Automating the Modeling of Production Networks in Virtual Labs

04-FEB-2025

Preamble

04-FEB-2025

NANOG™

Who *is* this person talking to you?

- I’ve been “doing network stuff” for just over 25 years
- Long-time network architect
- Long-time automation “enthusiast”
- First-time NANOG-er

Overview

- Virtual Network Labs (VNLs) are *fantastic* resources
- They are also a real nuisance to use *effectively*
- This talk is somewhere between a case-study and a tutorial on *one* approach to automating away the nuisance

Target Audience

- If you *have* non-prod *hardware* where you *already* replicate your production network…
  - …*this* talk is below your pay-grade (but thanks for slumming it with us today!)
- For the rest of us, virtual labs are often the "*next best thing*"

Agenda

- The Cases For (and Against) VNLs
- Case Study: ‘ptovnetlab’
  - TL/DR Version
  - Initial Design Decisions
  - Functional/Task Analysis
  - Brass Tacks
  - Lessons Learned
- Thanks!
- Q&A

The Case For Virtual Network Labs

04-FEB-2025

NANOG™

Learning / Studying / Exam Prep

- Old hat / boring
- Not what I'm here to talk about
- But, *obviously*, *the* initial use-case for a lot of us

Change Modeling/Validation

- *I* always felt a lot better headed into a change window if I'd done something more tangible than desk-checking my configs
- Even validating configuration validity in the context of a single device would have been a win

Troubleshooting

- Say it out loud (if you please),
  - *“The **cure** can’t be worse than the **disease**.”*
- Physical labs have a long wait time
  - Virtual labs can be set up much faster than physical ones

Cost/Benefit

- Virtual labs have *low* costs
- Benefits in terms of FTE-hours saved add up quickly
  - Peer-review of basic syntax can be avoided
  - Re-work of changes due to logic-errors can be avoided
- MTTR for tough-to-diagnose issues can be reduced
  - An in-house virtual-lab might be set up in a few hours compared to a few *days* if relying on a physical lab

The Case Against VNLs

04-FEB-2025

NANOG™

Fidelity

- The VNF version of product-X is *not* identical to its physical form factor
- Virtual labs are *generally* running as either x86 virtual-machines or container images on an x86 OS.
  - Which cannot replicate the behavior of the ASICs/FPGAs/etc. in networking equipment
- The devices the *you* own are generally connected at some point to devices that you *don't* own

Level of Effort Increases Exponentially

- The more devices you want to model, the larger the LOE
  - The more *links* you have per device, the steeper the curve
- Bigger scope ~= higher fidelity
- Bigger scope also = higher LOE for setup

Availability of Virtual Form Factors

- If the equipment you're modeling has no virtual form factor, you're out of luck.
- If it only has a virtual-machine form-factor, your virtual-lab will have a substantial resource footprint
- If the equipment you want to "lab up" **does** have a container form-factor, you **are** the intended audience for this talk

Case Study: TL/DR

04-FEB-2025

NANOG™

Where We Started

- Two colocation data-center footprints
- Partially migrated to Netbox as DCIM/NSOT
- A moderately-robustly-provisioned GNS3 server instance

What We Wanted

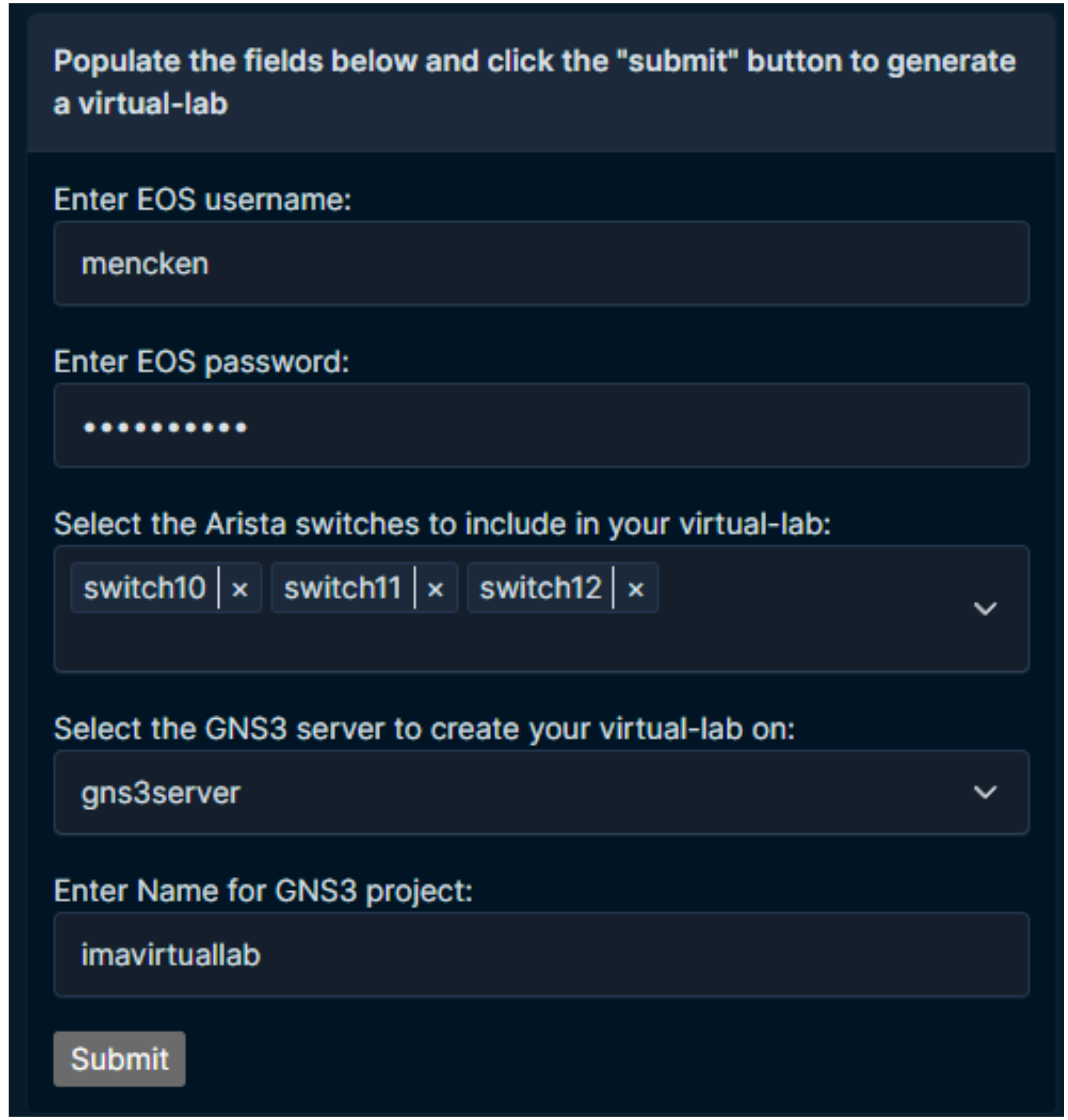
- A better juice-to-squeeze ratio for modeling production networks in virtual labs
  - Less time *building* virtual-labs
  - More time *using* virtual labs

