John Capobianco

Derick: [00:03:18] Thanks again, John, for joining us today. I think this is the second time that I've interviewed you and the network you work on and the work you do is really fascinating. So I'm really happy to have you as a guest again. And, I think we'll start with the parliament network.

Tell us how did you end up there and what sort of dumpster fire did you walk into?

A Brand-new, Green-field Network

John: [00:03:38] First, I really appreciate the invite and for reconnecting Derek, it's great to see you again. And, these discussions I think are valuable and we get to learn from each other. And so I joined the Canadian House of Commons, the Parliament of Canada in 2013, as part of what they called the Next Generation Parliamentary Precinct Network program, or NGPPN as we call it internally. And this was a very big vision, a multi-million-dollar complete overhaul of the existing House of Commons network. So the data center and all 50 buildings that make up the House of Commons in downtown Ottawa, as well as all the connectivity to the450 remote branches or what we call the constituency offices, the riding offices, where all of the members of parliament win their elections and serve the local communities.

A new perimeter, new firewalls. We introduced wireless. We introduced more security measures like 802.1X. We applied QoS models for video flows and multimedia traffic. So a complete transformation of what, when I joined was a Nortel legacy, flat-layer-two network that didn't offer any mobility - so no wireless. There was no voice over IP. It was all POTS telephony. They had problems with just keeping the network up, offering new services and it coincided with the digitization of parliament.

So within the government body, they wanted to move to things like paperless committees, instead of having everything on paper artifacts, move to tablets and mobile devices and databases and things like that, of that nature. But they needed that underlying infrastructure to provide those services, which they didn't have. So that's where we came along, to introduce a brand new green-field network that did have to -interact with the old network as we started transitioning services from old to new. You know, I'm going to have a core and an aggregation layer and a bunch of access switches in buildings, and it's all got to connect together through fiber and copper.

We're more of an ISP in that we have the parliament, the House of Commons, a discrete group, the Senate of Canada, another discrete group, the Library of Parliament, a third group, and then all of the multimedia services, which transformed to IP-based multimedia as well during this transformation and all of the protective services. So all of the cameras and security systems - they also moved to IP during this transformation. So it was... it's quite the complex system, of all of these moving parts to offer digital services for our democracy.

Derick: Wow. So you're doing the Internet of Things before people were using that term.

Smart Ovens!

John: [00:06:23] It's funny. One of the new facilities we had was the food services , including smart stoves in the kitchens at the parliament. So it went right down to new smart stoves and 3000 wireless access points and thousands of IP cameras and recording systems, and this also had to be highly secure and zoned off from each other. The tenants have virtual routing spaces to keep them isolated and perimeters to keep traffic inside of a zone or only permitted certain ports in cross-zone traffic so it's pretty complicated stuff.

Brandon: All I can think of right now is your toasters really are on the internet. They're just not little toasters. They're full-on ovens. What does a smart oven even do? And what does it need a connection for?

John: It actually lets the chef use his tablet to download recipes from the cloud directly and program the stove directly to some online recipe, for example, or get alerts through their phone that a certain temperature has been reached, you know, "The 30 turkeys are all ready to go. And they've reached a certain temperature based on the probes and sensors."

So like a very finely-tuned instrumentation in a stove that is cloud connected. And to your point, yes, we isolate our internet of things. It's not mixed in with our common traffic that has private network access, we dump them to the perimeter and let them consume the cloud.

The stove actually was something that brought a lot of people together IT security and network engineers and architects and application folks and our cloud people, for a stove, right?

Brandon: Is the food pretty good though?

John: It's pretty good. The food's great. The food is worth it. And, they're very happy with the service and it's one of the things I believe when they, when they do tours, to show off these smart stoves and how incredible the food production is for the House of Commons.

A lesson in Canadian Civics

Brandon: [00:08:18] And one thing I wanted to jump in with for the benefit of our listeners who may be in America or any place that's not Canada. Can you give us just a brief description of Canadian government? You had mentioned Parliament, you had mentioned a few other words there, and I want to make sure we have the right context.

John: Yeah. Jesus, you're testing my civics a little bit, but since I worked there, I should have a somewhat of an understanding. So we have a parliamentary system, like Great Britain or a Commonwealth country; Canada's a Commonwealth country. And we have, house of commons where we vote members into parliament based on "geographic ridings" is what they're known as, or the constituency. And, you know, our current government, you may have heard of Justin Trudeau, he's the current prime minister. And then there's the minister of finance, the minister of defense, different ministers.

They have committees at the House of Commons to make new bills and pass new laws. Then there's also the Senate of Canada. We have a Senate similar to the American Senate, but more like the parliamentary Senate where bills get passed along and ultimately approved by the crown through the governor general. I hope I didn't get my Canadians civics too off there...

