# Build your own Wi-Fi automation lab

## Lab Guide - Day 1

Andreas Koksrud / Telenor Bedrift

Kjetil Teigen Hansen / Conscia



telenor

# 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





# Prerequisites

- Cisco.com account with access to downloading WLC (9800-CL at https://software.cisco.com)
- Postman account (https://postman.com)
- Windows laptop with administrative privileges
- Complete the pre-lab exercises before the deep dive labs
  - Pre-lab exercises are distributed in a separate PowerPoint document
    `Build your own Wi-Fi automation lab - Pre-lab tasks (2024-xx-xx).pptx`

- Without these, parts of the Deep Dive might be difficult to complete, or steps might differ severely



# Communications

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

# Agenda

## Pre-lab

- Install VirtualBox
- Install Cisco 9800-CL
- Install Ubuntu Server w/Docker
- Install Postman
- Install VS Code

## Day 1

- Sort out pre-lab task problems
- Get to know the lab environment
- Configure your AP
- 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
  - Other vendors (MIST/Meraki/Aruba)

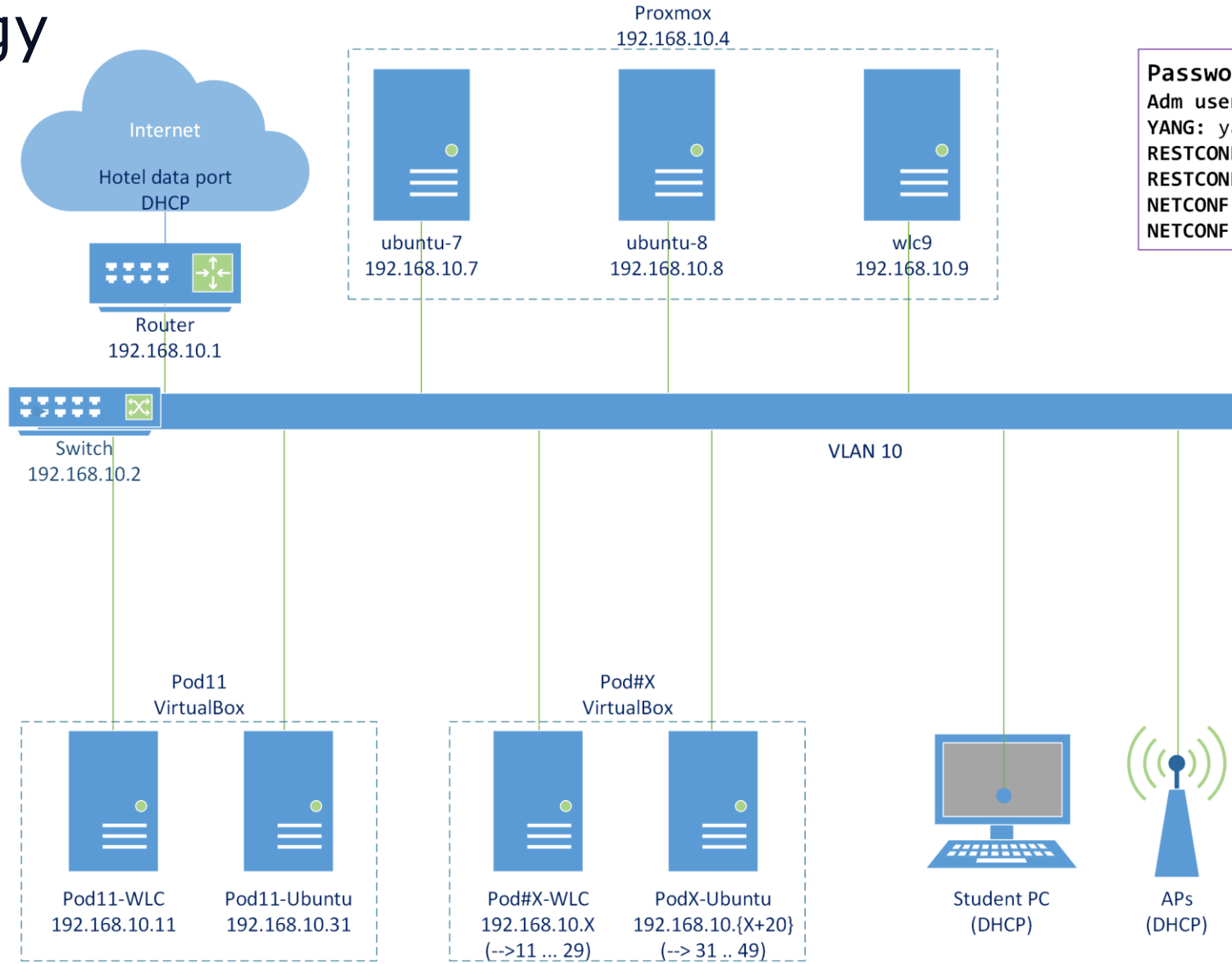


# Scope

- In scope
  - Getting started with various systems/languages/solutions for lab purposes
  - Get a lab environment up and running on your own 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



# Topology




**Passwords to common devices**

**Adm user:** devnet-adm / ChangeMe2024!
**YANG:** yang / yangyang
**RESTCONF (RW):** restconf-adm / restconf-pass
**RESTCONF (RO):** restconf-read / restconf-read
**NETCONF (RW):** netconf-adm / netconf-pass
**NETCONF (RO):** netconf-read / netconf-read

**IP plan**

| Range | Usage |
|---|---|
| 192.168.10.1-10 | Static adm IPs |
| 192.168.10.11-30 | Pod#X WLC |
| 192.168.10.31-50 | Pod#X Ubuntu |
| 192.168.10.51-70 | Reserved static |
| 192.168.10.71-250 | DHCP range |

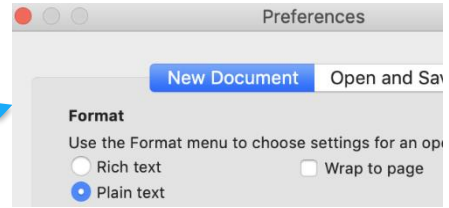
**POD addresses**

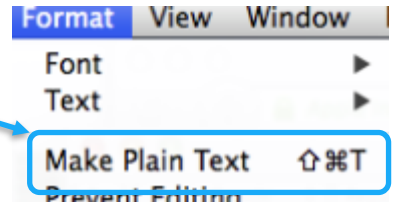
| Pod# | WLC IP | Ubuntu IP |
|---|---|---|
| 11 | 192.168.10.11 | 192.168.10.31 |
| 12 | 192.168.10.12 | 192.168.10.32 |
| 13 | 192.168.10.13 | 192.168.10.33 |
| 14 | 192.168.10.14 | 192.168.10.34 |

(... WLC IP = Pod# + 20 for all...)

# 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



# I only want the "automation" part, not the "building your own lab" part

- If you for some reason do NOT want to do the "building your own automation lab" part of this deep dive, only the "a taste from the automation buffet" part, you can skip some parts. We will try to note those parts at the intro slide to each part.
- In short, you can make these decisions to do less building and more automation
  - Use an Ubuntu server that we can host on a server that we bring along. This will be "your own" and we will delete it after the deep dive. The resources will be shared along with some other participants and the shared WLC
  - Use a C9800 WLC on a server that we bring along. This WLC will be shared by all students that need this. We might be able to host some more WLCs if neccessary
  - Instead of installing YANG Suite you can use it from a shared Ubuntu Server where it is already installed
  - Instead of installing the TIG stack (Grafana etc) you can use it from a shared Ubuntu Server where it is already installed
- Please note that by doing this you might have more time for automation during the deep dive, but you will be less familiar with building the items yourself when you get home or need it in some other situation.



# Tentative schedule for day 1

- Hour 1
  - Finish the pre-lab tasks
  - Get to know the lab environment
  - Configure your AP
  - Install and explore Ansible
- Hour 2
  - Explore Python automation
  - Install (optional) and explore YANG Suite
  - 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: Get to know the lab environment

- Connect your laptop to one of the ports in the switch
  - Ensure you get DHCP in the range 192.168.10.71-250 /24
  - Ensure you get internet access through the wired connection
- Refer to the lab topology, and try the following
  - SSH to the shared Ubuntu servers (192.168.10.7 and .8)
  - SSH and HTTPS to the shared WLC (192.168.10.9)
  - SSH to the switch (192.168.10.2)
  - You have admin access to all shared devices, please don't mess up for each other ☺
- Power up your VMs and ensure their access
  - Your Ubuntu should have access to your WLC
  - Both your Ubuntu and WLC should have internet access, and access to the shared servers
  - shared servers should have access to your servers



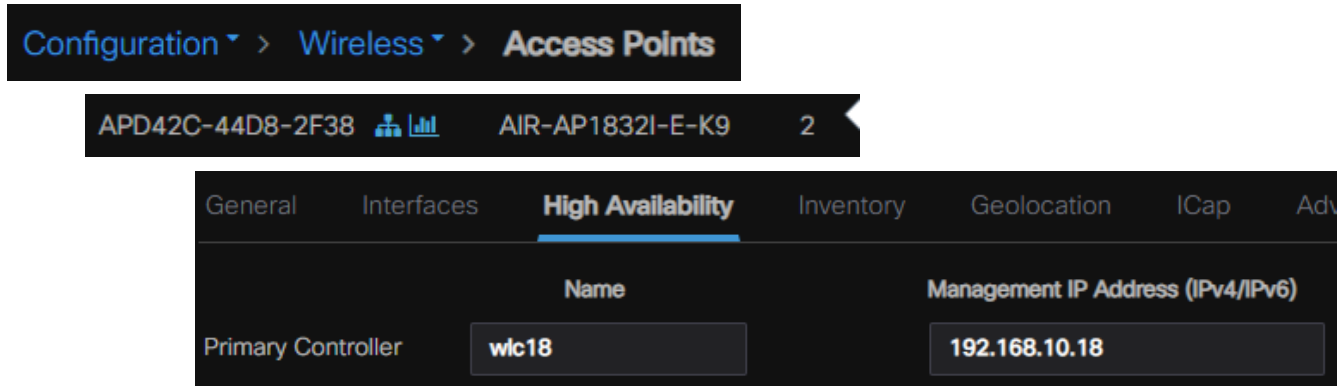
# Lab exercise #2: Configure the AP

- Get it connected to your WLC, or the shared WLC
- Method #1: Connect to AP using console cable, set primary WLC

```
APD42C-44D8-2F38#capwap ap primary-base wlc{pod-number} 192.168.10.{pod-number}
```

  - Example, for Pod#18 this would be: `capwap ap primary-base wlc18 192.168.10.18`

- Method #2: If your AP have joined another WLC, configure your WLC as primary controller




- Ensure the AP shows up on your WLC, else troubleshoot... There should be Cisco experts somewhere in the room if needed ☺

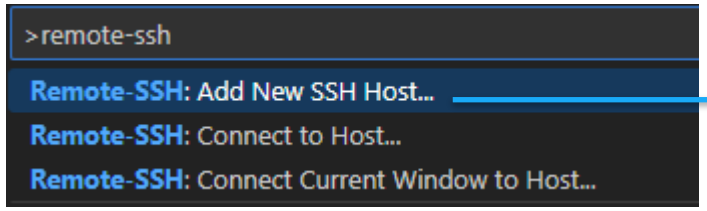
# Lab exercise #3: Explore VS Code

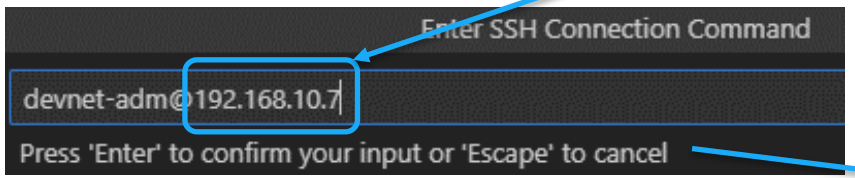
- We will now use our VS Code installation to open a directory on the Ubuntu server
- Instructions will follow to
  - Open the "Remote-SSH" extension
  - Create new host
  - Connect current window to remote host
  - Install local extensions in remote host
  - Open a remote folder as workspace
  - Open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



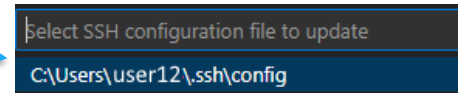
# VS Code - Connect to... Remote SSH

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

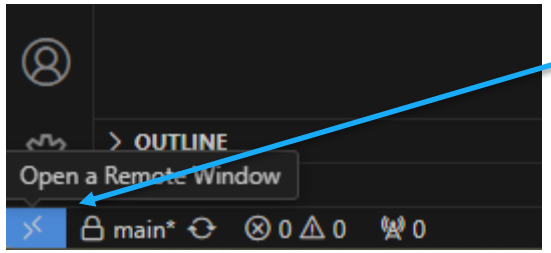


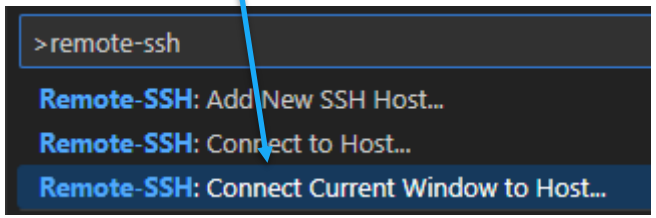




- Replace with your Ubuntu IP

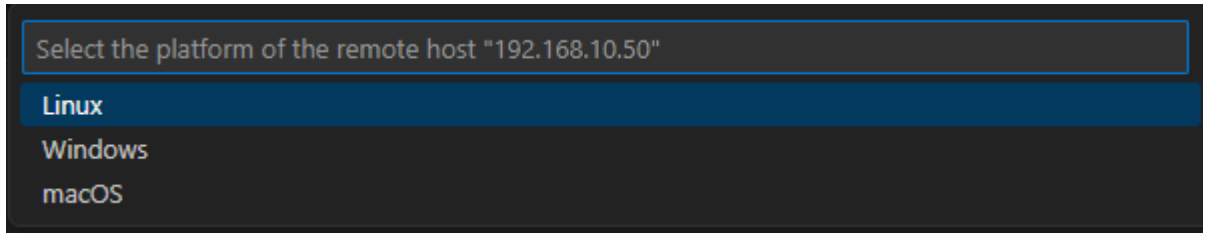



- Connect to the new Host, either using lower left corner of VS Code window or Command Palette (F1)



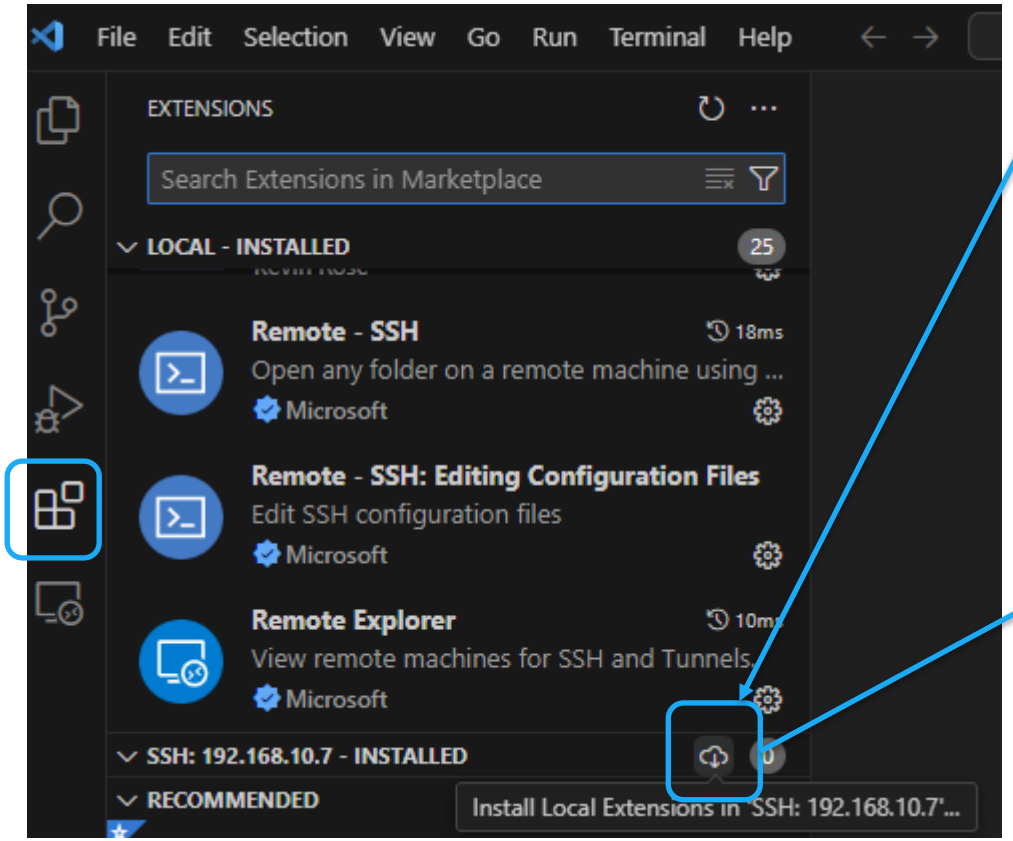




- When asked about platform type, select Linux

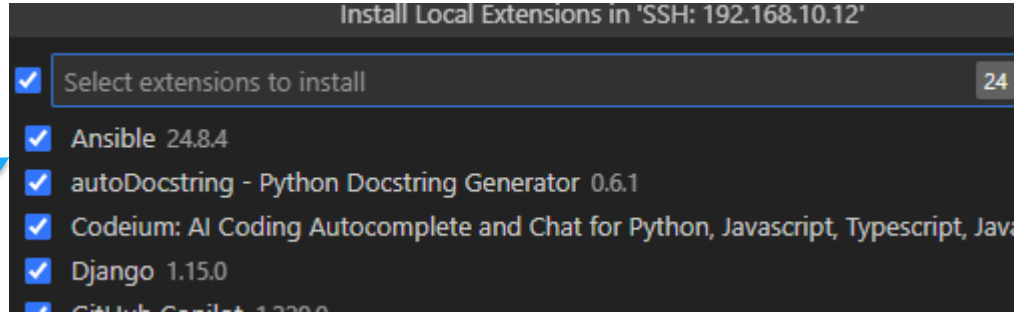


# VS Code - Install Extensions in Remote SSH

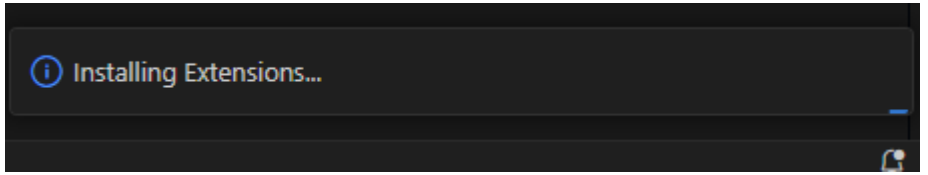
- The extensions have to be installed in each remote environment




