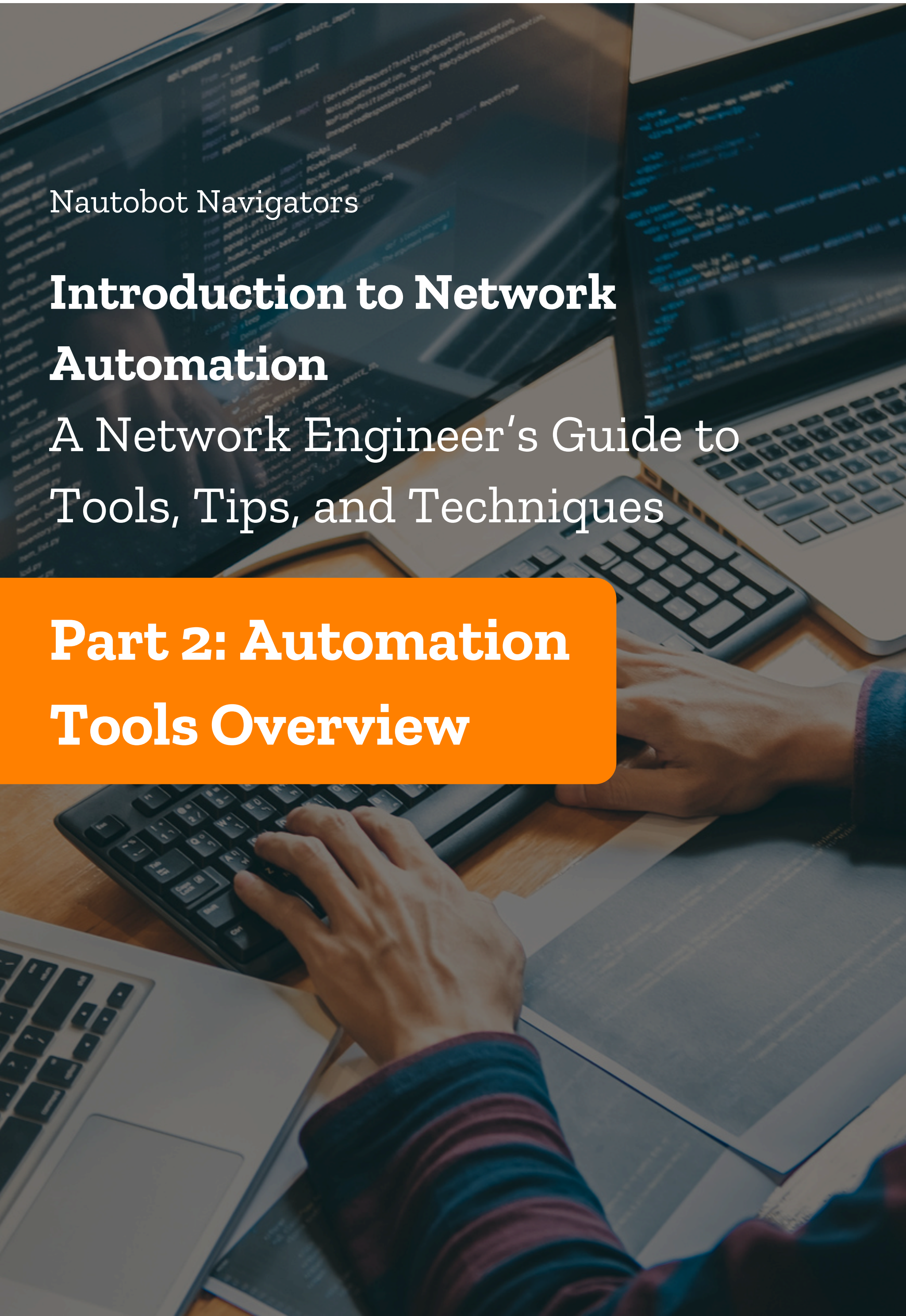


Nautobot Navigators

Introduction to Network Automation

A Network Engineer's Guide to Tools, Tips, and Techniques

Part 2: Automation Tools Overview

Introduction

Welcome back to the Introduction to Network Automation eBook series, dedicated to empowering Network Engineers with essential tools, tips, and techniques. Part 2 explores foundational technologies crucial for transitioning to Network Automation Engineering. From open source stalwarts like Python and Ansible to pivotal platforms such as Git and Nautobot, this eBook will equip you with the knowledge needed to navigate the evolving landscape of network automation tools.

