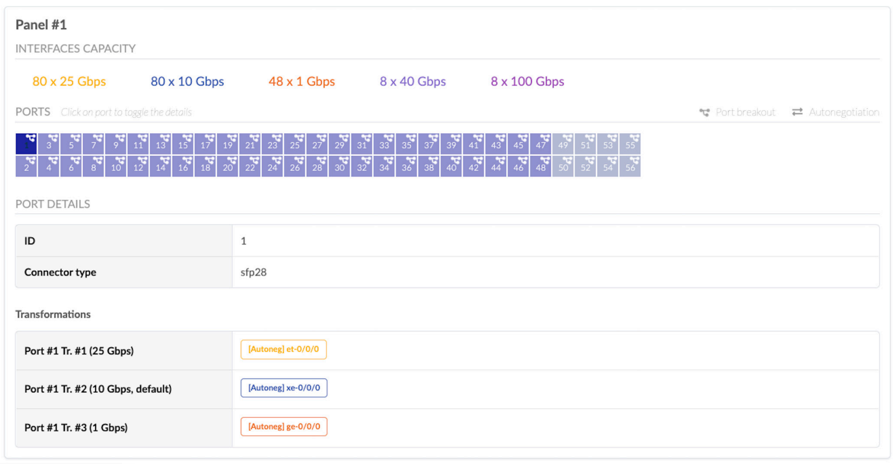
INTERFACES CAPACITY

80 x 25 Gbps 80 x 10 Gbps 48 x 1 Gbps 8 x 40 Gbps 8 x 100 Gbps

PORTS *Click on port to toggle the details*

Figure 3.2 *The Juniper QFX5120-48Y Device Profile that Ships with Apstra by Default*

It's important to notice the interface transformations listed for this switch. It's typical in many networking devices that a switch interface is capable of operating at different speeds. Apstra refers to each speed capability as a *transformation*. In Figure 3.3 it can be seen that Port 1 has three different speeds: 25, 10, and 1GE. The default is 10GE.



Panel #1

INTERFACES CAPACITY

80 x 25 Gbps   80 x 10 Gbps   48 x 1 Gbps   8 x 40 Gbps   8 x 100 Gbps

PORTS Click on port to toggle the details   Port breakout   Autonegotiation

1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39	41	43	45	47	49	51	53	55
2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40	42	44	46	48	50	52	54	56

PORT DETAILS

ID	1
Connector type	sfp28

Transformations

Port #1 Tr. #1 (25 Gbps)	[Autoneg] et-0/0/0
Port #1 Tr. #2 (10 Gbps, default)	[Autoneg] xe-0/0/0
Port #1 Tr. #3 (1 Gbps)	[Autoneg] ge-0/0/0

Figure 3.3 Example of a Switch Port Having Multiple Transformations

In this case Port 1 has 1/10/25 Gbps interfaces with 10 Gbps being the default. This means this switch can be mapped to many different logical device models that were discussed in Chapter Two to take advantage of the switch's capabilities.

Regardless of how the LD was originally designed, the port speed can be modified to any speed supported in the DP, as a *day 2* operation, including channelizing a QSFP28 port to 4x10/25GE. Any *day 2* operation as described above is validated against the DP, therefore, *will this switch* (as described in the DP) *support the required speeds*?

Interface Mapping (IM)

An interface mapping ties the logical device model with a device profile (see Figure 3.4). With this mapping, Apstra creates a link between the logical device profile used when creating the DC design, specifically in rack types and templates, with the physical device profile.

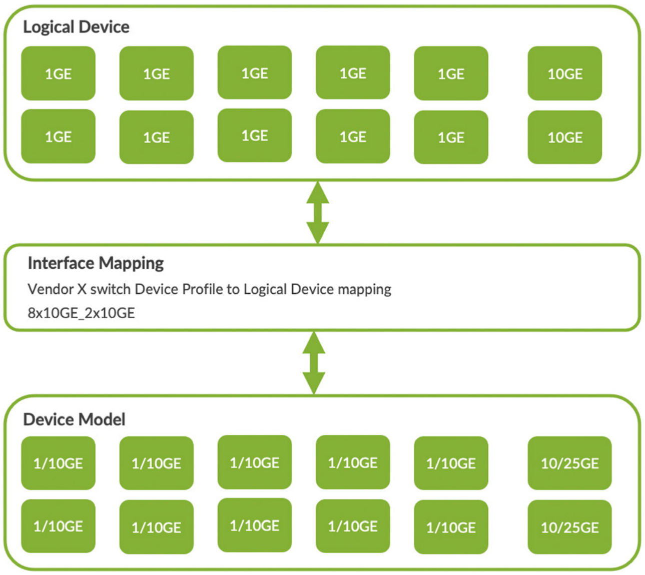


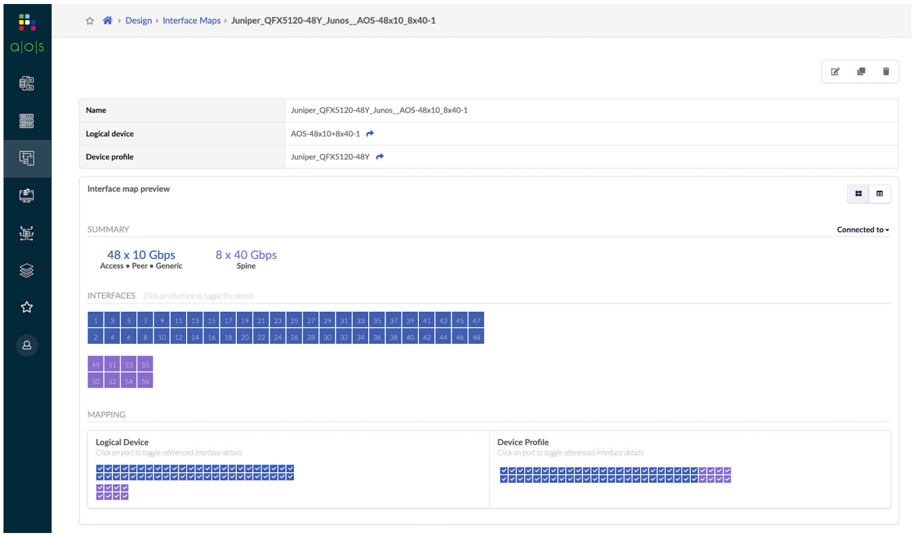
Figure 3.4 Example of an Interface Mapping Used to Tie a Logical Profile to a Device Model

When designing a data center fabric, Apstra uses the logical device profiles to create the network based on switch port count without defining the exact hardware model to be used. The logical device profile states how many ports, at a set speed, will be used during the deployment. On the other hand, a device profile contains all the relevant information Apstra needs to know about a particular model of a Juniper switch, including port count and possible transformation types.

For example, within a DC design, the operator might state they wish to have a switch that has 48 x 1GE ports and 8 x 40GE ports. They would choose a logical device model that represents this, but they would need to tie this to a physical device type described in device profiles when they come to deploy. A suitable switch could be a Juniper Networks QFX5120-48Y as it would fit this particular requirement; however, it can support many different port speed variations. This interface mapping is just one variation.

The interface mapping can be considered the ‘glue’ that ties together logical device models with device profiles.

Apstra ships with over a hundred interface mappings that project common device profiles to logical switch models; an example of a Juniper QFX5120-48Y can be seen in Figure 3.5. Interface mappings can be found by going to Design > Interface Maps. In Figure 3.5 you can see the logical device and the device profile.


Design > Interface Maps > Juniper_QFX5120-48Y_Junos__AOS-48x10_8x40-1

Name	Juniper_QFX5120-48Y_Junos__AOS-48x10_8x40-1
Logical device	AOS-48x10+8x40-1
Device profile	Juniper_QFX5120-48Y

Interface map preview

SUMMARY — Connected to

48 x 10 Gbps — Access • Peer • Generic

8 x 40 Gbps — Spine

INTERFACES — Click on interface to toggle the details

MAPPING

Logical Device — Click on port to toggle referenced interface details

Device Profile — Click on port to toggle referenced interface details

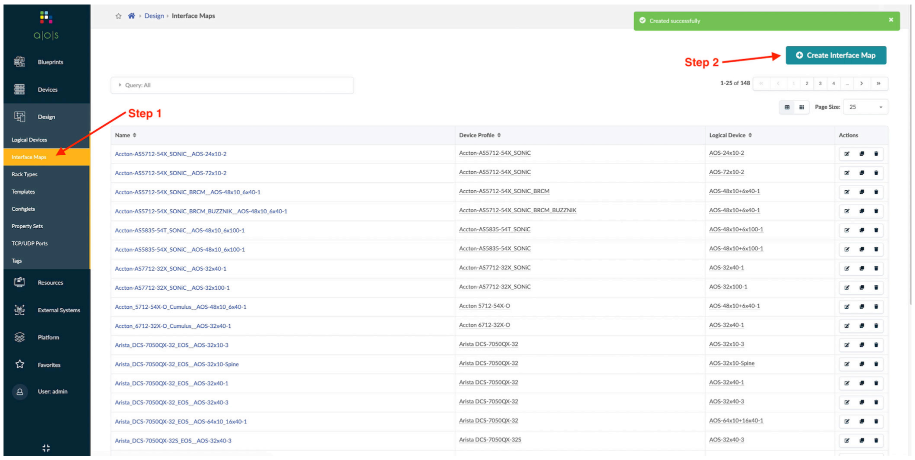
Figure 3.5 *Interface Mapping That Ties Together a Juniper Device Model with an Apstra Pre-Defined Logical Device*

Beyond the predefined interface maps you can create new ones via the web UI or the API. Apstra ships with common logical device types that were discussed in the previous chapter but it's possible that during the design stage a different logical device model definition is required. If a logical device type is required which is not included by default, a device model will need to be mapped to the new logical device model. A new interface mapping needs to be created for this purpose and below are the steps to achieve this.

Create an Interface Mapping Between a Juniper QFX5120-48Y Device Profile and the Logical Device Model

Step 1: Go to Design > Interface Maps. See Figure 3.6.

Step 2: Click Create Interface Map.



Name	Device Profile	Logical Device	Actions
Accton-AS5712-54X_SONiC__AOS-24x10-2	Accton-AS5712-54X_SONiC	AOS-24x10-2	
Accton-AS5712-54X_SONiC__AOS-72x10-2	Accton-AS5712-54X_SONiC	AOS-72x10-2	
Accton-AS5712-54X_SONiC_BRCM__AOS-48x10_6x40-1	Accton-AS5712-54X_SONiC_BRCM	AOS-48x10+6x40-1	
Accton-AS5712-54X_SONiC_BRCM_BUZZNIK__AOS-48x10_6x40-1	Accton-AS5712-54X_SONiC_BRCM_BUZZNIK	AOS-48x10+6x40-1	
Accton-AS5835-54T_SONiC__AOS-48x10_6x100-1	Accton-AS5835-54T_SONiC	AOS-48x10+6x100-1	
Accton-AS5835-54X_SONiC__AOS-48x10_6x100-1	Accton-AS5835-54X_SONiC	AOS-48x10+6x100-1	
Accton-AS7712-32X_SONiC__AOS-32x40-1	Accton-AS7712-32X_SONiC	AOS-32x40-1	
Accton-AS7712-32X_SONiC__AOS-32x100-1	Accton-AS7712-32X_SONiC	AOS-32x100-1	
Accton_5712-54X-O_Cumulus__AOS-48x10_6x40-1	Accton 5712-54X-O	AOS-48x10+6x40-1	
Accton_6712-32X-O_Cumulus__AOS-32x40-1	Accton 6712-32X-O	AOS-32x40-1	
Arista_DCS-7050QX-32_EOS__AOS-32x10-3	Arista DCS-7050QX-32	AOS-32x10-3	
Arista_DCS-7050QX-32_EOS__AOS-32x10-Spine	Arista DCS-7050QX-32	AOS-32x10-Spine	
Arista_DCS-7050QX-32_EOS__AOS-32x40-1	Arista DCS-7050QX-32	AOS-32x40-1	
Arista_DCS-7050QX-32_EOS__AOS-32x40-3	Arista DCS-7050QX-32	AOS-32x40-3	
Arista_DCS-7050QX-32_EOS__AOS-64x10_16x40-1	Arista DCS-7050QX-32	AOS-64x10+16x40-1	
Arista_DCS-7050QX-32S_EOS__AOS-32x40-3	Arista DCS-7050QX-32S	AOS-32x40-3	

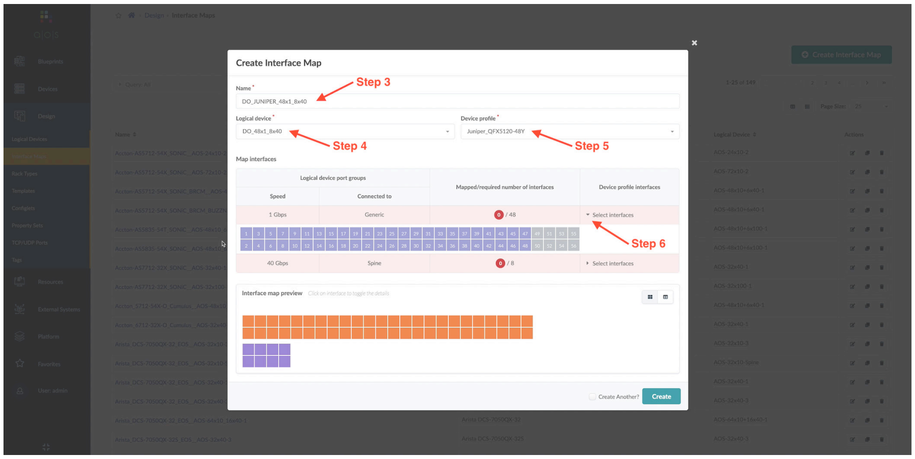
Figure 3.6 *Creating Interface Mappings in Apstra*

Step 3: Name the Interface Map. See Figure 3.7.

Step 4: Select the relevant logical device.

Step 5: Select the relevant device profile.

Step 6: Click on the arrow to expand the required panel. In this example the 48 port 1GE ports that connect to generic systems.



Name: DO_JUNIPER_48x1_8x40

Logical device: DO_48x1_8x40

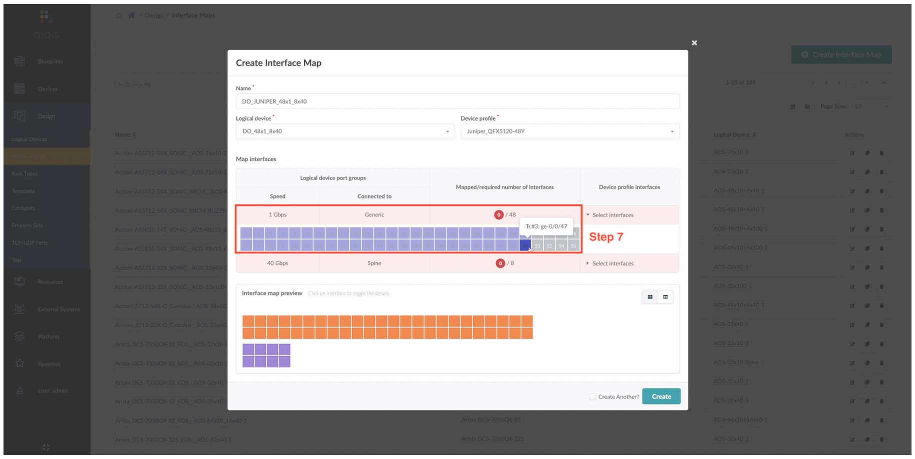
Device profile: Juniper_QFX5120-48Y

Map interfaces

Logical device port groups		Mapped/required number of interfaces	Device profile interfaces
Speed	Connected to		
1 Gbps	Generic	0 / 48	Select interfaces
40 Gbps	Spine	0 / 8	Select interfaces

Figure 3.7 *Creating Interface Mappings in Apstra*

Step 7: Click on a port and while holding the mouse button down slide/drag the cursor until the number of ports required are highlighted and then release the mouse button, this will be a port group. In this example the first 48 ports have been selected. See Figure 3.8.


Create Interface Map

Name *: DO_JUNIPER_48x1_8x40

Logical device *: DO_48x1_8x40

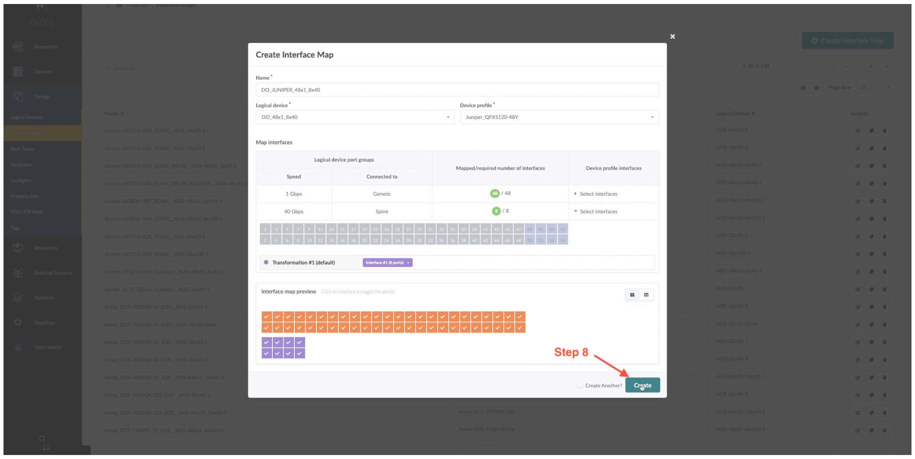
Device profile *: Juniper_QFX5120-48Y

Map interfaces

Logical device port groups		Mapped/required number of interfaces	Device profile interfaces
Speed	Connected to		
1 Gbps	Generic	0 / 48	Select interfaces
40 Gbps	Spine	0 / 8	Select interfaces

Figure 3.8 *Creating Interface Mappings in Apstra*

Step 8: Once all interface panels have had the correct ports assigned the user can click Create as in Figure 3.9. The column *Mapped/required number of interfaces* will show in green once the expected number of interfaces have been assigned to every interface panel.


Create Interface Map

Name *: DO_JUNIPER_48x1_8x40

Logical device *: DO_48x1_8x40

Device profile *: Juniper_QFX5120-48Y

Map interfaces

Logical device port groups		Mapped/required number of interfaces	Device profile interfaces
Speed	Connected to		
1 Gbps	Generic	48 / 48	Select interfaces
40 Gbps	Spine	8 / 8	Select interfaces

Transformation #1 (default) — Interface #1 (8 ports)

Figure 3.9 *Creating Interface Mappings in Apstra*

Now the logical devices created in the design stage are mapped to physical hardware. Remember that Apstra ships with many interface mappings by default so it's possible additional IMs are not required.

How Apstra Manages Devices

Devices controlled by Apstra use an *agent* to facilitate the communication between the NOS and Apstra. These agents come in two flavors: off-box and on-box. The agents handle configuration management, device-to-server communication, and telemetry collection.

When configuring these agents, it is possible to have them in *telemetry only* mode if Apstra is not utilized to manage the DC network. With telemetry only mode the agents would just be streaming telemetry information from the device to Apstra, however in this book, the devices onboarded will be fully controlled by Apstra.

Off-box System Agent

The off-box agent runs on the Apstra server and interacts with the NOS API to perform configuration and receive telemetry. Juniper network devices currently interact with the Apstra server using the off-box agent. An off-box agent is a Docker container that exists per Junos device that Apstra manages and runs locally on the Apstra server.

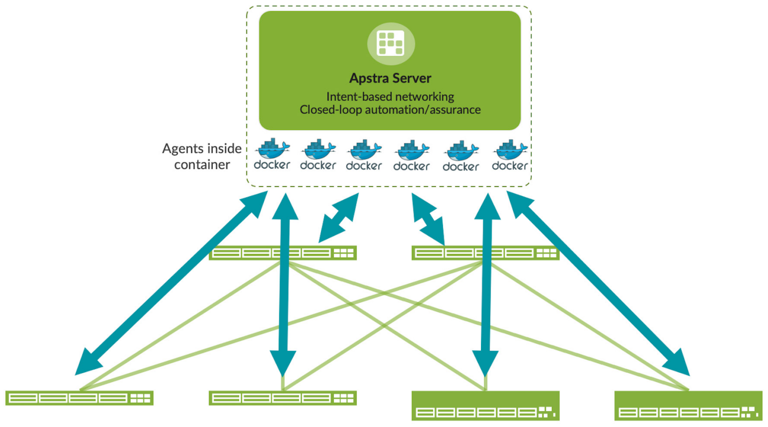


Figure 3.10 Representation of Docker Container per Junos Device Facilitating the Off-Box Agent Function

On-box System Agent

On-box agents are similar to off-box agent except they run on the network device instead of running on the Apstra server. Some network operating systems, such as Cumulus Linux and Enterprise SONiC (Software for Open Networking in the Cloud), can use the on-box agent.

Onboarding

Onboarding Juniper devices into Apstra can be achieved in two ways:

- The first option is by Zero-Touch Provisioning (ZTP) the device.
- The second option is to provide minimal configuration either by hand or by using a stub file. This gives the device basic connectivity and allows Apstra to complete the onboarding process.

These two ways to onboard devices in Apstra are discussed at length at the end of this chapter.

ZTP Onboarding

One option to onboard multiple devices into Apstra is to use a Zero-Touch Provisioning (ZTP) server. ZTP allows the provisioning of new Juniper devices in the network automatically, with minimal manual intervention. When using ZTP with the Apstra server, upon physically connecting the device to the network and booting it with the New out of the Box (Noob) configuration, the device upgrades (or downgrades) the Junos OS release and auto-onboards to the Apstra server.

It's important to understand what happens during the ZTP process:

- An IP address is assigned via DHCP service
- Sets the device root password
- Creates a device user for Apstra device system agent
- Upgrades / downgrades device OS, if needed
- An Apstra On-box or Off-box System Agent is created

If ZTP is the preferred option for onboarding devices to the fabric, then we recommend you use the Apstra ZTP server. AApstra has created a dedicated ZTP server that runs several services, all of which are needed to easily and successfully onboard a device (See Figure 3.11). The ZTP server is separate from the Apstra server, allowing placement in separate locations. Note that the ZTP server is only used for device onboarding.

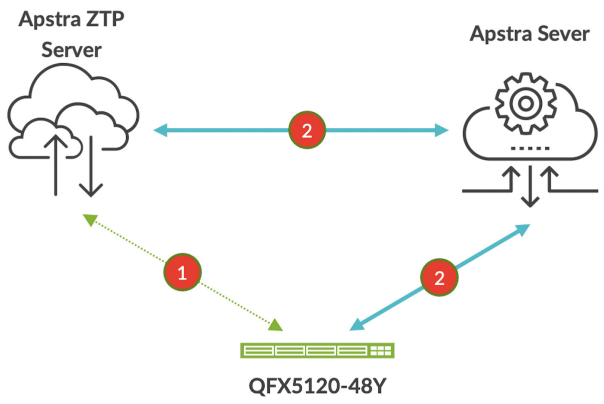


Figure 3.11 v Before Onboarding to Apstra

The ZTP server hosts several docker containers that perform functions such as DHCP and TFTP services. The following output shows the container services used and running on the Apstra ZTP Server:

```
admin@apstra-ztp:~$ docker ps
CONTAINER ID    IMAGE           COMMAND                   CREATED         STATUS        PORTS                            NAMES
82ad946682ac    apstra/tftp      "sh /init.sh"            2 months ago     Up 20 hours   0.0.0.0:69->69/udp              tftp
ae71435980be    apstra/nginx     "sh /init.sh"            2 months ago     Up 20 hours   0.0.0.0:80->80/tcp, 0.0.0.0:443->443/
tcp, 0.0.0.0:8080->8080/tcp, 0.0.0.0:31415->31415/tcp   nginx
d28ecf571973    apstra/status    "sh /init.sh"            2 months ago    Up 20 hours   8080/tcp                         status
7af41e63fa89    apstra_ztp_dhcpd  "sh /init.sh"           2 months ago    Up 19 hours   dhcpd
05ea92bc75d9    mysql:8          "docker-entrypoint.s…"  2 months ago    Up 20 hours   306/tcp, 33060/tcp              db
admin@apstra-ztp:~$
```

Preparing for ZTP

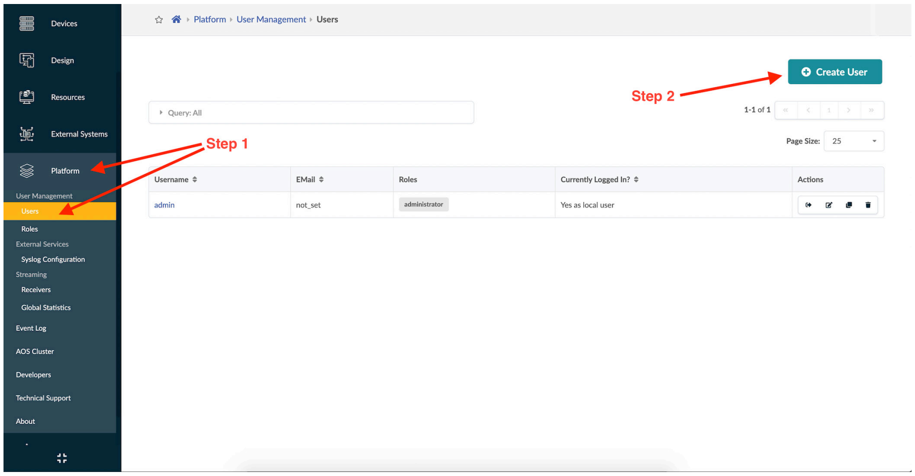
Now for the steps required to perform a ZTP onboarding with the Apstra ZTP server.

Creating a ZTP User

The initial step is creating a ZTP user in Apstra. The ZTP user is not created by default, and will be used by the ZTP server to talk to Apstra. A special user role is already created in Apstra so that any ZTP users are only able to perform ZTP functions. To configure the ZTP user:

Step 1: Go to Platform > Users. See Figure 3.12.

Step 2: Click Create User.



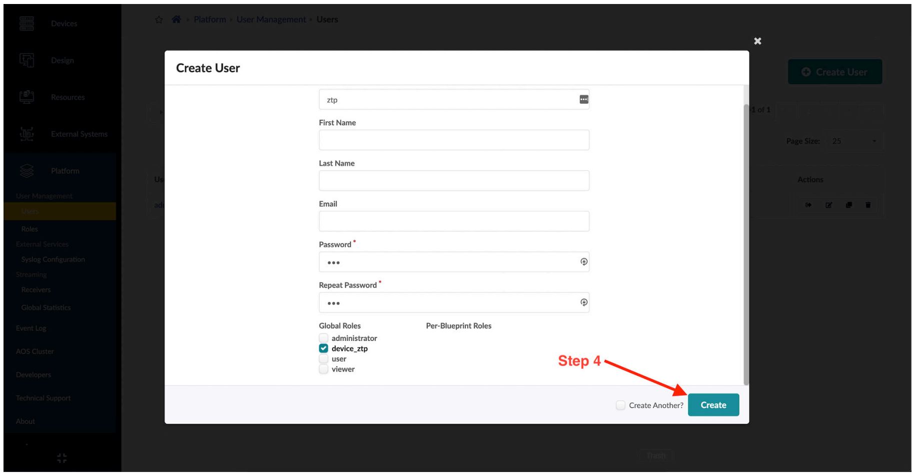
Username	EMail	Roles	Currently Logged In?	Actions
admin	not_set	administrator	Yes as local user	

Figure 3.12 *Creating a New User*

Step 3: Add in the required information. See Figure 3.13.

- Username – ZTP username
- Password: ZTP user password
- Roles: device ZTP role type

Step 4: Click Create User.



Create User

ztp

First Name

Last Name

Email

Password *: •••

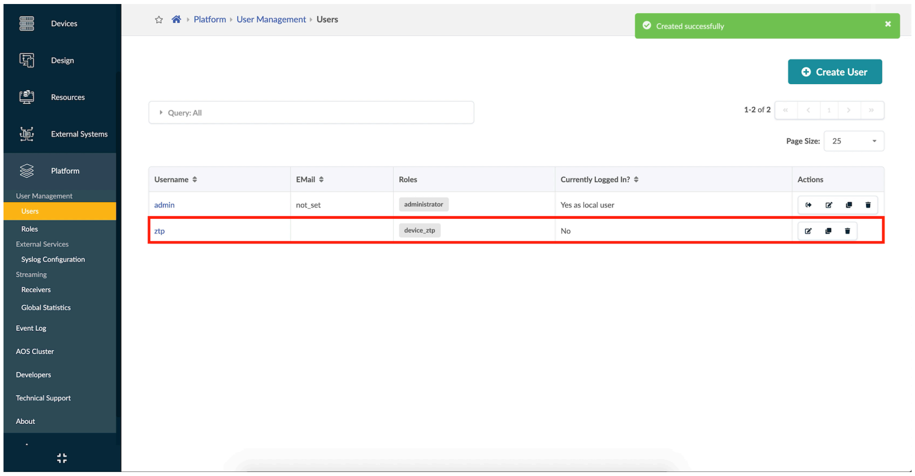
Repeat Password *: •••

Global Roles: administrator, [✓ device_ztp], user, viewer

Per-Blueprint Roles

Figure 3.13 *Creating a New User*

Step 5: See all users by going to Platform > Users as in Figure 3.14.



Username	EMail	Roles	Currently Logged In?	Actions
admin	not_set	administrator	Yes as local user	
ztp		device_ztp	No	

Figure 3.14 ZTP User Post Creation, Ready to Be Used for Onboarding

Once a ZTP user has been created the details can be added to the ZTP server configuration. This will be done later in this chapter after we install the ZTP server.

Install ZTP Server and Set Static IP

The Apstra ZTP server can be downloaded from the Juniper downloads portal as a VMware .ova or a .qcow for easy deployment. Once the .ova or .qcow is installed it is best practice to give the ZTP server a static IP address.

NOTE The Apstra ZTP server uses the credentials below, but these are subject to change, so please consult the latest documentation for up-to-date information (https://www.juniper.net/documentation/product/us/en/apstra):

- Username – admin
- Password – admin

The ZTP server provided via Apstra at the time of writing is based on the Ubuntu 18.04.5 LTS (Long Term Support). Setting a static IP address is done via the Netplan utility. Let's look at the Netplan file used for this book's lab.

Step 1: Boot up the virtual machine image, log in via console with the default credentials (username admin / password admin) and edit the `/etc/netplan/01-netcfg.yaml` file using a tool such as vim or nano. The person editing the file needs to be part of the Sudoers file or be the root user. Here's the 01-netcfg.yaml file for our lab implementation::

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      dhcp4: no
      addresses: [172.30.32.173/25]
      gateway4: 172.30.32.130
      nameservers:
        addresses: [8.8.8.8]
```

Step 2: Perform `sudo netplan apply` to apply the new interface configuration. Note that that `sudo netplan --debug apply` can be used to debug issues caused by applying new interface configuration.

Get EM0 MAC for All Switches (optional)

Collecting the MAC addresses of the switches is optional, however, it can be greatly beneficial in a lab environment. When creating this book, we chose to associate the EM0 management interface hardware MAC address with a host name, so when a switch was onboarded into Apstra it was easy to identify which device it was.

To quickly get the hardware MAC address off a Juniper devices EM0 interface use the `show interfaces em0 | match "Hardware address:"` command. It is important to remember that pre-ZTP this will need to be done via a console connection. Retrieving a Juniper switch em0 MAC address:

```
root@Leaf-1>
root@Leaf-1> show interfaces em0 | match "Hardware address:"
  Current address: 00:c5:2c:8b:62:b8, Hardware address: 00:c5:2c:8b:62:b8
root@Leaf-1>
```

ZTP Host Configuration

To configure the ZTP settings on the Apstra ZTP server there are several files that need editing and configuring: the aos.conf, dhcpd.conf, and ztp.json files.

Aos.conf File

The aos.conf file is very simple and holds the credentials that allow the ZTP server to communicate with Apstra. The structure of the file is JavaScript Object Notation (JSON) and must be written as such.

This file needs to be edited using a tool such as vim or nano. The user editing the file needs to be part of the Sudoers file or be the root user.

- ip – IP address of the Apstra server
- user – the username of the ZTP user created in the previous part of this chapter

- password – the password of the ZTP user created in the previous part of this chapter
- Path to file: /containers_data/status/app/aos.conf

An Apstra aos.conf file example:

```
{
   “ip”: “172.30.32.172”,
   “user”: “ztp”,
   “password”: “ztp”
}
```

This is a JSON file, so it is recommended to use a JSON validator to confirm that your configuration file complies with the JSON standard. Online JSON validators are available for free.

Dhcpd.conf File

Dhcpd is a DHCP server program that operates as a daemon on the ZTP server to provide Dynamic Host Configuration Protocol (DHCP) service to a network. The DHCP implementation that is part of Apstra’s current ZTP server is based on the open-source ISC server. This lightweight DHCP service needs to be provided with a configuration file. Once this configuration file is loaded switches may solicit an IP address from a DHCP server when they boot with a factory default config.

By default, the Apstra ZTP server ships with some pre-defined configuration. For the purposes of this lab environment the needed default configuration and the additional DHCP configuration has been merged. Let’s look at the core aspects of the dhcpd configuration using the path to file: `/containers_data/dhcp/dhcpd.conf`.

The complete file looks like a standard dhcpd configuration and tells the DHCP service the range for which it can distribute IP addresses, the default gateway, and in this book’s case, a MAC address for hostname mapping.

Here’s the example of the complete dhcpd.conf file needed for onboarding devices into Apstra:

```
log-facility local7;
default-lease-time 7200;
max-lease-time 9200;
option space JUNIPER;
option JUNIPER.config-file-name code 1 = text;
option JUNIPER-encapsulation code 43 = encapsulate JUNIPER;
class "juniper" {
    match if (substring(option vendor-class-identifier, 0, 7) = "Juniper");
    option JUNIPER.config-file-name "junos_apstra_ztp_bootstrap.sh";
}
subnet 172.30.32.128 netmask 255.255.255.128 {
    range 172.30.32.223 172.30.32.248;
    option routers 172.30.32.130;
    option domain-name-servers 8.8.8.8, 1.1.1.1;
}
```

```
option tftp-server-name "172.30.32.173";
host leaf1 {
    hardware ethernet 00:C5:2C:8B:62:B8;
    option host-name "Leaf-1";
}
host leaf2 {
    hardware ethernet 00:c5:2c:8b:67:b8;
    option host-name "Leaf-2";
}
host leaf3 {
    hardware ethernet b4:8a:5f:16:29:6c;
    option host-name "Leaf-3";
}
host leaf4 {
    hardware ethernet b4:8a:5f:15:e8:6c;
    option host-name "Leaf-4";
}
host spine1 {
    hardware ethernet 80:AC:AC:71:BC:08;
    option host-name "Spine-1";
}
host spine2 {
    hardware ethernet 80:AC:AC:2F:94:CC;
    option host-name "Spine-2";
}
```

Let's break down the main configuration for the DHCP server a little and explain the steps, starting with the subnet declaration.

Here is the desired subnet that will be served via the DHCP server, including the range; the default gateway is defined:

```
subnet 172.30.32.128 netmask 255.255.255.128 {
range 172.30.32.223 172.30.32.248;
option routers 172.30.32.130;
}
```

You can see the subnet being served is 172.30.32.128/25 and clients can be assigned IP addresses in the 172.30.32.223 to 172.30.32.248 range. The `option routers` command is used to set the default gateway.

The next part of the configuration is the declaration of the TFTP server:

```
option tftp-server-name “172.30.32.173”;
```

The TFTP server holds the ZTP script that will be executed by the QFX switch, which will bootstrap the switch's configuration. The Apstra ZTP script handles the off-box agent creation, local user creation, and can set other system configurations using a custom script.

The final part of the dhcpd.conf file used in this book's lab is the host declaration. The host statement is used to define host specific information. Here, if a client's MAC addresses match the one defined for a given host, that host-name is issued but additional IP addresses for a host can be assigned in this configuration stanza:

```
host leaf1 {
            hardware ethernet 00:C5:2C:8B:62:B8;
            option host-name “Leaf-1”;
        }
```

MORE Additional dhcpd options can be seen here: https://linux.die.net/man/5/dhcpd-options

Ztp.json File

The third configuration file is the ztp.json file, which is used by the ztp.py script at runtime. Each variable defined will be used by the ztp.py script and it is executed locally on each Junos device.

As with the aos.conf file, the ztp.json is written in JavaScript Object Notation (JSON) and must be formatted as such. This is a JSON file, so it is recommended to use a JSON validator to confirm that your configuration file complies with the JSON standard. Online JSON validators are available for free.

The example shown here is very simple and can serve as a good base for onboarding Junos devices, however this file can be a lot more complex if wanted or needed.

This file needs to be edited using a tool such as vim or nano. The user editing the file needs to be part of the Sudoers file or be the root user.

The path to file is: /containers_data/tftp/ztp.json. Here’s the ztp.json file for ZTP onboarding:

```
{
  “defaults”: {
    “junos-versions”: [],
    “junos-image”: “http://server_address/path/to/jinstall_package_name.tgz”,
    “device-root-password”: “password123!”,
    “device-user”: “aosadmin”,
    "device-user-password": "aosadmin",
    "custom-config": "",
    "system-agent-params": {
      "platform": "junos",
      "agent_type": "offbox",
      "install_requirements": true
    }
  }
}
```

Let’s break down these options and explain them:

junos-versions: Valid versions for Juniper Junos devices. If a device is not running a version in this list, ZTP upgrades the device with the junos-image. If this is left blank, then Apstra will accept any Junos version installed and will not attempt to upgrade it.

```
"junos-versions": []
“junos-versions": [ "18.4R3-S4.2" ]
“junos-versions": [ "18.4R3-S4.2", "19.4R3"  ]
```

junos-image: This is the filename of the Junos TGZ image to be loaded if the running version does not match a version in the `junos-versions` list. By default, the image name will be loaded from the ZTP server /container_data/tftp/ directory via TFTP. To use any server for image transfer, enter a valid URL with IP address. For example:

```
"junos-image": http://192.168.59.4/jinstall-host-qfx-5-18.4R3-S4.2-signed.tgz
```

device-root-password: The ZTP process will set the device root password to the Junos device as this value:

```
"device-root-password": "root-password"
```

device-user / device-user-password: This is the username and password Apstra will use for the device system agent. Also, if necessary, the ZTP process will create a user on the device with this username and password:

```
"device-user": "aosadmin",
"device-user-password": "aosadmin-password"
```

custom-config: The filename of the custom configuration shell script in the TFTP directory or a URL pointing to the file on a HTTP server. This shell script will be run during ZTP allowing the user to add custom configuration to the device:

```
"custom-config": ""
```

system-agent-params: The information that Apstra uses to create a new device system agent. This tag holds multiple pieces of information needed by Apstra to create the agent.

agent_type: The agent type needs to be specified as Apstra can create either, on-box or off-box agents. At the time of this writing Junos only supports off-box agents. Here, the agent type defined in the ztp.json file:

```
"agent_type": "offbox"
```

platform: (Required for Apstra off-box agents only.) Set to the device platform (`junos`). Must be in all lowercase.