Where We Landed

- A homegrown python package for the heavy lifting of collecting switch run-state and converting/instantiating to a network virtual-lab
- A homegrown Netbox plugin for WebUI and inventory integration with the python package
- Roughly two orders of magnitude decrease in LOE to model a full data-center switch fabric.
- Takes 4:22 (m:s) to fully model a 42 node folded clos topology.
  - 6.3 seconds/switch

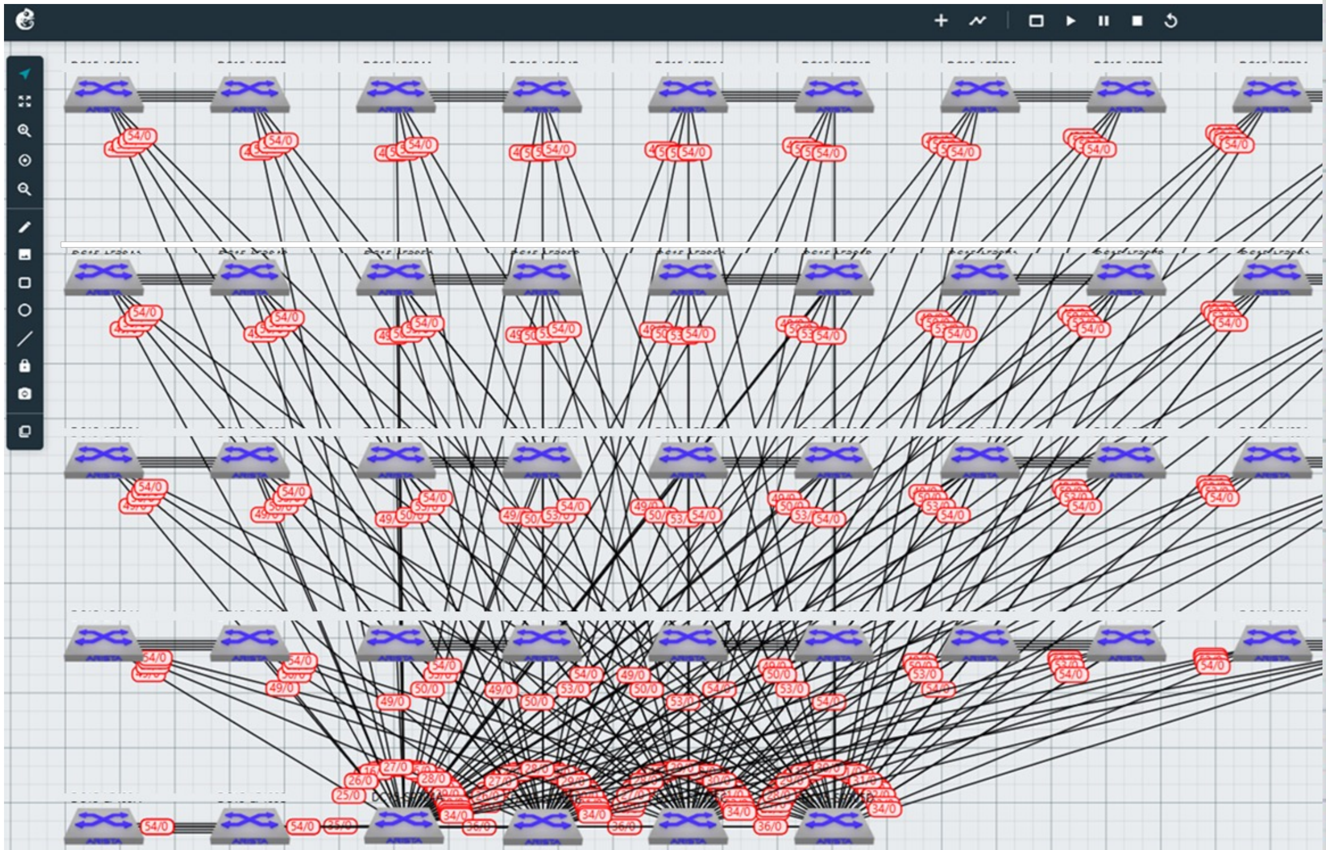
Easy in...

- Netbox plugin for WebUI
- Select devices to model from existing inventory
- Select virtualization-server from existing inventory
- Go, go, go!