Let's Talk Automation

Derick: [00:09:21]I understand that, uh, you are a, a huge proponent of automation. I think you've written a book on it. How does automation help in the transition? That's a very complex move, it sounds like.

John: SoI don't want to give any misconceptions that in 2013, when we did this, -that it was automated in any way. We did do some plug-and-play automation where you physically assemble your stack of switches. You plug them into a certain subnet and they pick up a configuration, which was, good for one-time onboarding of switches.

So that helped us scale to the 50 buildings and the 10 or 15 floors per building for the initial build. That was about the only sense of automation we were using. But a few years later, once it was built... how do you maintain it? That's a different problem than building it and cutting over to it. How do you care and feed for a network of that size, on an ongoing basis, once you're out of the green field?So that's where the challenges of scale, the size of the network and complexity- VRFs, NetFlow, QoS, 802.1X, different VLANs, different VRFs - and maintaining consistency at scale with that complexity. The bottleneck where it hurts us is that agility, we need to offer a new service, we want to spin up a new website, we want to bring on a new zone under the network or whatever network engineering that you can think of. We weren't very agile. So those three things drove me towards automation, to help maintain consistency at scale and be more agile.

So we had this routing by rumor. An engineer drafts the change in a ticket. That ticket gets passed on to operations in a waterfall, and then operations pick it up and CLIs into X number of devices and copy-and-pastes X number of lines of code.

It just wasn't working anymore for us. The first ping on my radar about network automation was an email from April 2017 I sent to a colleague that I had found docs.ansible.com. And what can this do for us? And look at some of these tools that we have. And I don't need to learn Python. I can actually, but I can still programmatically orchestrate a series of changes.

As part of the next generation program, a phase three was to introduce a brand new data center. Once we had the campus cut over, we then had to retrofit our data centers with next-gen data center equipment and topology and IP space. And what we found was there was some things that we would like to do better on the campus, as we had learned, and some things that were necessary for us to integrate with this new data center. And I thought a bit through - partially through arrogance and partially through ignorance that this might make a good first attempt at automating a change.

Brandon: So - John, in this stage, did you have any experience with programming or development? Was this starting fresh, I think is the question?

John: So it was, and it wasn't. So I'm a little unique in that 20 years ago, my formal education was as a computer programmer analyst. But my career, my co-op placement was in client-server and networks. So I graduated with client-server network experience in the field, through my co-op with no real certifications behind it, and a diploma in programming with no real-world programmatic experience. So through my career, 20 years, I have not revisited programming and It was like coming home. It was like the circle was now complete. And here I could actually apply my programming experience and knowledge of interacting with things, programmatically to the network that I've been working at, typically through the CLI for the last 20 years.

That was really exciting to me. But at the same time, I hadn't done any real-world programming at scale or in real life for about 20 years. So it was like starting fresh. it wasn't like I'd been writing C++ programs for the last 20 years, and now suddenly I was going to do that for networks.

It was more like, I've discovered a new tool set. so that is how it connected for me. It's a very low barrier for entry. You don't have to have a computer programming background like I do necessarily, to start with tools like Ansible.

Brandon: So I'm curious what it was like for the rest of the people on your team, about how many people are there, who are doing operations and engineering?

John: Yeah, eh - probably 25, if you count architects and operators and monitoring and other engineers and the team that I work with, other network designers. Um, that is a good point - it did not come as easily or as quickly to some others and I, myself, had to learn a lot. And there was a lot of trial and error, and I've tried to help others learn from my failures. they're not really failures - what I've learned through trial and error and trying to write my own playbooks and data models.

Some of my more junior staff, who've just more recently graduated from college or university, seem to pick it up a lot faster and quicker and adapt to it - this new approach - then some of my other colleagues that have been pure network engineers for 15 or 20 years. I think because coming up - Python and programming is actually more integrated in network training and education now than it was when I learned my CCNA, for example.

Brandon: Got it. So how did this initial step in your automation journey go? you discovered docs.ansible.com. What happened?

John: I had mixed success and the difficulties and the challenges that I faced were not because Ansible is difficult or challenging. It was because of the tools that I was using when I first started to try to do this. So writing a playbook in YAML, thinking about orchestrated steps and tasks building in some fail-safe mechanisms to not push a mistake out at scale, a lot of testing, a lot of iteration. looking back, had I not been using notepad to write my YAML files and WinSCP to copy that file to a Linux host, to trial-and-error that playbook - It's

embarrassing, now that I look at it, but I, there's files like johnsthirdattempt_mightwork_version3.