```
"platform": "junos"
```

Activating Configuration Files

Once these configuration files have edited the dhcpd Docker container needs to be reloaded. This reload will add the new dhcpd configuration file meaning clients will be served IP addresses and the additional information based on your configuration.

The `docker restart dhcpd` command needs to be run as either root or using sudo:

```
docker restart dhcpd
```

The output next verifies that the container restarted after execution of the `docker restart dhcpd` command:

```
root@apstra-ztp:/home/admin# docker restart dhcpd
dhcpd
root@apstra-ztp:/home/admin# docker ps
CONTAINER ID      IMAGE            COMMAND              CREATED            STATUS        PORTS                   NAMES
82ad946682ac      apstra/tftp      "sh /init.sh"         2 months ago       Up 2 days     0.0.0.0:69->69/udp      tftp
ae71435980be      apstra/nginx     "sh /init.sh"         2 months ago       Up 2 days     0.0.0.0:80->80/tcp, 0.0.0.0:443->443/
tcp, 0.0.0.0:8080->8080/tcp, 0.0.0.0:31415->31415/tcp   nginx
d28ecf571973      apstra/status    "sh /init.sh"         2 months ago       Up 2 days     8080/tcp                 status
7af41e63fa89      apstra_ztp_dhcpd  "sh /init.sh"        2 months ago       Up 9 seconds  dhcpd
05ea92bc75d9      mysql:8          "docker-entrypoint.s…"  2 months ago     Up 2 days     3306/tcp, 33060/tcp      db
root@apstra-ztp:/home/admin#
```

Check the dhcpd logs by issuing the `docker logs dhcpd` command. You can see that the DHCP server has successfully started:

```
root@apstra-ztp:/home/admin# docker logs dhcpd
Internet Systems Consortium DHCP Server 4.4.2
Copyright 2004-2020 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/
Config file: /etc/dhcp/dhcpd.conf
Database file: /logs/dhcpd.leases
PID file: /run/dhcp/dhcpd.pid
Wrote 0 class decls to leases file.
Wrote 0 deleted host decls to leases file.
Wrote 0 new dynamic host decls to leases file.
Wrote 7 leases to leases file.
Listening on LPF/eth0/00:0c:29:5a:86:63/172.30.32.128/25
Sending on   LPF/eth0/00:0c:29:5a:86:63/172.30.32.128/25
Sending on   Socket/fallback/fallback-net
Started.
root@apstra-ztp:/home/admin#
```

Troubleshooting the ZTP Process

It's possible that you may need to use some common troubleshooting techniques during the ZTP process. Let's review a few.

DHCP Config File

One of the common issues using the ZTP server is making sure the DHCP server is running, and the configuration was accepted. By issuing the `docker logs dhcpd` command you can see the state of the dhcpd process. In this example you can see the configuration caused an error notice:

```
root@apstra-ztp:/containers_data/dhcp# docker logs dhcpd
ERROR: DHCPD did not start.  retrying with -lf /dhcpd.leases
Internet Systems Consortium DHCP Server 4.4.2
Copyright 2004-2020 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/
/etc/dhcp/dhcpd.conf line 23: semicolon expected.
ption tftp-server-name
       ^
Configuration file errors encountered -- exiting
If you think you have received this message due to a bug rather
than a configuration issue please read the section on submitting
bugs on either our web page at www.isc.org or in the README file
before submitting a bug.  These pages explain the proper
process and the information we find helpful for debugging.
exiting.
ERROR: DHCPD did not start.  (will keep container alive to debug)
got sigterm
root@apstra-ztp:/containers_data/dhcp#
```

DHCP Lease Information

Once the dhcpd process is running correctly it is important to check that clients are getting IP addresses allocated. This can be achieved by looking at the dhcpd lease logs. As this runs in a Docker container, to view the container logs use the following:

```
docker exec dhcpd sh -c "cat /logs/dhcpd.leases"
```

In this dhcp server leases log file, clients are being served and DHCP leases are being accepted:

```
root@apstra-ztp:/home/admin# docker exec dhcpd sh -c "cat /logs/dhcpd.leases"
lease 172.30.32.228 {
  starts 1 2021/04/12 16:57:40;
  ends 1 2021/04/12 19:31:00;
  cltt 1 2021/04/12 16:57:40;
  binding state active;
  next binding state free;
  rewind binding state free;
  hardware ethernet 80:ac:ac:2f:94:cc;
  set vendor-class-identifier = "Juniper-qfx10002-36q-DC730";
  client-hostname "DC730";
}
lease 172.30.32.228 {
  starts 1 2021/04/12 16:57:40;
  ends 1 2021/04/12 16:58:17;
  tstp 1 2021/04/12 16:58:17;
  cltt 1 2021/04/12 16:57:40;
  binding state free;
  hardware ethernet 80:ac:ac:2f:94:cc;
}
root@apstra-ztp:/home/admin#
```

On-box Junos Logs

Another way to troubleshoot the ZTP process is to log in to the Junos device itself via console connection and view the logs. Here the user can examine the DHCP client and Apstra logs.

Junos DHCP logs can be viewed in the *dhcp_logfile* that is present by default. To view the logs simply enter `show log dhcp_logfile`.

The Apstra logs can been seen by going into the Junos devices Linux shell. To start the Linux shell type `start shell` and the command prompt will change.

Starting the Junos shell on a Juniper device:

```
root@Spine-2> start shell
root@Spine-2:RE:0%
```

Once in the shell normal Linux commands can be used. To view the logs any tool can be used, but in the example below the `tail` utility was used.

The Apstra logs are stored in `/var/preserve/apstra/` as well as the ZTP script that is executed. Let's view the Apstra logs on a Juniper device:

```
root@Spine-2:RE:0% tail -f /var/preserve/apstra/aosztp.log
configuration check succeeds
commit complete
Exiting configuration mode
root@Spine-2:RE:0%
2021-04-12 14:57:46,814 INFO Custom config..
2021-04-12 14:57:46,851 INFO Creating System Agent in Apstra..
2021-04-12 14:57:46,885 INFO System Agent params: {'username': u'aosadmin', u'platform': u'junos',
'management_ip': '172.30.32.228', u'agent_type': u'offbox', u'install_requirements': True,
'password': '**masked**'}
2021-04-12 14:57:47,035 INFO System agent created: f9ae246a-6c7c-4680-aa77-9d63811fe03f
2021-04-12 14:57:47,077 INFO Ztp End.
Enabling management VRF
```

System Agent Creation via ZTP Server

Devices connected to Apstra are managed with device agents. The agents handle configuration management, device-to-server communication, and telemetry collection. In order for Apstra to onboard a Junos device a system agent must be created.

The agent creation can be done in two ways, either by the ZTP process or manually via the web UI/API. At the time of this writing Junos devices require an off-box agent, meaning each device onboarded requires its own agent on the Apstra server.

Each individual agent is a separate Docker container meaning a new Docker container is created per Junos devices managed by Apstra. This means in larger fabrics scalability needs to be considered and, in some cases, additional Apstra servers may be required to support agent management.

When using the ZTP server to bootstrap the devices the agent is created automatically using the provided ztp.json file. When the device boots it solicits an IP address via DHCP, this in turn will point the device to the TFTP server. The bootstrap script, ztp.py, is executed on the Junos device and an API call is made to Apstra to create the Agent.

All of this is transparent to the user once the ZTP server has been configured.

Manual Onboarding

Manually onboarding switches to Apstra is a similar process and may be the preferred option for small fabrics. The basic premise of manual onboarding is to apply a base config, also sometimes called a stub file, to the switch so it has basic IP reachability and access. Once this is complete an agent needs to be created in Apstra to establish a session between Apstra and the switch. This can be either an on- or off-box agent, and remember, at the time of this writing Juniper switches only support off-box agents.

Minimum Junos Configuration

A minimum configuration needs to be added when manually creating a system agent to onboard a Junos device.

Here are the basic configurations that need to be applied to a Junos switch for it to be discovered by Apstra:

- IP address – This can either be assigned statically or via DHCP.
- Default static route – The default gateway on the management network.
- Netconf access enabled – The Apstra agent talks to the Junos device via Netconf.
- Root password set – Needed by any Junos device for a commit to be successful.
- Root Login Allowed (optional) – In the minimum Junos configuration example the agent uses the root account, however, a separate user account can be configured.
- Apstra user account created (optional) – Agent can connect to the Junos device with this account.

Here's an example of the minimum configuration required on a Juniper device in order to be onboarded:

```
system {
    management-instance;
    root-authentication {
        encrypted-password <encrypted password>
    }
    services {
        ssh {
            root-login allow;
        }
        netconf {
            ssh;
        }
    }
    host-name Leaf-1;
}
interfaces {
    em0 {
        unit 0 {
            family inet {
                address <IP Address>/<subnet>;
            }
        }
    }
}
routing-instances {
    mgmt_junos {
        routing-options {
            static {
                route 0.0.0.0/0 next-hop <Next Hop Address>;
            }
        }
    }
}
```

This configuration can be deployed in different ways, however the simplest is manually configuring the box via console. Other options, such as using DHCP and TFTP to download this base configuration, are possible.

System Agent Creation via Manual Onboarding

Once the device has basic connectivity to the Apstra server and some basic parameters have been set, such as enabling Netconf, a new device agent can be created to handle configuration management, device-to-server communication, and telemetry collection.

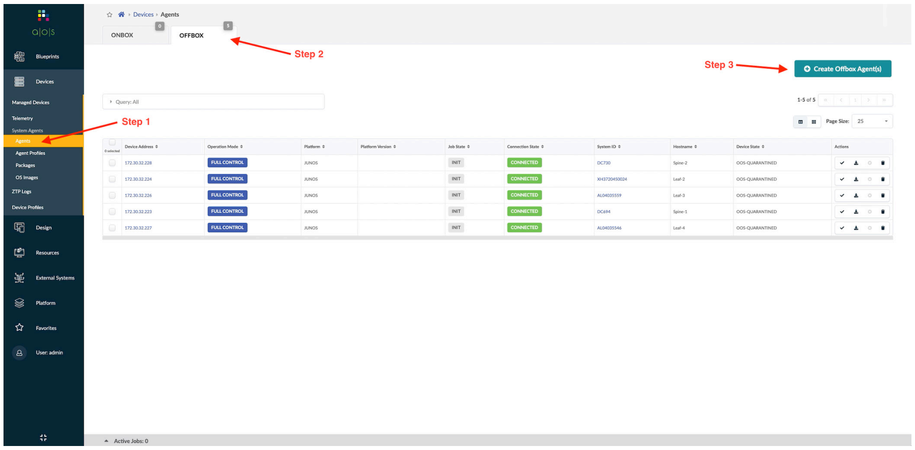
NOTE Each individual agent is a separate Docker container running on Apstra, meaning that in larger fabrics scalability needs to be considered. For example, an off-box agent managing a Juniper device could consume up to 600MB of memory on the Apstra server per switch. To check scalability requirements please visit the latest Apstra documentation.

For a Junos device only a simple off-box agent is needed. Let's walk through how to achieve this and how to view the device agent once created.

Step1: Navigate to Devices > Agents. See Figure 3.15.

Step 2: Select the off-box agent (Junos uses off-box agents only).

Step 3: Click Create Off-box Agent(s).



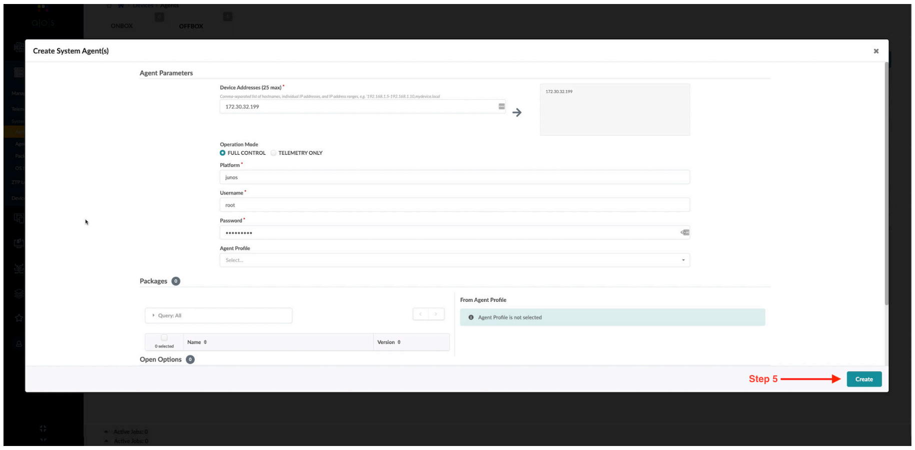
0 selected	Device Address	Operation Mode	Platform	Platform Version	Job State	Connection State	System ID	Hostname	Device State	Actions
	172.30.32.228	FULL CONTROL	JUNOS		INIT	CONNECTED	DC730	Spine-2	OOS-QUARANTINED	
	172.30.32.224	FULL CONTROL	JUNOS		INIT	CONNECTED	XH3720450024	Leaf-2	OOS-QUARANTINED	
	172.30.32.226	FULL CONTROL	JUNOS		INIT	CONNECTED	AL04035559	Leaf-3	OOS-QUARANTINED	
	172.30.32.223	FULL CONTROL	JUNOS		INIT	CONNECTED	DC694	Spine-1	OOS-QUARANTINED	
	172.30.32.227	FULL CONTROL	JUNOS		INIT	CONNECTED	AL04035546	Leaf-4	OOS-QUARANTINED	

Figure 3.15 Creating an Off-box Agent in Apstra

Step 4: Fill in the following information. See Figure 3.16.

- IP address (this can be a list to create multiple agents at once)
- Operational Mode – Full Control is need for Apstra to control the device configuration
- Username – Junos device username
- Password – Junos device password

Step 5: Click Create.



Device Addresses (25 max) *
Comma-separated list of hostnames, individual IP addresses, and IP address ranges, e.g. '192.168.1.5-192.168.1.10,mydevice.local'
172.30.32.199
172.30.32.199

Operation Mode
[✓ FULL CONTROL] TELEMETRY ONLY

Platform *: junos

Username *: root

Password *: •••••••••

Agent Profile: Select...

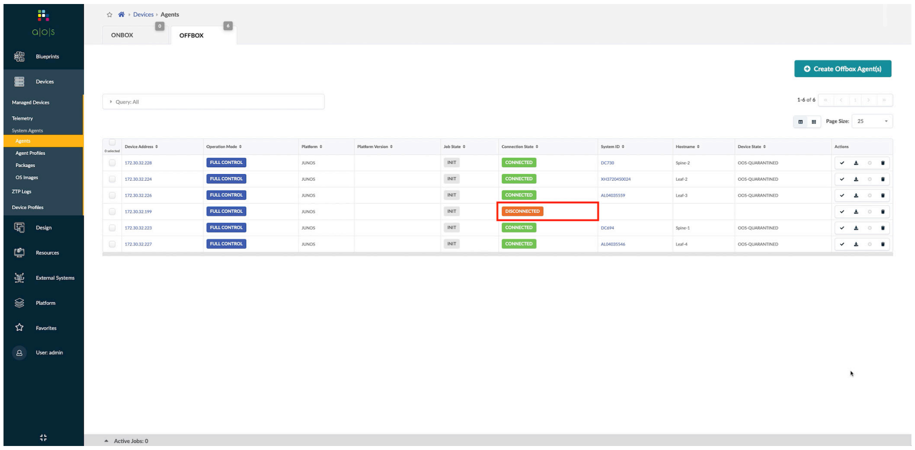
Agent Profile is not selected

0 selected	Name	Version

Figure 3.16 Creating an Off-box Agent in Apstra

Once the agent has been created it will transition through several stages until the agent has synchronized with the Junos device and received the needed information. The states the box transition through are as follows:

- *Disconnected* – The agent has yet to make contact with the device. See Figure 3.17.

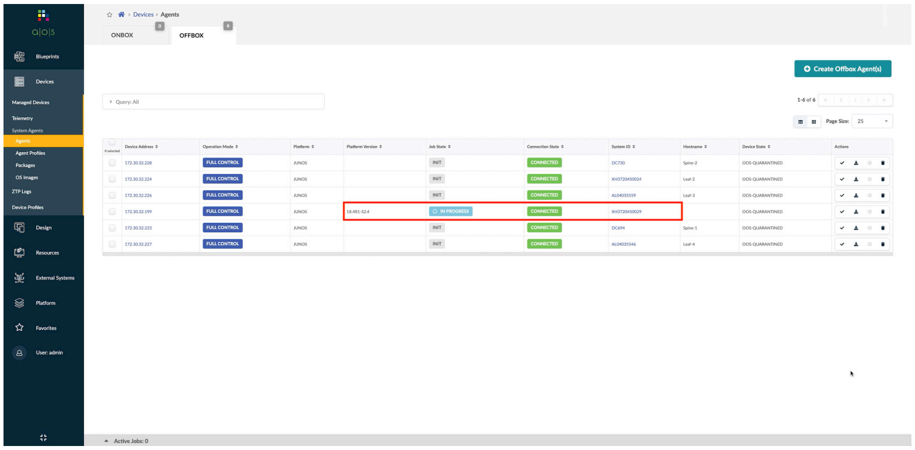

Query: All

1-6 of 6 Page Size: 25

Device Address	Operation Mode	Platform	Platform Version	Job State	Connection State	System ID	Hostname	Device State
172.30.32.228	FULL CONTROL	JUNOS		INIT	CONNECTED	DC730	Spine-2	OOS-QUARANTINED
172.30.32.224	FULL CONTROL	JUNOS		INIT	CONNECTED	XH3720450024	Leaf-2	OOS-QUARANTINED
172.30.32.226	FULL CONTROL	JUNOS		INIT	CONNECTED	AL04035559	Leaf-3	OOS-QUARANTINED
172.30.32.199	FULL CONTROL	JUNOS		INIT	DISCONNECTED			
172.30.32.223	FULL CONTROL	JUNOS		INIT	CONNECTED	DC694	Spine-1	OOS-QUARANTINED
172.30.32.227	FULL CONTROL	JUNOS		INIT	CONNECTED	AL04035546	Leaf-4	OOS-QUARANTINED

Figure 3.17 *Disconnected Agent State*

- *In-Progress, Connected* – The agent has established a connection to the device and is pulling information. This can be seen by clicking on the 'OFFBOX' tab See Figure 3.18.

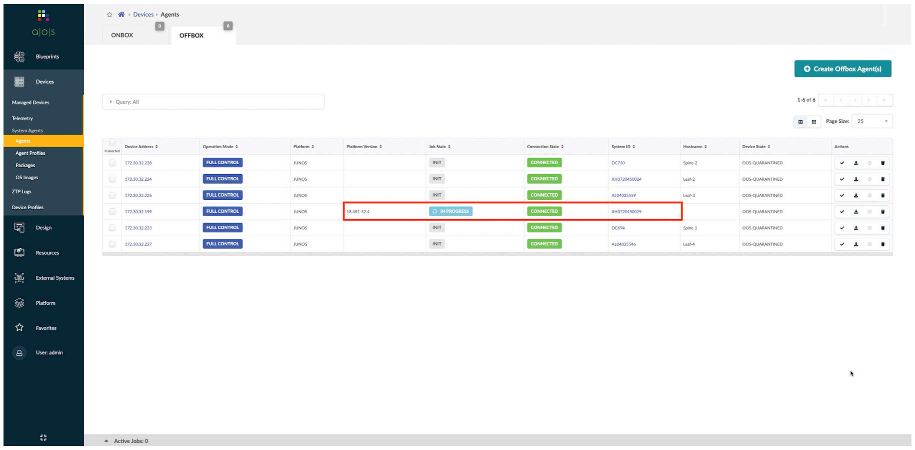

Query: All

1-6 of 6 Page Size: 25

Device Address	Operation Mode	Platform	Platform Version	Job State	Connection State	System ID	Hostname	Device State
172.30.32.228	FULL CONTROL	JUNOS		INIT	CONNECTED	DC730	Spine-2	OOS-QUARANTINED
172.30.32.224	FULL CONTROL	JUNOS		INIT	CONNECTED	XH3720450024	Leaf-2	OOS-QUARANTINED
172.30.32.226	FULL CONTROL	JUNOS		INIT	CONNECTED	AL04035559	Leaf-3	OOS-QUARANTINED
172.30.32.199	FULL CONTROL	JUNOS	18.4R1-S2.4	IN PROGRESS	CONNECTED	XH3720450029		OOS-QUARANTINED
172.30.32.223	FULL CONTROL	JUNOS		INIT	CONNECTED	DC694	Spine-1	OOS-QUARANTINED
172.30.32.227	FULL CONTROL	JUNOS		INIT	CONNECTED	AL04035546	Leaf-4	OOS-QUARANTINED

Figure 3.18 *In Progress, Connected Agent State*

- *Success, Connected* – The agent has established a connection and synced state. See Figure 3.19.



Device Address	Operation Mode	Platform	Platform Version	Job State	Connection State	System ID	Hostname	Device State
172.30.32.228	FULL CONTROL	JUNOS		INIT	CONNECTED	DC730	Spine-2	OOS-QUARANTINED
172.30.32.224	FULL CONTROL	JUNOS		INIT	CONNECTED	XH3720450024	Leaf-2	OOS-QUARANTINED
172.30.32.226	FULL CONTROL	JUNOS		INIT	CONNECTED	AL04035559	Leaf-3	OOS-QUARANTINED
172.30.32.199	FULL CONTROL	JUNOS	18.4R1-S2.4	IN PROGRESS	CONNECTED	XH3720450029		OOS-QUARANTINED
172.30.32.223	FULL CONTROL	JUNOS		INIT	CONNECTED	DC694	Spine-1	OOS-QUARANTINED
172.30.32.227	FULL CONTROL	JUNOS		INIT	CONNECTED	AL04035546	Leaf-4	OOS-QUARANTINED

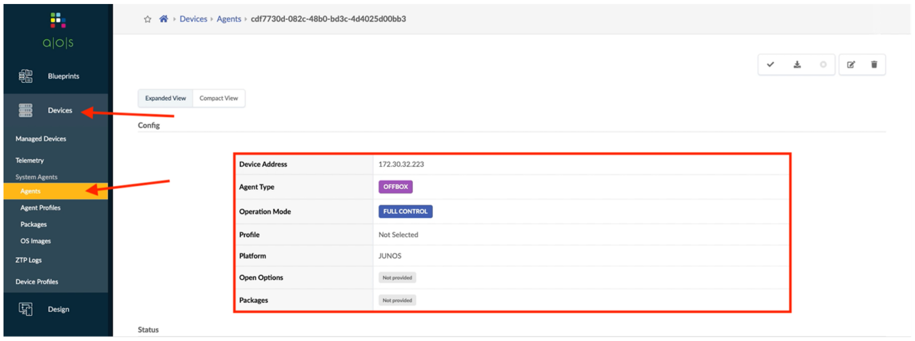
Figure 3.19 Connected Agent State

Post Agent Creation

Once the agents have been created they can be viewed by going to Devices > Agents. See Figure 3.20.

Select a device from the list and more information about the device details is presented:

- Platform (NOS)
- IP Address
- Operation Mode (Full Control or Telemetry Only)
- State (Connected/Disconnected)
- Agents Container Status (Off-box Agents Only)



Device Address	172.30.32.223
Agent Type	OFFBOX
Operation Mode	FULL CONTROL
Profile	Not Selected
Platform	JUNOS
Open Options	Not provided
Packages	Not provided

Figure 3.20 Viewing Device Agents in Apstra

Managed Devices

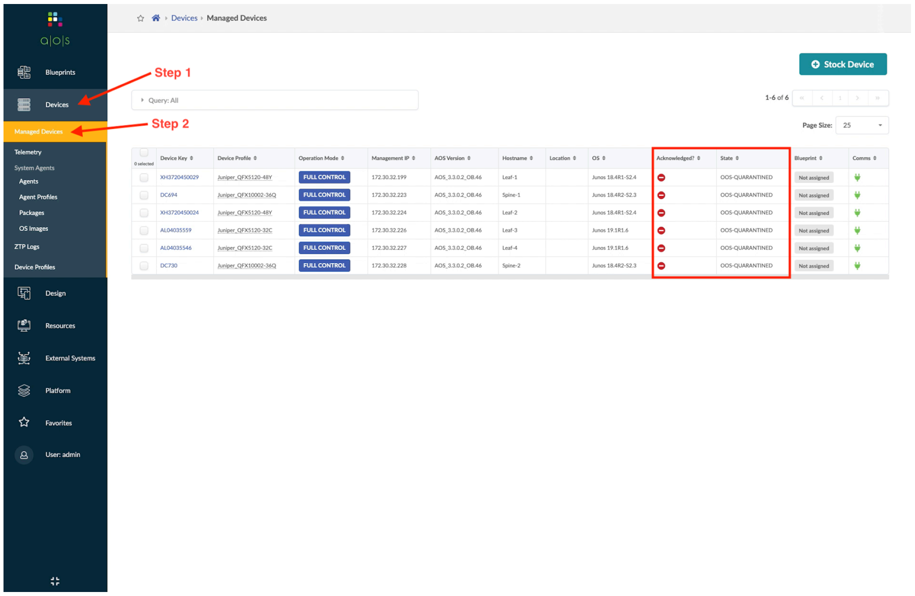
Since the device agents created previously in this chapter provide connectivity between Apstra and the Junos device, and the connection has been established, Apstra needs to formally adopt the switch and bring it under its control. This is done by *acknowledging* the switch.

Acknowledging Devices

Acknowledging devices puts them in the *OOS-READY* state and signals the intent to have Apstra manage them from now on. Prior to acknowledgment, the devices sit in a state called OOS-QUARANTINED, meaning they are not yet controlled by Apstra. To acknowledge devices follow these next steps.

Step1: Navigate to Devices > Managed Devices and take note of the Acknowledge and State columns highlighted in Figure 3.21.

Step 2: Select the devices to be managed by Apstra.

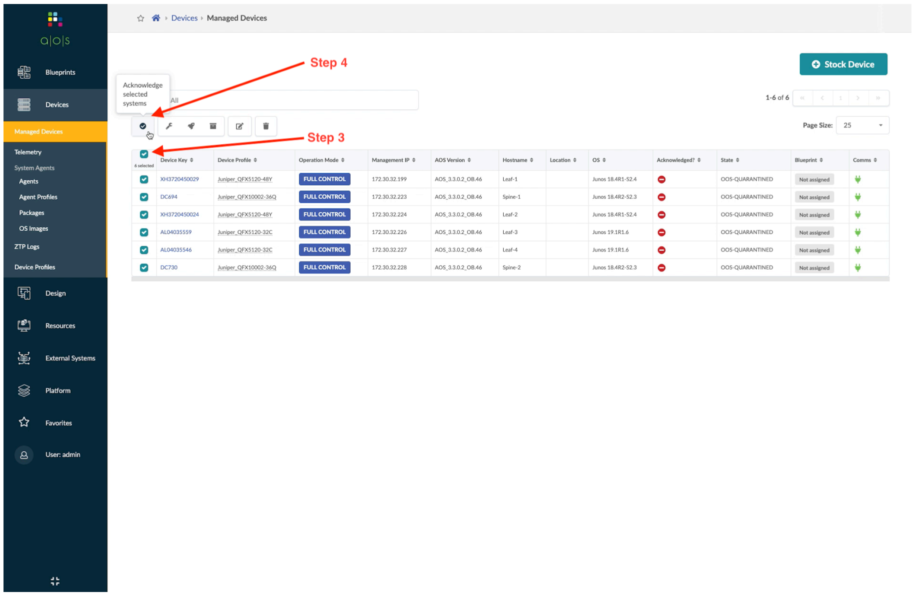


0 selected	Device Key	Device Profile	Operation Mode	Management IP	AOS Version	Hostname	Location	OS	Acknowledged?	State	Blueprint	Comms
☐	XH3720450029	Juniper_QFX5120-48Y	FULL CONTROL	172.30.32.199	AOS_3.3.0.2_OB.46	Leaf-1		Junos 18.4R1-S2.4	⛔	OOS-QUARANTINED	Not assigned	🔌
☐	DC694	Juniper_QFX10002-36Q	FULL CONTROL	172.30.32.223	AOS_3.3.0.2_OB.46	Spine-1		Junos 18.4R2-S2.3	⛔	OOS-QUARANTINED	Not assigned	🔌
☐	XH3720450024	Juniper_QFX5120-48Y	FULL CONTROL	172.30.32.224	AOS_3.3.0.2_OB.46	Leaf-2		Junos 18.4R1-S2.4	⛔	OOS-QUARANTINED	Not assigned	🔌
☐	AL04035559	Juniper_QFX5120-32C	FULL CONTROL	172.30.32.226	AOS_3.3.0.2_OB.46	Leaf-3		Junos 19.1R1.6	⛔	OOS-QUARANTINED	Not assigned	🔌
☐	AL04035546	Juniper_QFX5120-32C	FULL CONTROL	172.30.32.227	AOS_3.3.0.2_OB.46	Leaf-4		Junos 19.1R1.6	⛔	OOS-QUARANTINED	Not assigned	🔌
☐	DC730	Juniper_QFX10002-36Q	FULL CONTROL	172.30.32.228	AOS_3.3.0.2_OB.46	Spine-2		Junos 18.4R2-S2.3	⛔	OOS-QUARANTINED	Not assigned	🔌

Figure 3.21 *Acknowledging Devices in Apstra*

Step 3: Click the checkmark to select the devices.

Step 4: Click the Acknowledge toggle. See Figure 3.22.



Device Key	Device Profile	Operation Mode	Management IP	AOS Version	Hostname	Location	OS	Acknowledged?	State	Blueprint	Comms
XH3720450029	Juniper_QFX5120-48Y	FULL CONTROL	172.30.32.199	AOS_3.3.0.2_OB.46	Leaf-1		Junos 18.4R1-S2.4	⛔	OOS-QUARANTINED	Not assigned	
DC694	Juniper_QFX10002-36Q	FULL CONTROL	172.30.32.223	AOS_3.3.0.2_OB.46	Spine-1		Junos 18.4R2-S2.3	⛔	OOS-QUARANTINED	Not assigned	
XH3720450024	Juniper_QFX5120-48Y	FULL CONTROL	172.30.32.224	AOS_3.3.0.2_OB.46	Leaf-2		Junos 18.4R1-S2.4	⛔	OOS-QUARANTINED	Not assigned	
AL04035559	Juniper_QFX5120-32C	FULL CONTROL	172.30.32.226	AOS_3.3.0.2_OB.46	Leaf-3		Junos 19.1R1.6	⛔	OOS-QUARANTINED	Not assigned	
AL04035546	Juniper_QFX5120-32C	FULL CONTROL	172.30.32.227	AOS_3.3.0.2_OB.46	Leaf-4		Junos 19.1R1.6	⛔	OOS-QUARANTINED	Not assigned	
DC730	Juniper_QFX10002-36Q	FULL CONTROL	172.30.32.228	AOS_3.3.0.2_OB.46	Spine-2		Junos 18.4R2-S2.3	⛔	OOS-QUARANTINED	Not assigned	

Figure 3.22 Acknowledging Devices in Apstra

Step 5: Click Confirm to acknowledge the devices as shown in Figure 3.23.

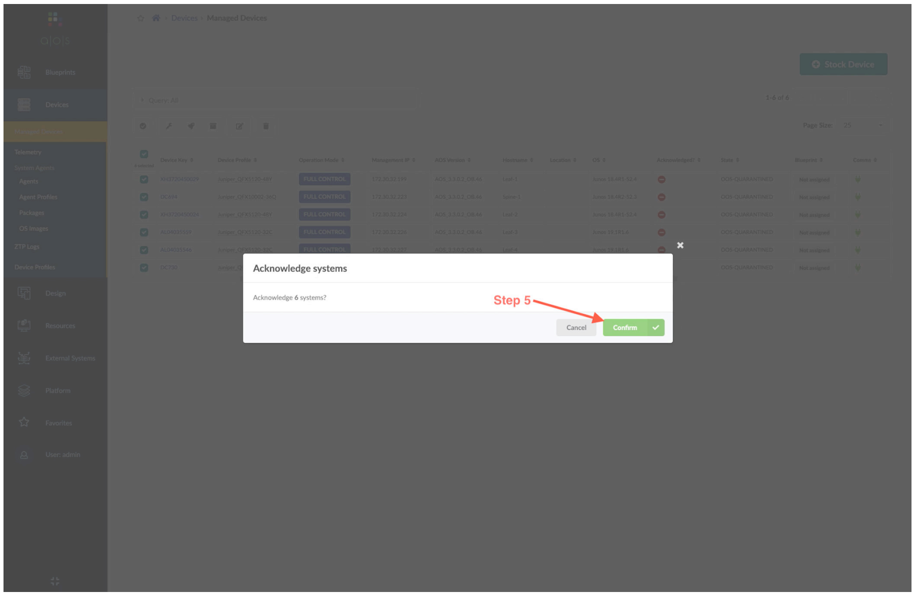
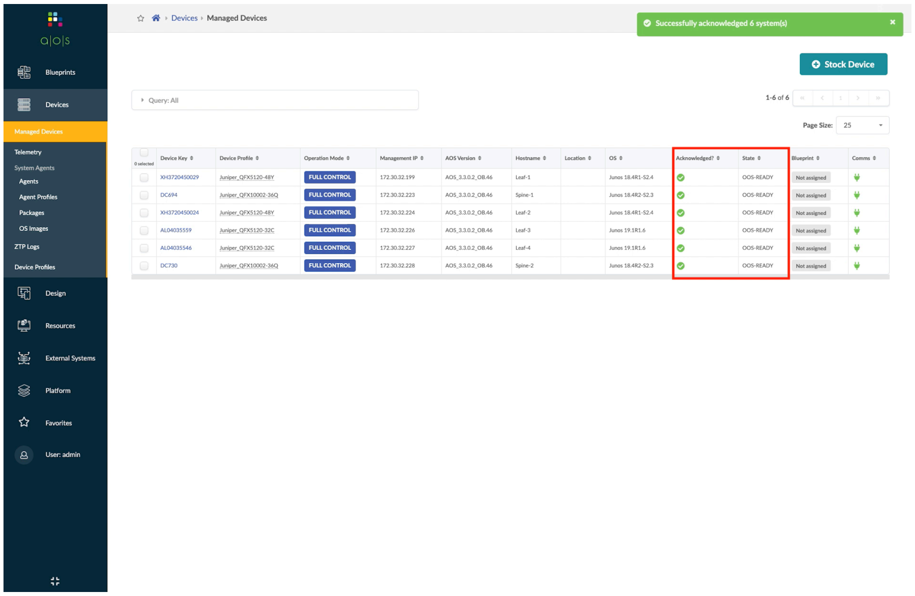


Figure 3.23 Confirming Acknowledgement of Devices in Apstra

Step 6: Once confirmed Apstra pushes an initial basic config called *Discovery 1 Config*. (This is discussed in more detail in the next section).

Step 7: Apstra returns to the list view and the Acknowledged column should show a green tick and the State column should display OOS-READY as in Figure 3.24.



0 selected	Device Key	Device Profile	Operation Mode	Management IP	AOS Version	Hostname	Location	OS	Acknowledged?	State	Blueprint	Comms
	XH3720450029	Juniper_QFX5120-48Y	FULL CONTROL	172.30.32.199	AOS_3.3.0.2_OB.46	Leaf-1		Junos 18.4R1-S2.4	✓	OOS-READY	Not assigned	
	DC694	Juniper_QFX10002-36Q	FULL CONTROL	172.30.32.223	AOS_3.3.0.2_OB.46	Spine-1		Junos 18.4R2-S2.3	✓	OOS-READY	Not assigned	
	XH3720450024	Juniper_QFX5120-48Y	FULL CONTROL	172.30.32.224	AOS_3.3.0.2_OB.46	Leaf-2		Junos 18.4R1-S2.4	✓	OOS-READY	Not assigned	
	AL04035559	Juniper_QFX5120-32C	FULL CONTROL	172.30.32.226	AOS_3.3.0.2_OB.46	Leaf-3		Junos 19.1R1.6	✓	OOS-READY	Not assigned	
	AL04035546	Juniper_QFX5120-32C	FULL CONTROL	172.30.32.227	AOS_3.3.0.2_OB.46	Leaf-4		Junos 19.1R1.6	✓	OOS-READY	Not assigned	
	DC730	Juniper_QFX10002-36Q	FULL CONTROL	172.30.32.228	AOS_3.3.0.2_OB.46	Spine-2		Junos 18.4R2-S2.3	✓	OOS-READY	Not assigned	

Figure 3.24 Post Acknowledgement State of Devices in Apstra – OOS-READY

These devices have now been acknowledged and are fully controlled by Apstra. The devices sit in the OOS-READY state and can be used by Apstra in a DC deployment.

Device States

In the previous section two device states were discussed, *OOS-QUARANTINED* and *OOS-READY*. An onboarded device managed by Apstra is guaranteed to transition through many different states during its lifecycle. This section explains those states from onboarding to deployment.

First it’s important to understand the terminology used by Apstra to identify its various configuration stages:

- *Pristine Config*: Consists of preexisting config, loaded during the ZTP process or via an initial stub file, plus config added during agent installation. Normally, the pristine config does not change throughout the device’s lifecycle.

- *Discovery 1 Config*: Initial basic configuration is added to the device when it is Acknowledged. This includes enabling of LLDP (Link Layer Discovery Protocol) on all interfaces used in the fabric and putting all interfaces into a routed only mode.
- *Discovery 2 Config*: Additional basic configuration is added to the device when it is assigned to a blueprint and deploy mode is "Ready." This includes device hostnames, interface descriptions, and port speed / breakout config.
- *Service Config*: Additional configuration required by Apstra is added when the device's deploy mode is set to "Deploy." When referring to Service Config this means the accumulation of Discovery 1, Discovery 2, and this additional config.
- *Rendered Config*: Complete Apstra rendered configuration for the device, as per the Apstra Reference Design.
- *Incremental Config*: Staged changes. The configuration that will be applied when the staged changes are committed.
- *Golden Config*: When Apstra successfully commits a change this is followed with a new collection of the running config. This is called the *Golden config* and serves as Intent. The running configuration is continuously matched against this Golden config.

The different device states are listed in Table 3.1 with their configuration state during the device lifecycle.

Table 3.1 Apstra Device States

Event	Resulting Device Configuration	Apstra Device State	Apstra Blueprint Deployment
New Device	Factory Default Configuration (Pre-onboarding) or Factory Default Configuration and custom configuration	N/A	Not Assigned
Apstra Device Agent Created	Pristine Config: Factory + Pre- Apstra + Agent Install config	OOS-QUARANTINED	Not Assigned
Acknowledge Device	Discovery 1: Pristine, plus Inter- faces enabled	OOS-READY	Not Assigned
Assign Device to blueprint (not deployed)	Discovery 2: Discovery 1, plus various basic config	IS-READY	Ready
Deploy Device	Service Config: Discovery 2, plus full Apstra Rendered config	IS-ACTIVE	Deploy

Add/Commit Incremental Configuration	Delta of resulting config changes from blueprint modifications	IS-ACTIVE	Deploy
Drain Device	"Drain" Configuration is added	IS-READY	Drain
Undeploy Device	Apstra rendered config is removed	IS-READY	Undeploy
Unassign Device	Discovery 1 config is re-applied	OOS-READY	Not Assigned

Summary

This chapter covered many aspects of device management in Apstra, starting with how a Juniper switch is defined in Apstra and mapped to a logical device using interface mappings. These mappings are mandatory when it comes to deployment of a DC fabric, and fortunately Apstra comes with many of them preconfigured.

The core of this chapter explained different ways to onboard devices to Apstra, either using ZTP or by manually adding devices, including creating device agents. These agents facilitate the communication between Apstra and the Junos devices. As noted, at the time of this writing Junos devices only support off-box agents. These off-box agents are Docker containers run on the Apstra server.

Finally, this chapter demonstrated how discovered devices are acknowledged by Apstra and then throughout their lifecycles as their device states change.

Chapter 4

Deploy an IP Clos Fabric

The previous chapters explored the building blocks required to create an EVPN-VXLAN fabric blueprint with Asptra: logical devices, interface maps, rack templates, resource pools, and DC templates. Once the user has created a design and addressed all the dependencies it is possible to instantiate the fabric.

This chapter focuses on the steps required to deploy an Apstra IP Clos fabric and verify its functionality. Subsequent chapters will discuss how tenants and overlay networks are created in Apstra and how the endpoints communicate.

Blueprints

Apstra uses the concept of *blueprints* to bring network designs to life. The previous design stage ultimately led us to create a template that held all information about how that DC fabric should be created. A blueprint takes a template and creates an instance of a data center fabric that the user can configure and assign devices to. Figure 4.1 shows this in more detail.

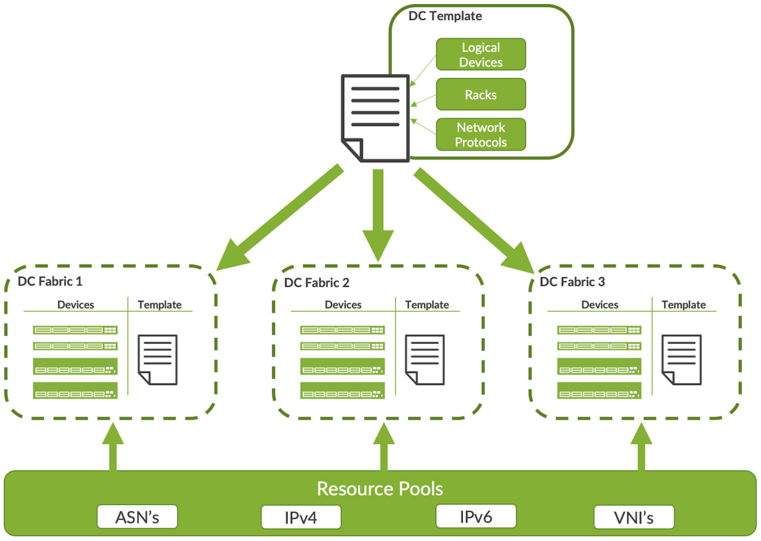


Figure 4.1 *Example of a Template Being Used to Create Data Center Blueprints*

Once the initial creation of the blueprint is complete the user is then able to assign resource pools to the blueprint along with physical devices that have been on-boarded. Device onboarding and resource pool creation was discussed Chapter 3.

The remainder of this chapter focuses on the steps required to create a blueprint for a simple L3 Clos fabric:

- Blueprint Name: DO_DC
- Template Name: DO_DC_Template

Creating Blueprints

The initial step is to create the base blueprint using a template in a straightforward process.

Step 1. Go to Blueprints. See Figure 4.2.

Step 2. Click on the Create Blueprint button on the right.

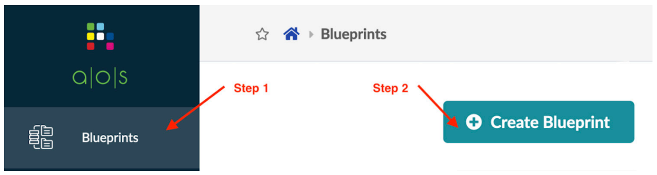
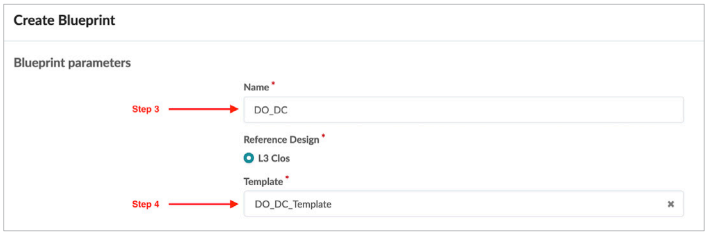


Figure 4.2 *Creating a New Blueprint*

Now select the source DC template, in this case the one created in the previous chapter, and give the blueprint a name. Click Create to create the blueprint and return to the blueprint summary view.

Step 3. Add a name for your new blueprint. See Figure 4.3.

Step 4. Choose a template on which the blueprint will be based.


Create Blueprint

Blueprint parameters

Name *: DO_DC

Reference Design *: ● L3 Clos

Template *: DO_DC_Template

Figure 4.3 *Creating a New Blueprint*

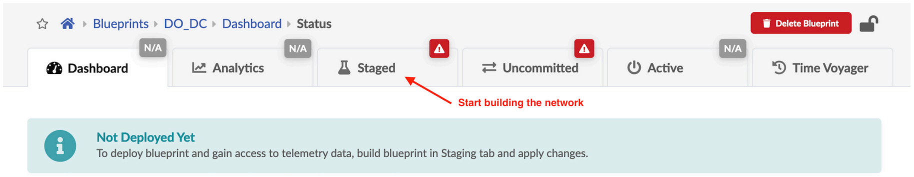
Once the user clicks on Create the initial blueprint is built in Apstra and ready for the user to customize.

Building Blueprints

The next step after creating the blueprint is beginning to build out the remainder of the data center network. The building of the network is done in the blueprint *staged* view. Any changes made in Apstra are done in the staged view, meaning they are not automatically applied; the user must commit the changes before they take effect. This means that operators can make multiple changes to a blueprint and only make a single commit. The staging of the configuration can be done hours before the actual commit, meaning operations teams can preplan and prepare for changes to the network. This approach will be familiar to readers who have experienced the Junos OS.

NOTE “Commit” has a different meaning in Apstra. “Commit” implies applying changes to Apstra’s graph DB and potentially to Junos devices. It depends on the type of changes being made to the blueprint being worked on.

Figure 4.4 shows how to access the staged view. The red status indicator next to a tab means Apstra expects the user to take an action or assign a resource in that specific tab.



Not Deployed Yet

To deploy blueprint and gain access to telemetry data, build blueprint in Staging tab and apply changes.

Figure 4.4 *Starting to Stage a New DC Fabric*

Assigning Resource Pools

Let's show how resource pools are assigned to the blueprint. Based on this book's topology, the resource pools created in previous chapters are listed in Table 4.1.

Table 4.1 Blueprint Resource Information Based on this Book's Topology

Blueprint Resource	Resource Pool
ASNs – Spines	DO_ASN_Fabric
ASNs – Leafs	DO_ASN_Fabric
Loopback IPs – Spines	DO_IP_Loopback
Loopback IPs – Leafs	DO_IP_Loopback
Link IPs	DO_IP_Fabric

To assign resources go to Staged > Physical > Build > Resources.

Step 1. Click on the Staged tab. (See Figure 4.5.)

Step 2. Click on the Physical tab shown below this.

Step 3. Click on the Build button on the right of the screen.

Step 4. Choose the Resources section.

Step 5. Assign resources to the required components (ASNs, IP addresses, etc.).

Step 6. Once a resource is assigned, click its Update Assignments button.

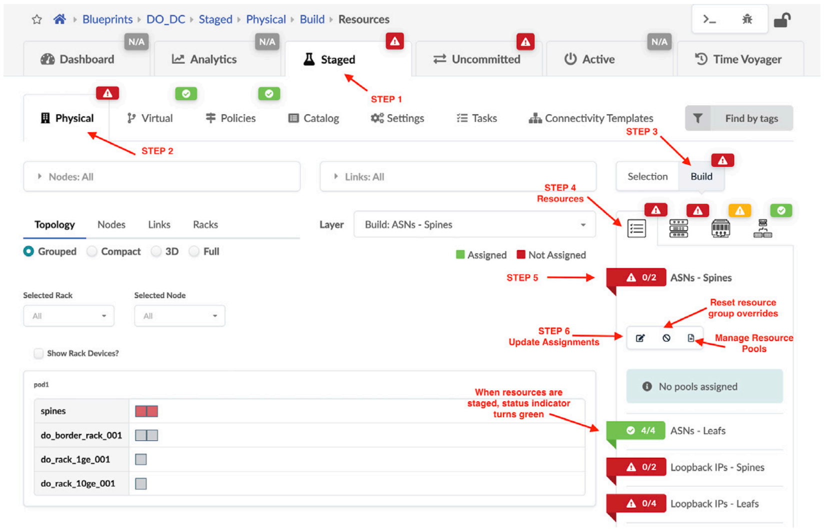


Figure 4.5 Staging a New DC Fabric

When the red status indicator turns green, the resource assignment has been successfully staged as shown in Figure 4.6. It is recommended that you complete each resource within the blueprint before moving on to assign device profiles.

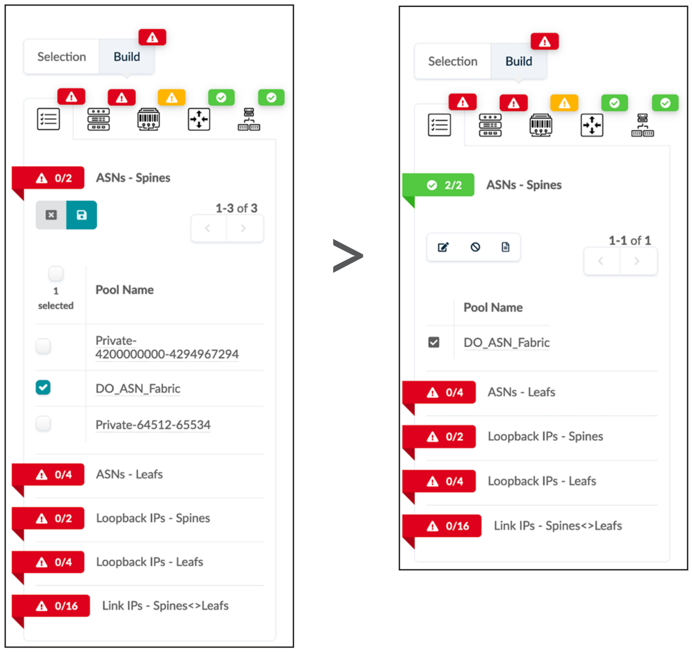


Figure 4.6 *Assigning Resources to Fabric Components*

NOTE You can also assign resources on a per-device basis, which is especially useful if predefined resource mappings are required. For this select the device from the topology view, or nodes view, then assign the resource from the Staged > Physical > Selection > Properties panel section (right-side).

Assigning Device Profiles

The next required step in building a blueprint so it can be deployed is to assign device profiles. Chapter 2 discussed how dummy devices (logical devices) were used as placeholders during the design phase, so that actual hardware models need not be tied to a design. When building the blueprint these logical devices need to map to a device profile (see Chapter 3). Now it's time to point out which physical device type will be used in the actual deployment to the blueprint.

To give context, Table 4.2 shows the deployment mappings.

Table 4.2 Logical Device-to-Device Profile Mapping

Logical Devices	Device Profile	Hardware Model
AOS-32x40-1	DO_JUNIPER_32x40	QFX5120-32C
DO_72x10_18x40	DO_JUNIPER_72x10_16x40	QFX10002-36Q
DO_48x1_8x40	DO_JUNIPER_48x1_8x40	QFX5120-48Y
AOS-48x10+8x40-1	Juniper_QFX5120-48Y_Junos__AOS-48x10_8x40-1	QFX5120-48Y

You can see these interface maps in Table 4.2:

- The custom DO_JUNIPER_32x40 maps to the QFX5120-32C Spine switch-es.
- The custom DO_JUNIPER_72x10_16x40 maps to the QFX10002-36Q Border Leaf switches.
- The custom DO_JUNIPER_48x1_8x40 maps to the QFX5120-48Y Server Leaf switch deployed for 1GbE server access.
- The default Juniper_QFX5120-48Y_Junos__AOS-48x10_8x40-1 maps to the QFX5120-48Y Server Leaf switch deployed for 10GbE server access.

The first three DO_JUNIPER interface maps were created and customized in a previous chapter. The last interface map comes pre-built into Apstra by default and didn't need to be created or modified.

The numbers included in the Interface Map names represent connectivity options: for example, 32x40 means the map supports 32 ports at 40Gbbps each, 48x1_8x40 means the map supports 48 ports at 1Gbps and 8 ports at 40Gbps, etc.

Okay now let's show how to assign devices profiles to the blueprint.

Step 1. Click on the Staged tab. See Figure 4.7

Step 2. Click on the Physical tab shown below this.

Step 3. Click on the Build button on the right of the screen.

Step 4. Choose the Device Profiles section. Device profiles are staged by assigning interface maps.

Step 5. Choose devices shown in red one by one.

Step 6. Click their red status indicator, then click the Change interface maps assignment button (looks like an edit button). Device profiles are staged by assigning interface maps.

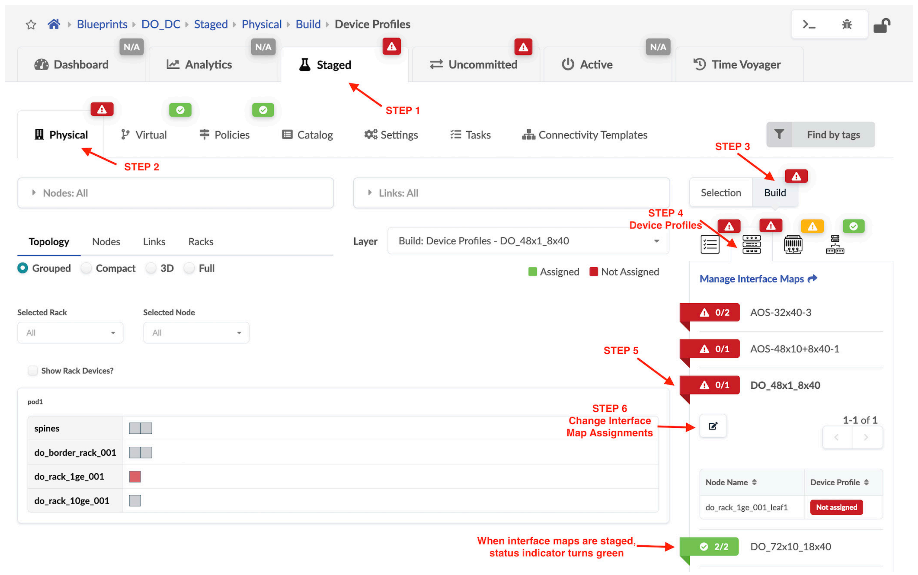


Figure 4.7 *Assigning Resources to Fabric Components*

NOTE Optionally (not pictured here for brevity) if you want to assign the same interface map to multiple nodes, select the ones that use the same interface map (or all of them with one click), then select the interface map from the drop-down list located above the selections, and click Assign Selected. Finally, click Update Assignments to finish this step.

Assigning Devices

The final stage of creating a basic fabric is assigning the devices to the blueprint. As discussed previously, devices must have interface maps associated with them before they can have system IDs assigned.

When a device is assigned to a blueprint, it performs a discovery configuration. During this phase all interfaces are changed to L3-only mode allowing interfaces to be up. There is no BGP configuration and no routing expectations – nothing that can influence the network. A device in discovery mode is benign; it does not participate in the data center fabric, and it does not forward any packets. Any issues, such as mis-cabling or physical link errors, will cause a telemetry alarm. These can be addressed and corrected before deploying the device.

The following steps demonstrate how to assign devices to the blueprint.

Step 1. Click on the Staged tab. See Figure 4.8.

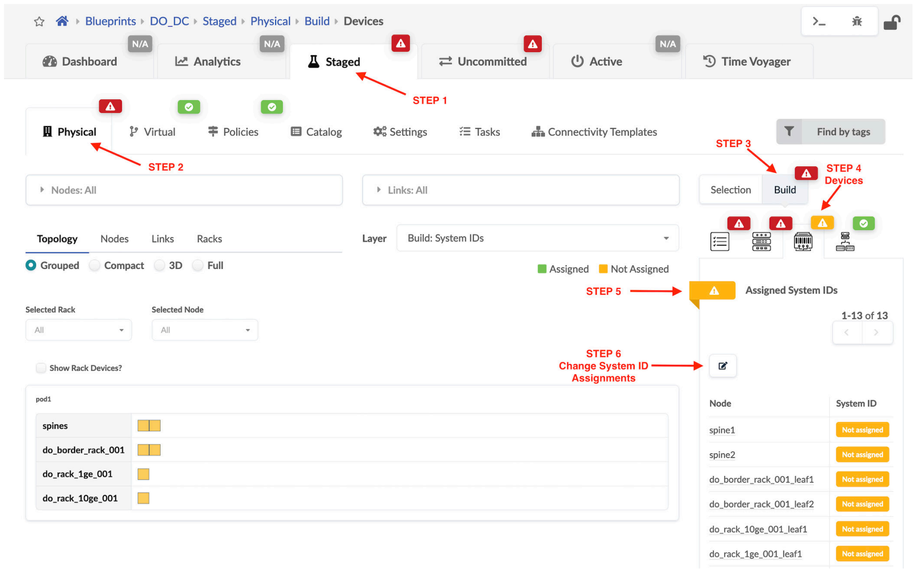
Step 2. Click on the Physical tab.

Step 3. Click on the Build button to the right of the screen.

Step 4. Choose the Devices tab.

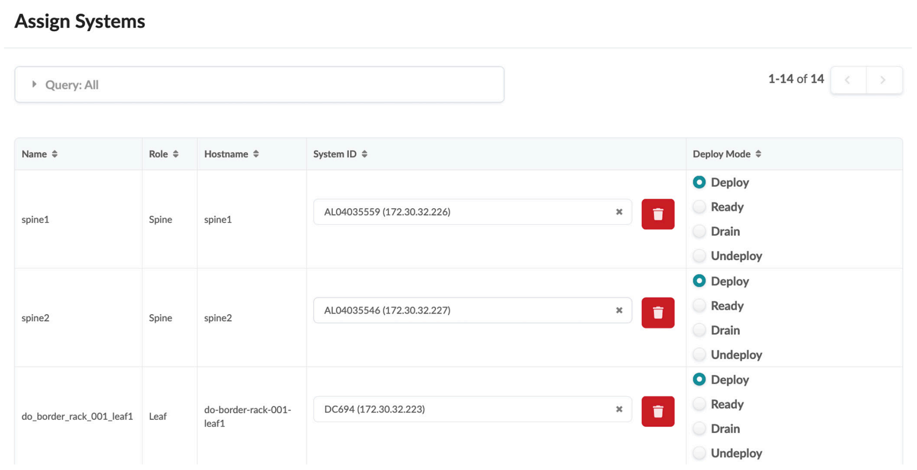
Step 5. Click the status indicator for Assigned System IDs if the nodes list is not already displayed. Unassigned devices are indicated in yellow.

Step 6. Click the Change System IDs assignments button (below Assigned System IDs) and, for each node, select system IDs from the drop-down list shown in Figure 4.9.



Node	System ID
spine1	Not assigned
spine2	Not assigned
do_border_rack_001_leaf1	Not assigned
do_border_rack_001_leaf2	Not assigned
do_rack_10ge_001_leaf1	Not assigned
do_rack_1ge_001_leaf1	Not assigned

Figure 4.8 Assigning System IDs to Fabric Devices



Assign Systems

Query: All

1-14 of 14

Name	Role	Hostname	System ID	Deploy Mode
spine1	Spine	spine1	AL04035559 (172.30.32.226)	[✓ Deploy] Ready Drain Undeploy
spine2	Spine	spine2	AL04035546 (172.30.32.227)	[✓ Deploy] Ready Drain Undeploy
do_border_rack_001_leaf1	Leaf	do-border-rack-001-leaf1	DC694 (172.30.32.223)	[✓ Deploy] Ready Drain Undeploy

Figure 4.9 Assigning System IDs to Fabric Devices

When a system ID is selected, the deploy mode changes to Deploy by default. If the user does not want to deploy the device during the initial and subsequent commits, the deploy mode can be changed here. When ready to deploy the device, return here to set the deploy mode back to Deploy. See the next section for more information about deploy modes. Click Update Assignments to stage the changes.

NOTE Users can set the state to “Ready” rather than “Deploy” to validate the fabric cabling and once verified changing the state to “Deploy” so that initial configuration can be pushed to devices. This method stops cabling anomalies being raised after an initial deployment.

Okay, with all of these steps enabled a basic fabric can be deployed ready for virtual overlays. Let’s commit the staged configurations just created and show how to verify a successful deployment.

Committing and Verifying Configurations

The desired fabric configuration and connectivity is now complete and ready to be committed. During a commit all changes are merged into a *device specific service configuration*. A service configuration holds an entire device’s config and is generated automatically by Apstra.

The yellow warning triangle on the uncommitted tab (See Figure 4.10) informs the user that there is staged configuration waiting to be committed.

The following steps show how a commit is performed within a blueprint.

Step 1. Go to the Uncommitted tab at the top. See Figure 4.10.

Step 2. Click the Commit button on the top right.

Step 3. When a new pop-up asks for a Revision Description just add your custom description and click the Commit button. (Not shown.)

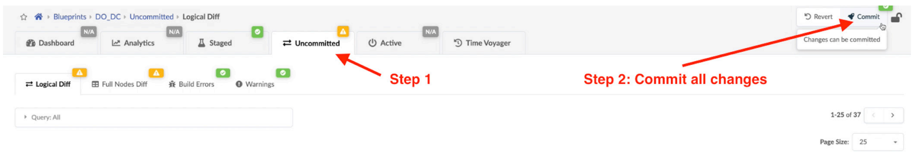


Figure 4.10 Commuting Configuration Changes to the Blueprint

After a few minutes, during which Apstra configures all switches according to the intended design, you can go to the Dashboard tab to confirm that Apstra has successfully deployed the service config to all six devices and that all probe indicators in the Anomalies section are green (no functional errors reported) see Figure 4.11.

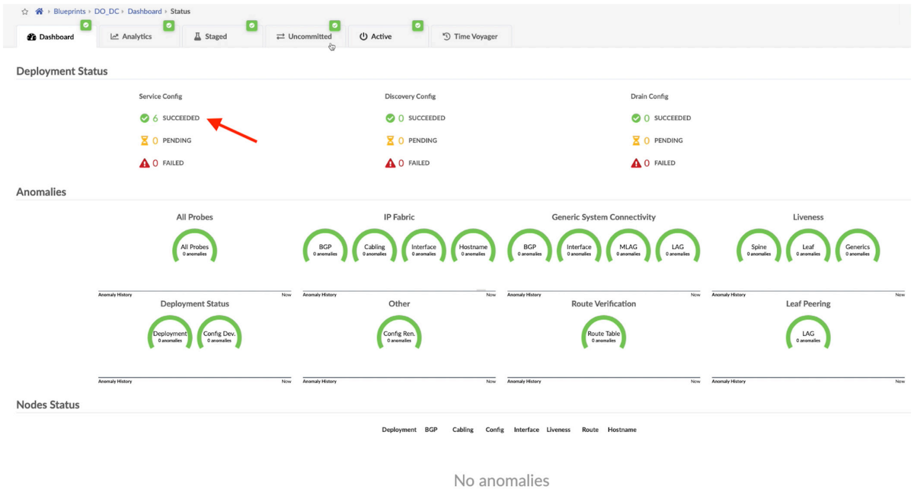
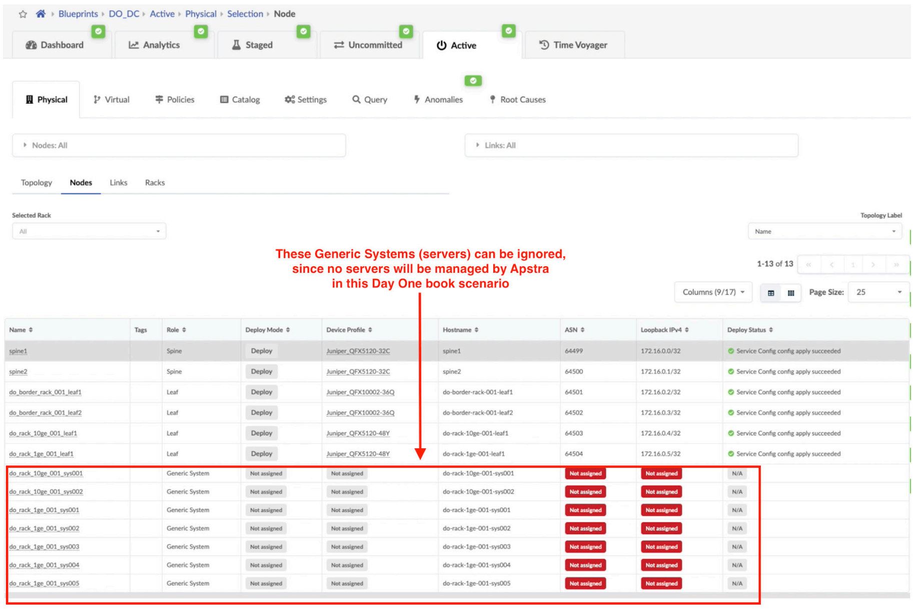


Figure 4.11 Dashboard of a Newly Deployed Blueprint

In Figure 4.12, the Active > Physical tab shows that all services are green, meaning no anomalies are detected, and that the service config was successfully pushed to each device.

You can also see a number of Generic System devices that don't have an ASN nor a Loopback assigned. These generic systems are part of the rack template used to build this fabric. In the case of this book's lab the only servers being used are connecting to the fabric via Layer 2 and do not have a requirement for running BGP on them.



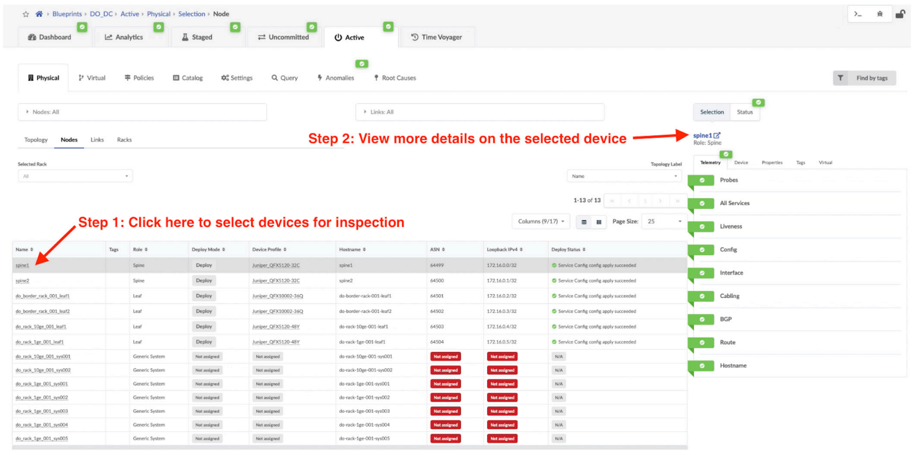
Name	Tags	Role	Deploy Mode	Device Profile	Hostname	ASN	Loopback IPv4	Deploy Status
spine1		Spine	Deploy	Juniper_QFX5120-32C	spine1	64499	172.16.0.0/32	Service Config config apply succeeded
spine2		Spine	Deploy	Juniper_QFX5120-32C	spine2	64500	172.16.0.1/32	Service Config config apply succeeded
do_border_rack_001_leaf1		Leaf	Deploy	Juniper_QFX10002-36Q	do-border-rack-001-leaf1	64501	172.16.0.2/32	Service Config config apply succeeded
do_border_rack_001_leaf2		Leaf	Deploy	Juniper_QFX10002-36Q	do-border-rack-001-leaf2	64502	172.16.0.3/32	Service Config config apply succeeded
do_rack_10ge_001_leaf1		Leaf	Deploy	Juniper_QFX5120-48Y	do-rack-10ge-001-leaf1	64503	172.16.0.4/32	Service Config config apply succeeded
do_rack_1ge_001_leaf1		Leaf	Deploy	Juniper_QFX5120-48Y	do-rack-1ge-001-leaf1	64504	172.16.0.5/32	Service Config config apply succeeded
do_rack_10ge_001_sys001		Generic System	Not assigned	Not assigned	do-rack-10ge-001-sys001	Not assigned	Not assigned	N/A
do_rack_10ge_001_sys002		Generic System	Not assigned	Not assigned	do-rack-10ge-001-sys002	Not assigned	Not assigned	N/A
do_rack_1ge_001_sys001		Generic System	Not assigned	Not assigned	do-rack-1ge-001-sys001	Not assigned	Not assigned	N/A
do_rack_1ge_001_sys002		Generic System	Not assigned	Not assigned	do-rack-1ge-001-sys002	Not assigned	Not assigned	N/A
do_rack_1ge_001_sys003		Generic System	Not assigned	Not assigned	do-rack-1ge-001-sys003	Not assigned	Not assigned	N/A
do_rack_1ge_001_sys004		Generic System	Not assigned	Not assigned	do-rack-1ge-001-sys004	Not assigned	Not assigned	N/A
do_rack_1ge_001_sys005		Generic System	Not assigned	Not assigned	do-rack-1ge-001-sys005	Not assigned	Not assigned	N/A

Figure 4.12 *Placeholders for Generic Systems Included In the Standard Blueprint by Default*

In the Active > Physical tab the user can inspect each Fabric device by clicking on its name and opening its own detailed view. See Figure 4.13.

Step 1. Select the device you want to inspect from the list in the Active tab.

Step 2. Click the button with the device's hostname, which opens the detailed view.



Name	Tags	Role	Deploy Mode	Device Profile	Hostname	ASN	Loopback IPv4	Deploy Status
spine1		Spine	Deploy	Juniper_QFX5120-32C	spine1	64499	172.16.0.0/32	Service Config config apply succeeded
spine2		Spine	Deploy	Juniper_QFX5120-32C	spine2	64500	172.16.0.1/32	Service Config config apply succeeded
do_border_rack_001_leaf1		Leaf	Deploy	Juniper_QFX10002-36Q	do-border-rack-001-leaf1	64501	172.16.0.2/32	Service Config config apply succeeded
do_border_rack_001_leaf2		Leaf	Deploy	Juniper_QFX10002-36Q	do-border-rack-001-leaf2	64502	172.16.0.3/32	Service Config config apply succeeded
do_rack_10ge_001_leaf1		Leaf	Deploy	Juniper_QFX5120-48Y	do-rack-10ge-001-leaf1	64503	172.16.0.4/32	Service Config config apply succeeded
do_rack_1ge_001_leaf1		Leaf	Deploy	Juniper_QFX5120-48Y	do-rack-1ge-001-leaf1	64504	172.16.0.5/32	Service Config config apply succeeded
do_rack_10ge_001_sys001		Generic System	Not assigned	Not assigned	do-rack-10ge-001-sys001	Not assigned	Not assigned	N/A
do_rack_10ge_001_sys002		Generic System	Not assigned	Not assigned	do-rack-10ge-001-sys002	Not assigned	Not assigned	N/A
do_rack_1ge_001_sys001		Generic System	Not assigned	Not assigned	do-rack-1ge-001-sys001	Not assigned	Not assigned	N/A
do_rack_1ge_001_sys002		Generic System	Not assigned	Not assigned	do-rack-1ge-001-sys002	Not assigned	Not assigned	N/A
do_rack_1ge_001_sys003		Generic System	Not assigned	Not assigned	do-rack-1ge-001-sys003	Not assigned	Not assigned	N/A
do_rack_1ge_001_sys004		Generic System	Not assigned	Not assigned	do-rack-1ge-001-sys004	Not assigned	Not assigned	N/A
do_rack_1ge_001_sys005		Generic System	Not assigned	Not assigned	do-rack-1ge-001-sys005	Not assigned	Not assigned	N/A

Figure 4.13 *Inspecting Newly Deployed Devices*

Let's look at a detailed view of spine1 (Step 1 in Figure 4.13) in a new window. Apstra shows a graphical representation of all links connecting to its fabric neighbors under the Active > Physical tab as shown in Figure 4.14.

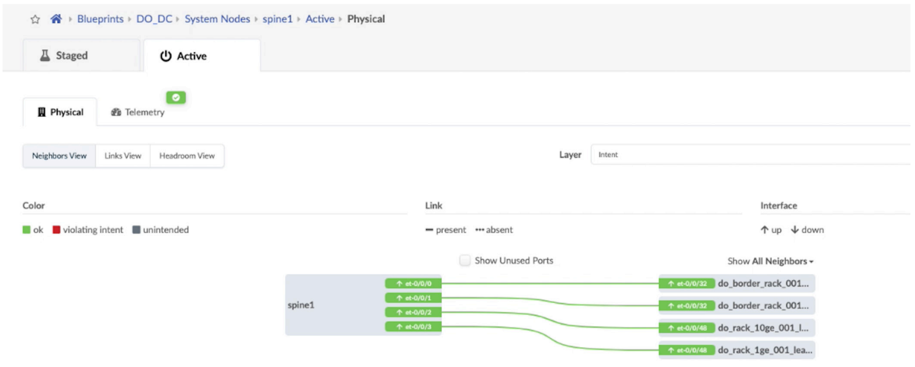
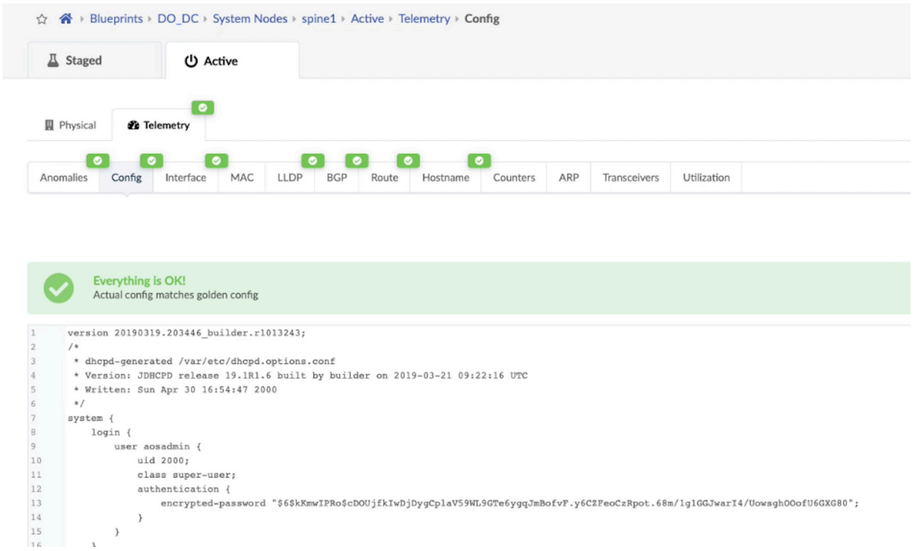

Blueprints › DO_DC › System Nodes › spine1 › Active › Physical
Staged
Active
Physical
Telemetry
Neighbors View
Links View
Headroom View
Layer
Intent
Color
🟩 ok 🟥 violating intent ▨ unintended
Link
— present ••• absent
Interface
↑ up ↓ down
Show Unused Ports
Show All Neighbors
spine1
et-0/0/0
et-0/0/1
et-0/0/2
et-0/0/3
et-0/0/32 do_border_rack_001...
et-0/0/32 do_border_rack_001...
et-0/0/48 do_rack_10ge_001_l...
et-0/0/48 do_rack_1ge_001_lea...

Figure 4.14 *Active Interfaces on a Newly Deployed Fabric Spines*

If the user wants to view the overall node configuration generated by Apstra, they can do so by opening the Active > Telemetry > Config tab as shown in Figure 4.15. The initial configuration should always be matching the golden config as shown:



Everything is OK!
Actual config matches golden config