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

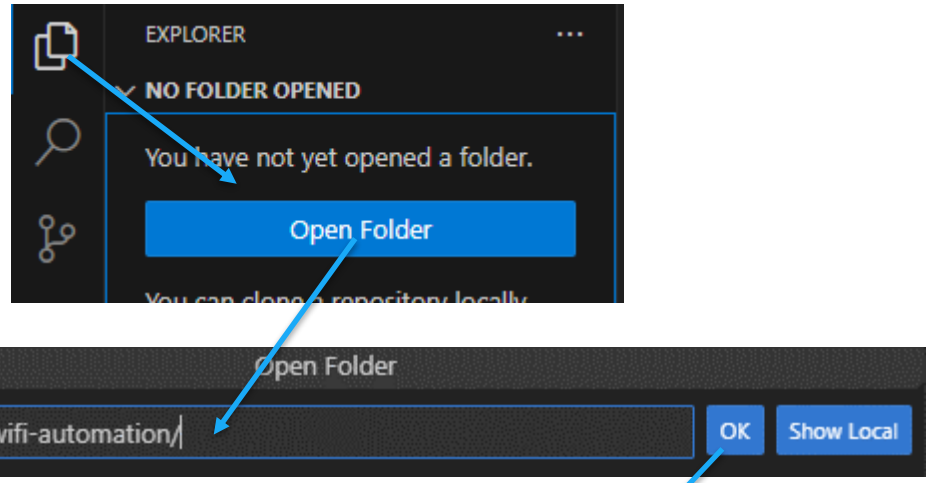



It will take a minute or so to install the extensions in the server



# VS Code

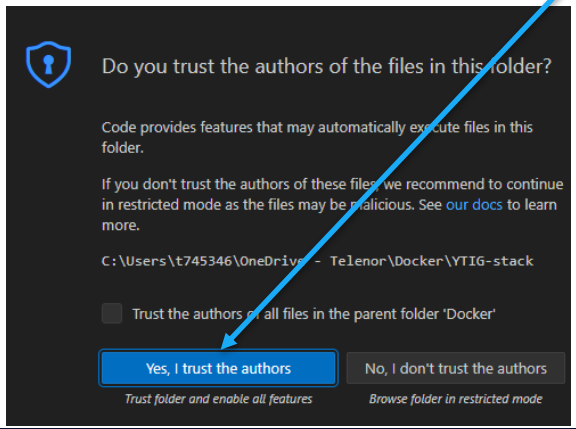
- Open Folder



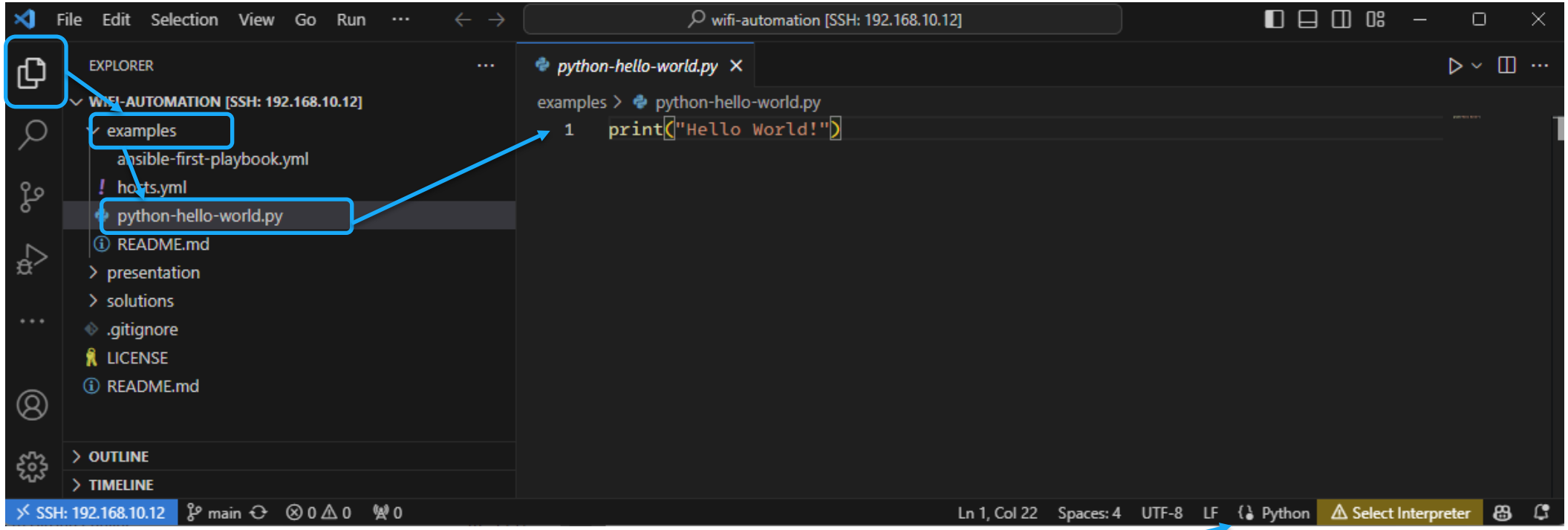




- Trust authors

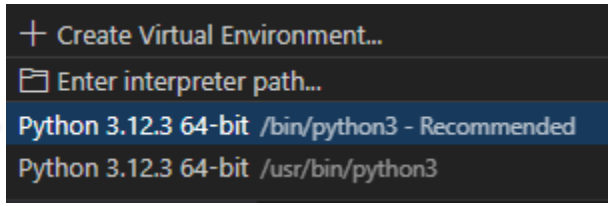


- Using Explorer in VS Code, open the example file `python-hello-world.py`

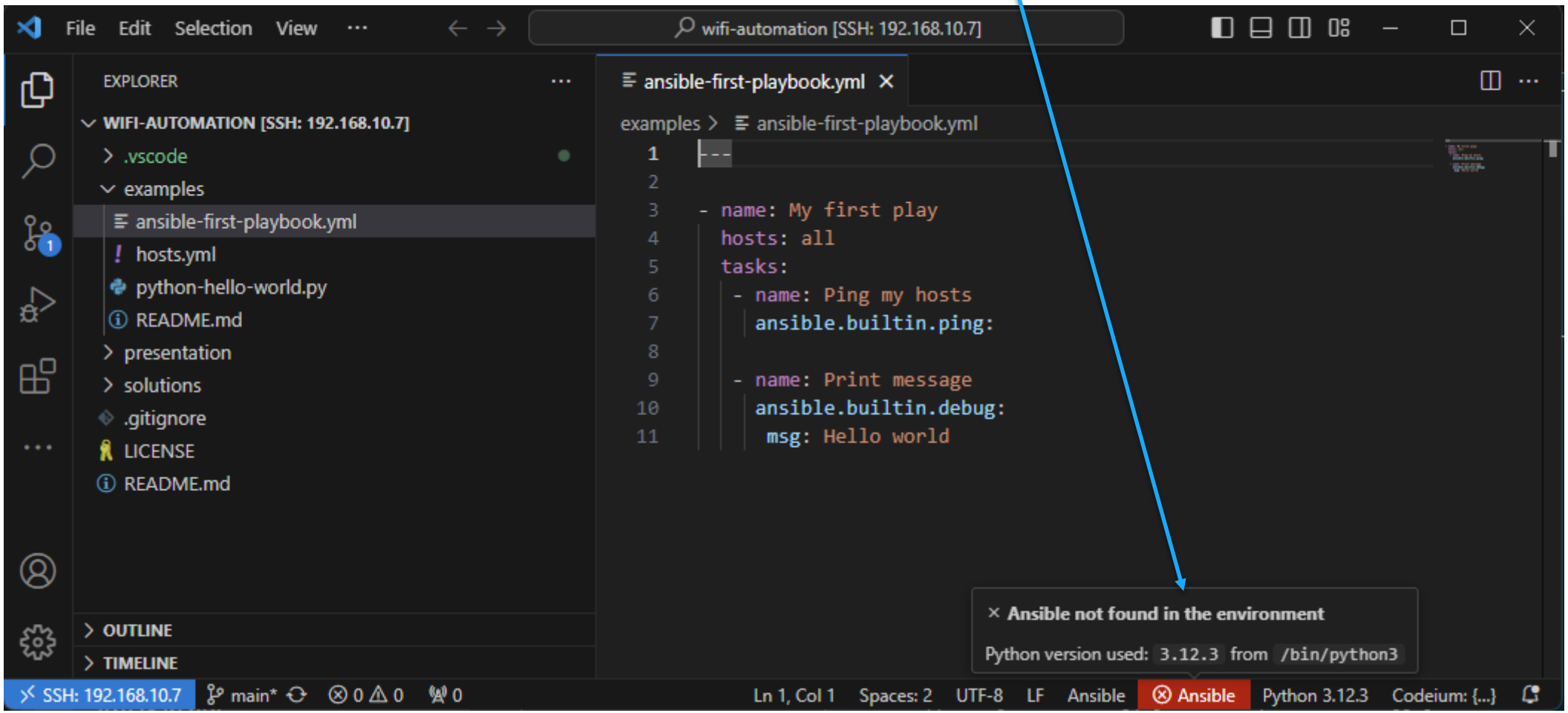



Detected language for active file
(can be manually overridden)

Active Python environment
for active file. Press here, and
for now select one of the options,
like /bin/python3



- 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



# Lab exercise #4: Install Ansible

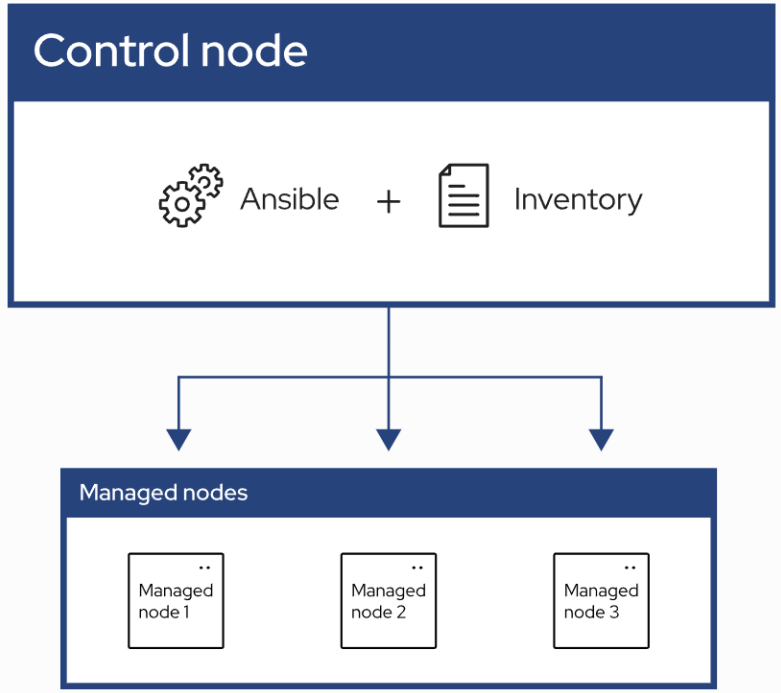
- In this exercise, we will
  - Create a Python virtual environment to use for Ansible stuff
  - Install Ansible on the Ubuntu Server
  - Configure VS Code to use Ansible on the remote server
- The first couple of slides will explain briefly about Ansible (for your reference)
- Then the installation 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 ~/ansible-venv
devnet-adm@ubuntu-devnet:~$ python3 -m venv ~/ansible-venv/
devnet-adm@ubuntu-devnet:~$ source ~/ansible-venv/bin/activate
(ansible-venv) devnet-adm@ubuntu-devnet:~$ pip install ansible ansible-lint
```

```
(ansible-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/ansible-venv/lib/python3.12/site-packages/ansible
  ansible collection location = /home/devnet-adm/.ansible/collections:/usr/share/ansible/collections
  executable location = /home/devnet-adm/ansible-venv/bin/ansible
  python version = 3.12.3 (main, Apr 10 2024, 05:33:47) [GCC 13.2.0] (/home/devnet-adm/ansible-venv/bin/python3)
  jinja version = 3.1.4
  libyaml = True
(ansible-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 is 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

- Installing in a Python VENV (Virtual Environment)
  - 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

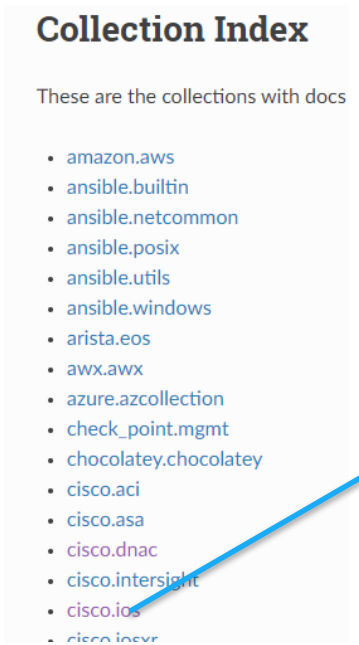
```
(ansible-venv) devnet-admin@ubuntu-devnet:~$ ansible-galaxy collection list

# /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
arista.eos                                9.0.0
awx.awx                                   24.5.0
```

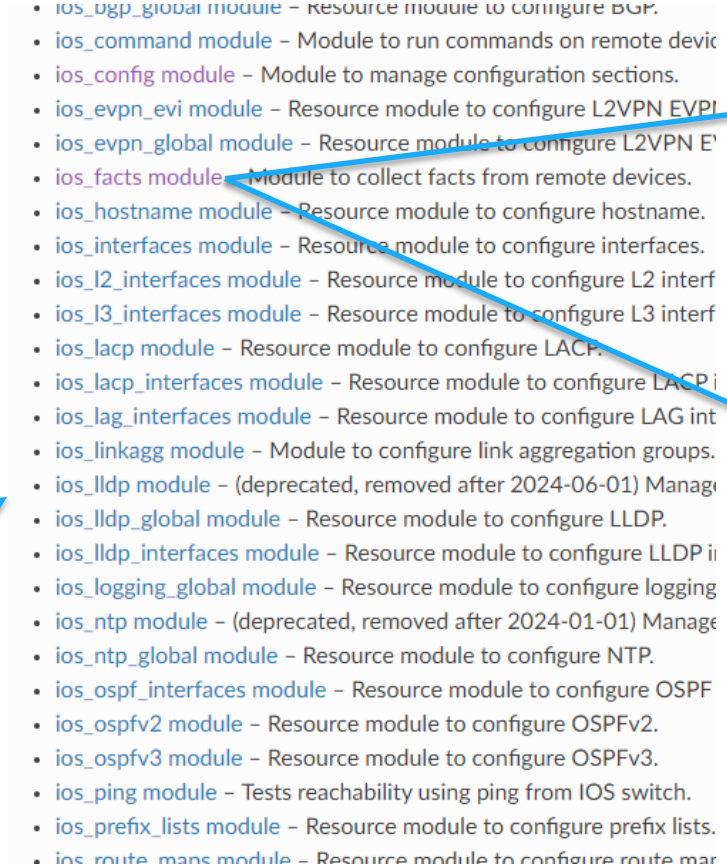
```
(ansible-venv) devnet-admin@ubuntu-devnet:~$ ansible-galaxy collection list | grep "cisco"
cisco.aci                                 2.9.0
cisco.asa                                 5.0.1
cisco.dnac                                6.16.0
cisco.intersight                          2.0.9
cisco.ios                                 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




## 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
```

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

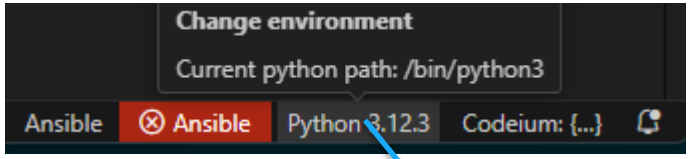


# VS Code - Python environment for Ansible

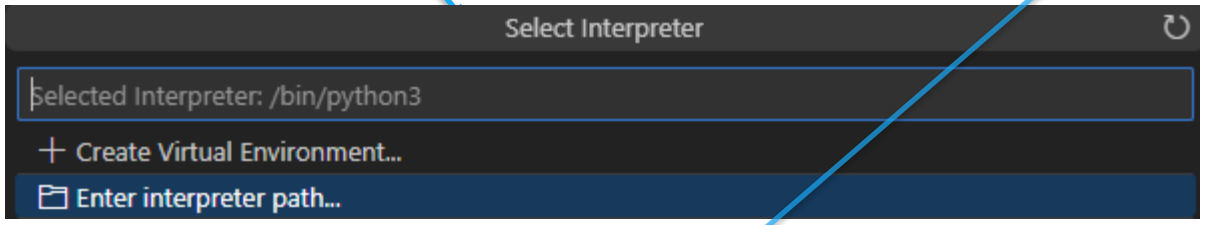
- Check which Python environment is used by Ansible

```
devnet-adm@ubuntu-devnet:~$ source ~/ansible-venv/bin/activate
(ansible-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/ansible-venv/bin/python3)
```

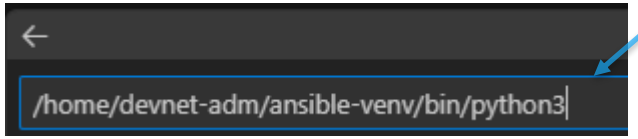
- Select python environment (VS Code, bottom line)




- Select «Enter interpreter path...»




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

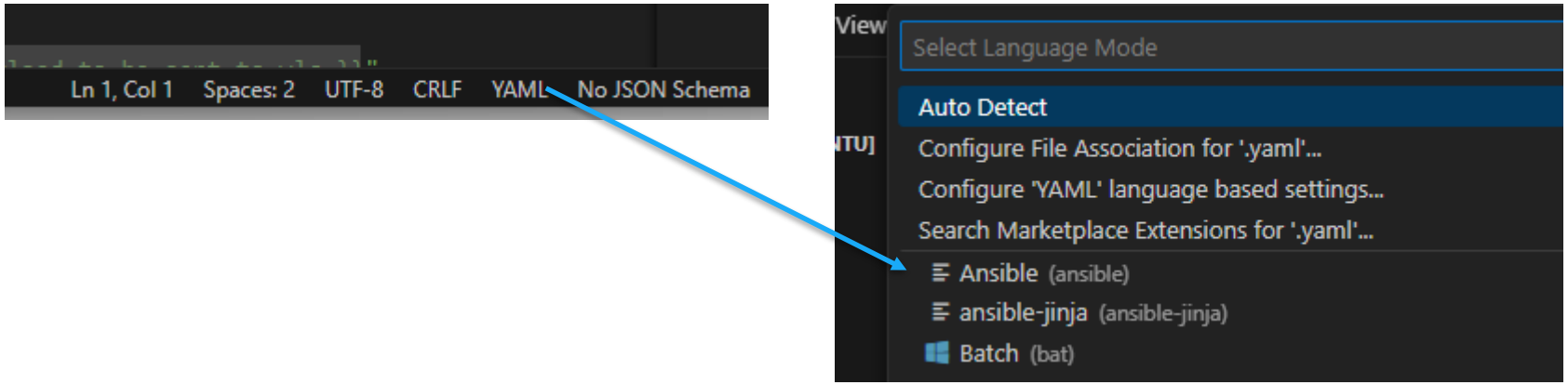



```
/home/devnet-adm/ansible-venv/bin/python3
```

- Bottom line should now show the Ansible version (2.17.3) and Python version and venv

# VS Code - Choose file type

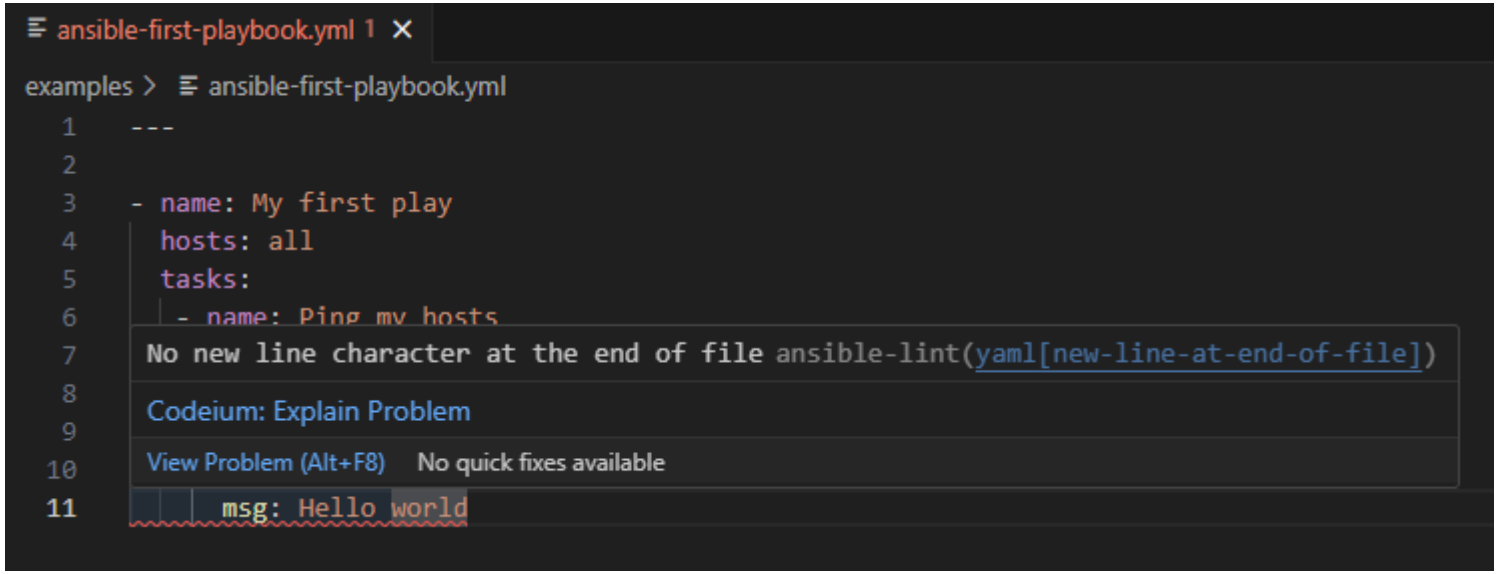
- If not auto detected, file type can be changed for open files



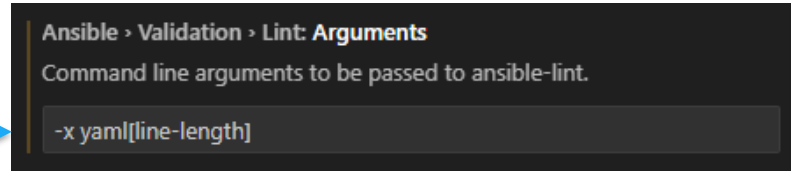

- Tips: If the file name contains the word «playbook», it will automatically be recognized as Ansible by VS Code (and not general YAML)
  - interface-playbook.yaml
  - demo-playbook.yaml

# 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




- (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

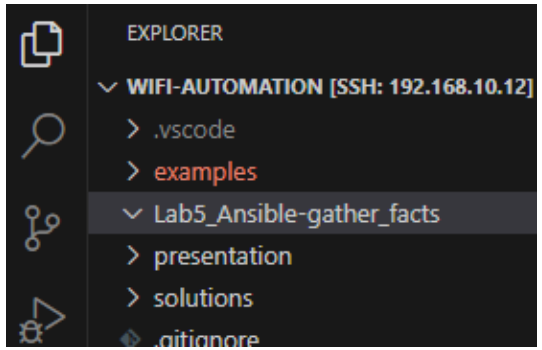


# Gather facts

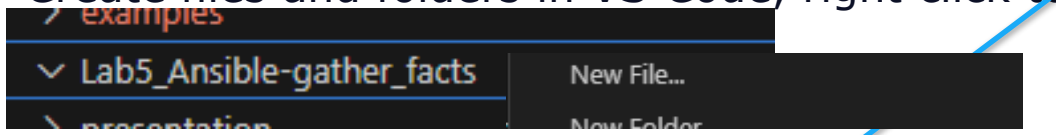
- Create a folder

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

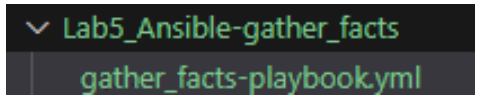
- Shows up automatically in VS Code




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



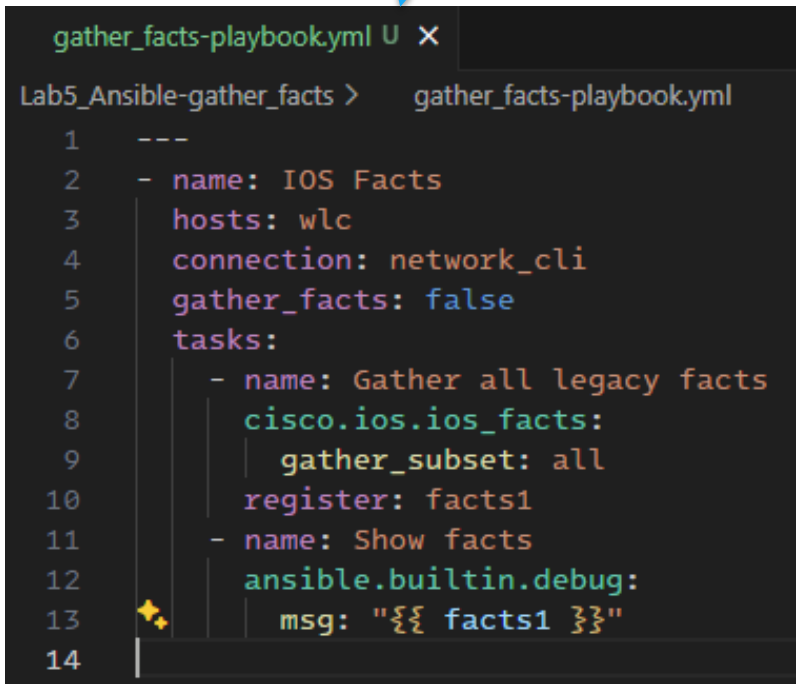

- Create the file "gather_facts-playbook.yml"




- Copy-paste from this textbox to VS Code

gather_facts-playbook.yml

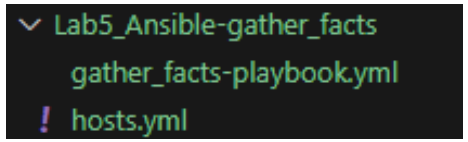
```
---
- name: IOS Facts
  hosts: wlc
  connection: network_cli
  gather_facts: false
  tasks:
    - name: Gather all legacy facts
      cisco.ios.ios_facts:
        gather_subset: all
      register: facts1
    - name: Show facts
      ansible.builtin.debug:
        msg: "{{ facts1 }}"
```






# Ansible hosts file

- Let's create a file named 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$
```

hosts.yml

```
wlc:
  hosts:
    192.168.10.{WLC-IP}:
  vars:
    ansible_connection: network_cli
    ansible_network_os: ios
    ansible_ssh_pass: ChangeMe2024!
    ansible_password: ChangeMe2024!
    ansible_user: devnet-adm
```

Change to YOUR WLC IP

```
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: ChangeMe2024!
8      ansible_password: ChangeMe2024!
9      ansible_user: devnet-adm
```



# Fix dependencies

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

```
(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] *************************************************************************************************************************
[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

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

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

```
(ansible-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
(ansible-venv) devnet-adm@ubuntu-devnet:~$
```



# SSH host key checking

- Try running again. The SSH connection is failing

```
(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] *************************************************************************************************************
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'.\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: ChangeMe2024!
8       ansible_password: ChangeMe2024!
9       ansible_user: devnet-adm
10      ansible_host_key_checking: False
11
```

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: ChangeMe2024!
    ansible_password: ChangeMe2024!
    ansible_user: devnet-adm
    ansible_host_key_checking: False
```



# Gather facts

- Successful play

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                   bash - Lab5_Ansible-gather_facts

(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: "{{ facts1.ansible_facts.ansible_net_interfaces }}"
```

```
- name: Show facts
  debug:
    msg: "{{ facts1.ansible_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",
```



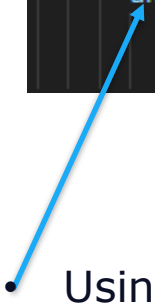
gather_facts-playbook.yml

```
---
- 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' }}"
```

# 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: 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



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: 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' }}"
        mode: "0600"
```

# Linting errors, and fixing them

- When saving this file you should get two linting errors

```
5   gather_facts: no
6   tasks:
7     - name: Gather all legacy facts
8       cisco.ios.ios_facts:
9         gather_subset: all
10      register: facts1
11    - name: Write run-config to file
```

```
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"

```
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      - name: Write run-config to file
12        ansible.builtin.copy:
13          content: "{{ facts1.ansible_facts.ansible_net_config }}"
14          dest: "{{ './' + facts1.ansible_facts.ansible_net_hostname + '-run-conf.txt' }}"
15          mode: "0600"
16
```