NANOG™

...Easy Out



NANOG™

Case Study: Initial Design Decisions

04-FEB-2025

NANOG™

Initial Design Decisions

- Use GNS3 server as virtual-lab platform
- Develop in Python
- Use APIs for as much as possible
- Model devices based on actual *run*-state
- Use/create open-source tools whenever feasible
- Prefer container form-factor for VNFs

Case study: Task Analysis

04-FEB-2025

NANOG™

If You Can *Describe** it, You Can *Automate* it

- Modeling our production devices in a virtual lab was a highly deterministic process
- Thus, a good candidate for automation
- Nothing ground-breaking here
  - Just write down the process you follow, step-by-step
  - Have a ten-year old follow the process you write down
  - Go fill in the pieces you didn't write down
  - (Repeat)

NANOG™

***Precisely and accurately**

Initial Iteration of VNL Process (1)

- Choose a list of devices to model
- Gather run-state of each device
- Loop through the devices while
  - Sanitizing configuration for non-prod/lab environment
  - Converting configuration details from physical form-factor to virtual-form-factor

NANOG™

***Precisely and accurately**

Initial Iteration of VNL Process (2)

- Create a new lab/project on the virtualization platform
- Loop through devices' converted run-states while
  - Instantiating a virtual instance of the device in the VNL project
  - Pushing the scrubbed run-state into each device

Initial Iteration of VNL Process (3)

- Loop though the physical links inferred from the devices' run-states while
  - Instantiating a virtual-link in the VNL which terminates on the devices/interfaces that correspond to the production devices/interfaces

Case study: Brass Tacks

04-FEB-2025

NANOG™

Project Structure

- Source hosted on Github
- Published with modified BSD license
- Used pyproject.toml (PEP518) for build-specs
  - With setuptools_scm for version mgmt
- Docs hosted on Github pages
  - Used Sphinx for documentation
  - With autoapi for autodocumentation of code from Python docstrings and typehints
- Package published to pypi.org

Python Modules in Package

- `data_classes`
  - Where data structures ("Switch", "Connection") are defined
- `ptovnetlab`
  - The packages entry-point module
- `arista_poller`
  - Collection of run-state data from Arista switches
- `arista_sanitizer`
  - Conversion of physical run-state to virtual form-factor
- `gns3_worker`
  - Creation/population of the virtual-lab on a GNS3 server
- `ptovnetlab_cli`
  - Entry-point for CLI/script usage of the package

Lessons Learned

04-FEB-2025

NANOG™

Python asycnio/aoihttp

- If you're doing a long list of things that are I/O-bound, asyncio/aoihttp are critical
- They bring cooperative multitasking into Python and into HTTP sessions in particular
- Tasks within a single socket/session/connection can all share the same execution context / task-list
- Different sessions/connections have to run in their own threads (using asyncio's "to_thread" method) if you want them to execute semi-concurrently

NANOG™

Python asycnio/aoihttp (2)

- We were able to use a single HTTPS session object for the vast majority of the API calls to GNS3:
- And we were able to run them as three serialized groups of parallel-executing tasks
  - <u>Step 1</u>: Create the nodes via GNS3 API
  - <u>Step 2</u>: Push configs to nodes via Docker API
  - <u>Step 3</u>: Create inter-device links via GNS3 API

dockerd and Volume Access

- We wanted to get the startup configuration into the virtual device through API calls
- Docker API will let you write a file to a container's file-system
- But you have to provide the data as a byte-stream container a tar archive (the contents of the archive get written to the container's FS)
- Fun with ascii encoding, io.BytesIO, and tar Python modules.

dockerd and Volume Access (2)

- Docker will only let you write to the container's *root* volume when the container is not running.
- We needed to write the configuration files to '*/mnt/flash/*'
- We had to *start* the container to get it to mount its configured volume ('/mnt/flash')
- Once it was running, we could use the Docker API to write the file where we wanted
- And then use the Docker API to stop the container

Woopsie When Docker Runs as Root

- We noticed that we couldn't run more than a dozen or so cEOS containers at once
- cEOS containers were inadvertently setting the *host's* *fs.inotify.max_user_instances* value to 1256.
- That's fine for a single cEOS instance, but not if you're trying to start-up dozens of them at once
- We wound up altering the value in the cEOS docker image itself prior to creating the GNS3 templates
- In the '99-eos.conf' file within the image

Sphinx For Documentation

- Powerful/flexible tool for building documentation
  - auto_api extension is great!

`arista_sanitizer.applySysMac`(*switchConfigIn, sysMacIn*) [source]

Construct and append the event handler configuration to to apply the system-mac-address of the modeled switch to the cEOS container's configuration

Parameters:
- **switchConfigIn** (*list*) – List of lines of a switch's configuration
- **sysMacIn** (*str*) – The system MAC address of the original switch

Returns: **switchConfigIn** – List of lines of a switch's configuration

Return type: list

Previous

Github Actions Are Your Friend

- We used one GH actions workflow to auto-build/publish documentation on every commit
- We used a second GH action workflow to publish the package to PyPi on every GH "release"
  - Inferring version number from GH release tags
- Packaging/publishing workflow LOE approaching zero; let us concentrate on the code itself

What Next?

04-FEB-2025

NANOG™

Core Functional Enhancements

- Integrate additional virtualization platforms and NOSes
- Add “generic” nodes as stand-ins for un-managed devices
  - Mimic their behavior as best as possible
  - (routing/STP/LLDP neighbors’ observable properties)
- Add option to populate virtual-devices with configuration built from network source-of-truth instead of actual run-state
- Additional front/back-end integrations
  - Nautobot(?)
  - Standalone Django app(?)

NANOG™

Build A Community

- Find like-minded folks to accelerate development of the package
- Get more users (in more organizations) to adopt/use the tools

Questions?

04-FEB-2025

NANOG™

Where to engage

- ptovnetlab
  - github repository
  - documentation
- netbox_ptov
  - github repository
  - documentation
- And/or find me at
  - https://menckend.github.io/alpha
  - https://github.com/menckend

Thank you

04-FEB-2025

NANOG™

Preceding Slide Left Intentionally Blank

04-FEB-2025

NANOG™

More Detail Than Time Permits…

04-FEB-2025

NANOG™

data_classes Module

04-FEB-2025

NANOG™

Data Class: Switch

```
@dataclass
class Switch:
    name: str
    model: str
    eos_version: str
    system_mac: str
    serial_number: str
    lldp_system_name: str
    lldp_system_name: str
    ethernet_interfaces: int
    gns3_template_id: str
    docker_container_id: str
    gns3_node_id: str
    vendor_platform: str
    qemu_vm_id: str
    config: list =[]
```

NANOG™

Data Class: Connection

```
@dataclass
class Connection:
    switch_a: str # LLDP system name
                  # of the first switch
    port_a: str   # Port on the first
                  #switch
    switch_b: str # LLDP system name of
                  # the second switch
    port_b: str   # Port on the second
                  #switch
```

