



From CLI to "Agentic AI", the Arc of Networking Bends Toward Automation... right?

Frank Seesink



<https://frank.seesink.com>

Good morning. I admit the title is a little click-bait-y in nature, but I will explain this along the way.

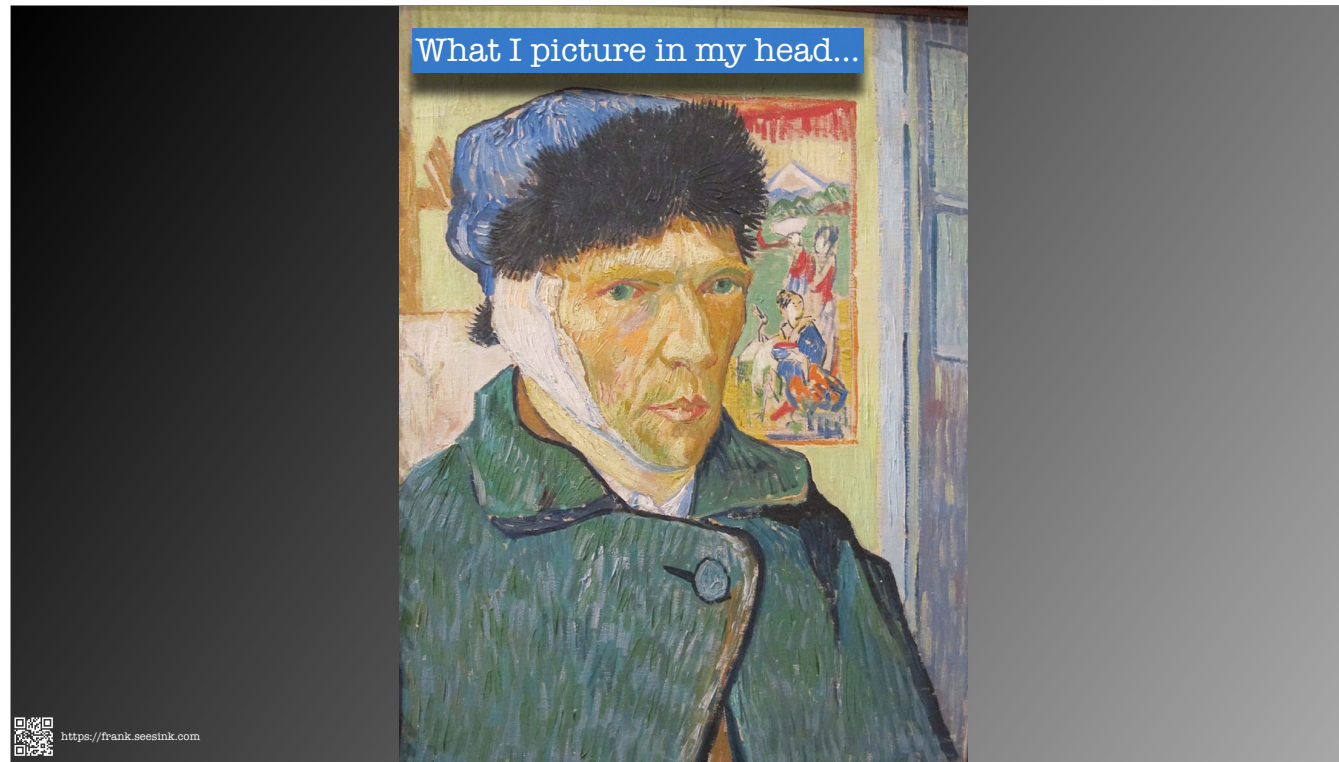
We have 1 hour for this session, and I would like to reserve as much time as possible for questions and discussion. So I will try to tear through this slide deck.

Please note these slides are available in various file formats at the QR code in the upper left, and we will provide the link as well in the chat. So here we go...

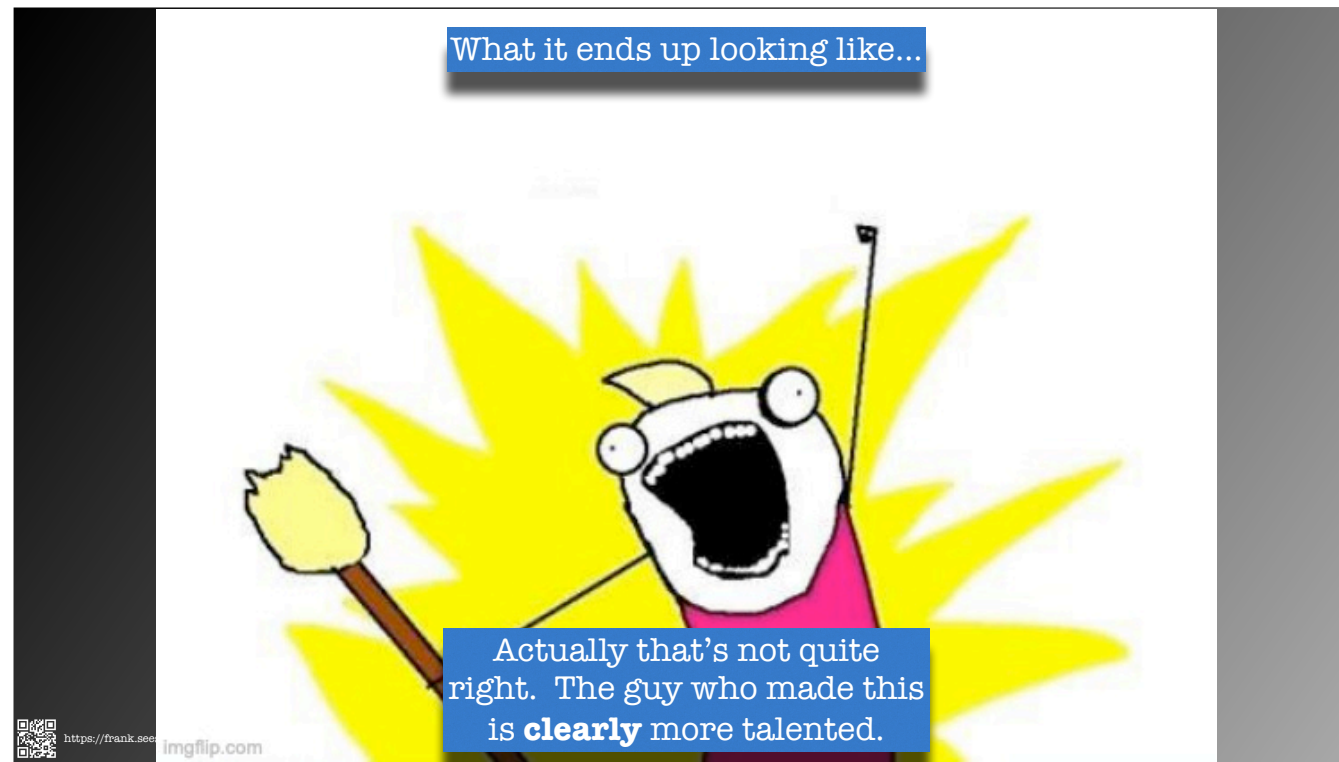
First, a message from
our sponsor...



<https://frank.seesink.com>



When I work on presentations, this is what I picture in my head.



Unfortunately, THIS is what things typically end up looking like.

Just managing expectations.

Disclaimer



<https://frank.seesink.com>

Quick disclaimer.

All opinions here are my own and do not represent nor necessarily reflect those of my employer. That said, should you find any of this information useful, I would still like to give credit to my employer as I continue to hone my skills and knowledge there.

Of course, any and all mistakes/errors/etc. are my own. I would like to blame "AI", but I cannot. :-)

Who am I?

Frank Seesink

- Senior Network Engineer, UNC Chapel Hill
- Part of network DevOps group
- Involved in network automation for years
- Love languages, both human & computer
- Programming since I was 12 years old
- Formally B.S. in Computer Science with all coursework for an M.S. in C.S.
- JOAT - databases, OSes, networking,...



<https://frank.seesink.com>

I only mention my background to give context.

Networking as a whole is made up a very diverse group of people. Many come to networking from non-technical backgrounds. And of those who came from a C.S. or engineering background, many did so specifically BECAUSE they did NOT want to program. And the topic of this session is network automation, which many will associate with programming.

So I just want to say I understand.

Who am I?

Frank Seesink

- 20+ years at state REN/Internet2(I2) Connector
- 6+ years at a large, public university/I2 Member
- I2 NTAC Network Automation SIG chair since its creation in 2017 (technically)
- Automating things most of my life

Push me,
Dave



<https://frank.seesink.com>

NTAC = Network Technical Advisory Committee

One of my mantras: “If you want me to do something once, I will do it. If you want me to do something 10 times, I will program the computer the first time and then IT can do the other 9.”

Tell the story of Dave with his request for a “Push me, Dave” button to do his job for him at the international shipping firm I worked at in the summer during my undergraduate years.

As the topic of this session is automation, some folks may be worried that automation is going to take their jobs. I don’t believe this to be the case with networking. It’s 2025, and if anything, network engineers are needed more now than ever. Automation, if anything, will simply help free us up to be able to do what we should be doing.

Key Takeaways



<https://frank.seesink.com>

While working on my slide deck, it occurred to me that I was completely missing the absolutely key bits I really wanted to convey. So before I delve deeper, let me cover these quickly.

Key Takeaways

Culture Beats Tech



<https://frank.seesink.com>

If you remember nothing else from this session, remember this. Success or failure in developing a network automation strategy relies far more on the culture you cultivate than on any tool chain you decide to use. Nothing kills faster than a bad culture.

This is akin to how in business in general you have expressions such as “People don’t quit jobs. People quit managers.”

If you are in a managerial role and looking to bring in network automation, it matters greatly how you handle things. Ideally you want to create win-win situations. You want to create a “safe space” for your workers. You want your staff engaged and involved, ideally taking ownership of any automation projects, where they feel invested and know that you have their back. This means having them help with the entire process, including helping select the tools to be used.

Play to Your Strengths



<https://frank.seesink.com>

On this topic, whenever someone asks things like “What tool or programming language should I use?”, the answer, as it always is in tech, is “it depends.”

Are you a shop with a group of folks experienced in Perl? Then maybe you should consider using that.

Don’t have a deep bench in any particular language or tool chain? Great! This opens the door for your team to explore more and learn together. The key thing is do not forget to play to your strengths. It can make all the difference.



This one applies to both those in managerial roles and those doing the automation work.

There is a great difference between operational work and doing things like programming or putting together automation projects. The former involves a lot of mental context switching (e.g., add this VLAN, fix that ACL). The latter involves what are called “makers hours.”

When you program, it takes time to get into the flow of things. On average it can take 15–20 minutes to get in the right head space to do work. Every time you are interrupted from that workflow, you lose another 15–20 minutes easy. You should also be blocking out time in 4 or 8 hour increments to focus on this kind of deep work.

To those doing automation/development work, control your space to avoid any unnecessary interruptions. Get out of chat/collab applications. Disable notifications. Turn off your cell and/or set your work phone to DND.

To those in managerial positions over such workers, your job is to FIERCELY protect them from being interrupted, both by others as well as yourself. Failure to do so will have serious consequences on their ability to meet goals and deadlines. I cannot overstate this.