It wasn't programmatic right there - I was doing it like I would in any other network change. But the good news was through a few weeks of struggle and trial and error, the playbook was successful. I made a change at scale. I touched 50 routers, a hundred virtual routers, routing tables a fine orchestrated series of tasks that were the same tasks that a human would have performed manually at the CLI except automated through this Ansible playbook, um, serially executing, SSH commands through the CLI. And it was very successful. I thought the playbook had failed. There were quite a few of us in the room during our change window when we ran the playbook in production and, uh, you know, after 45 seconds or something, it crashed.

It didn't even crash. It just stopped. And, uh, I apologize. Sorry. It didn't work. There's gotta be a bug or maybe I've missed the line of code. And the operators said, no. Look, the routes are here. It worked, the changes are all there. And we looked through our artifacts that were automatically generated - our post change validation, and we had all our routing tables and "Holy Cow, this really worked!". So we were very excited about that. but there was some trade-off like it was, it was, arduous, a lot of trial and error. I didn't quite know what I was doing. It was successful in that, it opened our eyes to a new approach and everyone, that team of 25 or 30 of us was excited and everybody thought this is great. We've actually automated something at scale. What does this mean for the future of how we do things - If I do something three times, four or five times a day, is that something I could automate now through the same tools?

Are there things that we used to log in, device by device and make the change? Can we do that a little bit differently now, but I was approached by the director of software in the software side of things, not in the IT department who had heard about the success, but at the same time was kind of teasing me, where's your version control? Where's your source control? Point me to your git repo where these files are...

How to know it's time for automation (in general)

Derick: [00:17:10] That's a great story though. Right? That's how you get started with automation. Ansible has a lot of tools. It's something you can experiment with. I'm curious about something you said in the beginning before, right before this, and it actually stuck in my head.

you said that opening tickets and passing them down to network engineers and this waterfall-like method wasn't working anymore. I'm just curious, as a network engineer or a person who runs a network engineer shop, what were some of the clues or symptoms that were screaming at you? "We gotta automate?"

John: It was mainly consistency at scale was the real driving factor for us. It shouldn't be like a cookie cutter. It should be like a pattern. A port channel is a port channel that has a certain config stanza. How can we do that every single time without making a fat-finger or "I'm on the wrong context" or "I'm in the wrong device?"

Some of the symptoms were having to repeatedly revisit the same ticket because there maybe was a mistake or there was something lost in translation along this waterfall path. The business is waiting for deliverables and we're still saying you can only ask so much of a human - right? Spend four hours on a Friday night, logging into 50 devices and putting in a hundred new lines of config and validating the state of the network as you go and capturing logs.

There's a lot involved with a multi-step, multi-change process...and how much can you ask of a human operator, to be efficient and effective and not make mistakes and not be in the wrong device and also by the way, make sure your change has the resulting intent on the network.

It's one thing on paper that it looks like it's going to work. So I'm asking the same operator to one, implement the change, which is a task in itself and now post monitor and validate and test and confirm - possibly roll back. What's the backup plan? There were a lot of human-based challenges there.

And it doesn't scale as well. Just because there might be some report that says for X number of devices, you need Y number of operators. I don't think that translates to a formula of, "OK, we have a hundred routers. I need four people." "OK, I have 200 routers - now I need six people." I don't think that scale keeps up at the human level.

When to Automate (a specific task)

Brandon: [00:19:25] Interesting. So the question on my mind from earlier was after you had that initial success, and you'd spent a few weeks to press a button and then across 50 devices, get all the routing tables - was there someone who raised their hand in the back and said, "So it took you three weeks to do this thing that takes five minutes?"

John: Yes. I'm a convert to believing in automation. When I first started this, it didn't feel like the future way of doing things to me. I could've already logged into these routers and made this change! What am I doing? What am I wasting my time, writing YAML files and templates and data models and complexity and is it worth it in the long run? So there was a tipping point where the development time starts to get smaller, as you get more comfortable with the tools and there is kind of a point where it tips and feels like, okay, this was valuable. Okay, this was worth those three weeks, six weeks of development pain, because the next solution I try to develop might only take me two weeks, followed by, five days followed by an hour, so you'll get better at it.

And the return on investment starts to increase but there are a lot of naysayers. There's a lot of, even in your own mind as you're doing it. Geez, like we could have just, I could have gotten four or five people. We could have each logged into 10 routers and typed in these commands, but I think that's missing the real value of having a source of truth and intent-based network - looking at things as code. That was what was appealing to me was that it offered more than just the magic button that we were building, that we push when we need to make a change. It's about all of the rich artifacts that come along with it and the version and source control.