NANOG™

ptovnetlab Module

04-FEB-2025

List: `image_map`

- *n*x2 array/list
- `n[0]` = The GNS3 UID of the template
- `n[1]` = The “tag” of the Docker image used by the GNS3 template

NANOG™

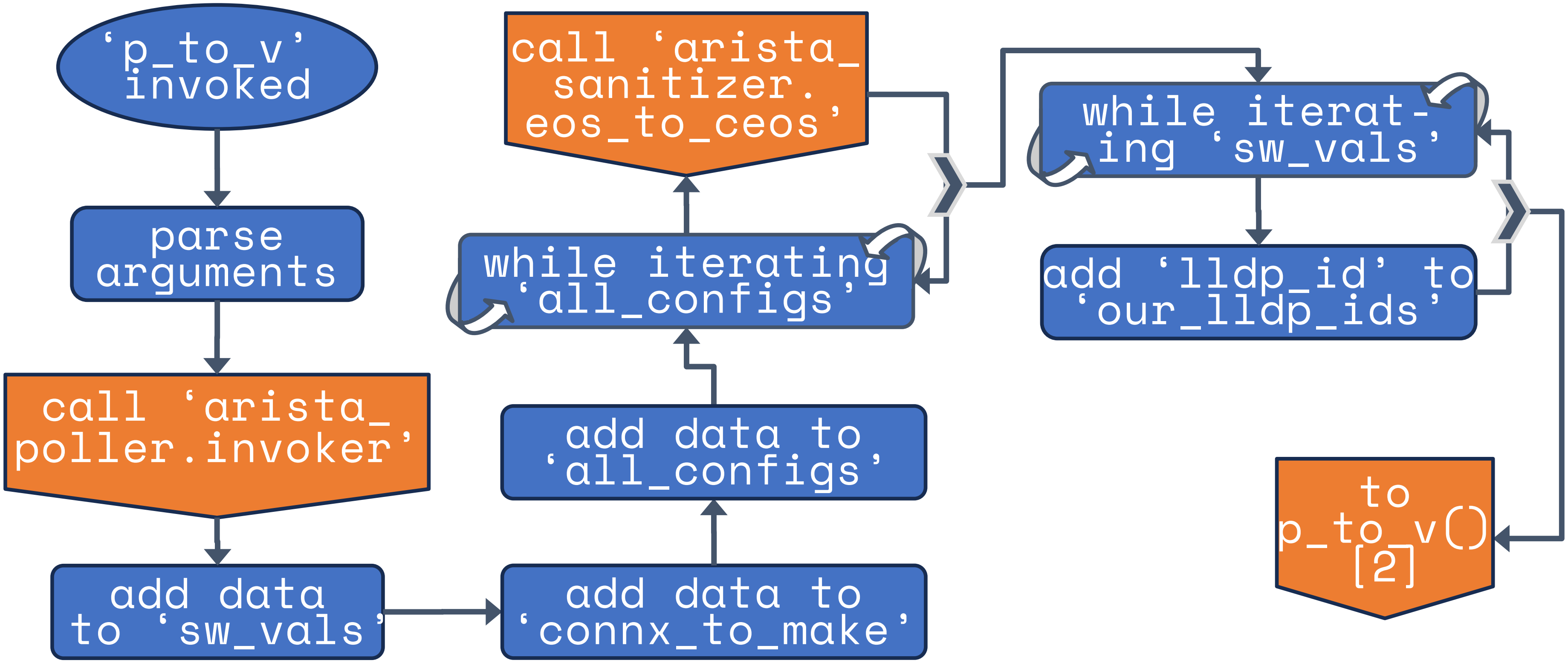
`p_to_v` Function (1)

- Receive/collect input
- Call `arista_poller.invoker`
  - Retrieve Switches and Connections
- Call `arista_sanitizer.eos_to_ceos`
  - Retrieve converted runstate data
- Remove duplicate (inverted) entries from Connections
- Retrieve all existing template objects from GNS3server

`p_to_v` Function (2)

- Iterate through templates from GNS3server and add all docker templates `image_map` list
- While iterating through `Switches`, iterate through `image_map`, and write the UID of the matching GNS3 template to the `gns3_template_id` for each `Switch`
- Create the new project in GNS3server
- Call `gns3_worker.invoker`
- Close the new gns3 project
- Return the URL of the new project

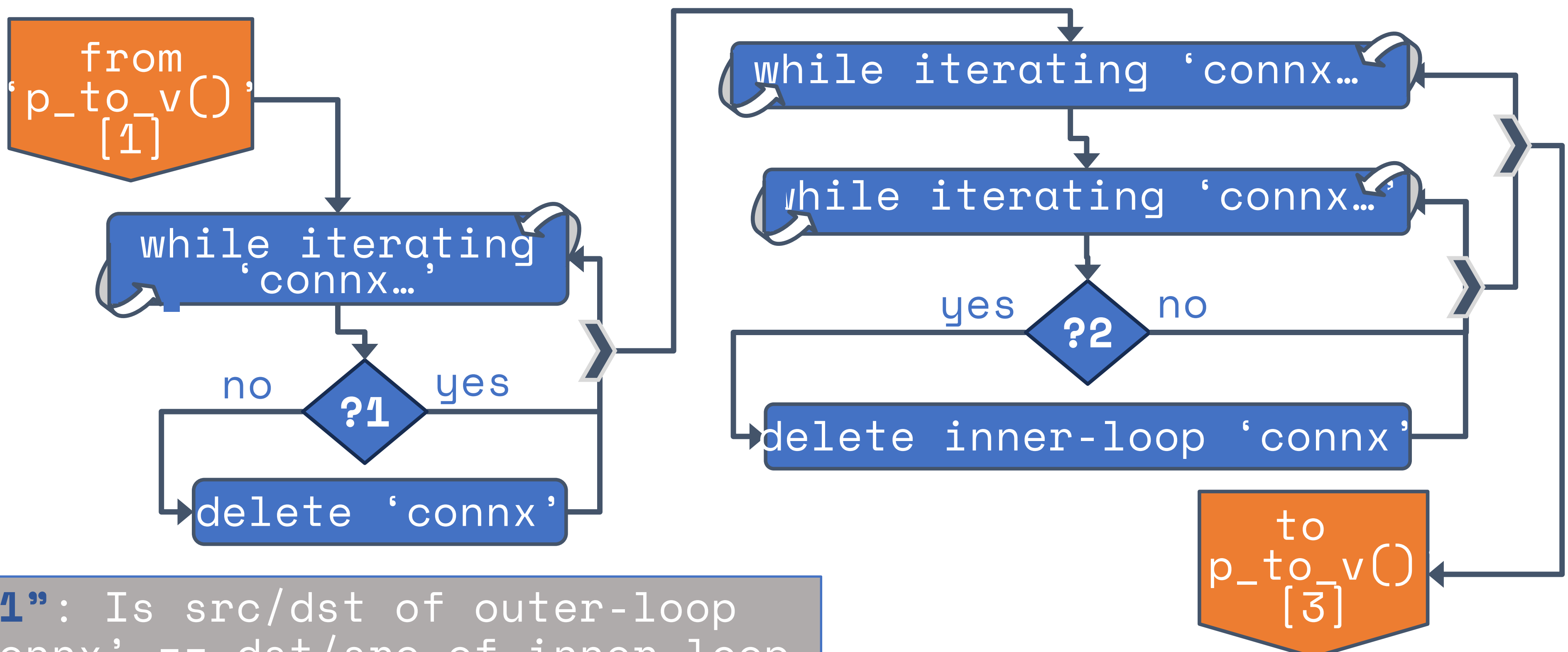
p_to_v() [1]