Key Takeaways

It's a journey,
not a destination



<https://frank.seesink.com>

If there is one word you are likely to hear a good bit when it comes to network automation, it is the word “journey.” People will refer to their journey in network automation. And while it becomes a bit tired to hear after awhile, it is nonetheless true.

For those just starting out, it can feel like an insurmountable subject. Trust me, it is not. Everyone who is involved in network automation, whether for one year or ten, will tell you that it is a process. And no matter how much you know, there is always more to learn. The trick is to find someone who is ahead of you on this journey, and to try as best you can to learn from their mistakes to shorten your own journey.

But do understand that some things will simply take time. For example, if you have never programmed before, you cannot expect to level up at the same pace as someone who has for years. But be patient. It will come.

You Cannot Automate What You Do Not Know



<https://frank.seesink.com>

Automation does not save you from having to do your proper due diligence in first documenting your network, or doing things like fleshing out your “Source of Truth” (more on this later).

That is, before you can automate, you need to know your network. If you think automation is going to magically organize your network for you, you are going to be disappointed.



Finally, for those just starting out, when it comes to how to get started in network automation, this is the answer.

A few years ago while preparing to do a presentation at Internet2's Technology Exchange, I asked various folks involved in network automation what their advice would be to someone that was new and looking to get started. And the one overwhelming response was "Just do SOMETHING, ANYTHING." That is, do not try to figure out everything up front. Do not try to boil the ocean. Just start with something small. It can be absolutely anything. But START.

It's like the old joke: How do you eat an elephant? One bite at a time.

History of Networking (abridged)



<https://frank.seesink.com>

Now in order to know where we are going, we need to know where we have been.

“Those who do not learn from history are bound to repeat it.”

“History does not repeat itself, but it does rhyme.”

RFC1925

Story, Time... Type, Done...

- In the beginning...
- Command Line Interfaces (CLIs)
- text configuration files (e.g., Cisco IOS), or...
- binary config files (e.g., Bay Networks)



<https://frank.seesink.com>

It is 2025 and this is still true today. Not much has changed in 30+ years.

Text Files...

- Advantages
 - Easy to edit and backup
 - Copy/paste to deploy new devices
- Disadvantages
 - Easy to make mistakes / typos
 - Depending on NOS, commands go active immediately



<https://frank.seesink.com>

Cisco IOS famously is such that the moment you hit enter while in configuration mode, the command you type goes active. This compared to, for example, JunOS, which uses the commit/confirm approach where you can type a series of commands, but only once you commit those changes do they go live. The impact this could have when performing certain actions like modifying ACLs could not be overstated.

To this day, some automation systems will perform full config file loads vs. modifying individual lines in a config for this reason.

Network Management

- ICMP (ping, traceroute)
- syslog
- SNMP (“Simple” Network Management Protocol)
- AAA (e.g., RADIUS, TACACS+)



<https://frank.seesink.com>

Also during this time the general notion that we needed some way to perform network management was going on. The most common (and still used) standards/protocols were ICMP (ping, traceroute), syslogs for textual logging information sent from network equipment, as well as SNMP traps, and SNMP.

Originally intended to be used to actually manage network equipment, even the networking world has said “Only use SNMP for monitoring” for at least 2 decades now due to the potential security implications. Yet even today, most network monitoring tools rely on these protocols and ICMP.

Config Backups

- RANCID (Really Awesome New Cisco config Differ)
- Oxidized (RANCID replacement written in Ruby)
- Newer tools in this class include
 - Kiwi CatTools (now part of SolarWinds)
 - rConfig
 - Unimus



<https://frank.seesink.com>

So what were the early attempts at automation? The first was likely performing configuration backups.

Scripting

- Shell scripts (e.g., SH, BASH)
- Perl
- Expect (extension of Tcl scripting language)
- Python
- Oh, and “screen scraping”

“Artisinal coding”

All of these relied on TELNET/SSH and the NOS CLI.



<https://frank.seesink.com>

Beyond basic config backups, the next thing likely involved some form of basic scripting.

This in itself was a challenge due to all the variance in TELNET and SSH implementations.

SDN/OpenFlow (circa 2010-)

- Software Defined Networking
 - Separation of data/forwarding plane and the control plane
 - Distributed data plane vs. centralized control plane
 - OpenFlow (Nick McKeown, Martin Casado)



<https://frank.seesink.com>

While SDN had been researched earlier, it was OpenFlow which really brought this to the forefront.

Notable implementations include those done by Google.

SDN/OpenFlow (circa 2010-)

- Controllers
 - Faucet, ONOS, OpenDaylight, OpenSwitch, Ryu
 - Nicira (eventually bought by VMware (NSX))
- White box networking
 - Cumulus Networks (NVidia), IP Infusion OcNOS, PicOS, SONiC (Software for Open Networking in the Cloud)
- ONIE (Open Network Install Environment)



<https://frank.seesink.com>

This was also the time of “white box networking”. Think how folks buy “industry standard” PCs with Intel or AMD chips in them, on which they run their OS of choice, now applied to networking gear. The dream was that this would bring prices down and provide more freedom to move between vendors. It did not exactly turn out that way.

But it still moved the industry forward. For example, one good thing that came out of this was ONIE. ONIE is to networking what the BIOS or UEFI chip is to PCs. It provides a standard boot process for network equipment, allowing you to run different NOSes on the same hardware.

DevOps



<https://frank.seesink.com>

Time to define some terms...

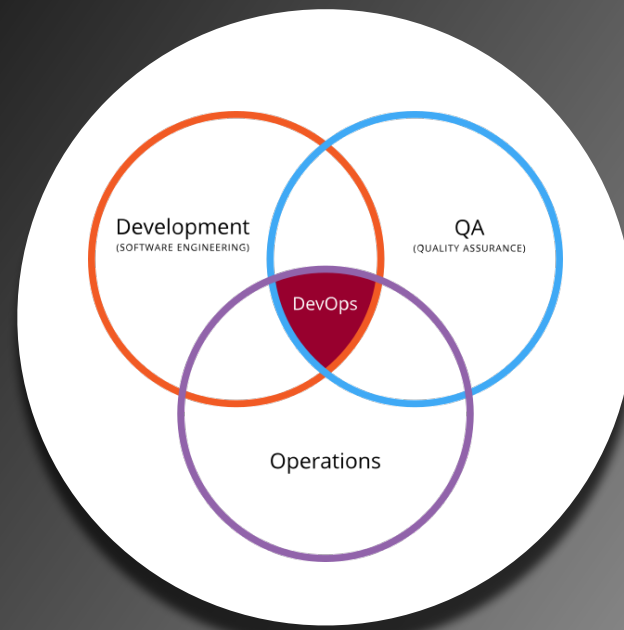
What is this DevOps of which you speak?

“DevOps (a clipped compound of "development" and "operations") is a software engineering practice that aims at unifying software development (Dev) and software operation (Ops).”

Source: <https://en.wikipedia.org/wiki/DevOps>



<https://frank.seesink.com>



NetDevOps



<https://frank.seesink.com>

NetDevOps

Combines DevOps principles with Network Operations (NetOps) to bring automation, programmability, and collaborative practices to network infrastructure management



<https://frank.seesink.com>

In Plain English?

The love child between systems/network
administrators and programmers



<https://frank.seesink.com>

System Configuration Management Tools

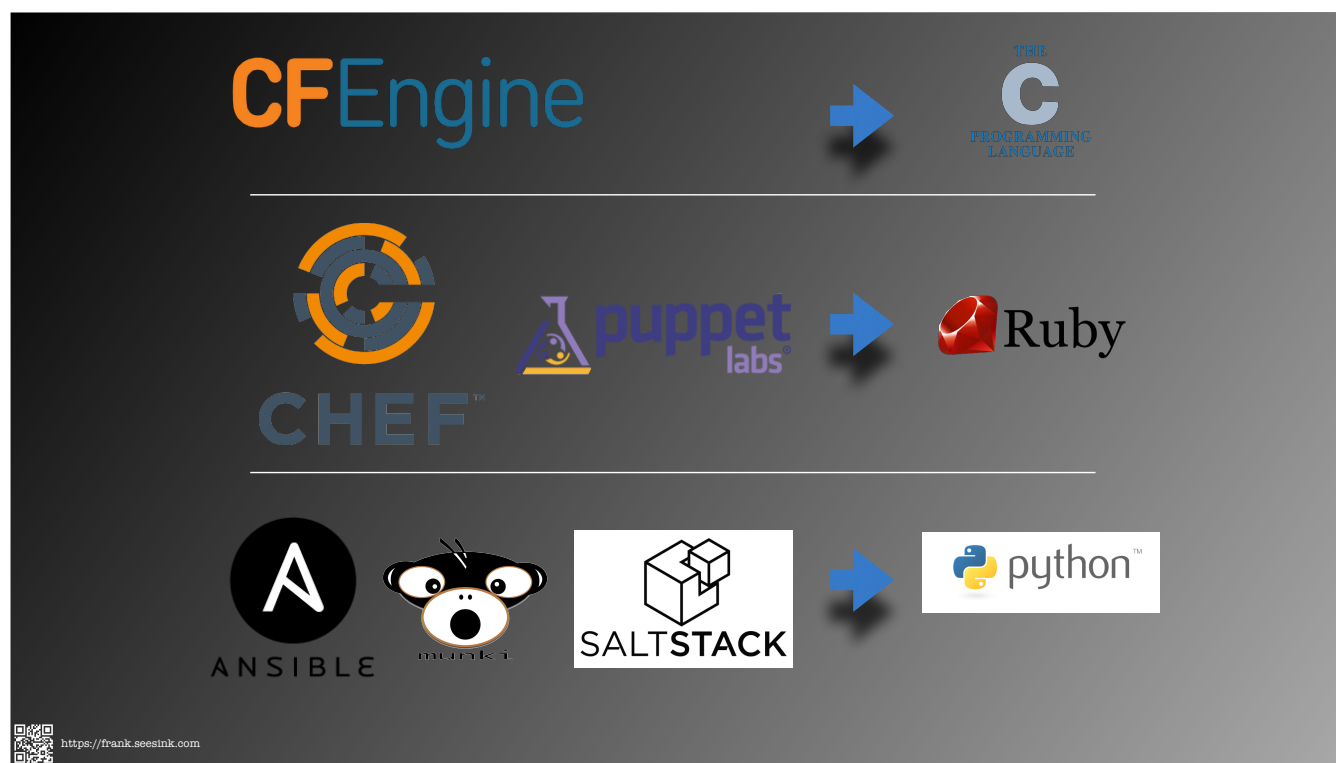


<https://frank.seesink.com>

To understand the next phase in network automation history, you need to understand a little bit about systems configuration management tools.



Many tools existed for doing sysadmin work. These included the following...



Now while not relevant directly to network automation, if you look at the languages used to develop these various tools, you might notice a pattern around which ones ended up being used more in network automation.

ANSIBLE



<https://frank.seesink.com>

Ansible

The name "Ansible" references a fictional instantaneous hyperspace communication system (as featured in Orson Scott Card's **Ender's Game** (1985),[9][10] and originally conceived by Ursula K. Le Guin for her novel Rocannon's World (1966)).[11]

Source: [https://en.wikipedia.org/wiki/Ansible_\(software\)](https://en.wikipedia.org/wiki/Ansible_(software))



<https://frank.seesink.com>

Ansible

- In 2016, Ansible, a tool originally intended for systems automation, began being used for network automation.
- With the release of Ansible 2.2, things really took off.