```
version 20190319.203446_builder.r1013243;
/*
 * dhcpd-generated /var/etc/dhcpd.options.conf
 * Version: JDHCPD release 19.1R1.6 built by builder on 2019-03-21 09:22:16 UTC
 * Written: Sun Apr 30 16:54:47 2000
 */
system {
    login {
        user aosadmin {
            uid 2000;
            class super-user;
            authentication {
                encrypted-password "$6$kKmwIPRo$cDOUjfkIwDjDygCplaV59WL9GTe6ygqJmBofvF.y6CZFeoCzRpot.68m/1g1GGJwarI4/UowsghOOofU6GXG80";
            }
        }
```

Figure 4.15 *Viewing the Device's Configuration Generated by Apstra*

Other details available to check in the other tabs are the LLDP neighbors and details, BGP neighbors, ARP/MAC records, interface counters, and the make and model of all transceivers installed.

Summary

A complete fully working DC fabric has been deployed automatically by Apstra, combining the user's intended design with Juniper best practices, without requiring the user to manually produce or apply any custom Junos configuration.

The next chapter, Chapter 5, adds the first Tenant / Routing Zone with two virtual networks for services to communicate.

Chapter 5

Intra-tenant Routing

Chapter 4 discussed and demonstrated how to build and deploy an IP Clos fabric using Apstra. Building an underlaying Clos fabric is a big step towards a fully functioning DC, however the overlay network now needs to be created and applied. An overlay network provides connectivity between endpoints in the same VXLAN network and allows routing between hosts that are located in the same tenant.

This chapter expands on Chapter 4, showing how to add a tenant to the fabric and then to multiple VXLANs. Later in this chapter intra-tenant routing is demonstrated. The design for the first tenant is shown in Figure 5.1; the tenant is called *Production* and within this tenant two virtual networks exist, *Prod-DB* and *Prod-App*.

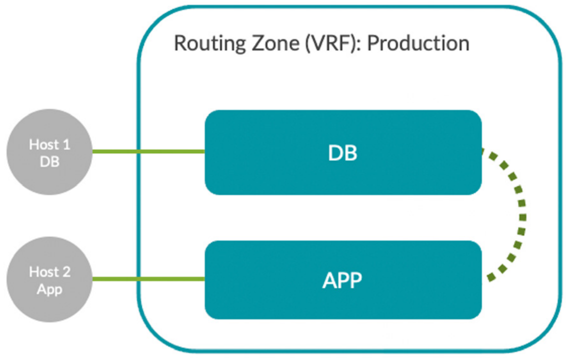


Figure 5.1 *Example Tenant Design*

Routing Zones

The first step when creating an overlay network is to define a *routing zone*. A routing zone can be considered as a user community (such as a company, or departments) that contains groups of endpoints. Creating a routing zone in Apstra means creating a virtual routing and forwarding (VRF) instance on each switch in the fabric. These routing zones provide logical separation, permitting multiple tenants to exist on top of the same DC fabric infrastructure without being able to communicate with one another and allowing for overlapping IP subnets.

NOTE By default Apstra creates a routing zone called *default*. The default routing zone is reserved for the underlay network. VXLAN based virtual networks must be associated with a non-default routing zone.

Per this book's lab the following routing zone will be created:

- Name: Production
- VLAN-ID: Auto-Assign
- VNI: Auto-Assign
- Routing Policies: N/A

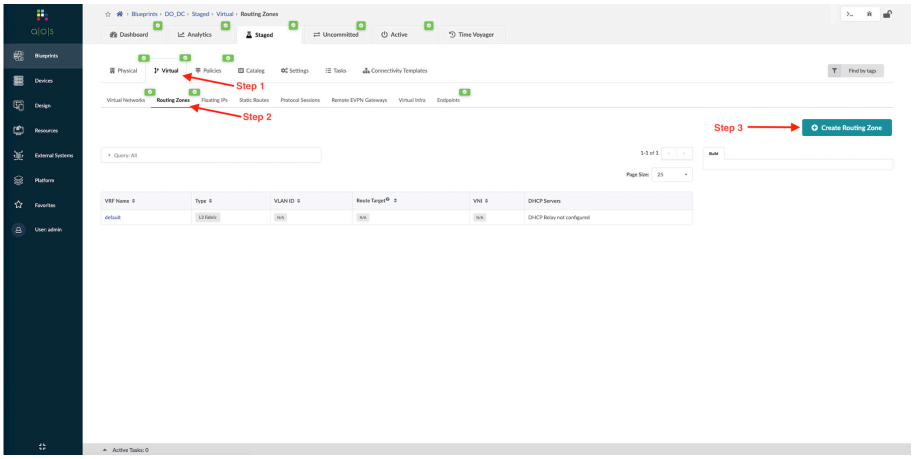
Creating a Routing Zone

Creating a routing zone is a simple process.

Step 1: Within the desired blueprint select Staged > Virtual. See Figure 5.2.

Step 2: Select Routing Zone.

Step 3: Then click Create Routing Zone.

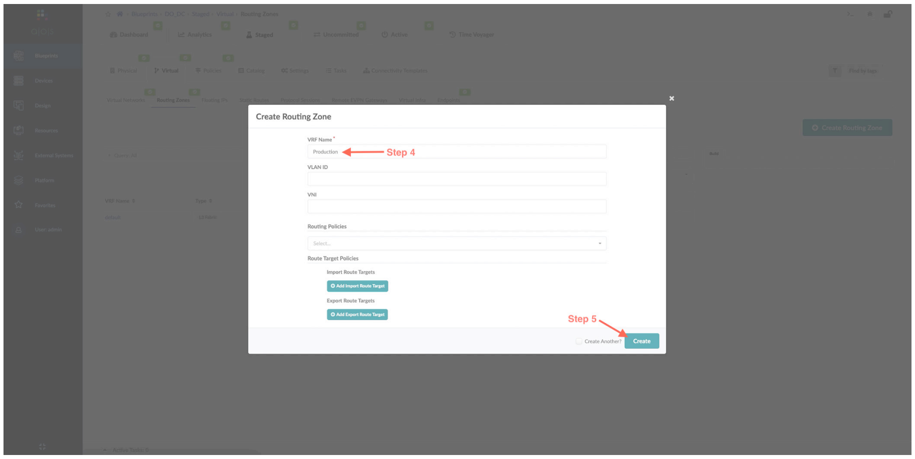


VRF Name	Type	VLAN ID	Route Target	VNI	DHCP Servers
default	L3 Fabric	N/A	N/A	N/A	DHCP Relay not configured

Figure 5.2 Adding A Routing Zone via Apstra

Step 4: For a simple Routing Zone in Figure 5.3, only a VRF Name is required. VXLAN and VLAN IDs are automatically assigned in later steps.

Step 5: Click Create


Create Routing Zone

VRF Name *: Production

VLAN ID:

VNI:

Routing Policies: Select...

Route Target Policies

Import Route Targets

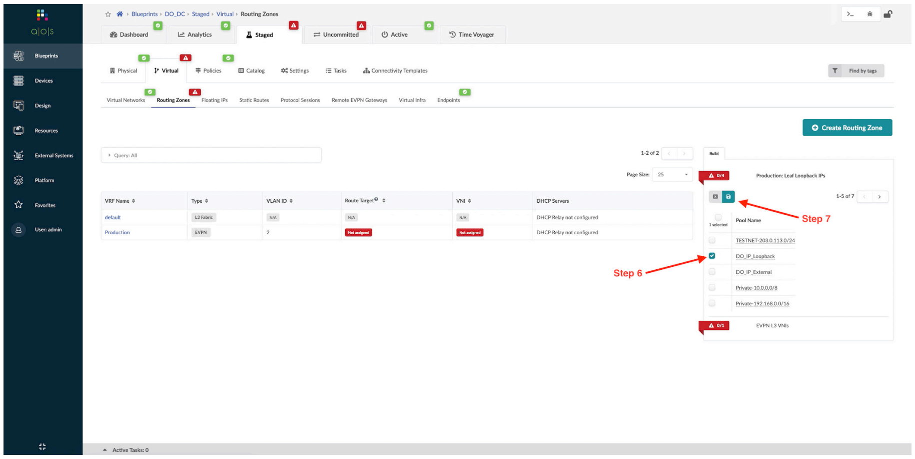
Export Route Targets

Figure 5.3 *Adding A Routing Zone via Apstra*

Once the routing zones are created some additional resources need to be assigned.

Step 6: Select the desired resource pool for the loopback IPs that the switch will use to terminate the VXLAN tunnels. Resource pools were covered in Chapter 2.

Step 7: Click the Save icon.


Query: All

VRF Name	Type	VLAN ID	Route Target	VNI	DHCP Servers
default	L3 Fabric	N/A	N/A	N/A	DHCP Relay not configured
Production	EVPN	2	Not assigned	Not assigned	DHCP Relay not configured

Production: Leaf Loopback IPs (0/4)

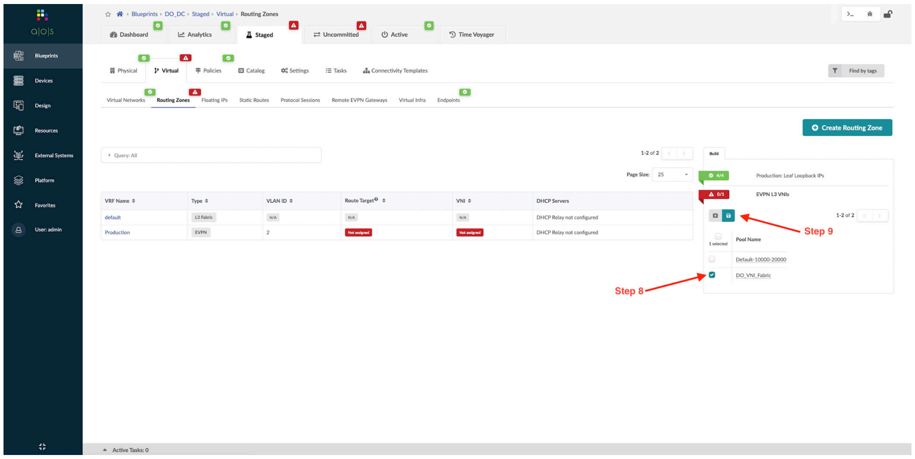
	Pool Name
	TESTNET-203.0.113.0/24
✓	DO_IP_Loopback
	DO_IP_External
	Private-10.0.0.0/8
	Private-192.168.0.0/16

EVPN L3 VNIs (0/1)

Figure 5.4 *Adding Loopback IPs to a New Routing Zone via Apstra*

Step 8: For every VRF, Apstra instantiates an L3 VN for inter-VXLAN domain routing. This is an automatic process, so you will need to select the desired VNI pool from which to take the VNI ID, and this will automatically assign an VNI to the virtual network. Any subsequent VRF creation will take a VNI from the initially chosen pool.

Step 9: Click the Save icon



1-2 of 2 Page Size: 25

VRF Name	Type	VLAN ID	Route Target	VNI	DHCP Servers
default	L3 Fabric	N/A	N/A	N/A	DHCP Relay not configured
Production	EVPN	2	Not assigned	Not assigned	DHCP Relay not configured

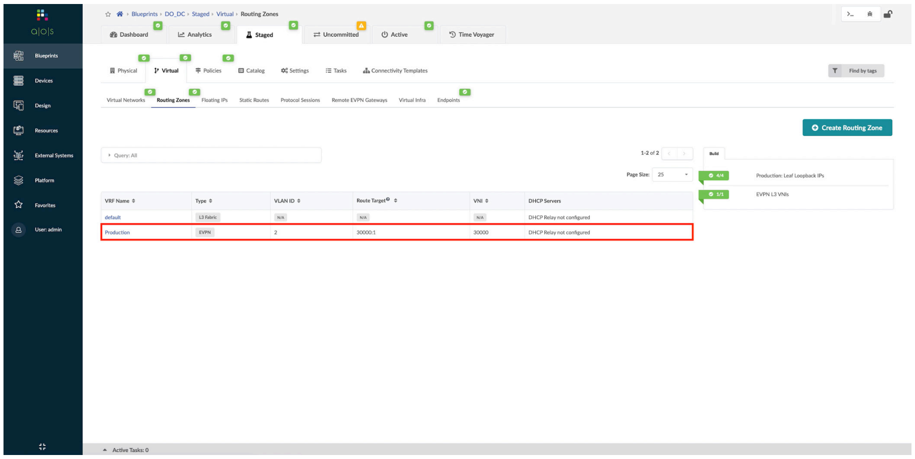
4/4 Production: Leaf Loopback IPs
0/1 EVPN L3 VNIs

1-2 of 2, 1 selected

Pool Name
Default-10000-20000
DO_VNI_Fabric

Figure 5.5 *Adding a VNI to a New Routing Zone via Apstra*

Once complete the routing zone will have assigned all the required information including VNIs and route targets, as shown in Figure 5.6.



1-2 of 2 Page Size: 25

VRF Name	Type	VLAN ID	Route Target	VNI	DHCP Servers
default	L3 Fabric	N/A	N/A	N/A	DHCP Relay not configured
Production	EVPN	2	30000:1	30000	DHCP Relay not configured

4/4 Production: Leaf Loopback IPs
1/1 EVPN L3 VNIs

Figure 5.6 *Example of a Completed Routing Zone in Apstra*

Virtual Networks

Once a routing zone has been created in Apstra the next step is to define the virtual networks that sit within the routing zone (tenant). These virtual networks form the overlay and allow endpoints to communicate in a L2 fashion, and in the case of VXLAN, over the top of an L3 DC fabric.

Apstra offers two types of virtual networks, VLANs and VXLANs. This book focuses on a VXLAN overlay. You might note, however, that a VLAN virtual network is only applicable to a single switch that performs local switching. Refer to the Apstra documentation for more information on this topic (https://www.juniper.net/documentation/product/us/en/apstra).

Virtual Extensible LAN protocol (VXLAN) technology allows networks to support more VLANs. The VXLAN protocol overcomes the VLAN limitation by using a longer logical network identifier. This theoretically allows for many more VLANs and, therefore, more logical network isolation for large networks, such as those that include many virtual machines. There are other advantages of using VXLAN such as an L2 environment that can span an entire DC fabric and even be stretched across multiple DC fabrics, plus use all available paths unlike in the past when spanning tree protocol (STP) was required to prevent loops.

When creating a new virtual network in Apstra it's important to consider if the overlay needs to provide L3 services, such as a gateway and DHCP relay. When IP services are enabled they allow virtual networks to communicate via L3 with other virtual networks in the same routing zone. This is called *intra-tenant routing*, allowing the VXLAN networks in the same VRF to communicate with each other.

In Table 5.7 the two virtual networks that need to be created are Prod-DB and Prod-App. They both reside within the production routing zone. Table 5.1 shows the two virtual networks that must be created for this book's data center fabric scenario.

Table 5.1 Virtual Networks to be Created Based on this Book's Data Center Fabric Scenario

Name	Routing Zone	Type	VNI	Subnet
Prod-DB	Production	VXLAN	Auto-Assign	Auto-Assign
Prod-App	Production	VXLAN	Auto-Assign	Auto-Assign

Creating a Virtual Network

Step 1: Within the desired blueprint got to Staged > Virtual. See Figure 5.7.

Step 2: Select Virtual Networks.

Step 3: Click Create Virtual Networks.

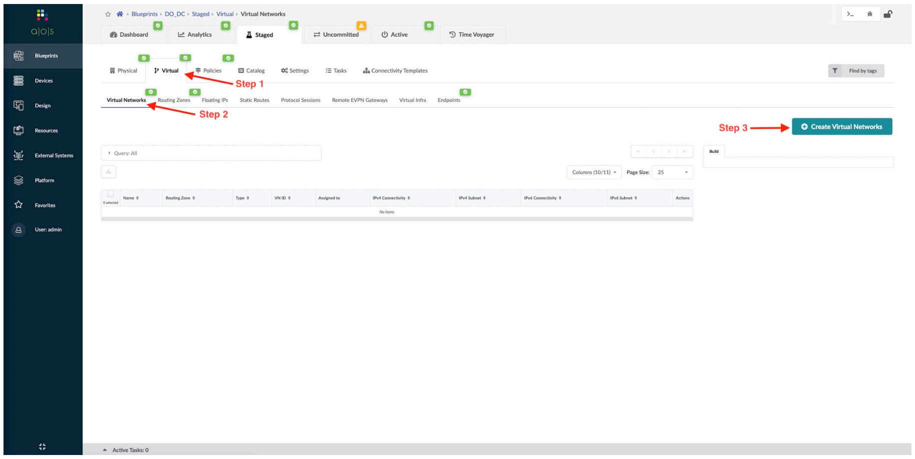


Figure 5.7 Adding a Virtual Network via Apstra

Step 4: Select the virtual network type. See Figure 5.8.

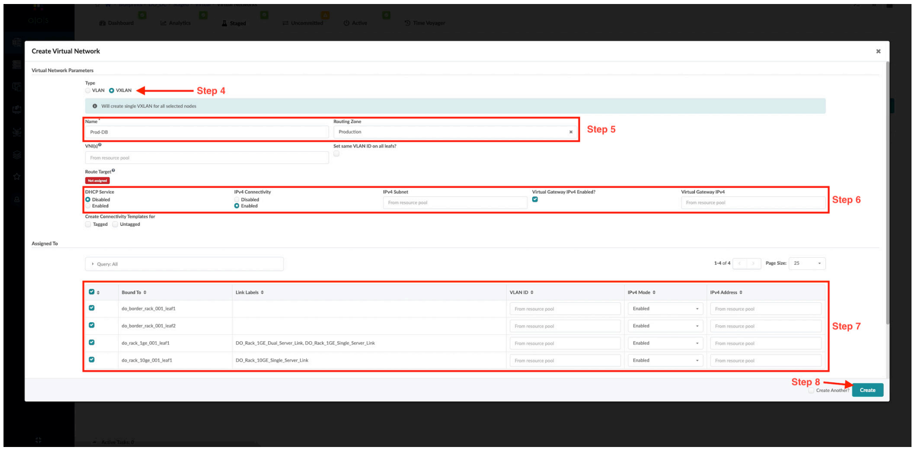
Step 5: Create a name for the VN and assign it to a routing zone. The VNI can be assigned, but this is optional.

Step 6: Assign the Layer 3 information required for the VN:

- IPv4 Connectivity - Enables IRB IPv4 routing features for this VN.
- IPv4 Subnet - IP address range used on this VN. This can be explicitly set with an IPv4 subnet, or this can auto-assigned using the IPv4 resource pools.
- Virtual Gateway - Enable and specify the Virtual Gateway IPv4 address for this VN.
- DHCP Service – If set to *Enabled*, and the blueprint template has DHCP enabled, a DHCP relay forwarder is configured on the IRB.

Step 7: Assign the virtual network to the desired racks within the DC fabric, in this example shown in Figure 5.8 the VNI is assigned to all racks.

Step 8: Click Create.



Create Virtual Network

Virtual Network Parameters

Type: VLAN [✓ VXLAN]

Will create single VXLAN for all selected nodes

Name: Prod-DB

Routing Zone: Production

VNI(s): From resource pool

Set same VLAN ID on all leafs? ☐

Route Target: Not assigned

DHCP Service: [✓ Disabled] Enabled

IPv4 Connectivity: Disabled [✓ Enabled]

IPv4 Subnet: From resource pool

Virtual Gateway IPv4 Enabled? ✓

Virtual Gateway IPv4: From resource pool

Create Connectivity Templates for: ☐ Tagged ☐ Untagged

Assigned To

Query: All — 1-4 of 4 — Page Size: 25

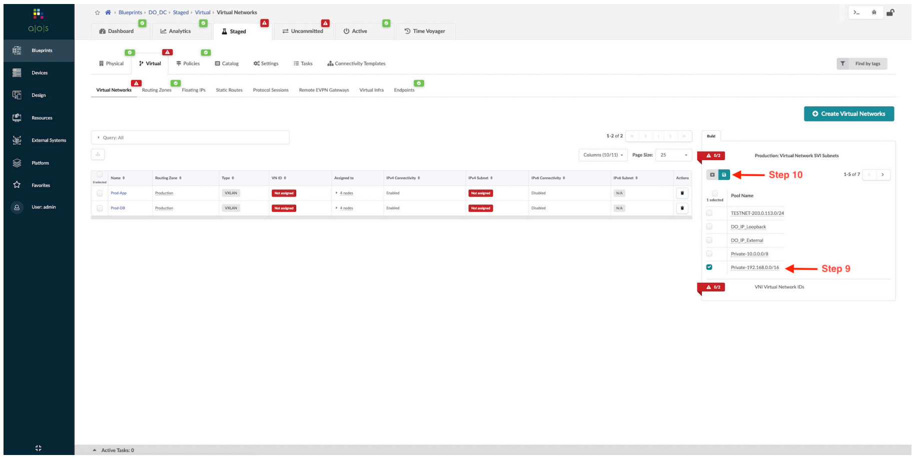
✓	Bound To	Link Labels	VLAN ID	IPv4 Mode	IPv4 Address
✓	do_border_rack_001_leaf1		From resource pool	Enabled	From resource pool
✓	do_border_rack_001_leaf2		From resource pool	Enabled	From resource pool
✓	do_rack_1ge_001_leaf1	DO_Rack_1GE_Dual_Server_Link, DO_Rack_1GE_Single_Server_Link	From resource pool	Enabled	From resource pool
✓	do_rack_10ge_001_leaf1	DO_Rack_10GE_Single_Server_Link	From resource pool	Enabled	From resource pool

☐ Create Another?

Figure 5.8 *Adding a Virtual Network via Apstra*

Step 9: IPv4 information was not explicitly defined in the previous steps but Apstra can use the IPv4 resource pools created in Chapter 2. Select the desired IPv4 pool, and Apstra will assign the next free IPv4 subnet in the pool.

Step 10: Click the Save icon.



Query: All — 1-2 of 2 — Columns (10/11) — Page Size: 25

0 selected	Name	Routing Zone	Type	VN ID	Assigned to	IPv4 Connectivity	IPv4 Subnet	IPv6 Connectivity	IPv6 Subnet	Actions
☐	Prod-App	Production	VXLAN	Not assigned	4 nodes	Enabled	Not assigned	Disabled	N/A	🗑
☐	Prod-DB	Production	VXLAN	Not assigned	4 nodes	Enabled	Not assigned	Disabled	N/A	🗑

0/2 Production: Virtual Network SVI Subnets — 1-5 of 7

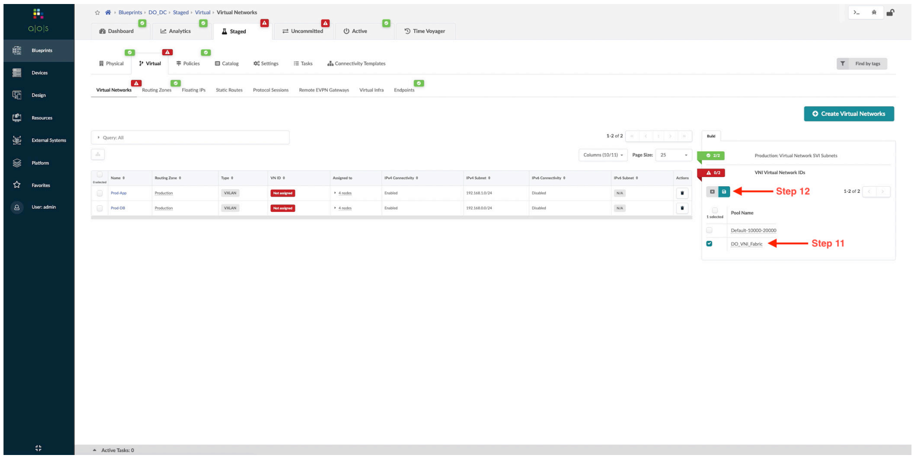
1 selected	Pool Name
☐	TESTNET-203.0.113.0/24
☐	DO_IP_Loopback
☐	DO_IP_External
☐	Private-10.0.0.0/8
✓	Private-192.168.0.0/16

0/2 VNI Virtual Network IDs

Figure 5.9 *Adding a L3 Gateway to Newly Created Virtual Networks Via Apstra*

Step 11: In the previous steps VNI information was not explicitly defined, but Apstra can use the VNI resource pools created in Chapter 2. Select the desired VNI pool, and Apstra will assign the next free VNI in the pool.

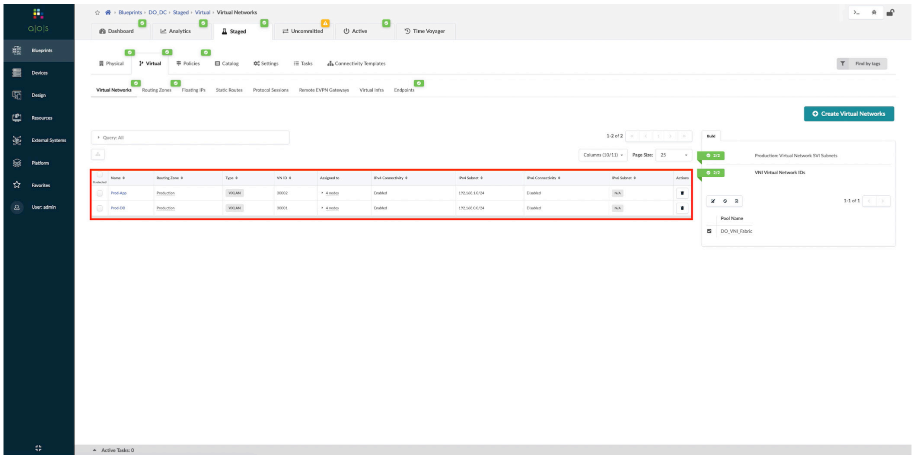
Step 12: Click the Save icon.



Name	Routing Zone	Type	VN ID	Assigned to	IPv4 Connectivity	IPv4 Subnet	IPv6 Connectivity	IPv6 Subnet	Actions
Prod-App	Production	VXLAN	Not assigned	4 nodes	Enabled	192.168.1.0/24	Disabled	N/A	
Prod-DB	Production	VXLAN	Not assigned	4 nodes	Enabled	192.168.0.0/24	Disabled	N/A	

Figure 5.10 *Adding a VNI to Newly Created Virtual Networks*

According to Table 5.1, this book's scenario needs two virtual networks to be created. Figure 5.11 shows the result of creating both virtual networks and having the IPv4 and VNI resources auto assigned.



Name	Routing Zone	Type	VN ID	Assigned to	IPv4 Connectivity	IPv4 Subnet	IPv6 Connectivity	IPv6 Subnet	Actions
Prod-App	Production	VXLAN	30002	4 nodes	Enabled	192.168.1.0/24	Disabled	N/A	
Prod-DB	Production	VXLAN	30001	4 nodes	Enabled	192.168.0.0/24	Disabled	N/A	

Figure 5.11 *Example of Newly Created Virtual Networks in Apstra*

Note that the virtual networks are created but they are not assigned to any ports on the devices as shown in Figure 5.12. So the next step is to assign the virtual networks to interfaces so they can be utilized.

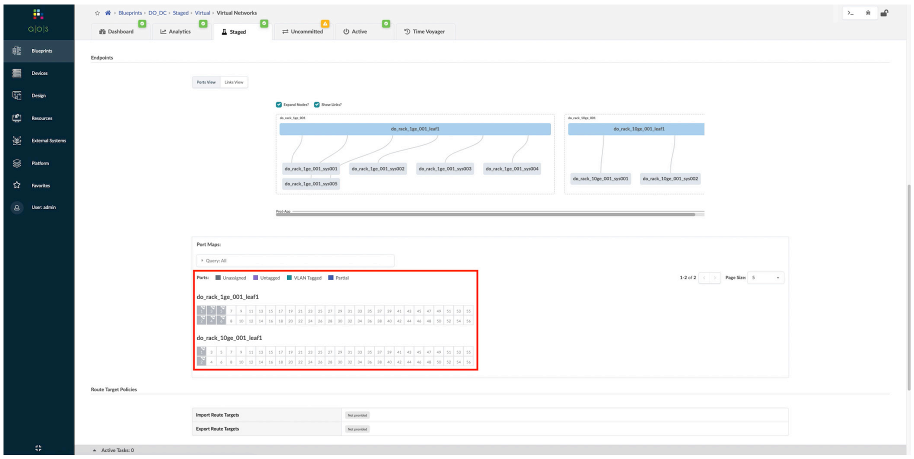


Figure 5.12 Example of a Newly Created Virtual Network with No Ports Assigned

Connectivity Templates

Once virtual networks are created, they need to be utilized by ports on the network. Mapping the physical ports on a switch to a virtual network (VXLAN or VLAN) is achieved in Apstra by using *connectivity templates*.

Connectivity templates allow administrators to easily create and apply configuration to switches that reach outside the fabric to generic devices such as servers, routers, and firewalls. Connectivity templates can be used for a wide range of tasks such as creating BGP peering between the fabric and external devices or creating static routes. However, connectivity templates are used in this chapter to map a virtual network (VXLAN) to a physical port on a switch. By creating these mappings, end hosts will be able to communicate both within the same virtual network and between virtual networks (if in the same routing zone or an external router is used).

To create this mapping of a virtual network to the underlying switch hardware it's first important to understand how connectivity templates are structured. Connectivity templates consist of one or more *primitives*. A primitive is a single low-level action on a networking device. An example is creating a subinterface on a port or creating a single static route. Figure 5.13 shows two different connectivity templates, one preforming a simple mapping to a virtual network, the other a BGP session with an external device.

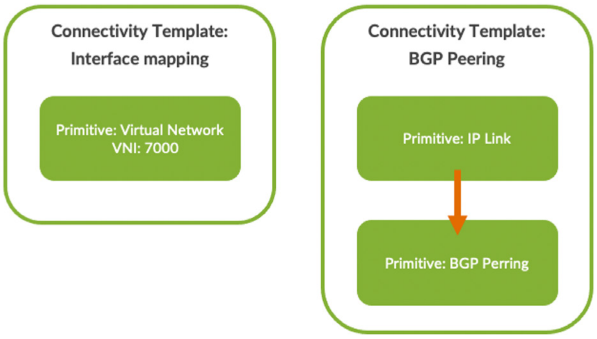


Figure 5.13 Two Different Connectivity Templates

The primitive virtual network shown in Figure 5.13 does a single job, when applied to an interface it will encapsulate any inbound traffic destined to another switch with a VNI of 7000.

Figure 5.14 shows how this is achieved:

Step 1: The connectivity template is applied to the interface of a switch.

Step 2: The host in this instance is sending untagged traffic to the switch.

Step 3: The switch encapsulates the traffic and when it leaves the switch it is encapsulated in a VXLAN packet with a VNI of 7000.

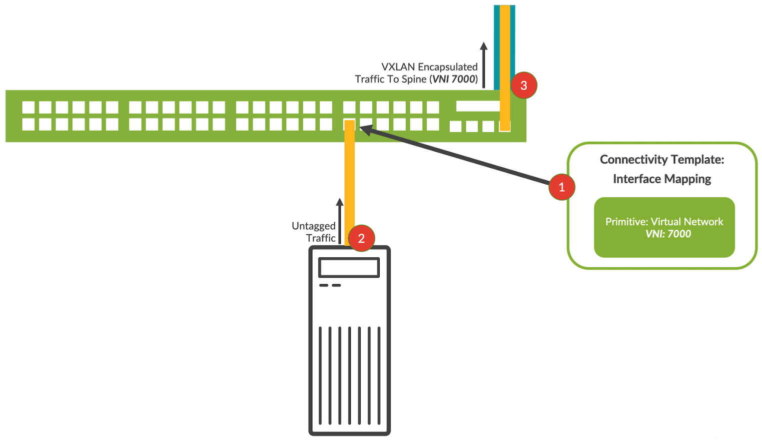


Figure 5.14 Example of Connectivity Templates Being Applied to an Interface

Based on this book's scenario the two separate connectivity templates are listed in Table 5.3.

Table 5.3 Connectivity Templates to be Created Based on This Book's Scenario

Connectivity Template Name	VXLAN	Traffic Type
Prod-DB Untagged Interface	Prod-DB	Untagged
Prod-App Untagged Interface	Prod-App	Untagged

Now let's show how to create the connectivity templates and apply them to different interfaces.

Creating Connectivity Templates

NOTE The method below assigns the ports per VNI but a batch update is also available. Please refer to the latest Apstra documentation to learn about connectivity template batch updating (https://www.juniper.net/documentation/product/us/en/apstra).

Step 1: Within the desired blueprint go to Staged > Connectivity Templates. See Figure 5.15.

Step 2: Click Add Template.

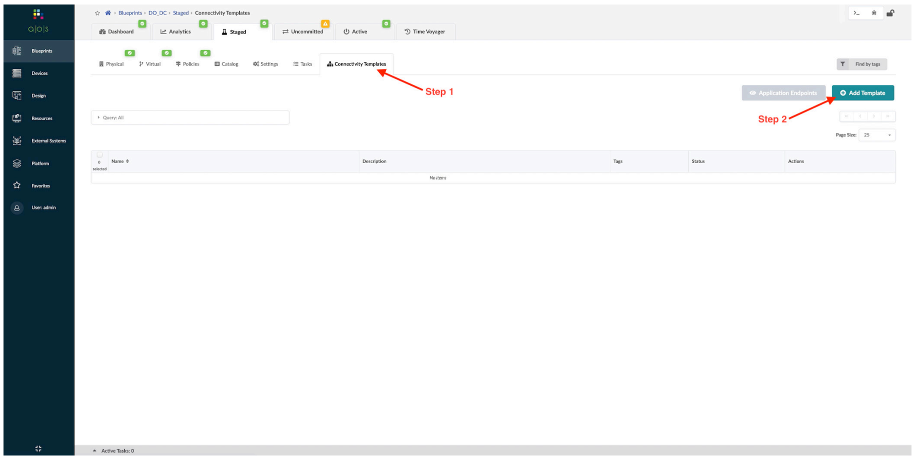
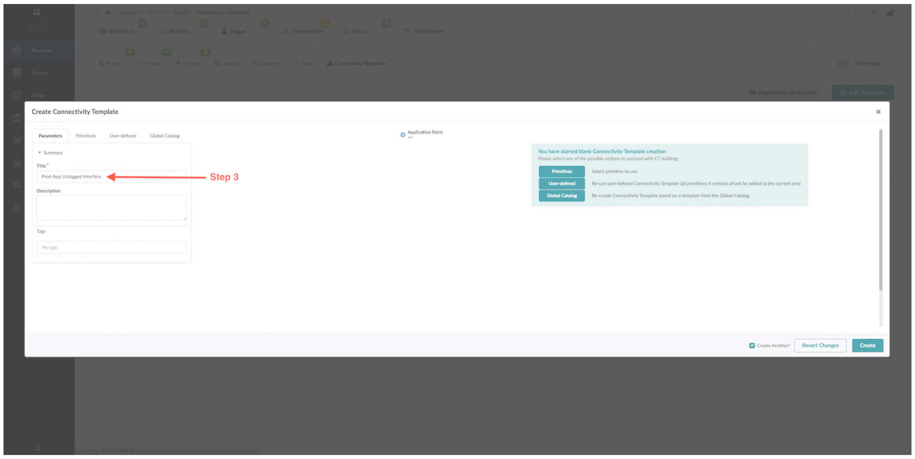


Figure 5.15 Creating a Connectivity Template via Apstra

Step 3: Assign a name to the connectivity template. See Figure 5.16.



Title: Prod-App Untagged Interface

Description:

Tags: No tags

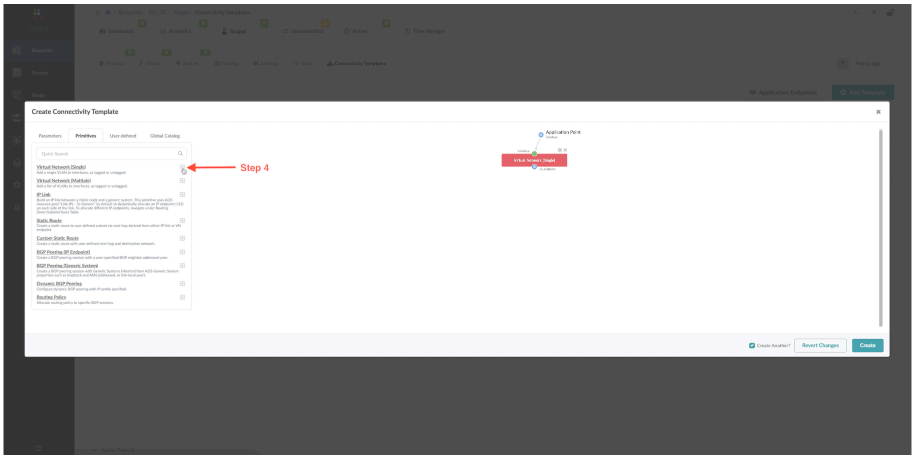
You have started blank Connectivity Template creation

Please select one of the possible options to proceed with CT building:

Primitives	Select primitive to use
User-defined	Re-use user-defined Connectivity Template (all primitives it consists of will be added to the current one)
Global Catalog	Re-create Connectivity Template based on a template from the Global Catalog

Figure 5.16 *Creating a Connectivity Template via Apstra*

Step 4: Under the Primitives tab select Virtual Network (Single). See Figure 5.17.



Virtual Network (Single)
Add a single VLAN to interfaces, as tagged or untagged.

Virtual Network (Multiple)
Add a list of VLANs to interfaces, as tagged or untagged.

IP Link
Build an IP link between a fabric node and a generic system. This primitive uses AOS resource pool "Link IPs - To Generic" by default to dynamically allocate an IP endpoint (/31) on each side of the link. To allocate different IP endpoints, navigate under Routing Zone>Subinterfaces Table.

Static Route
Create a static route to user defined subnet via next hop derived from either IP link or VN endpoint.

Custom Static Route
Create a static route with user defined next hop and destination network.

BGP Peering (IP Endpoint)
Create a BGP peering session with a user-specified BGP neighbor addressed peer.

BGP Peering (Generic System)
Create a BGP peering session with Generic Systems inherited from AOS Generic System properties such as loopback and ASN (addressed, or link-local peer).

Dynamic BGP Peering
Configure dynamic BGP peering with IP prefix specified.

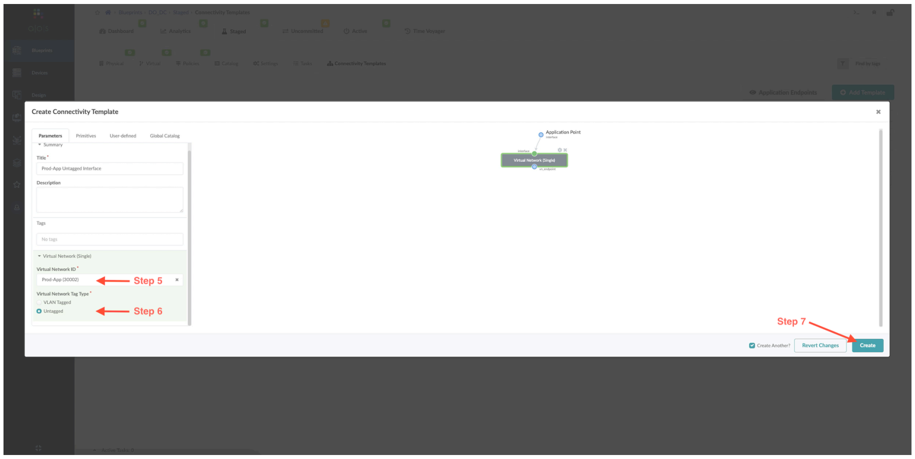
Routing Policy
Allocate routing policy to specific BGP sessions.

Figure 5.17 *Creating a Connectivity Template via Apstra*

Step 5: Under the Parameters tab select the desired VNI. See Figure 5.18.

Step 6: Under the Parameters tab select if the traffic from the host will be VLAN tagged or untagged

Step 7: Click Create.



Summary

Title: Prod-App Untagged Interface

Description:

Tags: No tags

Virtual Network (Single)

Virtual Network ID: Prod-App (30002)

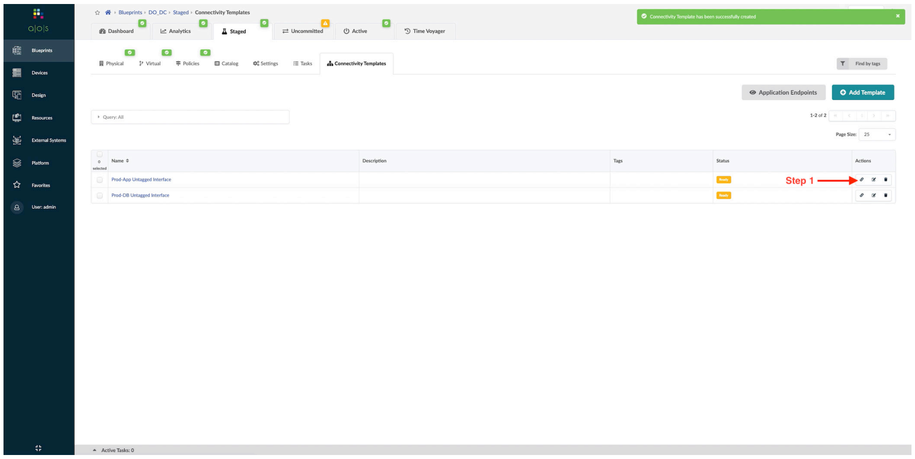
Virtual Network Tag Type: VLAN Tagged [✓ Untagged]

Figure 5.18 *Creating a Connectivity Template via Apstra*

Once created a connectivity template needs to be applied, which is shown next.

Applying Connectivity Templates

Step 1: Under Staged > Connectivity Templates click the chain icon for the desired VNI. See Figure 5.19.



Connectivity Template has been successfully created

Query: All

1-2 of 2

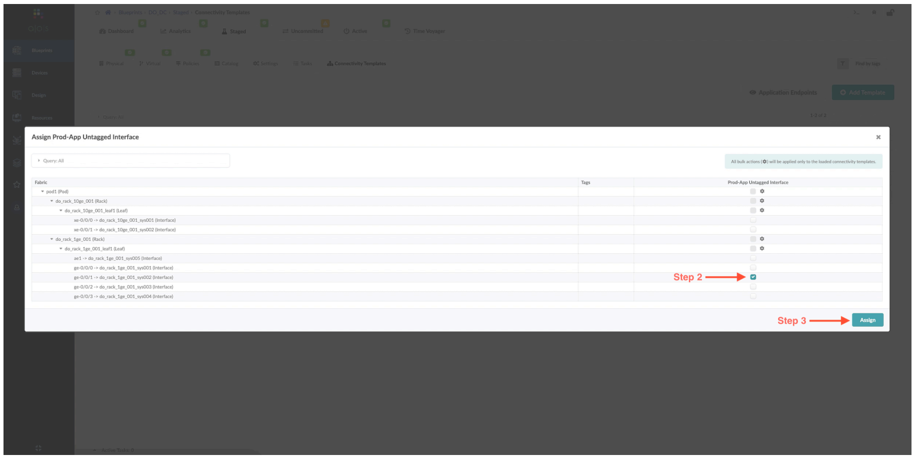
Page Size: 25

0 selected	Name	Description	Tags	Status	Actions
	Prod-App Untagged Interface			Ready	
	Prod-DB Untagged Interface			Ready	

Figure 5.19 *Assigning a Connectivity Template via Apstra*

Step 2: Select the interface that the VNI should be assigned to. See Figure 5.20.

Step 3: Click Create.


Assign Prod-App Untagged Interface

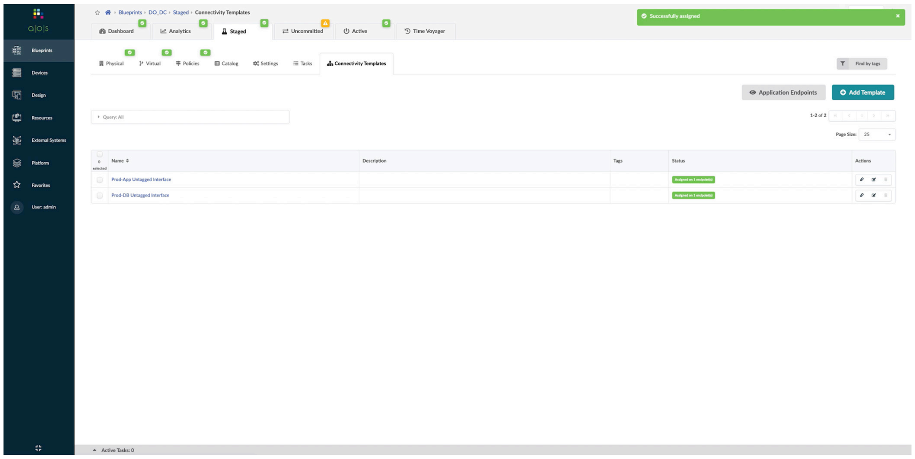
Query: All

All bulk actions (⚙) will be applied only to the loaded connectivity templates.

Fabric	Tags	Prod-App Untagged Interface
pod1 (Pod)		☐ ⚙
do_rack_10ge_001 (Rack)		☐ ⚙
do_rack_10ge_001_leaf1 (Leaf)		☐ ⚙
xe-0/0/0 -> do_rack_10ge_001_sys001 (Interface)		☐
xe-0/0/1 -> do_rack_10ge_001_sys002 (Interface)		☐
do_rack_1ge_001 (Rack)		☐ ⚙
do_rack_1ge_001_leaf1 (Leaf)		☐ ⚙
ae1 -> do_rack_1ge_001_sys005 (Interface)		☐
ge-0/0/0 -> do_rack_1ge_001_sys001 (Interface)		☐
ge-0/0/1 -> do_rack_1ge_001_sys002 (Interface)		✓
ge-0/0/2 -> do_rack_1ge_001_sys003 (Interface)		☐
ge-0/0/3 -> do_rack_1ge_001_sys004 (Interface)		☐

Figure 5.20 *Assigning a Connectivity Template via Apstra*

Once created and assigned the number of endpoints show up as in Figure 5.21. Once this has been completed the configuration must be committed before taking effect.


Blueprints › DO_DC › Staged › Connectivity Templates

Successfully assigned

Query: All

1-2 of 2

Page Size: 25

Name	Description	Tags	Status	Actions
Prod-App Untagged Interface			Assigned on 1 endpoint(s)	
Prod-DB Untagged Interface			Assigned on 1 endpoint(s)	

Figure 5.21 *Example of a Connectivity Template that Has Been Applied to Interfaces*

Testing Connectivity

Two endpoints have been assigned two separate VXLAN based virtual networks with L3 interfaces that are present in the same routing zone. This means that a host from one VN can communicate with the host in the other VN as shown in Figure 5.22. It's important to remember that this can happen because both virtual networks sit within the same routing zone (VRF). If these virtual networks were placed in different routing zones an external router would be needed to route between the two.

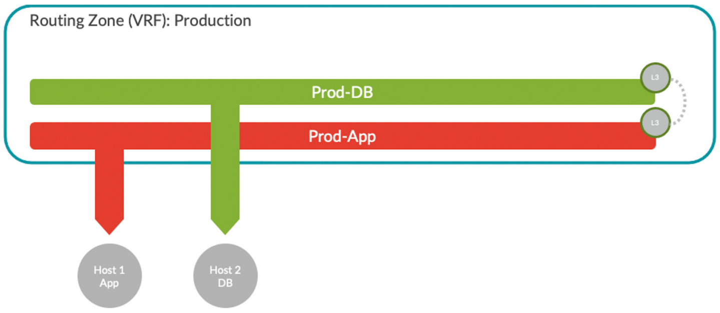
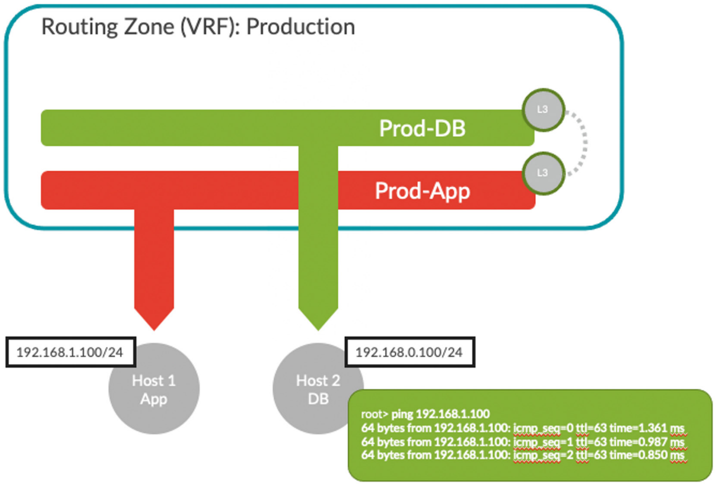


Figure 5.22 *Logical Network Diagram of the Tenant Created in this Chapter*

Perform a simple ping between the two nodes to verify connectivity has been achieved. As per this book's lab, the example in Figure 5.23 shows a successful ping between a host in the DB virtual network to a host in the App virtual network.