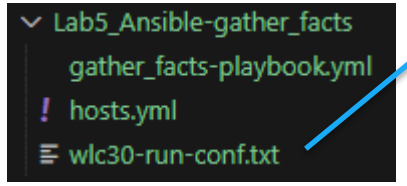
# Run the playbook

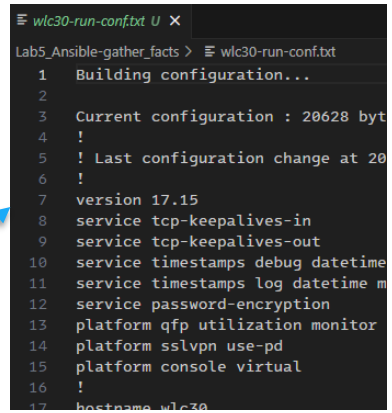
```
(ansible-venv) devnet-adm@ubuntu-devnet:~/wifi-automation/Lab5_Ansible-gather_facts$ ansible-playbook -i hosts.yml gather_facts-playbook.yml

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



# Lab exercise #6: Install YANG Suite (optional)

- !!! NOTE !!!
  - If you do not want to use time on installing YANG Suite at this point, you can use the already prepared YANG Suite installation on one of the shared servers:
    - https://192.168.10.7:8443
    - https://192.168.10.8:8443
  - You can skim through the following slides and continue from Lab Exercise 8

- The first couple of slides will explain a little about YANG models and YANG Suite
- Then, installation of YANG Suite as a docker container is explained
- After successful installation, you can connect to YANG Suite on
  - https://{your Ubuntu IP}:8443
- Then you will use YANG Suite to pull the YANG models from your own WLC



# 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
     |  +--ro ap-name?                       string
     |  +--ro ms-ap-slot-id?                 uint8
     |  +--ro ms-radio-type?                 wireless-client-types:ms-phy-radio-ty
     |  +--ro wlan-id?                       uint32
     |  +--ro client-type?                   wireless-client-types:ms-client-type
     |  +--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/17121



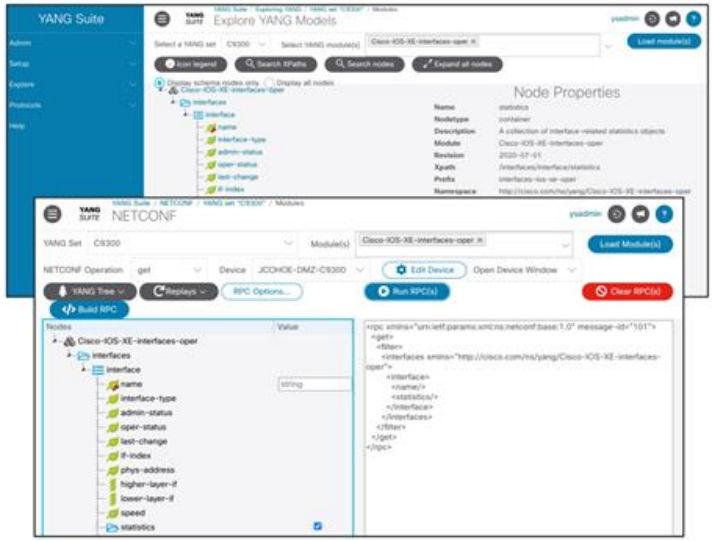
# YANG Suite

## 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

github.com/CiscoDevNet/yangsuite

**Cisco-IOS-XE-wireless: Modules**

| Config | | Oper | |
|---|---|---|---|
| ap | mesh | access-point | mesh |
| apf | mobility | ap | mobility |
| cts-sxp | mstream | awips | nmsp |
| dot11 | radio | ble-ltx | rfid |
| dot15 | rf | ble-mgmt | rogue |
| fabric | Rfid | client | rrm |
| file-transfer | rlan | cts-sxp | rule-mdns |
| flex | rogue | events | |
| fqdn | rrm | general | |
| general | rule | hyperlocation | |
| hotspot | security | lisp-agent | |
| image-download | site | location | |
| location | tunnel | mcast | |
| me-general | wlan | mdns | |

# 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: ChangeMe2024!
confirm password: ChangeMe2024!
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

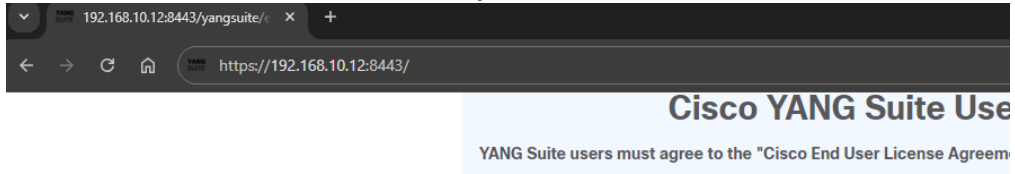
- When you get this output, it is ready to run

```
docker-yangsuite-1  | *** 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)
```

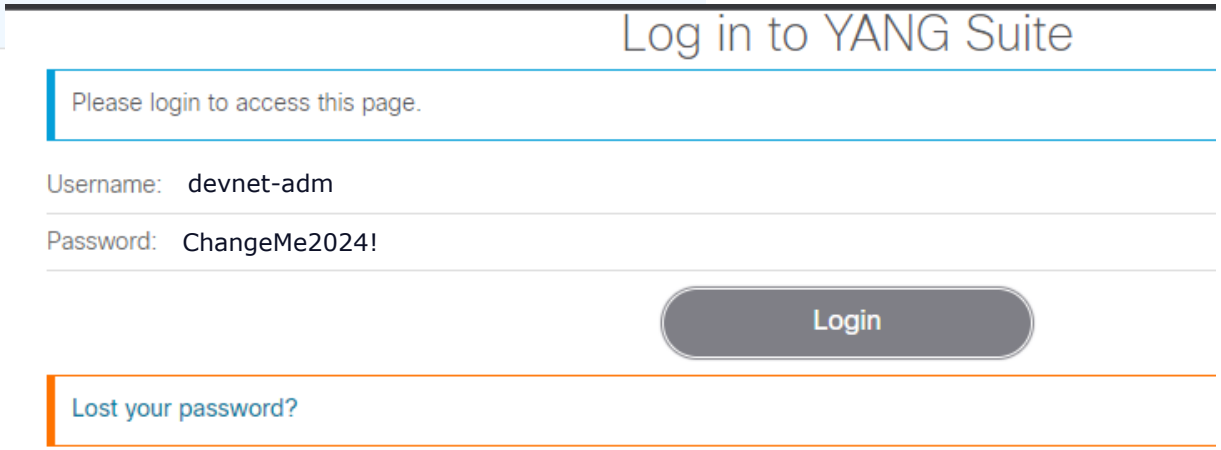
- To "free up" the command line, press Ctrl+C to stop YANG Suite
- Then, to start the container in "detached" mode:

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

- Access via HTTPS: https://{Your Ubuntu IP}:8443

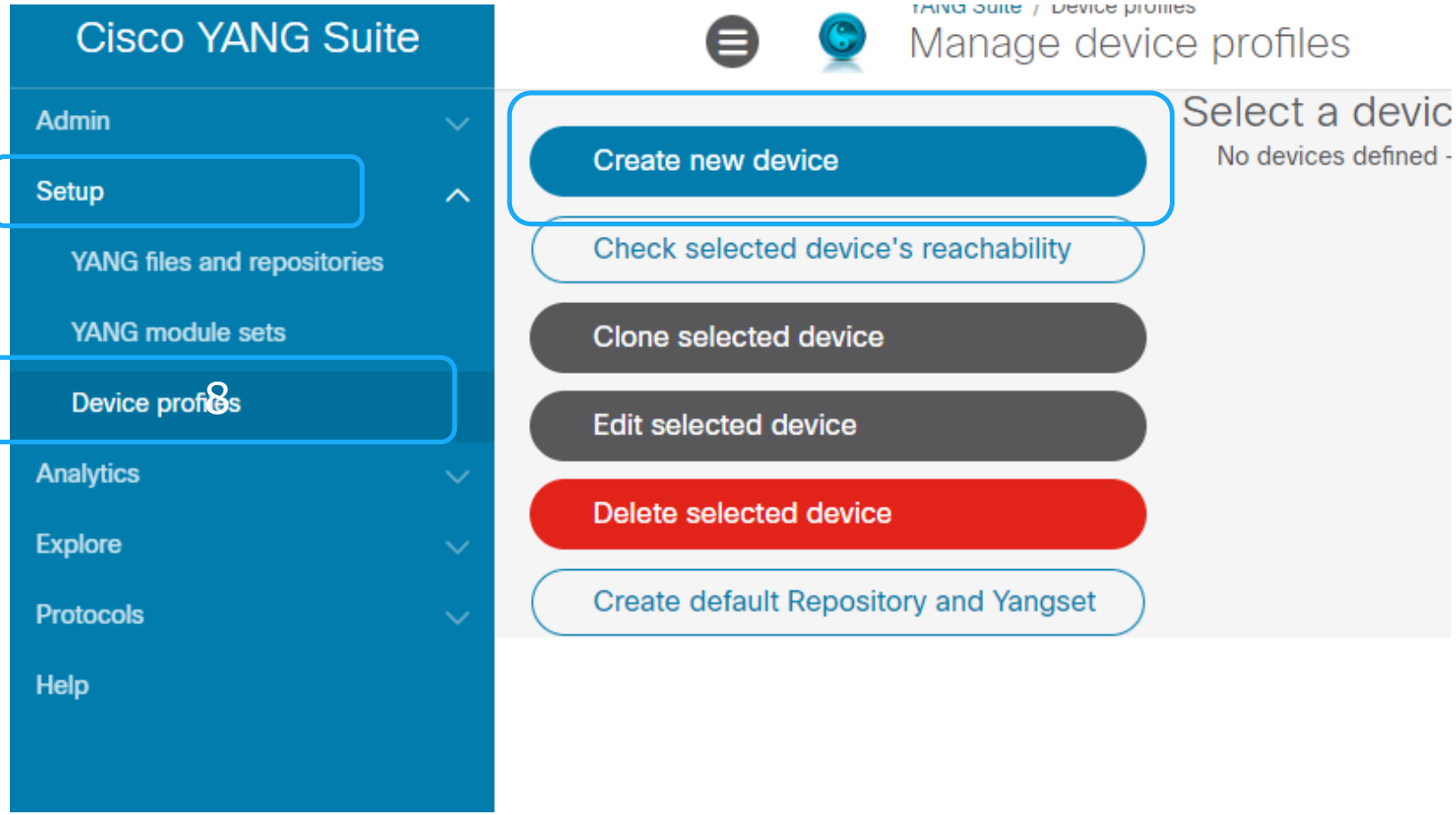


# YANG Suite EULA

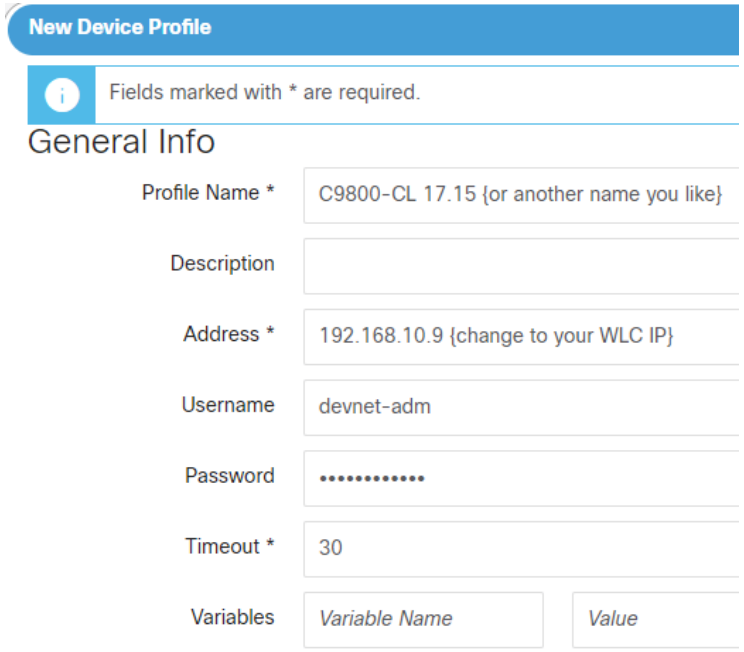
# Create new device

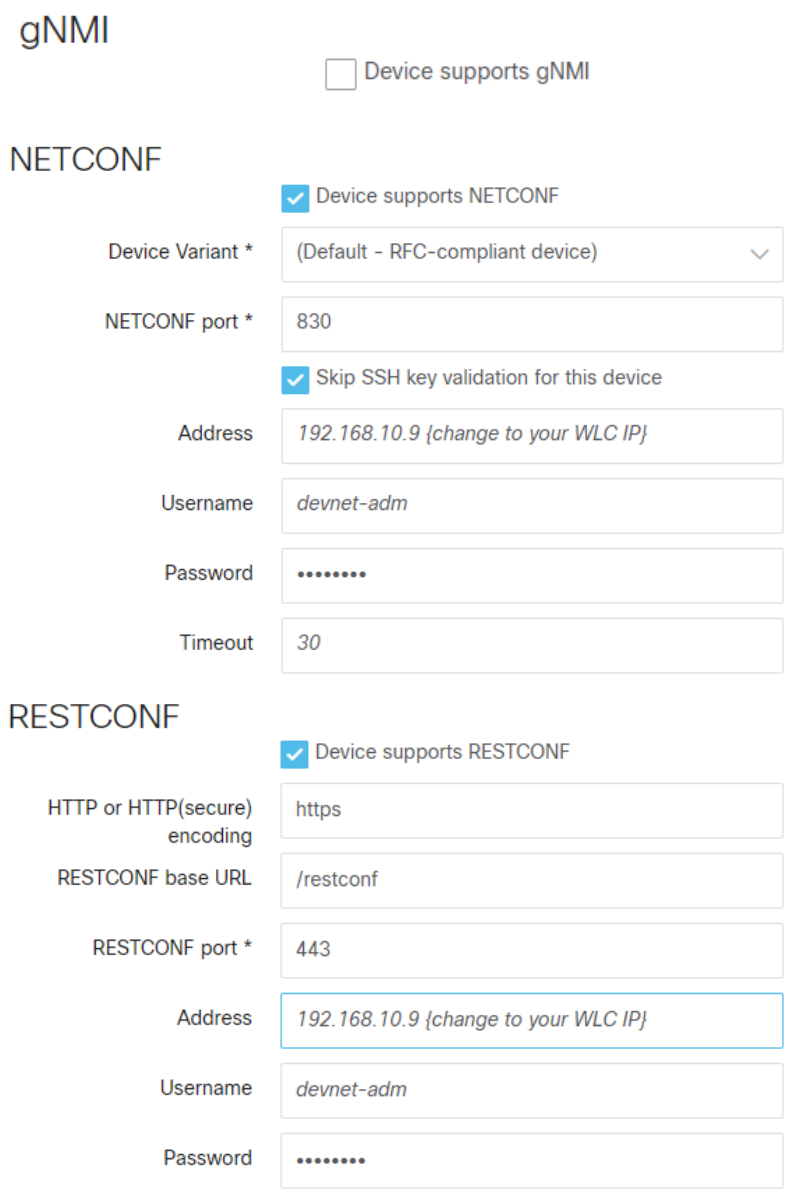
- Create new device

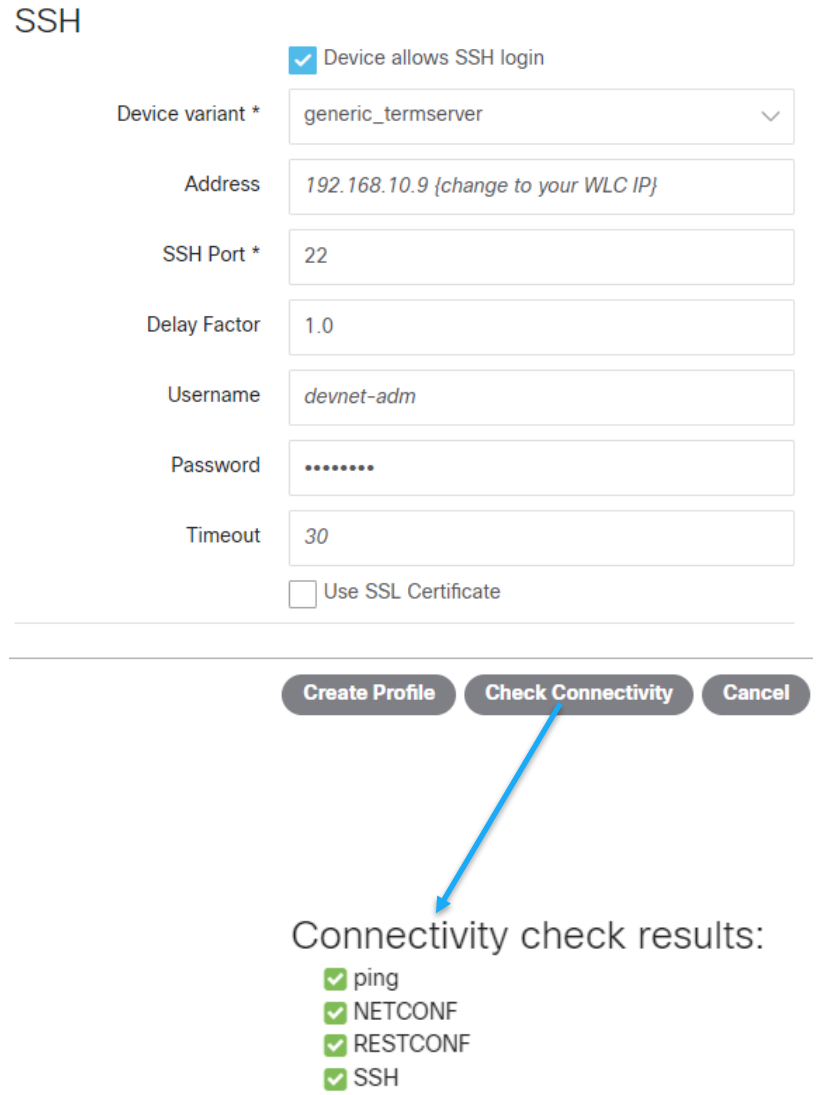


# Create new device

- Create new device
- Use your WLC IP

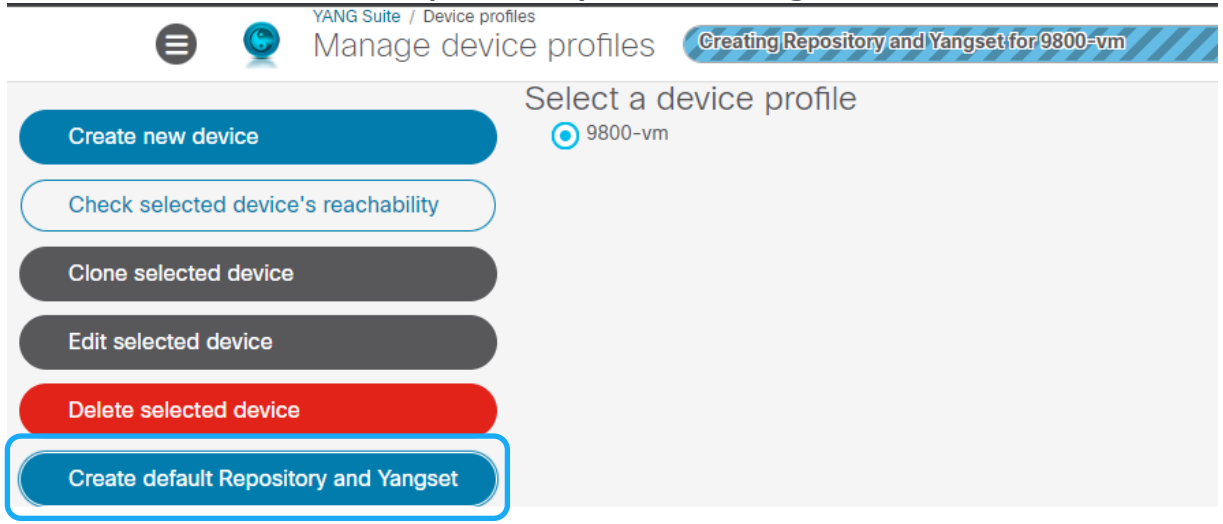


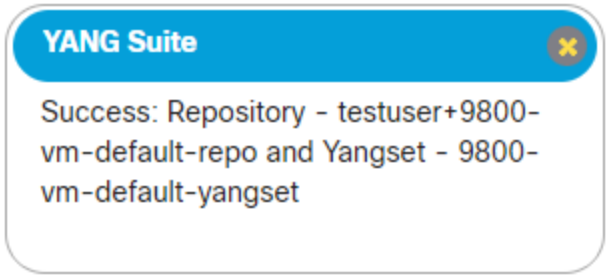



# Create repository and Yangset

- Create default repository and Yangset



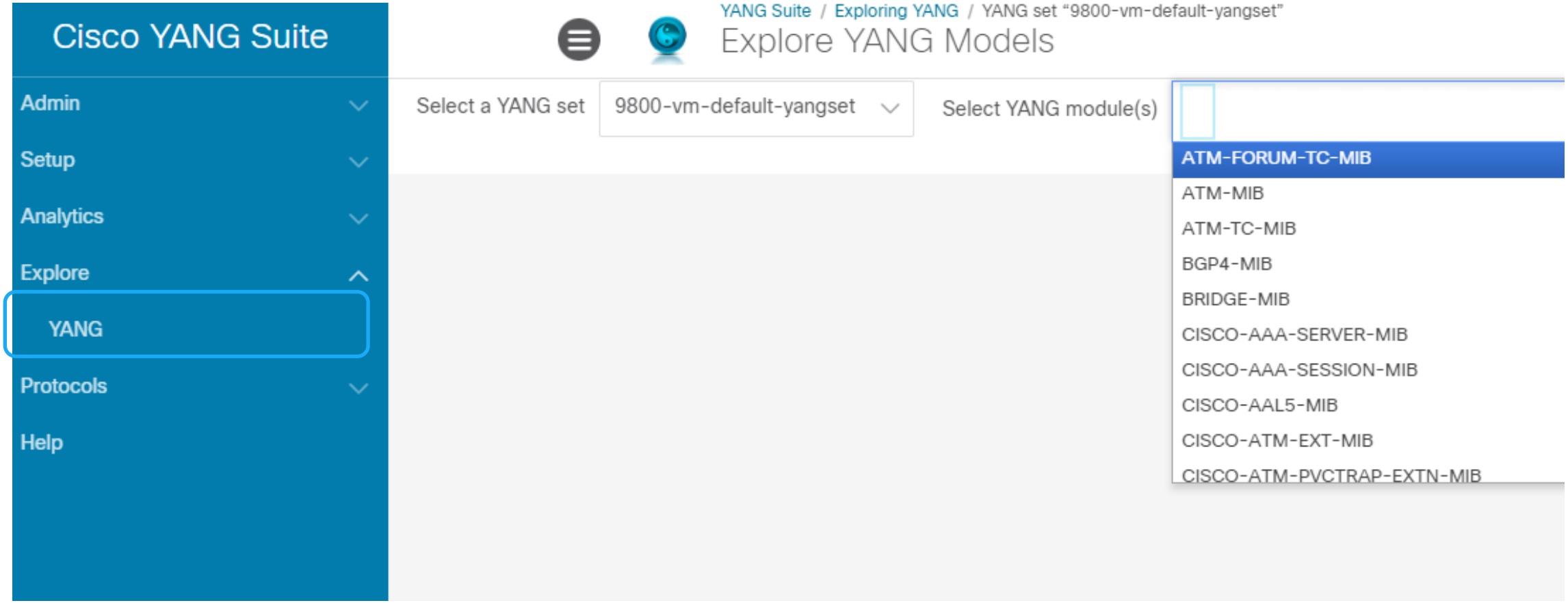

- (this will take a couple of minutes, so just be patient, while waiting you can sneak-peek on the next slides)



# Explore YANG models

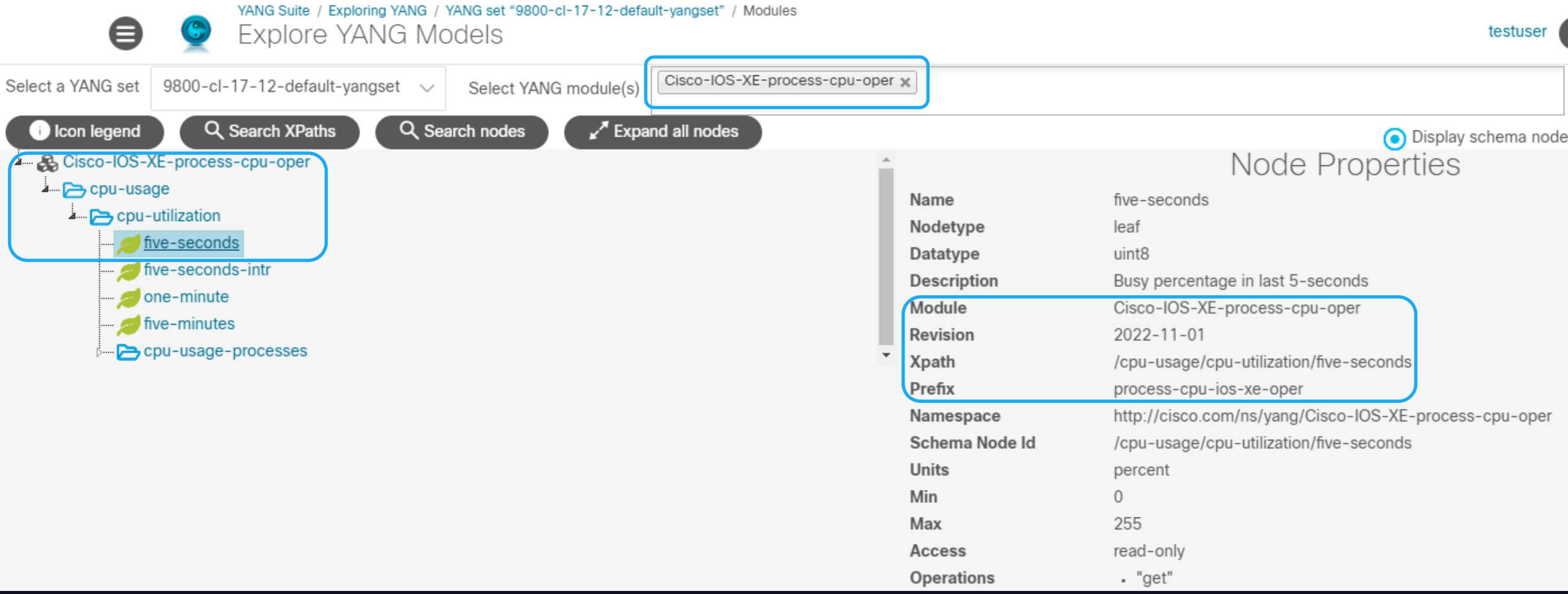
- «oper» -> Operational data
- «cfg» -> Configuration and config data

- Here is how the window looks
- The next slides will show some models to explore in detail



# YANG Suite

- Explore YANG Models



# A word about gNMI (optional)

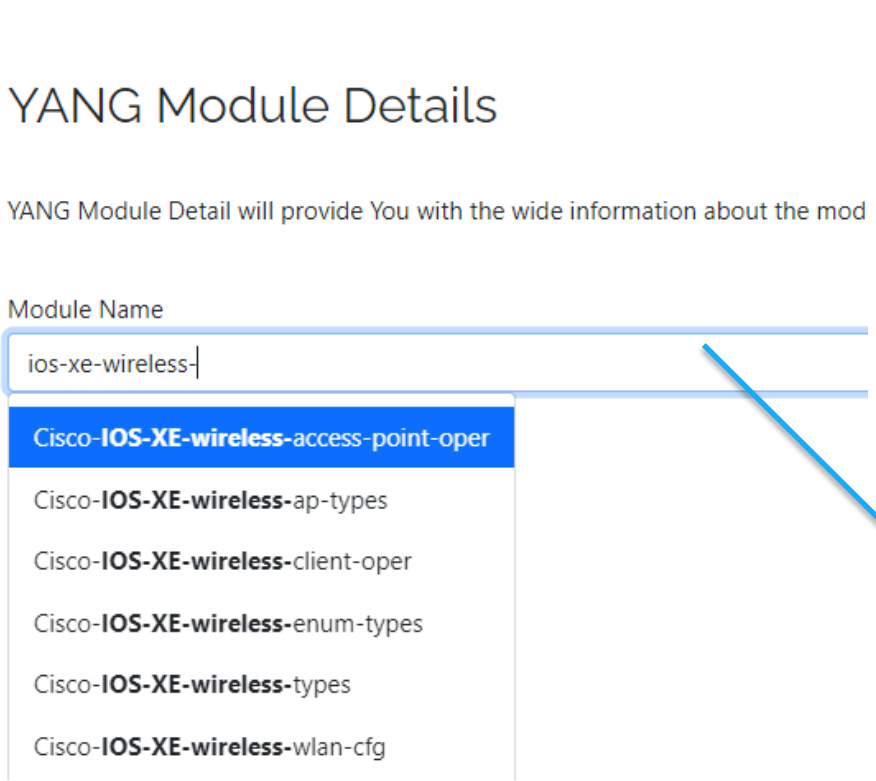
- Another programmatic interface (like NETCONF and RESTCONF)
- gNMI is not covered in this lab, but instructions to enable is found in the link below
- 9800 programmability & telemetry (see the gNMI section)
  - https://www.cisco.com/c/en/us/td/docs/wireless/controller/9800/technical-reference/catalyst-9800-programmability-telemetry-deployment-guide.html



# yangcatalog (optional)

- Just for reference, here is another web-based way to look at the YANG models

https://www.yangcatalog.org/yang-search/module_details