<https://frank.seesink.com>

So Why Ansible?



<https://frank.seesink.com>

The following was taken from a slide deck I worked on back in 2019. So SIX years ago already this was the state of things.

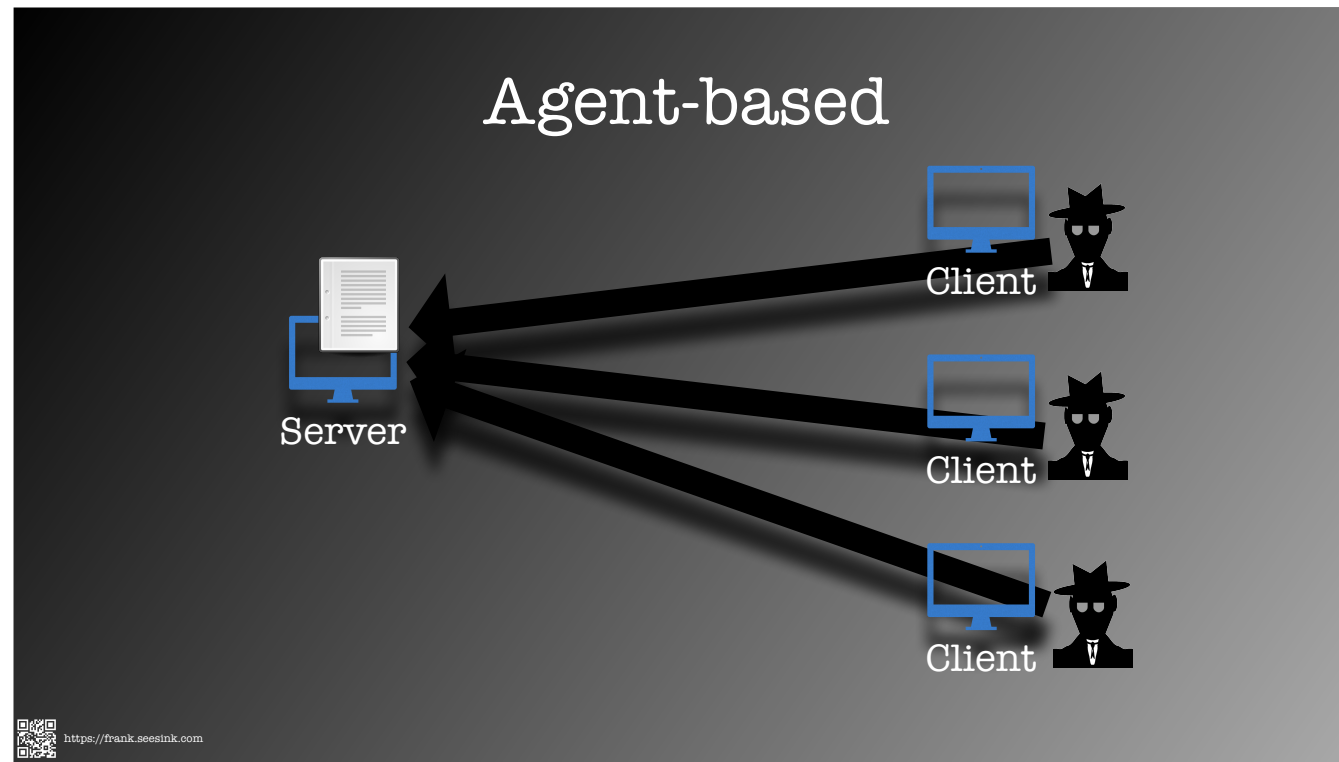
Agent-based vs. Agent-less*

- CFEngine
- Chef
- Munki
- Puppet
- SaltStack

- Ansible



<https://frank.seesink.com>



Terms:

Server == Puppet Master, Salt Master, etc.

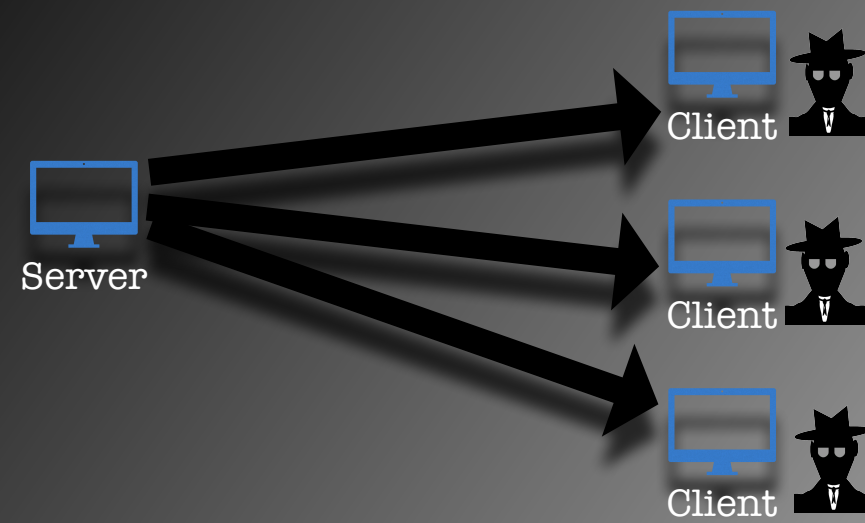
Client == Puppet Agent, Salt Minion, etc.

Configuration files == (Puppet) catalog, Salt States (SLS), etc.

Also have terms like grains, pillars, etc. for Salt, for example.

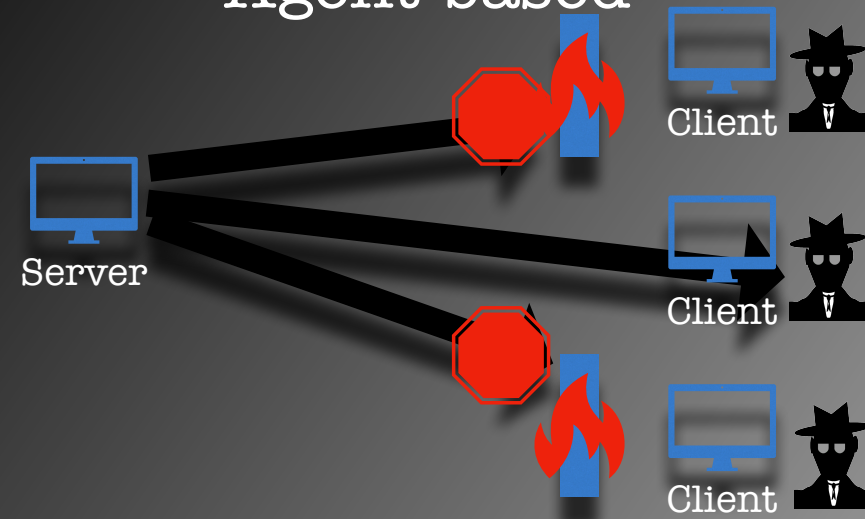
Typically agents check-in every so often—default for Puppet is every 15 minutes, for Munki is once every 4 hours—to make sure they are up-to-date.