'p_to_v' invoked
parse arguments
call 'arista_ poller.invoker'
add data to 'sw_vals'
add data to 'connx_to_make'
add data to 'all_configs'
while iterating 'all_configs'
call 'arista_ sanitizer. eos_to_ceos'
while iterat- ing 'sw_vals'
add 'lldp_id' to 'our_lldp_ids'
to p_to_v() [2]

NANOG™

p_to_v() [2]

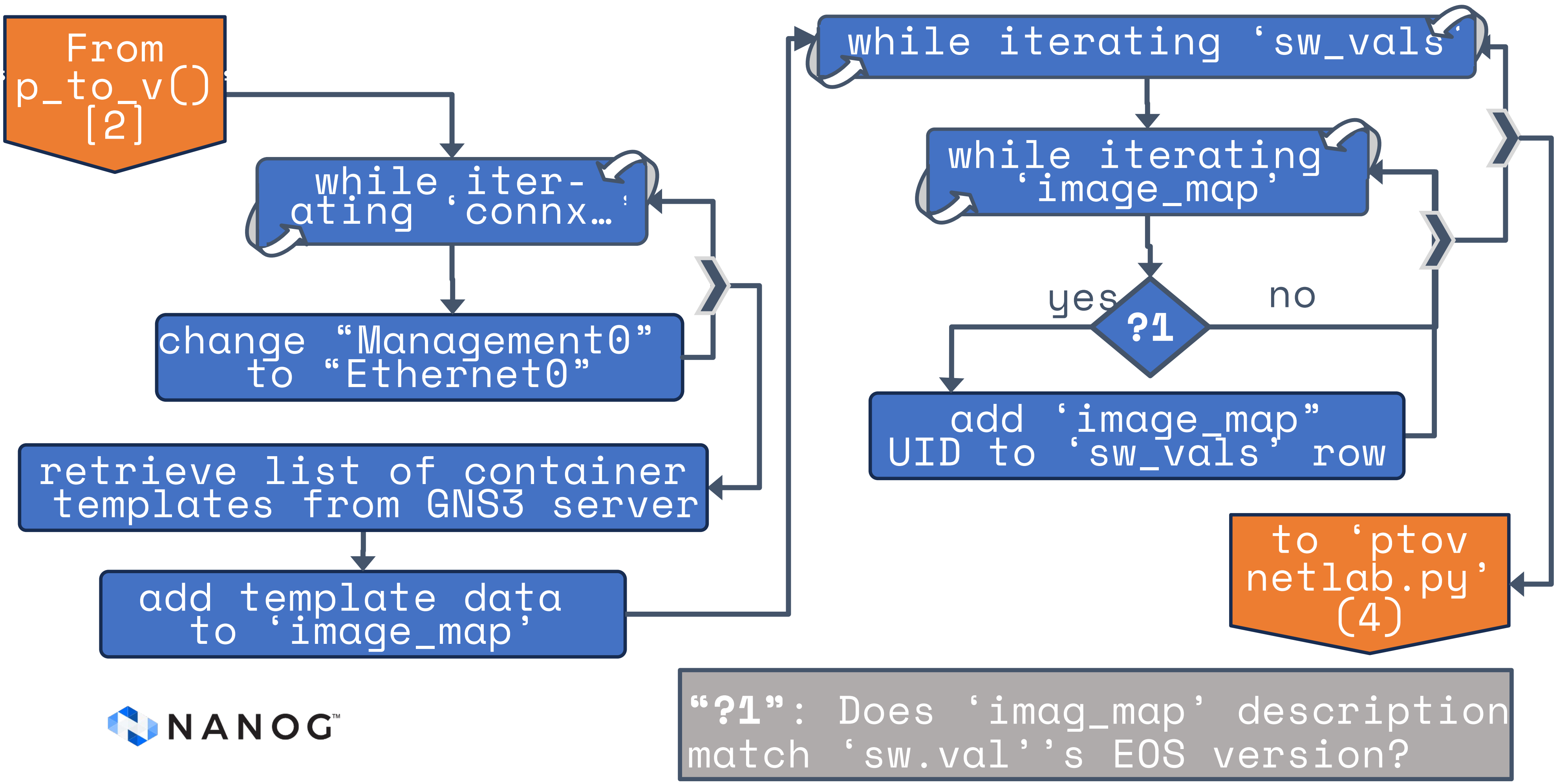


“?1”: Is src/dst of outer-loop ‘connx’ == dst/src of inner-loop ‘connx’?