```
module: Cisco-IOS-XE-wireless-wlan-cfg
  +--rw wlan-cfg-data
     +--rw calendar-profile-configs
     |  +--rw calendar-profile-config* [profile-name]
     |     +--rw profile-name                string
     |     +--rw start-time                  string
     |     +--rw end-time?                   string
     |     +--rw recurrence                  wireless-enum-typ
     |     +--rw calendar-weekly-configs
     |     |  +--rw calendar-weekly-config* [day]
     |     |     +--rw day    wireless-enum-types:work-day
     |     +--rw calendar-monthly-configs
     |        +--rw calendar-monthly-config* [date]
```

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



# Using YANG paths from https://yangcatalog.org

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

Use this for WLC telemetry config

Use this for RESTCONF path

| Element | Schema | Type | Flags | Opts | Status | Path | Sensor Path |
|---|---|---|---|---|---|---|---|
| Cisco-IOS-XE-wireless-access-point-oper | module | module | | | | | |
| access-point-oper-data | container | container | no config | | current | /wireless-access-point-oper:access-point-oper-data | /Cisco-IOS-XE-wireless-access-point-oper:access-point-oper-data |
| ap-radio-audit-info | list | list | no config | | current | /wireless-access-point-oper:access-point-oper-data/wireless-access-point-oper:ap-radio-audit-info | /Cisco-IOS-XE-wireless-access-point-oper:access-point-oper-data/ap-radio-audit-info |
| radio-oper-data | list | list | no config | | current | /wireless-access-point-oper:access-point-oper-data/wireless-access-point-oper:radio-oper-data | /Cisco-IOS-XE-wireless-access-point-oper:access-point-oper-data/radio-oper-data |

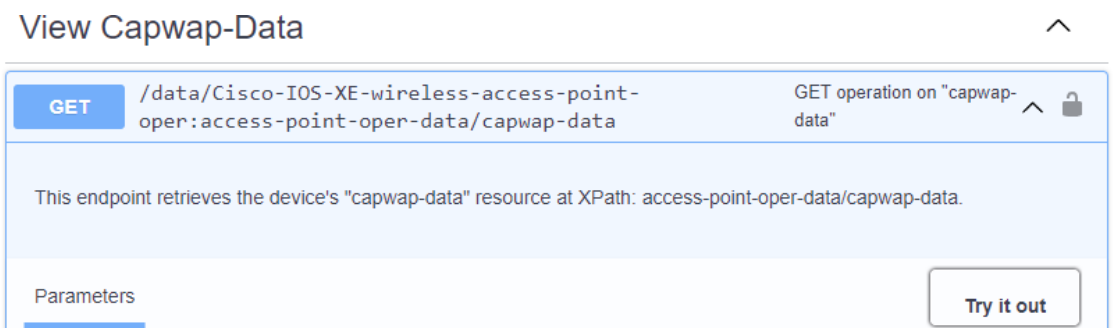
```
no telemetry ietf subscription 4
telemetry ietf subscription 4
 encoding encode-kvgpb
 filter xpath /wireless-access-point-oper:access-point-oper-data/wireless-access-point-oper:radio-oper-data
 stream yang-push
 update-policy periodic 500
 receiver ip address 192.168.10.7 57000 protocol grpc-tcp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

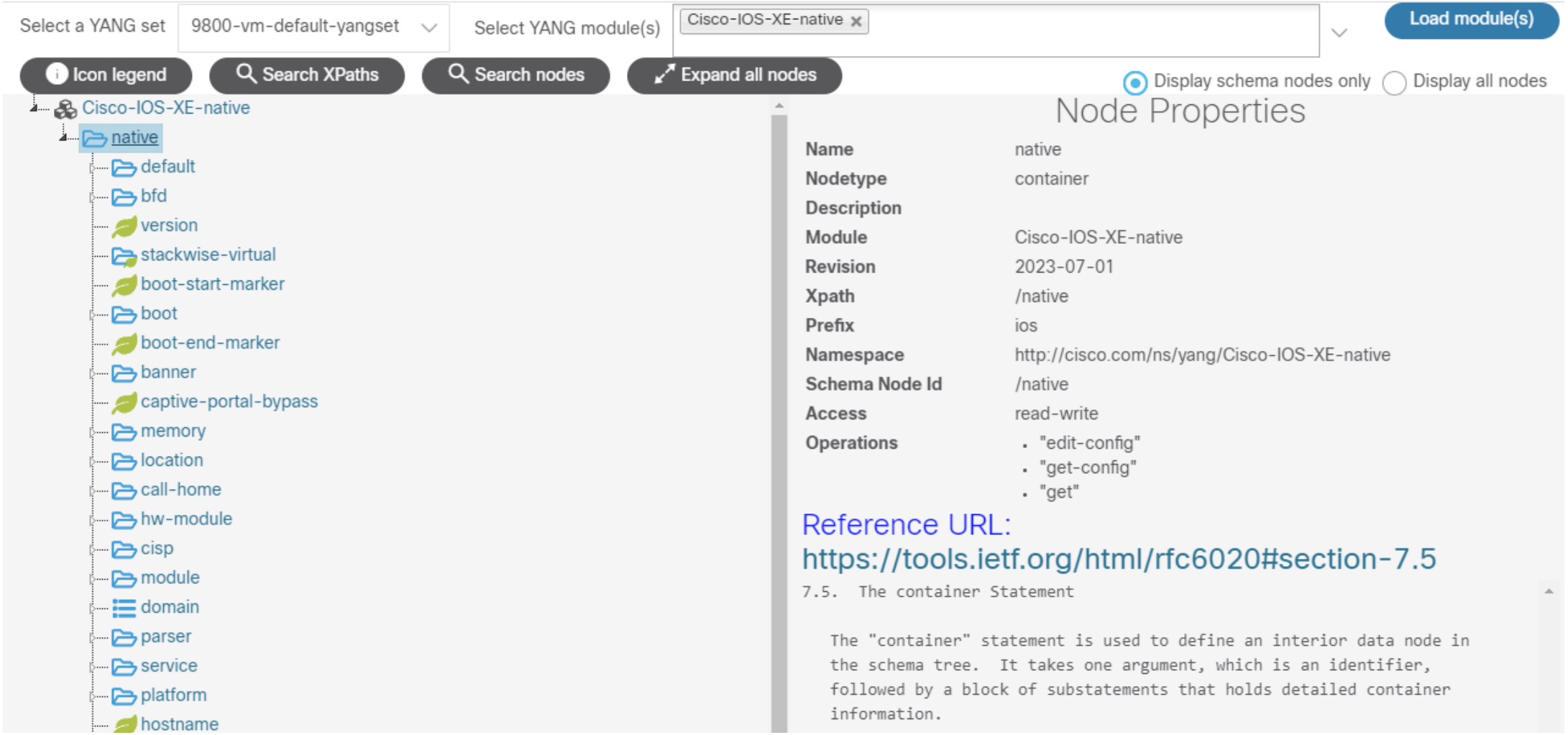


# Lab exercise #7: Explore YANG Suite

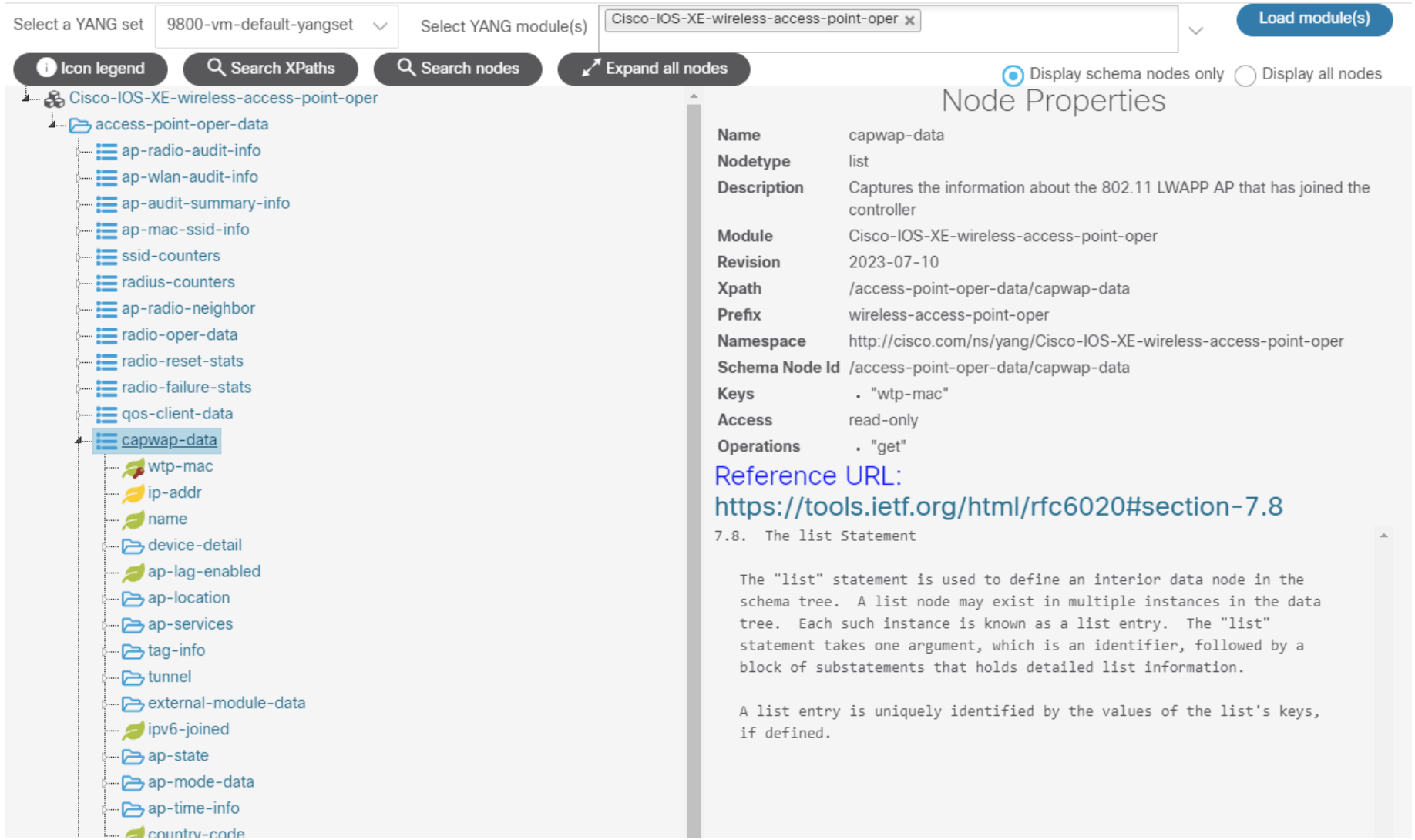
- Over the next pages, you will explore a few YANG models
  - Get config (YANG/JSON version)
  - Get AP summary
  - Get Clients
- Just get somewhat familiar, you will use these in the next exercises (Postman, Python, Ansible, Grafana)
- This will show you the model, not pull the actual data
- To pull data we will use these models i 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



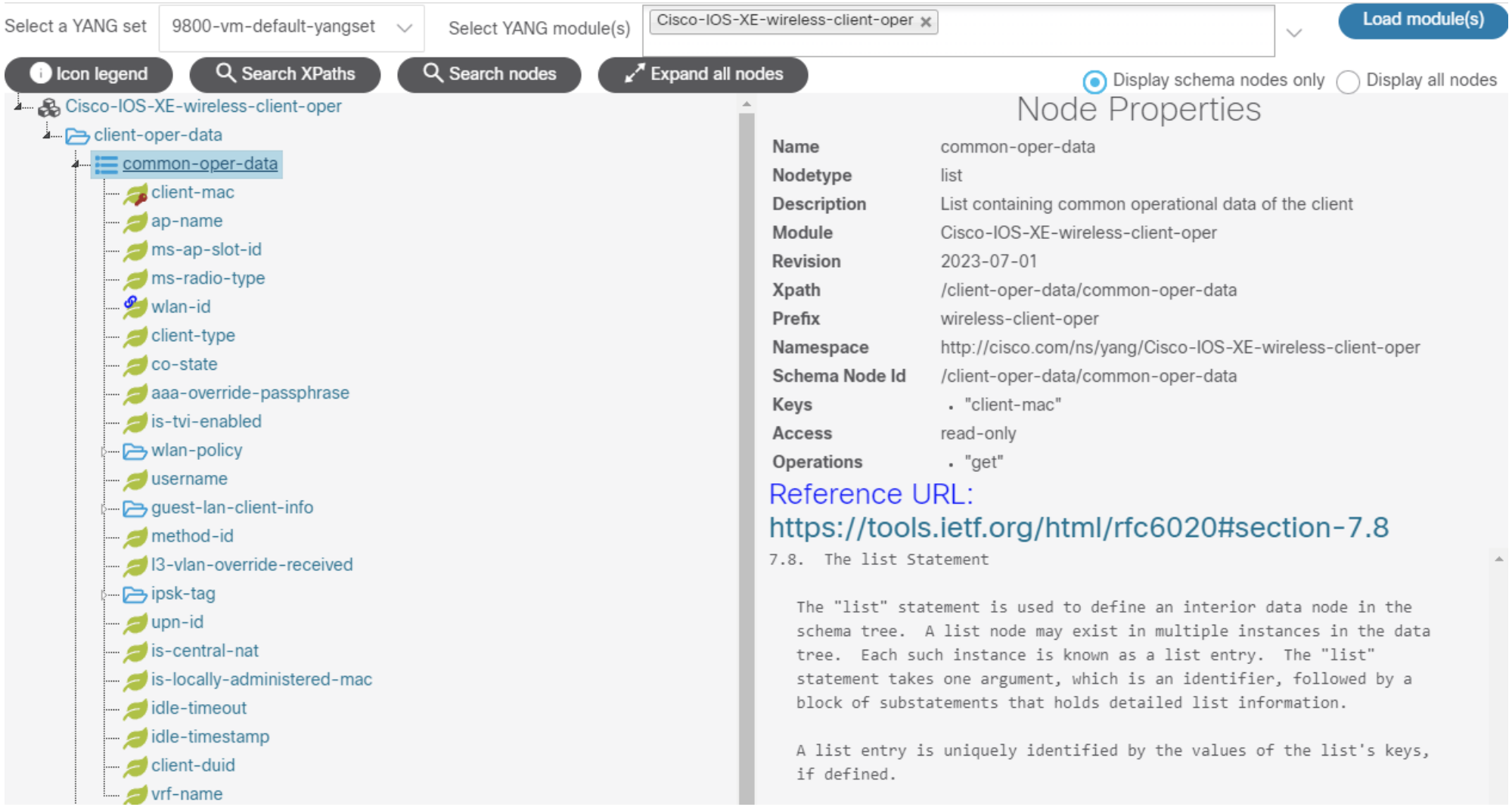

# Get config (YANG version)



# Get AP summary



# Get Client summary



# Lab exercise #8: Explore Postman

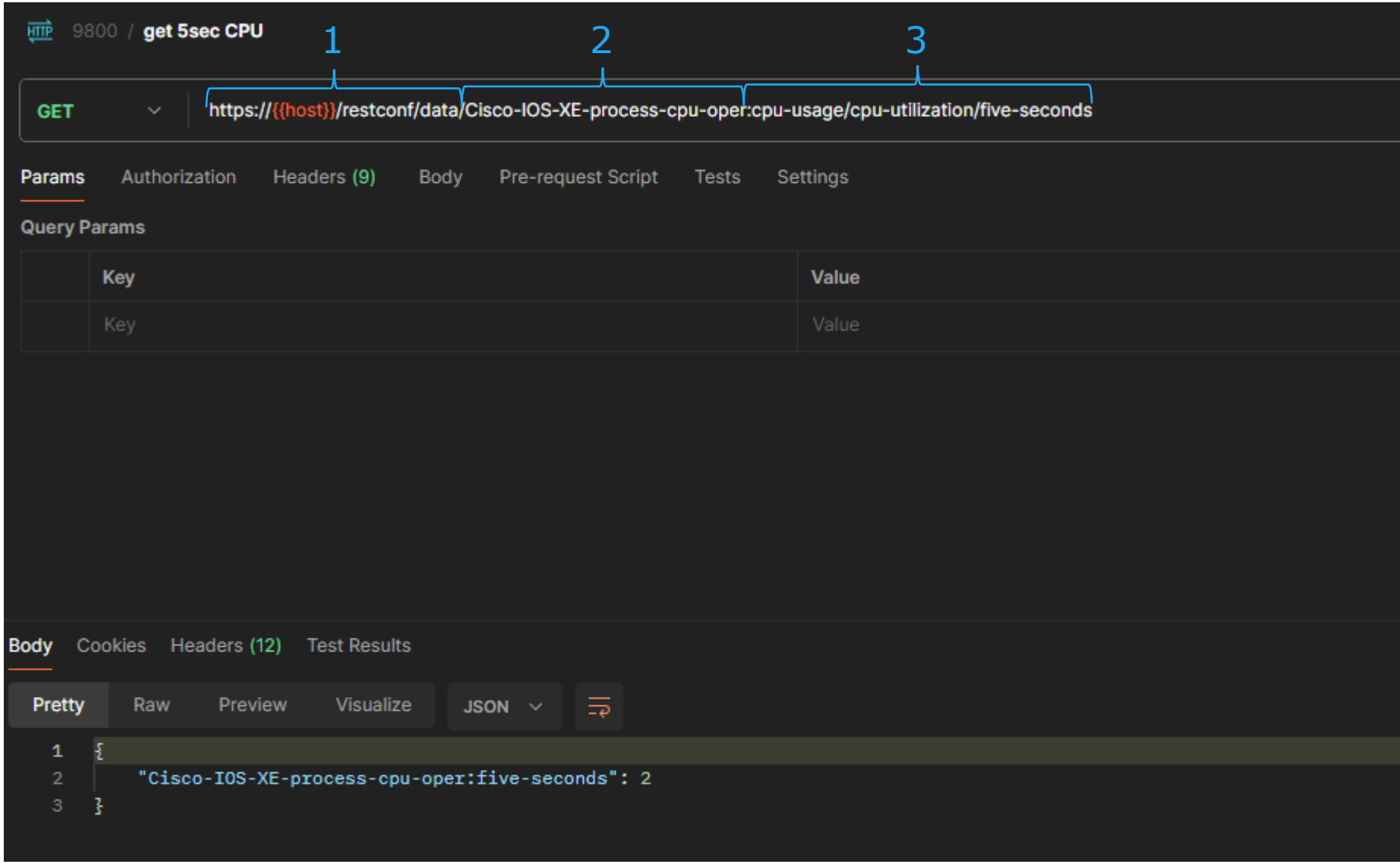
- In this lab exercise we will
  - Create a Workspace
  - Create an Environment
  - Create a Collection
  - Test some queries found from YANG Suite
  - Output to CURL or Python
  - (optional …and time-consuming) Test some queries to Meraki, Aruba, MIST or Cisco Always-On labs
    - (specific instructions for this is not finished to Prague'24, will be added to the slide deck later)
    - Most "cloud native" vendors have good guides to get started using the APIs with Postman. So you could try creating your API key in the relevant dashboard, and get your inventory of APs.



