



Wi-Fi automation lab

Lab Guide - Day 1

Andreas Koksrud / Telenor Bedrift

Co-authors:

Kjetil Teigen Hansen & François Vergès

References / inspiration

<https://github.com/CiscoDevNet/yangsuite/>

<https://postman.com>

https://docs.ansible.com/ansible/latest/collections/cisco/ios/ios_facts_module.html

<https://www.wifireference.com/2020/01/14/viewing-network-telemetry-from-the-catalyst-9800-with-grafana/>

<https://grafana.com/grafana/dashboards/13462-device-health-monitoring/>

<https://grafana.com/grafana/dashboards/12468-catalyst-9800-client-stats/>

<https://wirelessisfun.wordpress.com/2020/12/10/network-telemetry-data-and-grafana-part-1-the-advanced-netconf-explorer/>

<https://python.org>

<https://codeium.com>



Copyright

© Andreas Koksrud 2025. All rights reserved. This presentation is provided for educational and informational purposes only. You may distribute and learn from this presentation, but commercial use or any form of monetization is strictly prohibited without prior written consent

Any slides marked Kjetil Teigen Hansen or the Conscia logo will also have full or co-ownership by Kjetil

Any slides marked François Vergès or the SemFio logo will also have full or co-ownership by François



Prerequisites

- Cisco Meraki account (<https://dashboard.meraki.com>)
- Juniper MIST account (<https://manage.mist.com>)
- Postman account (<https://postman.com>)
- Complete the pre-lab exercises before the deep dive labs
- Bring an Ethernet dongle if you don't have built-in port
- (optional) Bring an extra screen to show lab guide (or prepare to Alt-Tab)



Communications

- WebEx space: Wi-Fi automation lab
- Please help each other
- Sharing is caring ☺



Agenda

Pre-lab

- Choose your hypervisor
- Install Ubuntu Server w/Docker
- Install Postman
- Install VS Code
- (optional) install 9800-CL

Day 1

- Sort out pre-lab task problems
- Get to know the lab environment
- Connect VS Code to Ubuntu
- Install and explore Ansible
- Explore Python automation
- Install and explore YANG Suite
- Explore Postman
- Install and explore Grafana

Day 2

- In-depth explore a topic of choice
 - Grafana / TIG-stack
 - Grafana Cloud
 - Ansible
 - Python
 - Cloud vendor automation (MIST or Meraki)

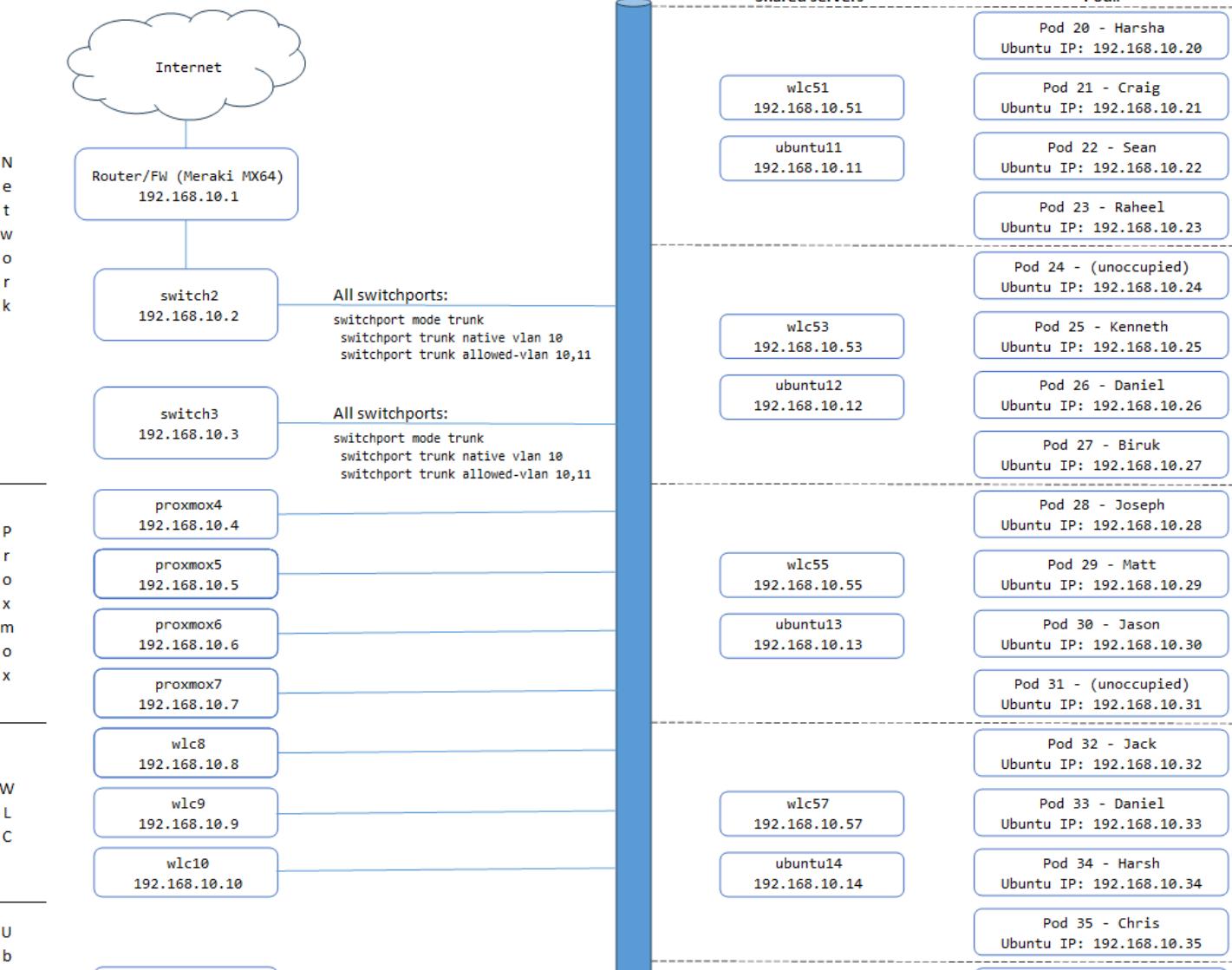


Scope

- In scope
 - Getting started with various systems/languages/solutions for lab purposes
 - Set up your own Ubuntu Linux server on your own laptop. It will be possible to use a shared server if you do not want or have possibility to install an Ubuntu VM on your laptop
 - Some nice examples to try various aspects of automation
 - Inspiring you to explore deeper on your own
- Out of scope topics
 - Git
 - Learning the languages (Ansible, Python, InfluxQL, etc)
 - Learning Linux
 - Deploying the systems for production use
 - Troubleshooting WLC/AP connection



Wi-Fi Automation deep dive - Lab topology



- Each student have an assigned Pod number. When using static IP, their Ubuntu Server should have the same last octet as the Pod#
- 2 students share a preconfigured WLC, as assigned in the topology map
- Everyone is connected to VLAN 10
- All switchports on all switches are trunk ports with VLAN 10 as native, and allowed vlan as VLAN 10 and 11
- For Wi-Fi clients on your SSIDs you can use VLAN 11 to not fill up VLAN 10

Login to shared devices

User: devnet-adm
Pass: ChangeMe2025!

IP Plan (VLAN 10)

Static adm IPs: 192.168.10.1 - 19
Ubuntus (per pod): 192.168.10.20 - 50
WLCs (shared): 192.168.10.51 - 70
DHCP range: 192.168.10.71 - 250

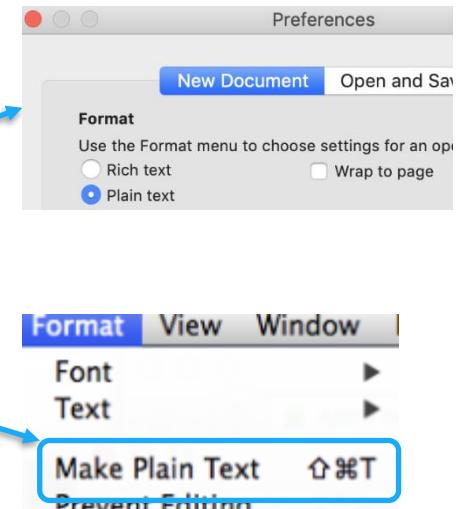
IP Plan (VLAN 11)

Static adm IPs: 192.168.11.1 - 10
DHCP range: 192.168.10.11 - 250



Notes for Mac users - TextEdit

- TextEdit (the default text editor on OS X) use rich text formatting by default. This is not a great idea when copy-pasting stuff to the command line, so there is a couple of things you must keep in mind
 - If you want, you can change a setting in TextEdit to make all new documents plain text
 - To change the current document to plain text go to Format -> Make Plain Text
 - Shortcut is: Shift-Cmd-T



Building your own lab

- Based on feedback from running this deep dive on previous events, if you are going to build and troubleshoot the full lab setup, it will take most or all of the time (2x 3h)
- The deep dive itself will have pre-built and pre-configured+tested WLAN Controllers (1 WLC per 2-4 students) for you to use. If you want to install your own WLC by following the optional part of this pre-lab guide that is OK, but we strongly encourage you to use the deep dive time to do actual automation exercises and not troubleshooting your WLC
- There will also be pre-built Ubuntu servers to use, running on our servers, for those who want to use that instead of their own laptop.
- If you have a mini computer or an old laptop to spare, you could install Proxmox on that and use for your VMs. Unfortunately it is outside the scope of this deep dive and this pre-lab guide to give detailed instructions on the Proxmox installation itself, outside what is described in task 1d. It is also outside the scope of the deep dive to help troubleshoot your Proxmox. But our servers run on that, so we are somewhat familiar with it and might answer some simple questions 😊
- We will also provide pre-built images with a finished installation of Ubuntu Server for Hyper-V and Virtualbox
- We encourage you to install your own Ubuntu Server, since it will give you more experience building the items yourself when you get home or need it in some other situation.



Tentative schedule for day 1

- Hour 1
 - Introductions and getting started
 - Finish the pre-lab tasks
 - Get to know the lab environment
 - Install and explore Ansible
- Hour 2
 - Explore Python automation
 - Explore YANG models
 - Explore Postman
- Hour 3
 - Install and explore TIG-stack / Grafana



Lab exercise #0: Finish the pre-lab tasks

- Get help to whatever you did not finish
- There will probably (always) be some cases which falls outside what is tested



Lab exercise #1: Connect your laptop and Ubuntu

- Connect your laptop to the wired network
 - Ensure you get DHCP in the range 192.168.10.71-250 /24
 - Ensure you get internet access through the wired connection
- Boot up your Ubuntu VM
 - If the Ubuntu have DHCP configured, follow the next slide to change it to static IP



Change from DHCP to static IP

- When you come to WLPC you would need to use a static IP in the lab range. Before WLPC you might just keep the server in DHCP mode, and come back to this slide during the deep dive.
- This will show one method to permanently change the IP of your server
 - Check your adapter name

```
devnet-adm@ubuntu-devnet:~$ netplan status | grep network
 2: ens18 ethernet UP (networkd: ens18)
```

2. Create a new file /etc/netplan/99_config.yaml

```
devnet-adm@ubuntu-devnet:~$ sudo nano /etc/netplan/99_config.yaml
devnet-adm@ubuntu-devnet:~$ sudo chmod 600 /etc/netplan/99_config.yaml
```

3. Delete the auto-created /etc/netplan/50-cloud-init.yaml

```
devnet-adm@ubuntu-devnet:~$ sudo rm /etc/netplan/50-cloud-init.yaml
```

4. Run "sudo netplan apply" to use the new config file

```
devnet-adm@ubuntu-devnet:~$ sudo netplan apply
```

5. Check that you have the new IP

```
devnet-adm@ubuntu-devnet:~$ sudo netplan apply
```

6. Reboot the server and you're good to go ☺

```
devnet-adm@ubuntu-devnet:~$ sudo reboot now
```

!!! In YAML, indentation is REALLY important !!!
In this file I use 2, 4, 6 or 8 spaces (multiple of 2).
It would be valid to use multiple of 3 or 4 instead.

Change to YOUR
Ubuntu IP

Get this value from
"netplan status". Enter
it into 99_config.yaml
Probable values:
- Proxmox: ens18
- Hyper-V: eth0
- VirtualBox: enp0s3

```
/etc/netplan/99_config.yaml
network:
  ethernets:
    ens18:
      #      dhcp4: true
      dhcp6: false
      dhcp4: false
      addresses:
        - 192.168.10.{YOUR_POD_IP}/24
      routes:
        - to: default
          via: 192.168.10.1
      nameservers:
        addresses:
          - 1.1.1.1
```

Most Common errors:

- Wrong file permissions (step 2)
- Errors in the indentation in the YAML file

To easily change between static and DHCP, I just comment/uncomment the lines. To change this to DHCP, just uncomment the "dhcp4: true" line, and comment the rest



Lab exercise #2: Test the lab environment

1. Access from your computer to the devices

- SSH to one or more of the shared Ubuntu servers (192.168.10.11 up to .18)
- SSH and HTTPS to your designated WLC (192.168.10.51, 53, 55, etc) shared with 3 other students
- You have admin access to all shared devices, please don't mess up for each other ☺

2. Access from your Ubuntu Server to the devices

- Ping from your Ubuntu VM to your designated WLC
- Ping from your Ubuntu VM to the Internet

3. Access in to your Ubuntu Server

- Ping from your designated WLC to your Ubuntu VM



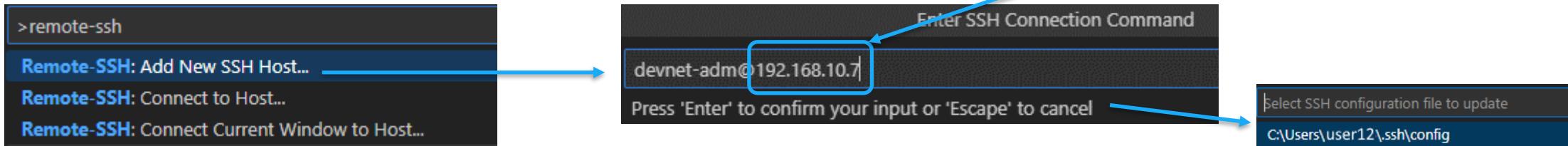
Lab exercise #3: Explore VS Code

- We will now use our VS Code installation to open a directory on the Ubuntu server
- Instructions will follow to
 - Open the "Remote-SSH" extension
 - Create new host
 - Connect current window to remote host
 - Install local extensions in remote host
 - Open a remote folder as workspace
 - Open some example files from the downloaded GIT repository



VS Code - Connect to... Remote SSH

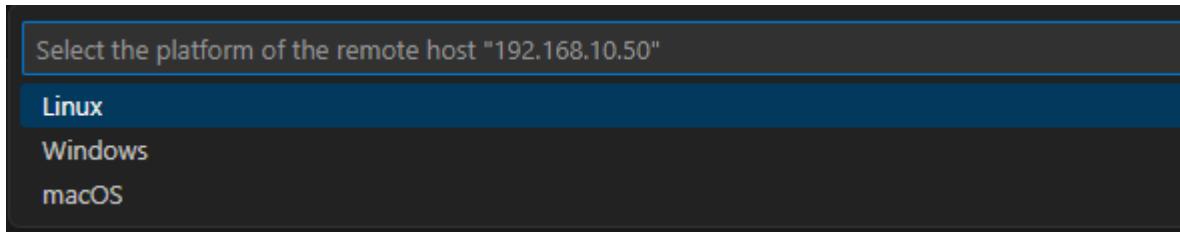
- Command Palette (F1)
 - Type "remote-ssh", select "Add New SSH Host...", type "devnet-adm@192.168.10.7"



- Connect to the new Host, either using lower left corner of VS Code window or Command Palette (F1)

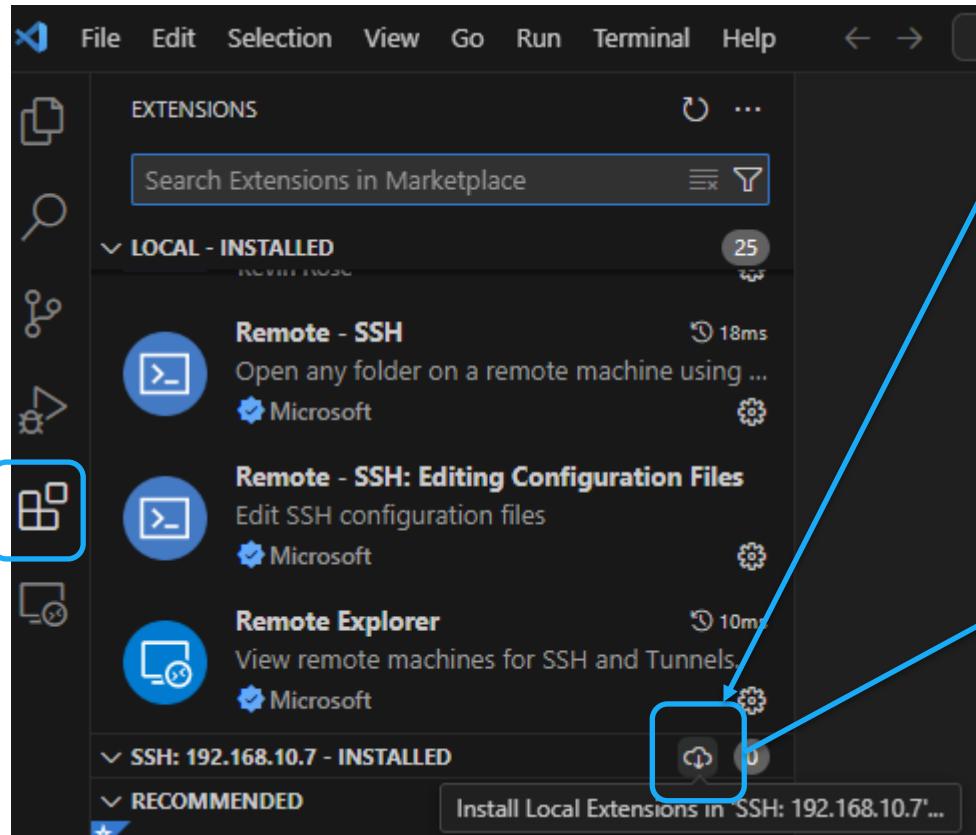


- When asked about platform type, select Linux

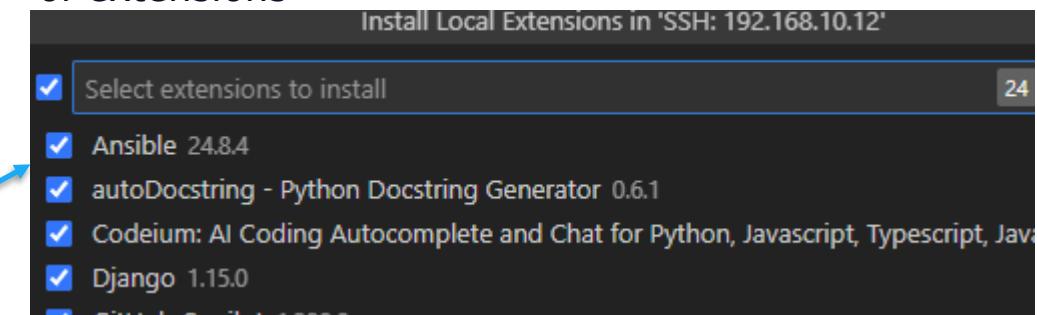


VS Code - Install Extensions in Remote SSH

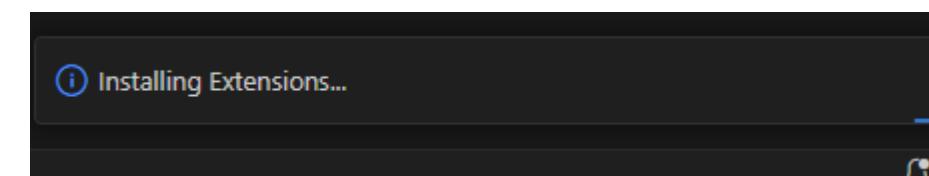
- The extensions have to be installed in each remote environment



When asked which ones to install, you can select everything, or select some of them if you have a lot of extensions

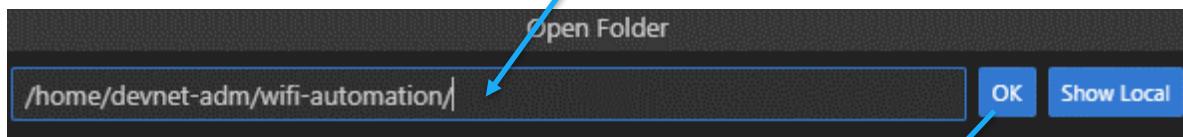
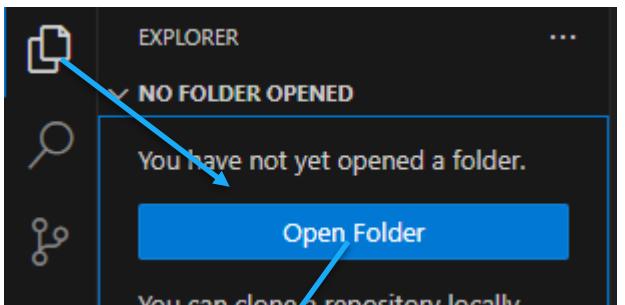


It will take a minute or so to install the extensions in the server

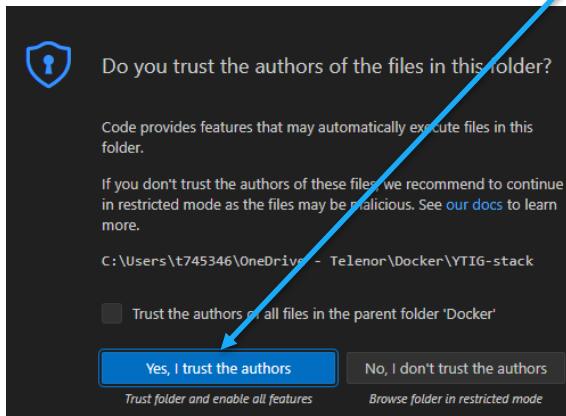


VS Code

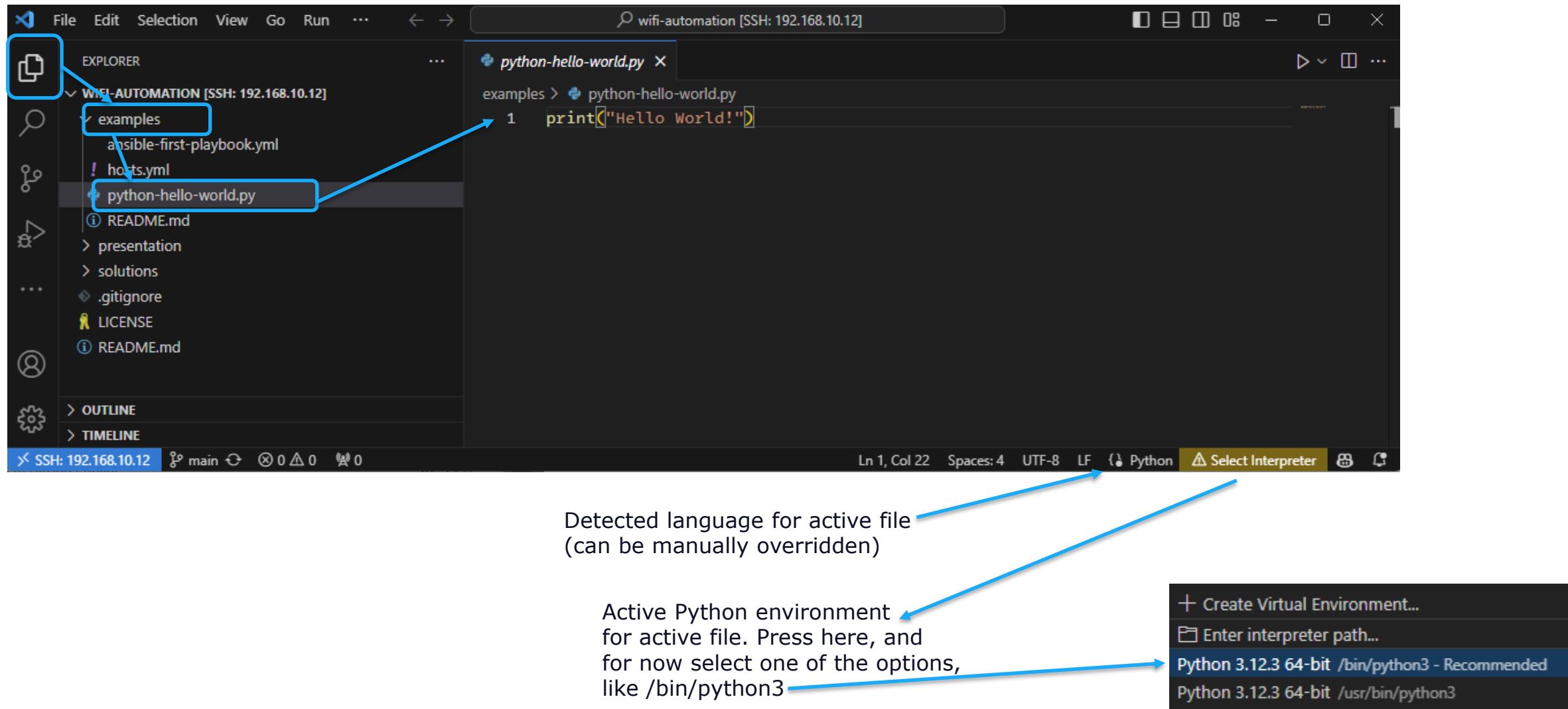
- Open Folder



- Trust authors



- Using Explorer in VS Code, open the example file `python-hello-world.py`



- If you open the Ansible file `ansible-first-playbook.yml` some errors will show. We will fix this later in the lab, after installing Ansible

The screenshot shows the Visual Studio Code interface. The left sidebar (Explorer) displays a project structure for "WIFI-AUTOMATION [SSH: 192.168.10.7]". The "examples" folder contains files like `hosts.yml`, `python-hello-world.py`, and `ansible-first-playbook.yml`. The main editor window shows the content of `ansible-first-playbook.yml`:

```
1  ---
2
3  - name: My first play
4    hosts: all
5    tasks:
6      - name: Ping my hosts
7        ansible.builtin.ping:
8
9      - name: Print message
10     ansible.builtin.debug:
11       msg: Hello world
```

A blue arrow points from the text above to a tooltip at the bottom right of the editor window:

Ansible not found in the environment
Python version used: 3.12.3 from /bin/python3

The status bar at the bottom of the screen shows the connection details: "SSH: 192.168.10.7", the file name "main*", and various status icons.



Terminal in VS Code

- VS Code has an integrated terminal you can use. In many cases this is a convenient way to SSH to your server when working on projects in VS Code
- Select "Terminal" and "New Terminal" You can also use the shortcut (sorry again for the stupid Norwegian characters ☺)
- The code window will split, and you will have a terminal in the lower half (I usually use a larger window than in this screenshot, it's just resized to show it all on the same screenshot)

The screenshot shows the VS Code interface with the following details:

- Top Bar:** Shows "Terminal Help". A blue arrow points from the text "Select 'Terminal' and 'New Terminal'" to the "Terminal" menu item.
- Terminal Shortcut:** "Ctrl+Shift+ø" is shown next to the "New Terminal" option.
- Code Editor:** An Ansible playbook named "ansible-first-playbook.yml" is open. The content is as follows:

```
1  ---  
2  - name: My first play  
3    hosts: all  
4    tasks:  
5      - name: Ping my hosts  
6        ansible.builtin.ping:  
7  
8      - name: Print message  
9        ansible.builtin.debug:  
10       msg: Hello world
```

- Terminal View:** The bottom half of the screen shows a terminal window titled "bash - examples". It displays the following command-line session:

```
devnet-adm@ubuntu-devnet:~/wifi-automation$ cd examples  
devnet-adm@ubuntu-devnet:~/wifi-automation/examples$ ll  
total 24  
drwxrwxr-x 2 devnet-adm devnet-adm 4096 Jan  5 15:20 ./  
drwxrwxr-x 9 devnet-adm devnet-adm 4096 Jan  7 19:14 ../  
-rw-rw-r-- 1 devnet-adm devnet-adm 177 Jan  5 15:20 ansible-first-playbook.yml  
-rw-rw-r-- 1 devnet-adm devnet-adm 275 Jan  5 15:20 hosts.yml  
-rw-rw-r-- 1 devnet-adm devnet-adm 21 Jan  5 15:20 python-hello-world.py  
-rw-rw-r-- 1 devnet-adm devnet-adm 111 Jan  5 15:20 README.md  
devnet-adm@ubuntu-devnet:~/wifi-automation/examples$
```

- Status Bar:** Shows file statistics: "Ln 11, Col 23 Spaces: 2 UTF-8 LF Ansible". It also indicates "Ansible" and "Python 3.12.3" are active.