Ansible @ Forward - an aside

Brandon: [00:21:08] I would love to jump into that a little bit more and inject a personal story. So at Forward, all of our infrastructure was originally set up manually. We set up servers manually. We used VMware for that. And then right around 2015, we discovered Ansible. And it was easy enough to onboard that within about a year or two, all of our infrastructure was now configuration- as- code almost a hundred percent of it. And that notion that you just said of - have a source of truth, have knowledge that lives in the code, where before it was just in someone's head, or it was really hard to extract from the result of a change to five different places that might not even be consistent anymore.

That's a real value. And when I think of implementing something that has a choice of being automated or not. One of the questions is, well, it's going to take longer sometimes to do it in the automated way, but there's an organizational value that comes and then there's a value to me later. So I've definitely written Ansible playbooks where three months later, I don't remember having written them, but I'm glad that I did because of that thing that I wouldn't be sure exactly how to do or wouldn't know that I wouldn't be able to do correctly is there. And so even if it takes a little longer upfront, there's this value-to-cost trade off that you can still make.

And often, if you care about the knowledge and you care about having it persist and you care about scaling out that process to more people and being able to delegate a task by saying, "Hey, here's a playbook that should do what we need." I feel like there's a lot of value there.

John: Yeah, and on the value add, to tie it back to the discussion that kind of came to me about the director saying where's your repo, where's your source of truth. Where's your version control. And I said I don't have any of that. I've got a file called John's playbook.yaml that works, but I don't have the development process. That's what he mentioned to me was the organizational value. Think about how much more valuable that playbook would be if it was expressed as code in a git repository that other developers can copy and refactor and clone and work with, or to see the version history.

So we moved from more - this magic button approach to truly - let's look at the whole infrastructure's code. You had mentioned, it took about a year and a half. We found the same trajectory. So in 2017 we started with one playbook with one specific orchestrated task and about a year ago -we reached this milestone of we've defined all of our campus infrastructure wide area infrastructure, data center infrastructure.

It's all reflected as code now. Not these tactical one-time playbooks, which are still very valuable and we still have, and we still run frequently, but we've moved up the stack to abstract the entire configuration into data models and templates which has been very effective as an approach for us.

Derick: I think we talked a little bit when we were preparing that there were a couple or a few different software engineering concepts that when applied to a network practice can fundamentally transform it for the better.

Let's talk Git

Derick: [00:23:50] And so let's start with, git in version source control. Why don't you tell us a little bit about that.

John: Okay. So that's, to me, that's been the linchpin that's made my automation journey successful right out of the gate. If you're new, you've likely heard of GitHub. But there's also get, so the two shouldn't be conflated. One is a repository, a public repository where you store your git repositories.

And the other is a version source control system, git, which you're going to install in your windows environment, which comes pre-installed in every Linux environment, because it was the version and source control system that Linus used when he was building the Linux project. So it goes hand in hand with Linux.

So what this means for infrastructure as code - is I could have a repository called, you know, "The production core, " which contains a data model. So a YAML file, and a template, a Jinja2 file and the necessary Ansible playbook or Python code or variables or host files. All of those things that make up a git repository are tracked through version control and source control.

The main branch in git should represent your source of truth, your intent, and it should be well-validated, well-tested code so the current routing table or the intended expression of "I need eight VRFs", and here's how they're going to be configured that should be in main, where you're going to run your playbooks from. Now, if I need to make a change, git lets you make a branch, which is a full copy of main, but it's self-contained for development purposes.

I add my static route. I change my VRF name. I do my network engineering stuff in my branch. Test it, validate it. And then that gets merged back into main.

Production + the Lab

Brandon: [00:25:42] That was a nice overview of git and how you like to use it. One question though, which is network-specific - do you have a network test lab and you run your playbooks for your private branch, the playbook you're trying to develop or modify?

John: It's a good question. So yes, I have a dedicated lab environment where I do my development work first, and this development work is typically making sure that I get the correct CLI stanzas or the correct JSON, if I'm working with an API where my API call works and I'm getting a 200 back and I go through all that development process in a lab environment. Now we're moving towards a CICD which might be a bit early in this conversation to introduce, but that pipeline can release to different environments meaning: I can release a bunch of JSON to a prod-test environment, and depending on the results of that release step, then move on to release it to a production environment.

Brandon: So by prod-test, you mean you've got a smaller chunk, a smaller set of devices that match those on the production network. They're configured using the same scripts or

playbooks. and so in every way, they're like production, but they're smaller as opposed to having one production network and doing a stage deployment where you test on one or two, and then you roll it out to everyone.