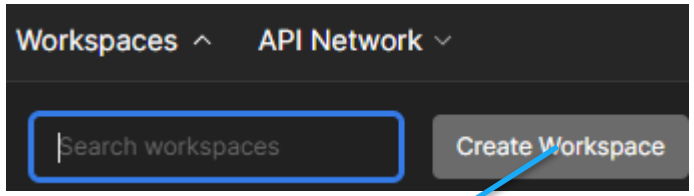
# Overview

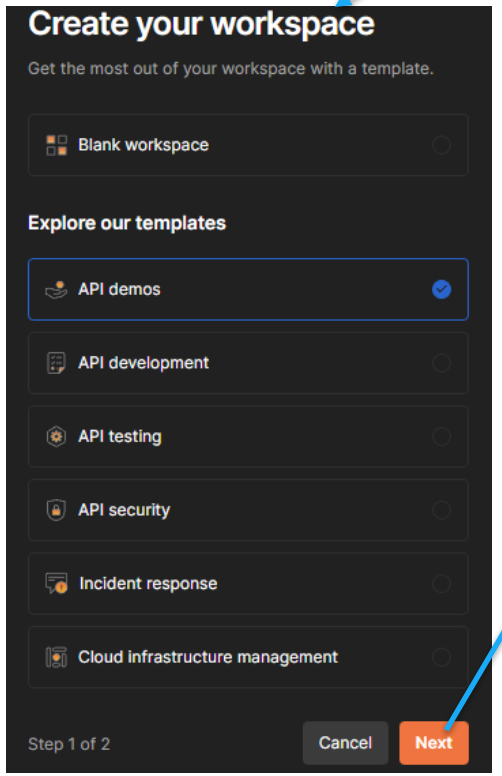
1. RESTCONF path
2. YANG module
3. Xpath

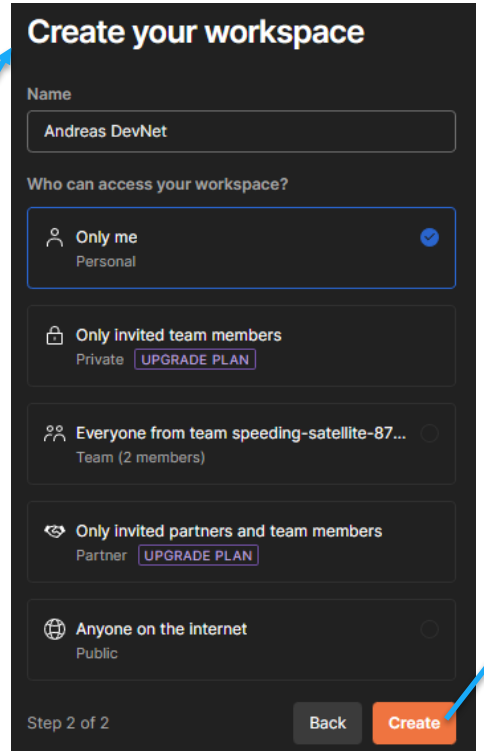
Postman is a "go-to" tool for RESTCONF APIs. Some examples for automation is testing and validating RESTCONF calls to IOS-XE devices before implementing in Python, Ansible, etc.

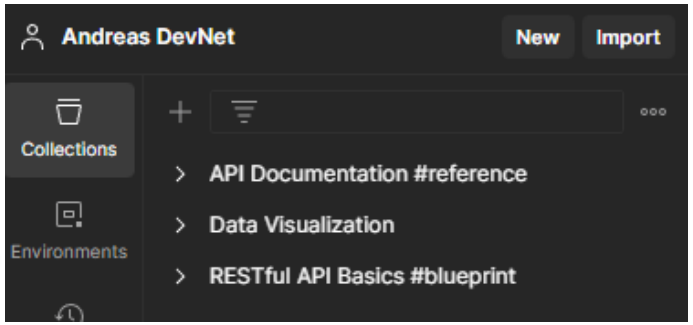


# Create Workspace

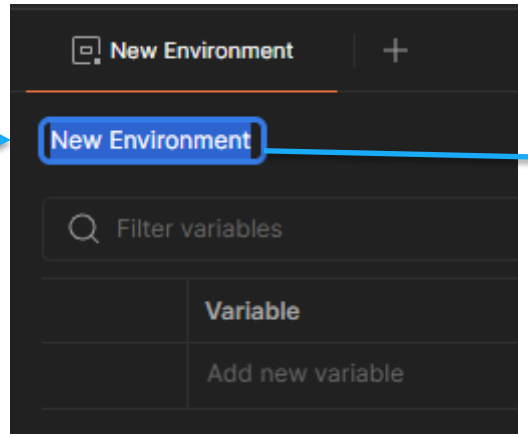


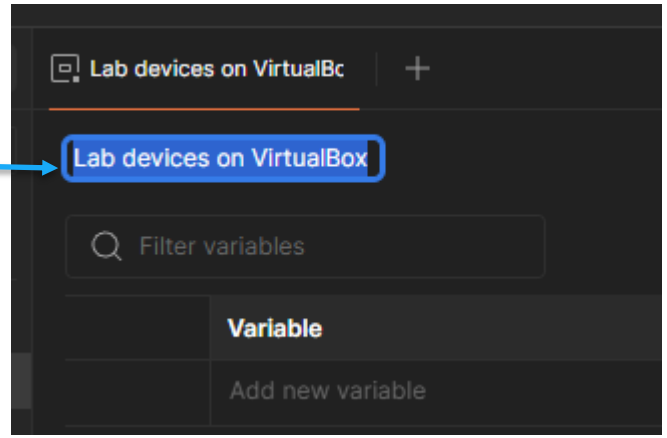



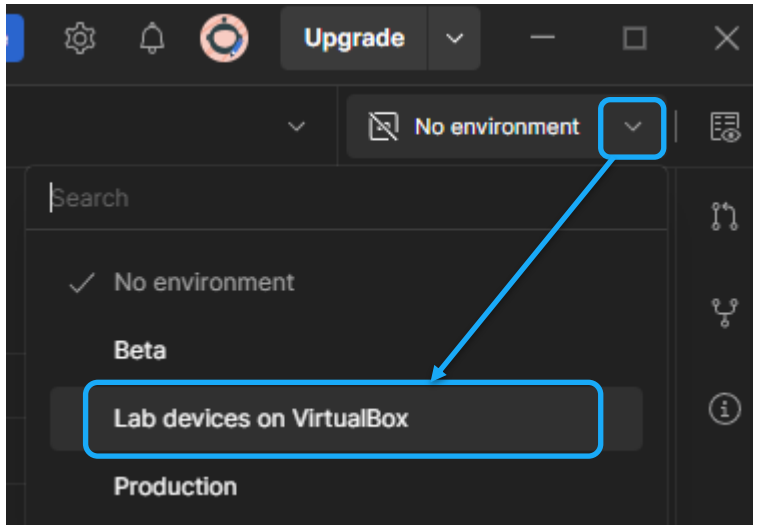



# Create Environment

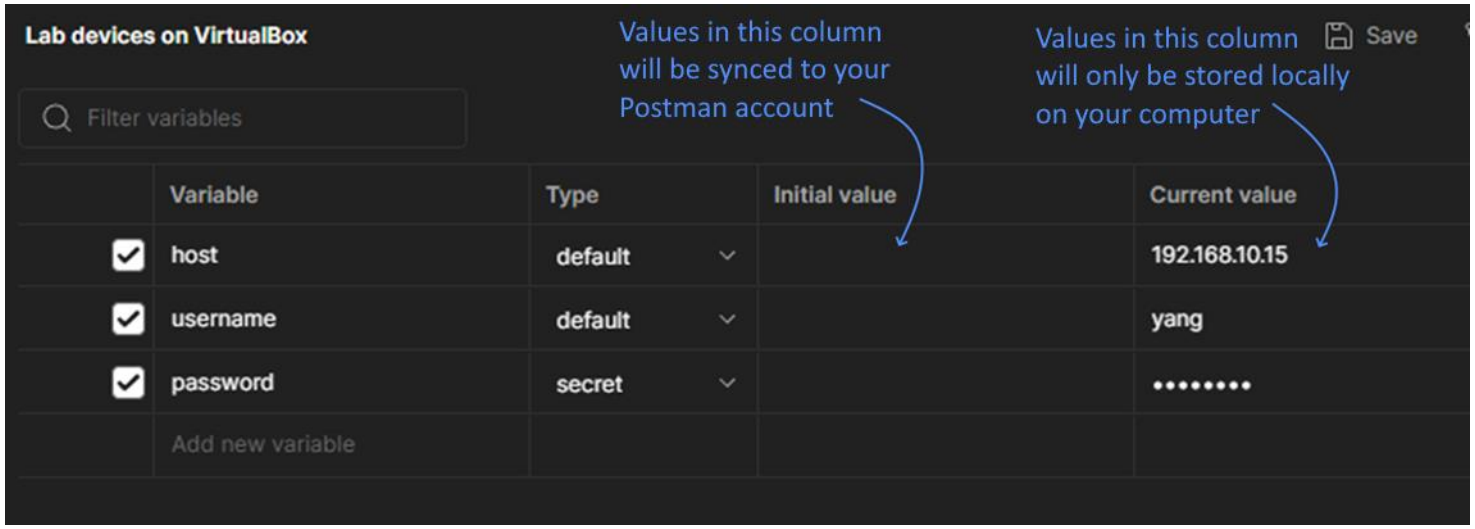



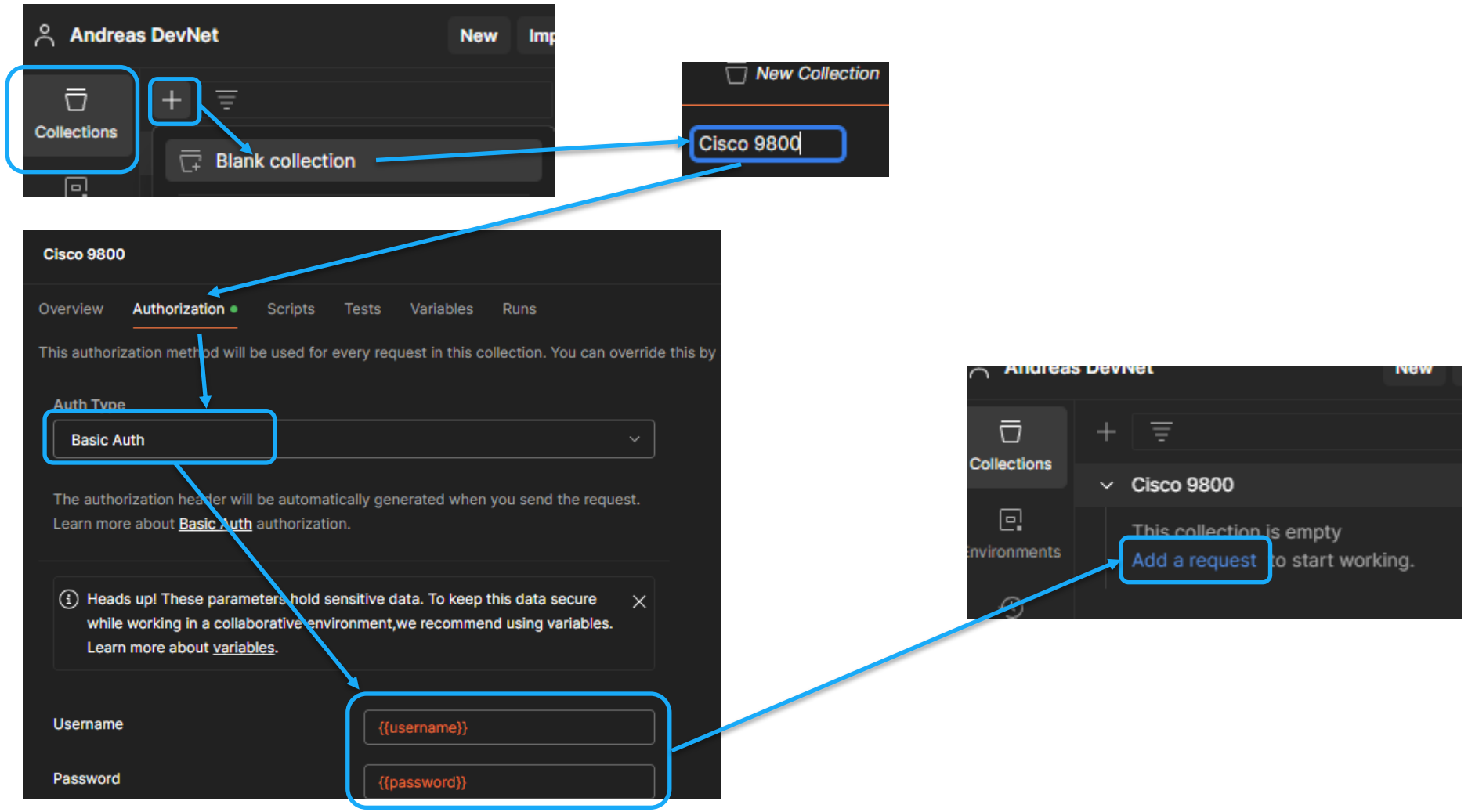

- Activate the new environment



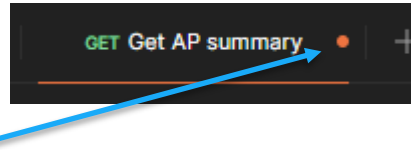

... and create some variables



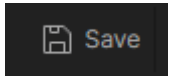
# Create Collection



# 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 

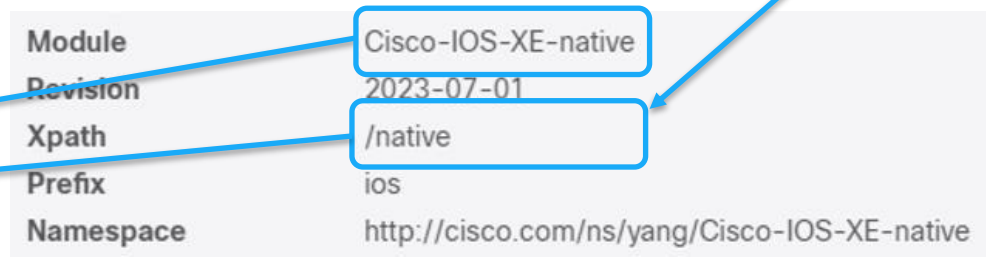

- An example: When you run a RESTCONF call with variables from an environment, like {{host}} or {{password}} it will use the value from your SAVED file. So if you do changes in the environment, remember to save the tab

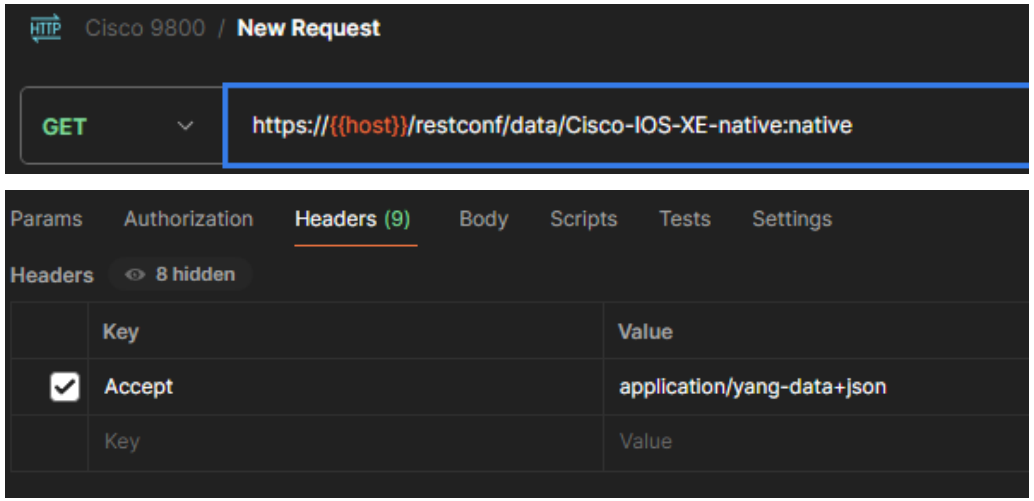
# Get config

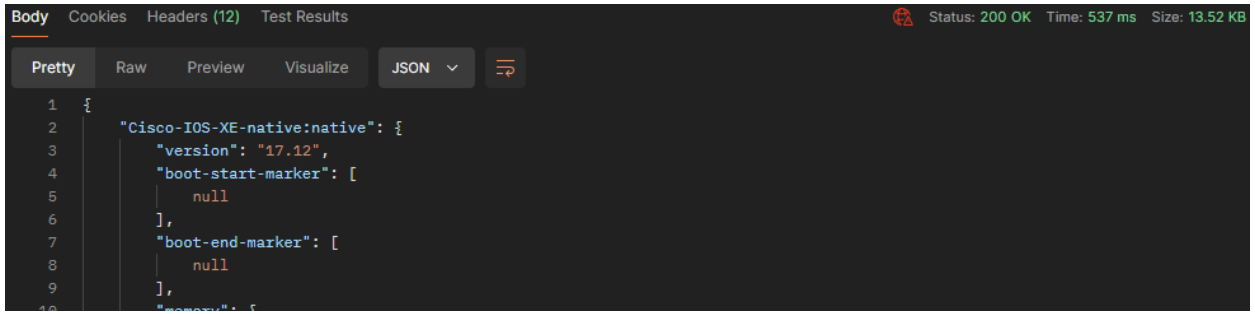
- Create the Postman queries based on the information explored from YANG Suite
  - Restconf path: https://{{host}}/restconf/data/
  - YANG-module: Cisco-IOS-XE-native
  - Xpath: native
  - Combined request path for copy-paste:

```
https://{{host}}/restconf/data/Cisco-IOS-XE-native:native
```

Use without the forward slash (/)



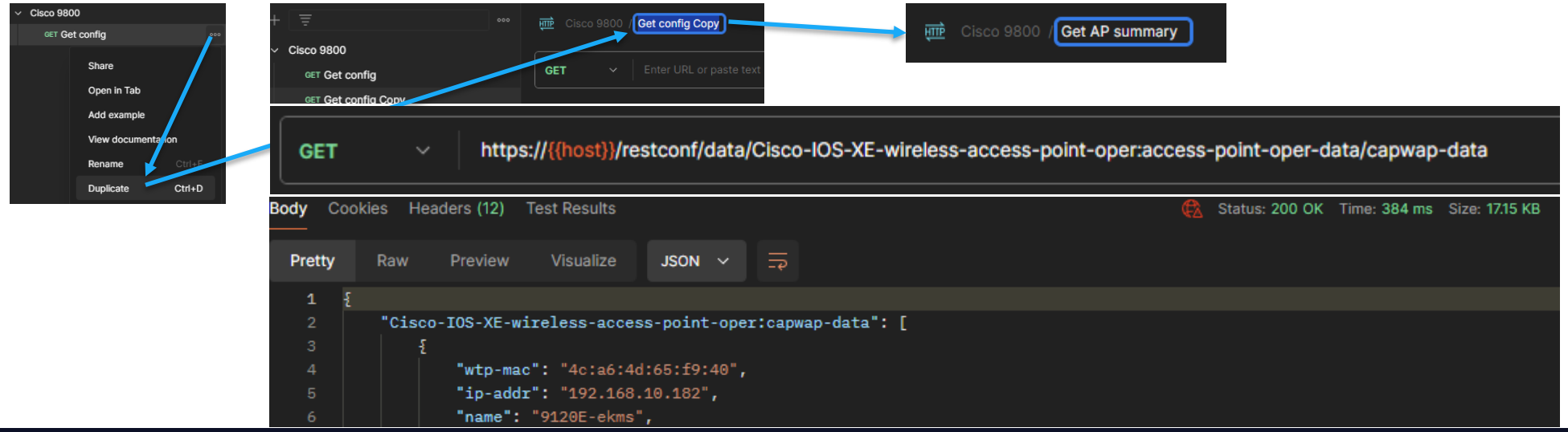



# Get AP summary

- Create the Postman queries based on the information explored from 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
```

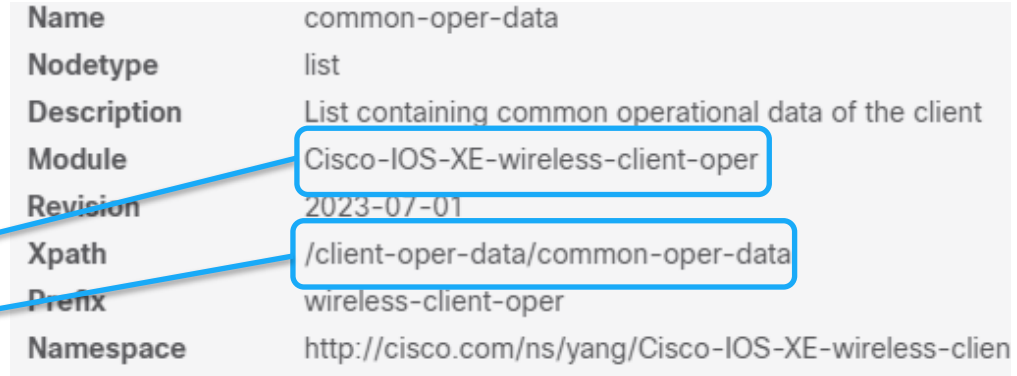
| | |
|---|---|
| Name | capwap-data |
| Nodetype | list |
| Description | Captures the information about the 802.11 LWAPP AP that has jo controller |
| Module | Cisco-IOS-XE-wireless-access-point-oper |
| Revision | 2023-07-10 |
| Xpath | /access-point-oper-data/capwap-data |
| Prefix | wireless-access-point-oper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

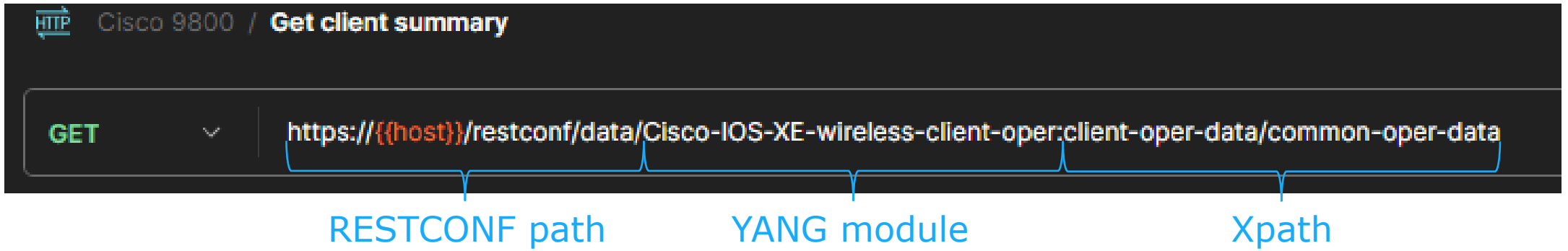


# Get client summary

- Create the Postman queries based on the information explored from 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/shared-oper-data
```