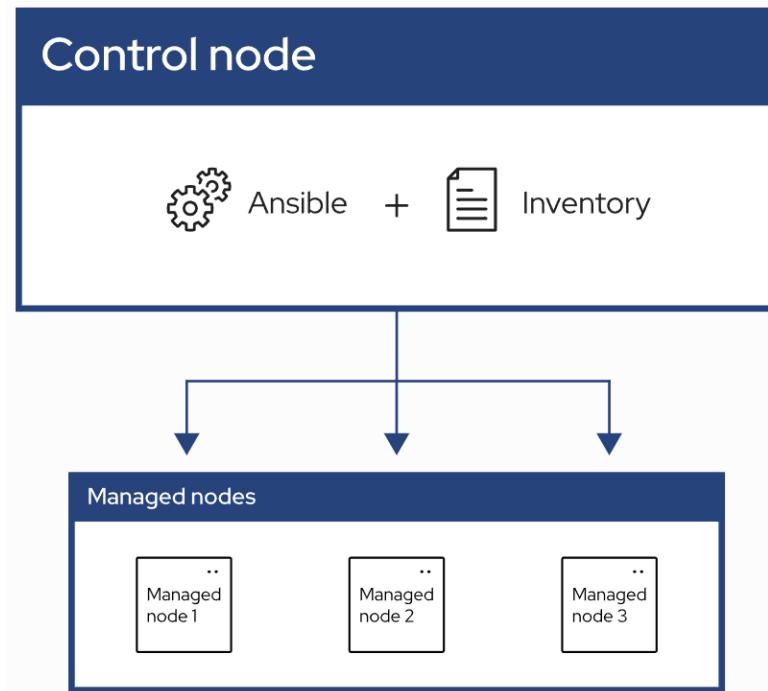
Lab exercise #4: Install Ansible

- In this exercise, we will
 - Create a Python virtual environment to use for Ansible stuff
 - Install Ansible on the Ubuntu Server
 - Configure VS Code to use Ansible on the remote server
- The first couple of slides will explain briefly about Ansible (for your reference)
- Then the installation process is explained step by step, before re-opening the first playbook example



What is Ansible

- Automation using "playbooks" written in YAML format
- Agent-less architecture
 - No installation on the managed nodes
- Idempotency
 - Does not change/write if target is already in state described by playbook
- Control node
 - System where Ansible is installed and runs the playbooks
- Inventory
 - List of managed nodes where the playbook will affect
- Managed nodes
 - Remote systems that are the targets of your playbooks



Plugins and modules

- Plugins
 - Extends the core functionality of Ansible
 - Examples

- cisco.ios.ios cliconf - To run CLI commands on Cisco IOS device
- Become plugins, how to get write access on various systems ("sudo", "enable" etc)
- Connection plugins, like SSH

- Modules
 - Built-in or custom code snippets to perform various tasks
 - A special type of plugin
 - Examples
 - cisco.ios.ios_config
 - cisco.ise.endpoint

```
- name: Configure interface settings
cisco.ios.ios_config:
  lines:
    - description test interface
    - ip address 172.31.1.1 255.255.255.0
  parents: interface Ethernet1
```

```
- name: Delete by id
cisco.ise.endpoint:
  ise_hostname: "{{ise_hostname}}"
  ise_username: "{{ise_username}}"
  ise_password: "{{ise_password}}"
  ise_verify: "{{ise_verify}}"
  state: absent
  id: string
```



Ansible installation

- Installing in a Python VENV (Virtual Environment)

```
devnet-adm@ubuntu-devnet:~$ sudo apt install pip python3-venv
devnet-adm@ubuntu-devnet:~$ mkdir ~/automation-venv
devnet-adm@ubuntu-devnet:~$ python3 -m venv ~/automation-venv/
devnet-adm@ubuntu-devnet:~$ source ~/automation-venv/bin/activate
(automation-venv) devnet-adm@ubuntu-devnet:~$ pip install ansible ansible-lint
```

```
(automation-venv) devnet-adm@ubuntu-devnet:~$ ansible --version
ansible [core 2.17.1]
  config file = None
  configured module search path = ['/home/devnet-adm/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /home/devnet-adm/automation-venv/lib/python3.12/site-packages/ansible
  ansible collection location = /home/devnet-adm/.ansible/collections:/usr/share/ansible/collections
  executable location = /home/devnet-adm/automation-venv/bin/ansible
  python version = 3.12.3 (main, Apr 10 2024, 05:33:47) [GCC 13.2.0] (/home/devnet-adm/automation-venv/bin/python3)
  jinja version = 3.1.4
  libyaml = True
(automation-venv) devnet-adm@ubuntu-devnet:~$
```

- (optional info) Ansible could be installed globally. But by using VENV we can control the environment more closely. Most projects will have some dependencies, and by controlling the environment you ensure that the packages in the environment are ones that are tested to be compatible with the project. Here we just use the latest, but it is shared to list ("freeze") the currently installed packages and versions from an environment when tested OK, and use this as a requirements input when reproducing the environment



Ansible Collections

- A collection is a premade collection of playbooks, roles, modules and plugins, usually for a specific purpose or type of devices
 - <https://docs.ansible.com/ansible/latest/collections/index.html#list-of-collections>
- This is how to look up which collections you have installed in your environment

```
(ansible-venv) devnet-admin@ubuntu-devnet:~$ ansible-galaxy collection list  
# /home/devnet-admin/ansible-venv/lib/python3.12/site-packages/ansible_collections  
Collection          Version  
-----  
amazon.aws          8.0.1  
ansible.netcommon   6.1.3  
ansible.posix        1.5.4  
ansible.utils        4.1.0  
ansible.windows      2.4.0  
cni.ctc             0.0.0
```

```
(ansible-venv) devnet-admin@ubuntu-devnet:~$ ansible-galaxy collection list | grep "cisco"  
cisco.aci           2.9.0  
cisco.asa           5.0.1  
cisco.dnac          6.16.0  
cisco.intersight    2.0.9  
cisco.ios            8.0.0  
cisco.iosxr          9.0.0  
cisco.ise            2.9.2  
cisco.meraki         2.18.1  
cisco.mso            2.6.0  
cisco.nxos           8.1.0  
cisco.ucs            1.10.0  
community.ciscosmb  1.0.9  
(ansible-venv) devnet-admin@ubuntu-devnet:~$ █
```



Ansible Collections

- Navigating the documentation
- <https://docs.ansible.com/ansible/latest/collections/index.html#list-of-collections>

Collection Index

These are the collections with docs

- amazon.aws
- ansible.builtin
- ansible.netcommon
- ansible.posix
- ansible.utils
- ansible.windows
- arista.eos
- awx.awx
- azure.azcollection
- check_point.mgmt
- chocolatey.chocolatey
- cisco.aci
- cisco.asa
- cisco.dnac
- cisco.intersight
- cisco.ios
- cisco.iosxr

- [ios_vtep_global module](#) – Resource module to configure BGP.
- [ios_command module](#) – Module to run commands on remote devices.
- [ios_config module](#) – Module to manage configuration sections.
- [ios_evpn_evi module](#) – Resource module to configure L2VPN EVPI.
- [ios_evpn_global module](#) – Resource module to configure L2VPN E
- [ios_facts module](#) – Module to collect facts from remote devices.
- [ios_hostname module](#) – Resource module to configure hostname.
- [ios_interfaces module](#) – Resource module to configure interfaces.
- [ios_l2_interfaces module](#) – Resource module to configure L2 interf
- [ios_l3_interfaces module](#) – Resource module to configure L3 interf
- [ios_lacp module](#) – Resource module to configure LACP.
- [ios_lacp_interfaces module](#) – Resource module to configure LACP i
- [ios_lag_interfaces module](#) – Resource module to configure LAG int
- [ios_linkagg module](#) – Module to configure link aggregation groups.
- [ios_lldp module](#) – (deprecated, removed after 2024-06-01) Manage
- [ios_lldp_global module](#) – Resource module to configure LLDP.
- [ios_lldp_interfaces module](#) – Resource module to configure LLDP i
- [ios_logging_global module](#) – Resource module to configure logging
- [ios_ntp module](#) – (deprecated, removed after 2024-01-01) Manage
- [ios_ntp_global module](#) – Resource module to configure NTP.
- [ios_ospf_interfaces module](#) – Resource module to configure OSPF
- [ios_ospfv2 module](#) – Resource module to configure OSPFv2.
- [ios_ospfv3 module](#) – Resource module to configure OSPFv3.
- [ios_ping module](#) – Tests reachability using ping from IOS switch.
- [ios_prefix_lists module](#) – Resource module to configure prefix lists.
- [ios_route_maps module](#) – Resource module to configure route map

Examples

- ```
- name: Gather all legacy facts
cisco.ios.ios_facts:
 gather_subset: all

- name: Gather only the config and default facts
cisco.ios.ios_facts:
 gather_subset:
 - config
```

|                                                                       |                                                                                                                    |
|-----------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|
| <code>ansible_net_serialnum</code><br>string                          | The serial number of the remote device<br><b>Returned:</b> always                                                  |
| <code>ansible_net_stacked_models</code><br>list / elements=string     | The model names of each device in the stack<br><b>Returned:</b> when multiple devices are configured in a stack    |
| <code>ansible_net_stacked_serialnums</code><br>list / elements=string | The serial numbers of each device in the stack<br><b>Returned:</b> when multiple devices are configured in a stack |



# VS Code - Python environment for Ansible

- Check which Python environment is used by Ansible

```
devnet-adm@ubuntu-devnet:~$ source ~/automation-venv/bin/activate
(automation-venv) devnet-adm@ubuntu-devnet:~$ ansible --version | grep "python version"
 python version = 3.12.3 (main, Apr 10 2024, 05:33:47) [GCC 13.2.0] (/home/devnet-adm/automation-venv/bin/python3)
```

- Select python environment (VS Code, bottom line)



- Select «Enter interpreter path...»



- Enter the python environment that Ansible use, into this line and press Enter



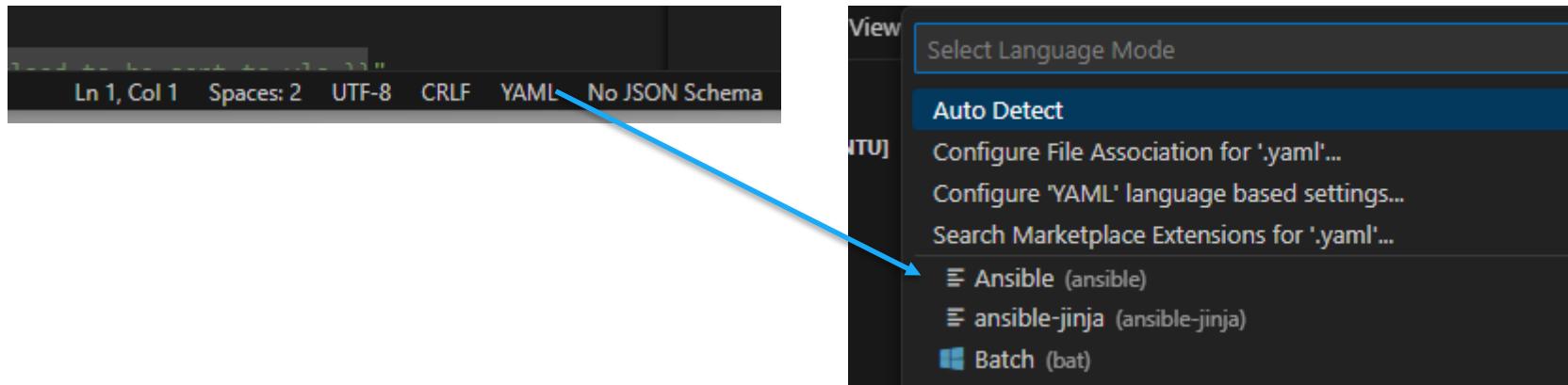
- Bottom line should now show the Ansible version (2.18.1) and Python version and venv

Ansible 2.18.1 Python 3.12.3 (automation-venv)



# VS Code - Choose file type

- If not auto detected, file type can be changed for open files



- Pro tip: If the file name contains the word «playbook», it will *automatically* be recognized as Ansible by VS Code (and not general YAML)
  - interface-playbook.yaml
  - demo-playbook.yaml
- There are also other ways to get VS Code to auto recognize playbooks, like putting them in a "playbooks" subfolder

- yaml files under `/playbooks` dir.
- files with the following double extension: `.ansible.yml` or `.ansible.yaml`.
- notable yaml names recognized by ansible like `site.yml` or `site.yaml`
- yaml files having playbook in their filename: `*playbook*.yml` or `*playbook*.yaml`



# Ansible-lint

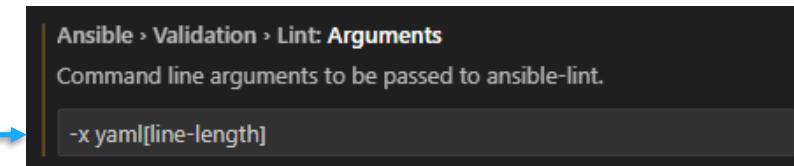
- Linting (syntax checking) is done on file save
- The «ansible-first-playbook.yml» should have an error that shows on first save, when ansible-lint runs in the background

The screenshot shows a code editor window with the file 'ansible-first-playbook.yml' open. The code is as follows:

```
examples > ansible-first-playbook.yml
1 ---
2
3 - name: My first play
4 hosts: all
5 tasks:
6 - name: Ping mv hosts
7 No new line character at the end of file ansible-lint(yaml[new-line-at-end-of-file])
8
9 Codeium: Explain Problem
10 View Problem (Alt+F8) No quick fixes available
11 msg: Hello world
```

A tooltip is displayed over the line 'No new line character at the end of file ansible-lint(yaml[new-line-at-end-of-file])'. The tooltip contains the text 'Codeium: Explain Problem' and 'View Problem (Alt+F8) No quick fixes available'.

- (optional info) You can also specify ansible-lint to exclude certain errors, like the "line too long" error.
  - Open Settings (Ctrl-,) and search for "ansible-lint"  
Enter `-x yaml[line-length]`



# Lab exercise #5: Explore Ansible

- In this lab, you will be given a simple example to get you started with Ansible
- The Ansible section of Day2 will include more exercises for you to do, including example solutions

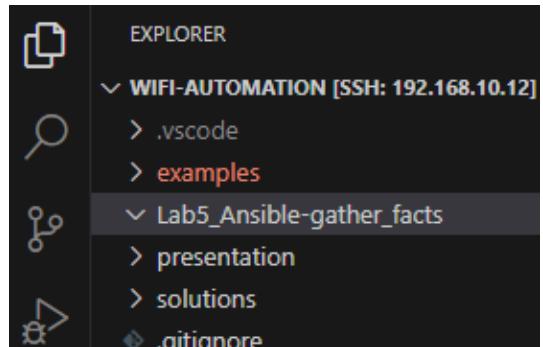


# Gather facts

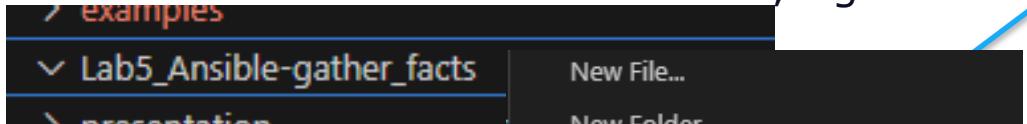
- Create a folder

```
(automation-venv) devnet-adm@ubuntu-devnet:~/wifi-automation$ mkdir Lab5_Ansible-gather_facts
```

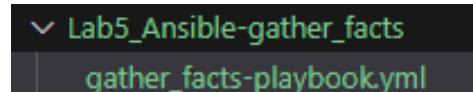
- Shows up automatically in VS Code



- Create files and folders in VS Code, right click to get menu



- Create the file "gather\_facts-playbook.yml"



- Copy-paste from this textbox to VS Code

gather\_facts-playbook.yml

```

- name: IOS Facts
 hosts: wlc
 connection: network_cli
 gather_facts: false
 tasks:
 - name: Gather all legacy facts
 cisco.ios.ios_facts:
 gather_subset: all
 register: facts1
 - name: Show facts
 ansible.builtin.debug:
 msg: "{{ facts1 }}"
```

gather\_facts-playbook.yml

```
Lab5_Ansible-gather_facts > gather_facts-playbook.yml

1 ---
2 - name: IOS Facts
3 hosts: wlc
4 connection: network_cli
5 gather_facts: false
6 tasks:
7 - name: Gather all legacy facts
8 cisco.ios.ios_facts:
9 gather_subset: all
10 register: facts1
11 - name: Show facts
12 ansible.builtin.debug:
13 msg: "{{ facts1 }}"
```



# Ansible hosts file

- Let's create a file named hosts.yml

```
└─ Lab5_Ansible-gather_facts
 └─ gather_facts-playbook.yml
 ! hosts.yml
```

- It shows up in the folder as well

```
(ansible-venv) devnet-adm@ubuntu-devnet:~/wifi-automation$ cd Lab5_Ansible-gather_facts/
(ansible-venv) devnet-adm@ubuntu-devnet:~/wifi-automation/Lab5_Ansible-gather_facts$ ls -l
total 4
-rw-rw-r-- 1 devnet-adm devnet-adm 285 Aug 24 18:06 gather_facts-playbook.yml
-rw-rw-r-- 1 devnet-adm devnet-adm 0 Aug 24 18:14 hosts.yml
(ansible-venv) devnet-adm@ubuntu-devnet:~/wifi-automation/Lab5_Ansible-gather_facts$ █
```

Change to YOUR  
WLC IP

```
hosts.yml

wlc:
 hosts:
 192.168.10.{WLC-IP}:
 vars:
 ansible_connection: network_cli
 ansible_network_os: ios
 ansible_ssh_pass: ChangeMe2025!
 ansible_password: ChangeMe2025!
 ansible_user: devnet-adm
```

```
1 wlc:
2 hosts:
3 192.168.10.{WLC-IP}:
4 vars:
5 ansible_connection: network_cli
6 ansible_network_os: ios
7 ansible_ssh_pass: ChangeMe2025!
8 ansible_password: ChangeMe2025!
9 ansible_user: devnet-adm
```



# Fix dependencies

- Try to run the playbook with the following command. Something is missing.

```
(automation-venv) devnet-adm@ubuntu-devnet:~/wifi-automation/Lab5_Ansible-gather_facts$ ansible-playbook -i hosts.yml gather_facts-playbook.yml

PLAY [IOS Facts] ****
TASK [Gather all legacy facts] ****
[WARNING]: ansible-pylibssh not installed, falling back to paramiko
fatal: [192.168.10.30]: FAILED! => {"changed": false, "msg": "paramiko is not installed: No module named 'paramiko'"}

PLAY RECAP ****
192.168.10.30 : ok=0 changed=0 unreachable=0 failed=1 skipped=0 rescued=0 ignored=0

(automation-venv) devnet-adm@ubuntu-devnet:~/wifi-automation/Lab5_Ansible-gather_facts$
```

- Let's install "ansible-pylibssh"

- First, make sure you are in your automation-venv
- If it is not showing (automation-venv) at the start of the line, enter `source ~/automation-venv/bin/activate`
- Then install the package in your venv by using "pip install {package name}"

```
(automation-venv) devnet-adm@ubuntu-devnet:~$ pip install ansible-pylibssh
Collecting ansible-pylibssh
 Downloading ansible_pylibssh-1.2.2-cp312-cp312-manylinux_2_28_x86_64.whl.metadata (36 kB)
 Downloading ansible_pylibssh-1.2.2-cp312-cp312-manylinux_2_28_x86_64.whl (2.9 MB)
 2.9/2.9 MB 7.1 MB/s eta 0:00:00
Installing collected packages: ansible-pylibssh
Successfully installed ansible-pylibssh-1.2.2
(automation-venv) devnet-adm@ubuntu-devnet:~$
```



# SSH host key checking

- Try running again. The SSH connection is failing

```
(automation-venv) devnet-adm@ubuntu-devnet:~/wifi-automation/Lab5_Ansible-gather_facts$ ansible-playbook -i hosts.yml gather_facts-playbook.yml

PLAY [IOS Facts] ****
TASK [Gather all legacy facts] ****
fatal: [192.168.10.30]: FAILED! => {"changed": false, "msg": "\nlibssh: The authenticity of host '192.168.10.30' can't be established due to 'Host is unknown: 41:86:3e:e4:fb:9f:de:06:5b:72:f3:ce:75:f1:da:ad:e7:a5:13:c0'.\n\nThe ssh-rsa key fingerprint is SHA1:QYY+5Puf3gZbcvPOdfHareelE8A."}

PLAY RECAP ****
192.168.10.30 : ok=0 changed=0 unreachable=0 failed=1 skipped=0 rescued=0 ignored=0
```

- Update the hosts.yml file

```
1 wlc:
2 hosts:
3 192.168.10.{WLC-IP}:
4 vars:
5 ansible_connection: network_cli
6 ansible_network_os: ios
7 ansible_ssh_pass: ChangeMe2025!
8 ansible_password: ChangeMe2025!
9 ansible_user: devnet-adm
10 ansible_host_key_checking: False
```

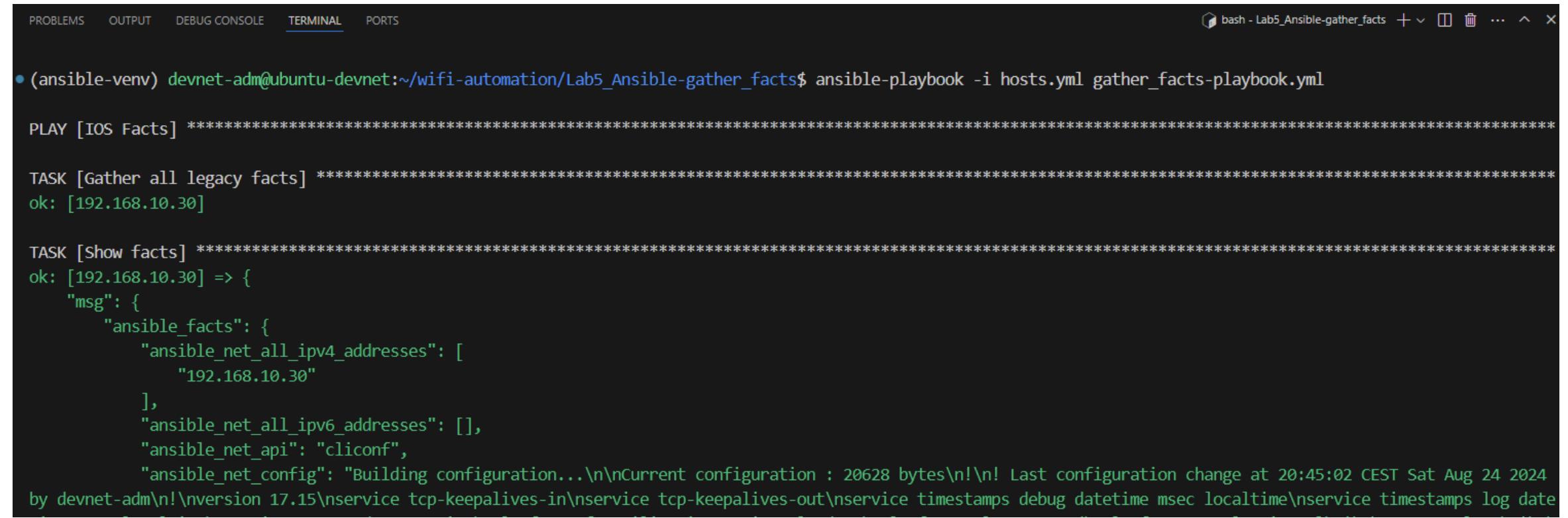
Change to YOUR  
WLC IP

```
hosts.yml
wlc:
 hosts:
 192.168.10.{WLC-IP}:
 vars:
 ansible_connection: network_cli
 ansible_network_os: ios
 ansible_ssh_pass: ChangeMe2025!
 ansible_password: ChangeMe2025!
 ansible_user: devnet-adm
 ansible_host_key_checking: False
```



# Gather facts

- Successful play



The screenshot shows a terminal window with the following details:

- Header: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (underlined), PORTS.
- Right side: bash - Lab5\_Ansible-gather\_facts, +, -, ^, X.
- Text output:
  - (ansible-venv) devnet-adm@ubuntu-devnet:~/wifi-automation/Lab5\_Ansible-gather\_facts\$ ansible-playbook -i hosts.yml gather\_facts-playbook.yml
  - PLAY [IOS Facts] \*\*\*\*
  - TASK [Gather all legacy facts] \*\*\*\*
  - ok: [192.168.10.30]
  - TASK [Show facts] \*\*\*\*
  - ok: [192.168.10.30] => {
    - "msg": {
      - "ansible\_facts": {
        - "ansible\_net\_all\_ipv4\_addresses": [
          - "192.168.10.30"],
        - "ansible\_net\_all\_ipv6\_addresses": [],
        - "ansible\_net\_api": "cliconf",
        - "ansible\_net\_config": "Building configuration...\n\nCurrent configuration : 20628 bytes\n!\n! Last configuration change at 20:45:02 CEST Sat Aug 24 2024 by devnet-adm\n!\nversion 17.15\nservice tcp-keepalives-in\nservice tcp-keepalives-out\nservice timestamps debug datetime msec localtime\nservice timestamps log date



# Use parts of return values

- The returned facts1 is a dictionary
  - You can use dot notation
  - Edit the "Show facts" task, to show only some subset of the info

```
- name: Show facts
 debug:
 msg: "{{ facts1ansible_facts.ansible_net_interfaces }}"
```

```
- name: Show facts
 debug:
 msg: "{{ facts1ansible_facts.ansible_net_interfaces }}"
```

```
(ansible-venv) devnet-adm@ubuntu-devnet:~/wifi-automation/Lab5_Ansible-gather_facts$ ansible-playbook -i hosts.yml gather_facts-playbook.yml

PLAY [IOS Facts] ****
TASK [Gather all legacy facts] ****
ok: [192.168.10.30]

TASK [Show facts] ****
ok: [192.168.10.30] => {
 "msg": {
 "GigabitEthernet1": {
 "bandwidth": 1000000,
 "description": "Uplink",
 "duplex": "Full",
 "ipv4": [],
 "lineprotocol": "up",
 "macaddress": "0800.2712.7826",
 "mediatype": "Virtual",
 "mtu": 1500,
 "operstatus": "up",
 "speed": 10000000
 }
 }
}
```



# Use return values

- We will use the returned facts, to write the run-config to a file
- Remove the "Show facts" task, and replace it with a new task

```

```

```
- name: IOS Facts
 hosts: wlc
 connection: network_cli
 gather_facts: no
 tasks:
 - name: Gather all legacy facts
 cisco.ios.ios_facts:
 gather_subset: all
 register: facts1
 - name: Write run-config to file
 ansible.builtin.copy:
 content: "{{ facts1.ansible_facts.ansible_net_config }}"
 dest: "{{ './' + facts1.ansible_facts.ansible_net_hostname + '-run-conf.txt' }}"
```

```
- name: Write run-config to file
 ansible.builtin.copy:
 content: "{{ facts1.ansible_facts.ansible_net_config }}"
 dest: "{{ './' + facts1.ansible_facts.ansible_net_hostname + '-run-conf.txt' }}"
```

- Using the "copy" module
- Content is using parts of the return values, this time we use "ansible\_net\_config"
- Dest is where to write the content. Here are some simple examples on Jinja syntax
- (optional info)
- Jinja2 is integrated in Ansible and widely used
- "Everything inside double curly brackets" = Jinja2



# Linting errors, and fixing them

- When saving this file you should get two linting errors

```

1 connection: network_cli
2
3 gather_facts: no
4
5 tasks:
6
7 - name: Gather all legacy facts
8 cisco.ios.ios_facts:
9 gather_subset: all
10 register: facts1
11
12 - name: Write run-config to file
13 ansible.builtin.copy:
14 content: "{{ facts1.ansible_facts.ansible_net_config }}"
15 dest: "{{ './' + facts1.ansible_facts.ansible_net_hostname + '-run-conf.txt' }}"
16 mode: "0600"

```

File permissions unset or incorrect. ansible-lint(risky-file-permissions)

- When writing files (like using the "copy" module), you should always explicitly specify the file permissions
- We add the line: mode: "0600"

```

1 ---
2 - name: IOS Facts
3 hosts: wlc
4 connection: network_cli
5 gather_facts: false
6 tasks:
7 - name: Gather all legacy facts
8 cisco.ios.ios_facts:
9 gather_subset: all
10 register: facts1
11
12 - name: Write run-config to file
13 ansible.builtin.copy:
14 content: "{{ facts1.ansible_facts.ansible_net_config }}"
15 dest: "{{ './' + facts1.ansible_facts.ansible_net_hostname + '-run-conf.txt' }}"
16 mode: "0600"

```

Truthy value should be one of \[false, true] ansible-lint(yaml[truthy])  
 Codeium: Explain Problem  
 A boolean that controls if the play will automatically run the 'setup' task to gather facts for the hosts.