John: Yes, that's our approach - is a smaller slice. Depending on the action, we actually have a physical lab as well as a prod test. So the lab environment, prod-test environment, prod environment. So for doing, let's say an iOS upgrade, something that affects the hardware kernel - we'll do that in the physical lab first. If I'm doing a routing topology change to something a little more logical, we'll do that in the prod-test world. There's also new virtual environments that we're exploring. where in a cloud, I could have a representation of my network and my traffic flows, which is ephemeral - I don't need physical metal to do this in. I can do it in a cloud space and do a similar release to that cloud space release to my prod-test, depending on my instrumentation and the test results I get back then move on to the production release.

Cultural Effects

Derick: [00:27:54] So it sounds like going to GitHub and version control - this is something that significantly impacted your whole process and you said when you posted your first Ansible playbook, that there were ripples throughout the organization. I'm very interested in the cultural angle of this, so can you expand on that a little bit?

John: So the ripple effect happened, uh, you know, that small close 25-30 knit group that handles the network enterprise architecture and operations and design - that group, everyone in the group. They knew that the way they were doing things was going to change. even if they didn't understand all of it, they at least understood enough that the new tools were here, a new way of doing things was here, and they all saw the advantages of doing things that new way. And they had some learning to do - either about Ansible or programming in general. Some of them were faced with their first if statement or for loop and these are highly accomplished, CCNP CCIE level engineers.

I don't want to say every one of them, but some of them are certainly intimidated by an if statement a for-loop like I understand I can make a BGP neighbor come up and exchange routes with the public internet, but I don't know how to write an if statement. So some of them have ground-up primary learning to do, whereas others embraced this or in their own time, they've done a little bit of coding or whatnot.

Seeing Results

John: [00:29:15] IT security was definitely interested because of our standards that could be pushed out. Especially once we moved to templates, that we know that every physical interface in the campus has been configured from a templated approach. And we can pass those templates along the config stanzas for GI101, right? That the native VLAN 99 has been set on every trunk interface on the campus, for example, or that every interface is correctly configured for 802.1X. There's a bunch of things that security, from an enterprise corporate standards position, they could take a deep breath and sigh of relief now, to know that the

network is configured to approve templates. And that we've enforced it right to every interface. Like we have tens of thousands of interfaces that we're dealing with. How do you know that one interface in the boardroom isn't publicly open, or on the wrong V LAN or, offering the wrong service?

It's hard to do as a human, device by device, port by port. So security was interested. The project management team was also excited in that we started to turn around projects much more rapidly, right? Yeah, no problem. I need a few days to develop the playbook. We'll run the playbook next change window. And now your service is ready to be consumed. And they couldn't believe it! These things used to take weeks to turn around, sometimes months, and you've turned it around in a weak development cycle now. So there were many ripples throughout the organization, absolutely. And it was because we started working programmatically, git offers so much rich metadata.

When I get a P1 on the network and I, they still come to me, like I still deal with priority one incidents. My light bulb is what changed? What has changed in the state of this was working five minutes ago or last night or whatever, to just suddenly "it's not working, the traffic's not flowing or I don't have an SPF neighbor." What has changed? What's the Delta in configuration from working to not working. Yes. There's cable breaks and things that don't aren't necessarily related to configs or bugs, but 80 20 rule, it's likely a configuration change that damaged the state of the network. I can go into my git repository and look at my git history and see, Oh, I see network engineer 2 did a commit that added this route. Engineer 3 did a git pull request that was approved that introduced the new VRF. So the artifacts are much more easy than the old way of, doing diffs between two show runs or combing over tickets and the ticketing system, or following your way back up the waterfall to see where the breakdown was. Now it's all right at your fingertips in code and in git history and information. Right.

Avoiding Issues

Brandon: [00:31:53] And are you finding that more change windows are succeeding the first time because you've eliminated some of the potential for error here or divergent ways of doing the same thing?

John: Yes, absolutely. Our, our consistency. We're approaching more nines. Because we are making less mistakes. There's less human error. The other thing is I'm going to use a software term here, but linting the ability for your IDE your development environment VS code or - VS code is the one I use there's many of them, but it's not notepad plus it's not TextPad anymore. The IDE will lint over your YAML and highlight with a red squiggly line, right? The space is off on line 22. this is not a valid well-formed file. So I'm catching problems in development. It's not once it reaches the wire and it's been pushed out and I go, oops, I got that wrong. The development process is weeding out so many mistakes right at the development time. So the agility is much better. Instead of here's 12 attachments for 12 routers that you need to add a VLAN on for this new service - do a git pull and run the playbook from main or master. So, you know, a three-line instruction set for operations to follow versus 12 notepad files full of iOS, configurations, or whatever, right?

A Use-Case Story: "Knifing the Committee Room"

Derick: [00:33:10] You have a really great story, an automation success story where you had some requirements around wireless and security. Can you go into that a little bit?