Open Source Tools

When getting started, we recommend beginning to learn a few common tools and then expanding from there.

Python: While everyone has their programming language of choice, there's no disputing Python's popularity and community support. If you've never written any code before, we highly recommend starting with Python. It's easy to learn and beginner-friendly, and there are lots of resources out there. Once you get familiar with Python, you can expand your skill set by learning additional programming languages.

Ansible: Ansible is one of the most popular open source automation tools. Learn how Ansible works and use it to automate basic tasks in your network.

Git: The de facto version control system out there, it's entirely open source and relatively easy to learn. Start by familiarizing yourself with how Git works, how to save (commit) changes, how to reverse (revert) changes when mistakes are made, etc. Note that Git is not the same as GitHub or GitLab. You do NOT need a GitHub/GitLab/etc. account to use Git.

Nautobot: An open source Network Source of Truth & Network Automation platform that allows you to model your network and store network data to drive network automation with the intended state in mind.

Tools Overview

Making the transition from Network Engineer to Network Automation Engineer requires a shift in how you think of your network. But with this shift comes a different set of tools you'll start to use, which will help you as you progress into this field.

There are so many tools out there for Software Developers, but as Network Engineers tend to think differently and use different tools, it can be overwhelming to figure out which ones to start with. In this section, we will explore some of the most popular ones used by Network Automation Engineers and explain them in a way that you not only understand but also show their real-world value to a Network Engineer.

> ***Disclaimer**: Just like with everything, there is no "one-size-fits-all" tool or solution that will always work in every situation. Some of these tools will work most of the time, and some will work only in specific instances. Our goal is to outline the tools most popular today with a wide range of community or commercial support.*

Tools Overview: Python

According to the TIOBE Index, as of September 2021, Python is the second-most-popular programming language in the world and about to take over the #1 spot from C. In fact, we would be surprised if you haven't heard of Python's benefits by now or even started learning it. If you haven't, the best time to start is today.

One of the driving factors behind its popularity is its ease of use and learning curve, specifically for people without programming backgrounds. The benefits of this make it very easy to get started writing something useful pretty fast, and yet it can still be used to write complex automation.

With such popularity comes large community support. While learning Python, when you run into a question, chances are someone else has already posted the answer online, and it's a quick Google search away.

Some of the cons can be realized after using Python for a few years. For example, there are other languages that are inherently faster with certain tasks, though they're more complicated to learn. Additionally, there are certain software development "best practices" that are taken care of for you under the hood of Python but need to be learned when using other languages.

In summary, from a Network Engineer just getting into scripting and automation to seasoned senior-level engineers, Python is perfect.

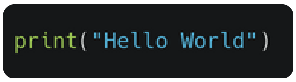
Fun fact: Did you know that some network hardware vendors have native Python support built right into their devices? For example, if it's installed and enabled, you can run the command python *from privileged exec mode on a Cisco 9k switch and load a Python prompt!*

Hello World

In the world of programming languages, there's a concept called the "Hello World" program. Essentially, it's when someone who is learning a new programming language learns just enough to print out the phrase "Hello World!" to the screen. It's seen as a good starting point while learning the basics.

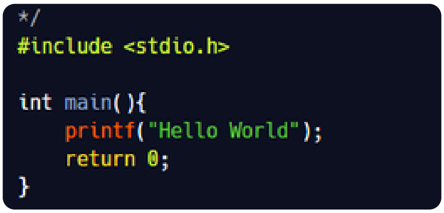
To demonstrate the simplicity of Python when compared to the other programming languages, here's a "Hello World" program written in the other 2 languages at the top of the TIOBE Index mentioned earlier, C and Java:

Python



```python
print("Hello World")
```

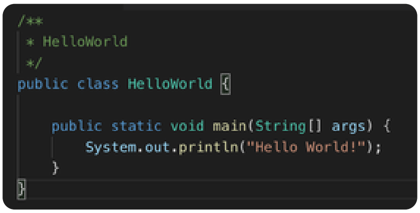
C



```c
*/
#include <stdio.h>

int main(){
    printf("Hello World");
    return 0;
}
```