Agent-less

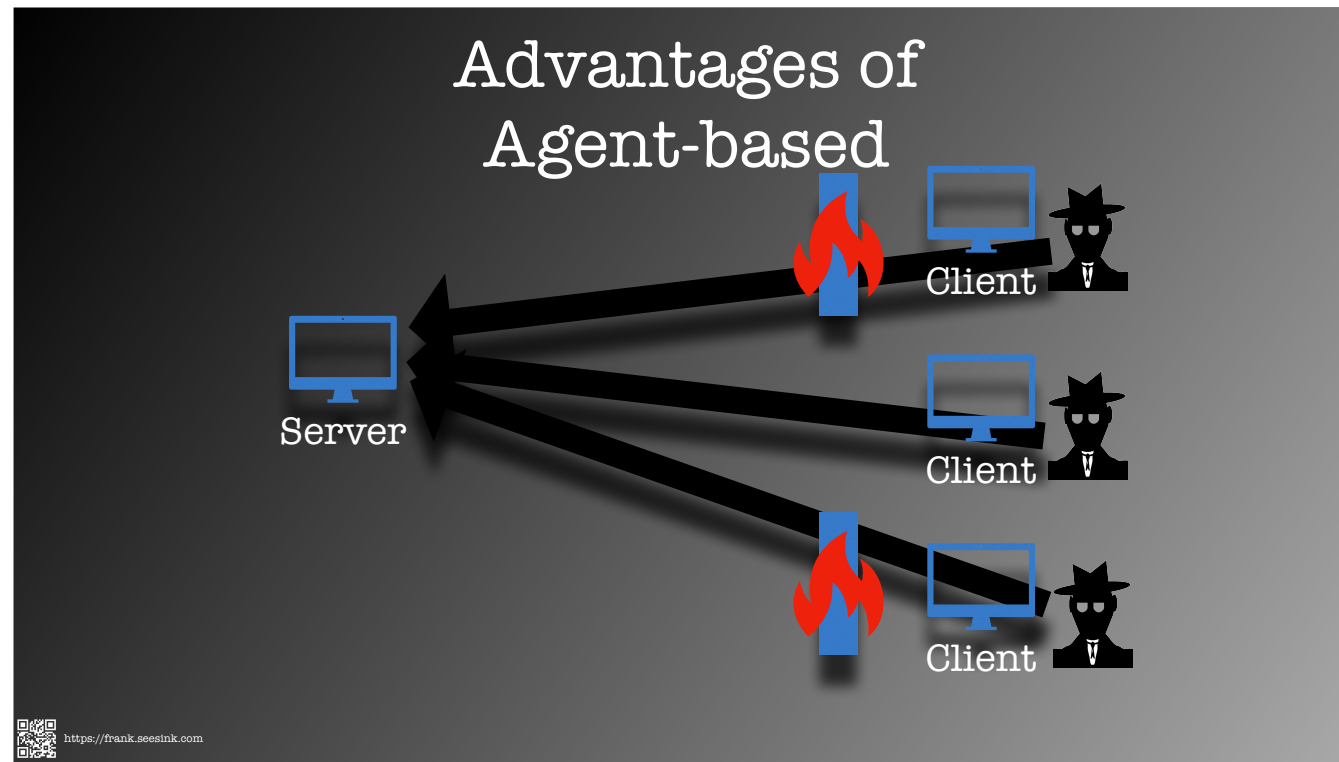


<https://frank.seesink.com>

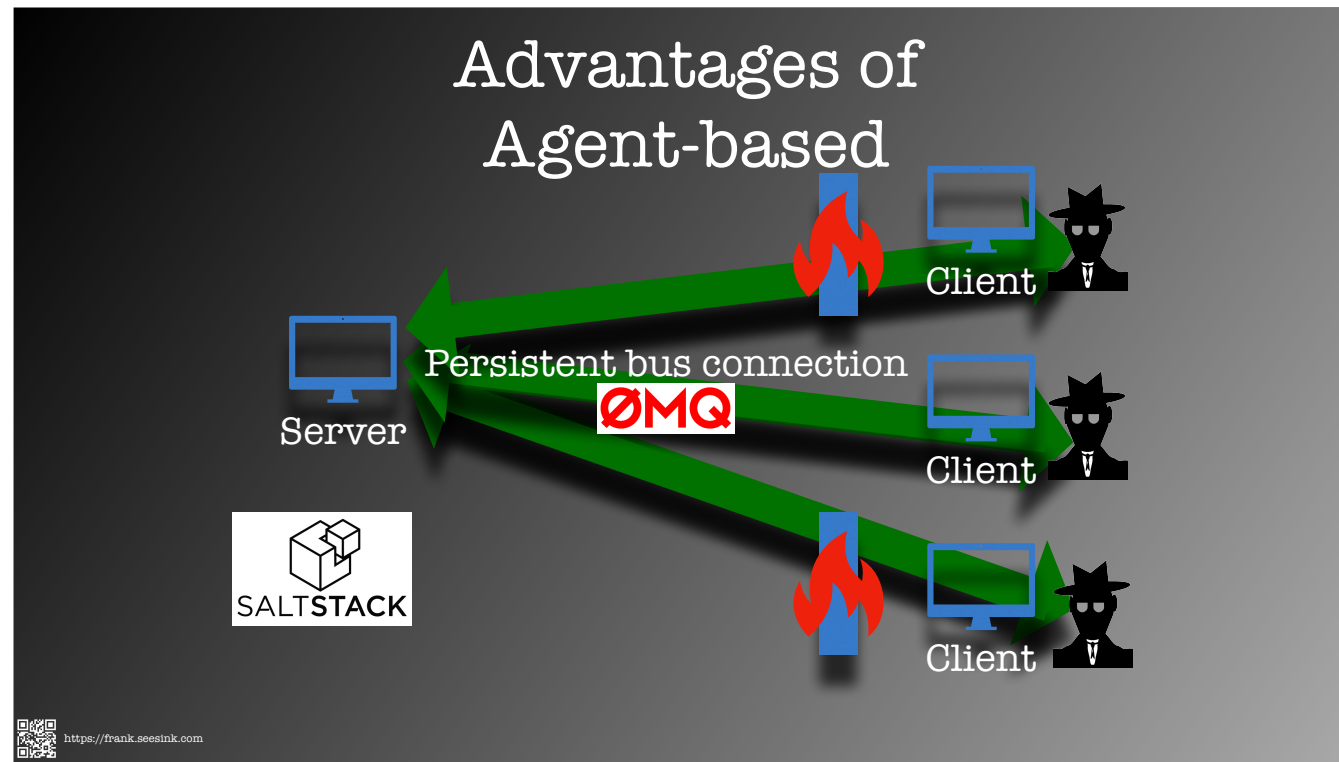
Advantages of Agent-based



<https://frank.seesink.com>



Typically agents check in, thus coming out through any firewalls vs. the server trying to come in. Of course, in a tightly regulated environment with proxy servers, etc., this may require additional work, but often things “just work.”



Salt Stack is different from other agent-based configuration management tools in that it creates a persistent connection back to the minions. This allows for immediate execution of commands. For example, you get a call that some of your users are experiencing issues getting to Google. With Salt, you could tell all of your minions to ping Google's servers and to report back. This gives you insight from across your network (and also gives you a kind of botnet of your very own!).

Advantages of Agent-less



SSH

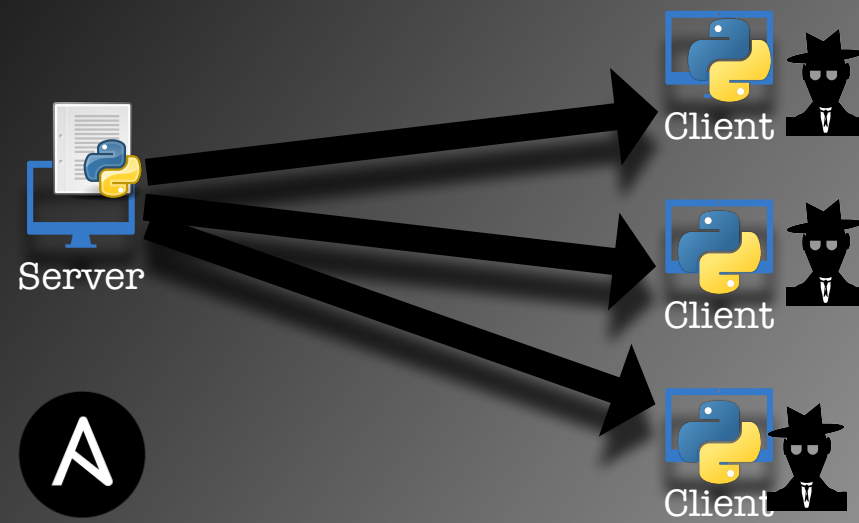


ANSIBLE



<https://frank.seesink.com>

Agent-less*

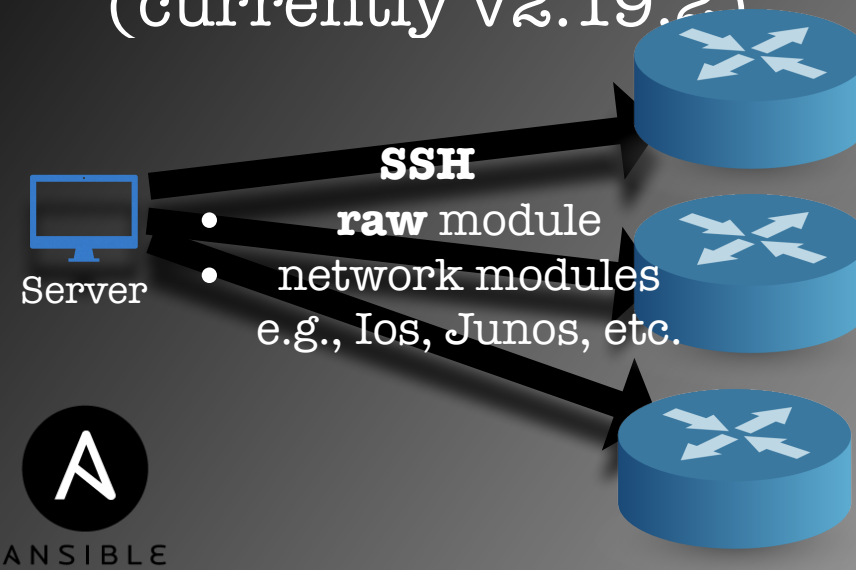


* for clients which support Python,
agent script sent through SSH
tunnel to run on far end



<https://frank.seesink.com>

Ansible 2.x (currently v2.19.2)



Network Modules (Fall 2017)

- A10
- ACI (Cisco)
- Aireos (Cisco)
- Aos
- Aruba
- Asa (Cisco)
- Avi
- Bigswitch
- Citrix
- Cloudengine
- Cloudvision (Arista)
- Cumulus
- Dellos10
- Dellos6
- Dellos9
- Eos (Arista)
- F5
- Fortios
- Illumos
- Interface
- Ios (Cisco)
- Iosxr (Cisco)
- Junos
- **Layer2**
- **Layer3**
- Lenovo
- Netconf
- Netscaler
- Netvisor
- Nuage
- Nxos (Cisco)
- Ordnance
- Ovs
- Panos
- **Protocol**
- Radware
- **Routing**
- Sros
- **System**
- Vynos



<https://frank.seesink.com>

Source: http://docs.ansible.com/ansible/latest/list_of_network_modules.html

Network Modules (Fall 2019)

- A10
- Aci
- Aireos
- Aruba
- Asa
- Avi
- Bigswitch
- Check_Point
- Cli
- Cloudengine
- Cloudvision
- Cnos
- Cumulus
- Dellos10



- Dellos6
- Dellos9
- Edgeos
- Edgeswitch
- Enos
- Eos
- Eric_Eccli
- Exos
- F5
- Files
- Fortianalyzer
- Fortimanager
- Fortios
- Frr



- Ftd
- Icx
- Illumos
- Ingate
- Interface
- Ios
- Iosxr
- Ironware
- Itential
- Junos
- **Layer2**
- **Layer3**



<https://frank.seesink.com>

Source: https://docs.ansible.com/ansible/latest/modules/list_of_network_modules.html

Network Modules (Fall 2019 cont.)

- Meraki
- Netact
- Netconf
- Netscaler
- Netvisor
- Nos
- Nso
- Nuage
- Nxos
- Onyx
- Opx
- Ordnance
- Ovs
- Panos

- **Protocol**
- Radware
- Restconf
- Routeros
- **Routing**
- Skydive
- Slxos
- Sros
- **System**
- Voss

- Vyos

65



<https://frank.seesink.com>

Source: https://docs.ansible.com/ansible/latest/modules/list_of_network_modules.html

Network Modules

IOS (Fall 2017)

Cisco IOS

- Ios
 - **ios_banner** - Manage multiline banners on Cisco IOS devices
 - **ios_command** - Run commands on remote devices running Cisco IOS
 - **ios_config** - Manage Cisco IOS configuration sections
 - **ios_facts** - Collect facts from remote devices running Cisco IOS
 - **ios_interface** - Manage Interface on Cisco IOS network devices
 - **ios_logging** - Manage logging on network devices
 - **ios_ping** - Tests reachability using ping from IOS switch
 - **ios_static_route** - Manage static IP routes on Cisco IOS network devices
 - **ios_system** - Manage the system attributes on Cisco IOS devices
 - **ios_user** - Manage the aggregate of local users on Cisco IOS device
 - **ios_vrf** - Manage the collection of VRF definitions on Cisco IOS devices



<https://frank.seesink.com>

Source: http://docs.ansible.com/ansible/latest/list_of_network_modules.html

Network Modules IOS (Fall 2019)

- **ios_banner** - Manage multiline banners on Cisco IOS devices
- **ios_bgp** - Configure global BGP protocol settings on Cisco IOS
- **ios_command** - Run commands on remote devices running Cisco IOS
- **ios_config** - Manage Cisco IOS configuration sections
- **ios_facts** - Collect facts from remote devices running Cisco IOS
- **ios_interface** - Manage Interface on Cisco IOS network devices (D)
- **ios_interfaces** - Manages interface attributes of Cisco IOS network devices
- **ios_l2_interface** - Manage Layer-2 interface on Cisco IOS devices (D)
- **ios_l2_interfaces** - Manage Layer-2 interface on Cisco IOS devices
- **ios_l3_interface** - Manage Layer-3 interfaces on Cisco IOS network devices (D)
- **ios_l3_interfaces** - Manage Layer-3 interface on Cisco IOS devices
- **ios_lacp** - Manage Global Link Aggregation Control Protocol (LACP) on Cisco IOS devices
- **ios_lacp_interfaces** - Manage Link Aggregation Control Protocol (LACP) on Cisco IOS devices interface
- **ios_lag_interfaces** - Manage Link Aggregation on Cisco IOS devices



<https://frank.seesink.com>

Source: https://docs.ansible.com/ansible/latest/modules/list_of_network_modules.html

Network Modules

IOS (Fall 2019 cont.)