- We fix this with using "false" instead of "no"
- Both will work, but correct YAML syntax is "false"



# Run the playbook

```
• (ansible-venv) devnet-adm@ubuntu-devnet:~/wifi-automation/Lab5_Ansible-gather_facts$ ansible-playbook -i hosts.yml gather_facts-playbook.yml

PLAY [IOS Facts] *****

TASK [Gather all legacy facts] *****
ok: [192.168.10.30]

TASK [Write run-config to file] *****
changed: [192.168.10.30]

PLAY RECAP *****
192.168.10.30 : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

• (ansible-venv) devnet-adm@ubuntu-devnet:~/wifi-automation/Lab5_Ansible-gather_facts$
```

- List the files, and you can see the file that was created

```
• (ansible-venv) devnet-adm@ubuntu-devnet:~/wifi-automation/Lab5_Ansible-gather_facts$ ls -l
total 32
-rw-rw-r-- 1 devnet-adm devnet-adm 445 Aug 24 18:54 gather_facts-playbook.yml
-rw-rw-r-- 1 devnet-adm devnet-adm 245 Aug 24 18:41 hosts.yml
-rw----- 1 devnet-adm devnet-adm 20689 Aug 24 18:54 wlc30-run-conf.txt
```

- You can view the file contents in the Linux terminal by running "cat", "more", "nano" or similar
- You will also see the file in VS Code explorer and can open it from there

The diagram illustrates the workflow for viewing Ansible files. A blue arrow points from the terminal command 'ls -l' output to the VS Code file explorer, indicating that the file 'wlc30-run-conf.txt' is now visible in the explorer. Another blue arrow points from the VS Code file viewer to the terminal window, showing the file's contents.

wlc30-run-conf.txt U

Lab5\_Ansible-gather\_facts > wlc30-run-conf.txt

```
1 Building configuration...
2 !
3 Current configuration : 20628 byte(s)
4 !
5 ! Last configuration change at 2018-08-24 18:54:00
6 !
7 version 17.15
8 service tcp-keepalives-in
9 service tcp-keepalives-out
10 service timestamps debug datetime msec
11 service timestamps log datetime msec
12 service password-encryption
13 platform qfp utilization monitor
14 platform sslvpn use-pd
15 platform console virtual
16 !
17 hostname wlc30
```



# Lab exercise #6: Explore YANG models

- First there will be a couple of slides to explain what YANG models are
- Then you will explore some of the YANG models of the 9800 WLAN Controller by using the <https://yangcatalog.org> website
- Lab 7 will explore YANG models using another tool (YANG Suite)
- Some of the later labs will have optional parts where you can look up RESTCONF paths that you need using one of these tools



# What is a YANG-model?

```
module: Cisco-IOS-XE-wireless-client-oper
++-ro client-oper-data
 +-ro common-oper-data* [client-mac]
 | +-ro client-mac yang:mac-address
 | string
 | +-ro ap-name? uint8
 | +-ro ms-ap-slot-id? wireless-client-types:ms-phy-radio-ty
 | uint32
 | +-ro ms-radio-type? wireless-client-types:ms-client-type
 | wireless-client-types:client-co-state
 | +-ro wlan-id? boolean
 | +-ro client-type? wireless-client-types:client-co-state
 | +-ro co-state? wireless-client-types:client-co-state
 | +-ro aaa-override-passphrase? boolean
 | +-ro is-tvi-enabled? boolean
 | +-ro wlan-policy
 | +-ro current-switching-mode? wireless-client-oper:client-switching-m
 | +-ro wlan-switching-mode? wireless-client-oper:client-switching-m
 | +-ro central-authentication? wireless-client-oper:client-authenticat
 | +-ro central-dhcp? boolean
 | +-ro central-assoc-enable? boolean
 | +-ro vlan-central-switching? boolean
```

- Structured representation of cfg og oper data
- Used primarily with NETCONF & RESTCONF
- Do you recognize what the tree is showing?

Reference/download: <https://github.com/YangModels/yang/tree/main/vendor/cisco/xe/1751>



# yangcatalog

- Open the following website:  
<https://www.yangcatalog.org/>

- Select "Module Detail Viewer"

- Here you can start typing parts of available YANG modules.  
Start by typing "IOS-XE-wireless".  
It is not case sensitive by the way

- Select the AP oper module  
"Cisco-IOS-XE-wireless-access-point-oper"  
and click "Get details"

- Then click "Tree View"

The screenshot shows the Yang Catalog website at <https://www.yangcatalog.org>. The search bar at the top contains the text "IOS-~~xe~~-wire". Below the search bar, a list of YANG module names is displayed, with the first item, "Cisco-IOS-XE-wireless-access-point-oper", highlighted in blue. A blue arrow points from the text "Start by typing \"IOS-XE-wireless\"." to this module name. To the right of the module name is a "Get details" button. Another blue arrow points from the text "Select the AP oper module" to this button. At the bottom of the page, there are links for "Tree View", "Impact Analysis", and "Implementations".

Module Name

IOS-~~xe~~-wire

Cisco-IOS-XE-wireless-access-point-oper

Cisco-IOS-XE-wireless-ap-types

Cisco-IOS-XE-wireless-client-oper

Cisco-IOS-XE-wireless-enum-types

Get details

Module Name

Cisco-IOS-XE-wireless-access-point-oper

Get details

Tree View Impact Analysis Implementations

| Property Name | Property Value                          |
|---------------|-----------------------------------------|
| name ⓘ        | Cisco-IOS-XE-wireless-access-point-oper |



# YANG Catalog - Navigating the Tree View

- Expand/collapse all nodes
- Expand/collapse this branch
- This is a container/list node
  - Contains other nodes
- Hover the mouse over an element name to get the description popup
- This is a leaf node
  - Contains a single value
- !! There are lots of info to the right of this screenshot. Sometimes the horizontal scroll bar is trying to hide, scroll down to the bottom to find it !!

| Element                                 | Schema    | Type             |
|-----------------------------------------|-----------|------------------|
| Cisco-IOS-XE-wireless-access-point-oper | module    | module           |
| access-point-oper-data                  | container | container        |
| ap-radio-audit-info                     | list      | list             |
| wtp-mac                                 | leaf      | yang:mac-address |
| radio-slot-id                           | leaf      | uint8            |
| channel-sync                            | leaf      | empty            |
| bandwidth-sync                          | leaf      | empty            |
| tx-power-sync                           | leaf      | empty            |



# Using values from YANG Catalog

- When navigating the tree view, here is where you find the values you will typically use, with examples
  - "Path": Used when configuring 9800 to send values as streaming telemetry. We will use this in lab 10 & 11 to configure WLC to send specific data to be visualized in Grafana
  - "Sensor Path": Used when you need a RESTCONF path. We will use this in the labs 8 (Postman), 9 (Python) and in many of the Day 2 labs (both Python and Ansible labs)

## YANG Catalog

[Home](#) | [About](#) | [Use cases](#) | [Contribute](#) | [Query](#) | [YANG M](#)

[YANG Search](#) | [YANG Module Detail Viewer](#) | [YANG Validator](#) | [YANG Impact A](#)

Module: Cisco-IOS-XE-wireless-access-point-oper@2022-07-01  
 Namespace: <http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-access-point-oper>  
 Prefix: wireless-access-point-oper

[Module details](#) [Impact Analysis](#)

| Element                                                 | Schema    | Type      | Flags     | Opts | Status  | Path                                                                                                |
|---------------------------------------------------------|-----------|-----------|-----------|------|---------|-----------------------------------------------------------------------------------------------------|
| <a href="#">Cisco-IOS-XE-wireless-access-point-oper</a> | module    | module    |           |      |         |                                                                                                     |
| <a href="#">access-point-oper-data</a> ⓘ                | container | container | no config |      | current | /wireless-access-point-oper:access-point-oper-data ⓘ                                                |
| <a href="#">ap-radio-audit-info</a> ⓘ                   | list      | list      | no config |      | current | /wireless-access-point-oper:access-point-oper-data/wireless-access-point-oper:ap-radio-audit-info ⓘ |
| <a href="#">radio-oper-data</a> ⓘ                       | list      | list      | no config |      | current | /wireless-access-point-oper:access-point-oper-data/wireless-access-point-oper:radio-oper-data ⓘ     |

Use this for WLC telemetry config

Use this for RESTCONF path

```
no telemetry ietf subscription 4
telemetry ietf subscription 4
encoding encode-kvvpb
filter xpath /wireless-access-point-oper:access-point-oper-data/wireless-access-point-oper:radio-oper-data
stream yang-push
update-policy periodic 500
receiver ip address 192.168.10.7 57000 protocol grpc-tcp
do show telemetry ietf subscription 4
```

GET <https://{{host}}/restconf/data/Cisco-IOS-XE-wireless-access-point-oper:access-point-oper-data/radio-oper-data>



# yangcatalog

- You can also get the tree view in a text form like this →
- Go to YANG Module Detail Viewer again,  
Search and find your module, and select "Get details"
- A way down in the details table you will find "yang-tree",  
click the link there

yang-tree

<https://yangcatalog.org/api/services/tree/Cisco-IOS-XE-wireless-wlan-cfg@2022-07-01.yang>

- The tree will open in a separate page

```
module: Cisco-IOS-XE-wireless-access-point-oper
++-ro access-point-oper-data
| +-ro ap-radio-audit-info* [wtp-mac radio-slot-id]
| | +-ro wtp-mac yang:mac-address
| | +-ro radio-slot-id uint8
| | +-ro channel-sync? empty
| | +-ro bandwidth-sync? empty
| | +-ro tx-power-sync? empty
| | +-ro admin-state-sync? empty
| | +-ro oper-state-sync? empty
| | +-ro radio-role-sync? empty
| | +-ro radio-role-oper-sync? empty
+-ro ap-wlan-audit-info* [wtp-mac slot-id wlan-id]
| +-ro wtp-mac yang:mac-address
| | +-ro slot-id uint8
| | +-ro wlan-id uint8
| | +-ro wlan-id-sync? empty
| | +-ro state-sync? empty
| | +-ro ssid-sync? empty
| | +-ro auth-type-sync? empty
| | +-ro other-flags-sync? empty
+-ro ap-audit-summary-info* [wtp-mac]
| +-ro wtp-mac yang:mac-address
| | +-ro radio-audit-sync? empty
| | +-ro wlan-audit-sync? empty
| | +-ro ipv4-acl-sync? empty
| | +-ro ipv6-acl-sync? empty
| | +-ro last-report-time? yang:date-and-time
+-ro ap-mac-ssid-info* [ap-mac wlan-id]
| +-ro ssid-clients-list-data* [client-mac-address]
| | +-ro client-mac-address yang:mac-address
| | +-ro ap-mac yang:mac-address
| | +-ro wlan-id uint32
+-ro ssid-counters* [wtp-mac slot-id wlan-id]
| +-ro bssid-mac? yang:mac-address
| | +-ro vap-id? uint16
| | +-ro tx-mgmt? uint64
| | +-ro rx-mgmt? uint64
| | +-ro tx-bytes-data? uint64
| | +-ro tx-data-dist
| | | +-ro bytes-0-64? uint64
| | | +-ro bytes-65-128? uint64
```



# Lab exercise #7: Explore YANG Suite

- YANG Suite is an installable tool that can be used to pull supported YANG models directly from the actual devices
- You can also use it to test communication to your device using RESTCONF and more

## Cisco YANG Suite



YANG API Testing and Validation Environment

Construct and test YANG based APIs over  
NETCONF, RESTCONF, gRPC and gNMI

IOS XE / IOS XR / NX OS platforms

Now Available !

[developer.cisco.com/yangsuite](http://developer.cisco.com/yangsuite)

[github.com/CiscoDevNet/yangsuite](https://github.com/CiscoDevNet/yangsuite)

## Cisco-IOS-XE-wireless: Modules

### Config

|                |          |
|----------------|----------|
| ap             | mesh     |
| apf            | mobility |
| cts-sxp        | mstream  |
| dot11          | radio    |
| dot15          | rf       |
| fabric         | Rfid     |
| file-transfer  | rlan     |
| flex           | rogue    |
| fqdn           | rrm      |
| general        | rule     |
| hotspot        | security |
| image-download | site     |
| location       | tunnel   |
| me-general     | wlan     |

### Oper

|               |           |
|---------------|-----------|
| access-point  | mesh      |
| ap            | mobility  |
| awips         | nmsp      |
| ble-ltx       | rfid      |
| ble-mgmt      | rogue     |
| client        | rrm       |
| cts-sxp       | rule-mdns |
| events        |           |
| general       |           |
| hyperlocation |           |
| lisp-agent    |           |
| location      |           |
| mcast         |           |
| mdns          |           |



# Lab exercise #7: Explore YANG Suite

- !!! NOTE !!! You will explore YANG Suite using one of the servers where it is pre-installed. The installation has some bindings to the IP of the server, so it should be installed when you have the permanent static IP of your server
- If you want to install it on your own VM, the instructions are included in the optional Lab Exercise #12, beware that it will take a little while so you might want to do it later instead of during the deep dive
- In the pre-installed servers, YANG models will already be pulled from one of the pre-installed WLCs
- To get an even distribution, you can spread so that if you have the following WLC, use the following Ubuntu Servers to explore YANG Suite:

| Shared WLC            | Shared Ubuntu Server     | YANG Suite URL                                                      |
|-----------------------|--------------------------|---------------------------------------------------------------------|
| wlc51 (192.168.10.51) | ubuntu11 (192.168.10.11) | <a href="https://192.168.10.11:8443">https://192.168.10.11:8443</a> |
| wlc53 (192.168.10.53) | ubuntu12 (192.168.10.12) | <a href="https://192.168.10.12:8443">https://192.168.10.12:8443</a> |
| wlc55 (192.168.10.55) | ubuntu13 (192.168.10.13) | <a href="https://192.168.10.13:8443">https://192.168.10.13:8443</a> |
| wlc57 (192.168.10.57) | ubuntu14 (192.168.10.14) | <a href="https://192.168.10.14:8443">https://192.168.10.14:8443</a> |
| wlc59 (192.168.10.59) | ubuntu15 (192.168.10.15) | <a href="https://192.168.10.15:8443">https://192.168.10.15:8443</a> |
| wlc61 (192.168.10.61) | ubuntu16 (192.168.10.16) | <a href="https://192.168.10.16:8443">https://192.168.10.16:8443</a> |



# YANG Suite

- «oper» -> Operational data
- «cfg» -> Configuration and config data
- Log in to a YANG Suite (URL on previous slide)
- Go to Explore -> YANG -> Select a YANG. Select a YANG model and click the "Load module(s)" button

The screenshot shows the Cisco YANG Suite interface. On the left, a vertical navigation bar has 'Explore' highlighted with a blue box. The main area is titled 'Explore YANG Models' and shows a list of YANG modules. One module, 'ATM-FORUM-TC-MIB', is highlighted with a blue box. A 'Load module(s)' button is visible at the top right of the module list.

Cisco YANG Suite

Admin

Setup

Analytics

Explore

YANG

Protocols

Help

YANG Suite / Exploring YANG / YANG set "9800-vm-default-yangset"

Explore YANG Models

Select a YANG set: 9800-vm-default-yangset

Select YANG module(s):

- ATM-FORUM-TC-MIB
- ATM-MIB
- ATM-TC-MIB
- BGP4-MIB
- BRIDGE-MIB
- CISCO-AAA-SERVER-MIB
- CISCO-AAA-SESSION-MIB
- CISCO-AAL5-MIB
- CISCO-ATM-EXT-MIB
- CISCO-ATM-PVCTRAP-EXTN-MIB

Load module(s)



# Lab exercise #7: Explore YANG Suite

- Explore YANG Models

YANG Suite / Exploring YANG / YANG set "9800-cl-17-12-default-yangset" / Modules

Explore YANG Models

Select a YANG set: 9800-cl-17-12-default-yangset Select YANG module(s): Cisco-IOS-XE-process-cpu-oper

Icon legend: Search XPaths, Search nodes, Expand all nodes, Display schema node

Node Properties:

|                |                                                                                                                             |
|----------------|-----------------------------------------------------------------------------------------------------------------------------|
| Name           | five-seconds                                                                                                                |
| Nodetype       | leaf                                                                                                                        |
| Datatype       | uint8                                                                                                                       |
| Description    | Busy percentage in last 5-seconds                                                                                           |
| Module         | Cisco-IOS-XE-process-cpu-oper                                                                                               |
| Revision       | 2022-11-01                                                                                                                  |
| Xpath          | /cpu-usage/cpu-utilization/five-seconds                                                                                     |
| Prefix         | process-cpu-ios-xe-oper                                                                                                     |
| Namespace      | <a href="http://cisco.com/ns/yang/Cisco-IOS-XE-process-cpu-oper">http://cisco.com/ns/yang/Cisco-IOS-XE-process-cpu-oper</a> |
| Schema Node Id | /cpu-usage/cpu-utilization/five-seconds                                                                                     |
| Units          | percent                                                                                                                     |
| Min            | 0                                                                                                                           |
| Max            | 255                                                                                                                         |
| Access         | read-only                                                                                                                   |
| Operations     | • "get"                                                                                                                     |



# Lab exercise #7: Explore YANG Suite

- Over the next pages is examples on a couple more YANG models you can check out
  - Get AP summary
  - Get Clients
- Just get somewhat familiar, you will use these in the next lab exercises (Postman, Python, Ansible, Grafana)
- This will show you the model, not pull the actual data
- To pull data we will use these models in Postman and Grafana later
- (optional) You can also do RESTCONF and NETCONF calls to the device from YANG Suite
  - Protocols -> RESTCONF -> Find your module -> Generate API(s) -> Authorize -> Try it out -> Execute

View Capwap-Data

GET /data/Cisco-IOS-XE-wireless-access-point-oper:access-point-oper-data/capwap-data GET operation on "capwap-data"

This endpoint retrieves the device's "capwap-data" resource at XPath: access-point-oper-data/capwap-data.

Parameters

Try it out

cess-point-oper:access-point-oper-data/capwap-data

Server response

| Code | Details       |
|------|---------------|
| 200  | Response body |

```
{
 "Cisco-IOS-XE-wireless-access-point-oper:capwap-data": [
 {
 "wtp-mac": "4c:a6:4d:65:f9:40",
 "ip-addr": "192.168.10.182",
 "port": 1
 }
]
}
```



# Get AP summary

Select a YANG set **9800-vm-default-yangset** Select YANG module(s) **Cisco-IOS-XE-wireless-access-point-oper** Load module(s)

Icon legend Search XPaths Search nodes Expand all nodes Display schema nodes only Display all nodes

**Cisco-IOS-XE-wireless-access-point-oper**

- access-point-oper-data
  - ap-radio-audit-info
  - ap-wlan-audit-info
  - ap-audit-summary-info
  - ap-mac-ssid-info
  - ssid-counters
  - radius-counters
  - ap-radio-neighbor
  - radio-oper-data
  - radio-reset-stats
  - radio-failure-stats
  - qos-client-data
  - capwap-data
    - wtp-mac
    - ip-addr
    - name
    - device-detail
    - ap-lag-enabled
    - ap-location
    - ap-services
    - tag-info
    - tunnel
    - external-module-data
    - ipv6-joined
    - ap-state
    - ap-mode-data
    - ap-time-info
    - country-code

**Node Properties**

|                |                                                                                   |
|----------------|-----------------------------------------------------------------------------------|
| Name           | capwap-data                                                                       |
| Nodetype       | list                                                                              |
| Description    | Captures the information about the 802.11 LWAPP AP that has joined the controller |
| Module         | Cisco-IOS-XE-wireless-access-point-oper                                           |
| Revision       | 2023-07-10                                                                        |
| Xpath          | /access-point-oper-data/capwap-data                                               |
| Prefix         | wireless-access-point-oper                                                        |
| Namespace      | http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-access-point-oper                  |
| Schema Node Id | /access-point-oper-data/capwap-data                                               |
| Keys           | • "wtp-mac"                                                                       |
| Access         | read-only                                                                         |
| Operations     | • "get"                                                                           |

**Reference URL:**  
<https://tools.ietf.org/html/rfc6020#section-7.8>

7.8. The list Statement

The "list" statement is used to define an interior data node in the schema tree. A list node may exist in multiple instances in the data tree. Each such instance is known as a list entry. The "list" statement takes one argument, which is an identifier, followed by a block of substatements that holds detailed list information.

A list entry is uniquely identified by the values of the list's keys, if defined.



# Get Client summary

Select a YANG set **9800-vm-default-yangset** Select YANG module(s) **Cisco-IOS-XE-wireless-client-oper** **Load module(s)**

**Icon legend** **Search XPaths** **Search nodes** **Expand all nodes** **Display schema nodes only** **Display all nodes**

**Cisco-IOS-XE-wireless-client-oper**

- client-oper-data
  - common-oper-data
    - client-mac
    - ap-name
    - ms-ap-slot-id
    - ms-radio-type
    - wlan-id
    - client-type
    - co-state
    - aaa-override-passphrase
    - is-tvi-enabled
    - wlan-policy
    - username
    - guest-lan-client-info
    - method-id
    - I3-vlan-override-received
    - ipsk-tag
      - upn-id
      - is-central-nat
      - is-locally-administered-mac
      - idle-timeout
      - idle-timestamp
      - client-duid
      - vrf-name

**Node Properties**

|                |                                                            |
|----------------|------------------------------------------------------------|
| Name           | common-oper-data                                           |
| Nodetype       | list                                                       |
| Description    | List containing common operational data of the client      |
| Module         | Cisco-IOS-XE-wireless-client-oper                          |
| Revision       | 2023-07-01                                                 |
| Xpath          | /client-oper-data/common-oper-data                         |
| Prefix         | wireless-client-oper                                       |
| Namespace      | http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-client-oper |
| Schema Node Id | /client-oper-data/common-oper-data                         |
| Keys           | • "client-mac"                                             |
| Access         | read-only                                                  |
| Operations     | • "get"                                                    |

**Reference URL:**  
<https://tools.ietf.org/html/rfc6020#section-7.8>

7.8. The list Statement

The "list" statement is used to define an interior data node in the schema tree. A list node may exist in multiple instances in the data tree. Each such instance is known as a list entry. The "list" statement takes one argument, which is an identifier, followed by a block of substatements that holds detailed list information.

A list entry is uniquely identified by the values of the list's keys, if defined.



# Lab exercise #8: Explore Postman

- In this lab exercise we will
  - Create a Workspace
  - Create an Environment
  - Create a Collection
  - Test some queries found from yangcatalog.org or YANG Suite (any of your choice)
  - Output to CURL or Python



# Overview

Here is an example view of Postman and how the YANG modules from previous exercises can look here. We will test and explain this in more detail in this exercise

1. RESTCONF path
2. YANG module
3. Xpath

Postman is a "go-to" tool for RESTCONF APIs. Some examples for automation is testing and validating RESTCONF calls to IOS-XE devices before implementing in Python, Ansible, etc.

The screenshot shows the Postman interface with the following details:

- HTTP Request:** GET /get 5sec CPU
- URL:** `https://{{host}}/restconf/data/Cisco-IOS-XE-process-cpu-oper:cpu-usage/cpu-utilization/five-seconds`
- Headers:** (9) - This tab is selected.
- Query Params:** A table with one row and two columns: Key (empty) and Value (empty).
- Body:** This tab is selected.
- Content:** JSON response:

```
1 {
2 "Cisco-IOS-XE-process-cpu-oper:five-seconds": 2
3 }
```

Three numbered callouts point to specific parts of the interface:

- 1 points to the RESTCONF path in the URL bar.
- 2 points to the YANG module name in the URL bar.
- 3 points to the Xpath query in the URL bar.



# Launching Postman

Start by launching Postman. You should already be logged in from the pre-lab task. If not, log in now. It might look something like this

The screenshot shows the Postman application window. The left sidebar displays a workspace named "Andreas DevNet" containing a collection titled "Cisco 9800". The collection includes several API requests such as "GET Get config", "GET Get AP summary", and "GET Get AP mac-name mapping". The main content area is titled "Overview" and contains sections for "Introduction", "Getting started with this workspace", "Reference collections", "Blueprint and workflow collections", "Environments", and "Best practices". The "About" section on the right provides a brief description of the workspace's purpose. The bottom navigation bar includes links for "Postbot", "Runner", "Start Proxy", "Cookies", "Vault", "Trash", and help documentation.

Andreas DevNet

Collections

Environments

History

Overview

Andreas DevNet

Overview Settings

**Introduction**

Welcome to the API demos workspace. This workspace contains Postman collections that you can quickly demo and share our APIs with consumers.

**Getting started with this workspace**

In this workspace, you'll find two types of Postman collections that help our customers understand and consume our APIs.

**Reference collections**

These collections contain all the requests and documentation associated with that API. Please share this with customers to provide a comprehensive overview of our APIs and how to use them.

**Blueprint and workflow collections**

These collections illustrate real-life use cases where our APIs can be used. Please use these to demo our API capabilities to consumers.

**Environments**

Environments are set up to manage variables like base URLs, API keys, and tokens whose values you can change depending on the context of your work (e.g., development, staging, production).

**Best practices**

About

This workspace contains resources that you can demo and share with customers and partners.

Contributors

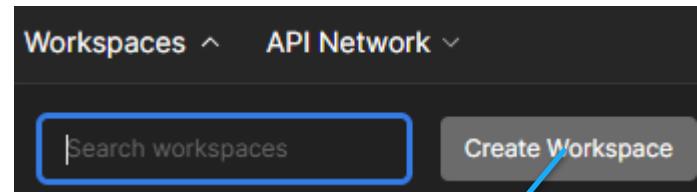
You

Online Find and replace Console

Postbot Runner Start Proxy Cookies Vault Trash ?



# Create Workspace



### Create your workspace

Get the most out of your workspace with a template.

Blank workspace

**Explore our templates**

- API demos
- API development
- API testing
- API security
- Incident response
- Cloud infrastructure management
- Partner Collaboration

Step 1 of 2

Cancel **Next**

A blue arrow points from the 'Blank workspace' checkbox in this dialog up to the 'Create your workspace' dialog.