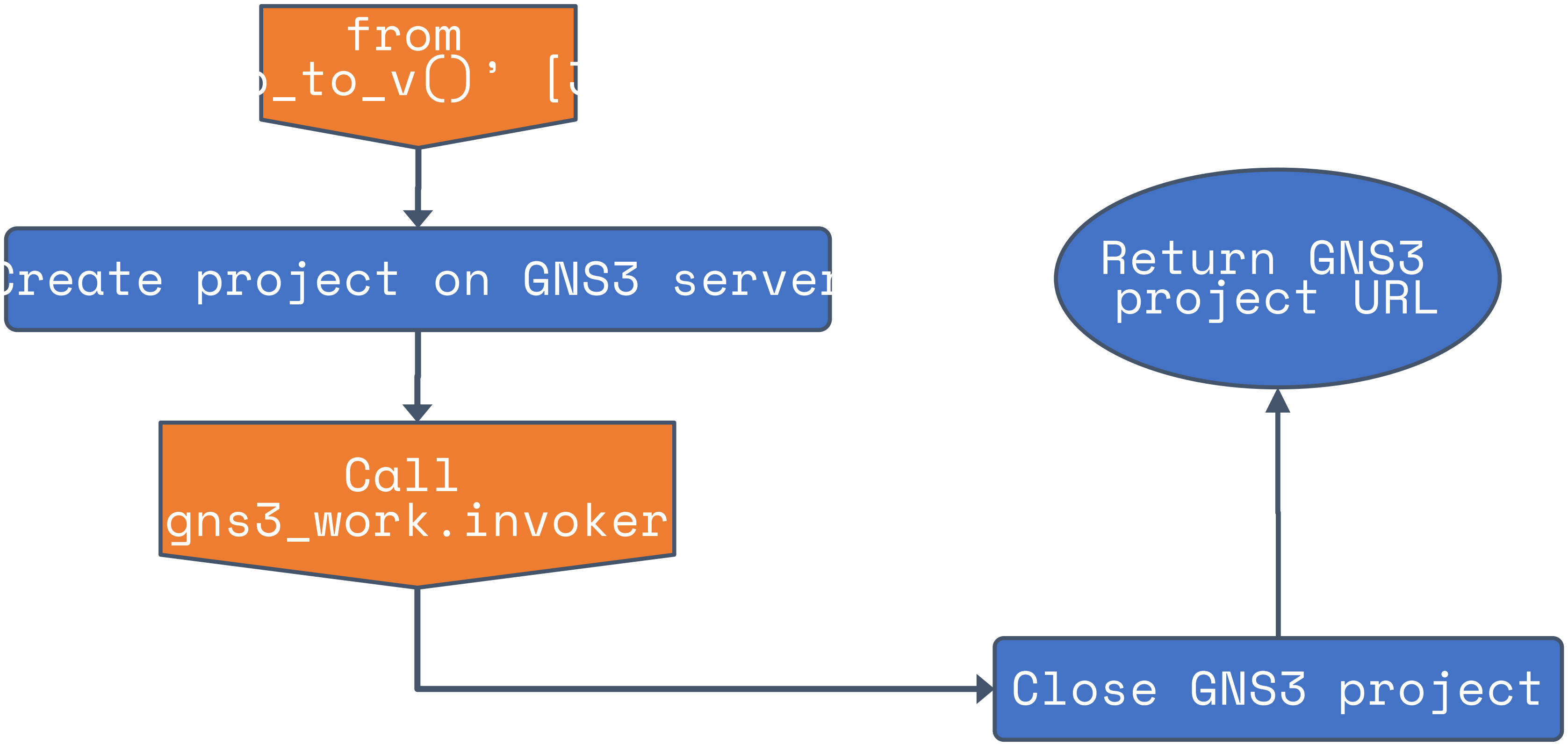
“?2”: Is either ‘lldp_id’ from ‘connx’ in ‘our_lldp_ids’?

NANOG

p_to_v() [3]


From 'p_to_v()' [2]
while iterating 'connx…'
change "Management0" to "Ethernet0"
retrieve list of container templates from GNS3 server
add template data to 'image_map'
while iterating 'sw_vals'
while iterating 'image_map'
yes
?1
no
add 'image_map" UID to 'sw_vals' row
to 'ptov netlab.py' (4)
"?1": Does 'imag_map' description match 'sw.val''s EOS version?

p_to_v() [4]


from
_to_v()' [
Create project on GNS3 server
Call
gns3_work.invoker
Close GNS3 project
Return GNS3
project URL

arista_poller Module

04-FEB-2025

NANOG™

‘invoker’ Function

- Entry-point for the module
- Accepts list of device names and user/pwrd credentials as arguments
- Calls arista_poller’s “main” function via asyncio.run
- Returns initial set of values for `Switch` properties
- Returns list of switch running configurations
- Returns list of discovered LLDP adjacenceis

`main` Function (async)

- Creates asyncio loop
- Sets asyncio loop's executor to spawn up to 20 threads
- Loops through the list of switches while
  - Creating a new asyncio loop "task" to the asyncio loop
  - Using the asyncio.to_thread method
  - Calling the `get_sw_data` function
- Waits for all of the asyncio tasks to finish
- Returns the switches' data to the invoker function

‘get_sw_data’ Function

- Establishes EAPI connection to the Arista switch
- Collects output of:
  - `node.enable(("show version", "show lldp neighbors", "show lldp local-info"), format="json")`
- Collect output of `node.running_config.splitlines`
- Return the collected data (as updated `Switch` objects)

arista_sanitizer Module

04-FEB-2025

NANOG™

badstarts List

List of strings that indicate a line in a Switch configuration needs to be commented out ("#" prepend) if they begin the line

```
radius
username
aaa
ip radius
hardware speed
queue
server
ip radius
ntp server
daemon TerminAttr
    exec /usr/bin/TerminAttr
```

NANOG™

`count_ether_interfaces` Function

- Loop through lines of the switch configuration while
- Counting the number of lines that start with "interface Ethernet…"
  - Excluding non-initial interface of any breakout interfaces

apply_sysmac Function

- Adds the following lines to the end of switch configuration:

```
'event-handler onStartup'
' trigger on-boot'
' action bash'
'  echo $var_sysmac >
/mnt/flash/system_mac_address'
'  truncate -s -1
/mnt/flash/system_mac_address'
'  EOF'
```