- **ios_linkagg** - Manage link aggregation groups on Cisco IOS network devices
- **ios_lddp** - Manage LLDP configuration on Cisco IOS network devices
- **ios_lddp_global** - Configure and manage Link Layer Discovery Protocol(LLDP) attributes on IOS platforms
- **ios_lddp_interfaces** - Manage link layer discovery protocol (LLDP) attributes of interfaces on Cisco IOS devices
- **ios_logging** - Manage logging on network devices
- **ios_ntp** - Manages core NTP configuration
- **ios_ping** - Tests reachability using ping from Cisco IOS network devices
- **ios_static_route** - Manage static IP routes on Cisco IOS network devices
- **ios_system** - Manage the system attributes on Cisco IOS devices
- **ios_user** - Manage the aggregate of local users on Cisco IOS device
- **ios_vlan** - Manage VLANs on IOS network devices (D)
- **ios_vlans** - Manage VLANs on Cisco IOS devices
- **ios_vrf** - Manage the collection of VRF definitions on Cisco IOS devices



<https://frank.seesink.com>

Source: https://docs.ansible.com/ansible/latest/modules/list_of_network_modules.html

Do note I took these slides from a 2019 presentation simply to show how quickly things were moving back then. But this IS from 6 years ago at this point. Please note much has changed in the world of Ansible since then. And since then, I have not done much with Ansible, having moved on to coding in Python using things like Netmiko and Nornir for network automation and Django (for the web framework). I provide this information solely as examples, not as endorsements or claims of the current state of Ansible.

I am NOT idempotent!
Wait... what?



<https://frank.seesink.com>

Idempotent

i·dem·po·tent
/ˌɪdɪmˈpɒt(ə)nt, ˈɛdɪmˌpɒt(ə)nt/ ⓘ

MATHEMATICS

adjective
adjective: idempotent

1. denoting an element of a set that is unchanged in value when multiplied or otherwise operated on by itself.

noun
noun: idempotent; plural noun: idempotents

1. an idempotent element.

Origin

LATIN
idem
same

→ idempotent
late 19th century

ENGLISH
potent

late 19th century: from Latin *idem* 'same' + *potent*¹.

Translate idempotent to

Use over time for: idempotent

Mentions

1800 1850 1900 1950 2010

Source: "The Google"

In simple terms, when you use a tool like Ansible and run a playbook, whether you run it one time or 10x, the end state will be the same. That is what being idempotent means.



<https://frank.seesink.com>



RED HAT ANSIBLE TOWER

Scale + operationalize your automation

CONTROL

KNOWLEDGE

DELEGATION

RED HAT ANSIBLE ENGINE

Support for your Ansible automation

SIMPLE

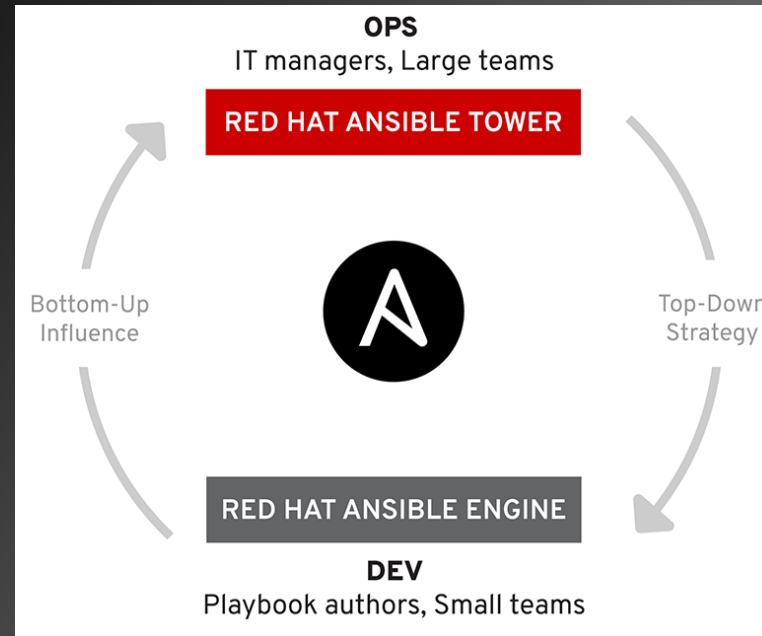
POWERFUL

AGENTLESS

FUELED BY AN INNOVATIVE **OPEN SOURCE** COMMUNITY



<https://frank.seesink.com>



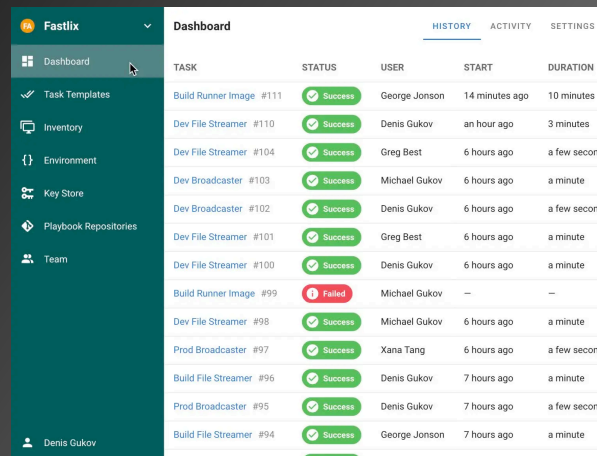
Red Hat Ansible

Ansible (source)	Red Hat Ansible Engine
AWX	Red Hat Ansible Tower
Fedora	RHEL



<https://frank.seesink.com>

Semaphore UI



The screenshot shows the Semaphore UI dashboard for a user named 'Fastlix'. The dashboard displays a list of tasks with columns for Task, Status, User, Start, and Duration. The tasks are listed in descending order of start time. Most tasks are in a 'Success' state, indicated by a green checkmark icon. One task, 'Build Runner Image #99', is in a 'Failed' state, indicated by a red exclamation mark icon. The left sidebar contains navigation links for Dashboard, Task Templates, Inventory, Environment, Key Store, Playbook Repositories, and Team. The bottom of the sidebar shows the user 'Denis Gukov'.

TASK	STATUS	USER	START	DURATION
Build Runner Image #111	Success	George Jonson	14 minutes ago	10 minutes
Dev File Streamer #110	Success	Denis Gukov	an hour ago	3 minutes
Dev File Streamer #104	Success	Greg Best	6 hours ago	a few seconds
Dev Broadcaster #103	Success	Michael Gukov	6 hours ago	a minute
Dev Broadcaster #102	Success	Denis Gukov	6 hours ago	a few seconds
Dev File Streamer #101	Success	Greg Best	6 hours ago	a minute
Dev File Streamer #100	Success	Denis Gukov	6 hours ago	a minute
Build Runner Image #99	Failed	Michael Gukov	—	—
Dev File Streamer #98	Success	Michael Gukov	6 hours ago	a minute
Prod Broadcaster #97	Success	Xana Tang	6 hours ago	a few seconds
Build File Streamer #96	Success	Denis Gukov	7 hours ago	a minute
Prod Broadcaster #95	Success	Denis Gukov	7 hours ago	a few seconds
Build File Streamer #94	Success	George Jonson	7 hours ago	a minute

<https://semaphoreui.com/>



<https://frank.seesink.com>

Today if I were to use Ansible and want to automate playbook execution, I would likely look into tools like Semaphore UI, which did not exist back in 2017.

So THAT's why
Ansible



<https://frank.seesink.com>



Now you cannot talk about Ansible without also talking about YAML.

What is YAML?

- Yet Another Markup Language
- DSL (Domain-Specific Language)
- Text-based, human-readable



<https://frank.seesink.com>

For those not familiar, YAML is a text-based, “human-readable” language used by Ansible for its configuration files, inventory, and playbooks.

Playbook (raw)

```
---
- name: Show version of IOS running on routers
  hosts: routers
  gather_facts: false

  tasks:
    - name: Use raw mode to show version
      raw: "show version"

      register: print_output

    - debug: var=print_output.stdout_lines
```



<https://frank.seesink.com>

These are extremely short (and outdated) examples of Ansible YAML playbooks. They are just to show the formatting/layout.

Playbook (ios_command)

```
---
- name: Backup running-config on routers
  hosts: routers
  gather_facts: false
  connection: local

  tasks:
    - name: Backup the current config
      ios_command:
        authorize: yes
        commands: show run

      register: print_output

    - name: save output to a file
      copy: content="{{ print_output.stdout[0] }}" dest="./output/
            {{ inventory_hostname }}.txt"
```



<https://frank.seesink.com>

Limitations of YAML

- Very finicky about indentation, spacing, quotes
- Once you try applying conditional logic, things get ugly fast



<https://frank.seesink.com>

The challenge with YAML is that you can really get caught up in the formatting of it. This also includes such things as how to properly escape strings within strings for proper dereferencing.

Also, it works well for basic definitions of a sequence of steps. But once you have to do anything for which an Ansible module does not exist, or you want to apply more complex conditional logic, it quickly falls apart. The key thing to understand is when to use Ansible... and when not to.



Once you reach the limits of Ansible itself, you often find the next step is a foray into the world of Python. This often begins with trying to learn just enough Python to create whatever Ansible module you need that currently does not exist or does not work properly.

But again, there is trying to shoe-horn a solution together, and then there is simply realizing that it is time to bring actual coding to bear.



Python

- Paramiko (SSHv2)
- Netmiko
 - “Multi-vendor library to simplify Paramiko SSH connections to network devices”
- Nornir
 - “Pluggable multi-threaded framework with inventory management to help operate collections of devices”
- NAPALM (Network Automation and Programmability Abstraction Layer with Multivendor support)
 - mostly Arista, Cisco, Juniper



<https://frank.seesink.com>

Python’s tagline is “batteries included.” This implies that Python comes with all the key abilities you need built-in. However, the real power of Python comes from the ecosystem of 3rd party libraries that exist.

For example, Paramiko let you SSH into single devices.

Netmiko leveraged Paramiko to SSH into single network devices, handling things like the CLI prompt.

Nornir added multi-threading to Netmiko and inventory similar to Ansible, letting you SSH simultaneously into multiple network devices.

NAPALM is another framework that simply caters more to just the 3 vendors mentioned.



Python

- Kirk Byers (<https://pynet.twb-tech.com/>)
- offers network automation courses on Ansible, Python, Nornir, and Git
- There are far too many books, courses, YouTube videos, etc. out there on Python to recommend specific ones.



<https://frank.seesink.com>

You can't talk about Netmiko, Nornir, and NAPALM without inevitably talking about Kirk Byers.

As for which books, courses, etc., to take to learn Python, it truly is a case of "it depends", as everyone learns best different ways. What I would suggest is that you figure out how you best learn, then focus on that. If your job gives you access to things like LinkedIn Learning, and that approach works for you, there are several good courses there.



Python

- Network Automation Cookbook: Over 100 recipes to effectively configure and manage network infrastructure with Ansible, 2nd Ed
- Network Programmability and Automation: Skills for the Next-Generation Network Engineer, 2nd Ed by by Matt Oswalt, Christian Adell, Scott S. Lowe, Jason Edelman



<https://frank.seesink.com>



Here are just two examples of 2nd editions of books which may or may not help.

Development Tools



<https://frank.seesink.com>

Once we head down the programming path, inevitably tooling comes into play.

Now this topic is far too dense to cover in an overview, so this will be an extremely high-level smattering at best.

Development Tools

- IDE (**I**ntegrated **D**evelopment **E**nvironment)
 - e.g., Visual Studio Code (VSCode) / VSCodium
- Git (official site)
 - a distributed version control software system
 - Hosting services, e.g., GitHub, GitLab, Gitea



<https://frank.seesink.com>

IDEs offer editing systems, typically complete with extensions/plugins to help with the workflow of program development. One of the most popular today is Microsoft's VSCode (or the VSCodium fork for those who do not trust Microsoft), though many others exist. For example, Sublime Text, JetBrains, Cursor, and more.

And yes, Cursor is an IDE that integrates AI, but we'll get to that soon.