John: Yeah, one of the functions of parliament is to have sessions and committees. Some of these have to be private; there's sensitive information, state secrets, corporate secrets, intellectual property, whatever the bill is that's being discussed.

And sometimes we have to do what internally, known as a knife, the committee room, meaning physically cut the fibers. We don't physically cut them every time, but we cut off the light to the building switch that handles that room to make sure none of the network ports in the room are active. They don't let people bring their phones in, for example, there's other security measures around certain committees. But, we've also just added 500 wireless access points in the building. And, and as you know, wireless wants to self-heal. So what we had to do was iteratively disable 30 or 25 or 15 wireless access points at a time and work with the RCMP, the Royal Canadian mounted police, the FBI of Canada, to sweep the room, looking for wireless signals.

And once we found the mix of access points that needed to be disabled, someone made a list of those interfaces and then on-demand or on-schedule, go and shut all those wireless access point interfaces. Now, if there's an on-demand meeting and they need them disabled to start doing their business, but they have to wait for ITto disable the 50 interfaces, that's not good, right? The business is truly, literally waiting yeah. On IT at this point. So now through this automation, that is quite simple to do through an API call and through an Ansible playbook to go and find those interfaces and disable them automatically either at a scheduled time or on demand. And now it's seconds.

Brandon: I really like this story cause it's very concrete and it's something, you know you're going to have to do pretty often. So you might as well make it something that isn't a source of pain. That's a button press here or there.

John: Yeah, it was, um, it's been quite a capability for us and it's, very sensitive information and it's very private and we have to ensure the fidelity of those meetings, And even doing it the manual way, it's not just about the speed of how quickly someone can go find those ports.

Did they get every port? Did they, you know, did they miss one? Did they get the wrong port? Did they leave wireless turned on? We cannot have that. So there's some stringent business requirements that automation has helped us guarantee the fidelity of those changes.

Derick: Yeah, this is a great example of how better network tooling benefits organizations outside of the network. Your security guys have to be very happy with this.

John: Security is very happy with it. The business is very happy with it. We in IT are very happy with it. And, in terms of automation this might sound a little, I don't mean

mischievous, but if you can find a Trojan horse in your environment. Find that one success story that operations just cannot live without. Give them that playbook that they can run automatically on-demand to go get them every ARP table on the network. Their eyes will start to open up and they can't live without these functions. And eventually they'll start coming to you for more.

Listen, you've automated the ARP table. Can you get me the MAC address table? Okay, great. Can we actually configure things this way? Okay. Even better. Can we move this into a pipeline now? You know, if you make it appealing, if you build it, they will come. But you can find very safe, easy wins. And, particularly if you have APIs, an HTTP get is known as a safe RESTful operation.

It's safe. You cannot damage the network or the state of the network in any way. But that's a rich source of data that you can start to transform into reports or just the raw JSON or interact programmatically with that data. The instrumentation that's being given to us now is so much better than SNMP polling and syslog and SNMP traps, you know, about the three only things we've had for the last 30 years to monitor the health and state of the network. It's all rapidly changing and evolving to a new programmatic approach. Right.

Talking About APIs and CLIs

Derick: [00:37:12] So let's talk a little bit about APIs. You have some opinions about them and they're pretty interesting. So, why don't you just dive into that?

John: One thing about, um, that's come up recently on Twitter and it's a bad penny. It comes up all of the time is the CLI is an API

Derick: A bad penny?

John: Yeah, bad penny. I don't know. It's just a term where you reach into your pocket and you get the bad penny and you put it back in your pocket and you try to come up with a different penny or a different coin in your pocket and you get that same bad penny again.

I feel pretty strongly about this more because I don't envy people trying to learn this being a little bit misled when they're told that the CLI is just another API or the API is just another CLI. They're not the same thing in my opinion. And I'm not saying you can't automate against a CLI - a command line interface.

That's what I've been doing. The Cisco platforms I've been having up until this new iteration of the Nexus and the Catalyst 9000, they actually offer a RESTConf API, but my, older platforms, my 4500s and 6500s, they don't have that, they have a command line interface.

So yes, my Ansible playbook is automatically SSH and running serially, a bunch of commands, That is automation, but it does not mean that it's an API or that it's RESTful or that I can get status codes back from those commands or do things declaratively through an HTTP post. So here's a good example: I've moved to templated configs, and they work 90% of the time. But if you think about a CLI operation, just pretend I'm NTP. I have an NTP server, 192.168.1.1. I want to remove that and add a different NTP server. So either I programmatically have to

write a second task in my automation to no that out or leave a followup step of, you know, when the playbook is done, log in and no out that old line.