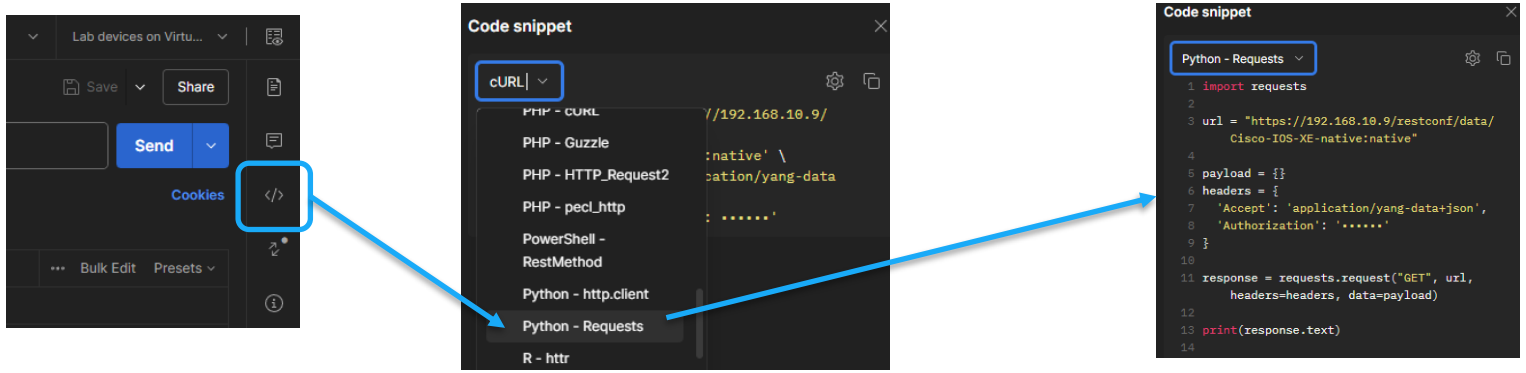



- If no users are connected, the return is "No Content" (status code 204)

# Output to Python

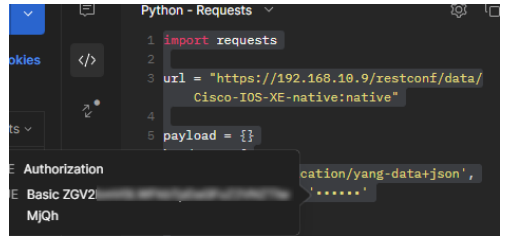
- You can output the RESTCONF call as various code snippets by clicking the "Code" button




- 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 ZGV2bmV0
... }
>>> response = requests.request("GET", url, headers=headers, data=payload, verify=False)
C:\Users\                                  \Python312\site-packages\urllib3\connectionpool.py:1103: Insecure
n is strongly advised. See: https://urllib3.readthedocs.io/en/latest/advanced-usage.html#tls-warnings
  warnings.warn(
>>> print(response.text)
{
  "Cisco-IOS-XE-native:native": {
    "version": "17.12",
```

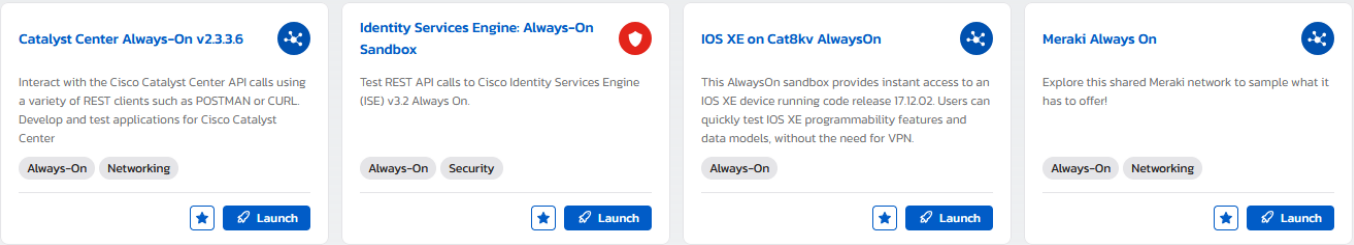
- You get this by hovering the mouse over the **** text in the Code snippet



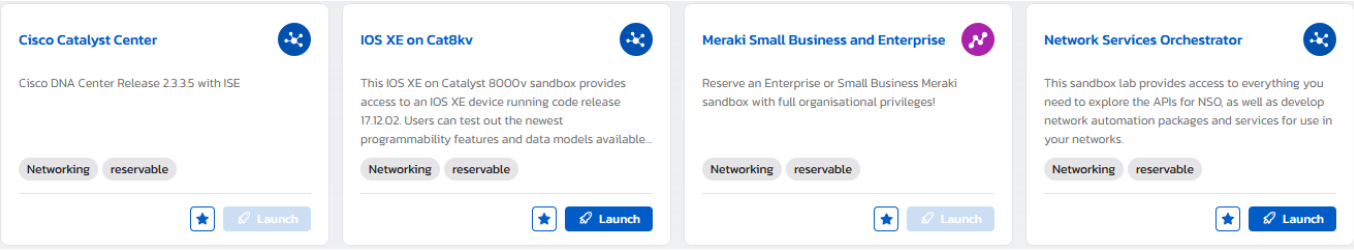

- Add "verify= False" to skip certificate checking on the host

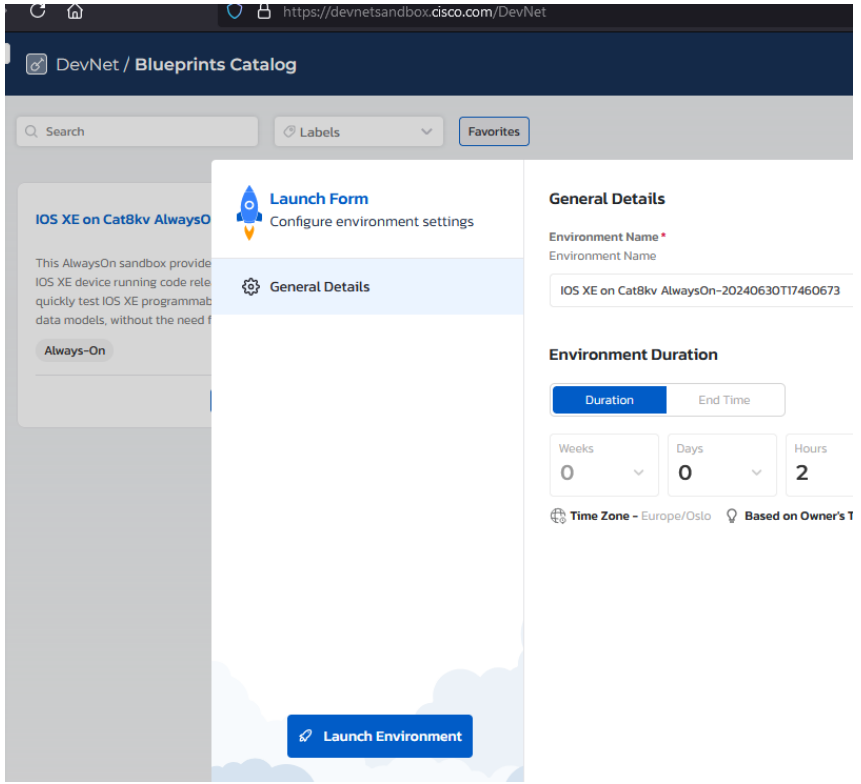
# Using Cisco DevNet Sandbox (optional)

- In this lab we will use C9800-L running on VirtualBox
- If you have time or want to do it later, you can also test against an always-on or on-demand Cisco IOS XE device at Cisco DevNet Sandbox
- https://developer.cisco.com/learning/labs/intro-dnac-api/authentication/
- Some always-on (RO) labs




- Some reservable (RW) labs



# Testing Aruba APIs (optional)

- This section is unfortunately not finished yet
- You may download the updated guide at a later stage, from https://github.com/akoksrud/wifi-automation
- To test this on your own, here is a starting point: https://developer.arubanetworks.com/hpe-aruba-networking-central/docs/postman-collection

# Testing Meraki APIs (optional)

- This section is unfortunately not finished yet
- You may download the updated guide at a later stage, from https://github.com/akoksrud/wifi-automation
- To test this on your own, here is a starting point: https://developer.cisco.com/meraki/api-v1/getting-started/#getting-started

# Testing MIST APIs (optional)

- This section is unfortunately not finished yet
- You may download the updated guide at a later stage, from https://github.com/akoksrud/wifi-automation
- To test this on your own, here is a starting point: https://www.mist.com/documentation/using-postman/



# Lab exercise #9: Explore Python automation

- In this exercise we will
  - Create a new python virtual environment (venv) for python lab-work
  - 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

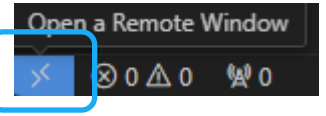
- Python venv for python files (not neccessarily the same as for ansible)
- Deactivate the Ansible venv if applicable
- Create a new venv for python lab work

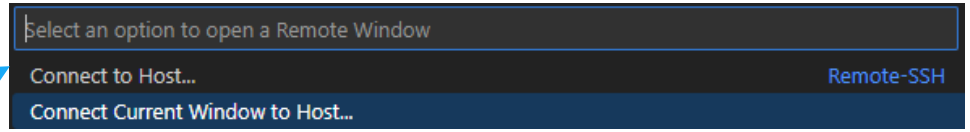
```
(ansible-venv) devnet-adm@ubuntu-devnet:~/wifi-automation/Lab5_Ansible-gather_facts$ cd ~
(ansible-venv) devnet-adm@ubuntu-devnet:~$ deactivate
devnet-adm@ubuntu-devnet:~$ mkdir ./python-lab-venv
devnet-adm@ubuntu-devnet:~$ python3 -m venv ./python-lab-venv
devnet-adm@ubuntu-devnet:~$ source ./python-lab-venv/bin/activate
(python-lab-venv) devnet-adm@ubuntu-devnet:~$ python --version
Python 3.12.3
(python-lab-venv) devnet-adm@ubuntu-devnet:~$ cd wifi-automation
(python-lab-venv) devnet-adm@ubuntu-devnet:~/wifi-automation$
```

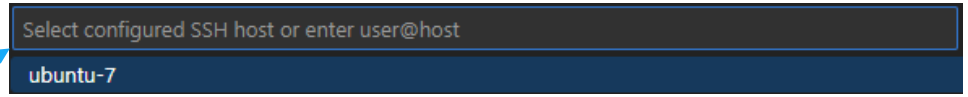


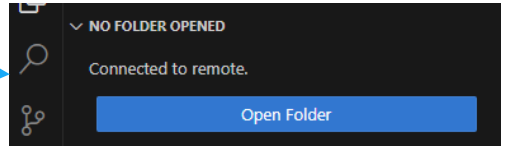
# Connect VS Code to the Ubuntu Server

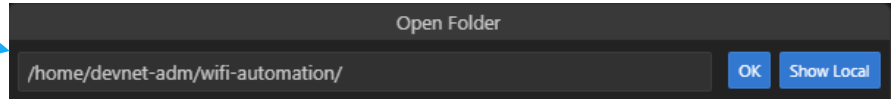
- Use VS Code remote to the Ubuntu Server (if you are not already connected already)
  - Press the Remote SSH icon (lower left corner)
  - Then, in the top line, select "Connect current window to Host..."
  - Select your host, or configure a new (see earlier slides)
  - Select "Open Folder" in the Explorer section
  - Select "/home/devnet-adm/wifi-automation/" and click OK
  - Select the "python-lab-venv" venv:
    Click the venv area on the bottom line
    (you may have to open a .py file for this to show)

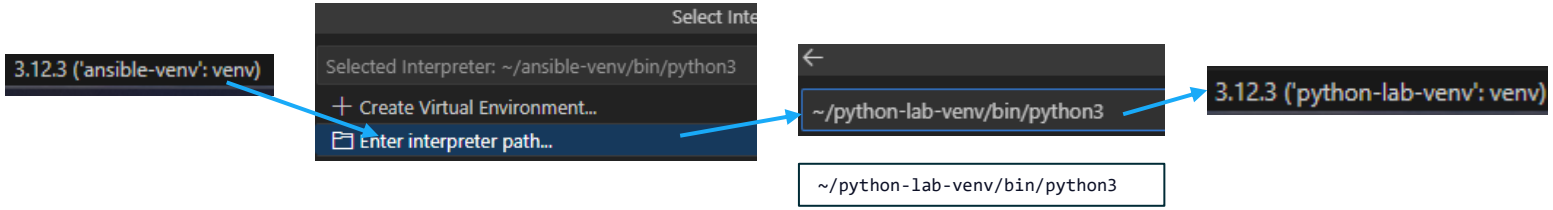


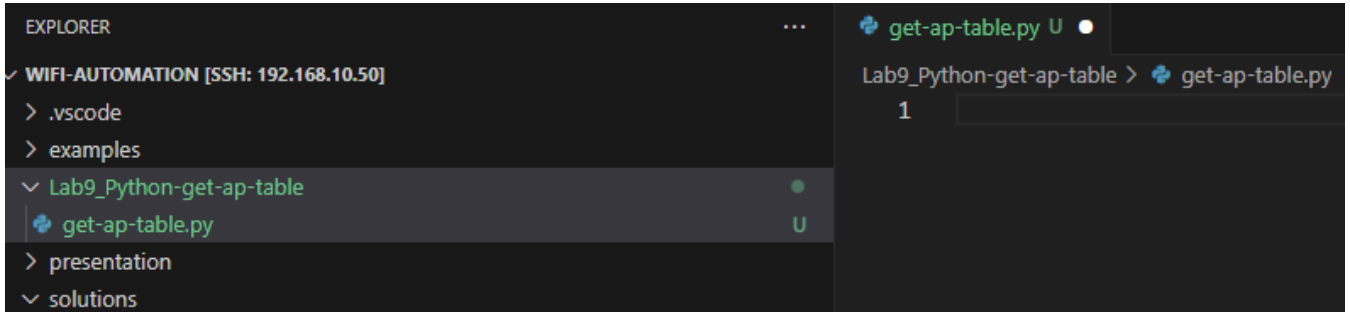



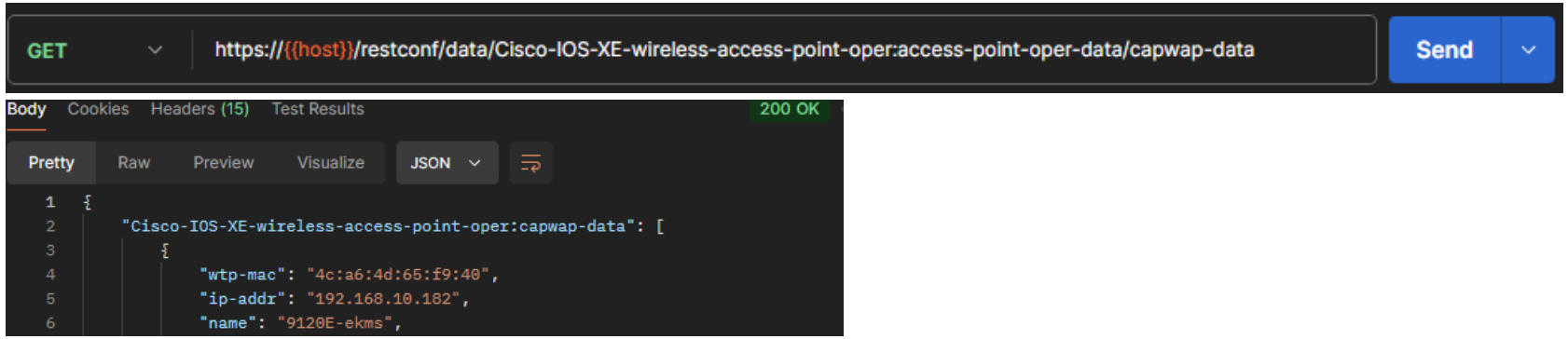
# Create the Python file

- 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`




  - 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`

- In the next exercise we will install and use YANG Suite, where you can explore YANG models to find out URLs like this for "everything"

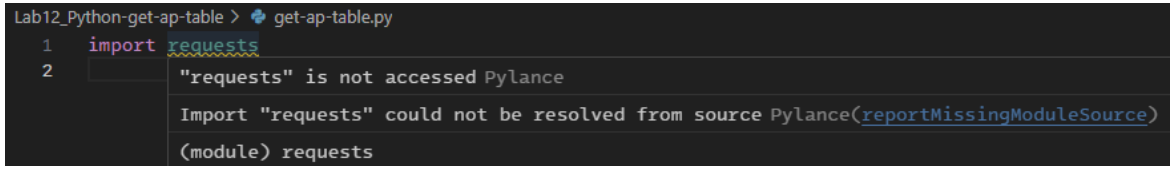


# Prepare Python packages

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

```
(python-lab-venv) devnet-adm@ubuntu-devnet:~/wifi-automation$ cd Lab9_Python-get-ap-table
(python-lab-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
(python-lab-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






# Get AP table

```python
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-poin

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

response = requests.get(url, auth=(user, password), headers=headers,

if (response.status_code==200):
    ap_table = pd.json_normalize(response.json()['Cisco-IOS-XE-wirele
    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}")
```

- 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/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"

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:capwap-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

```
(python-lab-venv) devnet-adm@ubuntu-devnet:~/wifi-automation$ cd Lab9_Python-get-ap-table
(python-lab-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
(python-lab-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
```

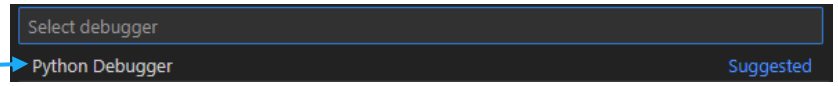
- Wrong username/password will look something like this:

```
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-automat
ion/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 (downloading the XLSX is not covered in this lab

| | 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 | nfo.board | info.board | atic-info.a | nfo.tag-so | tag-info.pt | te-tag.site | site-tag.a | site-tag.fle | rf-tag.rf-ta | lter-info.fi | e.ap-admi | ap-operat | e-info.boo | e-info.join-time | | |
| 0 | | 4c:a6:4d:6 | 9120E-ekn | 192.168.1( | FGL2417LS | a4:b4:39:2 | C9120AXE | tag-source | k32-stand | k32 | koksrud-a | koksrud-fl | k32-rf | Staging-re | adminstat | registerec | 2024-07-2 | 2024-07-25T00:30:04.335182 | | |
| 1 | | d4:2c:44:d | APD42C-4 | 192.168.1( | KWC2038( | d4:2c:44:d | AIR-AP183 | tag-source | Empty | k32 | koksrud-a | koksrud-fl | k32-rf | Empty | adminstat | registerec | 2024-07-3 | 2024-07-31T19:23:42.643078 | | |

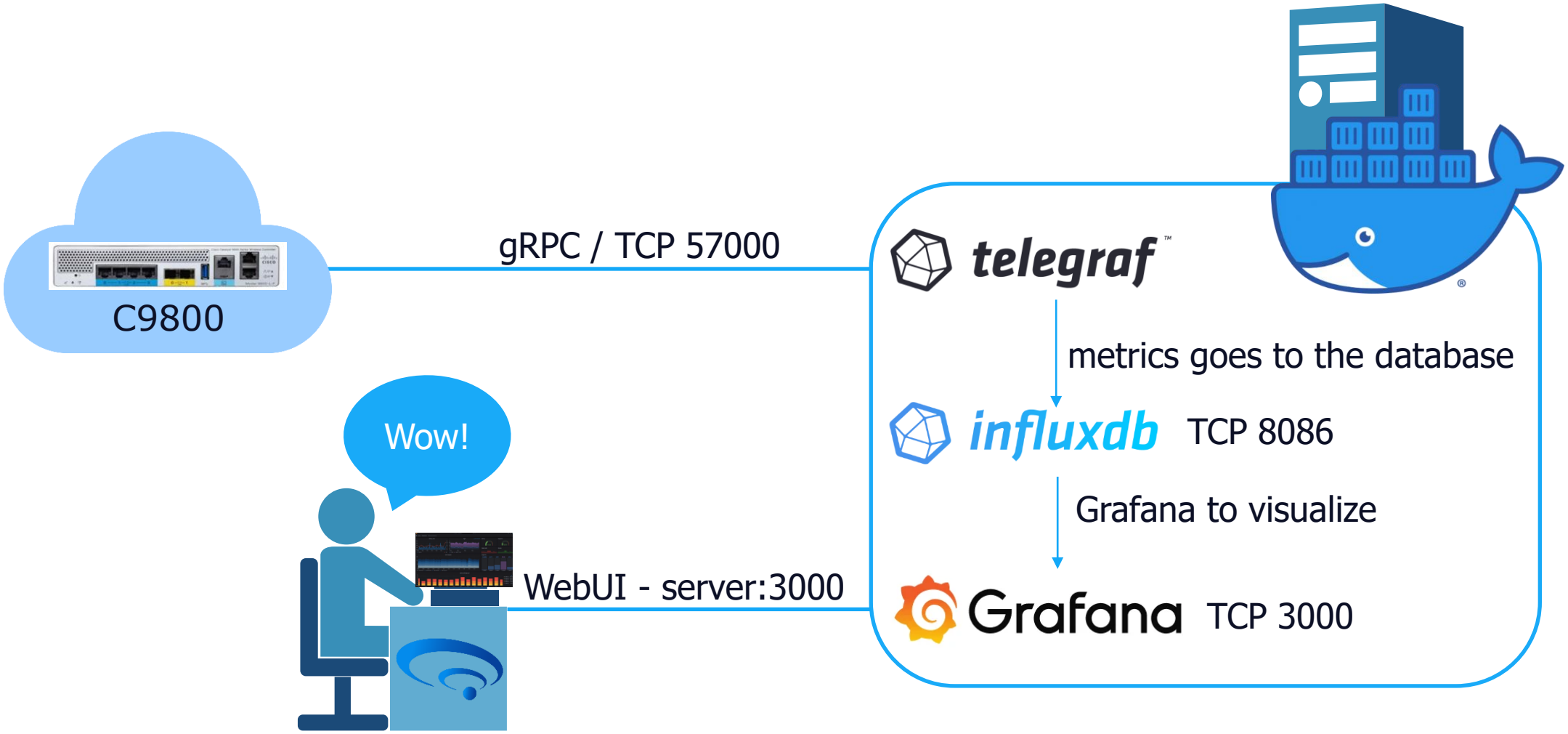
- You can open the CSV file in VS Code, or in the terminal using "cat ap_table.csv" (or several alternatives)

# 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 to 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: Build your first Grafana dashboard just to get a small touch of the Grafana-magic ☺
- 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



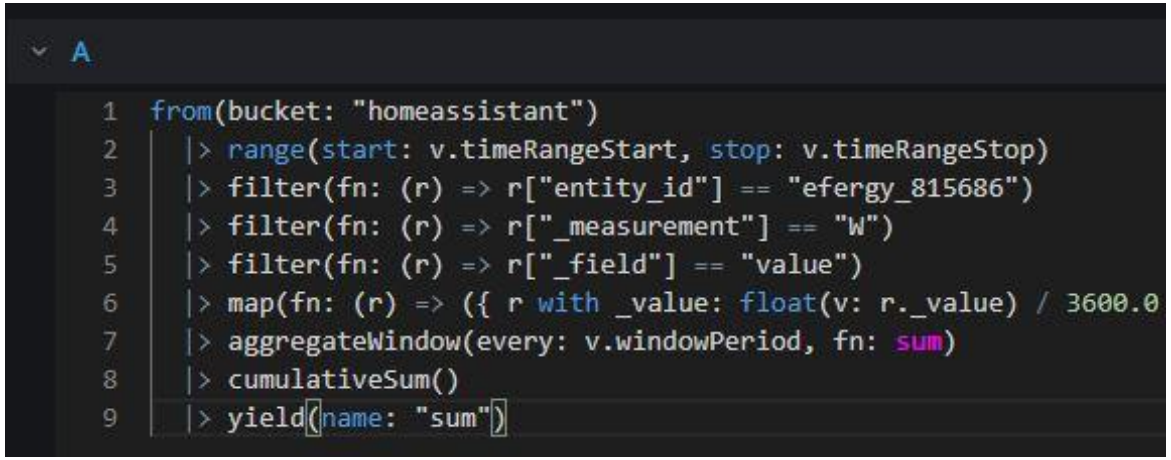

- **Open Source Agent:** Collects and sends metrics from databases, systems, and IoT devices
- **Versatile Monitoring:** Works with databases, systems, and IoT sensors

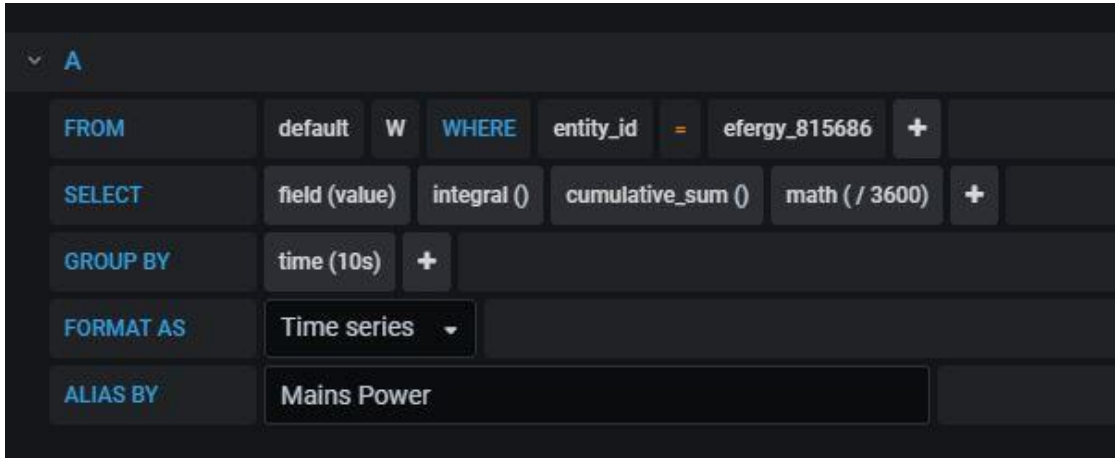
- **Time Series Data at Scale:** Store and query large volumes of time-stamped data
- **Real Time Analytics:** Fast processing for immediate insights
- **Seamless Integration:** Compatible with various tools and platforms

- **Visualize Anything:** Create dashboards for any data
- **Real Time Insights:** Customizable, interactive visuals
- **Flexible Integration:** Connects to multiple data sources

# Different flavours of Influx

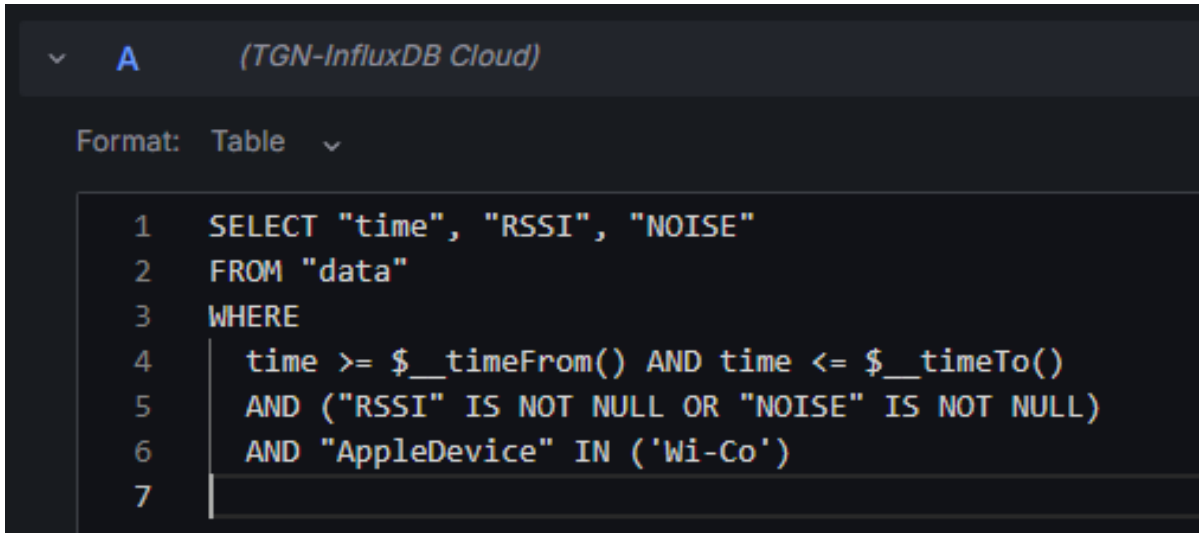
InfluxDB 2.x - FLUX




InfluxDB 1.x - InfluxQL

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




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}
Grafana token: {we will fill in this later}
Optional Telegraf 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=ChangeMe2024!
INFLUXDB_ORG=devnet
INFLUXDB_BUCKET=c9800-bucket
INFLUXDB_ADMIN_TOKEN=pleasechangeme2024
GF_SECURITY_ADMIN_USER=devnet-adm
GF_SECURITY_ADMIN_PASSWORD=ChangeMe2024!
```