```
root> ping 192.168.1.100
64 bytes from 192.168.1.100: icmp_seq=0 ttl=63 time=1.361 ms
64 bytes from 192.168.1.100: icmp_seq=1 ttl=63 time=0.987 ms
64 bytes from 192.168.1.100: icmp_seq=2 ttl=63 time=0.850 ms
```

Figure 5.23 *Example of Communication Between Two Hosts in Different Virtual Networks*

Summary

This chapter demonstrated how a new tenant (routing zone) can be added on top of the previously created Layer 3 Clos fabric, and how virtual networks can be added with demonstrated connectivity between them.

The next chapter, Chapter 6, is the final deployment piece, allowing separate tenants (routing zones) to communicate between each other using an external router.

Chapter 6

Inter-tenant Routing Using an External System

Chapter 5 created a brand new Production tenant (also known as a routing zone in Apstra) on the DC fabric. Two VNs were added with associated hosts and routing was established between these hosts.

In many DC environments multiple tenants will exist and routing between the two may be required, but strict routing and security policies are often needed. In addition to the security features already built into Apstra, perimeter security can be enforced by a dedicated firewall, or cluster of firewalls, where the highest service availability is required This firewall is sometimes connected to the DC access fabric(s) and provides routing between hosts in different routing environments on this access layer. Such design makes it easy to impose security policies at a central location in the DC and the firewalls can optionally provide a secure gateway / path out of the DC onto remote locations.

This chapter discusses how inter-tenant routing is achieved in Apstra with the use of an external Juniper SRX firewall connected to the data center fabrics overlay. More than previous chapters, this chapter leans on the book's scenario to provide context. Your deployment may differ.

Therefore, in this chapter a second fabric tenant called *Backup* will be created, with its own virtual networks and hosts, and a Juniper SRX1500 firewall will be added to the DC design as an additional device to perform routing between the Production and Backup routing zones. The final steps will be to prepare the Juniper SRX1500 firewalls configuration and create the BGP peering in Apstra.

NOTE Security configurations on the Juniper SRX are out of scope for this *Day One* book, but the Juniper SRX documentation can be used for further reading on this topic: https://www.juniper.net/documentation.

Adding a New Logical Tenant On the DC Fabric

To perform inter-tenant routing at least two tenants must exist. A new logical tenant will be added along with two virtual networks for its hosts. In order to complete the final phase of this book scenario, please refer to the Chapter 4 for the steps required to create a new tenant and make sure to assign it a new name, IP subnets, VLAN IDs, etc. This workflow will be omitted here for brevity. Table 6.1 is an overview of the new tenant details.

Table 6.1 Apstra Routing Zones and Virtual Networks Created

Routing Zone	VN Name	VN Type	VN Identifier	Subnet	Created
Production	Prod-DB	VXLAN	Auto-Assign (30001)	Auto-Assign (192.168.0.0/24)	Previously
Production	Prod-App	VXLAN	Auto-Assign (30002)	Auto-Assign (192.168.1.0/24)	Previously
Backup	Backup-DB	VXLAN	Auto-Assign (30004)	Auto-Assign (192.168.2.0/24)	NEW
Backup	Backup-App	VXLAN	Auto-Assign (30005)	Auto-Assign (192.168.3.0/24)	NEW

A graphical representation of the multi-tenancy concept is illustrated in Figure 6.1.

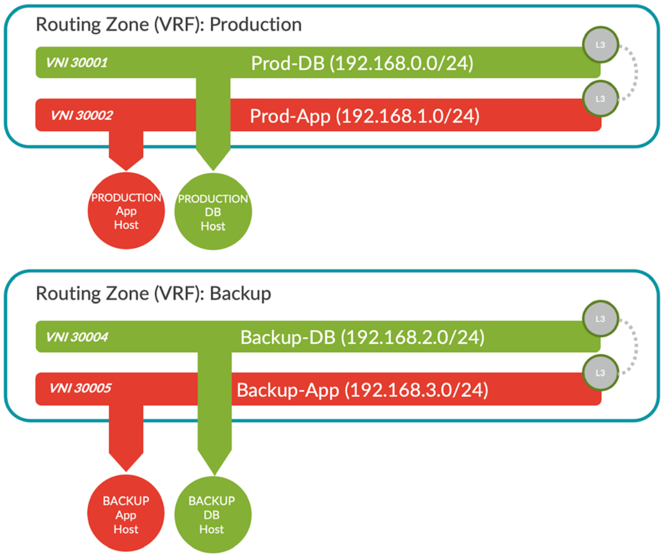

Routing Zone (VRF): Production
VNI 30001
Prod-DB (192.168.0.0/24)
L3
VNI 30002
Prod-App (192.168.1.0/24)
L3
PRODUCTION App Host
PRODUCTION DB Host
Routing Zone (VRF): Backup
VNI 30004
Backup-DB (192.168.2.0/24)
L3
VNI 30005
Backup-App (192.168.3.0/24)
L3
BACKUP App Host
BACKUP DB Host

Figure 6.1 Multi-Tenancy Addressing Details Used in This Book's Scenario

NOTE Remember to commit the configuration of the second tenant by going to the top Uncommitted tab and clicking the Commit button. Optionally, put in a revision description for this configuration change.

Each tenant has two hosts connected to the server leaf, this can be verified by preforming a ping to their L3 gateway, which in this Juniper EVPN-VXLAN fabric is the server leaf itself (see Figure 6.2):

- Host 192.168.1.100/24 is connected to Server Leaf 1 on interface ge-0/0/2, in the Prod-App virtual network.
- Host 192.168.2.100/24 is connected to Server Leaf 1 on interface ge-0/0/4, in the Backup-DB virtual network.

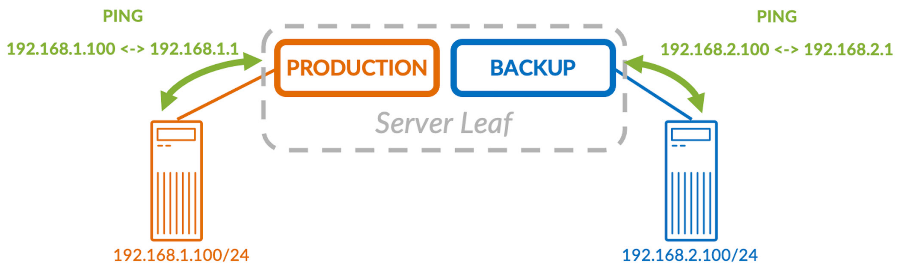


Figure 6.2 Testing Connectivity Between User Hosts and Their L3 Gateways

```
# ping from Prod-App host to gateway succeeds

user@PROD-APP-HOST> ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: icmp_seq=0 ttl=253 time=0.150 ms
64 bytes from 192.168.1.1: icmp_seq=1 ttl=253 time=0.201 ms
^C --- 192.168.1.1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss


# ping from Backup-DB host to gateway succeeds

user@BKP-DB-HOST> ping 192.168.2.1
PING 192.168.2.1 (192.168.2.1): 56 data bytes
64 bytes from 192.168.2.1: icmp_seq=0 ttl=253 time=0.189 ms
64 bytes from 192.168.2.1: icmp_seq=1 ttl=253 time=0.130 ms
^C --- 192.168.2.1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
```

The hosts in each tenant now have Layer 2 connectivity to hosts in the same virtual network and Layer 3 connectivity to hosts in the same tenant but different virtual networks. The remainder of this chapter focuses on the external router, in this case a firewall, and creating a connection between the tenants via BGP.

Adding an External Generic System

The SRX1500 firewall is connected to the DC fabric as shown in Figure 6.3, using channel 1 of both 40GbE channelized interfaces on the border leafs.

NOTE Channelized interfaces enable users to configure a number of individual channels that subdivide the bandwidth of a larger interface. For example, a 40GE interface can be subdivided into 4 x 10GE interfaces.

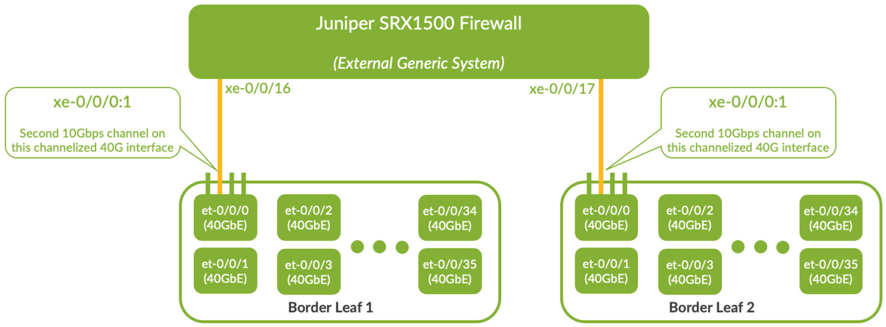


Figure 6.3 Physical Links Between the Firewall and Fabric Border Leafs

After the firewall is powered on and physically connected to the DC fabric, the first step is to create a logical entity called a *Generic System* in Apstra to logically represent the firewall in the data center design.

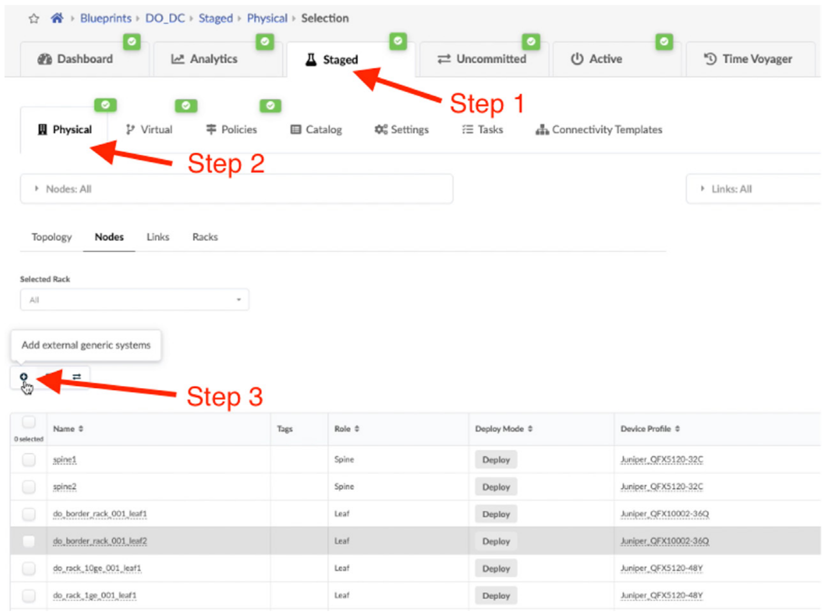
Generic Systems can be any device (router, firewall, server, etc.) that is connected to the fabric but not managed directly with Apstra. Generic Systems are added to a blueprint and then external links are added.

A new external Generic System is added to the existing blueprint to connect the firewall to the border leafs. Here are the steps to add a new external Generic System and links between the data center fabric and the SRX firewall.

Step 1: Click the Staged tab in the blueprint view. See Figure 6.4.

Step 2: Click the Physical tab and select the Nodes view.

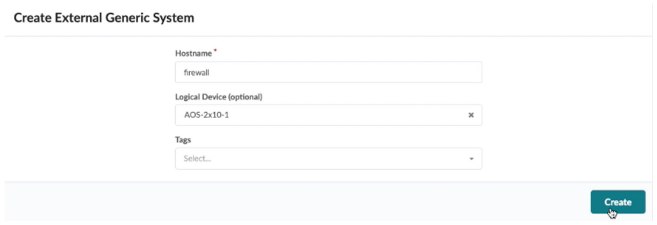
Step 3: Click the 'Add external generic systems' button on the left (looks like a + sign).



0 selected	Name	Tags	Role	Deploy Mode	Device Profile
	spine1		Spine	Deploy	Juniper_QFX5120-32C
	spine2		Spine	Deploy	Juniper_QFX5120-32C
	do_border_rack_001_leaf1		Leaf	Deploy	Juniper_QFX10002-36Q
	do_border_rack_001_leaf2		Leaf	Deploy	Juniper_QFX10002-36Q
	do_rack_10ge_001_leaf1		Leaf	Deploy	Juniper_QFX5120-48Y
	do_rack_1ge_001_leaf1		Leaf	Deploy	Juniper_QFX5120-48Y

Figure 6.4 Add External Generic System

In the new pop-up window (see Figure 6.5) the firewall is assigned a hostname and an optional Logical Device type of AOS-2x10-1, meaning this Generic System is expected to have 2 x 10GbE interfaces connected to the fabric. Click Create.


Create External Generic System

Hostname *: firewall

Logical Device (optional): AOS-2x10-1

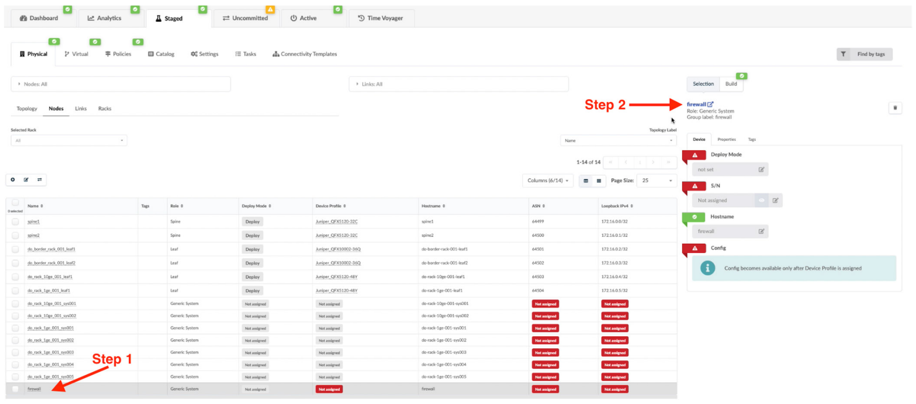
Tags: Select...

Figure 6.5 Create External Generic System

Once created, the Generic System will appear at the bottom of the Nodes list, as shown in Figure 6.6. Select the firewall from the Nodes list and open the device specific configuration by selecting its hostname on the right.

Step 1: Select the Firewall from the device list.

Step 2: Select the firewall name to focus on its device configuration.



firewall
Role: Generic System
Group label: firewall

Config becomes available only after Device Profile is assigned

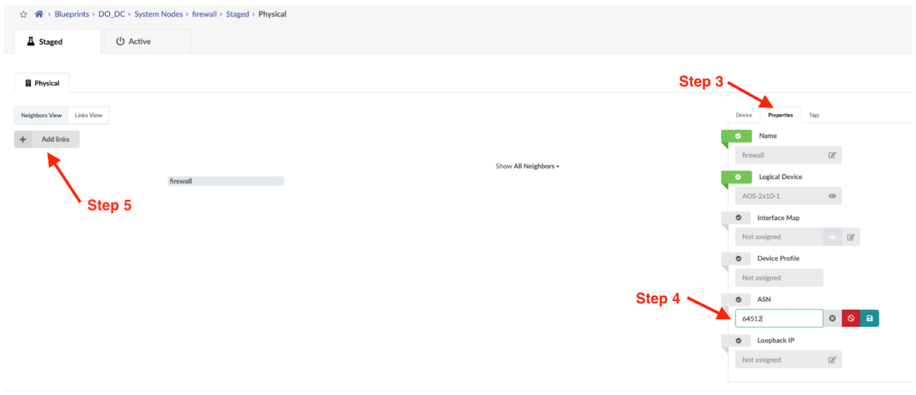
Name	Tags	Role	Deploy Mode	Device Profile	Hostname	ASN	Loopback IPv4
spine1		Spine	Deploy	Juniper_QFX5120-32C	spine1	64499	172.16.0.0/32
spine2		Spine	Deploy	Juniper_QFX5120-32C	spine2	64500	172.16.0.1/32
do_border_rack_001_leaf1		Leaf	Deploy	Juniper_QFX10002-36Q	do-border-rack-001-leaf1	64501	172.16.0.2/32
do_border_rack_001_leaf2		Leaf	Deploy	Juniper_QFX10002-36Q	do-border-rack-001-leaf2	64502	172.16.0.3/32
do_rack_10ge_001_leaf1		Leaf	Deploy	Juniper_QFX5120-48Y	do-rack-10ge-001-leaf1	64503	172.16.0.4/32
do_rack_1ge_001_leaf1		Leaf	Deploy	Juniper_QFX5120-48Y	do-rack-1ge-001-leaf1	64504	172.16.0.5/32
do_rack_10ge_001_sys001		Generic System	Not assigned	Not assigned	do-rack-10ge-001-sys001	Not assigned	Not assigned
do_rack_10ge_001_sys002		Generic System	Not assigned	Not assigned	do-rack-10ge-001-sys002	Not assigned	Not assigned
do_rack_1ge_001_sys001		Generic System	Not assigned	Not assigned	do-rack-1ge-001-sys001	Not assigned	Not assigned
do_rack_1ge_001_sys002		Generic System	Not assigned	Not assigned	do-rack-1ge-001-sys002	Not assigned	Not assigned
do_rack_1ge_001_sys003		Generic System	Not assigned	Not assigned	do-rack-1ge-001-sys003	Not assigned	Not assigned
do_rack_1ge_001_sys004		Generic System	Not assigned	Not assigned	do-rack-1ge-001-sys004	Not assigned	Not assigned
do_rack_1ge_001_sys005		Generic System	Not assigned	Not assigned	do-rack-1ge-001-sys005	Not assigned	Not assigned
firewall		Generic System	Not assigned	Not assigned	firewall	Not assigned	Not assigned

Figure 6.6 *Edit a New Generic Device*

Step 3: In the next window (Figure 6.7) select the Properties tab.

Step 4: Edit the ASN field to assign an AS Number to this system. Here the private ASN 64512 is used. Click the save button on the right of the ASN field.

Step 5: Assign a new link to this firewall by clicking the Add links button on the left.



Name
firewall
Logical Device
AOS-2x10-1
Interface Map
Not assigned
Device Profile
Not assigned
ASN
64512
Loopback IP
Not assigned

Figure 6.7 *Editing a New Generic Device*

The Generic System now has an assigned AS number and Apstra needs to know the links it uses to connect to the fabric devices under management. In the next screen (Figure 6.8) you need to select these links from the available physical interfaces and group them under a logical link.

If using channelized interfaces, such as in this book's example, it is not enough to select the physical link. Since Apstra does not manage external devices, it does not have any visibility to external physical links, so the correct channel needs to be selected (xe-0/0/0:1).

In this book the first 40GbE physical interface (et-0/0/0) on both border leafs is channelized into 4 x 10GbE channels (xe-0/0/0:1-3), as shown in previous chapters, and the firewall is connected to channel 1 (xe-0/0/0:1) on both border leafs.

Step 1. Select the first Fabric device connected to the firewall. In this case it is Border Leaf #1 with hostname do_border_rack_001_leaf1.

Step 2. Select the interface which connects to this external Generic System on the Device Profile.

Step 3. Select the physical interface from the available options. In this case it is xe-0/0/0:1 corresponding to channel 1

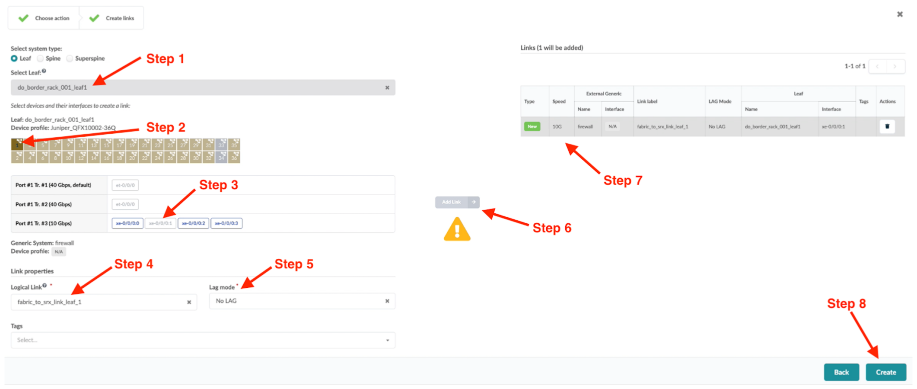
Step 4. Add in a custom unique description for this Logical Link. This will be used to uniquely identify the link in the blueprint.

Step 5. Choose 'No LAG' under LAG mode since this is a single point-to-point link.

Step 6. Click the 'Add link' button.

Step 7. The new link should be visible in the new table on the right.

Step 8. Click Create to go ahead and add this new Logical Link to the blueprint.



Select system type: [✓ Leaf] Spine Superspine

Select Leaf: do_border_rack_001_leaf1

Select devices and their interfaces to create a link:

Leaf: do_border_rack_001_leaf1
Device profile: Juniper_QFX10002-36Q

Port #1 Tr. #1 (40 Gbps, default)	et-0/0/0
Port #1 Tr. #2 (40 Gbps)	et-0/0/0
Port #1 Tr. #3 (10 Gbps)	xe-0/0/0:0 xe-0/0/0:1 xe-0/0/0:2 xe-0/0/0:3

Generic System: firewall
Device profile: N/A

Link properties

Logical Link *: fabric_to_srx_link_leaf_1

Lag mode *: No LAG

Tags: Select...

Links (1 will be added)

1-1 of 1

Type	Speed	External Generic		Link label	LAG Mode	Leaf		Tags	Actions
		Name	Interface			Name	Interface		
New	10G	firewall	N/A	fabric_to_srx_link_leaf_1	No LAG	do_border_rack_001_leaf1	xe-0/0/0:1		

Figure 6.8 *Adding a New Link Between a Fabric Leaf and a Generic System*

NOTE Please note the second link from border leaf 2 to the firewall was created in the same way and remember to use a unique Logical Link description. In this example this second link is omitted for brevity.

Once the required links are added, they will show up in the Staged > Physical view as shown in Figure 6.9.

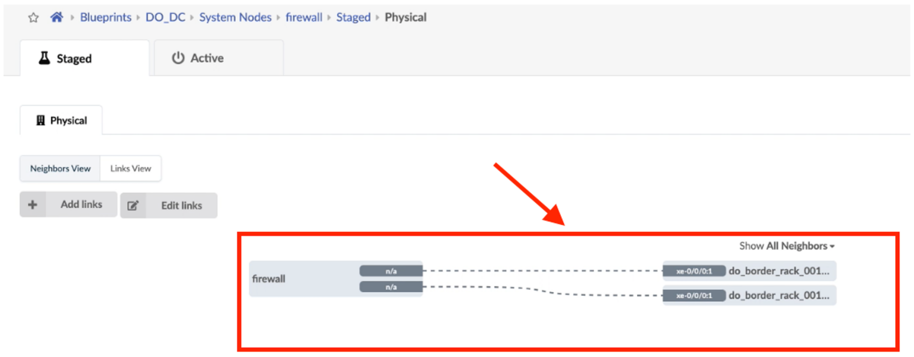


Figure 6.9 New Logical Links Between the Firewall and Border Leaf Devices

Okay, now we can add an external device (not controlled by Apstra) to the data center fabric and add links.

Configuring the Firewall

Before configuring anything on the firewall, let's see how it connects to the fabric:

- Interface xe-0/0/0:1 (the first 10GbE channel of the first 40GbE interface) on border leaf 1 will connect to xe-0/0/16 on the firewall. (See Figure 6.9).
- Interface xe-0/0/0:1 on border leaf 2 will connect to xe-0/0/17 on the firewall.
- Both links will be acting as L2 trunks with VLAN 2 allocated to the Production tenant and VLAN 5 allocated to the Backup tenant.

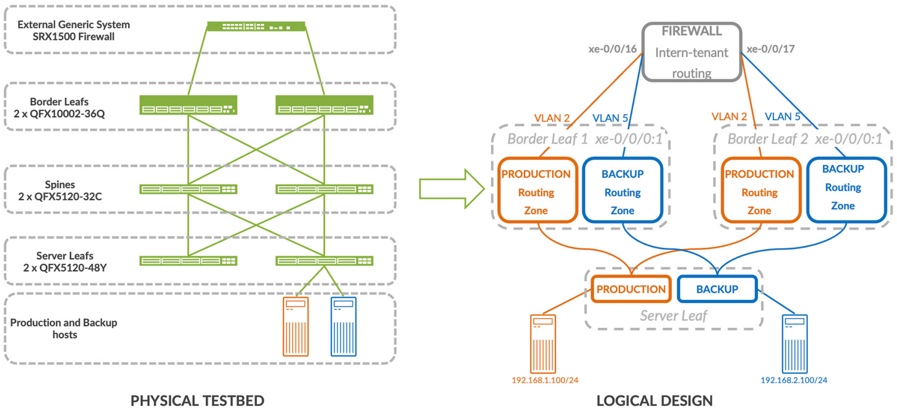


Figure 6.10 Logical Connectivity for Both Production and Backup Tenants

In order to keep complexity low, in this example the firewall will be managed exclusively by CLI. High-level management tools such as Juniper's Security Director, Junos Space, or Ansible are beyond the scope of this book.

Configuring the firewall is split into four steps:

- Configure the firewall's physical interfaces facing the border leafs.
- Configure basic routing policies to allow route advertisement to/from border leafs.
- Configure the BGP sessions for both Production and Backup tenants.
- Update the default 'deny all' security policy to permit all traffic through the firewall.

In the first step, interfaces xe-0/0/16 and xe-0/0/17 are configured assigning VLAN tags and IP addresses for both tenant endpoints:

```
# Configure first interface towards BL-1, with unit 2 for PRODUCTION
set interfaces xe-0/0/16 description "Connected to Border Leaf 1"
set interfaces xe-0/0/16 flexible-vlan-tagging
set interfaces xe-0/0/16 unit 2 description "To BL-1 PRODUCTION tenant"
set interfaces xe-0/0/16 unit 2 vlan-id 2
set interfaces xe-0/0/16 unit 2 family inet address 172.17.10.1/31

# Configure unit 5 for BACKUP
set interfaces xe-0/0/16 unit 5 description "To BL-1 BACKUP tenant"
set interfaces xe-0/0/16 unit 5 vlan-id 5
set interfaces xe-0/0/16 unit 5 family inet address 172.17.20.1/31

# Configure second interface towards BL-2, with unit 2 for PRODUCTION
set interfaces xe-0/0/17 description "Connected to Border Leaf 2"
set interfaces xe-0/0/17 flexible-vlan-tagging
set interfaces xe-0/0/17 unit 2 description "To BL-2 PRODUCTION tenant"
set interfaces xe-0/0/17 unit 2 vlan-id 2
set interfaces xe-0/0/17 unit 2 family inet address 172.17.10.3/31

# Configure unit 5 for BACKUP
set interfaces xe-0/0/17 unit 5 description "To BL-2 BACKUP tenant"
set interfaces xe-0/0/17 unit 5 vlan-id 5
set interfaces xe-0/0/17 unit 5 family inet address 172.17.20.3/31
```

In the second step, two routing policies will be configured on the firewall:

- The ACCEPT_ALL policy is meant to allow all routes and it will be applied on the INBOUND direction to all BGP sessions with the border leafs.
- The ONLY_ADVERTISE_DEFAULT policy is meant to advertise only the default route, which is artificially generated on the firewall, to all fabric tenant VRFs and it will be applied on the OUTBOUND direction to all BGP sessions with the border leafs. This method is an easy way of providing a routing path out of the fabric for each tenant and attracting all inter-tenant traffic to the firewall. See Figure 6.11.

```
# Configure an auto-generated default route and include it in a prefix list
set routing-options generate route 0.0.0.0/0
set policy-options prefix-list ONLY_DEFAULT 0.0.0.0/0

# Configure a policy to accept all routes, which will be applied inbound
set policy-options policy-statement ACCEPT_ALL term ACCEPT then accept

# Configure a policy to only advertise the default route, which will be applied outbound
set policy-options policy-statement ONLY_ADVERTISE_DEFAULT term DEFAULT from prefix-list ONLY_
DEFAULT
set policy-options policy-statement ONLY_ADVERTISE_DEFAULT term DEFAULT then next-hop self
set policy-options policy-statement ONLY_ADVERTISE_DEFAULT term DEFAULT then accept
set policy-options policy-statement ONLY_ADVERTISE_DEFAULT term ANYTHING_ELSE then reject
```

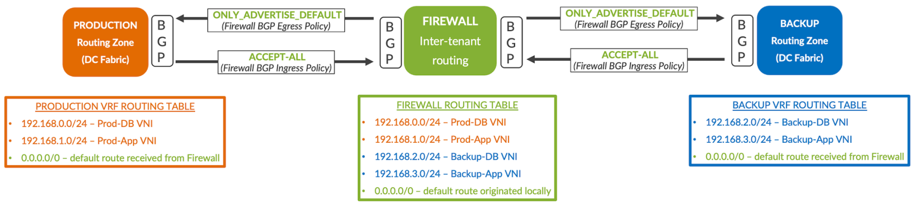

PRODUCTION Routing Zone (DC Fabric)
BGP
ONLY_ADVERTISE_DEFAULT (Firewall BGP Egress Policy)
ACCEPT-ALL (Firewall BGP Ingress Policy)
BGP
FIREWALL Inter-tenant routing
BGP
ONLY_ADVERTISE_DEFAULT (Firewall BGP Egress Policy)
ACCEPT-ALL (Firewall BGP Ingress Policy)
BGP
BACKUP Routing Zone (DC Fabric)
PRODUCTION VRF ROUTING TABLE
192.168.0.0/24 – Prod-DB VNI
192.168.1.0/24 – Prod-App VNI
0.0.0.0/0 – default route received from Firewall
FIREWALL ROUTING TABLE
192.168.0.0/24 – Prod-DB VNI
192.168.1.0/24 – Prod-App VNI
192.168.2.0/24 – Backup-DB VNI
192.168.3.0/24 – Backup-App VNI
0.0.0.0/0 – default route originated locally
BACKUP VRF ROUTING TABLE
192.168.2.0/24 – Backup-DB VNI
192.168.3.0/24 – Backup-App VNI
0.0.0.0/0 – default route received from Firewall

Figure 6.11 Firewall Ingress and Egress BGP Policies with Routing Table Outcome

In the third step, four BGP sessions will be configured with all fabric tenant VRFs:

```
# Configure a new external BGP neighbor group and set the local ASN
set protocols bgp group Fabric_Border_Leafs type external
set protocols bgp group Fabric_Border_Leafs family inet unicast
set protocols bgp group Fabric_Border_Leafs local-as 64512

# Configure a BGP session with the Production VRF on Border Leaf 1
set protocols bgp group Fabric_Border_Leafs neighbor 172.17.10.0 description “BL-1 Production”
set protocols bgp group Fabric_Border_Leafs neighbor 172.17.10.0 local-address 172.17.10.1
set protocols bgp group Fabric_Border_Leafs neighbor 172.17.10.0 import ACCEPT_ALL
set protocols bgp group Fabric_Border_Leafs neighbor 172.17.10.0 export ONLY_ADVERTISE_DEFAULT
set protocols bgp group Fabric_Border_Leafs neighbor 172.17.10.0 peer-as 64501

# Configure a BGP session with the Production VRF on Border Leaf 2
set protocols bgp group Fabric_Border_Leafs neighbor 172.17.10.2 description “BL-2 Production”
set protocols bgp group Fabric_Border_Leafs neighbor 172.17.10.2 local-address 172.17.10.3
set protocols bgp group Fabric_Border_Leafs neighbor 172.17.10.2 import ACCEPT_ALL
set protocols bgp group Fabric_Border_Leafs neighbor 172.17.10.2 export ONLY_ADVERTISE_DEFAULT
set protocols bgp group Fabric_Border_Leafs neighbor 172.17.10.2 peer-as 64502

# Configure a BGP session with the Backup VRF on Border Leaf 1
set protocols bgp group Fabric_Border_Leafs neighbor 172.17.20.0 description “BL-1 Backup”
set protocols bgp group Fabric_Border_Leafs neighbor 172.17.20.0 local-address 172.17.20.1
set protocols bgp group Fabric_Border_Leafs neighbor 172.17.20.0 import ACCEPT_ALL
set protocols bgp group Fabric_Border_Leafs neighbor 172.17.20.0 export ONLY_ADVERTISE_DEFAULT
set protocols bgp group Fabric_Border_Leafs neighbor 172.17.20.0 peer-as 64501

# Configure a BGP session with the Backup VRF on Border Leaf 2
set protocols bgp group Fabric_Border_Leafs neighbor 172.17.20.2 description “BL-2 Backup”
set protocols bgp group Fabric_Border_Leafs neighbor 172.17.20.2 local-address 172.17.20.3
set protocols bgp group Fabric_Border_Leafs neighbor 172.17.20.2 import ACCEPT_ALL
set protocols bgp group Fabric_Border_Leafs neighbor 172.17.20.2 export ONLY_ADVERTISE_DEFAULT
set protocols bgp group Fabric_Border_Leafs neighbor 172.17.20.2 peer-as 64502
```

And in the fourth and final step, the four new logical interface units are added to a 'trusted' zone and all incoming traffic on any of these units is permitted both destined to and routed through the firewall.

This fourth step is only required because a Juniper SRX firewall (and in fact many other firewalls out there) discards all incoming traffic by default. If desired, this basic configuration can be enhanced later by the user, with more detailed and granular security policies, but it is outside the scope of this book:

```
# Add all four units to the 'trust' security zone
set security zones security-zone trust interfaces xe-0/0/16.2
set security zones security-zone trust interfaces xe-0/0/17.2
set security zones security-zone trust interfaces xe-0/0/17.5
set security zones security-zone trust interfaces xe-0/0/16.5