When we shift to RESTful APIs and I'm using a JSON stanza - key pairs and lists and the JSON file as a body of a post to an API.

I don't have to worry about 'no'ing out the old config. It takes it, declaratively, through REST, and replaces the full config with my intent. So it's a lot easier to do intent-based source-of-truth infrastructure-as-code, in my opinion, with a true API, as opposed to, you know, we've made it work with SSH and CLI, but I don't think the two should be conflated.

I'd like to know what your opinion is.I know I'm a little strong and people have heard me say the CLI is dead. It's dying for sure. But I'd like to know what you guys think.

Declarative vs Imperative

Brandon: [00:40:00] So I've definitely been bitten by this one in the past. What you're talking about is the difference between declarative and imperative is let's execute a sequence of steps. We got some if-then, we've got some logical changes that we might need to implement . Ansible, to use one example, enables you to write logic. It's not very easy to follow compared to a programming language, but it gets the job done. It gives you the functionality you need.

Compare that to declarative as more of a model -based or more intent-based kind of system where you say, here's the state of the world that I want. Make it so. I want these three VLANs to be on my switch versus I want to add this VLAN, which means you didn't take into account the fact that there could be another VLAN, which means you need to 'no' out, like you were saying, the other ones that are there, and that's a source of confusion because if your network automation doesn't entirely describe the state of all the config you want, then you've got to reason about what it was to get it to what it should be. And so if you don't remove the stuff that you had, that's now going to affect the new stuff, then you can, you can have some caveats that you discover where the change you expected doesn't have the behavior that you wanted.

And so often with Ansible or any automation, you want to do checks at the end to make sure that even if you implemented, to the best of your knowledge, the change in exactly the way that you wanted and it actually added the lines of config, that the overall service is actually doing what you want. And this is something that forward networks does here, right? With network verification. We're not going to jump into it too much on this podcast, but this notion of after I've made a set of changes, have I really configured the network in a way that's meeting the end-to-end service goals, or has the whole service really worked?

And maybe I added one thing I needed, but I didn't remove the one thing that's now for some ports causing their behavior to not match my intent.

The Value of a Network Engineer - and how it's shifting

John: [00:41:40] I like that.You mentioned network verification in particular because in my opinion, I have solved configuration management. I've gotten off the hamster wheel of adding VLANs, adding VRFs, and that tedium has gone. It's all through pull requests and automation and playbooks. So I've handled config management. I have an intent and it, and I apply it and that is easy. You're not hiring me to know the stanzas to be able to assemble an OSPF neighbor. You're hiring me to confirm that you have the right neighbors and you're forwarding the right traffic through the right zones, through the right interfaces, or not forwarding if it's a secure zone, and we can move up the stack now because configuration management is easy and we've dealt with it. We can actually focus our energy on higher value. Initiatives for the business and we can start to focus on the fidelity of our flows and our traffic and getting right to instrumentation because we're truly dealing with it as code.

So I'll give you an example. We have Cisco Prime infrastructure,it's an appliance-based thing that can do SNMP polling and syslogging and configuration management, various things. I've used it since Cisco works and then it became LMS LAN Management Server and now it's Cisco Prime infrastructure. And I've never used the API until this year. And I feel foolish now that I'm using it because the data there is so rich and so easy to get that I can write an API playbook through Ansible and it gives me back 75 pieces of information about each wireless access point. And that's, there's 3000 of them on my campus.

And I can do this every hour, every four hours, every 15 minutes on demand, whatever. But because it's not just raw show unstructured data, I have key pair values and lists that I can inspect with Python or with Ansible or with Java script or whatever. I see something I can ingest the Jason into. I can look at fields like CA PWAP Tunnel, colon down. So I do a for loop and look for that key pair value of being down. And now I know of my 3000 access points. I don't know, 75 of them don't have a CAP WAP tunnel or, even client thresholds, right, those 75 pieces of information per access point. So whatever I want to know about wireless as a state, as a service, I'm doing that all through pure API calls now, right?

And, um, because of that git integration, the version control, I can look, report to report and see very easily that this last CSV file had a hundred APs that were down. And the new run of the report has 50 APS that are down and here they are. And here's the difference between yesterday's run and today's run.

A self-healing wireless network?

John: [00:44:20] Now I haven't done this yet, but what it leads my thinking is to a self-healing type network. What if I could, and I can do this easily through the key pair value of the upstream connected interface and the upstream switch. What's the first thing a human would do, right? They would "shut, no" shut the port.