Java



```java
/**
 * HelloWorld
 */
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Python is about as simple as you can get and makes it easy to get started! As mentioned before, you may not need to understand what everything in the C or Java examples is right away since Python handles most of that behind the scenes, but you will eventually want to learn what they are and why they're important.

Important: *When starting with Python, make sure you're using and learning Python 3, not Python 2.*

Tools Overview: Ansible

Ansible is an open source automation platform for managing multiple devices easily. It was acquired by Red Hat in 2015, and it remains one of the most popular open source automation tools for network automation engineers. Ansible can be used for relatively simple playbooks (scripts that you run) for a single switch, to complex fleet management systems for thousands of devices! While there are other open source tools available that are similar, Ansible is the best and most popular choice for managing network devices for a few reasons:

1. **Agentless**: Ansible connects directly to a network device, usually over SSH, but can use other methods. It does not require an "agent" or other piece of software to be preinstalled on the device. Installing an agent on a network device for management is not feasible, so this is where Ansible works well compared to similar tools like Chef or Puppet.
2. **Inventories**: Want to configure 100 switches without manually connecting to them one at a time? This is where Ansible really shines! Just provide it with a specially formatted inventory file, which includes a list of devices and a few other parameters, and Ansible will handle connecting to them all behind the scenes.
3. **Modular**: Similar to Python, with Ansible's popularity comes a wide range of modules (plugins) you can use. Some are submitted by the open source community, while others are officially supported by third-party modules (e.g., Arista EOS).
4. **Customization**: If you need Ansible to do something unique to your environment, or there is a feature not yet created by the open source community, you can write your own using… Python! That's right, Ansible runs off of Python and natively supports custom Python scripts to be imported into Ansible playbooks.
5. **Commercial Support**: Since Red Hat acquired Ansible in 2015, companies can now purchase commercial support for Ansible through Red Hat or even through third-party companies like Network to Code.

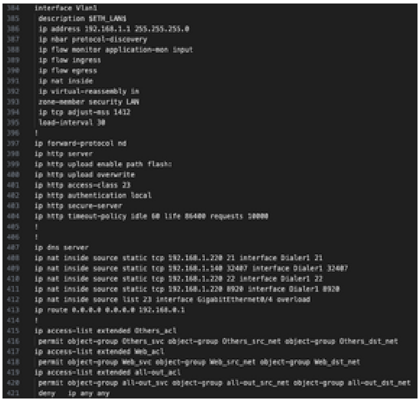
Tools Overview: Integrated Development Environment (IDE) and Text Editors

You may not hear about an IDE, but they're very important when working with automation.

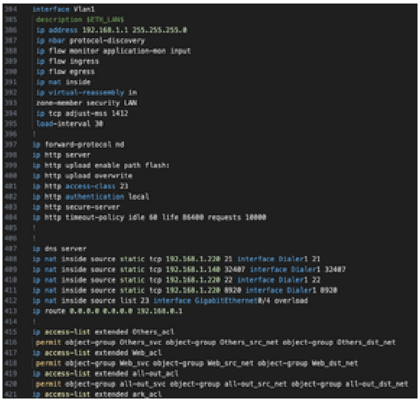
As a network engineer, your text editor of choice may often be a generic notepad-style application. This application is mostly used to write out a switch config before configuring the device by copying/pasting the text into the CLI.

When you get into automation, you'll spend much more time with scripts, configs, settings files, etc. For this reason, we highly recommend you select an IDE or text editor right away. The more you use it, the more familiar you'll become with it. Eventually, you'll never want to work without it!

One feature that is an absolute must for whichever program you choose is syntax highlighting. While different programs use different colors for syntax highlighting, they all essentially work the same. Instead of explaining what this is, look at the images below and ask yourself this question: Which one is easier to read through?