Once you truly start developing code, you will invariably need to learn about doing version control. And today that really means learning Git. Using Git solely on your own development system, with no need for any server, you can leverage its features to help you do versioning/etc. And you can easily work with a team that all use Git solely on their development systems.

But the real power comes when you also keep your code in a repository in a central location, so that you don't have to be online at the same time as your coworker in order to perform the typical actions involved such as pull requests and merges.

Python Tools

- PyPI website
- pip (**P**ackage **I**nstaller for **P**ython)
- Virtual Environments (venvs)
- Poetry
- uv / ruff



Now as you go further into Python, you will quickly realize that the “batteries included” ethos is about using 3rd party Python libraries. These are stored on a website called PyPi.org.

pip is standard with Python and lets you pull down Python modules (i.e., “LEGO pieces”) such as Netmiko to use in your Python code. However, everything you do has to be done manually. And pip is very slow.

venvs are the way in which you “sandbox”/isolate your Python projects from one another to avoid what I call “dependency hell” or collisions in module versions. Python again has support built-in for this, though other modules exist like ‘pipenv’ and ‘virtualenv’ which each do things slightly differently.

Poetry is yet another package manager out there that tried to improve things.

But today the “new hotness” is Astral’s uv, which is an “extremely fast Python package and project manager, written in Rust,” which replaces the need for several other tools. uv, in short, brings Python development about as close as you are going to get to a uniform workflow such as you might see in Go development.



Which brings me to a quick plug for Go.

To be clear, if you were only going to learn one programming language to do network automation, you should probably learn Python.

But if you are going to learn two languages, I strongly recommend that you consider Go.

Why Go?

- Python's creator, Guido van Rossum, worked at Google from 2005-2012.
- For years Google heavily used Python internally and even offered Python classes to its employees.
 - <https://developers.google.com/edu/python>
- Google had also hired Rob Pike and Ken Thompson of Bell Labs (UNIX, C) fame. They, along with Robert Griesemer, created Go.
- In 2013 Guido van Rossum went to work at Dropbox. (Dropbox was known to use Python.) That seemed odd.
- In 2014 Google publicly released Kubernetes, which is written in Go.

The writing was on the wall?



<https://frank.seesink.com>



Terraform



CoreDNS

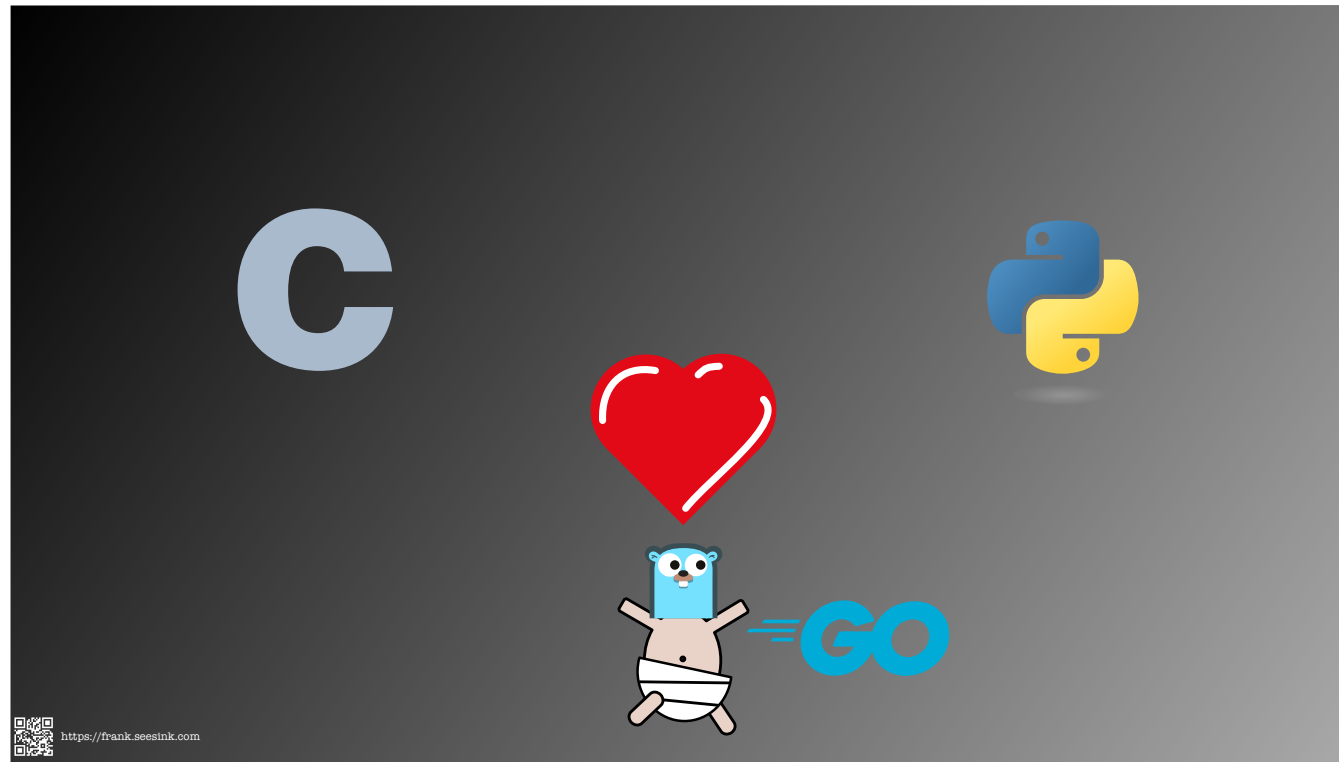


Cilium

Go, often called Golang (easier to find on the Web), is a statically compiled programming language that underpins much of the cloud. Originally released back in 2009, Go was intended to replace Python in Google's use cases. But it would become the foundation of cloud computing, with tools such as Docker, Kubernetes, Terraform, Grafana, Helm, etcd, CoreDNS, Traefik, Cilium, and many more all being written in Go. And that is not an accident.

And I genuinely believe that Go will eventually replace Python for many use cases in the coming decade or more.

But I only bring this up because when it comes to tooling, where in Python you need all these 3rd party tools like Poetry, uv, and ruff, with Go everything you need is in Go compiler. This is because Go includes everything to write, compile, test, and lint your code. The only 3d party bits, such as Go extensions in VSCode, are to hook into those very same tools within the Go toolchain.



Go is as if C and Python got together and had a baby, giving you best of both worlds.

You get all the benefits of a statically compiled language that creates binaries without any external dependencies combined with the rapid development cycle of a dynamic scripting languages with Pythonic characteristics.

Source of Truth (SoT)



<https://frank.seesink.com>

With Ansible, you have what is called an “inventory”, which is a simple directory structure of YAML text files which define all the devices you wish to interact with using Ansible playbooks. This might be considered the “source of truth”. But let’s discuss this further.

What is a Source of Truth?

- Source of Truth (SoT)
- Single Source of Truth (SSoT)
- Source of Intent



<https://frank.seesink.com>

When discussing network automation, the topic of a “Source of Truth” often comes up. Simply put, the SoT is what you see as the place that tells you what your network SHOULD be, not necessarily how it IS. Some refer to this as the Source of Intent.

The name has caused confusion as some folks would argue that the only real source of truth is the network itself, as only the network gear has both its configuration and current state (e.g., current ARP or routing tables, etc.). But the idea is that the SoT is that it is the DECLARATIVE place where you define how your network SHOULD be, and what any network automation tool will turn to in order to do its work.

Sources of Truth

- NetBox from NetBox Labs
 - originally created at Digital Ocean by Jeremy Stretch
- Nautobot from Network to Code (NtC)
 - a fork of NetBox (long story)
- Infrahub from OpsMill

* All are open-source and offer paid support options



<https://frank.seesink.com>

Some example of SoTs. NetBox is the “OG” of such tools. Nautobot came about after Jeremy Stretch left Digital Ocean, eventually went to work for NtC, they had a difference of opinion regarding NetBox’s future, so they parted ways and NtC forked NetBox into Nautobot. But both are Python/Django-based projects.

InfraHub is a newer player on the scene which leverages Neo4j and “is a graph-based data management platform with built-in version control, CI workflows, peer review, and API access. It’s purpose-built to power reliable infrastructure automation at scale.” Think “Git meets NetBox”.

Tools and platforms currently being recommended



<https://frank.seesink.com>

This brings us to one of the key topics requested.

Tools/Platforms

- Options range...
 - from FLOSS (Free/Libre Open Source Software)
 - to fully commercial, vendor-specific options

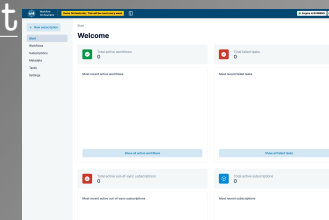
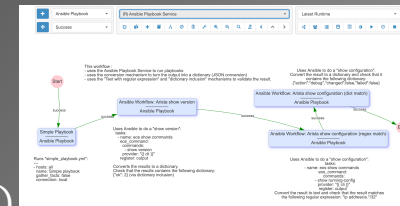


<https://frank.seesink.com>

For those looking for a more “canned” solution, this ranges from FLOSS where “if you break it, you get to keep both pieces” all the way to fully commercial, vendor-specific solutions.

Tools/Platforms (FLOSS)

- eNMS (GitHub repo)
- Workflow Orchestrator (GitHub repo)
- developed by SURF together with ESnet



 <https://frank.seesink.com>

In no particular order, you have tools such as eNMS and SURF/ESnet's Workflow Orchestrator, both written in Python.

Tools/Platforms (No-/Low-Code)

- No- to low-code commercial options that provide multi-vendor support:
 - Gluware
 - Itential



<https://frank.seesink.com>

On the no- to low-code commercial side of things, you have vendors such as Gluware and Itential offering solutions.

Tools/Platforms (Arista)

- Arista AVD (Architect Validate Deploy) “is an extensible data model that defines Arista’s Unified Cloud Network architecture as “code”.”
- There is an AVD Ansible Collection and PyAVD, both of which leverage EOS API (eAPI) and CloudVision Portal (CVP). CloudVision “is Arista’s modern, multi-domain management platform that leverages cloud networking principles to deliver a simplified NetOps experience.”



<https://frank.seesink.com>

Starting down the vendor-specific path, in purely alphabetical order we begin with Arista and AVD.

Tools/Platforms (Cisco)

- Cisco DNA Center (DNAC), now called Cisco Catalyst Center, and
- NSO (Network Services Orchestrator)
 - Formerly Tail-F, NSO is multi-vendor in nature.



<https://frank.seesink.com>

Next we have Cisco with what is now called Catalyst Center.

And then there is NSO. Formerly Tail-F, an independent company that built a truly multi-vendor solution, Cisco bought them and rebranded the software. It is still Tail-F underneath, however, and supposedly they have kept their multi-vendor support.

But as one example of an NSO customer, you have Internet2.

Tools/Platforms (Extreme Networks)

- ExtremeCloud IQ Site Engine (XIQ-SE) - on-prem
- ExtremeCloud IQ - cloud-based



<https://frank.seesink.com>

For those using Extreme Networks, for on-prem solutions you have what originally was Enterasys' Netsight tool, which later was rebranded to Extreme Management Console (XMC) after Extreme bought Enterasys, and later again was rebranded as ExtremeCloud IQ Site Engine (XIQ-SE) after Extreme bought Aerohive Networks and had rebranded their cloud-based management platform, HiveManager NG, as eXtreme Cloud IQ (XIQ for short).