# Permit all incoming traffic in the 'trust' security zone
set security policies from-zone trust to-zone trust policy default-permit match source-address any
set security policies from-zone trust to-zone trust policy default-permit match destination-
address any
set security policies from-zone trust to-zone trust policy default-permit match application any
set security policies from-zone trust to-zone trust policy default-permit then permit
set security zones security-zone trust host-inbound-traffic system-services all
set security zones security-zone trust host-inbound-traffic protocols all
```

NOTE Please don't forget to commit the Junos configuration!

The configuration demonstrated in this chapter is not an extensive look at the Juniper SRX, however it touches on the essential pieces needed to create a BGP peering with the data center fabric and to perform some basic route filtering. Interface addresses will be added later in this chapter.

Establishing Dynamic Routing Between Tenants

So far in this chapter a new external system was introduced and a second logical tenant with a couple of associated virtual networks was created on the fabric. The user-facing interfaces were configured to permit untagged L2 access in the new virtual networks, for which the fabric will be acting as L3 gateway using EVPN multi-homing as part of the Apstra blueprint. The firewall was also fully configured and is now ready to route traffic between the two fabric tenants.

Now it's time to focus on the border leafs again and use Apstra to configure the firewall-facing logical interfaces, as well as a dynamic routing protocol to enable routing of traffic in and out of the fabric for each tenant. Since BGP is the only supported external routing protocol in Apstra at the time of this writing, and it is also the cornerstone of Juniper's Data Center design, it will be used to establish dynamic routing between the fabric and the SRX1500 Firewall.

Figure 6.12 depicts a high-level view of the BGP policy between fabric tenants and the firewall. Tenants will advertise all their routes to the firewall and the firewall will advertise a default router back to the tenant.

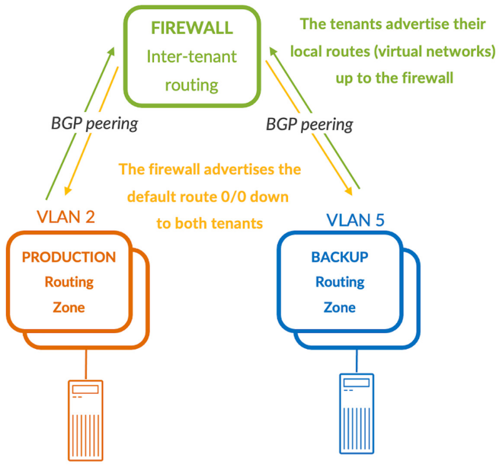


Figure 6.12 Multi-tenant BGP Routing Design Model

Establishing connectivity between the fabric and the partially configured SRX firewall will be done by performing the following tasks in order:

1. Create a connectivity template for the external BGP peering between the border leafs and the firewall. This will implicitly create the underlying point-to-point IP link.

2. Assign a set of IP addresses to each logical link connecting the border leafs to the firewall.

3. Lastly, configure the firewall itself with interface IP addresses and BGP peering details.

Connectivity templates were covered in Chapter 5, but two new primitives are being used here to achieve connectivity. It is important to remember that a connectivity template is a collection of primitives (low level networking actions). The two new primitives introduced in this chapter are:

- IP Link: This primitive uses Apstra resource pools to assign Link IPs between the Apstra fabric and generic systems, dynamically allocating an IP endpoint (/31) on each side of the link. They can create an IP link for any routing zone including the default routing zone and if the link is tagged, optionally a VLAN ID.

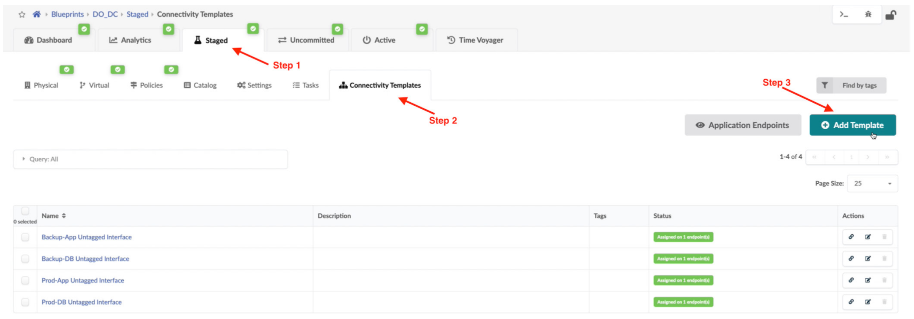
- BGP Peering (Generic System): The BGP peering (generic system) primitive creates a BGP peering session with a Generic System such as a router or firewall that is not explicitly controlled in Apstra, as shown earlier in this chapter. The Generic System is inherited from Apstra Generic System properties such as loopback and ASN. This primitive connects to a virtual network (single) or IP link connectivity point primitive.

Okay, let's create a connectivity template that enables BGP peering.

Step 1: Go to the Staged section in main blueprint view. (See Figure 6.13.)

Step 2: Click the Connectivity Templates tab.

Step 3: Click on Add Template.



0 selected	Name	Description	Tags	Status	Actions
	Backup-App Untagged Interface			Assigned on 1 endpoint(s)	
	Backup-DB Untagged Interface			Assigned on 1 endpoint(s)	
	Prod-App Untagged Interface			Assigned on 1 endpoint(s)	
	Prod-DB Untagged Interface			Assigned on 1 endpoint(s)	

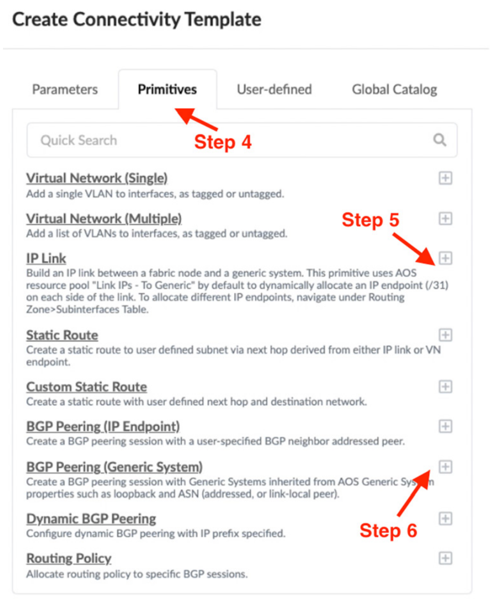
Figure 6.13 *Adding a New Connectivity Template for BGP Dynamic Routing*

In the next window (Figure 6.14), add two primitives to this new template.

Step 4: Click on the Primitives tab.

Step 5: Click on the plus sign (+) corresponding to the IP Link item.

Step 6: Click on the plus sign (+) for the BGP Peering (Generic System) item.



Virtual Network (Single)
Add a single VLAN to interfaces, as tagged or untagged.

Virtual Network (Multiple)
Add a list of VLANs to interfaces, as tagged or untagged.

IP Link
Build an IP link between a fabric node and a generic system. This primitive uses AOS resource pool "Link IPs - To Generic" by default to dynamically allocate an IP endpoint (/31) on each side of the link. To allocate different IP endpoints, navigate under Routing Zone>Subinterfaces Table.

Static Route
Create a static route to user defined subnet via next hop derived from either IP link or VN endpoint.

Custom Static Route
Create a static route with user defined next hop and destination network.

BGP Peering (IP Endpoint)
Create a BGP peering session with a user-specified BGP neighbor addressed peer.

BGP Peering (Generic System)
Create a BGP peering session with Generic Systems inherited from AOS Generic System properties such as loopback and ASN (addressed, or link-local peer).

Dynamic BGP Peering
Configure dynamic BGP peering with IP prefix specified.

Routing Policy
Allocate routing policy to specific BGP sessions.

Figure 6.14 *Adding a New Connectivity Template for BGP Dynamic Routing*

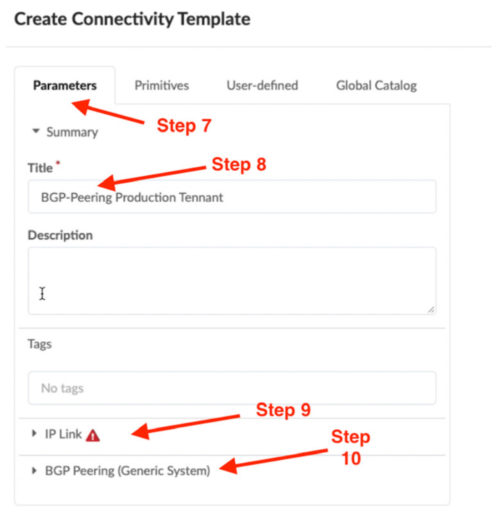
Next, edit both items in the Parameters tab. See Figure 6.15.

Step 7: Click on the Parameters tab.

Step 8: Name this template BGP Peering Production Tenant.

Step 9: Click on the IP Link item here, select the Production routing zone and assign it VLAN 2. Leave all other options as default.

Step 10: Click on the BGP Peering (Generic System) item, select Peer From Interface and Peer To Interface/IP Endpoint. Leave all other options as default and click the Create button on the right (not shown in the screen capture).


Create Connectivity Template

Title: BGP-Peering Production Tennant

Description:

Tags: No tags

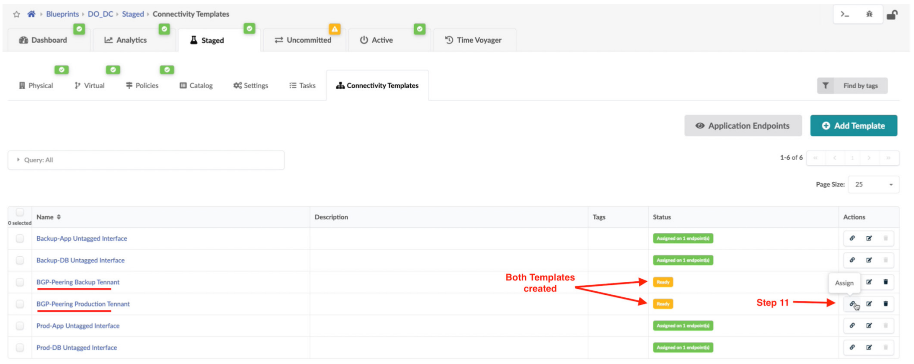
Figure 6.15 *Adding a New Connectivity Template for BGP Dynamic Routing*

We are not explicitly configuring any local or neighbor AS Numbers for this BGP session! This is because Apstra already knows what ASNs to use: the Generic System's ASN was configured earlier in this chapter when this new device was added to the blueprint and the fabric switches each already have an ASN assigned by Apstra during device onboarding.

NOTE Remember that this scenario requires another connectivity template following these same steps but name it *BGP Peering Backup Tenant* and assign VLAN 5 to its IP Link item. The second template's creation is omitted for brevity.

Once complete the new connectivity templates can be seen in the Connectivity Templates list. However they need to be assigned to the firewall-facing physical interfaces. Let's get that done.

Step 11: In the Connectivity Templates tab, click on the Assign button corresponding to the Production tenant. See Figure 6.16.



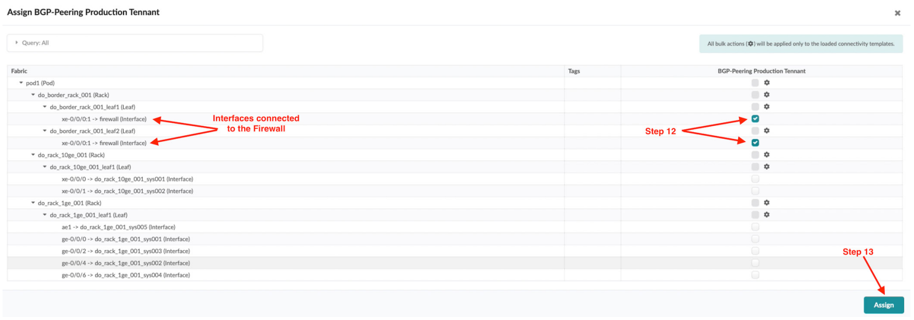
0 selected	Name	Description	Tags	Status	Actions
	Backup-App Untagged Interface			Assigned on 1 endpoint(s)	
	Backup-DB Untagged Interface			Assigned on 1 endpoint(s)	
	BGP-Peering Backup Tennant			Ready	
	BGP-Peering Production Tennant			Ready	
	Prod-App Untagged Interface			Assigned on 1 endpoint(s)	
	Prod-DB Untagged Interface			Assigned on 1 endpoint(s)	

Figure 6.16 *Assigning a Connectivity Template for BGP Dynamic Routing*

In the next pop-up window (Figure 6.17) identify the firewall-facing interfaces and assign the template.

Step 12: Click both checkboxes corresponding to the firewall-facing interfaces

Step 13: Click the Assign button.


Assign BGP-Peering Production Tennant
Query: All
All bulk actions (⚙) will be applied only to the loaded connectivity templates.
Interfaces connected to the Firewall
Step 12
Step 13
Assign

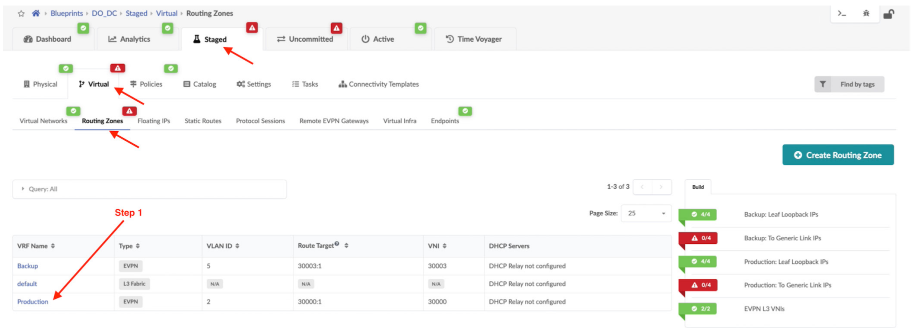
Fabric	Tags	BGP-Peering Production Tennant
pod1 (Pod)		☐ ⚙
do_border_rack_001 (Rack)		☐ ⚙
do_border_rack_001_leaf1 (Leaf)		☐ ⚙
xe-0/0/0:1 -> firewall (Interface)		✓
do_border_rack_001_leaf2 (Leaf)		☐ ⚙
xe-0/0/0:1 -> firewall (Interface)		✓
do_rack_10ge_001 (Rack)		☐ ⚙
do_rack_10ge_001_leaf1 (Leaf)		☐ ⚙
xe-0/0/0 -> do_rack_10ge_001_sys001 (Interface)		☐
xe-0/0/1 -> do_rack_10ge_001_sys002 (Interface)		☐
do_rack_1ge_001 (Rack)		☐ ⚙
do_rack_1ge_001_leaf1 (Leaf)		☐ ⚙
ae1 -> do_rack_1ge_001_sys005 (Interface)		☐
ge-0/0/0 -> do_rack_1ge_001_sys001 (Interface)		☐
ge-0/0/2 -> do_rack_1ge_001_sys003 (Interface)		☐
ge-0/0/4 -> do_rack_1ge_001_sys002 (Interface)		☐
ge-0/0/6 -> do_rack_1ge_001_sys004 (Interface)		☐

Figure 6.17 *Assigning a Connectivity Template for BGP Dynamic Routing*

NOTE Follow steps 11-13 for the Backup template and assign it to the same physical interfaces. In this example the second template is omitted for brevity.

Now that the connectivity templates are assigned, the last step required in Apstra is to assign an IP address to these physical links connecting to the firewall.

Step 1. In the Staged > Virtual > Routing Zones tab, click on the Production VRF. See Figure 6.18.



VRF Name	Type	VLAN ID	Route Target	VNI	DHCP Servers
Backup	EVPN	5	30003:1	30003	DHCP Relay not configured
default	L3 Fabric	N/A	N/A	N/A	DHCP Relay not configured
Production	EVPN	2	30000:1	30000	DHCP Relay not configured

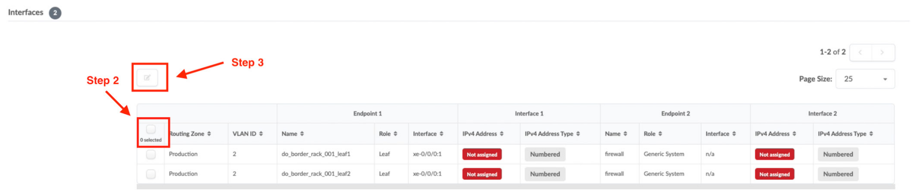
✓ 4/4	Backup: Leaf Loopback IPs
⚠ 0/4	Backup: To Generic Link IPs
✓ 4/4	Production: Leaf Loopback IPs
⚠ 0/4	Production: To Generic Link IPs
✓ 2/2	EVPN L3 VNIs

Figure 6.18 *Assigning IP Addresses to Physical Links*

On the next page, scroll to the bottom and select all available interfaces.

Step 2. Click the top checkbox in the Interfaces section.

Step 3. Click the Edit button.



			Endpoint 1			Interface 1		Endpoint 2			Interface 2	
	Routing Zone	VLAN ID	Name	Role	Interface	IPv4 Address	IPv4 Address Type	Name	Role	Interface	IPv4 Address	IPv4 Address Type
	Production	2	do_border_rack_001_leaf1	Leaf	xe-0/0/0:1	Not assigned	Numbered	firewall	Generic System	n/a	Not assigned	Numbered
	Production	2	do_border_rack_001_leaf2	Leaf	xe-0/0/0:1	Not assigned	Numbered	firewall	Generic System	n/a	Not assigned	Numbered

Figure 6.19 *Assigning IP Addresses to Physical Links*

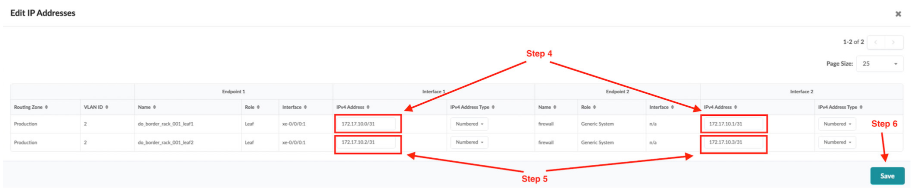
In the pop-up window go ahead and assign IP addresses to each of the two interfaces. These are just point-to-point interfaces so a /31 for each of them should suffice. These IP addresses will be used by Apstra to establish the BGP sessions between Border Leafs and the Firewall.

NOTE Based on this book's scenario, IP addresses were manually assigned to the links between both tenants and the firewall. It is possible for Apstra to auto-assign these.

Step 4. Assign 172.17.10.0/31 to the first interface on Border Leaf 1 and 172.17.10.1/31 to the Generic System's remote end of the link. See Figure 6.20.

Step 5. Assign 172.17.10.2/31 to the first interface on Border Leaf 2 and 172.17.10.3/31 to the Generic System.

Step 6. Click Save.


Edit IP Addresses

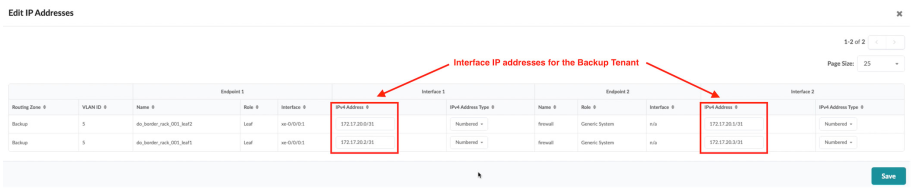
1-2 of 2

Page Size: 25

		Endpoint 1			Interface 1		Endpoint 2			Interface 2	
Routing Zone	VLAN ID	Name	Role	Interface	IPv4 Address	IPv4 Address Type	Name	Role	Interface	IPv4 Address	IPv4 Address Type
Production	2	do_border_rack_001_leaf1	Leaf	xe-0/0/0:1	172.17.10.0/31	Numbered	firewall	Generic System	n/a	172.17.10.1/31	Numbered
Production	2	do_border_rack_001_leaf2	Leaf	xe-0/0/0:1	172.17.10.2/31	Numbered	firewall	Generic System	n/a	172.17.10.3/31	Numbered

Figure 6.20 *Assigning IP Addresses to Physical Links for Production VRF*

NOTE Follow steps 2-6 for the Backup VRF as well and assign it similar IP addresses from the 172.17.20.x/31 range. See Figure 6.21.


Edit IP Addresses

1-2 of 2

Page Size: 25

		Endpoint 1			Interface 1		Endpoint 2			Interface 2	
Routing Zone	VLAN ID	Name	Role	Interface	IPv4 Address	IPv4 Address Type	Name	Role	Interface	IPv4 Address	IPv4 Address Type
Backup	5	do_border_rack_001_leaf2	Leaf	xe-0/0/0:1	172.17.20.0/31	Numbered	firewall	Generic System	n/a	172.17.20.1/31	Numbered
Backup	5	do_border_rack_001_leaf1	Leaf	xe-0/0/0:1	172.17.20.2/31	Numbered	firewall	Generic System	n/a	172.17.20.3/31	Numbered

Figure 6.21 *Assigning IP Addresses to Physical Links for Backup VRF*

All recent changes will need committing. This is done in the Uncommitted tab by clicking the Commit button and optionally adding a revision description.

BGP peering between the data center fabric and the Juniper SRX router is now complete and routing between the two tenants is now possible. The remainder of this chapter will verify the connectivity.

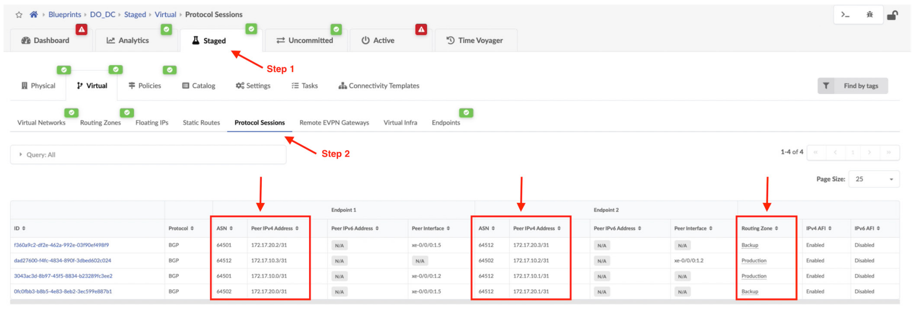
Verification

Now that changes are committed to the DC fabric, it's important to look at the routing protocol sessions in Apstra to ensure they are configured as expected.

Step 1. To verify the BGP session select the Staged tab in the blueprint. See Figure 6.22.

Step 2. Select the Virtual > Protocol Sessions tab to see the configured sessions.

Note that both session endpoints as well as the routing zone are shown in the captured table. In keeping with this book's scenario there are four sessions in total – two for each zone spread across both border leafs just as expected.



		Endpoint 1				Endpoint 2						
ID	Protocol	ASN	Peer IPv4 Address	Peer IPv6 Address	Peer Interface	ASN	Peer IPv4 Address	Peer IPv6 Address	Peer Interface	Routing Zone	IPv4 AFI	IPv6 AFI
f360a9c2-df2e-462a-992e-03f90ef498f9	BGP	64501	172.17.20.2/31	N/A	xe-0/0/0:1.5	64512	172.17.20.3/31	N/A	N/A	Backup	Enabled	Disabled
dad27600-f4fc-4834-890f-3dbed602c024	BGP	64512	172.17.10.3/31	N/A	N/A	64502	172.17.10.2/31	N/A	xe-0/0/0:1.2	Production	Enabled	Disabled
3043ac3d-8b97-45f5-8834-b23289fc3ee2	BGP	64501	172.17.10.0/31	N/A	xe-0/0/0:1.2	64512	172.17.10.1/31	N/A	N/A	Production	Enabled	Disabled
0fc0fbb3-b8b5-4e83-8eb2-3ec599e887b1	BGP	64502	172.17.20.0/31	N/A	xe-0/0/0:1.5	64512	172.17.20.1/31	N/A	N/A	Backup	Enabled	Disabled

Figure 6.22 *Configured BGP Sessions Between the Fabric and the Firewall*

As a final verification step, a ping session between the two hosts, Prod-APP-Host and Backup-DB-Host can be performed as shown in Figure 6.23 that a host in one tenant can reach another via the firewall.

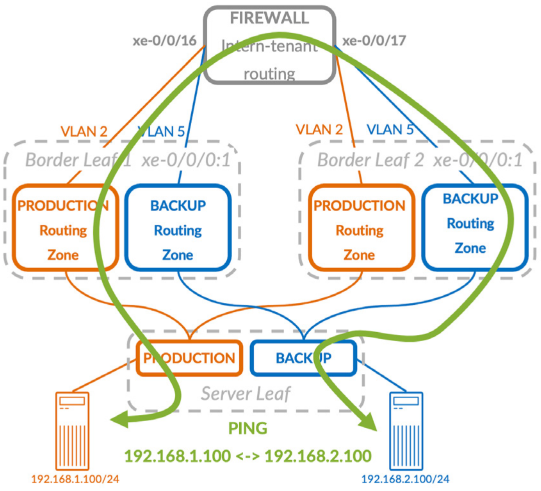


Figure 6.23 *One Possible Path for ICMP Traffic Between the Two Hosts (variable due to load-balancing)*

```
# ping from Prod-App host to Backup-DB host succeeds

user@PROD-APP-HOST> ping 192.168.2.100
PING 192.168.2.100 (192.168.2.100): 56 data bytes
64 bytes from 192.168.2.100: icmp_seq=0 ttl=253 time=1.028 ms
64 bytes from 192.168.2.100: icmp_seq=1 ttl=253 time=1.053 ms
```

```
^C --- 192.168.2.100 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
```

That concludes the main deployment pieces of this book, with two tenants deployed on the fabric and inter-tenant routed connectivity provided by the external firewall.

Summary

This chapter provided a step-by-step guide to add an external Generic System and its base configuration to the existing fabric testbed; it deployed an additional tenant called Backup with its own group of virtual networks and it established dynamic routing between the existing Production and the new Backup tenants.

Most live DC fabric deployments require much more detailed configuration than the bare minimum presented so far. The next chapter (Chapter 7) examines a few tools provided by Apstra for daily operations throughout a DC fabric lifecycle: adding custom configuration via Configlets and property sets, assigning new VNs to physical interfaces, and using the Time Voyager feature to perform a full fabric configuration rollback.

Chapter 7

Enhancing and Customizing a DC Deployment

When deploying and managing a fabric there are many different scenarios and tasks that will arise on a day-to-day basis. This chapter gives you some examples of such scenarios, first it shows you how to enhance deployment, and second it gives you an idea of tasks that can be seen during the day-to-day management of the network.

Deployment Enhancements Using Configlets and Property Sets

Deploying the fabric via Apstra enforces best practices and uses a tested design thanks to Apstra's reference architectures, however, at times this can feel very restrictive.

It's possible that in some deployments administrators will need to apply custom configurations to different devices within the fabric, outside the intent Apstra understands. The new configuration elements are not relevant to Apstra's deployment, but Apstra still needs to be aware of these additions to ensure consistency and compliance across the fabric. These additions could be, for example, adding configurations for NTP, DNS, SNMP, or even QoS.

This can be achieved in Apstra using two tools: *Configlets* and *property sets*. These tools allow for custom configuration to be added to the fabric in addition to managing the deployment of the configuration and enforcing the configuration throughout the fabric's lifecycle.

The first concept to dive into is Configlets. They are configuration templates that are applied to network devices. They augment Apstra's standard configuration and merge the changes onto the targeted devices. A Configlet is simply a way to apply standard switch configuration with Apstra that an engineer would typically apply via the CLI. For Junos, Apstra uses the text format but for other vendors Apstra uses different methods.

NOTE Apstra does not understand the intent of these. This means Configlets. Meaning that if incorrect configuration or configuration detrimental to the fabric's health is applied Apstra will still push this configuration. It is the administrator's responsibility to verify the configuration is correct, according to best practices and support before deployment.

For the purpose of providing an example in this section, SNMP configuration will be added to the already deployed fabric discussed in earlier chapters. By default, Apstra does not push any SNMP configuration to the fabric devices.

The SNMP configuration to be applied is as follows, it simply sets some basic SNMP parameters and tells the device to send any traps to a SNMP collector on port 162:

```
snmp {
    name “DO_DC_SNMP”;
    community do_dc_community {
        authorization read-only;
        clients {
            0.0.0.0/0;
        }
    }
    trap-group traps {
        destination-port 162;
        targets {
            192.168.1.100;
        }
    }
}
```

To create and publish these changes to the fabric, first a configlet needs to be created, then imported to the blueprint and finally deployed.

NOTE SNMP has been used in this example for simplicity, however a modern data center deployments could consider using Junos Telemetry Interface (JTI). JTI uses the push model to deliver data asynchronously, which eliminates polling from protocols such as SNMP. A request to send data is sent once by a management station to stream periodic updates from Junos devices. As a result, JTI is highly scalable and can support the monitoring of thousands of objects in a network.

Creating a Configlet

Step 1: Go to Design > Configlet. See Figure 7.1.

Step 2: Click Create Configlet.

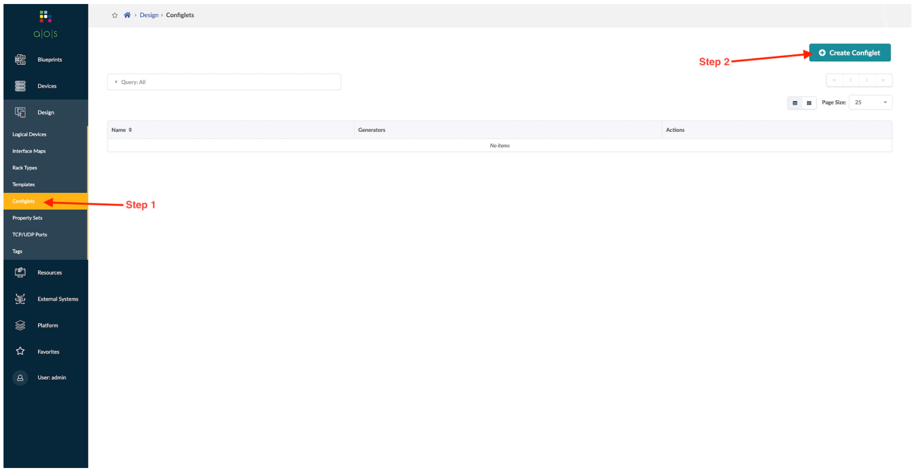
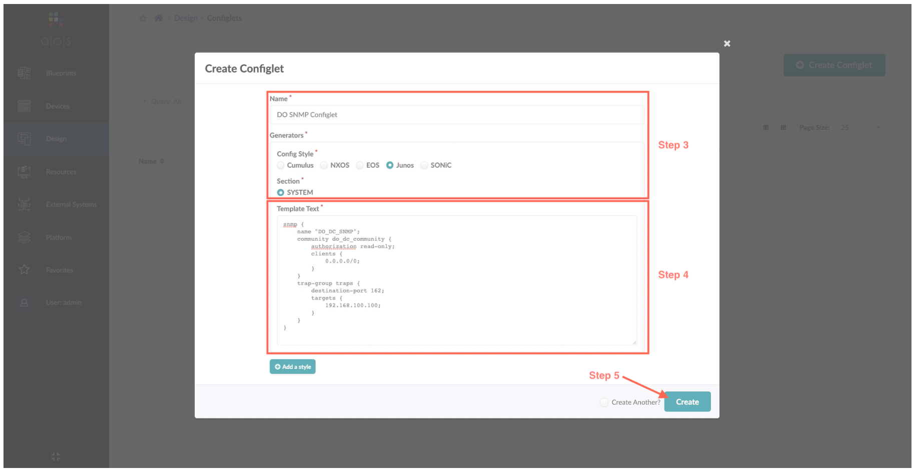


Figure 7.1 *Creating a New Configlet*

Step 3: Assign a Name, choose Junos as the Network OS type and text as the Configlet method. See Figure 7.2.

Step 4: Enter the Junos text-based configuration as shown.

Step 5: Click Create.



Name *: DO SNMP Configlet

Generators *

Config Style *: Cumulus NXOS EOS [✓ Junos] SONiC

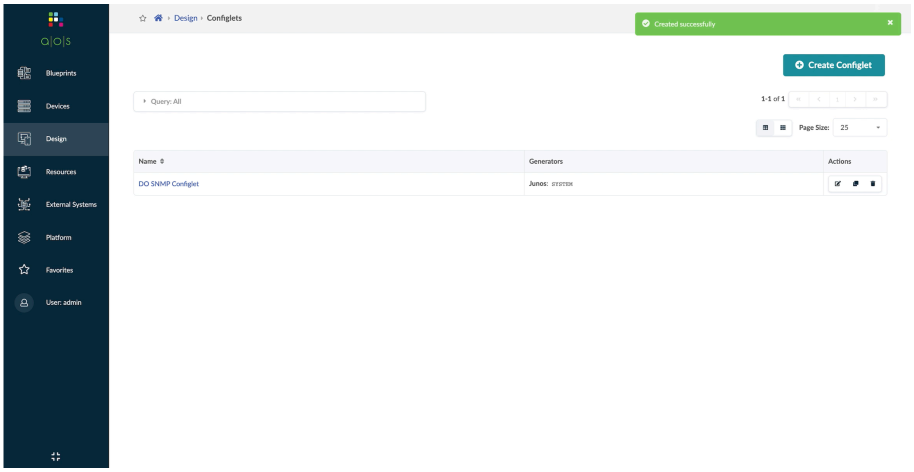
Section *: [✓ SYSTEM]

Template Text *

```
snmp {
    name "DO_DC_SNMP";
    community do_dc_community {
        authorization read-only;
        clients {
            0.0.0.0/0;
        }
    }
    trap-group traps {
        destination-port 162;
        targets {
            192.168.100.100;
        }
    }
}
```

Figure 7.2 *Creating a New Configlet*

Step 6: The new configlet is now created. See Figure 7.3.



Name	Generators	Actions
DO SNMP Configlet	Junos: SYSTEM	

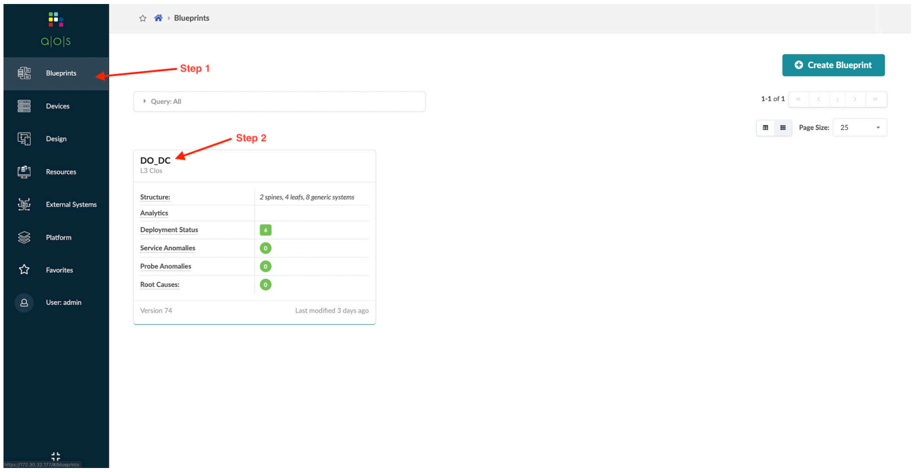
Figure 7.3 *Example of a Newly Created Configlet*

Now that the Configlet has been created, it needs to be imported into the chosen blueprint in order to be deployed.

Deploying a Configlet

Step 1: Go to Blueprints. See Figure 7.4.

Step 2: Select the desired blueprint.



Structure:	2 spines, 4 leafs, 8 generic systems
Analytics	
Deployment Status	6
Service Anomalies	0
Probe Anomalies	0
Root Causes:	0

Version 74 Last modified 3 days ago

Figure 7.4 *Deploying a New Configlet*

Step 3: Go to Staging > Catalog. See Figure 7.5.

Step 4: Select Configlets.

Step 5: Click Import Configlet.

NOTE The catalog found within an Apstra Blueprint is a copy of all the global objects created outside of the Blueprint such as Logical Devices, Interface Maps, Property Sets, and Configlets. The reason a Blueprint always references its internal copies, and not the global resources, is because global resources could be changed by a user long after a deployment, but this does not necessarily mean an already deployed DC fabric would want these changes applied.

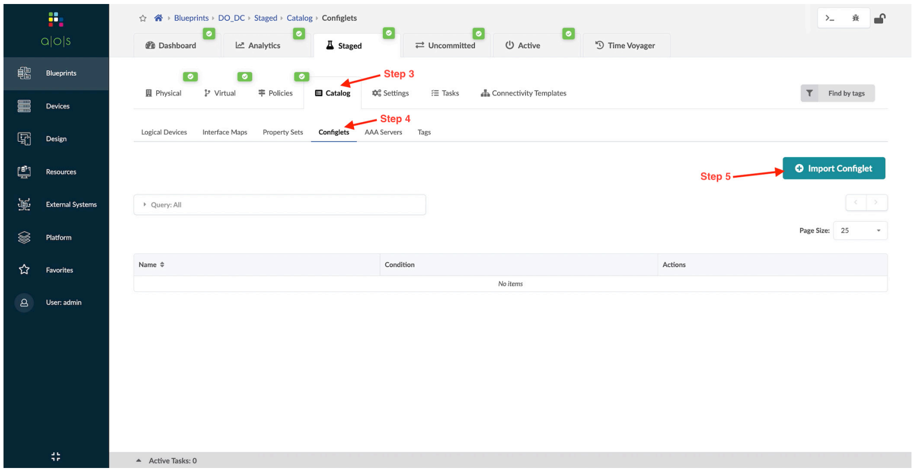


Figure 7.5 Deploying a New Configlet

Step 6: From the dropdown select the Configlet to be imported. See Figure 7.6.

Step 7: Select which devices the Configlet should be applied to, in this example the switch role is being used to determine the assignment.

Step 8: Click Import Configlet.

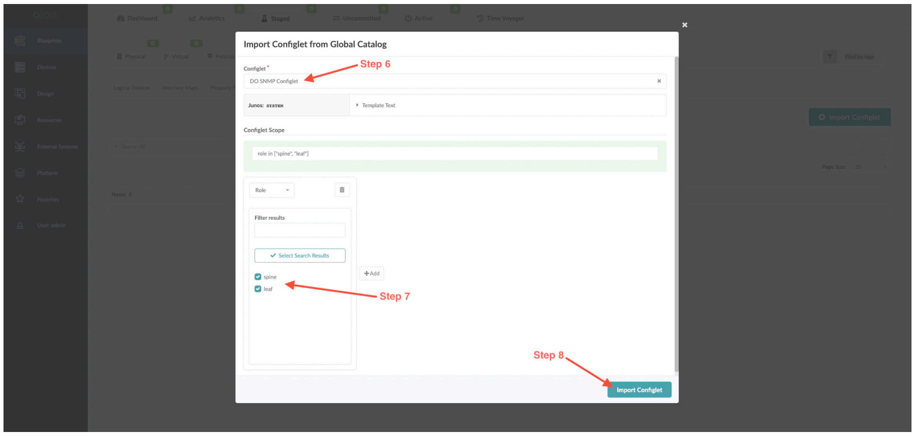
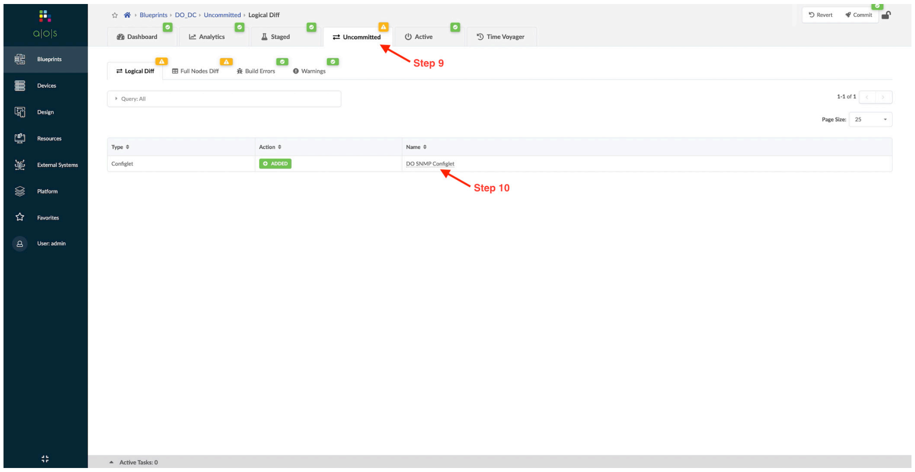


Figure 7.6 Deploying a New Configlet

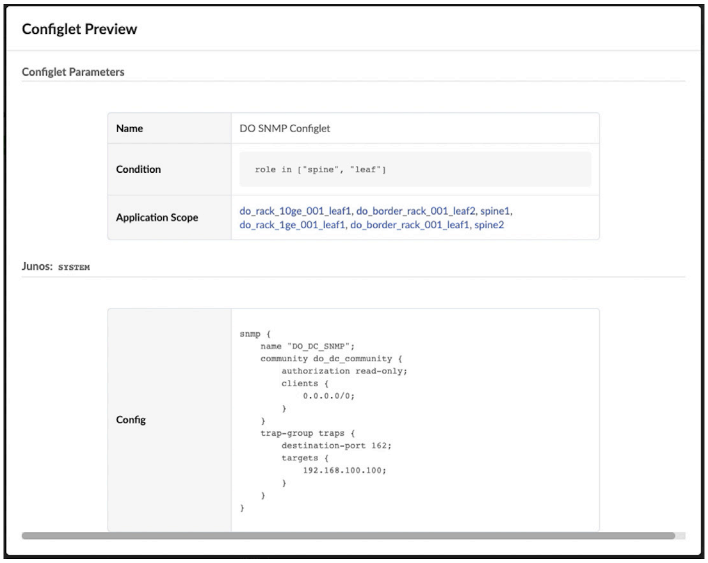
Step 9: Go to Uncommitted. See Figure 7.7.

Step 10: Click on the configlet name to view the staged configuration.



Type	Action	Name
Configlet	ADDED	DO SNMP Configlet

Figure 7.7 Reviewing a Staged Configlet



Configlet Preview

Configlet Parameters

Name	DO SNMP Configlet
Condition	`role in ["spine", "leaf"]`
Application Scope	do_rack_10ge_001_leaf1, do_border_rack_001_leaf2, spine1, do_rack_1ge_001_leaf1, do_border_rack_001_leaf1, spine2

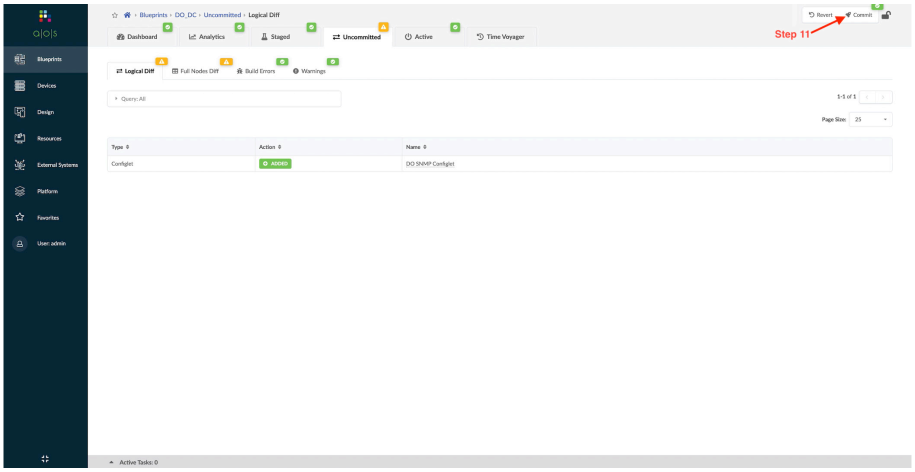
Junos: `SYSTEM`

Config