   - Add the INFLUXDB_ADMIN_TOKEN to your .txt file

Lab10_notes.txt

```
1 Admin token: pleasechangeme2024
```

2. Create persistant 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
```

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 ☺

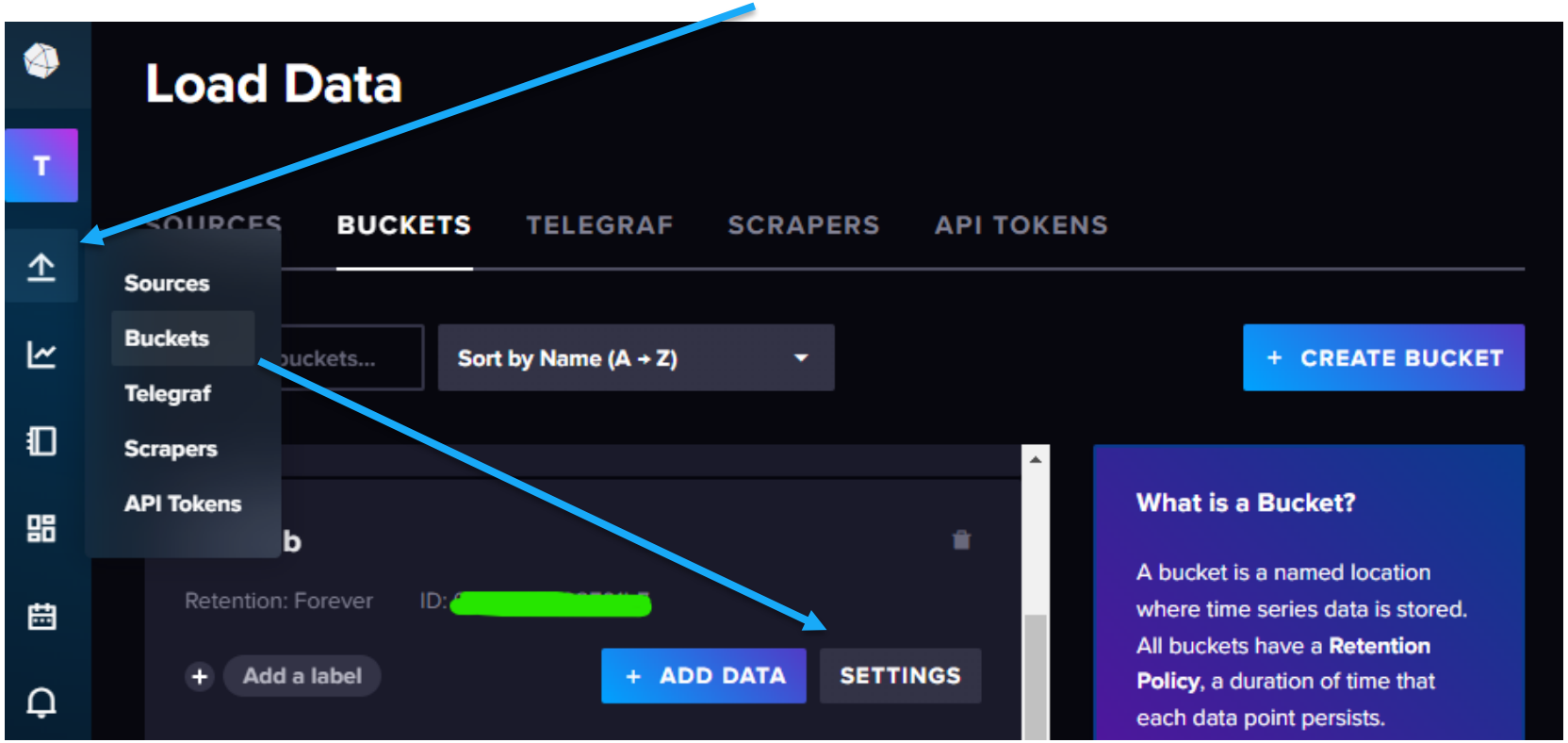
`docker-compose.yml`

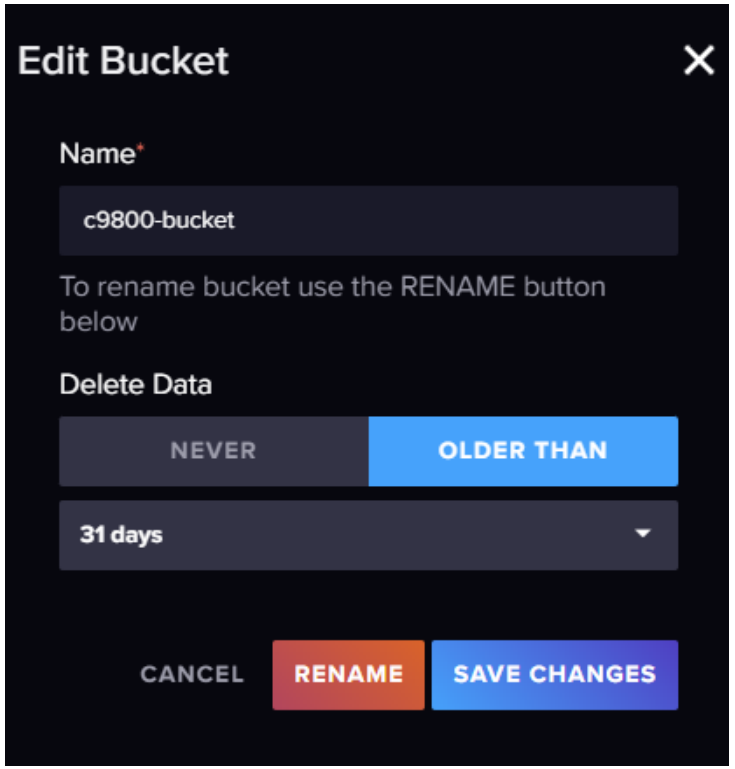
```
services:
  influxdb:
    container_name: influxdb
    image: influxdb:2.7.10
    expose:
      - 8086
    restart: always
    volumes:
      - influxdb-data:/var/lib/influxdb2
    ports:
      - 8086:8086
    environment:
      - DOCKER_INFLUXDB_INIT_MODE=setup
      - DOCKER_INFLUXDB_INIT_USERNAME=${INFLUXDB_ADMIN_USER}
      - DOCKER_INFLUXDB_INIT_PASSWORD=${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.

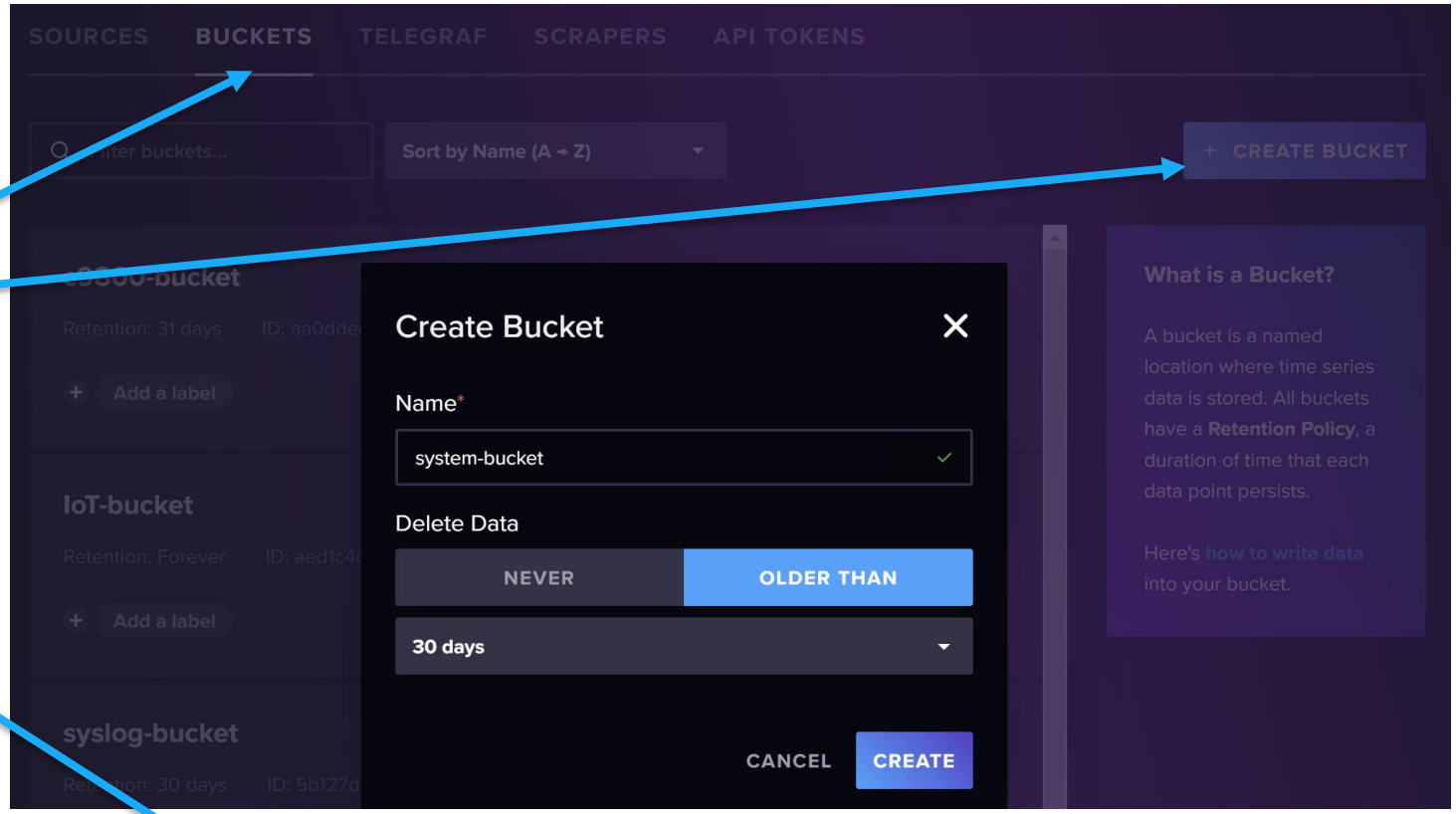


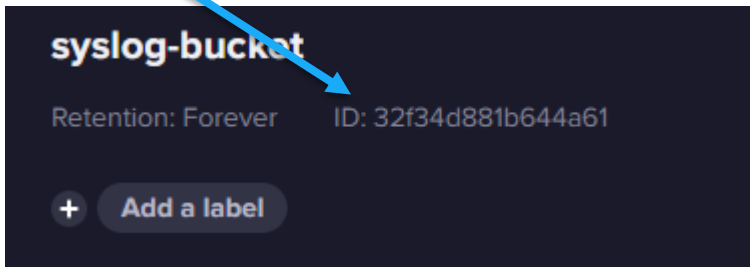



# InfluxDB 2.X

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



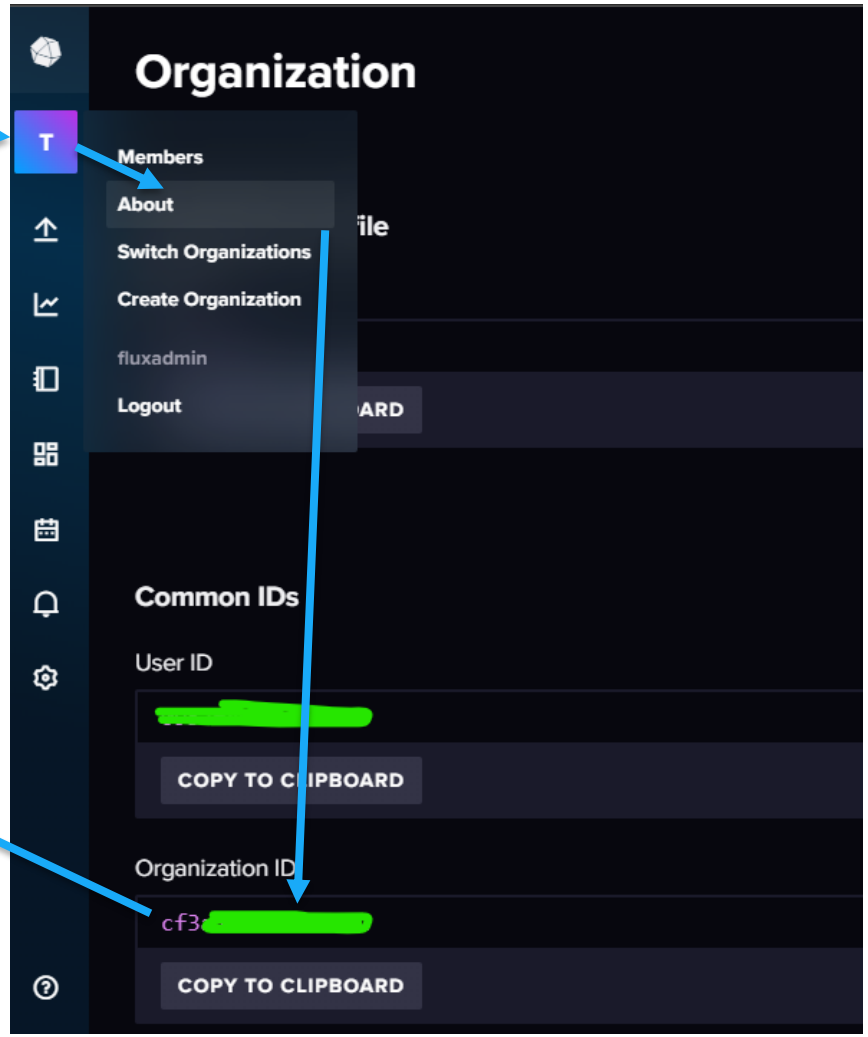



# Locate your Organization ID

- InfluxDB > About
- Note this down in the Lab10_notes.txt file

Lab10_notes.txt

```
1  Admin token: gmh03nwe0ng3t28y3y734yhw
2  Grafana token: 4T4uS45qjduLofzU5PHl7sgldFNg
3  Optional Telegraf token: 397uthg9u2qg5hr9g2t12
4  Org ID: cf312345678901234
5  Container ID: {we will fill 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: gmh03nwe0ng3t28y3y734yhw
2  Grafana token: 4T4uS45qjduLofzU5PHI7sgldF
3  Optional Telegraf token: 397uthg9u2qg5hr9g2t
4  Org ID: cf312345678901234
5  Container ID: {we will fill in this later}
```

- In the notes, you will then have
  - Admin token
  - Grafana token
  - Telegraf token (optional)
  - Org ID
  - Container IDs

```
Lab10_notes.txt
1  Admin token: gmh03nwe0ng3t28y3y734yhw
2  Grafana token: 4T4uS45qjduLofzU5PHI7sgldFNguxx
3  Optional Telegraf token: 397uthg9u2qg5hr9g2t125
4  Org ID: cf312345678901234
5  Container ID: 71b07ba58958
```

# 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«

Insert YOUR container-id

Insert YOUR org-id

Insert YOUR admin token

```
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$
```

- 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"

Repeat this step for the syslog-bucket

# 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-kvgpb
 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-kvgpb
 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 ☺

- !!! Note !!!
  - If you are doing this lab exercise using TIG Stack on a shared server
    - Have a word with one of the instructors if there is any neccessary coordination or steps that differ
    - Replace "Your Ubuntu IP" with the respective IP of the shared server you are using
    - If you have your own WLC, do the next slide (WLC telemetry subsctiptions), then skip to slide 95 (Create your first Dashboard)
    - If you are using shared WLC, skip to slide 95 (Create your first Dashboard)

# Add the following telemetry subscriptions on the WLC

```
wlc9# conf t
```

```
telemetry ietf subscription 105
 encoding encode-kvgpb
 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)

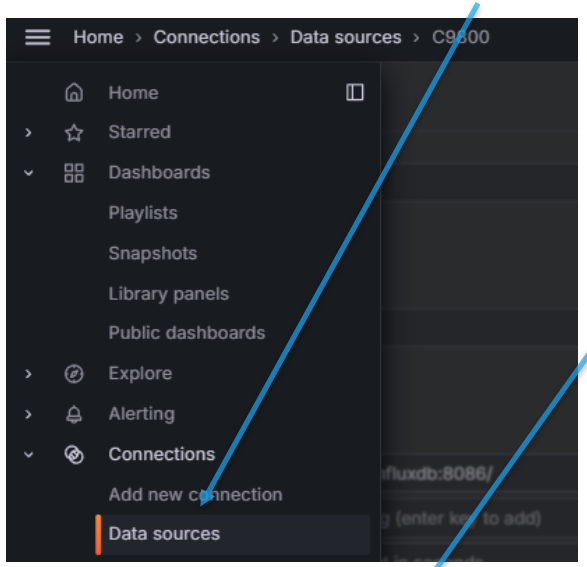
```
telemetry ietf subscription 106
 encoding encode-kvgpb
 filter xpath /wireless-rrm-oper:rrm-oper-data/wireless-rrm-oper:rrm-measurement/wireless-rrm-oper:load
 stream yang-push
 update-policy periodic 5000
 receiver ip address 192.168.10.7 57000 protocol grpc-tcp
 do show telemetry ietf subscription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

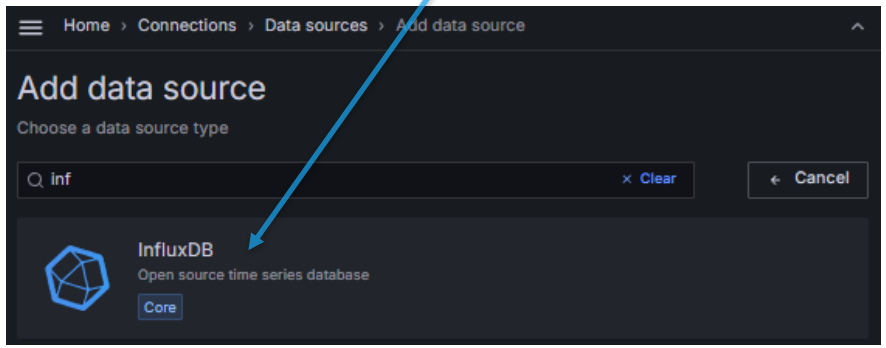
Change this to your Ubuntu IP (or the IP of the TIG stack you are using)

```
telemetry ietf subscription 107
 encoding encode-kvgpb
 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
```