### Create your workspace

Name

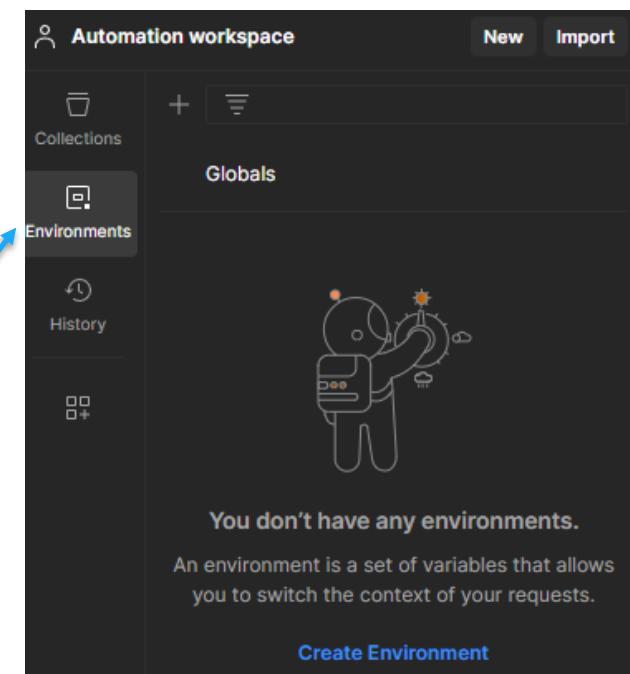
Who can access your workspace?

- Only me  
Personal
- Only invited team members  
Private [UPGRADE PLAN](#)
- Everyone from team speeding-satellite-87...  
Team (2 members)
- Only invited partners and team members  
Partner [UPGRADE PLAN](#)
- Anyone on the internet  
Public

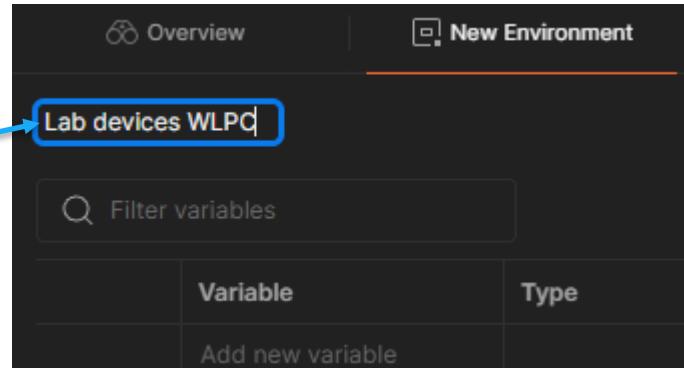
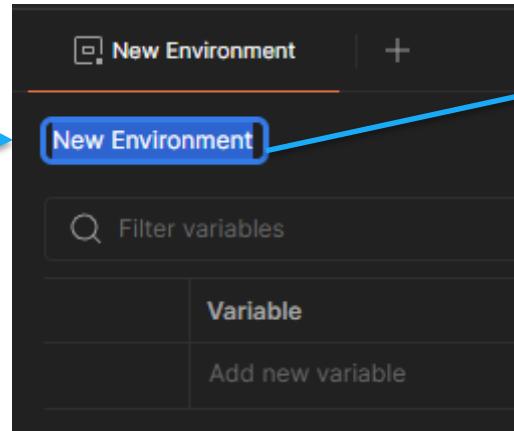
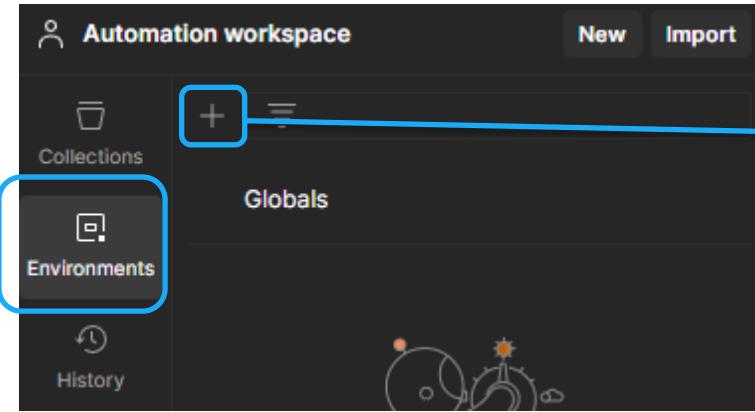
Step 2 of 2

Back **Create**

A blue arrow points from the 'Only me' radio button in this dialog up to the 'Create your workspace' dialog.

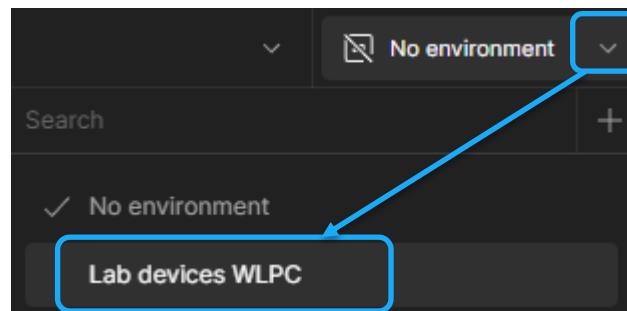


# Create Environment



- Activate the new environment

... and create some variables



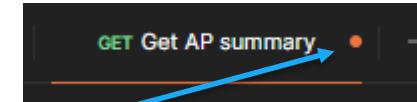
The screenshot shows the 'Lab devices WLPC' environment variables table. It has columns for 'Variable', 'Type', 'Initial value', and 'Current value'. Three annotations are present: 1) 'Values in this column are synced to your Postman account' points to the 'Initial value' column. 2) 'Values in this column are stored locally in Postman on your PC' points to the 'Current value' column. 3) A red box highlights the 'Save' button at the top right. The table data is as follows:

| Variable | Type    | Initial value | Current value |
|----------|---------|---------------|---------------|
| host     | default |               | 192.168.10.51 |
| username | default |               | devnet-adm    |
| password | secret  |               | ChangeMe2025! |

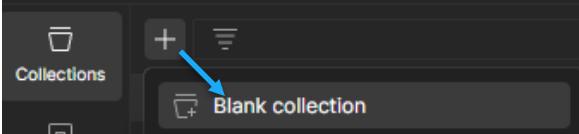
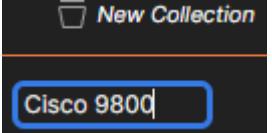


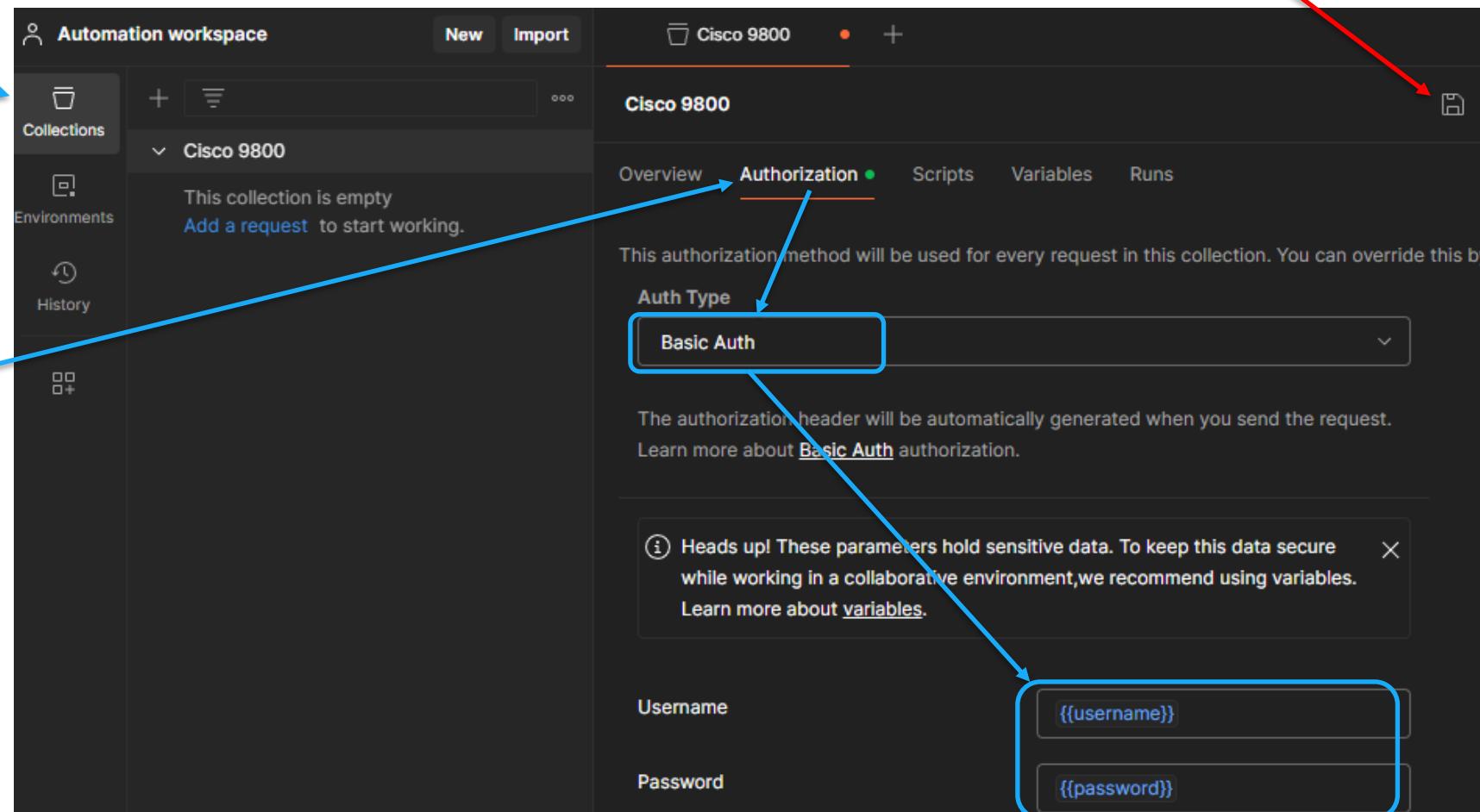
# A word about saving

- This little orange dot means the file/tab/whatever is not saved
- Use Ctrl+S to save the current tab, or click this icon
- An example: When you run a RESTCONF call with variables from an environment, like `{{host}}` or `{{password}}` it will use the value from your SAVED file. So if you do changes in the environment, remember to save the tab



# Create Collection

- Select "Collections"
- Click the +, create a Blank collection
- Rename the collection
- Click "Authorization" select "Basic Auth"  
Enter Username and Password like this
- Creating the variables like this will make Postman get them from the current active environment (that we just created)



Automation workspace

New Import

Cisco 9800

Collections

Environments

History

Cisco 9800

This collection is empty  
Add a request to start working.

Overview Authorization Scripts Variables Runs

This authorization method will be used for every request in this collection. You can override this by

Auth Type

Basic Auth

The authorization header will be automatically generated when you send the request.  
Learn more about [Basic Auth](#) authorization.

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables.  
Learn more about [variables](#).

Username {{username}}

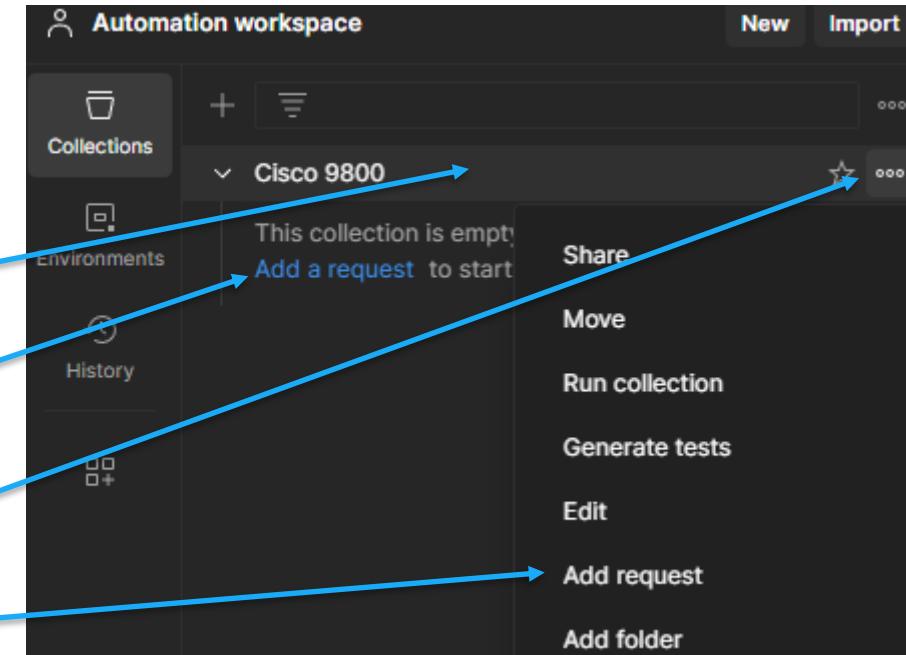
Password {{password}}

!! Remember to save !!

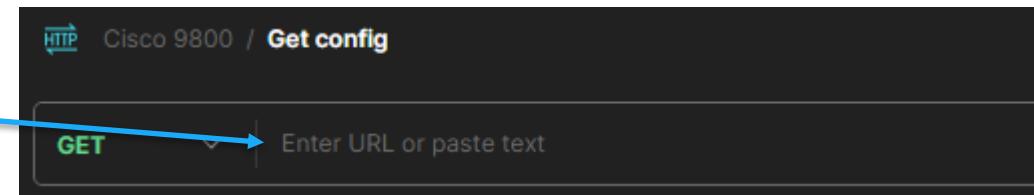


# Creating your first Request

- In the "Collections" tab and on the "Cisco C9800" collection you created in last step, Add a new request by either
  - Right clicking "Cisco C9800"
  - Clicking "Add a request" if you haven't created any before
  - Clicking the three dots on the "Cisco C9800" line
- Then select "Add request"
- Name the new request "Get config"



- Write your Request on the top line, default starting with "GET" and empty line
- The next slides will give example of creating requests using yangcatalog and YANG Suite, with some alternative slide layouts/explanations



# Get config

- Create the Request based on the information explored from <https://yangcatalog.org>

– Expand the tree until you find your module

The screenshot shows the "Module Name" search bar containing "Cisco-IOS-XE-native" with a green checkmark. A blue arrow points from the "Element" column of the table below to the "Cisco-IOS-XE-native" entry. Another blue arrow points from the "Sensor Path" column to the path "/Cisco-IOS-XE-native:native".

| Element             | Schema    | Type      | Flags  | Opts | Status  | Path        | Sensor Path                 |
|---------------------|-----------|-----------|--------|------|---------|-------------|-----------------------------|
| Cisco-IOS-XE-native | module    | module    |        |      |         | /ios:native | /Cisco-IOS-XE-native:native |
| native              | container | container | config |      | current |             |                             |

– Base restconf path:

`https://{{host}}/restconf/data`

– Add the "Sensor Path" to the base path

– Write the path in Postman

– Go to "Headers" and enter the following

– Press "Send" and you should get a response similar to this

The Postman interface shows a GET request to the URL `https://{{host}}/restconf/data/Cisco-IOS-XE-native:native`. The Headers tab is selected, showing a key-value pair `Accept: application/yang-data+json`.

The response body is displayed in JSON format, showing the configuration data for the Cisco-IOS-XE-native module.

```

200 OK · 1.84 s · 13.63 KB · Save Response
Body Cookies Headers (15) Test Results
{ } JSON Preview Visualize
1 {
2 "Cisco-IOS-XE-native": {
3 "version": "17.15",
4 "boot-start-marker": [
5 null
6],
7 "running": {
8 "configuration": {
9 "Cisco-IOS-XE-native": {
10 "version": "17.15"
11 }
12 }
13 }
14 }
15 }

```



# Get config (YANG Suite alternative)

- Here is an example of how to craft the same Request based on info you find in YANG Suite
  - Base Restconf path: `https://{{host}}/restconf/data/`
  - YANG-module: Cisco-IOS-XE-native
  - Xpath: native
  - Combined request path for copy-paste: `https://{{host}}/restconf/data/Cisco-IOS-XE-native:native`
  - Write the path in Postman
  - Go to "Headers" and enter the following
  - Press "Send" and you should get a response similar to this

|           |                                                           |
|-----------|-----------------------------------------------------------|
| Module    | Cisco-IOS-XE-native                                       |
| Revision  | 2023-07-01                                                |
| Xpath     | /native                                                   |
| Prefix    | ios                                                       |
| Namespace | <code>http://cisco.com/ns/yang/Cisco-IOS-XE-native</code> |

Cisco-IOS-XE-native

2023-07-01

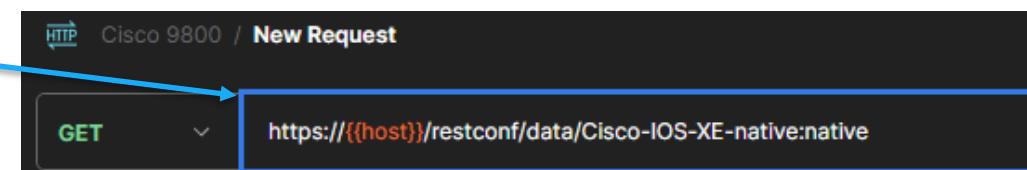
/native

ios

`http://cisco.com/ns/yang/Cisco-IOS-XE-native`

Use colon (:) instead of forward slash (/) in the Xpath

You may get some line breaks at the end. Remove them before sending the request



A screenshot of the Postman Headers tab. It shows a table with one row. The key is "Accept" and the value is "application/yang-data+json". A checkbox is checked next to the key column.

| Key    | Value                      |
|--------|----------------------------|
| Accept | application/yang-data+json |

A screenshot of the Postman Body tab. It displays a JSON response with the following content:

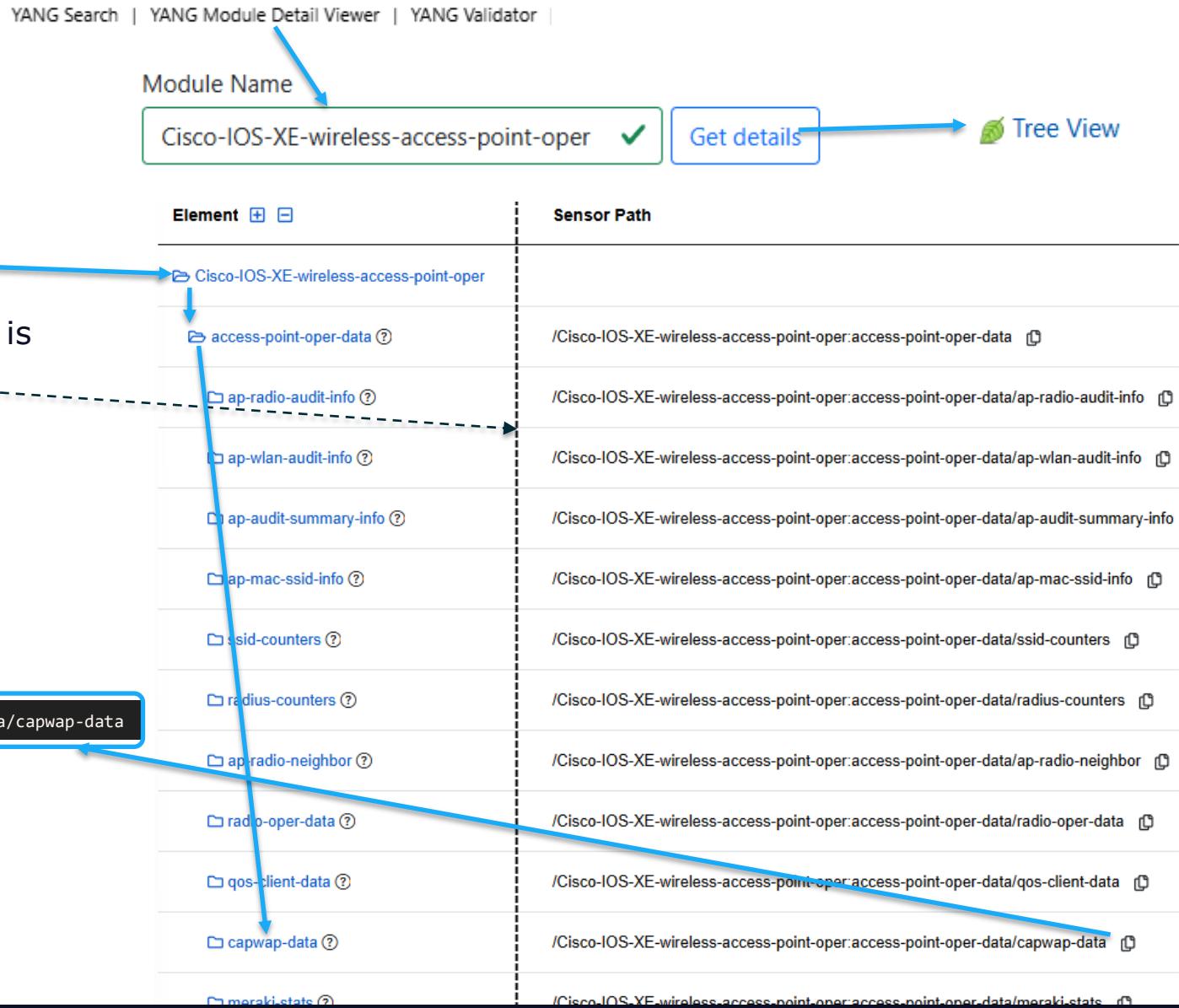
```

1 {
2 "Cisco-IOS-XE-native:native": {
3 "version": "17.15",
4 "boot-start-marker": [
5 null
6],
7 "boot-end-marker": [
8 null
9]
10 }
11 }
```



# Get AP Summary (yangcatalog)

- Create the Request based on the information explored from <https://yangcatalog.org>
  - Expand the tree until you find your module
  - The middle of the website is cut away, since it is too wide to show on this slide
  - Base restconf path:  
`https://{{host}}/restconf/data`



`https://{{host}}/restconf/data/Cisco-IOS-XE-wireless-access-point-oper:access-point-oper-data/capwap-data`

- Copy the Sensor Path after the base restconf path, then create the rest of the query as the instructions from the previous slides



# Get AP summary (YANG Suite)

- Here is an example of how to craft the same Request based on info you find in YANG Suite
  - Restconf path: `https://{{host}}/restconf/data/`
  - YANG-module: `Cisco-IOS-XE-wireless-access-point-oper`
  - Xpath: `access-point-oper-data/capwap-data`
  - Combined request path for copy-paste:

```
https://{{host}}/restconf/data/Cisco-IOS-XE-wireless-access-point-oper:access-point-oper-data/capwap-data
```

|             |                                                                               |
|-------------|-------------------------------------------------------------------------------|
| Name        | capwap-data                                                                   |
| Nodetype    | list                                                                          |
| Description | Captures the information about the 802.11 LWAPP AP that has joined controller |
| Module      | Cisco-IOS-XE-wireless-access-point-oper                                       |
| Revision    | 2023-07-10                                                                    |
| Xpath       | /access-point-oper-data/capwap-data                                           |
| Prefix      | wireless-access-point-oper                                                    |

The screenshot illustrates the process of generating a REST API request for an Access Point summary using the YANG Suite. It shows the following steps:

- The user starts with a 'Cisco 9800 / Get config' request.
- The user performs a 'Copy' operation ('Ctrl+C') on the 'Get config' tab.
- The user pastes the copied URL into the 'Get AP summary' tab.
- The final URL in the 'Get AP summary' tab is: `https://{{host}}/restconf/data/Cisco-IOS-XE-wireless-access-point-oper:access-point-oper-data/capwap-data`.
- The response body shows the JSON data for the 'Cisco-IOS-XE-wireless-access-point-oper:capwap-data' list, which includes fields like 'wtp-mac', 'ip-addr', and 'name'.



# Get Client Summary (yangcatalog.org)

- Create the Request based on the information explored from <https://yangcatalog.org>
  - Expand the tree until you find your module

YANG Search | YANG Module Detail Viewer | YANG Validator |

Module Name: Cisco-IOS-XE-wireless-client-oper

**Get details**

| Element                           | Schema    | Type      | Flags     | Opts | Status  | Path                                                                         | Sensor Path                                                          |
|-----------------------------------|-----------|-----------|-----------|------|---------|------------------------------------------------------------------------------|----------------------------------------------------------------------|
| Cisco-IOS-XE-wireless-client-oper | module    | module    |           |      |         |                                                                              |                                                                      |
| client-oper-data                  | container | container | no config |      | current | /wireless-client-oper:client-oper-data                                       | /Cisco-IOS-XE-wireless-client-oper:client-oper-data                  |
| common-oper-data                  | list      | list      | no config |      | current | /wireless-client-oper:client-oper-data/wireless-client-oper:common-oper-data | /Cisco-IOS-XE-wireless-client-oper:client-oper-data/common-oper-data |

- Base restconf path: `https://{{host}}/restconf/data`

`https://{{host}}/restconf/data/Cisco-IOS-XE-wireless-client-oper:client-oper-data/common-oper-data`

- Copy the Sensor Path after the base restconf path, then create the rest of the query as the instructions from the previous slides



# Get client summary (YANG Suite)

- Here is an example of how to craft the same Request based on info you find in YANG Suite
  - Restconf path: `https://{{host}}/restconf/data/`
  - YANG-module: `Cisco-IOS-XE-wireless-client-oper`
  - Xpath: `client-oper-data/shared-oper-data`
  - Combined request path for copy-paste:

```
https://{{host}}/restconf/data/Cisco-IOS-XE-wireless-client-oper:client-oper-data/common-oper-data
```

|             |                                                                   |
|-------------|-------------------------------------------------------------------|
| Name        | common-oper-data                                                  |
| Nodetype    | list                                                              |
| Description | List containing common operational data of the client             |
| Module      | <code>Cisco-IOS-XE-wireless-client-oper</code>                    |
| Revision    | 2023-07-01                                                        |
| Xpath       | <code>/client-oper-data/common-oper-data</code>                   |
| Prefix      | wireless-client-oper                                              |
| Namespace   | <code>http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-clien</code> |

HTTP Cisco 9800 / Get client summary

GET https://{{host}}/restconf/data/Cisco-IOS-XE-wireless-client-oper:client-oper-data/common-oper-data

RESTCONF path      YANG module      Xpath

The diagram illustrates the mapping between the RESTCONF path and the YANG module information. Blue arrows point from the RESTCONF path segments to the corresponding YANG module and Xpath components in the table above. Specifically, the 'client-oper-data' segment in the path maps to the 'Module' and 'Xpath' entries in the table, and the 'common-oper-data' segment maps to the 'Name' entry.

- If no users are connected, the return is "No Content" (status code 204)

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON

Status: 204 No Content Time: 94 ms Size: 501 B

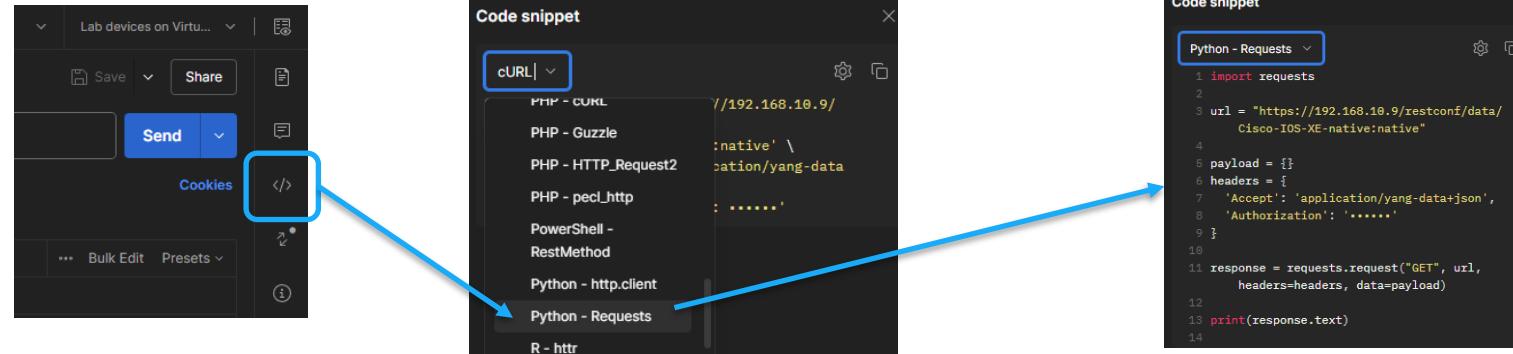
1

The screenshot shows the Postman interface with a successful 204 No Content response. The status bar at the top indicates the status, time, and size. The body section is empty, showing only the number '1'.