```
snmp {
    name "DO_DC_SNMP";
    community do_dc_community {
        authorization read-only;
        clients {
            0.0.0.0/0;
        }
    }
    trap-group traps {
        destination-port 162;
        targets {
            192.168.100.100;
        }
    }
}
```

Figure 7.8 Previewing a Staged Configlet

Step 11: Once viewed and reviewed click Commit.

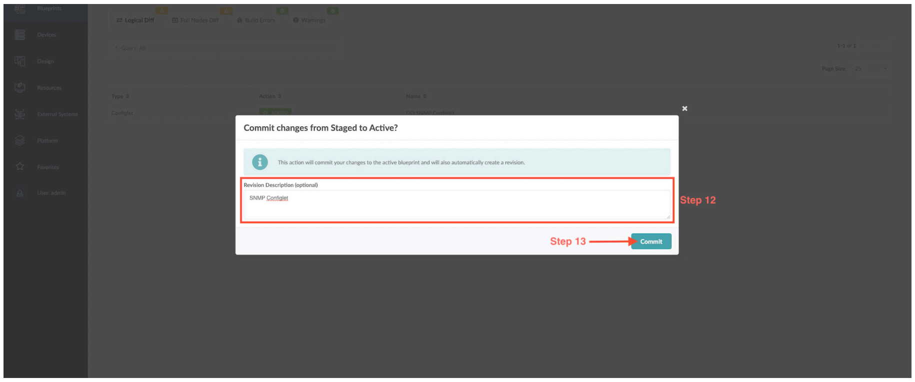


Type	Action	Name
Configlet	ADDED	DO SNMP Configlet

Figure 7.9 Committing Staged Configuration

Step 12: Provide a useful commit message. See Figure 7.10.

Step 13: Click Commit.


Commit changes from Staged to Active?

This action will commit your changes to the active blueprint and will also automatically create a revision.

Revision Description (optional): SNMP Configlet

Figure 7.10 Committing Staged Configuration

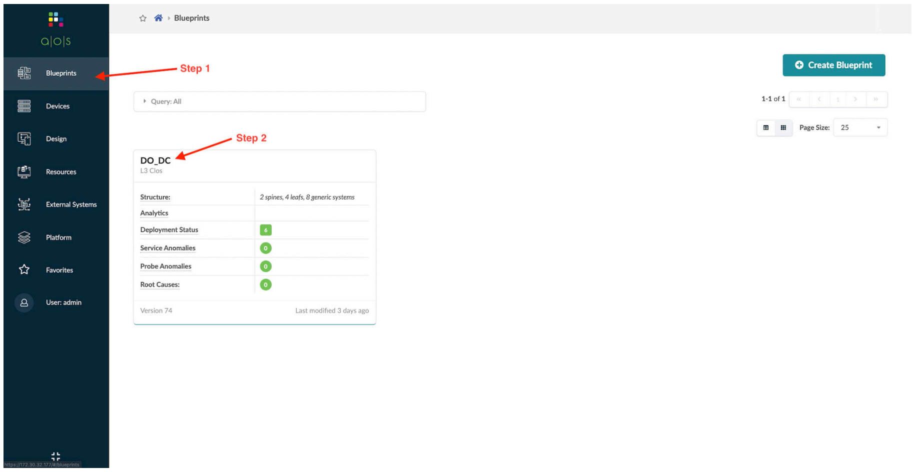
The configlet has now been deployed to the devices in the blueprint. The next stage is optional, but it is important to verify that the new Configlet has been applied to the target devices.

Verify Configlet

The final step is validating that the config has been pushed.

Step 1:Go to Blueprint. See Figure 7.11.

Step 2: Select the desired Blueprint.



DO_DC
L3 Clos

Structure:	2 spines, 4 leafs, 8 generic systems
Analytics	
Deployment Status	6
Service Anomalies	0
Probe Anomalies	0
Root Causes:	0

Version 74 Last modified 3 days ago

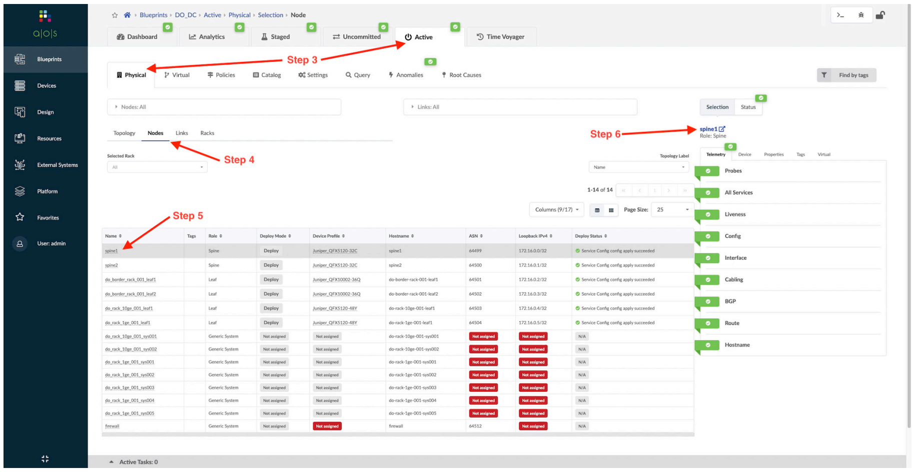
Figure 7.11 Verifying a Configlet Has Been Applied

Step 3: Go to Active > Physical. See Figure 7.12.

Step 4: Select Nodes.

Step 5: Select the nodes the change was pushed to.

Step 6: Select the node.



Name	Tags	Role	Deploy Mode	Device Profile	Hostname	ASN	Loopback IPv4	Deploy Status
spine1		Spine	Deploy	Juniper_QFX5120-32C	spine1	64499	172.16.0.0/32	Service Config config apply succeeded
spine2		Spine	Deploy	Juniper_QFX5120-32C	spine2	64500	172.16.0.1/32	Service Config config apply succeeded
do_border_rack_001_leaf1		Leaf	Deploy	Juniper_QFX10002-36Q	do-border-rack-001-leaf1	64501	172.16.0.2/32	Service Config config apply succeeded
do_border_rack_001_leaf2		Leaf	Deploy	Juniper_QFX10002-36Q	do-border-rack-001-leaf2	64502	172.16.0.3/32	Service Config config apply succeeded
do_rack_10ge_001_leaf1		Leaf	Deploy	Juniper_QFX5120-48Y	do-rack-10ge-001-leaf1	64503	172.16.0.4/32	Service Config config apply succeeded
do_rack_1ge_001_leaf1		Leaf	Deploy	Juniper_QFX5120-48Y	do-rack-1ge-001-leaf1	64504	172.16.0.5/32	Service Config config apply succeeded
do_rack_10ge_001_sys001		Generic System	Not assigned	Not assigned	do-rack-10ge-001-sys001	Not assigned	Not assigned	N/A
do_rack_10ge_001_sys002		Generic System	Not assigned	Not assigned	do-rack-10ge-001-sys002	Not assigned	Not assigned	N/A
do_rack_1ge_001_sys001		Generic System	Not assigned	Not assigned	do-rack-1ge-001-sys001	Not assigned	Not assigned	N/A
do_rack_1ge_001_sys002		Generic System	Not assigned	Not assigned	do-rack-1ge-001-sys002	Not assigned	Not assigned	N/A
do_rack_1ge_001_sys003		Generic System	Not assigned	Not assigned	do-rack-1ge-001-sys003	Not assigned	Not assigned	N/A
do_rack_1ge_001_sys004		Generic System	Not assigned	Not assigned	do-rack-1ge-001-sys004	Not assigned	Not assigned	N/A
do_rack_1ge_001_sys005		Generic System	Not assigned	Not assigned	do-rack-1ge-001-sys005	Not assigned	Not assigned	N/A
firewall		Generic System	Not assigned	Not assigned	firewall	64512	Not assigned	N/A

Figure 7.12 Verifying a Configlet Has Been Applied

Step 7: Click Telemetry. See Figure 7.13.

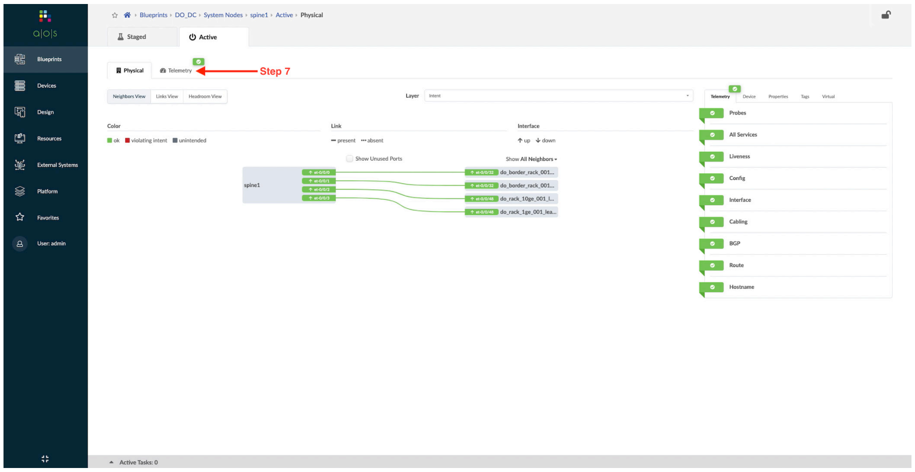


Figure 7.13 Verifying a Configlet Has Been Applied

Step 8: Click Config. See Figure 7.14.

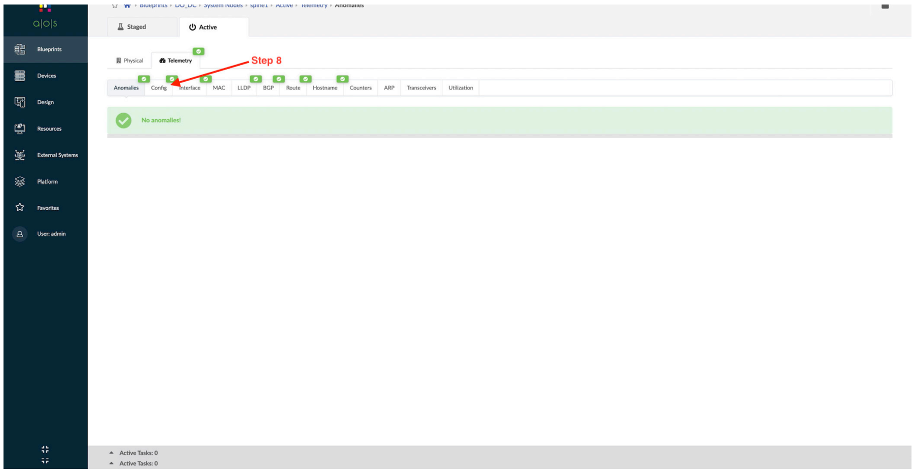
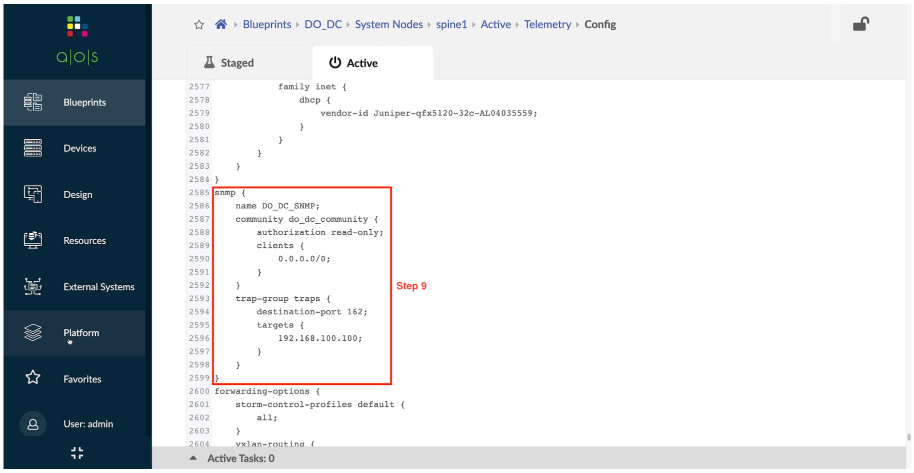


Figure 7.14 Verifying a Configlet Has Been Applied

Step 9: Find the added configuration. See Figure 7.15.



```
            family inet {
                dhcp {
                    vendor-id Juniper-qfx5120-32c-AL04035559;
                }
            }
        }
    }
}
snmp {
    name DO_DC_SNMP;
    community do_dc_community {
        authorization read-only;
        clients {
            0.0.0.0/0;
        }
    }
    trap-group traps {
        destination-port 162;
        targets {
            192.168.100.100;
        }
    }
}
forwarding-options {
    storm-control-profiles default {
        all;
    }
    vxlan-routing {
```

Figure 7.15 *Verifying a Configlet Has Been Applied*

Of course, this can also be done by logging on to the device via SSH or Telnet to verify the configuration change.

Once verified, the application of the custom configuration is complete! It's easy to see how powerful this can be as the fabric can be altered using Configlets, but with great power comes great responsibility. Making changes to the fabric that are outside of Asptra's reference design means that configuration should be rigorously tested to ensure no harm is brought to the DC fabric network.

Deployment Enhancement Using Configlets and Property Sets

Configlets are easy to create and deploy. However it's likely that details could differ across different DC fabric deployments. For example, DC1 may have a local SNMP collector with a host address of 192.168.100.100, whereas DC2's collector could have a host address of 10.0.0.100. Using the method just discussed, two different Configlets would need to be created. Repetition isn't always a good thing! This can be solved using *Property Sets*. Property sets are collections of key-value pairs that are imported into blueprints for use in Configlets. For example, instead of hard-coding values for NTP servers, SMTP servers, and syslog servers, they can be defined in a property set. These property sets can be associated with a Configlet once imported into a blueprint.

This means that the same Configlet can be used across multiple sites, with the use of a site-specific Property Set. The {{ }} brackets are used to define a variable in a configlet. The user simply replaces the configuration with a `{{ variable }}` and this is substituted for the actual configuration as defined in a property set. In the following example the same SNMP configuration we used before will be deployed, however some of the information will be parameterized.

Example Property Set

```
{{snmp_collector}} = 192.168.100.100
{{snmp_trap_port}} = 162
```

Examaple Configlet Config

```
snmp {
    name "DO_DC_SNMP";
    community do_dc_community {
        authorization read-only;
        clients {
            0.0.0.0/0;
        }
    }
    trap-group traps {
        destination-port {{snmp_trap_port}};
        targets {
            {{snmp_collector}};
        }
    }
}
```

Now we'll show how to create this Configlet and property set, import them into a blueprint, deploy the configuration, and finally verify it has been applied successfully.

Create a Property Set

Step1: Go to Design > Property Sets. See Figure 7.17.

Step 2: Click Create Property Set.

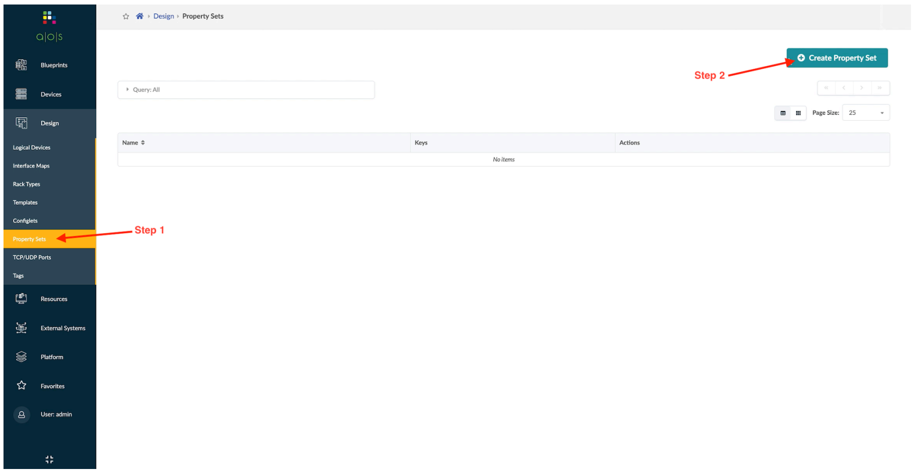
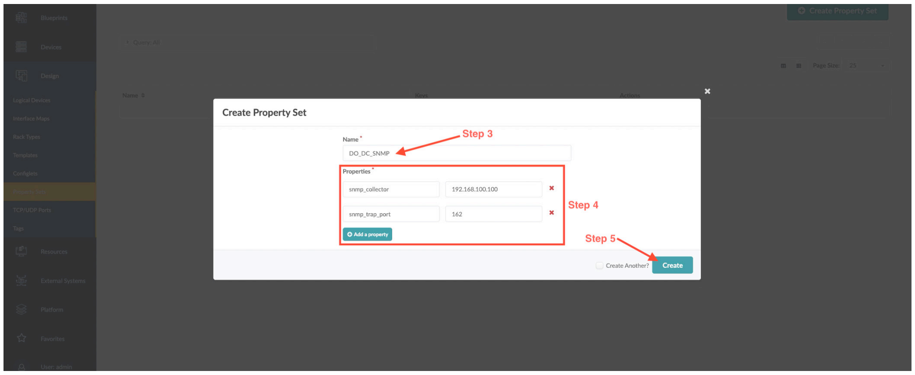


Figure 7.16 *Creating a Property Set*

Step 3: Assign a Name to the Property Set. See Figure 7.17.

Step 4: Enter the key value pairs. Multiple sets can be added by clicking Add a Property

Step 5: Click Create.



Create Property Set

Name *: DO_DC_SNMP

Properties *

snmp_collector	192.168.100.100
snmp_trap_port	162

Figure 7.17 Creating a Property Set

Create a Configlet

Step 1: Go to Design > Configlet. See Figure 7.18.

Step 2: Click Create Configlet.

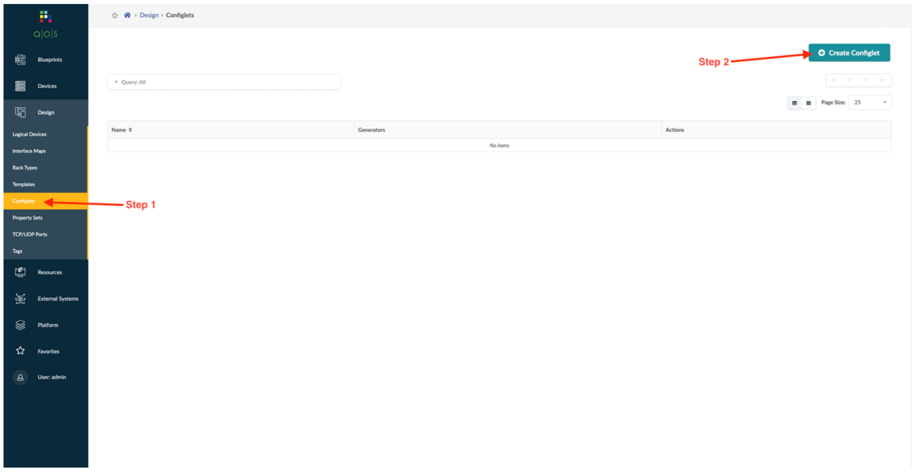
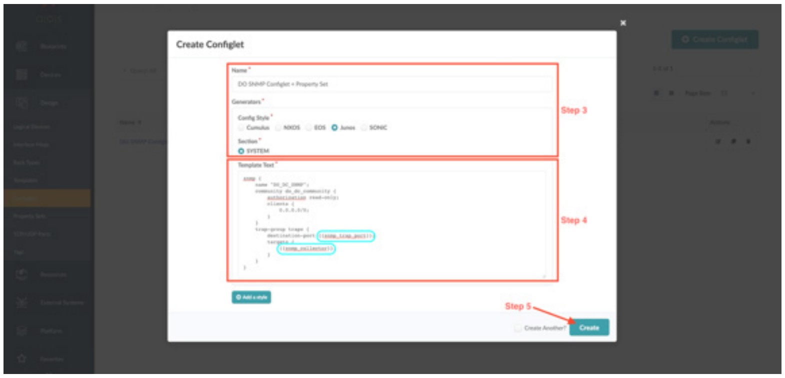


Figure 7.18 Creating a New Configlet

Step 3: Assign a Name, choose Junos as the Network OS type, and text as the Configlet method. See Figure 7.19.

Step 4: Enter the Junos text-based configuration as shown.

Step 5: Click Create.



Name: DO SNMP Configlet + Property Set

Generators: Config Style: Cumulus, NXOS, EOS, [✓ Junos], SONIC

Section: [✓ SYSTEM]

Template Text:

```
snmp {
    name "DO_DC_SNMP";
    community do_dc_community {
        authorization read-only;
        clients {
            0.0.0.0/0;
        }
    }
    trap-group traps {
        destination-port {{snmp_trap_port}};
        targets {
            {{snmp_collector}};
        }
    }
}
```

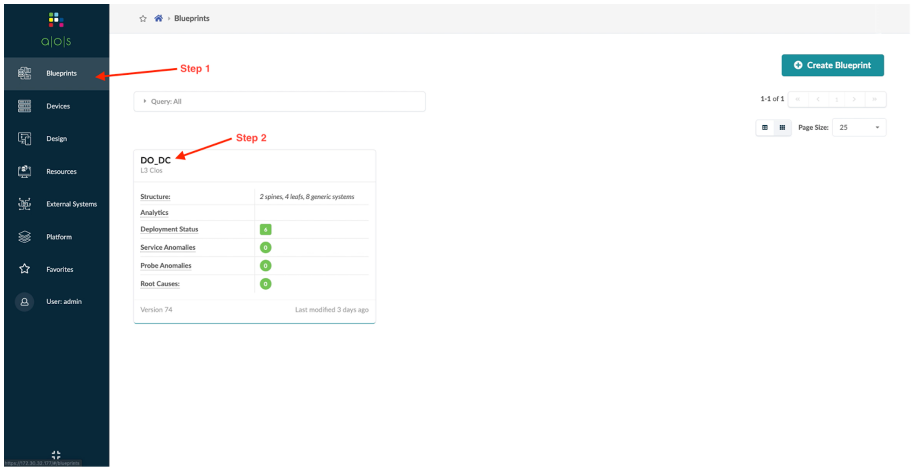
Figure 7.19 *Creating a New Configlet*

Now that the Configlet has been created it needs to be imported into the chosen blueprint.

Deploy Configlet and Property Set

Step 1: Go to Blueprint. See Figure 7.20.

Step 2: Select the desired blueprint.



DO_DC
L3 Clos

Structure:	2 spines, 4 leafs, 8 generic systems
Analytics	
Deployment Status	6
Service Anomalies	0
Probe Anomalies	0
Root Causes:	0

Version 74 Last modified 3 days ago

Figure 7.20 *Importing a Property Set*

Step 3: Go to Staged > Catalog. See Figure 7.21.

Step 4: Select Property Sets.

Step 5: Click Import Property Set.

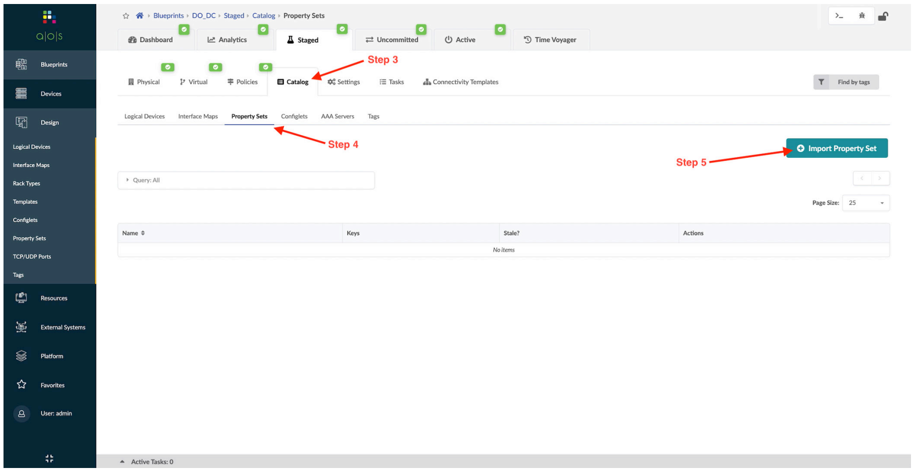


Figure 7.21 *Importing a Property Set*

Step 6: From the dropdown menuy select the Property Set to be imported. See Figure 7.22.

Step 7: Click Import Property Set.

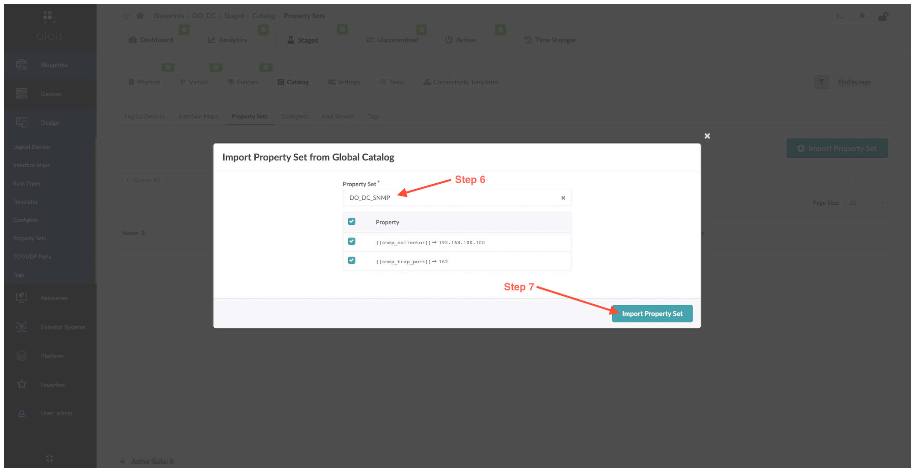


Figure 7.22 *Importing a Property Set*

Step 8: Go to Staged > Catalog. See Figure 7.23.

Step 9: Select Configlets.

Step 10: Click Import Configlets.

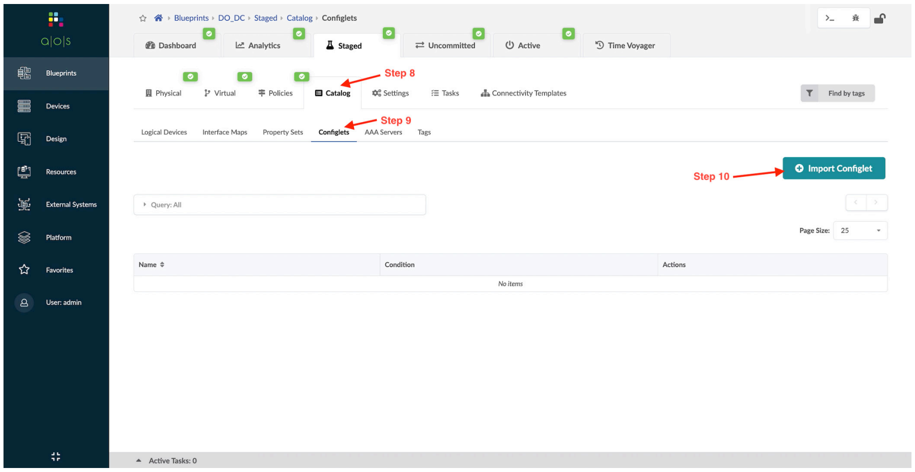


Figure 7.23 *Importing a Configlet*

Step 11: From the dropdown select the Configlet to be imported. See Figure 7.24.

Step 12: Select which devices the Configlet should be applied to. In this example the switch role is being used to determine the assignment.

Step 13: Click Import Configlet.

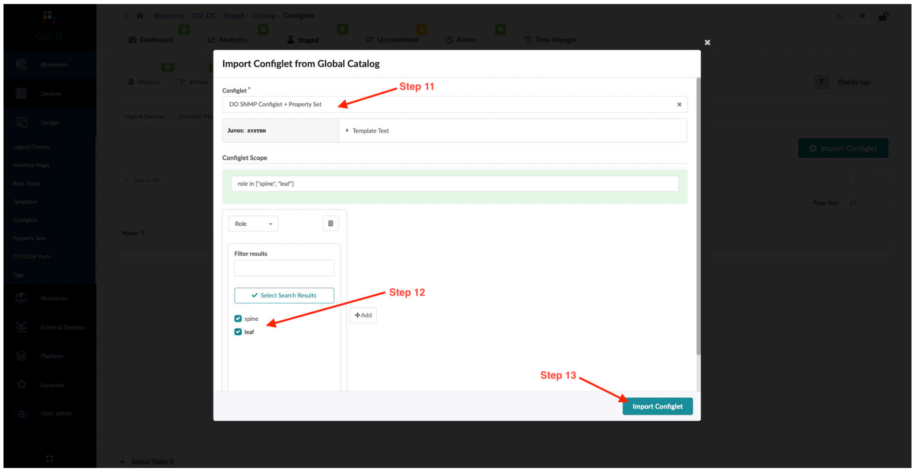
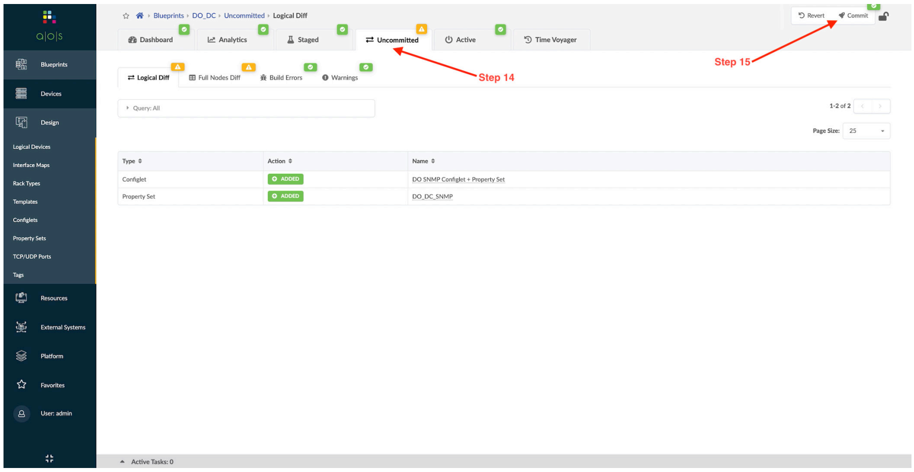


Figure 7.24 *Importing a Configlet*

Step 13: Go to Uncommitted. The configuration to be added can be viewed here. See Figure 7.25.

Step 14: Click Commit.

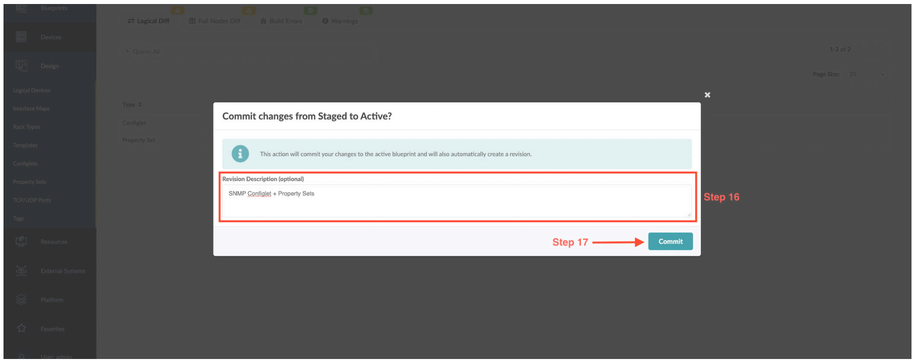


Type	Action	Name
Configlet	ADDED	DO SNMP Configlet + Property Set
Property Set	ADDED	DO_DC_SNMP

Figure 7.25 Committing Staged Configuration

Step 15: Provide a useful commit message. Adding a description is highly encouraged before committing, because otherwise it's not all that easy to remember which revision is which in Time Voyager, as discussed later in this chapter. See Figure 7.26.

Step 16: Click Commit.



Commit changes from Staged to Active?

This action will commit your changes to the active blueprint and will also automatically create a revision.

Revision Description (optional)

SNMP Configlet + Property Sets

Figure 7.26 Committing Staged Configuration

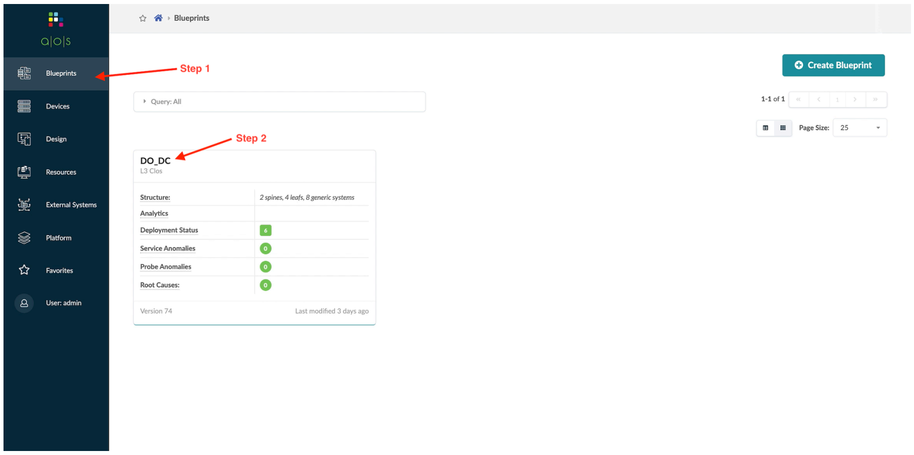
The Configlet has now been deployed to the devices in the blueprint, incorporating the values from the Property Set that was imported.

Verify Configlet

The final step is to validate the config has been pushed.

Step1: Go to Blueprint. See Figure 7.27.

Step 2: Select the desired Blueprint.



DO_DC
L3 Clos

Structure:	2 spines, 4 leafs, 8 generic systems
Analytics	
Deployment Status	6
Service Anomalies	0
Probe Anomalies	0
Root Causes:	0

Version 74 Last modified 3 days ago

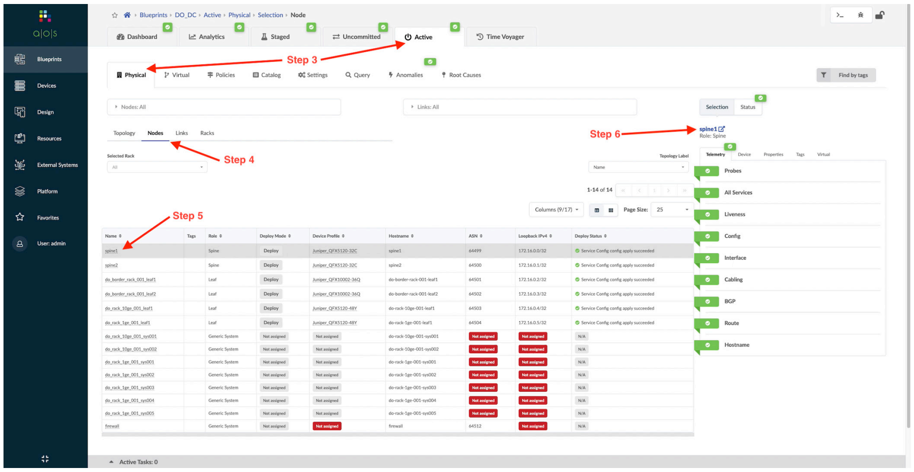
Figure 7.27 *Verifying a Configlet Has Been Applied*

Step 3: Go to Active > Physical. See Figure 7.28.

Step 4: Select Nodes.

Step 5: Select one of the nodes the change was pushed to.

Step 6: Select the node.



Name	Tags	Role	Deploy Mode	Device Profile	Hostname	ASN	Loopback IPv4	Deploy Status
spine1		Spine	Deploy	Juniper_QFX5120-32C	spine1	64499	172.16.0.0/32	Service Config config apply succeeded
spine2		Spine	Deploy	Juniper_QFX5120-32C	spine2	64500	172.16.0.1/32	Service Config config apply succeeded
do_border_rack_001_leaf1		Leaf	Deploy	Juniper_QFX10002-36Q	do-border-rack-001-leaf1	64501	172.16.0.2/32	Service Config config apply succeeded
do_border_rack_001_leaf2		Leaf	Deploy	Juniper_QFX10002-36Q	do-border-rack-001-leaf2	64502	172.16.0.3/32	Service Config config apply succeeded
do_rack_10ge_001_leaf1		Leaf	Deploy	Juniper_QFX5120-48Y	do-rack-10ge-001-leaf1	64503	172.16.0.4/32	Service Config config apply succeeded
do_rack_1ge_001_leaf1		Leaf	Deploy	Juniper_QFX5120-48Y	do-rack-1ge-001-leaf1	64504	172.16.0.5/32	Service Config config apply succeeded
do_rack_10ge_001_sys001		Generic System	Not assigned	Not assigned	do-rack-10ge-001-sys001	Not assigned	Not assigned	N/A
do_rack_10ge_001_sys002		Generic System	Not assigned	Not assigned	do-rack-10ge-001-sys002	Not assigned	Not assigned	N/A
do_rack_1ge_001_sys001		Generic System	Not assigned	Not assigned	do-rack-1ge-001-sys001	Not assigned	Not assigned	N/A
do_rack_1ge_001_sys002		Generic System	Not assigned	Not assigned	do-rack-1ge-001-sys002	Not assigned	Not assigned	N/A
do_rack_1ge_001_sys003		Generic System	Not assigned	Not assigned	do-rack-1ge-001-sys003	Not assigned	Not assigned	N/A
do_rack_1ge_001_sys004		Generic System	Not assigned	Not assigned	do-rack-1ge-001-sys004	Not assigned	Not assigned	N/A
do_rack_1ge_001_sys005		Generic System	Not assigned	Not assigned	do-rack-1ge-001-sys005	Not assigned	Not assigned	N/A
firewall		Generic System	Not assigned	Not assigned	firewall	64512	Not assigned	N/A

Figure 7.28 *Verifying a Configlet Has Been Applied*

Step 7: Click Telemetry. See Figure 7.29.

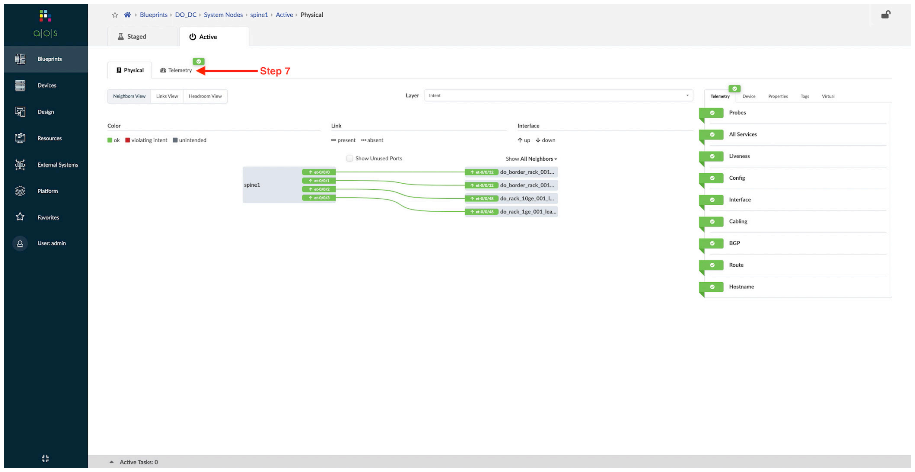


Figure 7.29 Verifying a Configlet Has Been Applied

Step 8: Click Config. See Figure 7.30.

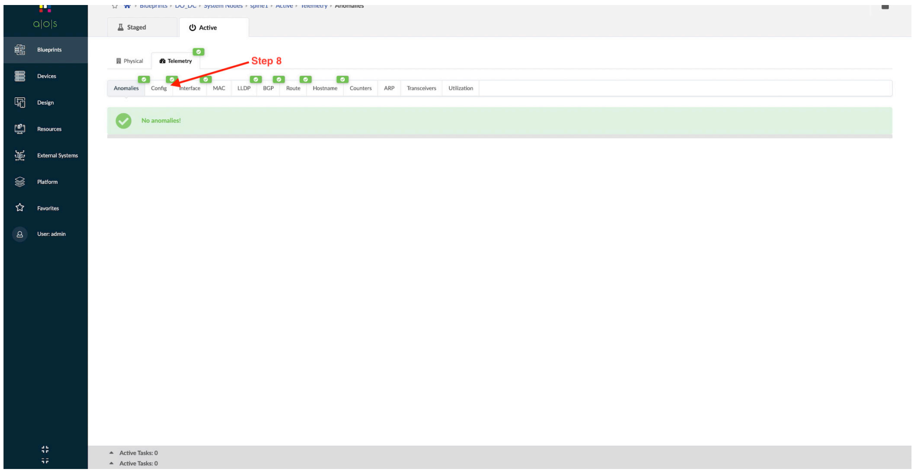
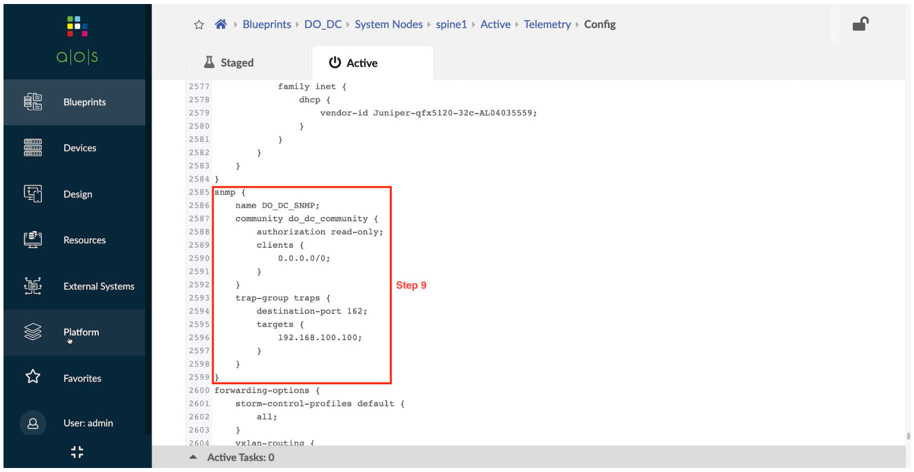


Figure 7.30 Verifying a Configlet Has Been Applied

Step 9: Find and view the added configuration. See Figure 7.31.

If the user wants to verify the configuration on the device, they can achieve this by logging onto the device via SSH or Telnet and displaying it on the CLI.