NANOG™

`eos_to_ceos` Function

- Module entry-point function
- While looping through the lines of the switch configuration:
- Changes any Management interface numbers to “0”
- Comments out any lines that lead with “badstarts” entries
- Comments out any configuration sections for breakout interfaces with indices != 0 (Ethernet24/2), etc.
- Calls `apply_sysmac_in` function
- Returns the sanitized/converted configuration

gns3_worker Module

04-FEB-2025

NANOG™

`invoker` Function

- Module entry point
- Arguments include servername, URL, sw_vals, allconf_in, prjid_in, connx_in
- Calls `gns3_nodes_create_async`
- Calls `gns3_connx_create_async`
- Returns GNS3 HTTP response code from connection creation requests

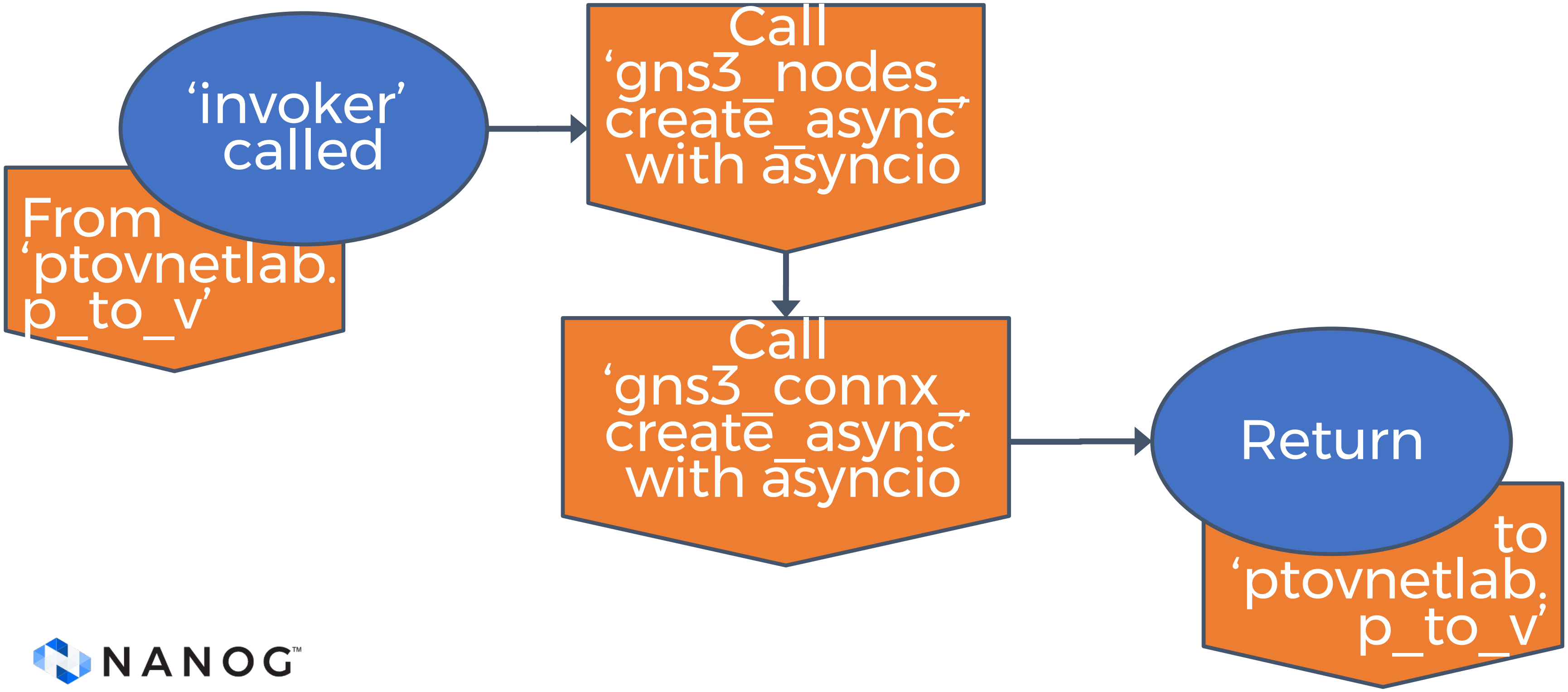
`gns3_nodes_create_async` Function

- Creates an aoihttp ‘async’ context (of an HTTPS session to the GNS3 server) within which it loops through all of the switches to be modeled
- In each iteration of the loop, it *synchronously* makes calls to GNS3:create a temporary template with the right number of interfaces; create a node using the template; delete the temporary template; start the node

gns3_nodes_create_async [2]