# Output to Python

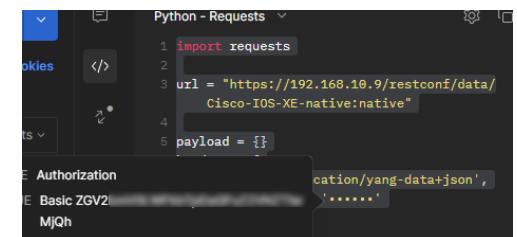
- You can output the RESTCONF call as various code snippets by clicking the "Code" button



- This can be used in your favorite Python editor, or just run it to test

```
PS H:\> python
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import requests
>>>
>>> url = "https://192.168.10.9/restconf/data/Cisco-IOS-XE-native:native"
>>>
>>> payload = {}
>>> headers = {
... 'Accept': 'application/yang-data+json',
... 'Authorization': 'Basic ZGV2bmV0MjU='
... }
>>> response = requests.request("GET", url, headers=headers, data=payload, verify=False)
C:\Users\... Python312\site-packages\urllib3\connectionpool.py:1105: InsecureRequestWarning: Unverified HTTPS request is being made. This is strongly advised. See: https://urllib3.readthedocs.io/en/latest/advanced-usage.html#tls-warning
 warnings.warn(
>>> print(response.text)
{
 "Cisco-IOS-XE-native:native": {
 "version": "17.12",
}
```

You get this by hovering the mouse over the \*\*\*\* text in the Code snippet



Add "verify= False" to skip certificate checking on the host



# Lab exercise #9: Explore Python automation

- In this exercise we will
  - Use the virtual environment "automation-venv" created in Lab 4
  - Connect VS Code to the Ubuntu server
  - Create a Python script that
    - Connects to the WLC using RESTCONF
    - Gets a table of currently connected APs on the WLC
    - Saves the AP list as an Excel file



# Create Python venv

- This lab will use the same venv for Python and Ansible automation. It could be different, and scaling to production and prototyping could often mean having a venv per project

```
devnet-adm@ubuntu-devnet:~$ source ~/automation-venv/bin/activate
(automation-venv) devnet-adm@ubuntu-devnet:~$ python --version
Python 3.12.3
(automation-venv) devnet-adm@ubuntu-devnet:~$ cd wifi-automation
(automation-venv) devnet-adm@ubuntu-devnet:~/wifi-automation$
```

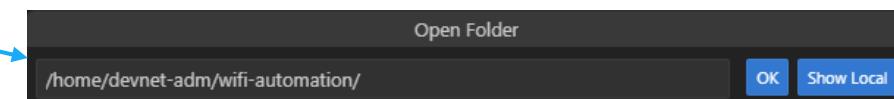
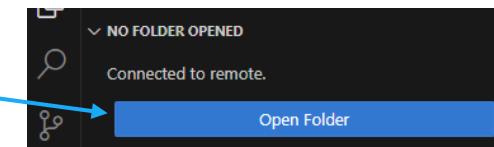
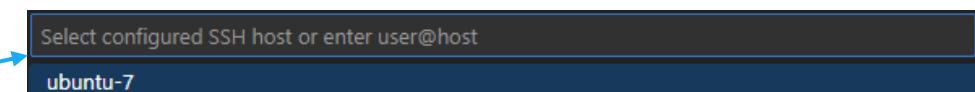
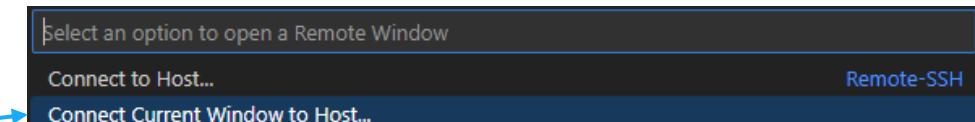
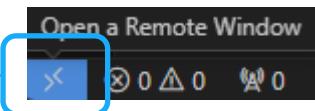
- If you did not do Lab 4, here is a copy of the commands where we created the venv. If you have created and activated your venv already these commands will not do anything (except throw some error messages)

```
devnet-adm@ubuntu-devnet:~$ sudo apt install pip python3-venv
devnet-adm@ubuntu-devnet:~$ mkdir ~/automation-venv
devnet-adm@ubuntu-devnet:~$ python3 -m venv ~/automation-venv/
devnet-adm@ubuntu-devnet:~$ source ~/automation-venv/bin/activate
(automation-venv) devnet-adm@ubuntu-devnet:~$ python --version
```

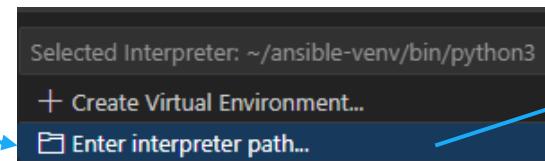


# Connect VS Code to the Ubuntu Server

- Use VS Code remote to the Ubuntu Server (if you are not connected already)
  - Press the Remote SSH icon (lower left corner)
  - Then, in the top line, select "Connect current window to Host..."
  - Select your host, or configure a new (see Lab 3)
  - Select "Open Folder" in the Explorer section
  - Select "/home/devnet-adm/wifi-automation/" and click OK
  - Open the "examples/python-hello-world.py"
  - Select the "automation-venv" venv:  
Click the Python version area on the bottom line



3.12.3 64-bit



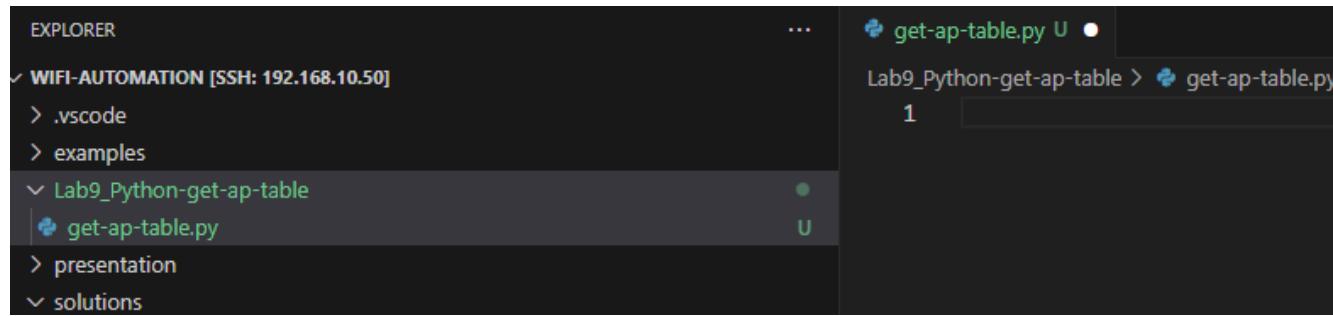
←  
~/automation-venv/bin/python3

3.12.3 ('automation-venv': venv)



# Create the Python file

- Create a folder "Lab9\_Python-get-ap-table" and a python file "get-ap-table.py"
- The new file will automatically open in the editor window



# Prepare the RESTCONF calls

- Using Postman, test the RESTCONF calls to get this info from your WLC
  - This is the URL of the RESTCONF call we will be using in this exercise

- <https://{{host}}/restconf/data/Cisco-IOS-XE-wireless-access-point-oper:access-point-oper-data/capwap-data>

The screenshot shows a Postman interface with a 'GET' method selected. The URL is https://{{host}}/restconf/data/Cisco-IOS-XE-wireless-access-point-oper:access-point-oper-data/capwap-data. The response status is 200 OK. The body of the response is displayed in 'Pretty' JSON format:

```
1 {
2 "Cisco-IOS-XE-wireless-access-point-oper:capwap-data": [
3 {
4 "wtp-mac": "4c:a6:4d:65:f9:40",
5 "ip-addr": "192.168.10.182",
6 "name": "o120E-ekms",
7 }
8]
9 }
```

- You can try specifying some fields to reduce the data size
  - <https://{{host}}/restconf/data/Cisco-IOS-XE-wireless-access-point-oper:access-point-oper-data/capwap-data?fields=wtp-mac;wtp-ip;name;ap-state;device-detail/static-info/board-data/wtp-serial-num;device-detail/static-info/board-data/wtp-enet-mac;device-detail/static-info/ap-models/model;tag-info/tag-source;tag-info/policy-tag-info;tag-info/site-tag;tag-info/rf-tag;tag-info/filter-info/filter-name;ap-time-info/boot-time;ap-time-info/join-time>

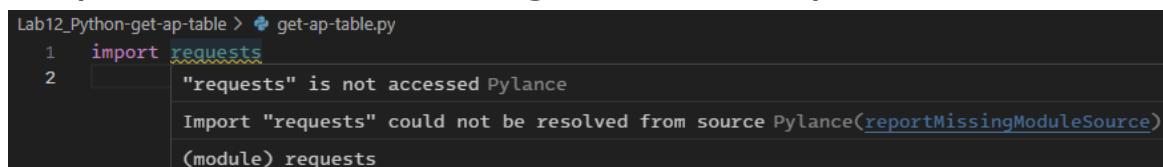


# Prepare Python packages

- For this exercise, we will use the following modules
  - requests
  - pandas
  - openpyxl
- To install these, use the command "pip install requests pandas openpyxl"

```
(automation-venv) devnet-adm@ubuntu-devnet:~/wifi-automation$ cd Lab9_Python-get-ap-table
(automation-venv) devnet-adm@ubuntu-devnet:~/wifi-automation/Lab9_Python-get-ap-table$ pip install requests pandas openpyxl
Collecting requests
 Downloading requests-2.32.3-py3-none-any.whl.metadata (4.6 kB)
Collecting pandas
 Downloading pandas-2.2.2-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (19 kB)
Collecting openpyxl
 ...
Successfully installed certifi-2024.7.4 charset-normalizer-3.3.2 et-xmlfile-1.1.0 idna-3.8 numpy-2.1.0 openpyxl-3.1.5 pandas-2.2.2 python-dateutil-2.9.0.post0 pytz-2024.1 requests-2.32.3 six-1.16.0 tzdata-2024.1 urllib3-2.2.2
(automation-venv) devnet-adm@ubuntu-devnet:~/wifi-automation/Lab9_Python-get-ap-table$
```

- If you want, you can wait installing these until you see the errors highlighted in VS Code when we try to use them



A screenshot of a VS Code editor window. The file is named 'get-ap-table.py'. The code contains two lines: 'import requests' and 'print(requests)'. A tooltip from the Pylance extension highlights the word 'requests' in the second line with the message "'requests' is not accessed Pylance'. Below it, another message states 'Import "requests" could not be resolved from source Pylance(reportMissingModuleSource)'.

Make sure you are in the "automation-venv", indicating by the text (automation-venv) before each line in the terminal. If not, you will get an error message  
**error: externally-managed-environment**



# Get AP table

```

1 import requests
2 import pandas as pd
3 import getpass
4
5 wlc = input("Enter WLC IP: ")
6 user = input("Enter user: ")
7 password = getpass.getpass("Enter password: ")
8 url = f"https://{{wlc}}/restconf/data/Cisco-IOS-XE-wireless-access-point-oper:access-point-oper-data?fields=wtp-mac;wtp-ip;name;ap-state;device-detail/static-info/board-data/wtp-serial-num;device-detail/static-info/board-data/wtp-enet-mac;device-detail/static-info/ap-models;model;tag-info/tag-source;tag-info/policy-tag-info;tag-info/site-tag;tag-info/rf-tag;tag-info/filter-info/filter-name;ap-time-info/boot-time;ap-time-info/join-time"
9
10 payload = {}
11 headers = {
12 'Accept': 'application/yang-data+json',
13 'Content-Type': 'application/yang-data+json'
14 }
15
16 response = requests.get(url, auth=(user, password), headers=headers,
17
18 if (response.status_code==200):
19 ap_table = pd.json_normalize(response.json()['Cisco-IOS-XE-wireless-access-point-oper:access-point-oper-data'])
20 print(ap_table)
21 ap_table.to_excel('ap_table.xlsx')
22 ap_table.to_csv('ap_table.csv')
23 else:
24 print(f"Status code: {{response.status_code}}: {{response.reason}}")
25

```

- Import the modules we will use

- WLC IP, username and password as input fields, it will be asked when running the script

- Create the RESTCONF call using the IP of your WLC, and the path you tested with Postman

- Make the call and save the response
- Some very basic error checking
- The response is given as JSON. We use a pandas function "json\_normalize" to flatten parts of the JSON object to present it in a table
- Print the table in the terminal
- Save the table to Excel and CSV

get-ap-table.py

```

import requests
import pandas as pd
import getpass

wlc = input("Enter WLC IP: ")
user = input("Enter user: ")
password = getpass.getpass("Enter password: ")
url = f"https://{{wlc}}/restconf/data/Cisco-IOS-XE-wireless-access-point-oper:access-point-oper-data?fields=wtp-mac;wtp-ip;name;ap-state;device-detail/static-info/board-data/wtp-serial-num;device-detail/static-info/board-data/wtp-enet-mac;device-detail/static-info/ap-models;model;tag-info/tag-source;tag-info/policy-tag-info;tag-info/site-tag;tag-info/rf-tag;tag-info/filter-info/filter-name;ap-time-info/boot-time;ap-time-info/join-time"

payload = {}
headers = {
 'Accept': 'application/yang-data+json',
 'Content-Type': 'application/yang-data+json'
}

response = requests.get(url, auth=(user, password), headers=headers, data=payload, verify=False)

if (response.status_code==200):
 ap_table = pd.json_normalize(response.json()['Cisco-IOS-XE-wireless-access-point-oper:access-point-oper-data'])
 print(ap_table)
 ap_table.to_excel('ap_table.xlsx')
 ap_table.to_csv('ap_table.csv')
else:
 print(f"Status code: {{response.status_code}}: {{response.reason}}")

```



# Running the Python script from terminal

- One way to run a Python script, is by using the terminal. When in the folder of the script, run using "python" and the filename

```
(automation-venv) devnet-adm@ubuntu-devnet:~/wifi-automation$ cd Lab9_Python-get-ap-table
(automation-venv) devnet-adm@ubuntu-devnet:~/wifi-automation/Lab9_Python-get-ap-table$ ls -l
total 4
-rw-rw-r-- 1 devnet-adm devnet-adm 978 Aug 25 19:24 get-ap-table.py
(automation-venv) devnet-adm@ubuntu-devnet:~/wifi-automation/Lab9_Python-get-ap-table$ python get-ap-table.py
Enter WLC IP: 192.168.10.30 ←
Enter user: devnet-adm
Enter password:
```

Use YOUR WLC IP

- If there are no APs on the WLC it will have status code 204 (No Content). So it will not print the table etc, if it tried it would produce an error on the "json()[blablabla]" part"

```
Status code: 204: No Content
```

- Successful run will look something like this:

| wtp-mac             | name             | wtp-ip         | ... | ap-state.ap-operation-state | ap-time-info.boot-time    | ap-time-info.join-time           |
|---------------------|------------------|----------------|-----|-----------------------------|---------------------------|----------------------------------|
| 0 4c:a6:4d:65:f9:40 | 9120E-ekms       | 192.168.10.182 | ... | registered                  | 2024-07-25T00:28:12+00:00 | 2024-07-25T00:30:04.335182+00:00 |
| 1 d4:2c:44:d9:24:40 | APD42C-44D8-2F38 | 192.168.10.177 | ... | registered                  | 2024-07-31T19:22:07+00:00 | 2024-07-31T19:23:42.643078+00:00 |

- Wrong username/password will look something like this:

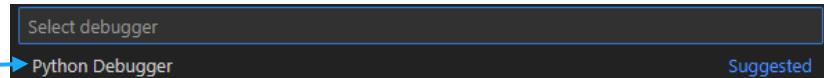
```
Status code: 401: Unauthorized
```



# Running the Python script from VS Code

- Press Ctrl+F5 to run the script in VS Code

- Select "Python Debugger" if asked



```
(python-lab-venv) devnet-adm@ubuntu-devnet:~/wifi-automation$ /usr/bin/env /home/devnet-adm/python-lab-venv/bin/python3 /home/devnet-adm/.vscode-server/extensions/ms-python.debugpy-2024.10.0-linux-x64/bundled/libs/debugpy/adapter/../../debugpy/launcher 57251 -- /home/devnet-adm/wifi-automation/Lab6_Python-get-ap-table/get-ap-table.py
Enter WLC IP: 192.168.10.30
Enter user: devnet-adm
Enter password:
/home/devnet-adm/python-lab-venv/lib/python3.12/site-packages/urllib3/connectionpool.py:1099: InsecureRequestWarning: Unverified HTTPS request is
being made to host '192.168.10.30'. Adding certificate verification is strongly advised. See: https://urllib3.readthedocs.io/en/latest/advanced-us
age.html#tls-warnings
 warnings.warn(
Status code: 204: No Content
(python-lab-venv) devnet-adm@ubuntu-devnet:~/wifi-automation$
```

- The Excel file will contain the info, but will be otherwise unformatted. You can download the file by right-clicking it and selecting "Download..." and save to a folder of your choice on your local machine.

| A | B                   | C             | D         | E                                | F                                | G                                              | H          | I                  | J                 | K         | L         | M          | N                          | O                          | P | Q | R | S | T |
|---|---------------------|---------------|-----------|----------------------------------|----------------------------------|------------------------------------------------|------------|--------------------|-------------------|-----------|-----------|------------|----------------------------|----------------------------|---|---|---|---|---|
|   | wtp-mac             | name          | wtp-ip    | info.board.info.boardatic-info.a | info.tag-sotag-info.pte-tag.site | site-tag.apsite-tag.flrf-tag.rf-tater-info.fil | ap-admin   | ap-operate-info.bo | de-info.join-time |           |           |            |                            |                            |   |   |   |   |   |
| 0 | 4c:a6:4d:69120E-ekn | 192.168.10.30 | FGL2417LS | a4:b4:39:2C9120AXE               | tag-source:k32-stand:k32         | koksrud-a                                      | koksrud-fl | k32-rf             | Staging-re        | adminstat | register  | 2024-07-21 | 2024-07-25T00:30:04.335187 |                            |   |   |   |   |   |
| 1 | d4:2c:44:dAPD42C-4  | 192.168.10.30 | KWC20380  | d4:2c:44:dAIR-AP18E              | tag-source:Empty                 | k32                                            | koksrud-a  | koksrud-fl         | k32-rf            | Empty     | adminstat | register   | 2024-07-31                 | 2024-07-31T19:23:42.643078 |   |   |   |   |   |

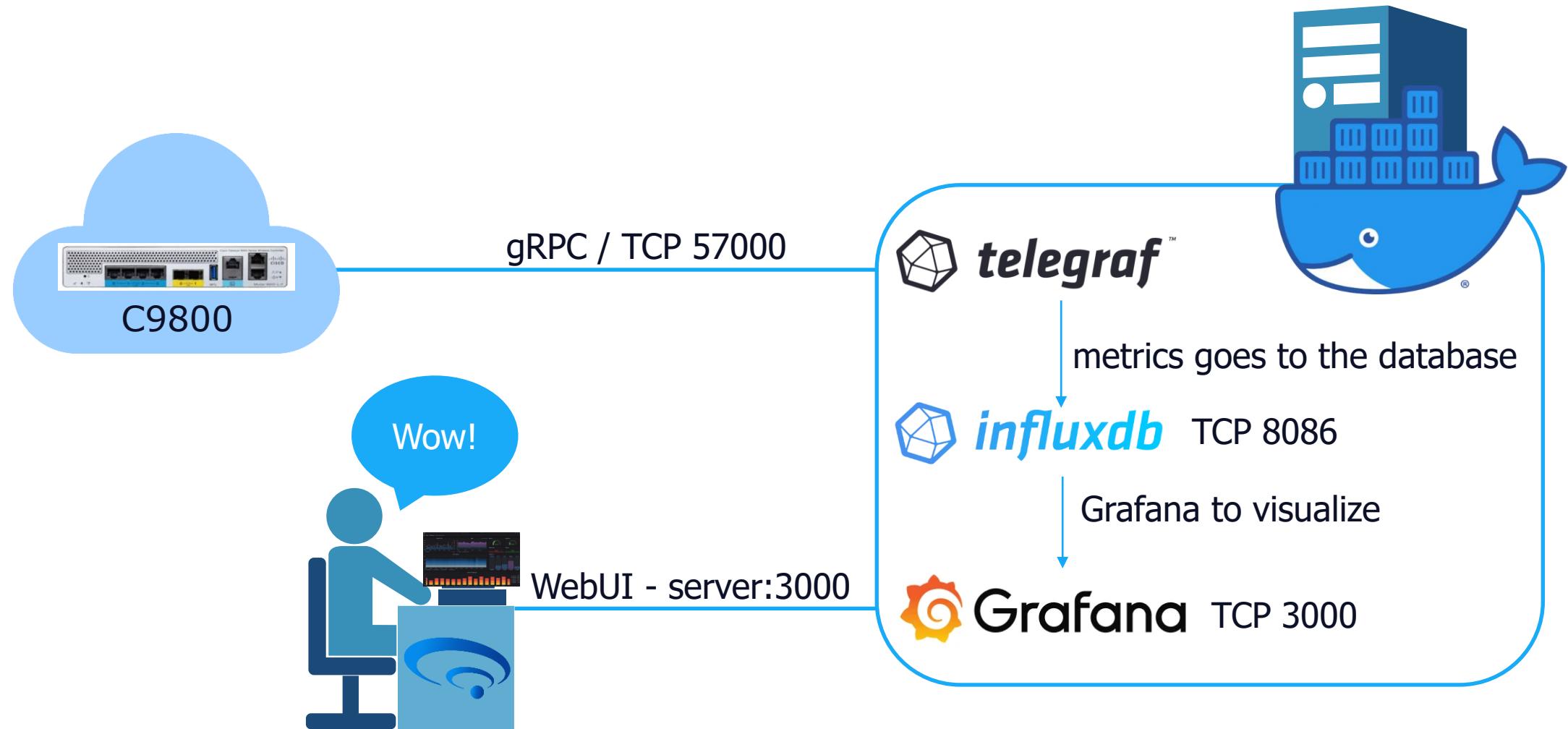
- You can open the CSV file in VS Code, or in the terminal using "cat ap\_table.csv" (or several alternatives)



# Lab exercise #10: Install TIG stack (optional)

- **!!! NOTE !!!**
  - This exercise is marked as optional. However, if you plan on doing more Grafana stuff on day 2 we highly recommend that you do this, even if it will take some extra time today, or continue on to day 2
  - If you plan doing primarily Python or Ansible stuff on day 2, you can skip exercise 10 (installing TIG stack) and skip to Lab exercise 11. You can then use the installation on one of the pre-installed Ubuntu Servers.
- In this lab exercise, we will install "TIG Stack" as a Docker container on the Ubuntu Server
- As part of the installation we will
  - Create .env file containing variables
  - Ensure persistant storage
  - Create new bucket and adapt it so we can use InfluxDB
- The next slides will give some basic overview of the TIG Stack, and how all fits together
  - T = Telegraf
  - I = InfluxDB
  - G = Grafana

# TIG Stack overview



# The TIG in the TIG stack



- **Open Source Agent:** Collects and sends metrics from databases, systems, and IoT devices
- **Versatile Monitoring:** Works with databases, systems, and IoT sensors
- **Time Series Data at Scale:** Store and query large volumes of time-stamped data
- **Real Time Analytics:** Fast processing for immediate insights
- **Seamless Integration:** Compatible with various tools and platforms
- **Visualize Anything:** Create dashboards for any data
- **Real Time Insights:** Customizable, interactive visuals
- **Flexible Integration:** Connects to multiple data sources

# Different flavours of Influx

InfluxDB 2.x - FLUX

The screenshot shows the InfluxDB 1.x user interface. It has a top navigation bar with tabs like 'A', 'B', 'C', etc. Below the navigation is a query editor with several sections: 'FROM' (set to 'default'), 'SELECT' (containing 'field (value)', 'integral ()', 'cumulative\_sum ()', and 'math (/ 3600)'), 'GROUP BY' (set to 'time (10s)'), 'FORMAT AS' (set to 'Time series'), and 'ALIAS BY' (set to 'Mains Power'). The 'WHERE' section is expanded, showing a condition 'entity\_id = efergy\_815686'. There are also '+' buttons for adding more conditions or clauses.

```

1 from(bucket: "homeassistant")
2 |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
3 |> filter(fn: (r) => r["entity_id"] == "efergy_815686")
4 |> filter(fn: (r) => r["_measurement"] == "W")
5 |> filter(fn: (r) => r["_field"] == "value")
6 |> map(fn: (r) => ({ r with _value: float(v: r._value) / 3600.0 })
7 |> aggregateWindow(every: v.windowPeriod, fn: sum)
8 |> cumulativeSum()
9 |> yield(name: "sum")

```

InfluxDB 1.x - InfluxQL

We will use InfluxDB 2.x but query data using InfluxQL

The screenshot shows the InfluxDB 3.x user interface. At the top, it says '(TGN-InfluxDB Cloud)' and has a 'Format: Table' dropdown. Below is a code editor containing the following InfluxQL query:

```

1 SELECT "time", "RSSI", "NOISE"
2 FROM "data"
3 WHERE
4 time >= $__timeFrom() AND time <= $__timeTo()
5 AND ("RSSI" IS NOT NULL OR "NOISE" IS NOT NULL)
6 AND "AppleDevice" IN ('Wi-Co')
7

```

InfluxDB 3.x – InfluxQL (v2)

# Make some notes

- There are some variables that need to be typed later, so to save time you can make a .txt file while we continue in this lab

Lab10\_Notes.txt

```
Admin token: {we will fill in this later}
Org ID: {we will fill in this later}
Container ID: {we will fill in this later}
```

- Even better, you could use your favorite password manager (KeePass etc)

# TIG stack installation

## 1. Create a .env file to store variables

```
devnet-adm@ubuntu-devnet:~$ cd ~
devnet-adm@ubuntu-devnet:~$ mkdir tig-stack
devnet-adm@ubuntu-devnet:~$ cd tig-stack
devnet-adm@ubuntu-devnet:~/tig-stack$ nano .env
!
#add the following, and change the variables if needed
!
```

.env

```
INFLUXDB_ADMIN_USER=devnet-adm
INFLUXDB_ADMIN_PASSWORD=ChangeMe2025!
INFLUXDB_ORG=devnet
INFLUXDB_BUCKET=c9800-bucket
INFLUXDB_ADMIN_TOKEN=ChangeMe2025!
GF_SECURITY_ADMIN_USER=devnet-adm
GF_SECURITY_ADMIN_PASSWORD=ChangeMe2025!
```

– Add the INFLUXDB\_ADMIN\_TOKEN to your .txt file

lab10\_notes.txt

```
1 Admin token: ChangeMe2025!
```

## 2. Create persistent storage

```
devnet-adm@ubuntu-devnet:~/tig-stack$ docker volume create grafana-data
devnet-adm@ubuntu-devnet:~/tig-stack$ docker volume create influxdb-data
```

## 3. Create the Telegraf config file

```
devnet-adm@ubuntu-devnet:~/tig-stack$ mkdir -p telegraf/conf
devnet-adm@ubuntu-devnet:~/tig-stack$ nano telegraf/conf/telegraf.conf
```

```
[global_tags]
host = "My-Telegraf"
[agent]
interval = "10s"
round_interval = true
metric_batch_size = 1000
metric_buffer_limit = 10000
collection_jitter = "0s"
flush_interval = "10s"
flush_jitter = "0s"
precision = ""
debug = false
quiet = false
logfile = ""
hostname = "My-Telegraf"
omit_hostname = false

#####
OUTPUT PLUGINS
#####

Configuration for influxdb server to send metrics to

[[outputs.influxdb_v2]]
urls = ["http://influxdb:8086"]
token = "${INFLUXDB_ADMIN_TOKEN}"
organization = "${INFLUXDB_ORG}"
bucket = "${INFLUXDB_BUCKET}"
tagexclude = ["influxdb_database"] ### Optional: Exclude specific tags if needed
[outputs.influxdb_v2.tagpass]
influxdb_database = ["${INFLUXDB_BUCKET}"]

#####
INPUT PLUGINS
#####

[[inputs.cisco_telemetry_mdt]]
transport = "grpc"
service_address = ":57000"
[inputs.cisco_telemetry_mdt.tags]
influxdb_database = "${INFLUXDB_BUCKET}"
```



# TIG stack installation

## 1. Configure docker-compose.yml

```
devnet-adm@ubuntu-devnet:~$ cd ~/tig-stack
devnet-adm@ubuntu-devnet:~$ nano ./docker-compose.yml
```

Add the following config to docker-compose.yml



## 2. Docker compose

```
devnet-adm@ubuntu-devnet:~$ docker compose up -d
```

- This will take a few seconds to download all images.
- To view logs from a service (telegraf, influxdb, grafana) use this

```
devnet-adm@ubuntu-devnet:~$ docker logs telegraf
```

## 3. Connect to Grafana using web browser

- Notice we use HTTP, not HTTPS for this lab install
- <http://192.168.10.{your Ubuntu IP}:3000>
- For now, just test that you can log in using the username and password from the .env file we created earlier. For this lab it will be the same as everywhere else unless you changed it ☺

```
services:
 influxdb:
 container_name: influxdb
 image: influxdb:2.7.10
 expose:
 - 8086
 restart: always
 volumes:
 - influxdb-data:/var/lib/influxdb2
 ports:
 - 8086:8086
 environment:
 - DOCKER_INFLUXDB_INIT_MODE=setup
 - DOCKER_INFLUXDB_INIT_USERNAME=${INFLUXDB_ADMIN_USER}
 - DOCKER_INFLUXDB_INIT_PASSWORD=${INFLUXDB_ADMIN_PASSWORD}
 - DOCKER_INFLUXDB_INIT_ORG=${INFLUXDB_ORG}
 - DOCKER_INFLUXDB_INIT_BUCKET=${INFLUXDB_BUCKET}
 - DOCKER_INFLUXDB_INIT_RETENTION=2w
 - DOCKER_INFLUXDB_INIT_ADMIN_TOKEN=${INFLUXDB_ADMIN_TOKEN}

 telegraf:
 container_name: telegraf
 image: telegraf:1.32.0
 restart: always
 ports:
 - 57000:57000
 env_file:
 - .env
 environment:
 - INFLUXDB_TOKEN=${INFLUXDB_ADMIN_TOKEN}
 - INFLUXDB_ORG=${INFLUXDB_ORG}
 - INFLUXDB_BUCKET=${INFLUXDB_BUCKET}
 volumes:
 - ./telegraf/conf/telegraf.conf:/etc/telegraf/telegraf.conf:ro
 - /var/run/docker.sock:/var/run/docker.sock
 depends_on:
 - influxdb

 grafana:
 container_name: grafana
 image: grafana/grafana:11.1.4
 expose:
 - 3000
 restart: always
 ports:
 - 3000:3000
 environment:
 - GF_SECURITY_ADMIN_USER=${GF_SECURITY_ADMIN_USER}
 - GF_SECURITY_ADMIN_PASSWORD=${GF_SECURITY_ADMIN_PASSWORD}
 - GF_INSTALL_PLUGINS=
 volumes:
 - grafana-data:/var/lib/grafana
 #- ./grafana/grafana.ini:/etc/grafana/grafana.ini:ro
 depends_on:
 - influxdb
 - telegraf

volumes:
 grafana-data:
 external: true
 influxdb-data:
 external: true
```



# InfluxDB 2.X

<http://192.168.10.{your Ubuntu IP}:8086>

- Username/password from .env file (same as always unless changed)
- Notice that we here also use HTTP, not HTTPS
- Change retention on your bucket to f.ex 31 days.