```
                family inet {
                    dhcp {
                        vendor-id Juniper-qfx5120-32c-AL04035559;
                    }
                }
            }
        }
}
snmp {
    name DO_DC_SNMP;
    community do_dc_community {
        authorization read-only;
        clients {
            0.0.0.0/0;
        }
    }
    trap-group traps {
        destination-port 162;
        targets {
            192.168.100.100;
        }
    }
}
forwarding-options {
    storm-control-profiles default {
        all;
    }
    vxlan-routing {
```

Figure 7.31 Verifying a Configlet Has Been Applied

As before, once verified, the application of the custom configuration is complete! The addition of property sets means that Configlets can be used across multiple data centers when a unique configuration is needed at each site.

Focusing on Day 2 Operations

The final part of this chapter provides examples of how traditional day-to-day tasks are performed when using Apstra. This is by no means an extensive list, but it gives insight into how operators might perform typical Day 2 tasks on the network.

Assign Interfaces to VNIs

One of the more common Day 2 activities an operator will undertake is assigning physical interfaces to different tenants and virtual networks. In a traditional EVPN/VXLAN fabric this can be a tedious job, especially when having to assign multiple ports on different leaf switches.

Apstra makes this simple and with a few clicks an operator can quickly and reliably perform this task.

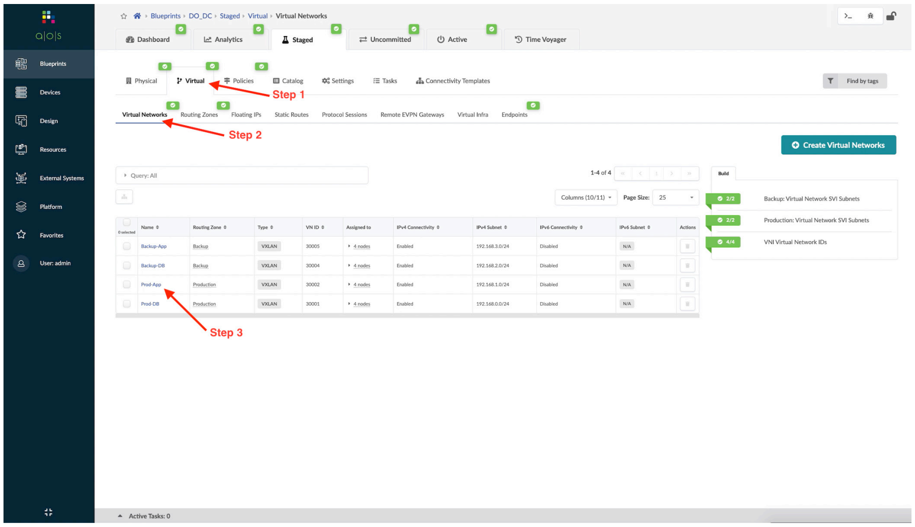
In the following example a single VNI will be assigned to a physical interface. It is also possible to assign multiple VNIs to a physical interface. To understand how to configure multiple VNIs to a single interface please consult the Apstra's documentation: https://www.juniper.net/documentation/product/us/en/apstra.

Assigning VNIs

Step 1: From the given blueprint got to Staged > Virtual. See Figure 7.32.

Step 2: Select Virtual Networks.

Step 3: Select the Virtual Network to review current physical port mappings. Figure 7.33 shows the current physical port assignments.



0 selected	Name	Routing Zone	Type	VN ID	Assigned to	IPv4 Connectivity	IPv4 Subnet	IPv6 Connectivity	IPv6 Subnet	Actions
	Backup-App	Backup	VXLAN	30005	4 nodes	Enabled	192.168.3.0/24	Disabled	N/A	
	Backup-DB	Backup	VXLAN	30004	4 nodes	Enabled	192.168.2.0/24	Disabled	N/A	
	Prod-App	Production	VXLAN	30002	4 nodes	Enabled	192.168.1.0/24	Disabled	N/A	
	Prod-DB	Production	VXLAN	30001	4 nodes	Enabled	192.168.0.0/24	Disabled	N/A	

✓ 2/2	Backup: Virtual Network SVI Subnets
✓ 2/2	Production: Virtual Network SVI Subnets
✓ 4/4	VNI Virtual Network IDs

Figure 7.32 Selecting VNI

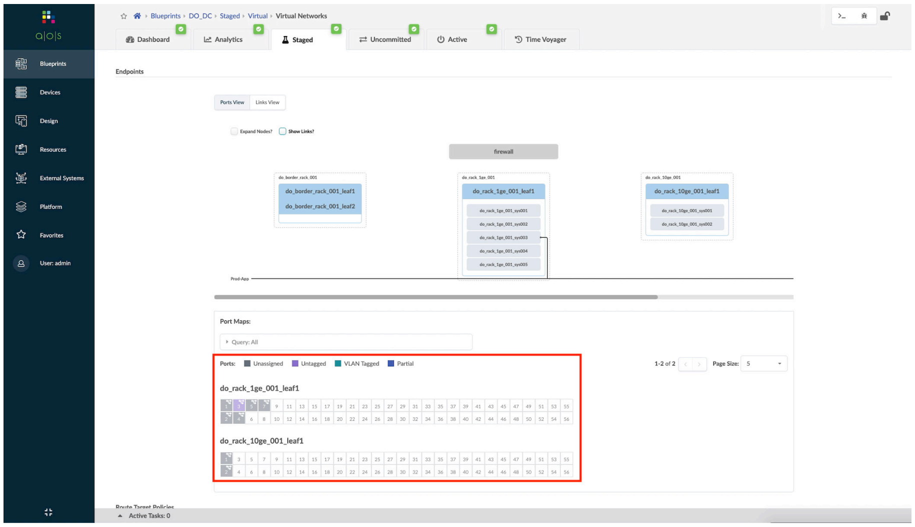
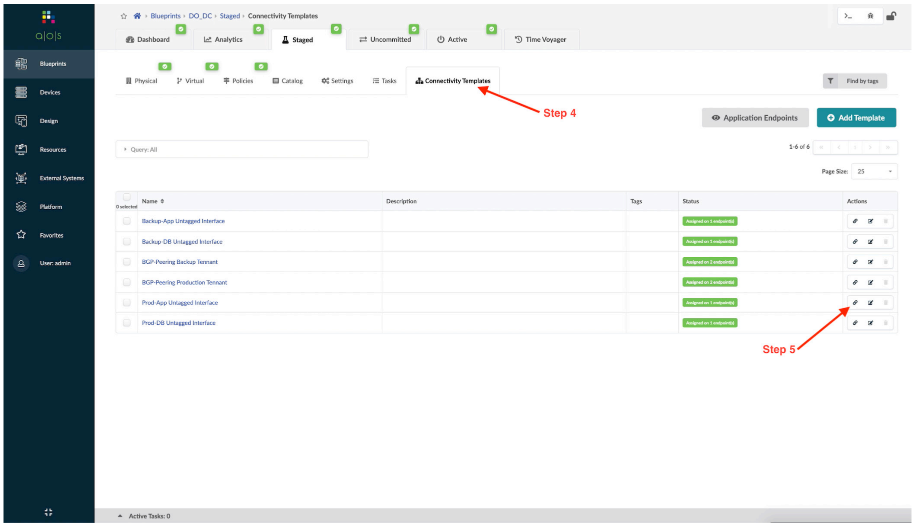


Figure 7.33 Viewing VNI Port Assignment Pre-Change

Step 4: To assign a new port, go to Staged > Connectivity Templates. See Figure 7.34.

Step 5: Click the chain link symbol of the connectivity template that will be assigned to the port. The connectivity shown in Figure 7.34 has already been created. Please refer to the creation of connectivity templates in previous chapters.

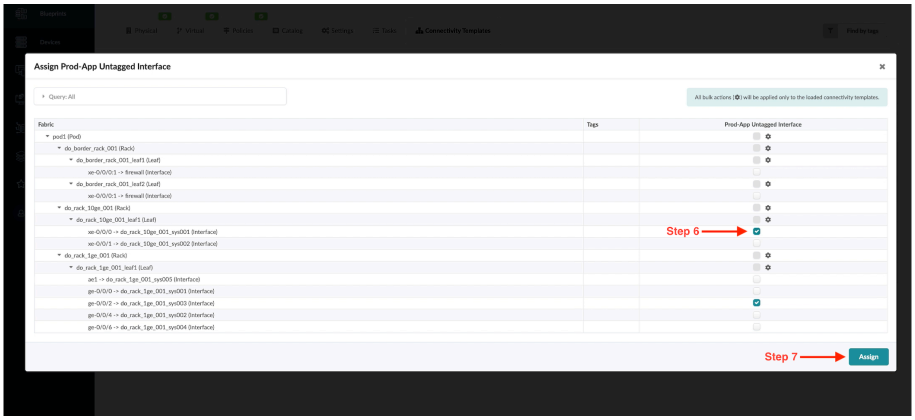


0 selected	Name	Description	Tags	Status	Actions
	Backup-App Untagged Interface			Assigned on 1 endpoint(s)	
	Backup-DB Untagged Interface			Assigned on 1 endpoint(s)	
	BGP-Peering Backup Tennant			Assigned on 2 endpoint(s)	
	BGP-Peering Production Tennant			Assigned on 2 endpoint(s)	
	Prod-App Untagged Interface			Assigned on 1 endpoint(s)	
	Prod-DB Untagged Interface			Assigned on 1 endpoint(s)	

Figure 7.34 Choosing a Connectivity Template

Step 6: Select the port that is to be assigned to the correlating VNI. See Figure 7.35.

Step 7: Click Create.


Assign Prod-App Untagged Interface
Query: All
All bulk actions (⚙) will be applied only to the loaded connectivity templates.
Step 6
Step 7
Assign

Fabric	Tags	Prod-App Untagged Interface
pod1 (Pod)		⚙
do_border_rack_001 (Rack)		⚙
do_border_rack_001_leaf1 (Leaf)		⚙
xe-0/0/0:1 -> firewall (Interface)		
do_border_rack_001_leaf2 (Leaf)		⚙
xe-0/0/0:1 -> firewall (Interface)		
do_rack_10ge_001 (Rack)		⚙
do_rack_10ge_001_leaf1 (Leaf)		⚙
xe-0/0/0 -> do_rack_10ge_001_sys001 (Interface)		✓
xe-0/0/1 -> do_rack_10ge_001_sys002 (Interface)		
do_rack_1ge_001 (Rack)		⚙
do_rack_1ge_001_leaf1 (Leaf)		⚙
ae1 -> do_rack_1ge_001_sys005 (Interface)		
ge-0/0/0 -> do_rack_1ge_001_sys001 (Interface)		
ge-0/0/2 -> do_rack_1ge_001_sys003 (Interface)		✓
ge-0/0/4 -> do_rack_1ge_001_sys002 (Interface)		
ge-0/0/6 -> do_rack_1ge_001_sys004 (Interface)		

Figure 7.35 Assigning Connectivity Template to a Switch Port

Once complete the changes are staged, ready to be committed. To view the changes redo steps 1-3 as shown in Figure 7.32 and 7.33. Figure 7.36 shows the changes applied after these steps are complete.

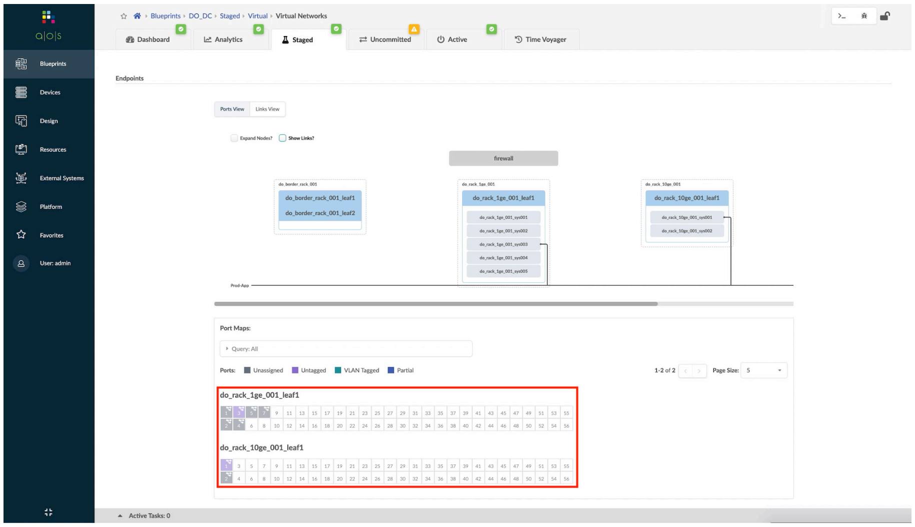


Figure 7.36 Viewing VNI Port Assignment Post-Change

Once completed, interfaces are successfully assigned to the target VNI, allowing a host to connect to other hosts within and outside the VNI.

Time Voyager

During the life cycle of a network, it is more than likely changes will be made that need to be rolled back, sometimes due to undesirable consequences. Apstra looks at the network not as individual boxes but as a single holistic unit. It has a built-in feature, called Time Voyager, that allows for a rollback of an entire fabric to a previous state.

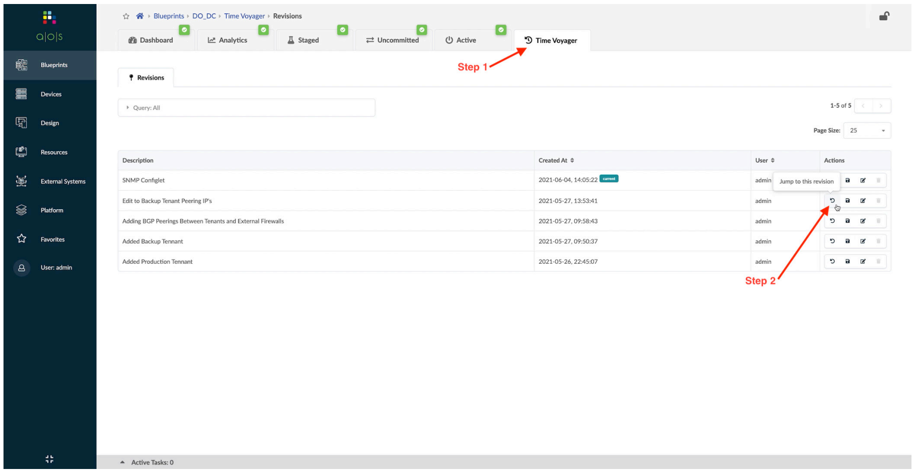
Time Voyager retains the last five blueprint commits, enabling users to jump back to any one of those previous revisions, and when the user commits a sixth time, the first revision is discarded. The user can choose to retain a particular revision indefinitely by saving it. When keeping a revision, it is not included in the five revisions that cycle out. Keep in mind that each revision requires storage space. If it's decided that a version is no longer required, it can be deleted.

When committing to a blueprint it is possible to add a revision description to help identify the changes made in that revision and this is highly recommended. These descriptions are displayed in the revision history section of the blueprint if that revision is retained.

The steps to perform a rollback of the entire fabric to a previous version are next.

Step 1: While in the Blueprint go to Time Voyager. See Figure 7.37.

Step 2: Select the version the fabric should jump back to.

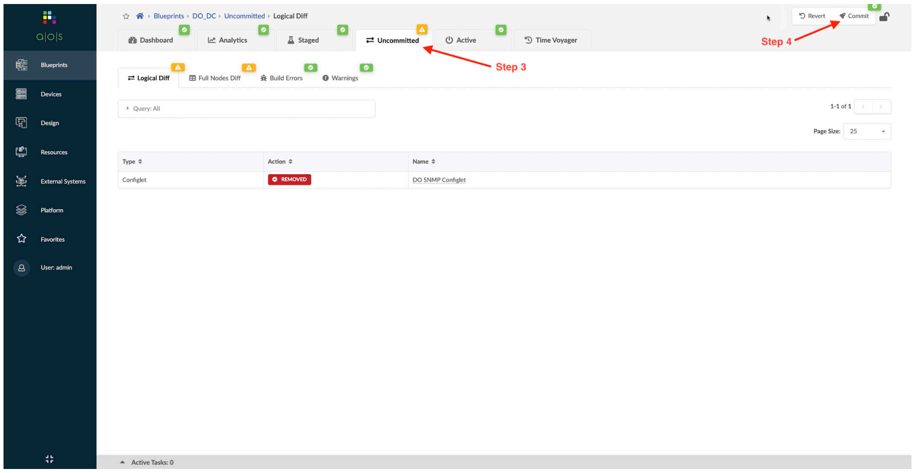


Description	Created At	User	Actions
SNMP Configlet	2021-06-04, 14:05:22 current	admin	
Edit to Backup Tenant Peering IP's	2021-05-27, 13:53:41	admin	
Adding BGP Peerings Between Tenants and External Firewalls	2021-05-27, 09:58:43	admin	
Added Backup Tennant	2021-05-27, 09:50:37	admin	
Added Production Tennant	2021-05-26, 22:45:07	admin	

Figure 7.37 *Selecting the Target Rollback Version*

Step 3: Go to Uncommitted and review changes. See Figure 7.38.

Step 4: Once happy with the changes click Commit.

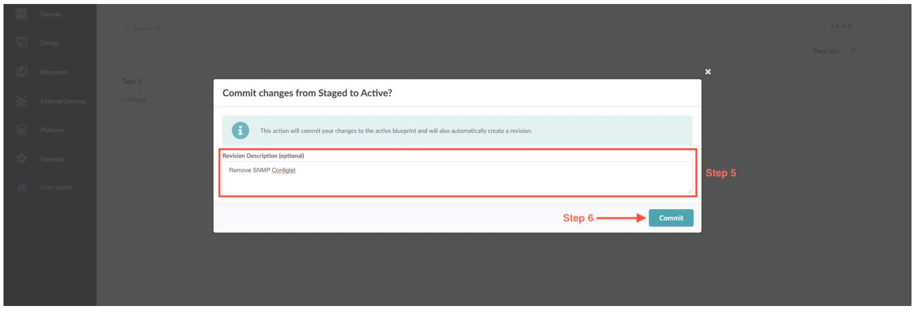


Type	Action	Name
Configlet	REMOVED	DO SNMP Configlet

Figure 7.38 *Committing the New Configuration to the Fabric*

Step 5: Provide a meaningful full description. See Figure 7.39.

Step 6: Commit the changes to roll back to a previous state.


Commit changes from Staged to Active?

This action will commit your changes to the active blueprint and will also automatically create a revision.

Revision Description (optional)

Remove SNMP Configlet

Figure 7.39 Committing the New Configuration to the Fabric

Once complete the fabric rolls back to previously selected state. Remember that this means that every device in the select DC fabric deployed will roll back with a few clicks.

Summary

The aim of this chapter was to give a brief overview of how a user can customize the DC fabric they have deployed using Configlets and property sets as well as how common Day 2 tasks can be carried out. This was by no means a comprehensive look into all the features offered by Apstra, so for more information please visit the latest Apstra documentation: https://www.juniper.net/documentation/product/us/en/apstra.

Chapter 8

API Operations

The role of a network engineer is slowly evolving and there has been much more emphasis on leveraging automation and orchestration tools to deploy and manage networks. Apstra takes on much of this burden by simplifying operations and deployment; however, you probably will want to take this further and integrate Apstra with third-party systems such as an IPAM or ticketing system.

Apstra is a completely API (Application Programming Interface) driven platform, meaning anything you can do via the GUI can be done in a programmatic manner as well. This is key to enabling integration with other tools and building complex workflows that stretch across multiple systems.

An API provides a programmatic way of interacting with a system. The purpose of API is to provide a structured framework by which different systems can communicate with each other and exchange information.

This final chapter shows you how Apstra's documentation can be accessed, how it can be read, and finally, different ways of interacting with the API.

Documentation Usage

The key to understanding the API is understanding the documentation. Apstra's API documentation is very comprehensive and takes form using a tool called *Swagger UI*. API documentation is the information that is required to successfully consume and integrate with an API. Swagger UI shows you the API endpoints available, how they are expected to be consumed, and the data model needed.

NOTE When viewing the Apstra API documentation there are two subsections: the *platform* and the *reference design* documentation. The difference between them is that the platform configuration is for interacting with non-blueprint specific API endpoints, such as resources and design elements. The reference design is for interacting with actual blueprint specific API endpoints.

Accessing API Documentation

These next steps show how Apstra's API documentation can be accessed.

Step 1: Go to Platform > Developer > Rest API Documentation. See Figure 8.1.

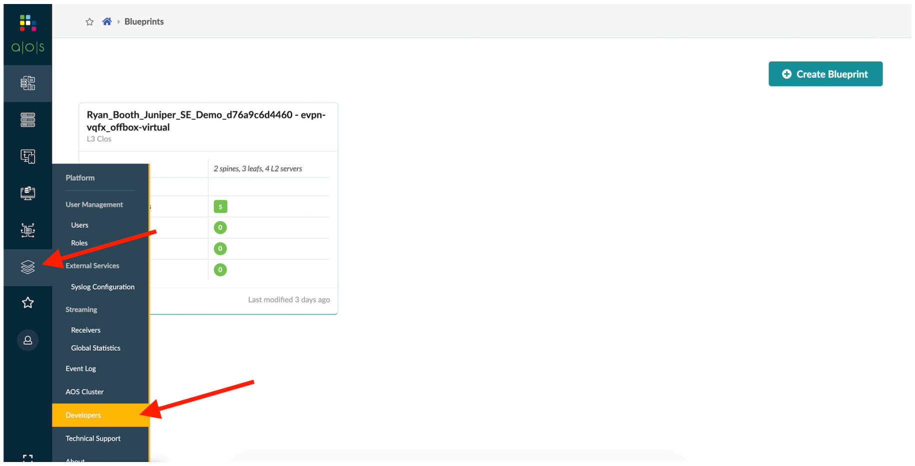


Figure 8.1 Accessing the REST API Documentation

Step 2: Select REST API Documentation. See Figure 8.2.

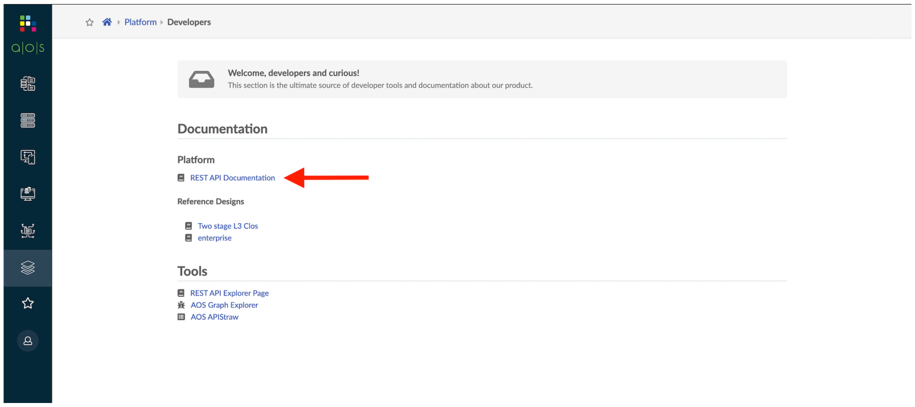


Figure 8.2 Accessing the REST API Documentation

Once selected the reader is able to view the API end points available in Apstra and understand the parameters required to interact with the API.

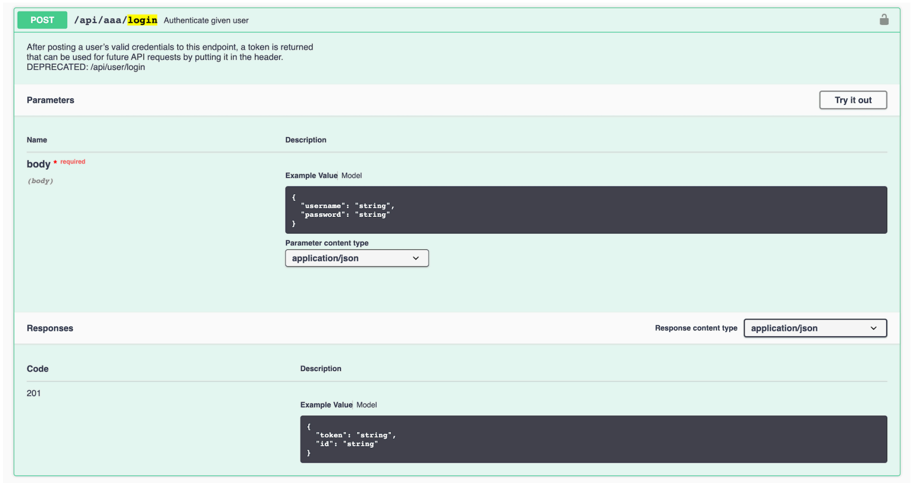
Creating an API Token

You must create an *API token* to begin using the API. In general terms, an API token is a unique identifier used by an application requesting access to a service. In this context, Apstra generates an API token for an application to use. This token is used to authenticate access. It's important to securely store the generated API token, otherwise it could be used by an unauthorized party.

Here are two separate ways to generate an API token. The first uses a tool called Postman and the second method uses the Python programming language.

First, it's important to find out which API endpoint you need to interact with, what information is needed, and what the return should look like. (The previous section showed how to access the API documentation).

The documentation for generating an API token is found under the AAA section, using the api/aaa/login route as show in Figure 8.3.



POST `/api/aaa/login` Authenticate given user

After posting a user's valid credentials to this endpoint, a token is returned
that can be used for future API requests by putting it in the header.
DEPRECATED: /api/user/login

Parameters — Try it out

Name	Description
body * required (body)	Example Value Model Parameter content type: application/json

```
{
  "username": "string",
  "password": "string"
}
```

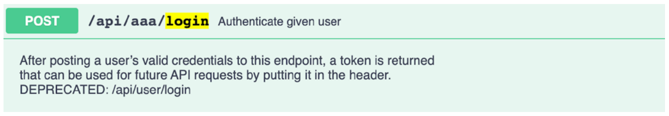
Responses — Response content type: application/json

Code	Description
201	Example Value Model

```
{
  "token": "string",
  "id": "string"
}
```

Figure 8.3 Login API Documentation Example

The first part shows the API endpoint you need to interact with to generate the token and the HTTP verb needed. See Figure 8.4. Generating an API token requires a POST to /api/aaa/login.

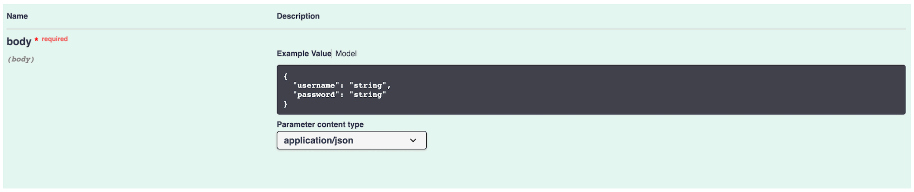


POST `/api/aaa/login` Authenticate given user

After posting a user's valid credentials to this endpoint, a token is returned
that can be used for future API requests by putting it in the header.
DEPRECATED: /api/user/login

Figure 8.4 Login API REST Endpoint

Next the required parameters needed to successfully authenticate a user. The body of the request must contain both the username and password for the user account in a JSON format. See Figure 8.5. The parameter content type declares that the body of the message should be formatted in JSON.


Name | Description

body * required
(body)

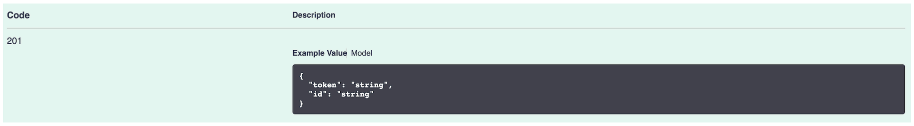
Example Value Model

```
{
  "username": "string",
  "password": "string"
}
```

Parameter content type
application/json

Figure 8.5 REST API Body Structure

The final step shows you the how a successful response is formatted (see Figure 8.6). It will return a HTTP code of 201, meaning that the request has been fulfilled, with the API token and an ID for the task. This response will be formatted in JSON.


Code | Description

201

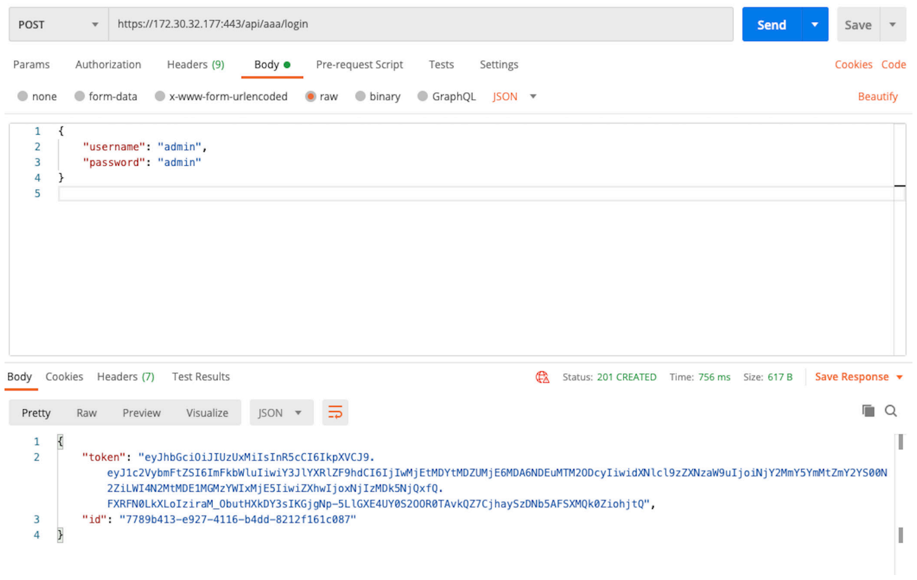
Example Value Model

```
{
  "token": "string",
  "id": "string"
}
```

Figure 8.6 Example of the API Response Structure

API Token Generation - Postman

Postman is a great tool when trying to dissect RESTful APIs because it gives a simple graphical interface to interact with API endpoints. This book does not deep dive into Postman's functionality, it just gives a quick example of how it can be used to generate an authorization token.



https://172.30.32.177:443/api/aaa/login

```
{
    "username": "admin",
    "password": "admin"
}
```

Status: 201 CREATED Time: 756 ms Size: 617 B

```
{
    "token": "eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.
        eyJ1c2VybmFtZSI6ImFkbWluIiwiY3JlYXRlZF9hdCI6IjIwMjEtMDYtMDZUMjE6MDA6NDEuMTM2ODcyIiwidXNlcl9zZXNzaW9uIjoiNjY2MmY5YmMtZmY2YS00N
        2ZiLWI4N2MtMDE1MGMzYWIxMjE5IiwiZXhwIjoxNjIzMDk5NjQxfQ.
        FXRFN0LkXLoIziraM_ObutHXkDY3sIKGjgNp-5LlGXE4UY0S2OOR0TAvkQZ7CjhaySzDNb5AFSXMQk0ZiohjtQ",
    "id": "7789b413-e927-4116-b4dd-8212f161c087"
}
```

Figure 8.7 Postman

API Token Generation - Python

An alternative to using Postman is using a programming or scripting language. Python is a powerful general-purpose programming language used in web development, data science, network automation, and so on. Python has simple, easy-to-use syntax which makes it an excellent language for this example.

An example of generating an authorization token is shown here in a simple Python script.

Script

```
import requests
URL = 'https://172.30.32.177:443/api/aaa/login'
BODY = '{"username":"admin","password":"admin"}'
Apstra_RESPONSE = requests.post(URL, data=BODY, verify=False)

print(Apstra_RESPONSE.json()['token'])

API_TOKEN = Apstra_RESPONSE.json()['token']
```

Response

```
{
"token":"eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9J1c2VybmFtZSI6ImFkbWluIiwiY3JlYXRlZF9hdCI.
J1c2VybmFtZSI6ImFkbWluIiwiY3JlYXRlZF9hdCI6IjIwMiLWI4N2MtMDE1MGMzYWIxMjE5IiwiZXhwIjoxNj.
wcn0gN-fojz7rupXNa1vUdVUmprfSPh6deLl89B6TIMHHEuzVw-BtldgYY0WRD1-NBUmxhk9EchM-eVgO2z2Ng"
"id": "7789b413-e927-4116-b4dd-8212f161c087",
}
```

Now that an authorization token has been created it can be used to interact with the Apstra API. The API is extensive, and it's recommended to read through the documentation to understand its full potential. The small script below uses the API token to retrieve all the IP pools that have been configured in Apstra.

Script

```
import requests
URL_LOGIN = 'https://172.30.32.177:443/api/aaa/login'
BODY = '{"username":"admin","password":"admin"}'
Apstra_RESPONSE = requests.post(URL_LOGIN, data=BODY, verify=False)
API_TOKEN = Apstra_RESPONSE.json()['token']

URL_IP_POOLS = 'https://172.30.32.177:443/api/resources/ip-pools'
HEADERS = {'Content-Type':'application/json', 'AUTHTOKEN':f'{API_TOKEN}'}
Apstra_RESPONSE = requests.get(URL_IP_POOLS, headers=HEADERS, verify=False)

print(Apstra_RESPONSE.json())
```

Response

```
{
    "items": [
        {
            "status": "in_use",
            "subnets": [
                {
                    "status": "pool_element_in_use",
                    "used": "14",
                    "network": "172.16.0.0/16",
                    "total": "65536",
                    "used_percentage": 0.0213623046875,
                }
            ],
            "used": "14",
            "display_name": "DO_IP_Loopback",
            "tags": [],
            "created_at": "2021-05-26T20:24:44.600979Z",
            "last_modified_at": "2021-05-26T20:24:44.600979Z",
            "used_percentage": 0.0213623046875,
            "total": "65536",
            "id": "426050aa-4b63-459d-a634-b7a2c4dcefba",
        },
        {
            "status": "not_in_use",
            "subnets": [
                {
                    "status": "pool_element_available",
                    "used": "0",
                    "network": "172.17.0.0/16",
                    "total": "65536",
                    "used_percentage": 0.0,
                }
            ],
            "used": "0",
            "display_name": "DO_IP_External",
            "tags": [],
            "created_at": "2021-05-26T20:24:57.783229Z",
            "last_modified_at": "2021-05-26T20:24:57.783229Z",
            "used_percentage": 0.0,
            "total": "65536",
            "id": "d46584c3-c4b0-474a-99a7-dc5b7248417f",
        },
        {
            "status": "in_use",
            "subnets": [
                {
                    "status": "pool_element_in_use",
                    "used": "16",
                    "network": "192.168.0.0/16",
                    "total": "65536",
                    "used_percentage": 0.0244140625,
                }
            ],
            "used": "16",
            "display_name": "DO_IP_Fabric",
            "tags": [],
            "created_at": "2021-05-26T20:24:25.452142Z",
            "last_modified_at": "2021-05-26T20:24:25.452142Z",
            "used_percentage": 0.0244140625,
            "total": "65536",
            "id": "3a7b9bbf-3a08-4788-8578-22e85bd7b87a",
        },
    ]
}
```

Summary

This chapter introduced Apstra's API, its documentation, and simple ways to interact with it. It is by no means a definitive guide to Apstra but provides a foundation that you can build on.

For users who wish to pursue this further, reading the documentation (https://www.juniper.net/documentation/product/us/en/apstra) is a must, but the Apstra UI uses these APIs to perform actions. This allows you to view the API calls using tools such as Google Chrome developer tools. It's a great way to really understand what is going on under the hood and give additional context to the API documentation.