```
interface Vlan1
 description $ETH_LAN$
 ip address 192.168.1.1 255.255.255.0
 ip nbar protocol-discovery
 ip flow monitor application-mon input
 ip flow ingress
 ip flow egress
 ip nat inside
 ip virtual-reassembly in
 zone-member security LAN
 ip tcp adjust-mss 1412
 load-interval 30
!
ip forward-protocol nd
ip http server
ip http upload enable path flash:
ip http upload overwrite
ip http access-class 23
ip http authentication local
ip http secure-server
ip http timeout-policy idle 60 life 86400 requests 10000
!
!
ip dns server
ip nat inside source static tcp 192.168.1.220 21 interface Dialer1 21
ip nat inside source static tcp 192.168.1.140 32407 interface Dialer1 32407
ip nat inside source static tcp 192.168.1.220 22 interface Dialer1 22
ip nat inside source static tcp 192.168.1.220 8920 interface Dialer1 8920
ip nat inside source list 23 interface GigabitEthernet0/4 overload
ip route 0.0.0.0 0.0.0.0 192.168.0.1
!
ip access-list extended Others_acl
 permit object-group Others_svc object-group Others_src_net object-group Others_dst_net
ip access-list extended Web_acl
 permit object-group Web_svc object-group Web_src_net object-group Web_dst_net
ip access-list extended all-out_acl
 permit object-group all-out_svc object-group all-out_src_net object-group all-out_dst_net
 deny   ip any any
```



```
interface Vlan1
 description $ETH_LAN$
 ip address 192.168.1.1 255.255.255.0
 ip nbar protocol-discovery
 ip flow monitor application-mon input
 ip flow ingress
 ip flow egress
 ip nat inside
 ip virtual-reassembly in
 zone-member security LAN
 ip tcp adjust-mss 1412
 load-interval 30
!
ip forward-protocol nd
ip http server
ip http upload enable path flash:
ip http upload overwrite
ip http access-class 23
ip http authentication local
ip http secure-server
ip http timeout-policy idle 60 life 86400 requests 10000
!
!
ip dns server
ip nat inside source static tcp 192.168.1.220 21 interface Dialer1 21
ip nat inside source static tcp 192.168.1.140 32407 interface Dialer1 32407
ip nat inside source static tcp 192.168.1.220 22 interface Dialer1 22
ip nat inside source static tcp 192.168.1.220 8920 interface Dialer1 8920
ip nat inside source list 23 interface GigabitEthernet0/4 overload
ip route 0.0.0.0 0.0.0.0 192.168.0.1
!
ip access-list extended Others_acl
 permit object-group Others_svc object-group Others_src_net object-group Others_dst_net
ip access-list extended Web_acl
 permit object-group Web_svc object-group Web_src_net object-group Web_dst_net
ip access-list extended all-out_acl
 permit object-group all-out_svc object-group all-out_src_net object-group all-out_dst_net
ip access-list extended ark_acl
```

Integrated Development Environment (IDE)

An Integrated Development Environment, or IDE for short, is an application that contains many common tools for writing software (or even basic scripts) and is frequently used by software developers. IDEs have many benefits, and the popular ones have too many features to list.

Some of the downsides are also found in their strengths. They can be complex, with many settings that may make little sense when first starting out scripting. However, their benefits strongly outweigh the negatives, and we encourage you to try one out and give it some time before giving up on it right away. Don't worry about every button or feature, and focus on the basics. As you become more familiar with them, you'll start learning their features and other benefits more and more.

Two of the most popular ones available today used by Network Automation Engineers are VSCode by Microsoft and PyCharm by JetBrains. The syntax highlighting example above is from a free VSCode extension for Cisco IOS configs, though both support many other color variations and file formats.

Text Editors

There are many, many good text editors out there. Ask anyone who's been in IT long enough, and they'll not only have a favorite but a list of reasons why it's the best. The real answer is there is no "best" text editor, only the one that works best for you.

Most popular GUI-based text editors now offer some level of built-in syntax highlighting, but not all. If you're uncomfortable with starting out with an IDE right away, or if you just want something better than Notepad to use in your day-to-day activities, take a look at three of the more popular ones available right now that are free to use: **Sublime, Atom, Notepad++.**

Tools Overview: APIs

An API allows one application or system to be able to interact with a completely different application or system in a structured, predefined manner with expected inputs and outputs on both sides. Simplified, it is a way for two programs to be able to talk to each other.

An analogy would be how people talk to each other using the English language. If two people can both speak English, then they can both understand what each word means, know how to talk to someone so they understand what was said, know what to expect as a response, and what that response means. One person's response may even differ from the exact same request if it comes from someone they know (authenticated) vs. coming from someone they do not (unauthenticated).

Think of an API like this: Amy natively speaks Spanish (application 1), and Bob natively speaks French (application 2). They normally can't understand each other, but if they both agree to speak to each other in their secondary language, English (APIs), they can communicate in a limited but effective manner. In that analogy, an API is not a translator but a predefined set of rules (English) for Amy and Bob to talk to each other.

To take it a step further, some types of APIs (like REST APIs) can require another application to be authenticated before it will listen to what it has to say (process the data). In the previous analogy, it is similar to how if Amy is friends with Bob (authenticated), they may respond to each other in one way. However, if a complete stranger named Charley (unauthenticated) walked up to Amy and started saying the same thing Bob was saying, she may respond differently.

***Note**: There are multiple types of APIs, each with its own sets of rules, data formats, communication methods, etc.*

As a Network Engineer, you may not fully understand the need to use APIs, but as a Network Automation Engineer, you will begin using APIs for automation scripts to be able to interact with network devices.

Traditionally, if you want to enable an interface on a Cisco switch, you have to connect to the CLI on the switch over SSH and run these commands:

```
switch01# config t
switch01(config)# interface FastEthernet1/1
switch01(config-if)# no shutdown
switch01(config-if)# end
switch01# copy running-config startup-config
```

Simple right? Well, at least simple for humans to perform and understand. However, when you start writing scripts to do this, you'll find it's a lot harder and very unreliable to do it this way, and there are faster, more reliable, and easier ways of doing so.

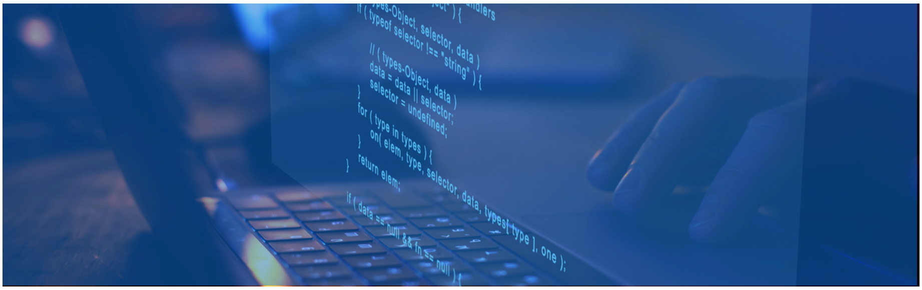
A common scenario occurs when you try to configure a setting across multiple network devices with different OSs or sometimes even different versions of the same OS. For example, if you look at the AAA configuration for Cisco IOS compared to Cisco NX-OS and then compare it again to Cisco ASA, they are all different. As an engineer, you are able to manually adjust the commands in the CLI on the fly, but in scripting, you have to account for each variation you might encounter.

This is where APIs come in. Instead of worrying about variations in each OS or how each command is different, what if you could have your script configure it using the same method and know for certain it will get configured as expected? Or if it fails, have it tell you there's an error without breaking anything? Using an API, you absolutely can!

In this example, you can use a network device's or application's built-in API to send specific data. It will then receive the request, and since it already knows what to expect, it's able to parse it out, perform any action requested, and return data back to your script in a preformatted and expected way. If you send it information in a way that it isn't expecting or is missing necessary information, it will let you know as well.

Examples of data returned can be anything, including:

- ***Was the job successful?***
- ***Command output***
- ***Configurations***
- ***Errors encountered***
- ***etc.***



Tools Overview: Git

Git is a Version Control Software or VCS for short. VCS is essentially software used to manage configuration changes to one or more files and track any changes made to them. A good VCS system will let you see what changes were made to different files, who made them, and when they were made, and will allow you to roll back one or more changes at a time.

Git is one of the most important tools a Network Engineer can learn to use. Even if you never write a single script or piece of automation, Git can still be very useful for a Network Engineer.

There are many non-automation-related use cases, including:

Config backups
Script backups
Version controlling
Auditing

Note: *Regarding auditing, it can be helpful internally or for external auditors, but may require some advanced setup to meet different auditing and compliance criteria.*

Git can be used locally without ever needing to create an account with popular online services like GitHub or Bitbucket. As a new Network Automation Engineer, you can use it to create backups of scripts you write or device configurations. These backups can be used later to undo a mistake you made or even answer the timeless question, *"When was the last change made, and what was it?"*

Note: *Git is not GitHub, just like the Linux kernel is not Ubuntu.*