The screenshot shows the InfluxDB 2.0 interface. The top navigation bar includes tabs for SOURCES, BUCKETS, TELEGRAF, SCRAPERS, and API TOKENS. The left sidebar lists SOURCES, BUCKETS, TELEGRAF, SCRAPERS, and API TOKENS. The main area displays a table with columns for Name, Type, and Status. A modal window titled 'Edit Bucket' is open, showing a 'Name' field with 'c9800-bucket'. Below it, a note says 'To rename bucket use the RENAME button below'. Under 'Delete Data', there are two options: 'NEVER' (selected) and 'OLDER THAN', with '31 days' chosen. At the bottom of the modal are 'CANCEL', 'RENAME' (in red), and 'SAVE CHANGES' (in blue) buttons. A large blue arrow points from the 'BUCKETS' tab in the top navigation to the 'Edit Bucket' modal. Another blue arrow points from the 'CREATE BUCKET' button in the main area to the explanatory text about buckets.

The screenshot shows the 'Edit Bucket' dialog box. It has a 'Name\*' field containing 'c9800-bucket'. Below it is a note: 'To rename bucket use the RENAME button below'. Under 'Delete Data', there are two options: 'NEVER' (selected) and 'OLDER THAN', with '31 days' chosen. At the bottom are 'CANCEL', 'RENAME' (highlighted in red), and 'SAVE CHANGES' buttons.

# InfluxDB 2.X

<http://192.168.10.{your Ubuntu IP}:8086>

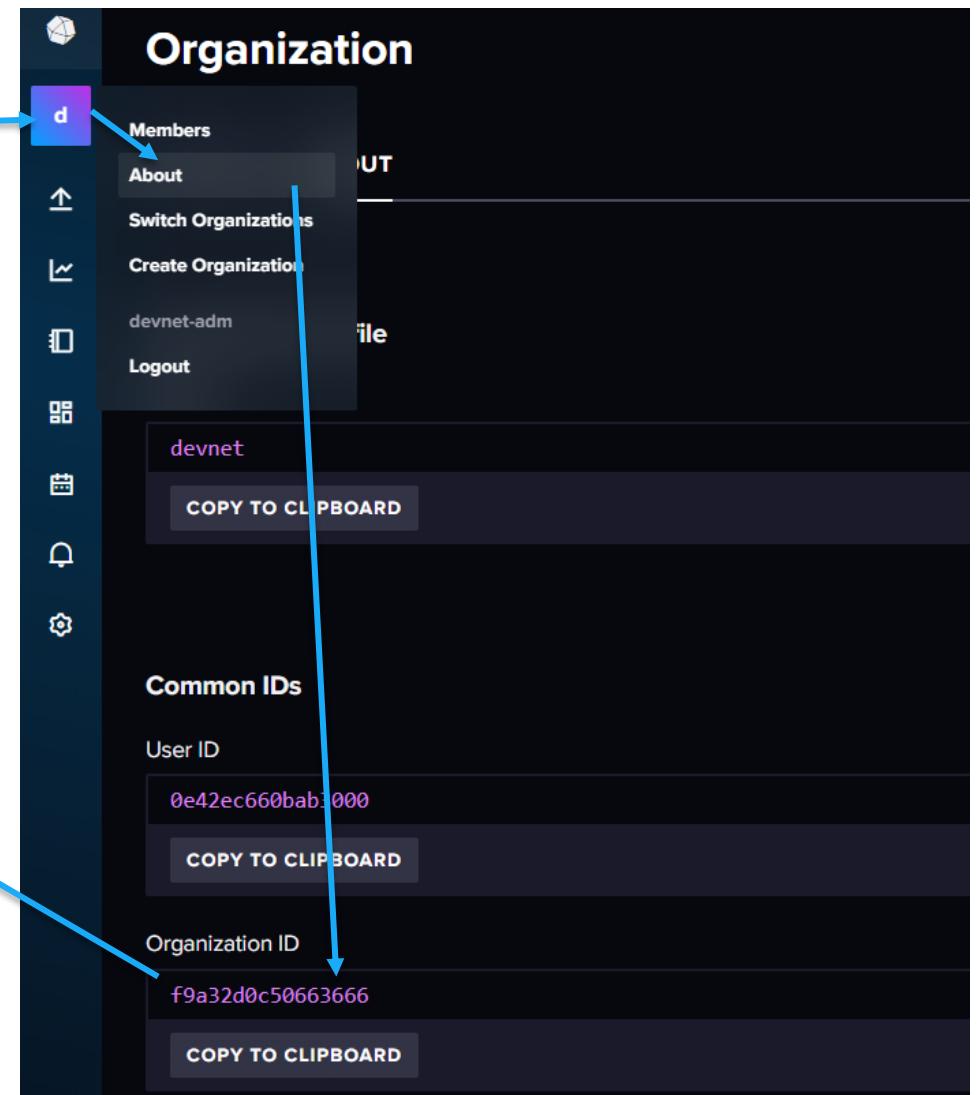
- Create a an additonal bucket
  - Name it: **syslog-bucket**
- Set retention to 30 days
- You can find the bucket-ID here

The screenshot shows the InfluxDB 2.0 web interface. At the top, there are tabs for SOURCES, BUCKETS, TELEGRAF, SCRAPERS, and API TOKENS. The BUCKETS tab is selected, and the main area displays three existing buckets: 'logos-bucket', 'IoT-bucket', and 'syslog-bucket'. Each bucket has a retention policy (e.g., 31 days, Forever) and an ID. A search bar at the top allows filtering by bucket name. To the right of the buckets, there is a modal window titled 'Create Bucket' with fields for 'Name\*' (set to 'system-bucket') and 'Delete Data' (set to 'OLDER THAN 30 days'). Below the modal, a purple sidebar provides information about what a bucket is and how to write data into it. A blue arrow points from the 'Create a an additional bucket' bullet point in the list above to the 'CREATE BUCKET' button in the modal. Another blue arrow points from the 'Retention: 30 days' bullet point in the list to the '30 days' dropdown in the 'Delete Data' section of the modal. A third blue arrow points from the 'You can find the bucket-ID here' bullet point in the list to the 'ID: 32f34d881b644a61' field in the 'syslog-bucket' card below.

# Locate your Organization ID

- InfluxDB > About
- Note this down in the Lab10\_notes.txt file

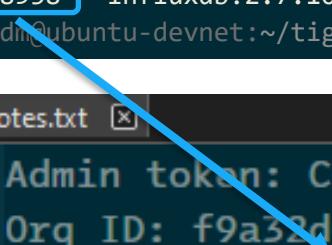
```
lab10_notes.txt
1 Admin token: ChangeMe2025!
2 Org ID: f9a32d0c50663666
3 Container ID: {we will find in this later}
```



# Locate your InfluxDB Docker container ID

- Note this down in the Lab10\_notes.txt file

```
devnet-adm@ubuntu-devnet:~/tig-stack$ docker ps -a --filter name=influxdb
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
71b07ba58958 influxdb:2.7.10 "/entrypoint.sh infl..." 2 minutes ago Up 2 minutes 0.0.0.0:8086->8086/tcp, :::8086->8086/tcp influxdb
devnet-adm@ubuntu-devnet:~/tig-stack$
```



```
lab10_notes.txt
1 Admin token: ChangeMe2025!
2 Org ID: f9a32d0c50663666
3 Container ID: 71b07ba58958
```

- In the notes, you will then have
  - Admin token
  - Org ID
  - Container ID
- For a live installation you should have at least one RO token, or separate tokens for each bucket. Not use the admin token for everything like we do here ☺

# InfluxDB 2.X: Query data in Grafana with InfluxQL

- We will use the values from our lab notes
  - In v1 mode, it is called a "database"

```

devnet-adm@ubuntu-devnet:~/tig-stack$ docker exec -it <your_container-id> bash
root@71b07ba58958:/# influx v1 dbrp list
Error: must specify org ID or org name
root@71b07ba58958:/# influx v1 dbrp list --org-id <your_org-id> --token <your_admin_token>
ID Database Bucket ID Retention Policy Default Organization ID
VIRTUAL DBRP MAPPINGS (READ-ONLY)

ID Database Bucket ID Retention Policy Default Organization ID
b53d2ec9d932635a _monitoring b53d2ec9d932635a autogen true 6a6f13e3a65de450
09aac3a6cee8ab99 _tasks 09aac3a6cee8ab99 autogen true 6a6f13e3a65de450
574c74f705d86be2 syslog-bucket 294c74f678d86ab3 autogen true 6a6f13e3a65de450
574c74f705d86be2 c9800-bucket 574c74f705d86be2 autogen true 6a6f13e3a65de450

root@71b07ba58958:/# influx v1 dbrp create --db c9800-db --rp c9800-rp --bucket-id 574c74f705d86be2 --default --org-id <your_org-id> --token <your_admin_token>
root@71b07ba58958:/# influx v1 dbrp create --db syslog-db --rp syslog-rp --bucket-id 294c74f678d86ab3 --default --org-id <your_org-id> --token <your_admin_token>

ID Database Bucket ID Retention Policy Default Organization ID
0d94a420be168000 c9800-db bc926b463de1d39a c9800-rp true 6a6f13e3a65de450

root@71b07ba58958:/# exit
devnet-adm@ubuntu-devnet:~/tig-stack$
```

Insert YOUR container-id

Insert YOUR org-id

Insert YOUR admin token

Repeat this step for the syslog-bucket

- We now have
  - a c9800-bucket where we can query using FLUX
  - a c9800-db where we can query the same data, using InfluxQL
  - See earlier slide "Different flavors of InfluxDB"

# A bit about telemetry subscriptions

- Below are a couple of sample "telemetry subscriptions"
- Telemetry subscriptions are the same as Catalyst Center use to get telemetry data from IOS-XE devices
- We will use telemetry subscriptions to get data from the 9800 over to our TIG stack  
(9800 -> Telegraf -> InfluxDB -> Grafana)
- The syntax is inside configuration mode (conf t) on 9800
  - Each subscription has an ID
  - The filter is built from the YANG model
  - Update policy decide how often 9800 will push this to Telegraf (in 100ths of a second -> 500 = 5 sec)
  - Receiver IP is the IP of Telegraf (in this lab: your Ubuntu server)
- You will not configure these now, they are only examples. We will use some other data ☺

```
telemetry ietf subscription 101
encoding encode-kvvpb
filter xpath /ios:native/version
stream yang-push
update-policy periodic 30000
receiver ip address 192.168.10.7 57000 protocol grpc-tcp
```

```
telemetry ietf subscription 102
encoding encode-kvvpb
filter xpath /process-cpu-ios-xe-oper:cpu-usage/cpu-utilization/five-seconds
stream yang-push
update-policy periodic 500
receiver ip address 192.168.10.7 57000 protocol grpc-tcp
```

# Lab exercise #11: Build your first Grafana Dashboard

- In this exercise you will be hand-held to build your first dashboard in Grafana
- The dashboard will be an AP Dashboard
- More dashboards to be explored on day 2
- !!! A word of caution when making your first dashboards... don't use "Esc" key without saving, you might have to do some work again. You will probably do just that, and THEN learn it the hard way ☺ !!!
- If you are doing this lab exercise using TIG Stack on a shared server, use your shared Ubuntu Server referring to the Lab topology:

| Students                         | Shared Ubuntu Server     | Grafana URL                                                       | InfluxDB URL                                                      |
|----------------------------------|--------------------------|-------------------------------------------------------------------|-------------------------------------------------------------------|
| Harsha, Craig, Sean, Raheel      | ubuntu11 (192.168.10.11) | <a href="http://192.168.10.11:3000">http://192.168.10.11:3000</a> | <a href="http://192.168.10.11:8086">http://192.168.10.11:8086</a> |
| Kenneth, Daniel J, Biruk         | ubuntu12 (192.168.10.12) | <a href="http://192.168.10.12:3000">http://192.168.10.12:3000</a> | <a href="http://192.168.10.12:8086">http://192.168.10.12:8086</a> |
| Joseph, Matt, Jason              | ubuntu13 (192.168.10.13) | <a href="http://192.168.10.13:3000">http://192.168.10.13:3000</a> | <a href="http://192.168.10.13:8086">http://192.168.10.13:8086</a> |
| Jack, Daniel W, Harsh, Chris     | ubuntu14 (192.168.10.14) | <a href="http://192.168.10.14:3000">http://192.168.10.14:3000</a> | <a href="http://192.168.10.14:8086">http://192.168.10.14:8086</a> |
| Brian, James, Szymon, Todd       | ubuntu15 (192.168.10.15) | <a href="http://192.168.10.15:3000">http://192.168.10.15:3000</a> | <a href="http://192.168.10.15:8086">http://192.168.10.15:8086</a> |
| Terry, Eddie, Sitarama, Francois | ubuntu16 (192.168.10.16) | <a href="http://192.168.10.16:3000">http://192.168.10.16:3000</a> | <a href="http://192.168.10.16:8086">http://192.168.10.16:8086</a> |

# Add the following telemetry subscriptions on the WLC

```
wlc9# conf t
```

```
telemetry ietf subscription 105
encoding encode-kvvpb
filter xpath /wireless-access-point-oper:access-point-oper-data/wireless-access-point-oper:ap-name-mac-map
stream yang-push
update-policy periodic 30000
receiver ip address 192.168.10.7 57000 protocol grpc-tcp
do show telemetry ietf subscription 105
```

Change this to your Ubuntu IP (or IP of the TIG stack you are using)

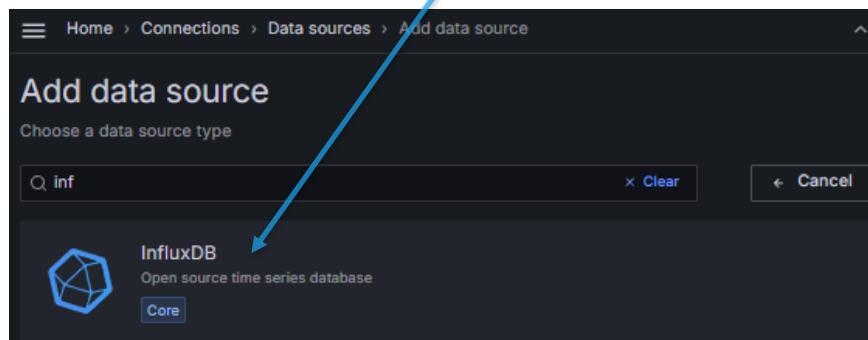
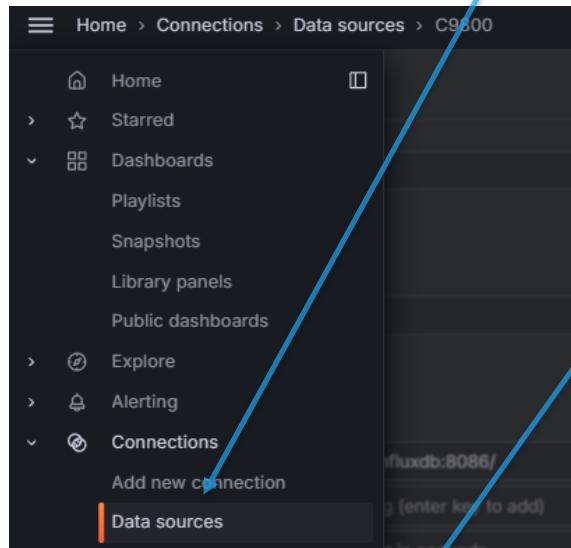
```
telemetry ietf subscription 106
encoding encode-kvvpb
filter xpath /wireless-rrm-oper:rrm-oper-data/wireless-rrm-oper:rrm-measurement/wireless-rrm-oper:load
stream yang-push
update-policy periodic 5000
receiver ip address 192.168.10.7 57000 protocol grpc-tcp
do show telemetry ietf subscription 106
```

Change this to your Ubuntu IP (or the IP of the TIG stack you are using)

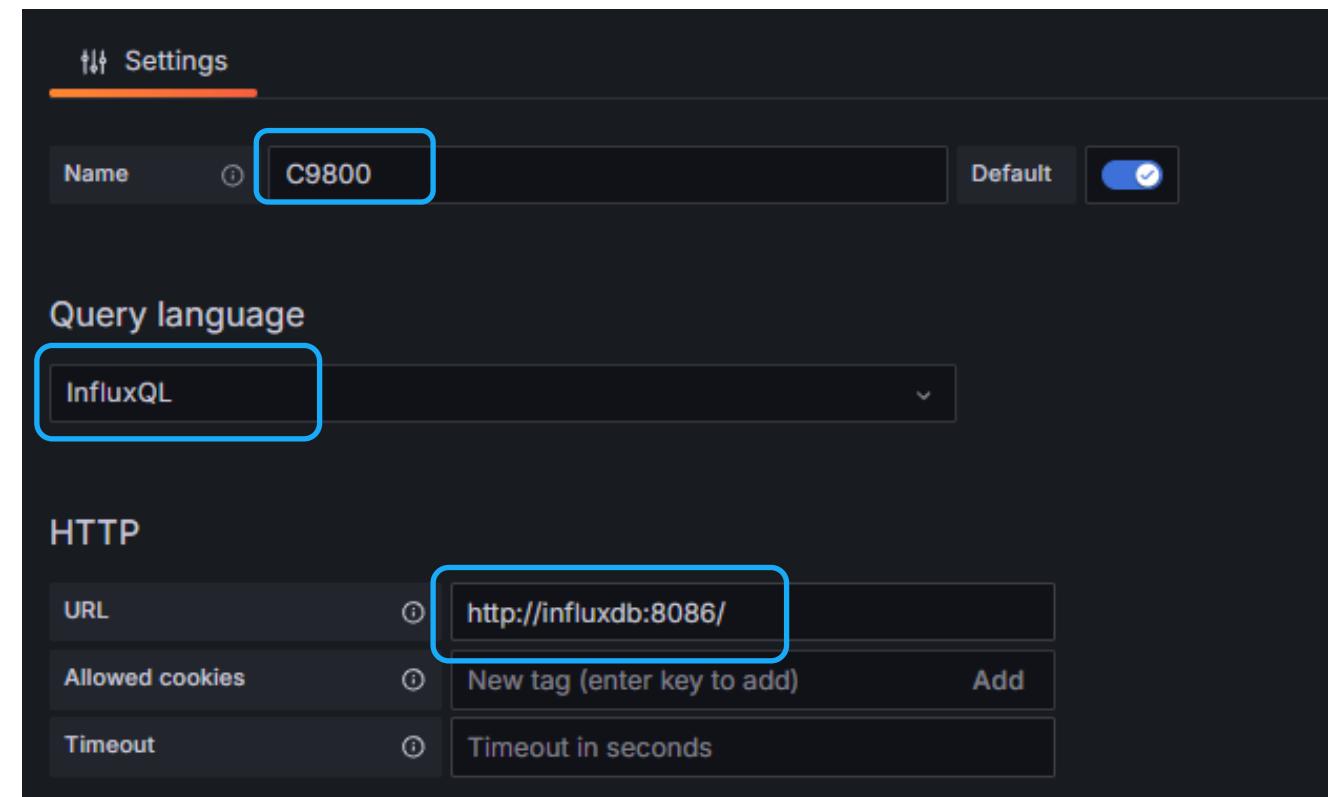
```
telemetry ietf subscription 107
encoding encode-kvvpb
filter xpath /wireless-access-point-oper:access-point-oper-data/wireless-access-point-oper:radio-oper-data/wireless-access-
point-oper:phy-ht-cfg
stream yang-push
update-policy periodic 5000
receiver ip address 192.168.10.7 57000 protocol grpc-tcp
do show telemetry ietf subscription 107
```

# Grafana (<http://{{Ubuntu-IP}}:3000>)

- Add your first data source -> InfluxDB



- Set a name for your database
- Set Query language to InfluxQL
- URL: <http://influxdb:8086>



# Grafana (<http://{{Ubuntu-IP}}:3000>)

- Grafana needs to use a token to be able to read from the database, and InfluxDB password.
  - For this lab we will use the Admin token
  - In a live system, create a RO token and use it here
- Add a custom HTTP Headers
  - Header: Authorization
  - Value: **Token <API-TOKEN>**
  - There is a space between Token and <token>**

Custom HTTP Headers

|                              |               |       |                     |       |  |
|------------------------------|---------------|-------|---------------------|-------|--|
| Header                       | Authorization | Value | Token ChangeMe2025! | Reset |  |
| <a href="#">+ Add header</a> |               |       |                     |       |  |

- Database: c9800-db
  - User/Pass: devnet-adm / ChangeMe2025!

|             |               |
|-------------|---------------|
| Database    | c9800-db      |
| User        | devnet-adm    |
| Password    | ChangeMe2025! |
| HTTP Method | GET           |

[Reset](#)

Custom HTTP Headers

+ Add header

```
lab10_notes.txt
1 Admin token: ChangeMe2025!
2 Org ID: f9a32d0c50663666
3 Container ID: 09a55ace7e39
```

!!! You should have 3 measurements found !!!

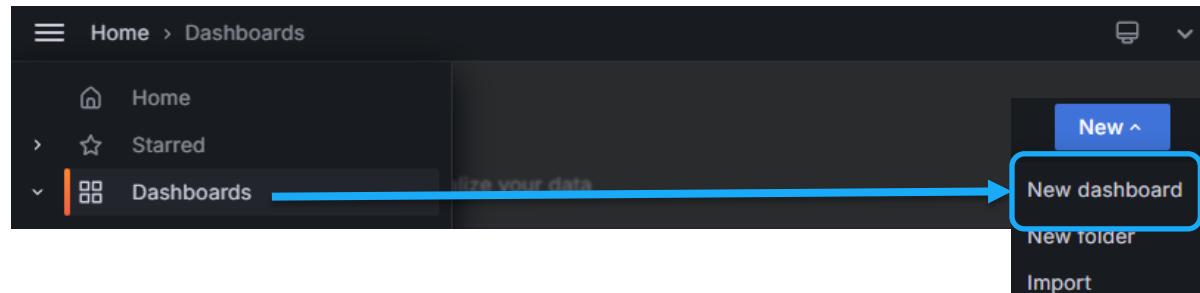
[Save & test](#)

✓ datasource is working. 3 measurements found

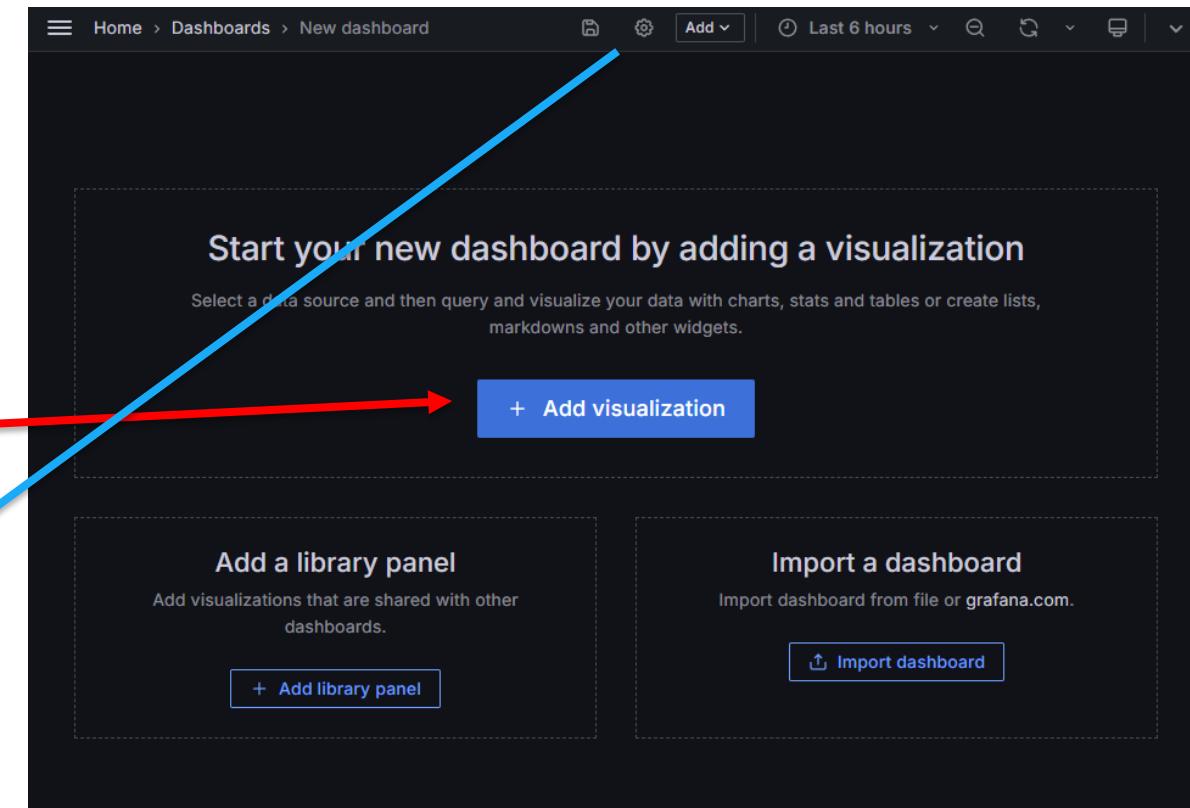
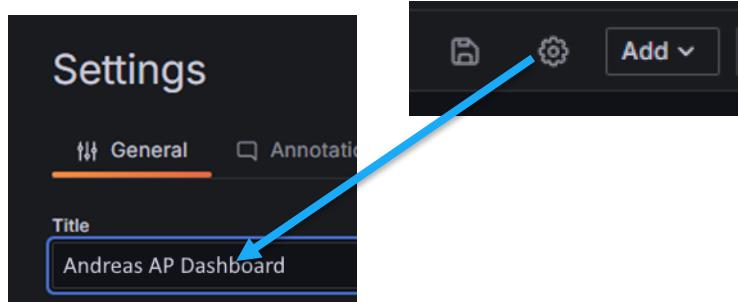
Next, you can start to visualize data by [building a dashboard](#), or by querying data in the [Explore view](#).