And this brings us to their cloud-based offering, eXtreme Cloud IQ (XIQ). XIQ-SE handles on-prem access while XIQ provides cloud-based management.

Tools/Platforms (Others)

- Aruba (Central)
- Juniper (Mist)
- Nokia (e.g., Event-Driven Automation (EDA))



<https://frank.seesink.com>

Beyond that, you have various other tools out there, such as those in the wireless space such as Aruba Central and Juniper's Mist (now under HPE).

And then there is Nokia with their EDA.

“Proof is left as an
exercise for the
reader”



<https://frank.seesink.com>

This was a favorite expression of one of my professors. We often thought it was just his way of getting out of having to explain something to us.

I am using this as a placeholder as there are just too many topics/areas to cover around network automation, so I am just listing them here without further explanation for now.

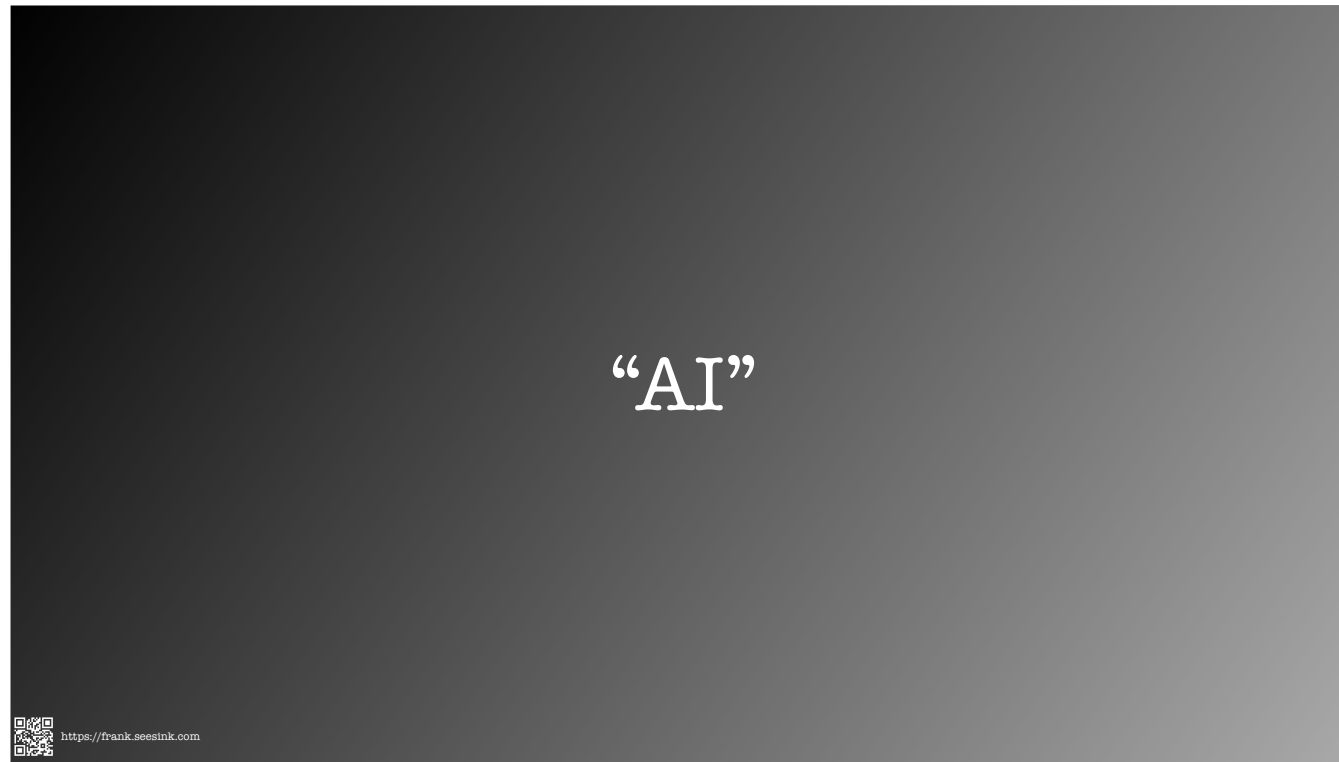
Other Topics Not Covered

- JSON - JavaScript Object Notation
- NETCONF (Network Configuration Protocol) defined by IETF. RPC using XML.
- OpenConfig - data models written in YANG (Yet Another Next Generation)
- RESTCONF defined by IETF. Uses HTTP and YANG.



<https://frank.seesink.com>

But you may well encounter them as you delve deeper into network automation.



You can't swing a dead cat without hitting something “AI” related these days. So let's talk about that a bit.

genAI

- OpenAI's ChatGPT
- Anthropic's Claude
- Google Gemini
- Meta Llama
- xAI's Grok
- Apple Intelligence
- Alibaba Cloud
- DeepSeek AI
- Mistral AI
- ...



<https://frank.seesink.com>

Unless you have been living under a rock the past few years, “generative AI” has been all the rage. (Even though it is truly neither of those things—neither “generative” nor “AI”, but rather more accurately “derivative ML”—I will do my best to refrain from getting on my soapbox about all this for now.)

Beginning with OpenAI’s ChatGPT leading the charge, now we have all the major “Big Tech” companies, along with several others, all scrambling to try and achieve AGI (Artificial General Intelligence).

LLMs

- **L**arge **L**anguage **M**odels
- Terms: Parameters, weights, tokens, prompts, context window size
- Think artificial brain trained on massive amounts of text, where in the end you have something which you can ask questions of or give commands to



Now the actual tech employed are LLMs (Large Language Models). These are computer programs intended to mimic the human brain of neurons and synapses using parameters and weights, where they feed these LLMs massive quantities of text which are tokenized to “train” the model.

The end result is this “brain” which, given a sequence of words (i.e., tokens), determines what the most probable set of words should be in the response. In short, LLMs are like word probability engines.

So when you hear “parameter”, think neuron.

When you hear “weights”, think the thickness of a synapse between two neurons.

When you hear “token”, think word.

When you hear “prompt”, think “ask a question/give a command.”

When you hear “context window size”, think “how much can the LLM ‘remember’ as you interact with it.” This is typically listed by the number of tokens the LLM can keep in memory before it begins “forgetting.”

This will help make sense of the language being used in the AI space.

Cloud AIs

- With each new version, they get... better(?)
 - e.g., ChatGPT-3.5, ChatGPT-4, ChatGPT-4o, and now ChatGPT-5
- Resource consumption
- Privacy issues



<https://frank.seesink.com>

As these cloud AI companies all race to achieve AGI, they are iterating their respective LLMs, tuning how they train their models, along with how many parameters their model holds, etc.

And with each iteration, as the models grow larger while they try to optimize the compute power needed to train the model, the overall needed compute capacity drives up the need for more datacenters, and therefore the amount of electricity, fresh water, etc., that these models consume.

There are also all the privacy issues around using such AIs, as any information you provide an AI while interacting with it (such as asking it to summarize an email you received) means in essence that you have uploaded a copy of that information to the cloud provider, who now holds a copy of that information, with the possibility of training their LLMs in a future iteration on that data. The security implications of this are immense.

Cloud vs. Offline LLMs

- “Open Source” versions of cloud LLMs



<https://frank.seesink.com>

Which brings us to the topic of offline LLMs. Beyond the usual cloud-based “AI”s such as ChatGPT, almost every AI company offers a stripped down version of their LLMs which you can download and run offline if you have sufficient resources. In the PC world, this means having a decent GPU such as an Nvidia-based graphics card. If you have an Apple Silicon Mac, you can leverage the GPU cores in those ARM64-based chips.

Each of these offline LLMs typically comes with an “open source” license. And I put that in quotation marks because the only thing that is open source is the license itself. The actual LLM you download is not. You do not have access to the training data used to train the model, so you cannot see how it “learned.”

Cloud vs. Offline LLMs

	Cloud LLM	Offline LLM
Parameters	TRILLIONS	Billions
Weights	32-bit	2-, 4-, 8-bit
Compute Resources Required	Datacenters	PC
Cost	Subscription	FREE



<https://frank.seesink.com>

So why wouldn't everyone simply run LLMs offline?

Well to understand the difference between cloud-based and offline LLMs, here is some context. As you can see, offline LLMs are basically “lobotomized” versions of their cloud-based cousins. Where a cloud-based AI has TRILLIONS of neurons where the thickness of their synapses are stored as 32-bit values, in order to bring that “brain” down to a size that you can run on a PC, it has to be paired down to just so many BILLIONS of neurons where those same synapse thicknesses are stored as 8-bit, 4-bit, or even just 2-bit values. Not quite the same level of detail.

And yet, even so, often such offline tools can be useful IF the right LLM is chosen for the task at hand.

Example of Offline LLMs

- OpenAI's gpt-oss 20B
- Qwen3 Coder 30B
- Deepseek R1
- Gemma 3 27B
- Hugging Face - think of it like "PyPI for AI"



<https://frank.seesink.com>

Here are some examples of offline LLMs that you can download and use locally. Mind you, there are thousands at this point, as folks pull down the ones offered by the major players, then tweak them in various ways to provide them with additional training/bits.

One very popular site for hosting such LLMs is Hugging Face. Do not ask me about the name. But know that this is a site where you can find just about any such LLM.

Offline LLM Tools



	UI/UX	FLOSS	Comments
Ollama	CLI	✓	• “OG” offline tool
LM Studio	GUI		• MLX/GGUF support • detects hardware
Jan	GUI	✓	
AnythingLLM	GUI	✓	Offline RAG client

- All are FREE and run on Linux, macOS, and Windows



<https://frank.seesink.com>

Here are some example offline LLM tools.

Ollama is the “OG” of this space. An open-source CLI-tool that lets you download LLMs much like doing Python “pip install” commands, Ollama is the basis for many other tools out there.

LM Studio is a proprietary program offering probably the nicest and simplest user interface, allowing you to easily download LLMs from Hugging Face by simply searching within the app and clicking. If you have any interest in trying to run LLMs offline, this would be my first choice for trying things out.

Jan is a truly open-source program similar to LM Studio in that it provides a GUI for running LLMs.

And AnythingLLM has some unique features, notably that you can use it to build vector databases of your own documents, what are referred to as RAGs (Retrieval-Augmented Generation), where, for example, you might create such a RAG of all your Python eBooks, then run LM Studio, where you have it tie into AnythingLLM, and when you prompt the AI in AnythingLLM, it can provide more accurate responses as it leverages all those eBooks in the RAG.

While LM Studio is capable of letting you load up to 4 documents into it for doing RAG, AnythingLLM has no such limit. And while the process of creating such a vector database can take a few minutes, once done it is immediately available any time you run your offline LLM.

IDEs with AI

- GitHub CoPilot (with VSCode)
- Cursor
- Many other VSCode-based startups



<https://frank.seesink.com>

Now when it comes to using AI in development, there are various options. These range from things like GitHub CoPilot and Cursor to a myriad of small startups which almost all are forking the VSCode IDE and bolting on some form of AI.

LLMs in Network Automation

- Mist Systems (bought by Juniper, who HPE bought)
- John Capobianco



<https://frank.seesink.com>

However, this session is not about AI, but rather about how AI is being used in network automation.