Git is the most popular version control system available and is used by the majority of developers around the world. It is open source, easy to use, and, with a wide user base, has a lot of community support available. While it is natively used with CLI commands, there are various GUI applications that can make learning and using Git easier when getting started. A brief list of some popular ones are:

Sourcetree
Sublime Merge
GitKraken
Fork
GitHub Desktop

Tools Overview: Nautobot

Nautobot is a powerful open source Network Source of Truth & Network Automation platform built as a web application atop the Django Python framework with a PostgreSQL or MySQL database, offering a wide array of features that streamline and enhance network management.

Nautobot allows you to model your network and store network data to drive network automation with the intended state in mind. Here's why a network engineer might want to start using Nautobot:

Flexible Network Source of Truth for Networking

Nautobot core data models are used to define the intended state of network infrastructure enabling it as a Network Source of Truth. While a baseline set of models are provided (such as IP networks and addresses, devices and racks, circuits and cable, etc.), it is Nautobot's goal to offer maximum data model flexibility. This is enabled through features such as user-defined relationships, custom fields on any model, and data validation that permits users to codify everything from naming standards to having automated tests run before data can be populated into Nautobot.

Extensible Data Platform for Automation

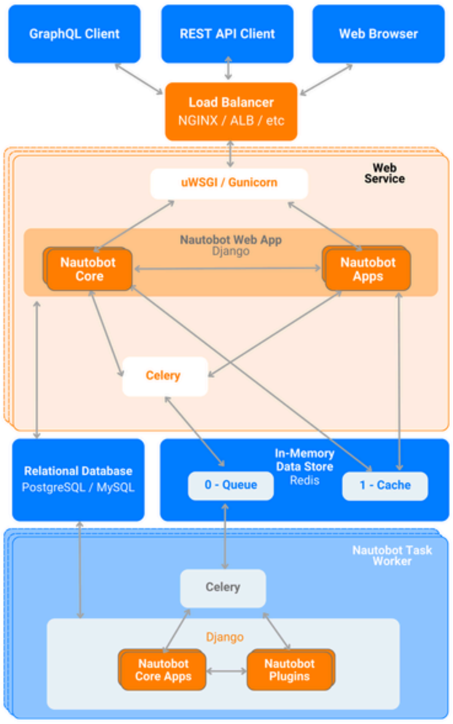
Nautobot has a rich feature set to seamlessly integrate with network automation solutions. Nautobot offers GraphQL and native Git integration along with REST APIs and webhooks. Git integration dynamically loads YAML data files as Nautobot config contexts. Nautobot also has an evolving App system that enables users to create custom models, APIs, and UI elements. The App system is also used to unify and aggregate disparate data sources using the Single Source of Truth (SSoT) app to streamline data management for network automation.

Platform for Network Automation Apps

The Nautobot App system enables users to create Network Automation Apps. Apps can be as lightweight or robust as needed based on user needs. Using Nautobot for creating custom applications saves up to 70% development time by reusing features such as authentication, permissions, webhooks, GraphQL, change logging, etc., all while having access to the data already stored in Nautobot. Some production-ready applications include:

- Golden Configuration
- Device Lifecycle
- Firewall Models
- SSoT
- ChatOps
- Circuit Maintenance
- Capacity Metrics
- Device Onboarding

The following diagram displays how data travels through Nautobot's application stack.



Chapter Summary

As you embark on your journey into network automation, remember that mastering these tools is a journey, not a destination. Each tool discussed here is a gateway to efficiency and innovation in network management. Whether you're just starting or looking to deepen your expertise, embracing these tools will undoubtedly transform how you approach network engineering. Stay curious, keep learning, and harness the power of automation to shape the future of network infrastructure!

Thank you for joining us on this journey. Stay tunes for Part 3, where we'll explore the power of Git!