# Create your first dashboard

- Go to "Dashboards" in the left menu, then create a new dashboard

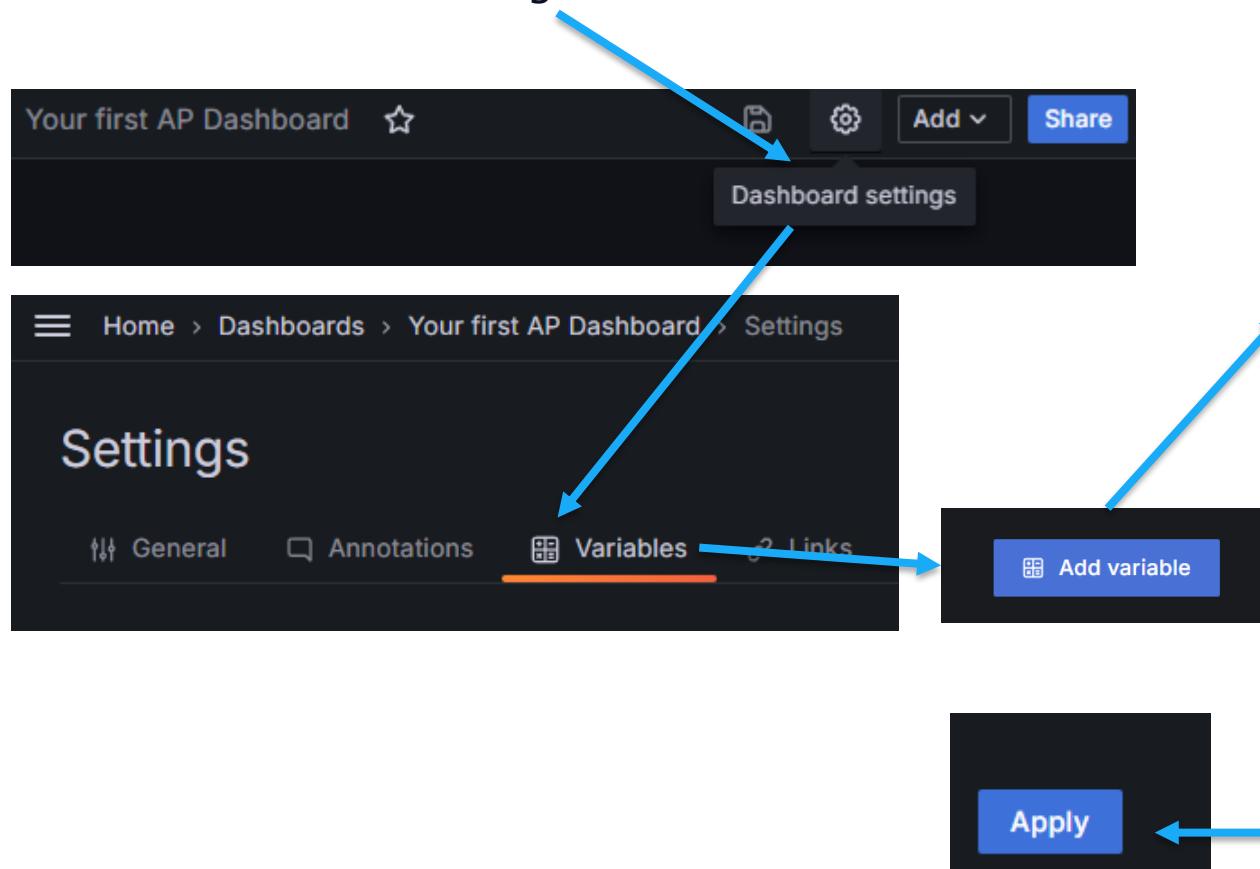


- !!! Don't press the very tempting "+Add Visualization" button yet, we will do some pre-work first 😊
- Start by giving the new dashboard a name



# Grafana

- We will create a variable that gets all AP names
- Go to Dashboard settings -> Variables



- Name: APName
- Query: SHOW TAG VALUES WITH KEY = "wtp\_name"

The dialog is titled "APName". It has a "Select variable type" dropdown set to "Query". Under "General", the "Name" field contains "APName". In "Query options", the "Data source" is set to "influxdb" and the "Query" field contains the text "SHOW TAG VALUES WITH KEY = \"wtp\_name\"". A note on the right says: "When copy-pasting this, make sure to remove the extra line break if it occurs".

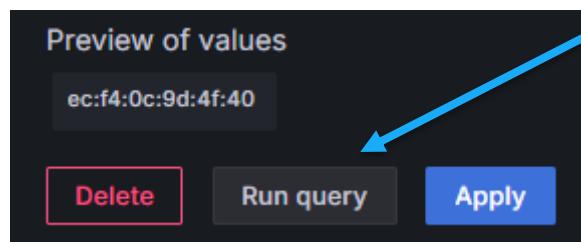
# Grafana

We create a second variable, the AP MAC address

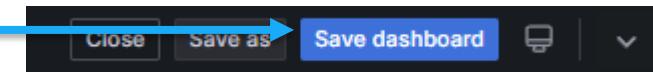
- Name: APMac
- Show on dashboard: Nothing
- Query: (when copy-pasting, make sure to remove the extra line-break if it occurs)

```
SELECT distinct("wtp_mac") FROM "Cisco-IOS-XE-wireless-access-point-oper:access-point-oper-data/ap-name-mac-map" WHERE ("wtp_name" =~ /$APName$/)
```

- Check if it is working with Run query



- Press Apply
- Then save your dashboard



APMac

Select variable type

Query

General

Name: APMac

Label: Label name

Description: Descriptive text

Show on dashboard

Label and value Value Nothing

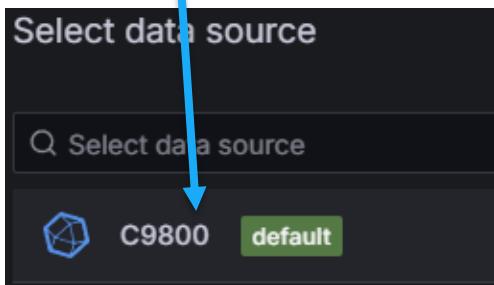
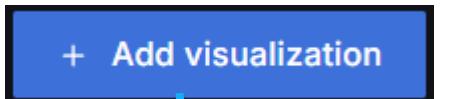
Query options

Data source: influxdb

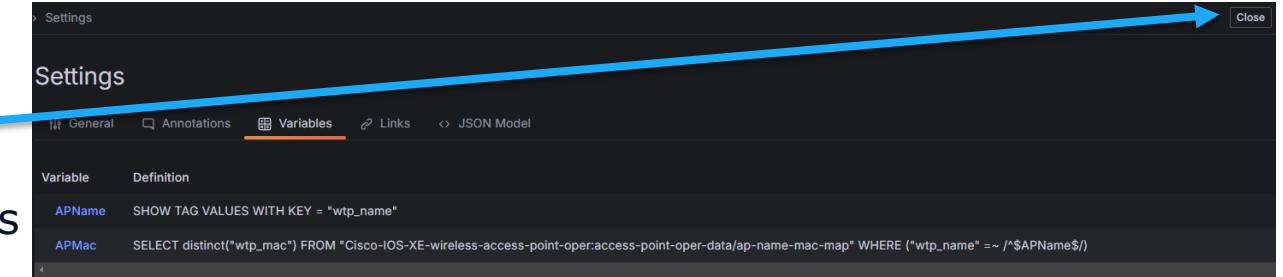
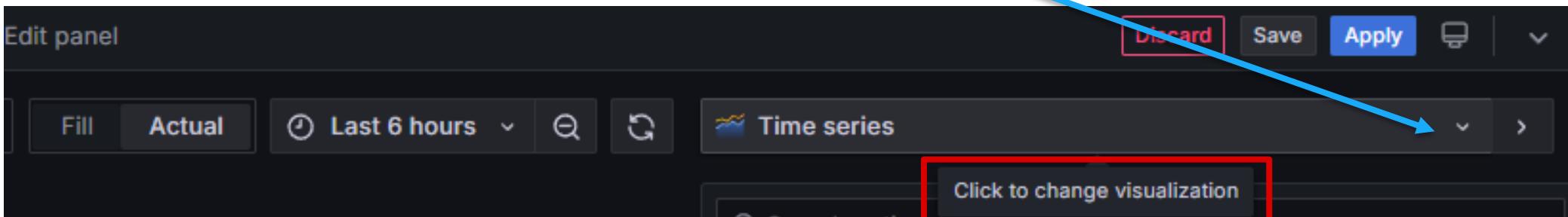
Query: SELECT distinct("wtp\_mac") FROM "Cisco-IOS-XE-wireless-access-point-oper:access-point-oper-data/ap-name-mac-map" WHERE ("wtp\_name" =~ /\$APName\$/)

# Grafana

- Go back to the Dashboard main page
- Add visualization and make one using time series

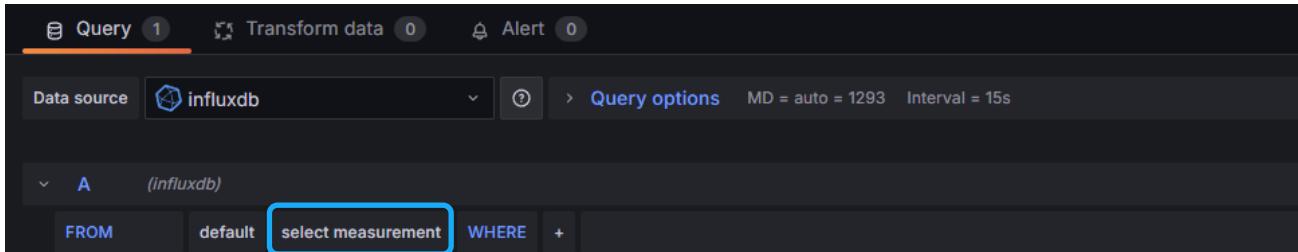


- "Time series" is automatically selected. If not, select it here

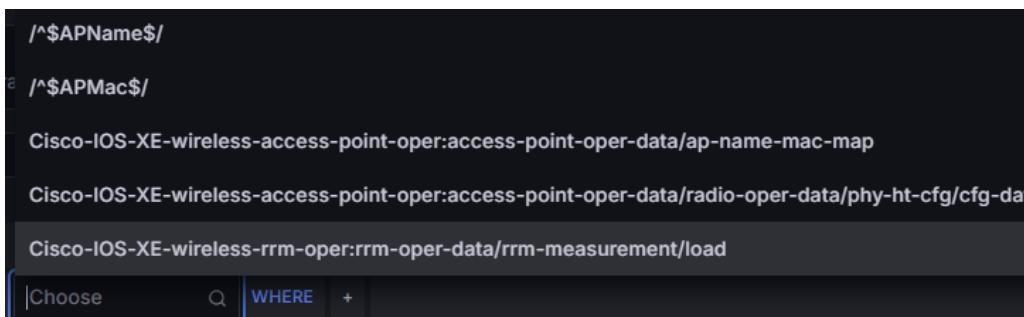


# Grafana

- In the lower part of the screen, edit the query



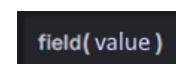
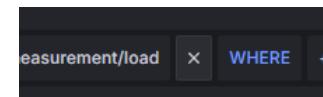
- Click "select measurement", you will get a menu of all measurements



- Choose the *Cisco-IOS-XE-wireless-rrm-oper:rrm-oper-data/rrm-measurement/load*
- (Optional)** You can enter [yangcatalog.org](http://yangcatalog.org), find the Cisco-IOS-XE-wireless-rrm-oper module and explore which values you could get from this YANG model

# Grafana

- To get the options, click the "+" signs, like here
- You can search in the "Choose" field over the options
- To specify the value inside the "field(value)" click it
- Try to make the screen match this image



A (influxdb)

**FROM**

default Cisco-IOS-XE-wireless-rrm-oper:rrm-oper-data/rrm-measurement/load WHERE wtp\_mac::tag =~ /\$APMac\$/ AND radio\_slot\_id::tag = 0

**SELECT**

field(cca\_util\_percentage) last() alias(CCA Util% - 2.4GHz) +  
 field(rx\_util\_percentage) last() alias(Rx Util% - 2.4GHz) +  
 field(tx\_util\_percentage) last() alias(Tx Util% - 2.4GHz) +  
 field(rx\_noise\_channel\_utilization) last() alias(Rx Noise Util% - 2.4GHz) +  
 field(stations) last() alias(Clients - 2.4GHz) +  
 field(non\_wifi\_inter) last() alias(Non-Wi-Fi interference - 2.4GHz) +

**GROUP BY**

time(\$\_\_interval) fill(null) +

**TIMEZONE**

(optional) ORDER BY TIME ascending

**LIMIT**

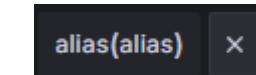
(optional) SLIMIT (optional)

**FORMAT AS**

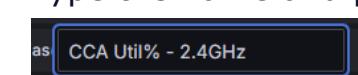
Time series ALIAS \$col



To create an alias, choose "alias" and click the alias(alias) button

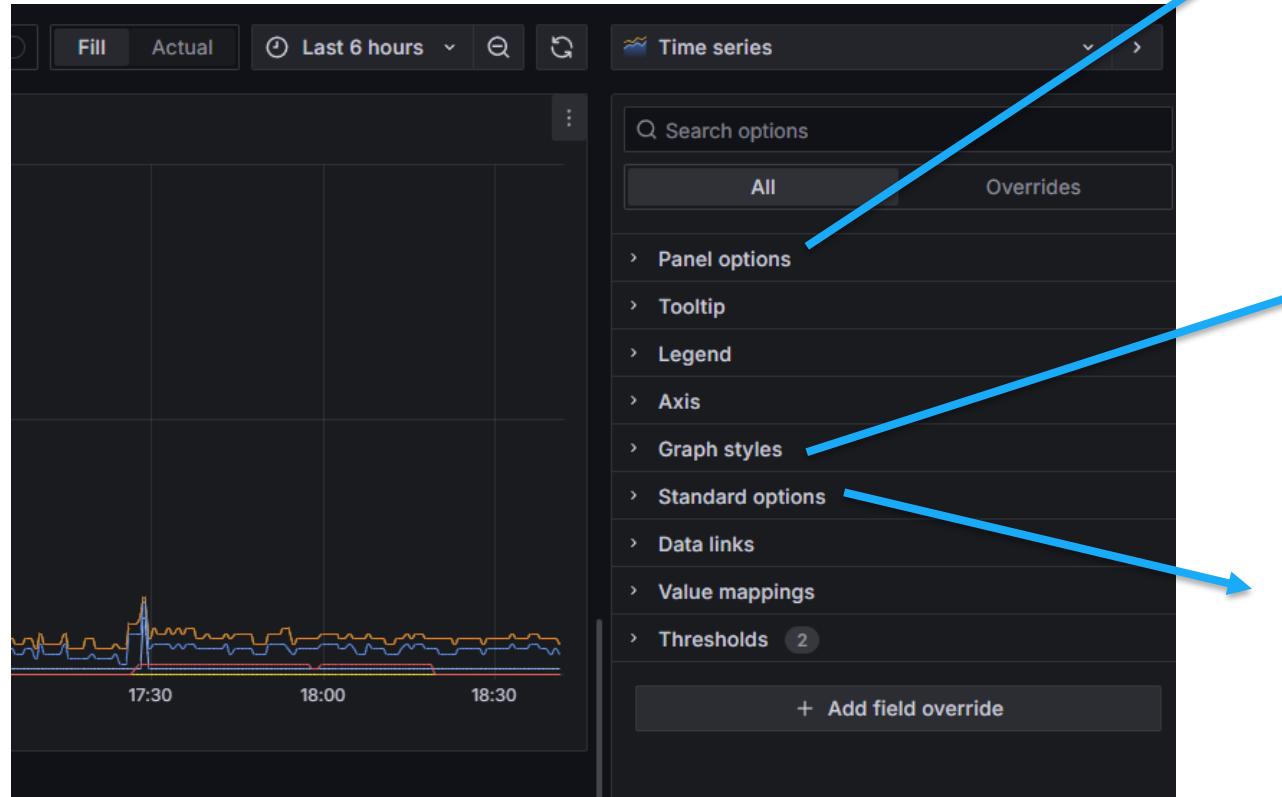


Type the name and press Enter

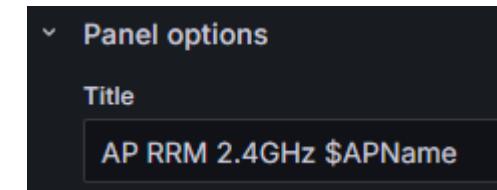


# Grafana

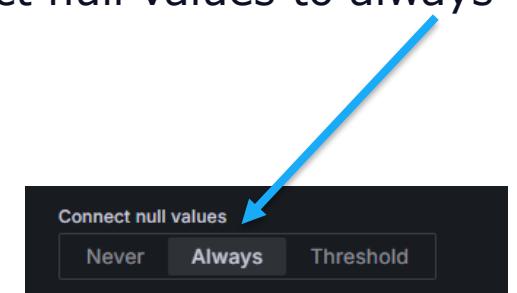
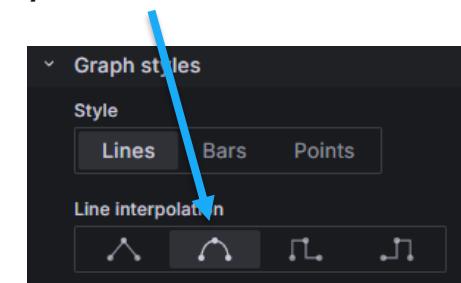
- In the field to the right, you have lots of options



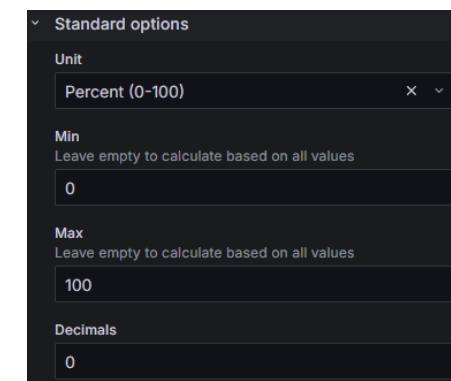
- Set the panel title



- Under graph styles you can make it look like you want ☺ Set Connect null values to always

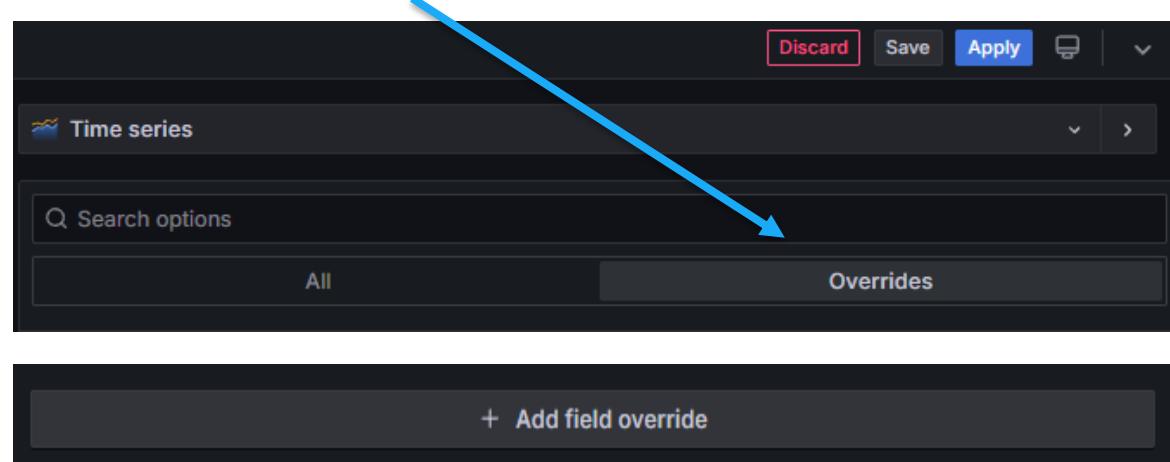


- Fix values to percent

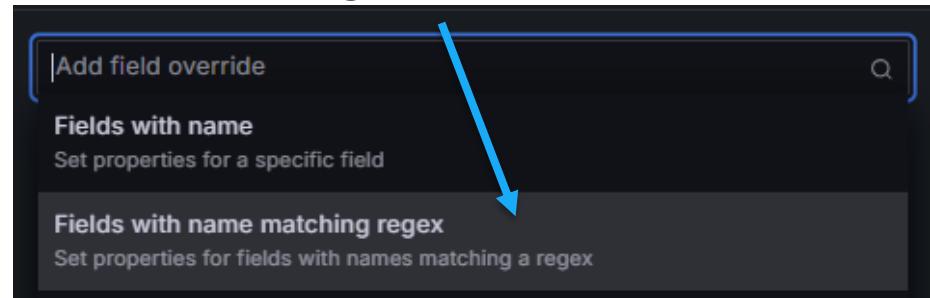


# Grafana

- You do not want number of clients to show as percent
- We use an override for the Clients graph



- Select the regex override



A screenshot of the 'Override 1' configuration panel. It shows a 'Fields with name matching regex' input field containing '.Clients.\*'. There is a small trash icon in the top right corner.

Standard options > Unit

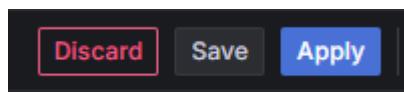
A screenshot of the 'Standard options > Unit' configuration panel. It shows a 'Fields with name matching regex' input field containing '.Clients.\*'. Below it is a search bar with 'un' typed in. A blue arrow points from the text 'Standard options > Unit' down to the 'Standard options > Unit' label.

Standard options > Unit = short

A screenshot of the 'Standard options > Unit' configuration panel. It shows a 'Fields with name matching regex' input field containing '.Clients.\*'. Below it is a 'Standard options > Unit' section with a dropdown menu. The word 'short' is highlighted in blue. A blue arrow points from the text 'Standard options > Unit = short' down to the 'short' selection.

# Grafana

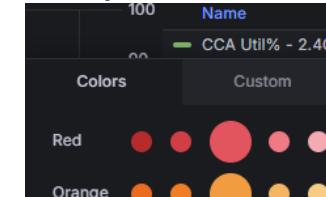
- **(Optional)** Tooltip -> Tooltip mode: All
- **(Optional)** Legend
  - Mode: Table
  - Placement: Right
  - Values: Last\*, Min & Max
- **(Optional)** Thresholds
  - 35% (Red)
  - Percentage
  - As lines (dashed)
- **(Optional)** Graph styles
  - Fill opacity: 10
- Remember to apply changes



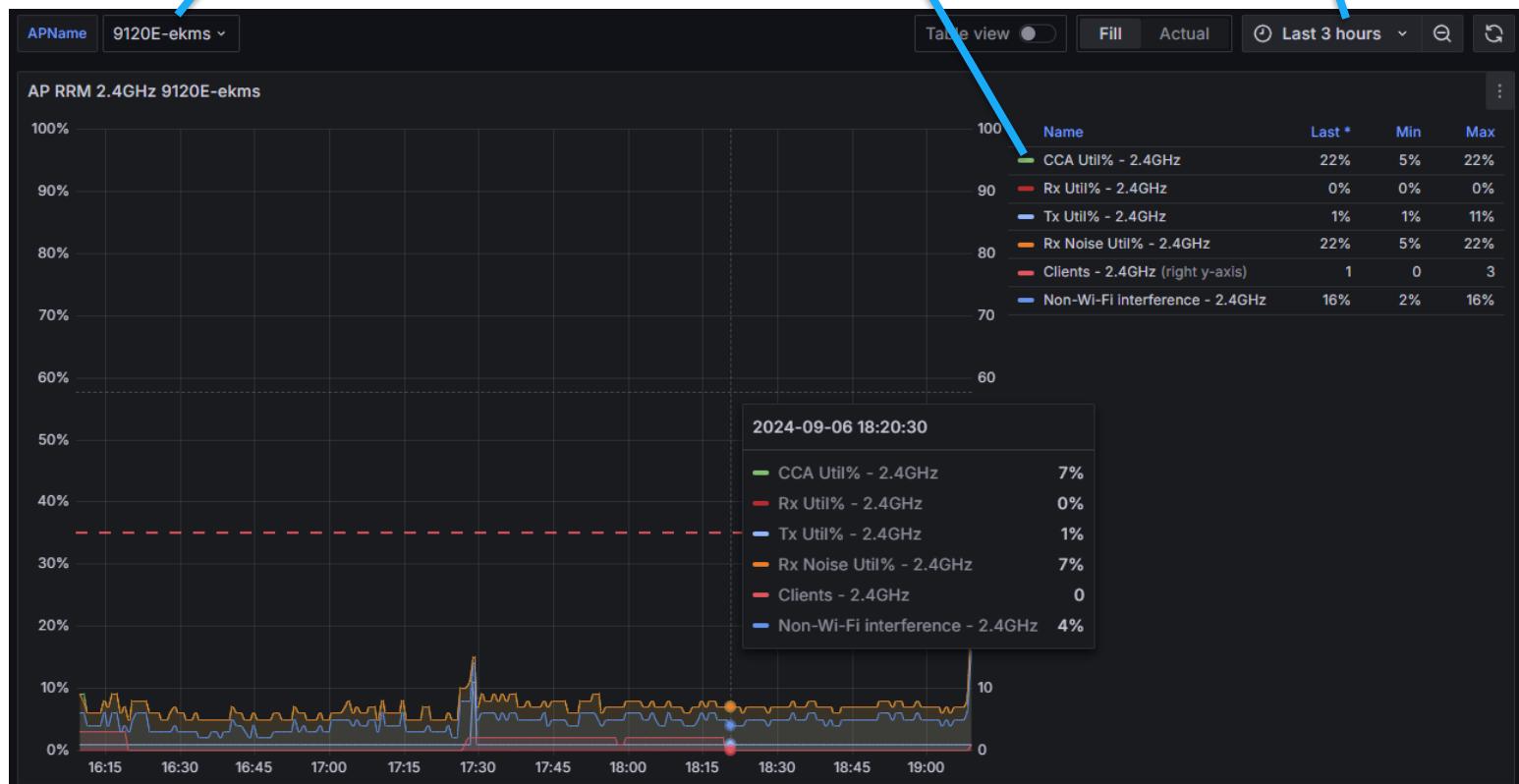
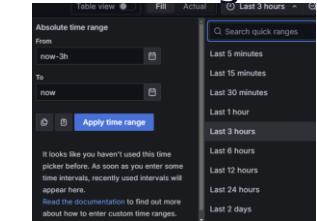
## Select variable

| Enter variable value |
|----------------------|
| 9120E-ekms           |
| APD42C-44D8-2F38     |

## Graph colors



## Time range

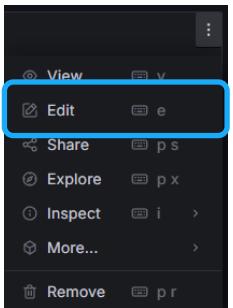


# Duplicate dashboards

- You can duplicate the visualization, then edit the second one to create a 5GHz variant