So now I can programmatically, say, do a second pass, and if the CAPWAP tunnel field is down, issue the "shut, no shut" command on the interface. And now I'm not saying that, that could be dangerous and there's corner cases, and I have to do a lot of thinking about the AI approach there, but in theory, you could build through APIs and basic if-then-else logic a self-healing wireless infrastructure.

Brandon: So what's, what's partially interesting to me here is you talked about how automation first was about data gathering and then making changes, right? You found tons of value in just getting the data and your higher value is not just, is it up? It's... is the network connected? Is it fault-tolerant? Is it secure? These are the basic questions that network operators and engineers get paid to answer because they're critical to whether the network is working. But now you're talking about getting easier answers or better answers to some of these because of automation. And then moving on to the next step of actually taking interventions and automating processes, which tells me you're pretty far down the path of automation.

You're getting closer to that top of the pyramid in Maslow's hierarchy of needs of self-actualization where everything is great.

Focusing on the process, not the outcome

Derick: [00:45:42] I've been thinking about - you're talking this whole time and -there's been a lot of times in my career where I made a lot of changes by hand and, you know, it seemed like a miracle in the end when everything was working. So we were like,and that was success. It's pinging, it's working, we're done, close up, it's four in the morning, we go home. But what you're talking about is you're less focused on a particular outcome and more you're focused on the process that produces better outcomes. It's all about the process more than it is any particular outcome.

John: To be successful, you are going to have to think about your processes. And it's very important to include things like code reviews and testing and releasing to non-production environments. The process has enabled me to get cultural and corporate adoption. I struggled swimming upstream for about a year, just wondering, like why won't they just do it this way? You know, like it's automated, it's easy. It's code. Why can't everyone just start doing it this way?

Brandon: I see the light. Why doesn't everyone else see the light?

John: Yeah, right, how do you actually integrate something into a corporate environment where there's existing processes. People are existing or comfortable doing things a certain way.

And this is a radical shift and humans naturally, They resist change. They don't want to do this new thing is a knee jerk reaction you have to face now. Yes, to me, it's still important to understand what that static route is and am I configuring it properly?

And what's the state change? Am I getting a valid route? Am I getting a valid neighbor? But that is all still important, the complexity of it has been abstracted, especially once you have a working template for a config stanza. And it's human readable, right? In a Y I'm adding a key pair value. You know, static route and next hop is a couple of values. And my template does the complex work of turning that into a valid Juniper command or Cisco command or Arista command. So I don't have to worry about that as much anymore. And I can focus on, I believe higher value.

How do we make a mechanism where we can start to work like software developers have been working for 30 years where we're building and releasing and we're testing code, ~~um,~~ immaterial of the metal or the router or the switch or the load balancer, right? The API helps you abstract that ultimate delivery. And you're really focusing on the code.

Closing Out

Derick: [00:48:07] I have to say we're actually out of time. I feel like there's a whole other half of this conversation that has to take place. You should come back. we'd need to have you on here again, actually. I'm putting you on the spot.

John: I appreciate the time and I hope I didn't take up too much of the time. I really just want other people to start. Just try it. Just try to do it this way. If you can figure out how to configure a router to forward traffic and it works like to Derek's point, you get the ping, that's harder than writing an if statement or a for-loop. It's so much more accessible than you might believe it to be.

Use these discussions as inspiration and go to try to write a hello world play or a hello world Python file. Just try it. It's so much easier than you think it is.

Derick: We have a question we like to ask every guest and that is, who would you like to see on this podcast?

Brandon: Setting the bar high.

Derick: Yeah. Well, what would you like to ask him, if you could ask him a question.

John: How does he feel about transforming. Humanity transforming the world. How does he really feel about his impact retrospectively, now that I can watch Netflix on my phone, how does he feel about bringing the world from analog to digital as a civilization transformation? That's what I'd like to know how he feels about. Does he bear that on his shoulders? Does he feel good about it? Does he feel like a superhero? I'd like to know.

Derick: That's a good question. The internet is his baby in a way and it's gotta be strange watching that baby turn into a crazy, huge mutant, right? What things did he not see coming? That's a good question.

John: Well, I appreciate it again. Thanks for inviting me, and I hope this was a valuable conversation. Now I'll come back anytime, and I'll just reach out. I would love to continue the discussion. I feel like we've kind of skimmed the surface. We dove in a little deep in certain spots, but there's just so much to talk about, right.

Brandon: Definitely. And I like to take notes on kind of the messages that each guest has to get the TLDR quickly. And so it sounds like in the Canadian parliament, automation has helped you to achieve consistency at scale, to stop revisiting the same tickets, to get changes done faster, to have a source of truth, but it took some time to get there.

And the first step that you recommend is to try, just try.

John: That's a great summary, yeah.

Brandon: Alright. Thanks again for joining.