Now one of the first products which touted using AI years before most others was Mist Systems with their WiFi offerings. Mist did such a good job with their product that Juniper bought them. And as they continued to grow, they became an existential threat to more traditional players. Some would say so much so that HPE ended up buying Juniper.

But beyond that, probably the name you will end up hearing the most at the moment is John Capobianco. He has been making YouTube videos for awhile now showing how he has been integrating AI into various aspects of networking. John is a good guy, though I will say that he does appear to have drunk a bit much of the AI KoolAid. But his videos are very informative should you wish to learn more about what is possible. (I will leave it to you to decide whether some of this falls in the camp of “Just because you can does not mean that you should.”)

Key Workflows That Can Be Automated



<https://frank.seesink.com>

Key Workflows to Automate

Read-only operations; e.g.,

- perform config backups
- pull state or config data; e.g.,
 - current date/time on all routers/switches
 - NTP server IPs
 - SNMPv2/v3 config
 - ARP tables from switches
 - VLAN configs from interfaces, etc.



<https://frank.seesink.com>

So quick comment. Regarding computer use in general, I have always said,

“If you are organized in real life, the computer can help you streamline your workflows and save you a lot of time. However, if you are disorganized in real life, the computer will only help speed you into oblivion.”

So regarding automation, it is important to understand the potential “blast radius” of any actions you take. This is why I always tell folks, “Start with read-only operations” and “Gun for the low hanging fruit first.” That is, find something repetitive that gives you no joy, ideally something simple to start with, and see if you can automate that away. Yes it will take time to build the automation initially. But pay attention to how much time it will save in the long run.

Key Workflows to Automate

Then what you can do with that read-only information; e.g.,

- verify all network gear has correct date/time
- verify all network gear has correct NTP/SNMP configs
- create search tool that, given a MAC or IP, provides the device/interface where it was last seen
- create a tool to find unused VLANs across your network



<https://frank.seesink.com>

The key thing here is to look for examples where you can save time.

Take the example of a boss who, when asked how to do something, can either

1. Do it in 10 minutes, or
2. Spend an hour teaching an underling how to do it.

Most tech people choose the former. But this is short-sighted. First, you never have knowledge transfer to the younger generation. But worse, you are shooting yourself in the foot. If the task is something that needs to be performed just once a week, that means you just short-changed yourself out of > 7 hours in the first year alone.

Key Workflows to Automate

Read-write operations such as

- modify NTP/SNMP/etc. settings
- modify interface VLANs, including adding a VLAN (the infamous `switchport vlan add...`) to prevent overwrites



<https://frank.seesink.com>

Once you get through read-only operations, the next step is to begin working on read-write operations, those things which actually CAN affect your network. And here you want to start small, testing against one device, then a few, until you develop enough confidence in your code to unleash it on your entire network.

Here is where you might write code to handle catching typos such as “switchport vlan <#>” instead of “switchport vlan add <#>”, thereby avoiding a common issue.

Efficiency Gains and Overall Impact



<https://frank.seesink.com>

Efficiency Gains/Impact

- In 2017 wrote Ansible playbook to upgrade 55 Cisco 3945s + their switch modules (i.e., 110 network device upgrades), located across the entire state, in < 1 hour
- Manually this would have required weeks, if not months, to coordinate/plan each one



<https://frank.seesink.com>

This was my first foray into network automation with Ansible. In the state I worked in, there were 55 counties, each with a county tax office and courthouse with Internet access that we provided. The playbook I wrote connected to all 55 county routers and the switch modules in them, where it

- checked which version of code each was running
- checked if the updated IOS image was in flash and, if not,
 - uploaded it to flash
 - verified the upload
 - adjusted the configuration to load the new version
 - reloaded the router/switch
 - waited for the device to come back online
 - logged back in to confirm that the device was on the new version
- and reported back the results

In short, the playbook, which I originally wrote against a single router, did every step that I would have had to do by hand, only against all 55 at the same time.

This took a manual process which would have required weeks if not months of planning to coordinate downtime for each of the 55 counties to less than 40 minutes after hours one evening for all 55 counties at once.

Efficiency Gains/Impacts

- In 2019 wrote a VPN Tool in Python/Django to automate work that previously, for each request, required interrupting me specifically to do ~15 minutes of manual work across multiple systems to collect the relevant information.
- Now each such request is reduced to ~10 seconds (mostly due to navigation to the webpage) AND, most importantly, does not require me and can be performed by anyone.



<https://frank.seesink.com>

When we first started our DevOps group where I work now, we settled on using Python as our language and Django as our web framework for any web-based tooling. The first project we took on was rewriting a Perl tool that had been called “RouterProxy” (taken from the online sites you can TELNET/SSH to in order to do testing) and whose developer no longer worked there. That tool has since been expanded well beyond its initial intent, and my VPN tool was simply another “LEGO piece” in that tool chain.

VPN Tools

Router Proxy

AnyConnect

ONYEN

VPN Group

Statistics

Frank Seesink (fseesink)

1


Enter the ONYEN of the user you need information about or are helping

ONYEN

Find

Copyright © 2025. All rights reserved. UNC at Chapel Hill | ITS.

Version 1.1 (2021-07-27)

<https://frank.seesink.com>

Challenges Faced and Lessons Learned



<https://frank.seesink.com>

Challenges/Lessons

- Operational work vs. “Makers hours”
- Subverting DevOps workflows/processes
- Do we hire network engineers that we teach programming or hire programmers that we teach networking?



<https://frank.seesink.com>

Automation/programming is very different from day-to-day operational work. The former requires "makers hours" while the latter requires shifting context regularly. Each disruption to someone programming/etc. can easily cost 15–30 minutes or more of productivity, as it takes time to get back into “the zone”.

If you are in a managerial role, FIERCELY protect your automation/programming/DevOps people from such disruptions or you will find things take much longer and it will impact morale.

Far worse on morale, however, is subverting workflows/processes that your DevOps group has put in place. If your group standardizes on some given language, frameworks, tools, etc., do not undermine that effort.

If you give your folks the time/freedom to work on something that interests them, you will often be pleasantly surprised by the outcome.

Future Plans / Opportunities to Explore



<https://frank.seesink.com>

Honestly, the sky is the limit here.

Future Plans / Opportunities to Explore

- Developing network-wide automations
- Developing network automation that spans across AS boundaries
- Integrating network and system automations



<https://frank.seesink.com>

Again, network automation is a journey, going from very simple tasks up to developing network-wide automations.

Beyond that, there has been some discussion around the possibility of extending automations across AS boundaries. That is, whether there are opportunities for different RENs and members, or even NRENs, to find ways of bringing collaborative automation to bear that goes beyond just one organization.

There are also opportunities in integrating systems with networking. For example, currently in many datacenters they are using BGP as the iBGP. But what if there were more ways for applications to adjust/tune the network for optimal performance?

Resources

- Internet2 Network Automation SIG
 - I2 Slack workspace #i2-ntac-networkautomation channel (free to join for any I2 member)
 - Mailing list (not terribly active)
 - Zoom meetings where we discuss issues
 - every 1st Thu. of the month @ 3:PM Eastern
 - every 3rd Tue. of the month @ 11:AM Eastern



<https://frank.seesink.com>

Now as for resources available, there are many.

For anyone who is an Internet2 Connector or Member, we have the I2 Network Automation SIG. This includes having our own channel, #i2-ntac-networkautomation, where we chat about all things network automation. We also have two (2) online Zoom meetings each month, where we discuss whatever the members are interested in. It is a very relaxed environment, and you are welcome to join and simply lurk around if you like. We use an EtherPad to let folks add agenda items, and I think you'll find the environment very supportive.

For anyone who wishes to be added to the Slack, please let me know and I will talk to Internet2. Or if you know her, reach out to Linda Roos and feel free to tell her that I sent you to her.

And if anyone is interested in joining the Zoom calls, also let me know and we'll get you the details.

Resources



<https://frank.seesink.com>

Resources

- [2025 Internet2 Technology Exchange](#), 8-12 Dec., Denver, CO
- Tutorials
 - [Implementing CI/CD Pipeline in a NetDevOps Environment](#)
 - [Get Good with GitOps](#)
 - [Network Troubleshooting with Generative AI](#)
 - [Gemini Code Assist Essentials](#)
- Various sessions in the Advanced Networking track
- NetGurus
- [Higher Education NetComm Workshop](#), 27 Oct., (Monday before EDUCAUSE in Nashville, TN), hybrid (in-person/virtual)

FREE



<https://frank.seesink.com>

Then there are the Internet2 conferences: the Community Exchange in the spring and the Technology Exchange in the fall. Of the two the Technology Exchange has more “meat” regarding network automation or technical matters. And the next TechEX is in early December in Denver, CO. I currently plan on being there.

At TechEX you will find various tutorials, including the ones listed, and there are several sessions in the Advanced Networking track during the week. I should know as I was invited to co-chair the Advanced Networking track this year. There is also the NetGurus meeting, held that Friday, which is mostly for the member institutions.

Then there is the shameless plug for the Higher Education NetComm Workshop going on the day before EDUCAUSE kicks off. Whether you are planning to attend EDUCAUSE in person or simply want to attend virtually, consider signing up. Registration is free. I know of at least one session that will be network automation related going on that day.

Resources

- Network Automation Forum (NAF) (website, blog)
- NAF Slack workspace (to join)
- AutoCon 4, 17-21 Nov., Austin, TX
- AutoCon 5, Spring 2026, Europe
- Packet Pushers (website, blog)
- PP Slack workspace (to join)
- Podcasts, including “Network Automation Nerds” with Eric Chou
- NtC’s “Awesome Network Automation” GitHub repo

ALL FREE!*



<https://frank.seesink.com>

* Ok not AutoCon

There is now also the Network Automation Forum (NAF), which if you only join one group online, it likely should be this one, as it is SPECIFICALLY about network automation. They offer a very active Slack workspace, and they organize two (2) conferences each year. Their conference, called AutoCon, is held in a tick-tock fashion, with one in the fall typically held in the U.S. and the other in the spring held somewhere in Europe.

AutoCon 0 was in Denver in Fall 2023.

AutoCon 1 was in Amsterdam in Spring 2024.

AutoCon 2 was again in Denver in Fall 2024.

And AutoCon 3 was in Prague in Spring 2025.

AutoCon 4 will be in Austin, TX, this November. Tickets do sell out, so be aware.

Beyond NAF, you also have the Packet Pushers. They offer their website/blog, a ton of podcasts including one specifically about automation. And they have a very active Slack workspace.

Finally, there is the Network to Code’s GitHub repo “Awesome Network Automation”, which contains a pile of links to various tools and information, far too much for me to list here.

Thank You

<https://frank.seesink.com/presentations/NYSERNet-Connect-Fall2025/>



“Hallway track/chat”
<https://tlk.io/nysernetconnect2025>

Frank Seesink
frank@seesink.com



Again, for anyone who would like a copy of this slide deck, visit the QRCode or URL at the top left.

And although this conference is virtual, I thought I would try to “be around” in the virtual hallway as it were for the remainder of the day. For anyone who would like to keep the conversation going today, or who has any questions, etc., I have created this simple web chat on tlk.io as an experiment. Just visit the QRCode or URL at the bottom right

Simply visit the URL, type in a name (or make one up if you prefer), and start typing. (You can also log in with Twitter or Facebook, but I intentionally searched for something that did not require doing so, let alone creating an account, etc. The idea here is that it is just a simple, “friction free” way to chat for the day.