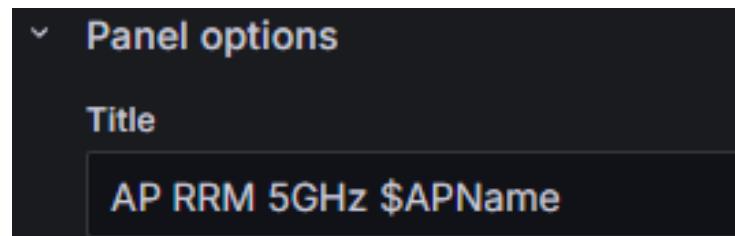


- Edit the duplicated visualization



# Edit the values for the 5GHz visualization

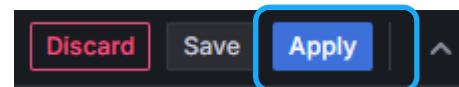
- Panel Options: Title



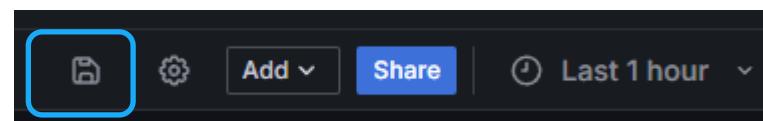
- FROM in Query: radio\_slot\_id::tag = 1
  - If the AP is dual 5GHz (for example 9136, 9166) 5GHz is at 1 and 2



- Apply to go back to the Dashboard

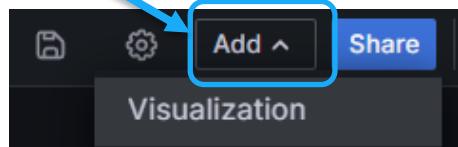


- Save the Dashboard



# Grafana

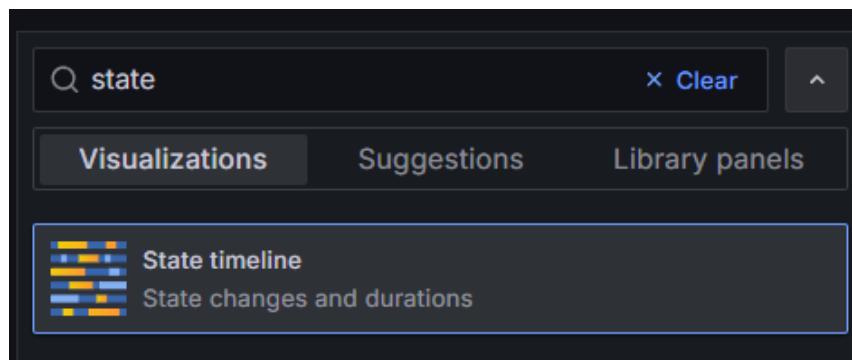
- Add a new visualization to your AP Dashboard



- Press the field where it defaults to "Time series"



- Search for and select "State timeline "



# Grafana

- First we create this visualization for 2.4GHz. To do this we select radio\_slot\_id 0

The screenshot shows the Grafana query editor interface. The top bar displays the title 'A (C9800)'. The query is defined as follows:

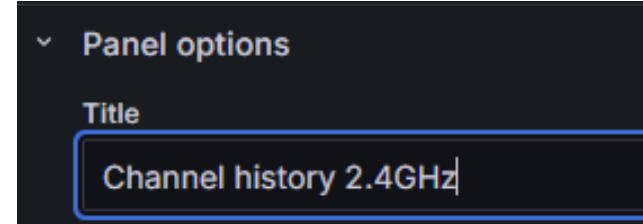
- FROM:** default Cisco-IOS-XE-wireless-access-point-oper:access-point-oper-data/radio-oper-data/phy-ht-cfg/cfg-data
- SELECT:**
  - field(curr\_freq) last() alias(Channel)
  - field(chan\_width) last() alias(Ch Width)
  - field(rrm\_channel\_change\_reason) last() alias(Ch Change Reason)
  - field(freq\_string) last() alias(Bonded channels)
- GROUP BY:** time(\$\_\_interval) fill(null)
- TIMEZONE:** (optional) ORDER BY TIME ascending
- LIMIT:** (optional) SLIMIT (optional)
- FORMAT AS:** Time series ALIAS \$col

- It doesn't look impressive yet:

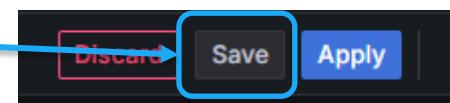
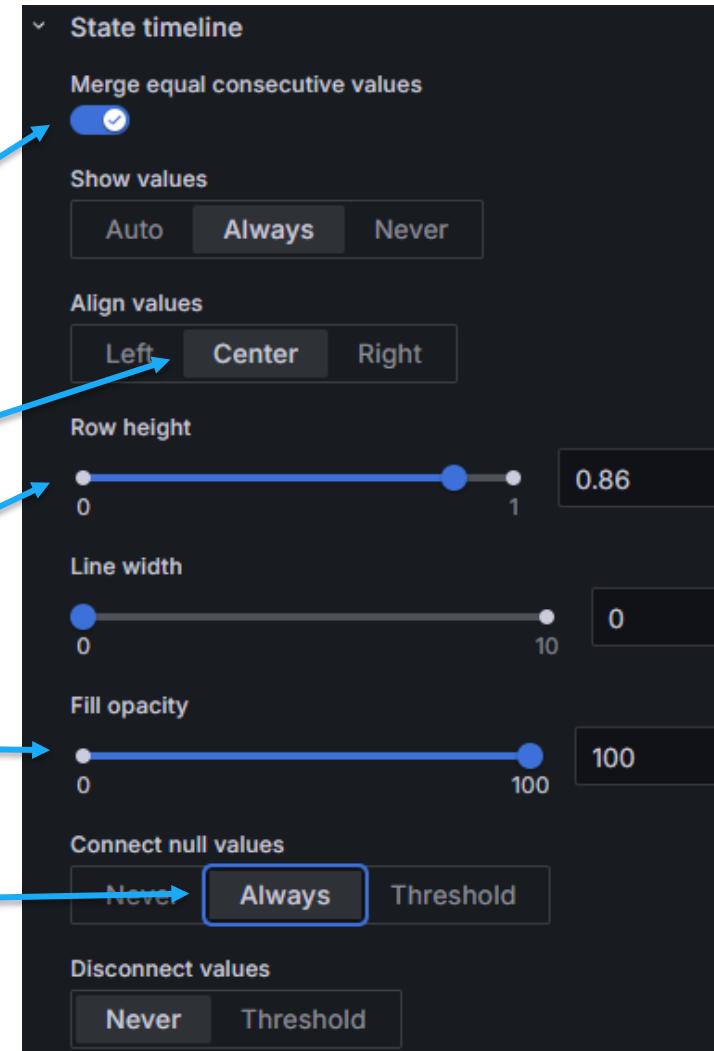


# Grafana

- We will edit the options  
Start by giving it a name

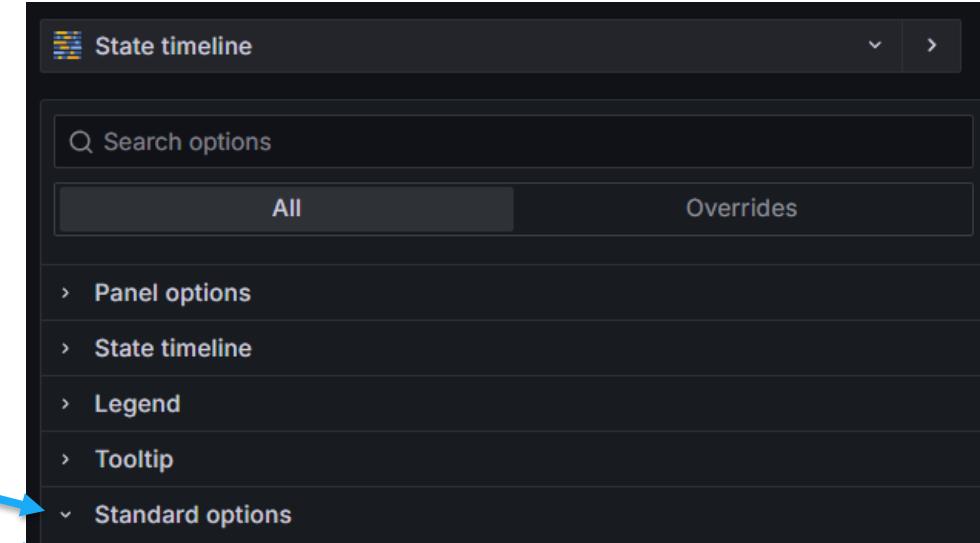
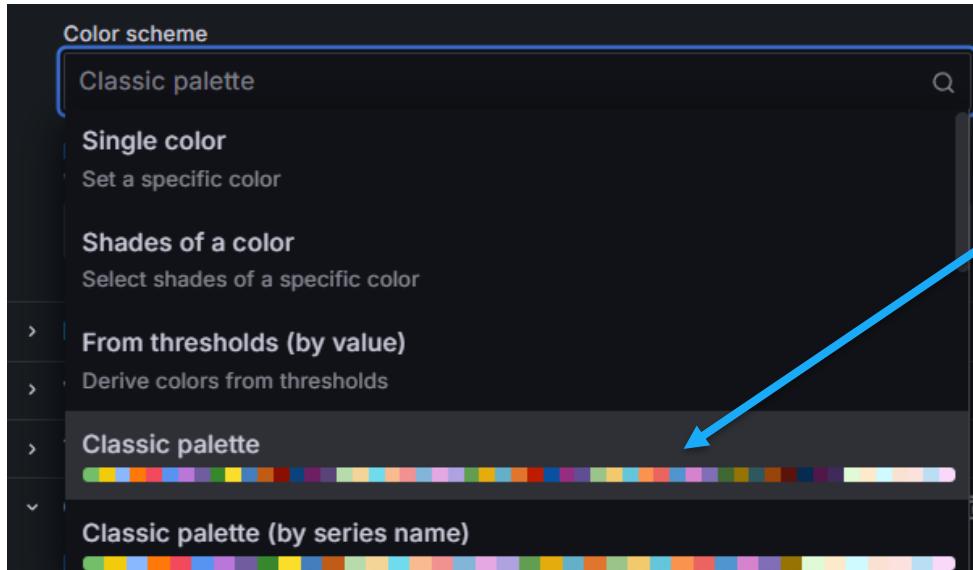


- Merge consecutive values
- Align to text to center
- Change row height and fill opacity
- Conect null values
- Remember to save



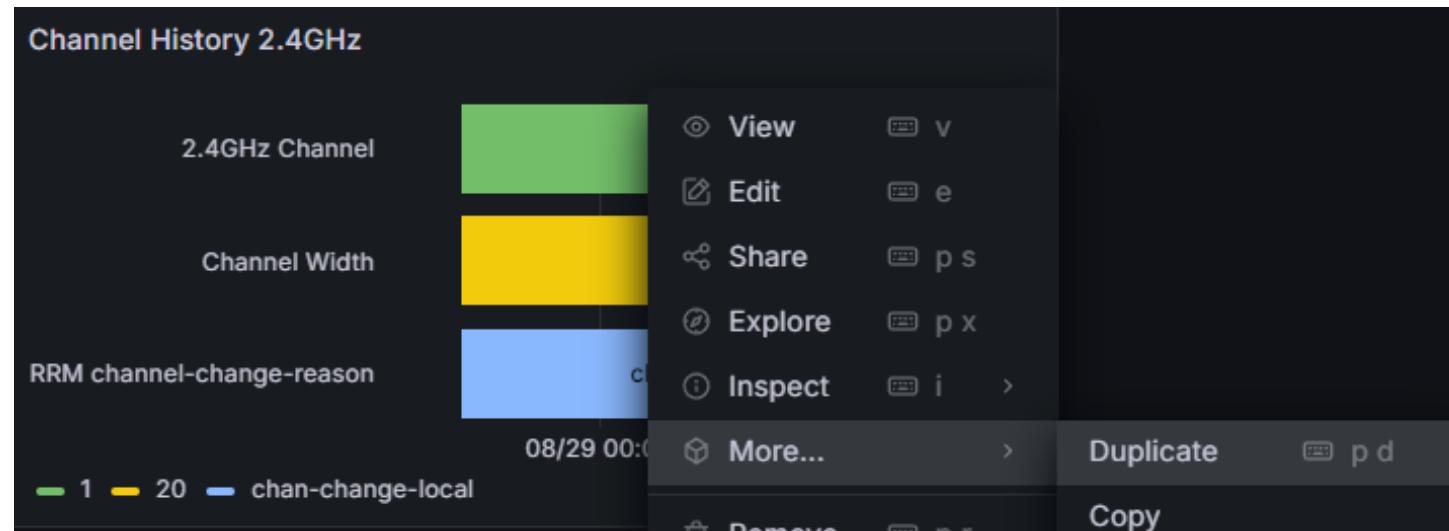
# Grafana

- The current color scheme derive colors from thresholds. The values also look a bit weird. We will change this to a scheme which give each bar a separate fixed color
- Naviagate to Standard Options
  - At the bottom you will find Color scheme
  - Change it to f.ex classic palette

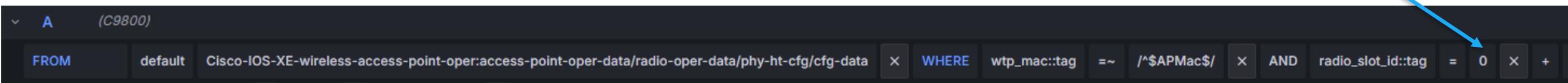


# Grafana

- Apply to get back to your Dashboard
- Duplicate the panel (top right corner and duplicate) and make one for 2.4, 5 and 6GHz (radio-slot)

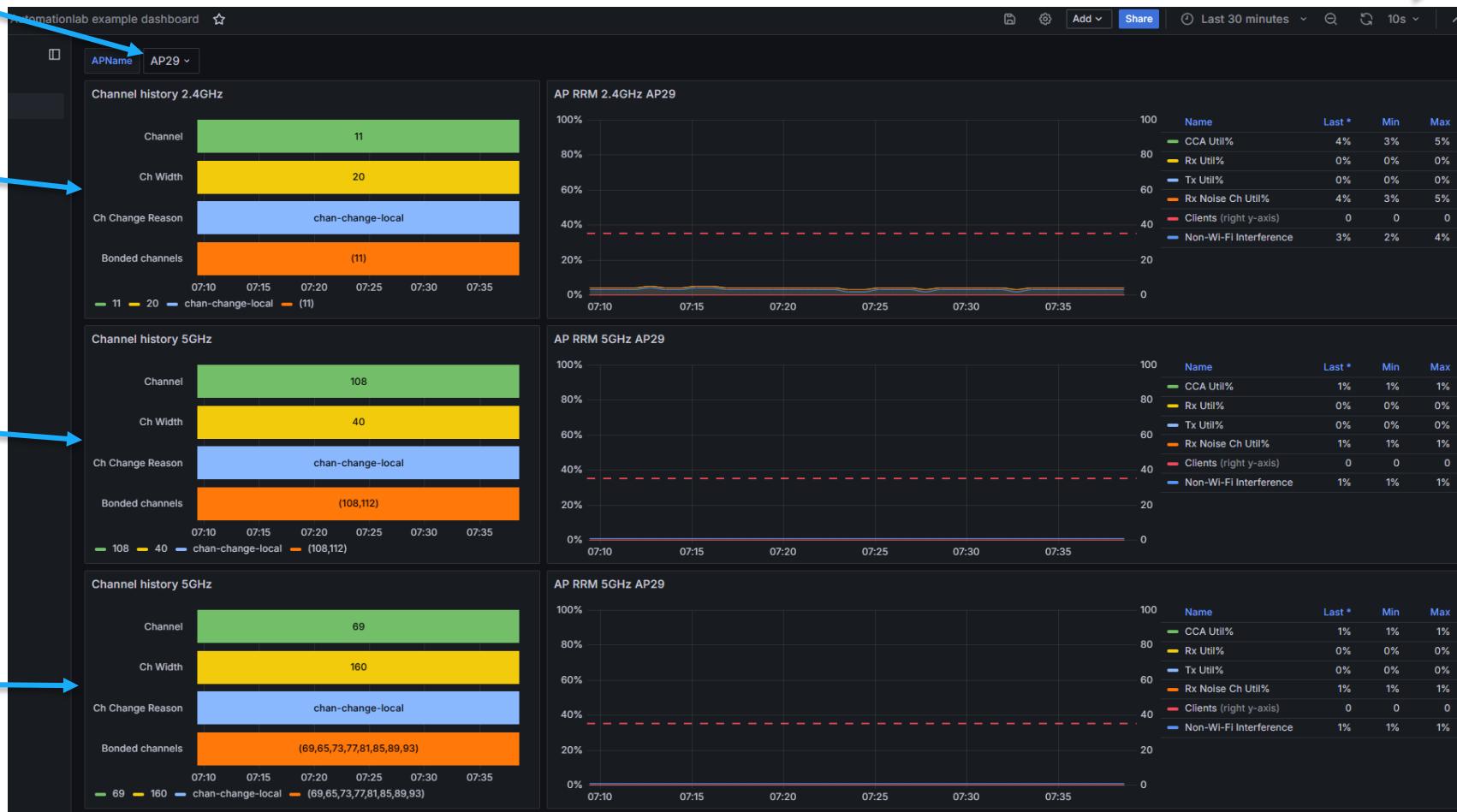


- Slot 0 = 2.4GHz
- Slot 1 = 5GHz
- Slot 2 (if AP is dual 5GHz) = 5GHz
- Slot 3 = 6GHz



# My first AP Dashboard (example)

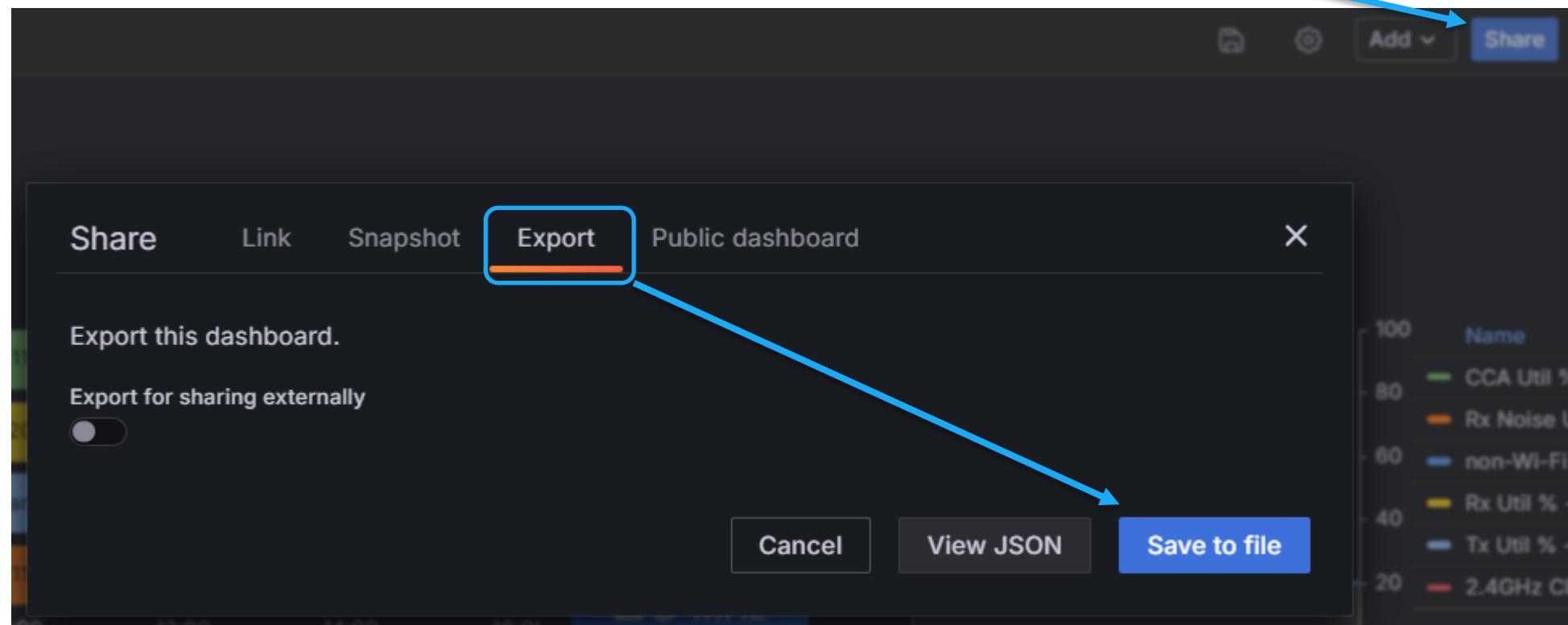
- Select AP



# My first AP Dashboard (example 2)



# Save your dashboard to JSON



grafana-dashboard.json

# Lab exercise #12: Install YANG Suite (optional)

- This exercise shows how to install YANG Suite yourself, as a Docker container
- After successful installation, you can connect to YANG Suite on
  - `https://{{your Ubuntu IP}}:8443`
- Note that there is a binding to the servers IP that you specify as part of the installation script. So if you change IP you will have to reconfigure (or just delete and reinstall)



# YANG Suite installation

- Installation (requires Docker installed)

```
devnet-adm@ubuntu-devnet:~$ cd ~
devnet-adm@ubuntu-devnet:~$ git clone https://github.com/CiscoDevNet/yangsuite
devnet-adm@ubuntu-devnet:~$ cd yangsuite/docker/
devnet-adm@ubuntu-devnet:~/yangsuite/docker$./start_yang_suite.sh
Hello, please setup YANG Suite admin user.
username: devnet-adm
password: ChangeMe2025!
confirm password: ChangeMe2025!
Will you access the system from a remote host? (y/n): y
Enter local host FQDN or IP: 192.168.10.7 ←
Setup test certificates? (y/n): y
#####
Generating self-signed certificates ##
(...)
email: devnet-adm@test.com
Setup test certificates? (y/n): y
(...blah blah blah) ←
```

- !!! Note: This will create problems if you try to use YANG Suite if you have changed the IP of your Ubuntu server to something else. In that case, delete YANG Suite and reinstall !!!
- To delete, use `rm -R ~/yangsuite`

• Change this to your Ubuntu IP

• From here you can just press Enter-Enter-Enter

- (...appx 4-5min depending on your VM speed)
- While waiting, please carry on to the next slides ☺
- Reference: <https://developer.cisco.com/yangsuite/>



# Run in detached mode

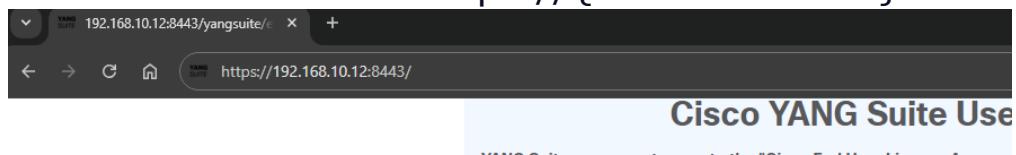
- When you get this output, it is ready to run

```
docker-yangsuite-1 | uWSGI running as root, you can use --uid, --gid, --chroot options
docker-yangsuite-1 | *** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
docker-yangsuite-1 | your server socket listen backlog is limited to 100 connections
docker-yangsuite-1 | your mercy for graceful operations on workers is 60 seconds
docker-yangsuite-1 | mapped 609456 bytes (595 KB) for 5 cores
docker-yangsuite-1 | *** Operational MODE: preforking ***
docker-yangsuite-1 | WSGI app 0 (mountpoint='') ready in 1 seconds on interpreter 0x56119e608280 pid: 27 (default app)
docker-yangsuite-1 | uWSGI running as root, you can use --uid/--gid/--chroot options
docker-yangsuite-1 | *** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
docker-yangsuite-1 | *** uWSGI is running in multiple interpreter mode ***
docker-yangsuite-1 | spawned uWSGI master process (pid: 27)
docker-yangsuite-1 | spawned uWSGI worker 1 (pid: 29, cores: 1)
docker-yangsuite-1 | spawned uWSGI worker 2 (pid: 30, cores: 1)
docker-yangsuite-1 | spawned uWSGI worker 3 (pid: 31, cores: 1)
docker-yangsuite-1 | spawned uWSGI worker 4 (pid: 32, cores: 1)
docker-yangsuite-1 | spawned uWSGI worker 5 (pid: 33, cores: 1)
```

- To "free up" the command line, press Ctrl+C to stop YANG Suite
- Then, to start the container in "detached" mode:

```
devnet-adm@ubuntu-devnet:~$ cd ~/yangsuite/docker/
devnet-adm@ubuntu-devnet:~/yangsuite/docker/$ docker compose up -d
```

- Access via HTTPS: <https://192.168.10.12:8443>



# YANG Suite EULA

## Cisco YANG Suite User Agreement

YANG Suite users must agree to the "Cisco End User License Agreement" and "Privacy Statement".

Choose to accept or decline "Cisco End User License Agreement":

Decline  Accept

Choose to accept or decline "Cisco Online Privacy Statement":

Decline  Accept

**Submit**

## Log in to YANG Suite

Please login to access this page.

Username: devnet-adm

Password: ChangeMe2025!

**Login**

[Lost your password?](#)



# Create new device

- Create new device

The screenshot shows the Cisco YANG Suite interface. On the left, a sidebar menu lists several options: Admin, Setup (highlighted with a blue border), YANG files and repositories, YANG module sets, Device profiles (highlighted with a blue border), Analytics, Explore, Protocols, and Help. The main content area is titled "Manage device profiles". It features a "Select a device" section with the message "No devices defined -". Below this are several buttons: "Create new device" (highlighted with a blue border), "Check selected device's reachability", "Clone selected device", "Edit selected device", "Delete selected device" (highlighted with a red border), and "Create default Repository and Yangset". The top right corner of the main area has a small circular icon with a gear and the text "YANG Suite / Device profiles".



# Create new device

- Create new device
- Use your WLC IP

**New Device Profile**

Fields marked with \* are required.

**General Info**

|                |                                                                              |               |       |
|----------------|------------------------------------------------------------------------------|---------------|-------|
| Profile Name * | C9800-CL 17.15 {or another name you like}                                    |               |       |
| Description    |                                                                              |               |       |
| Address *      | 192.168.10.9 {change to your WLC IP}                                         |               |       |
| Username       | devnet-adm                                                                   |               |       |
| Password       | *****                                                                        |               |       |
| Timeout *      | 30                                                                           |               |       |
| Variables      | <table border="1"> <tr> <td>Variable Name</td> <td>Value</td> </tr> </table> | Variable Name | Value |
| Variable Name  | Value                                                                        |               |       |

**gNMI**

Device supports gNMI

**NETCONF**

Device supports NETCONF

Device Variant \*

NETCONF port \* 830

Skip SSH key validation for this device

Address 192.168.10.9 {change to your WLC IP}

Username devnet-adm

Password \*\*\*\*\*

Timeout 30

**RESTCONF**

Device supports RESTCONF

HTTP or HTTP(secure) encoding https

RESTCONF base URL /restconf

RESTCONF port \* 443

Address 192.168.10.9 {change to your WLC IP}

Username devnet-adm

Password \*\*\*\*\*

**SSH**

Device allows SSH login

Device variant \* generic\_termserver

Address 192.168.10.9 {change to your WLC IP}

SSH Port \* 22

Delay Factor 1.0

Username devnet-adm

Password \*\*\*\*\*

Timeout 30

Use SSL Certificate

**Buttons:** Create Profile, Check Connectivity, Cancel

**Connectivity check results:**

- ping
- NETCONF
- RESTCONF
- SSH



# Create repository and Yangset

- Create default repository and Yangset

The screenshot shows the YANG Suite interface with the following elements:

- Top navigation bar: YANG Suite / Device profiles
- Left sidebar menu:
  - Create new device
  - Check selected device's reachability
  - Clone selected device
  - Edit selected device
  - Delete selected device
  - Create default Repository and Yangset
- Main content area: Select a device profile (radio button selected: 9800-vm)
- Bottom status bar: Creating Repository and Yangset for 9800-vm

- (this will take a couple of minutes, so just be patient)