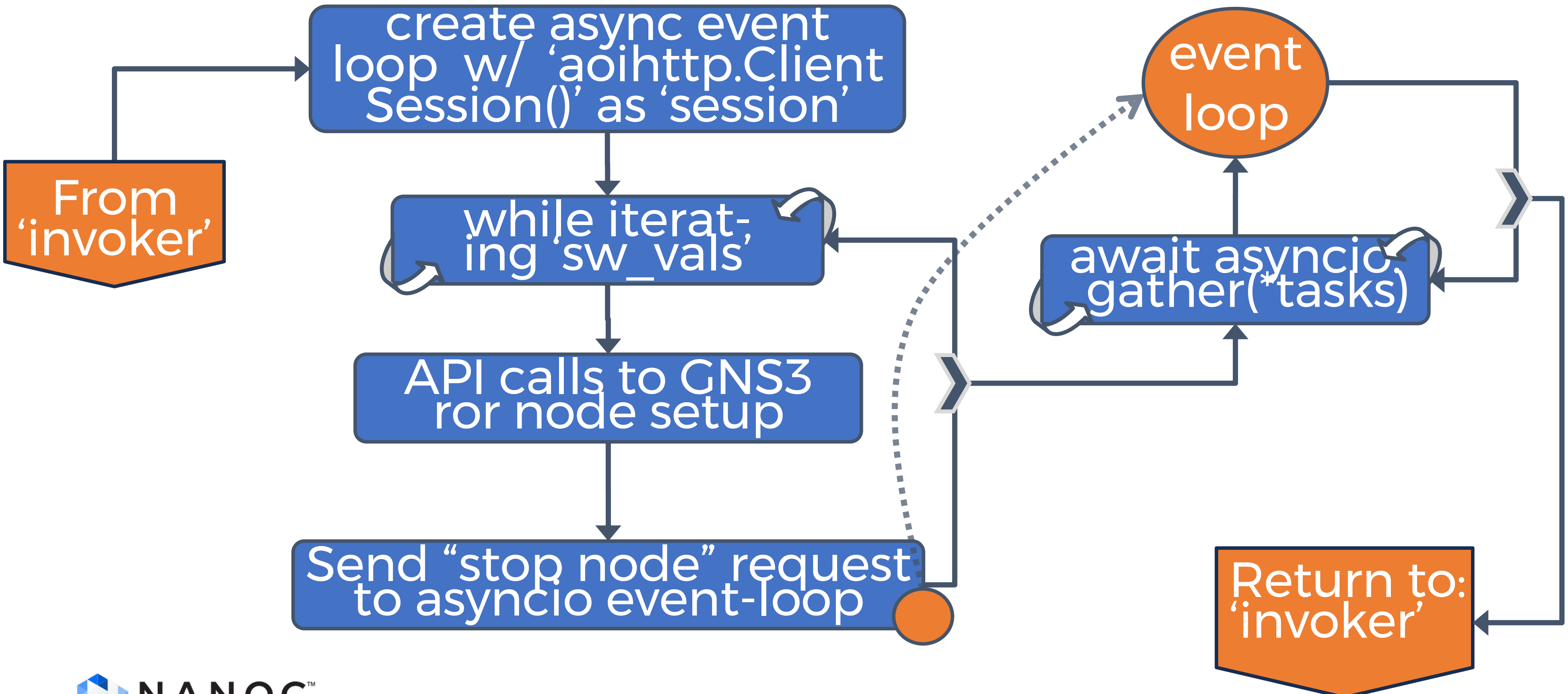
- Synchronously makes a call to the Docker API to copy the switch configuration to the container's filesystem (at /mnt/flash/startup.config)
- Adds a task to the asyncio loop, making an API call to the GNS3 server to stop the node.
  - This allows the module to move on to the next container, instead of waiting 5-20 seconds for GNS3 to confirm that the container has stopped.

invoker()

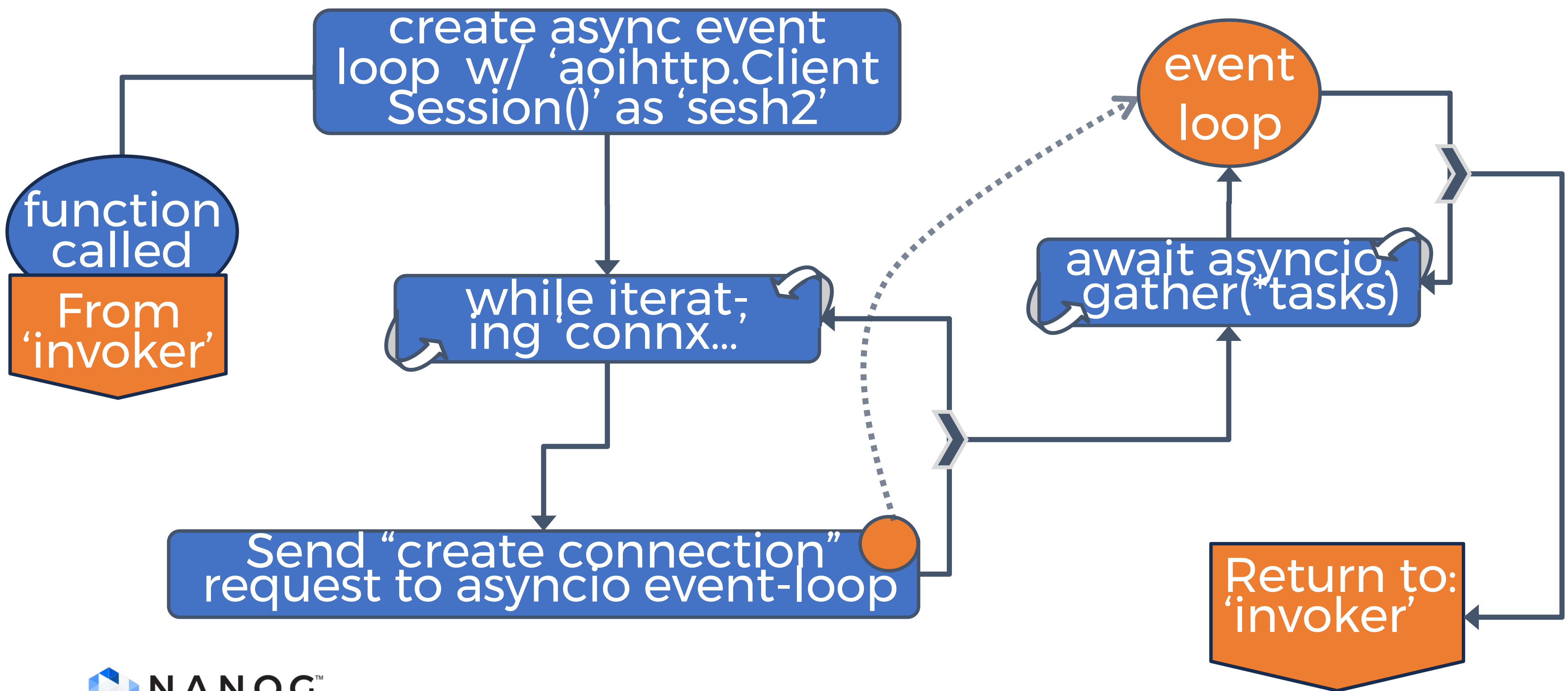


NANOG™

gns3_nodes_create_async()



gns3_connx_create_async()



NANOG™