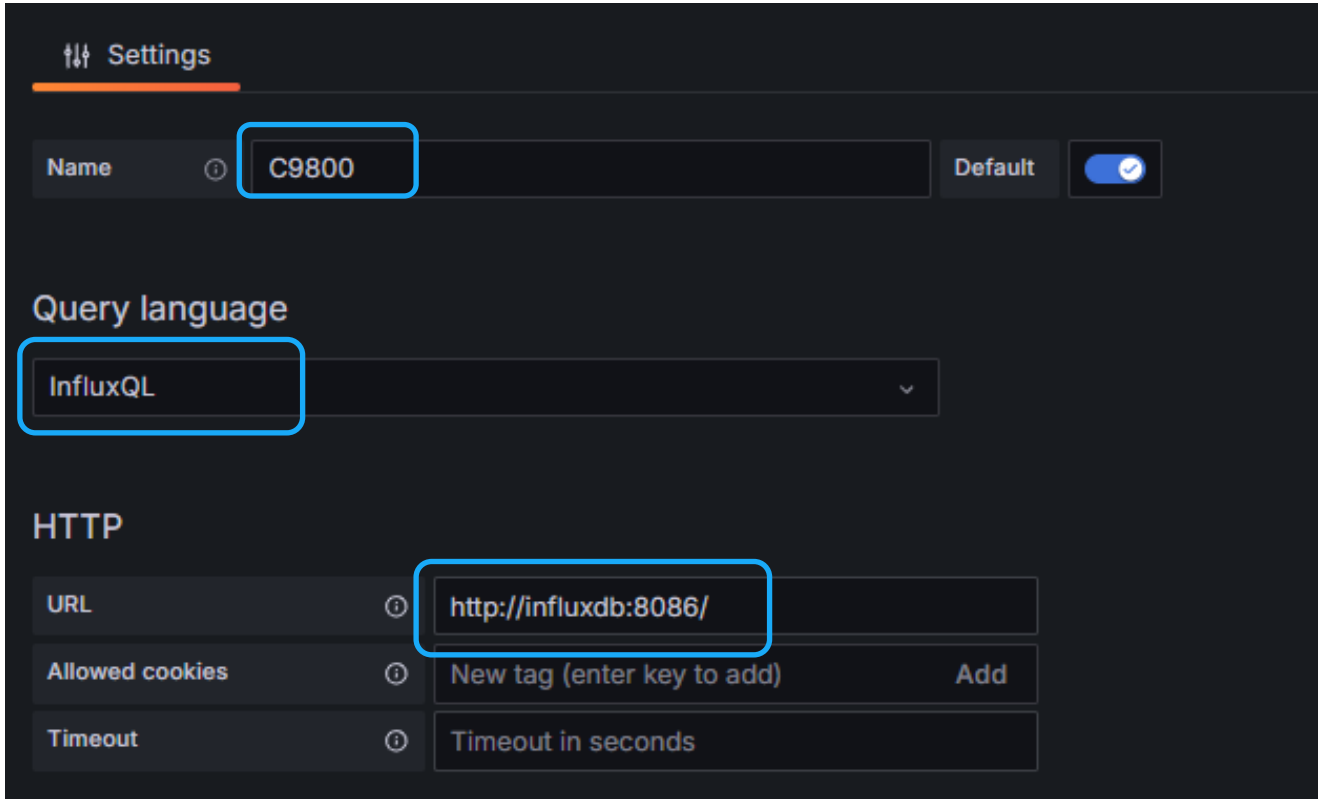
# Grafana (http://{Ubuntu-IP}:3000)

- Add your first data source -> InfluxDB



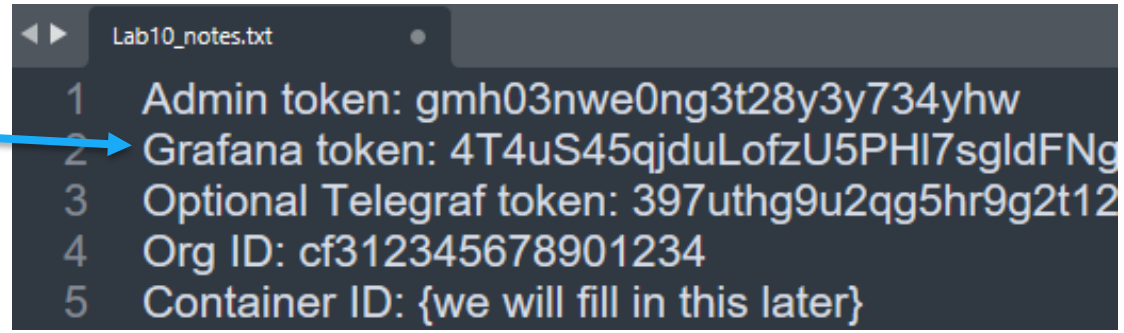




- Set a name for your database
- Set Query language to InfluxQL
- URL: http://influxdb:8086

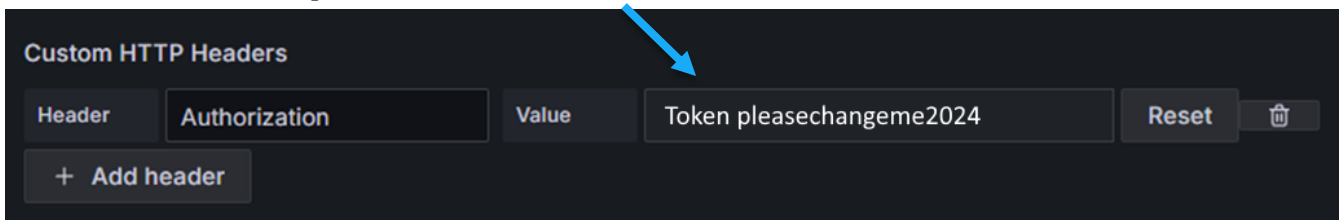


# Grafana (http://{Ubuntu-IP}:3000)

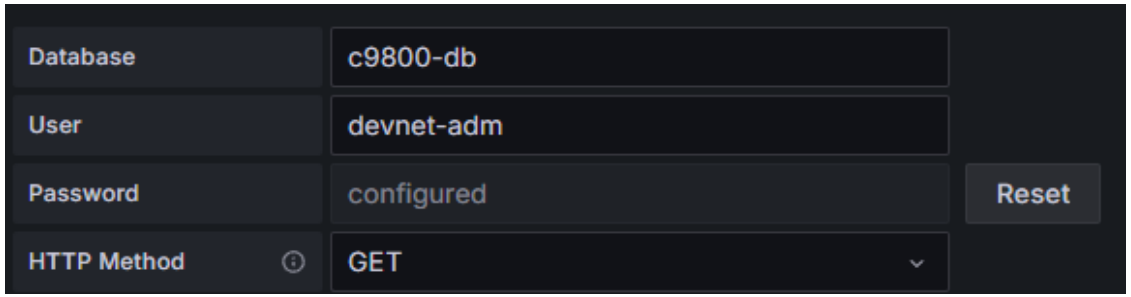
- Grafana needs to use a RO token to be able to read from the database, and InfluxDB password.
  - Use the Grafana token from Lab10 notes



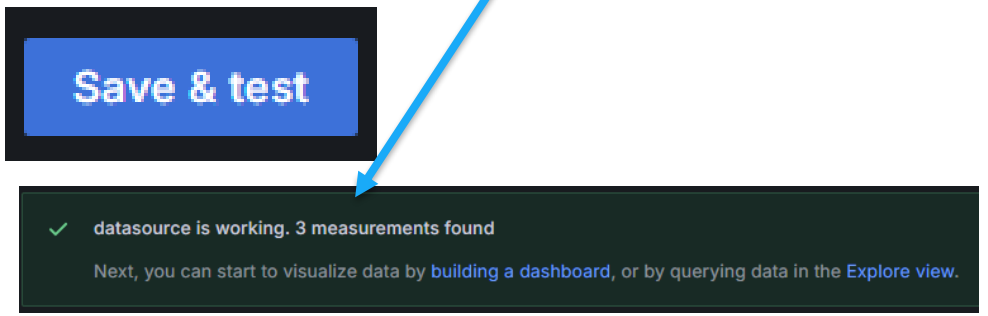

- Add a custom HTTP Headers
  - Header: Authorization
  - Value: **Token <API-TOKEN>**
  - **There is a space between Token and <token>**




  - Database: c9800-db
    - User/Pass: devnet-adm / ChangeMe2024!

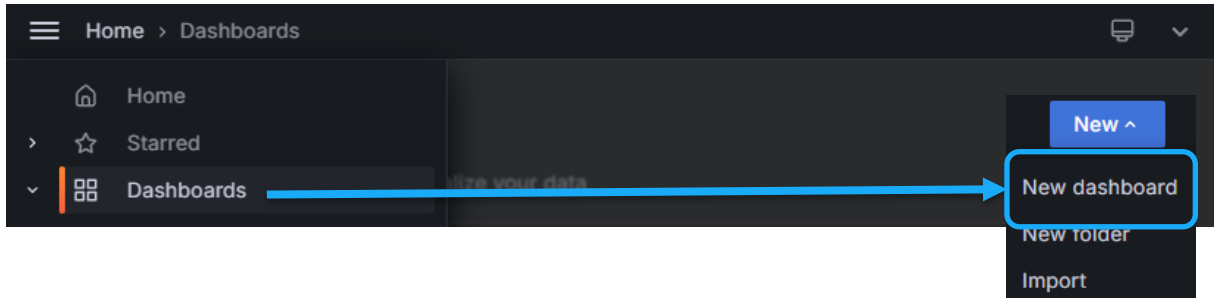


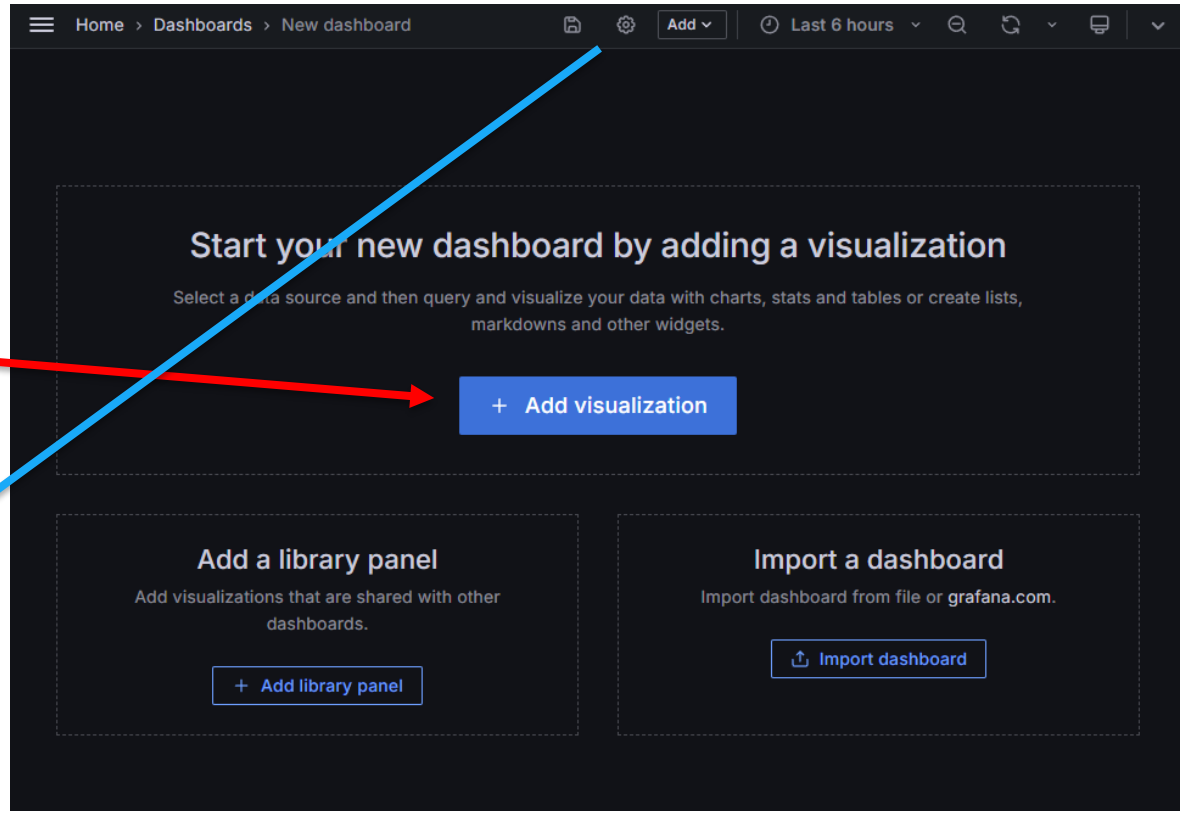

!!! You should have 3 measurements found !!!



# Create your first dashboard

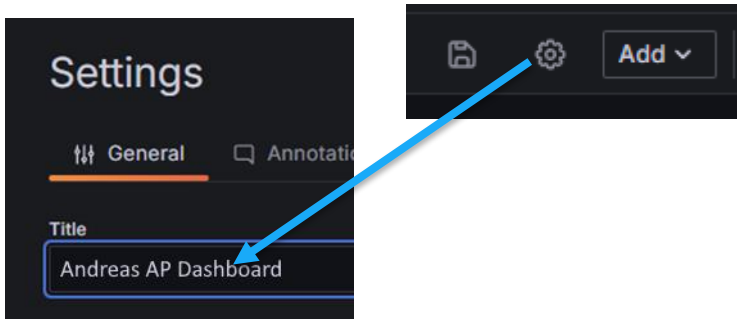
- Grafana (http://{Ubuntu-IP}:3000)



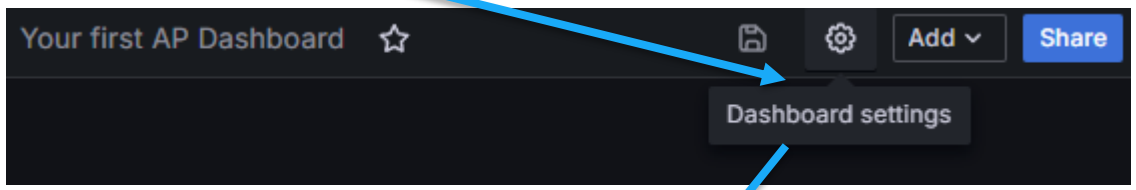

- !!! Don't press the very tempting "+Add Visualization" button yet, we will do some pre-work first ☺

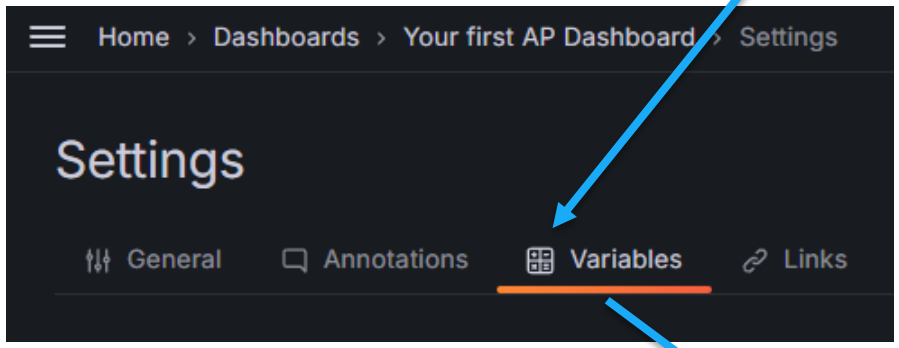
- Start by giving the new dashboard a name

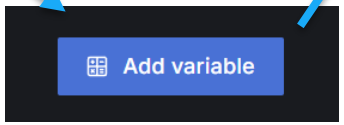


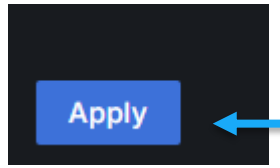
# Grafana

- Add variables



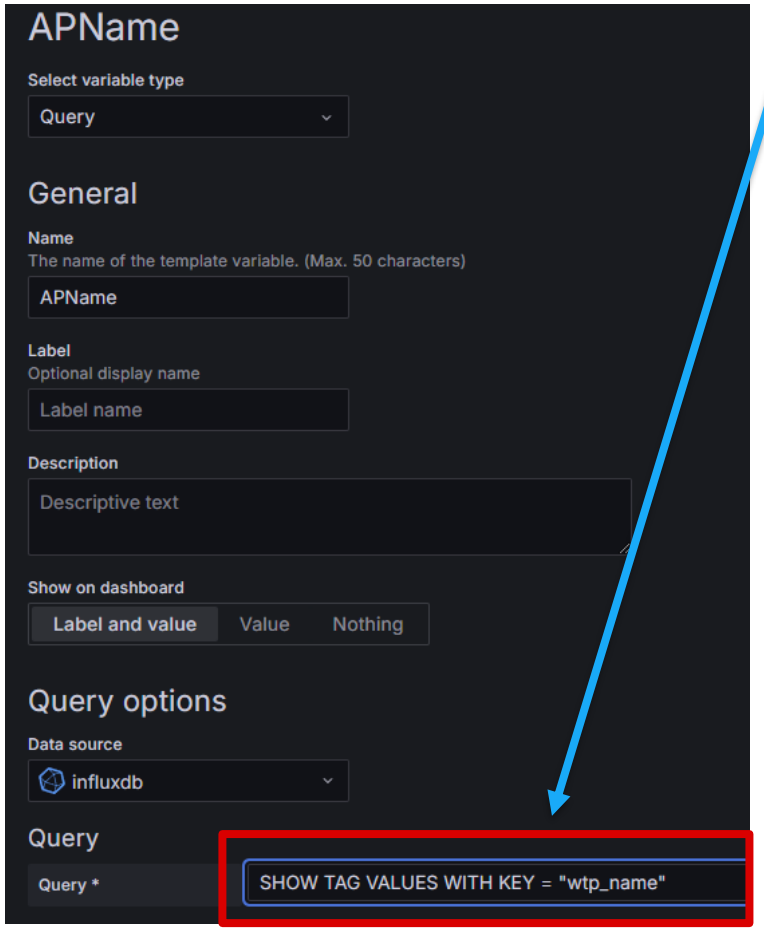




- Name: APName
- Query:

```
SHOW TAG VALUES WITH KEY = "wtp_name"
```



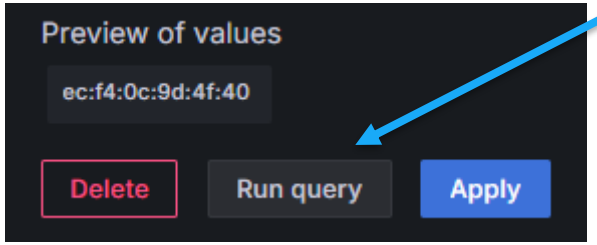
# Grafana

Create a new variable

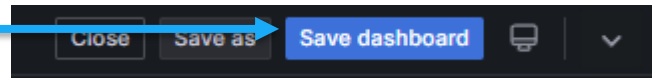
- Name: APMac
- Show on dashboard: Nothing
- Query:

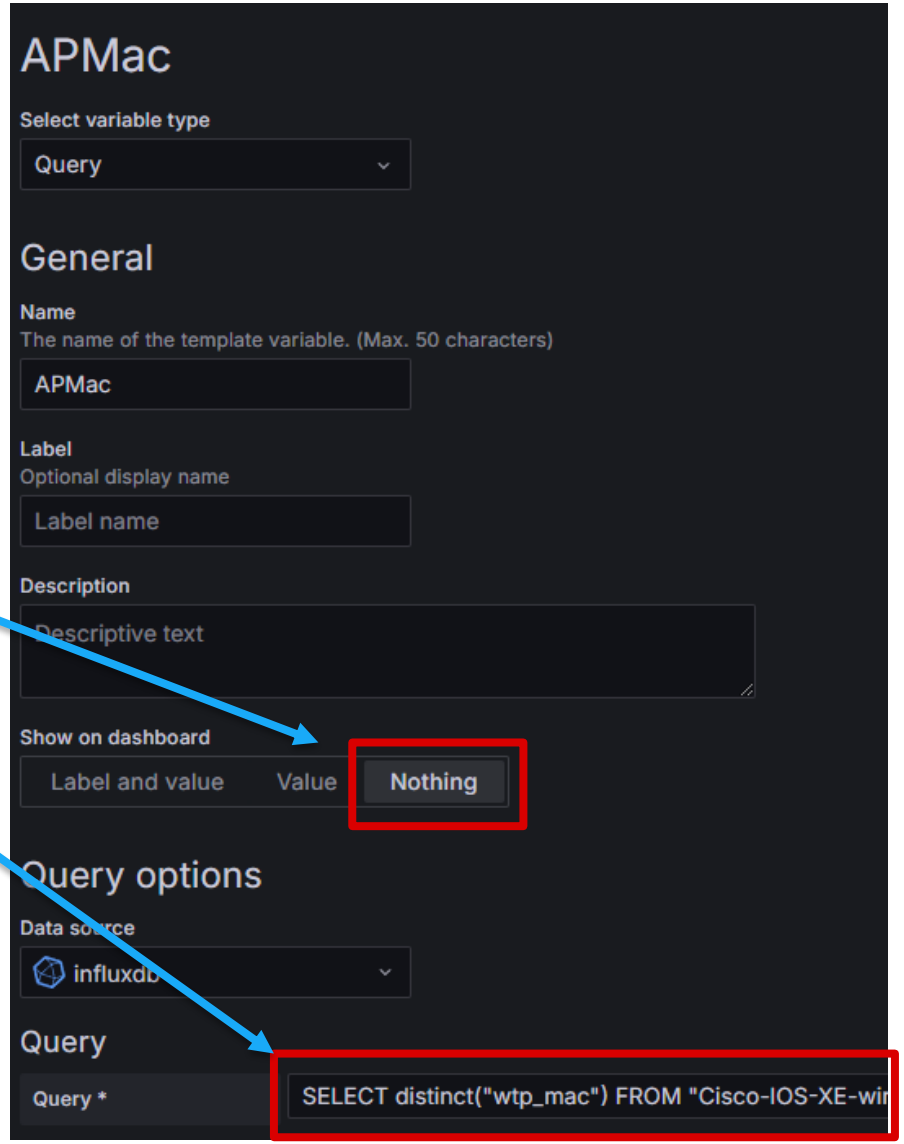
```
SELECT distinct("wtp_mac") FROM "Cisco-IOS-XE-wireless-access-point-oper:access-point-oper-data/ap-name-mac-map" WHERE ("wtp_name" =~ /^$APName$/)
```

- Check if it is working with Run query



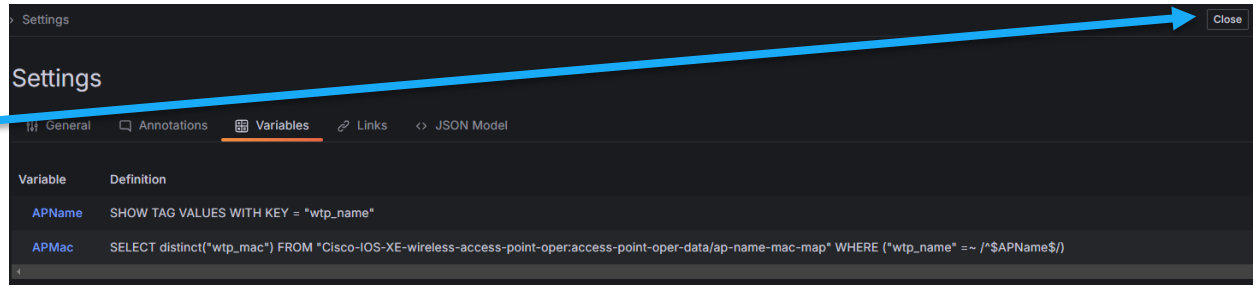

- Press Apply
- Then save your dashboard



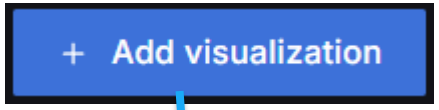



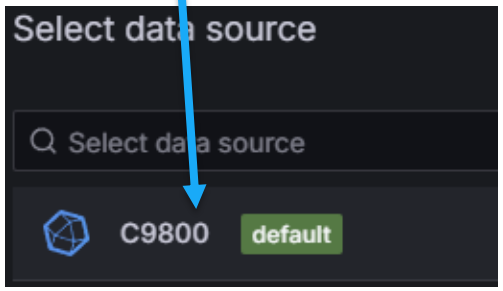
# Grafana

- Go back to the Dashboard main page

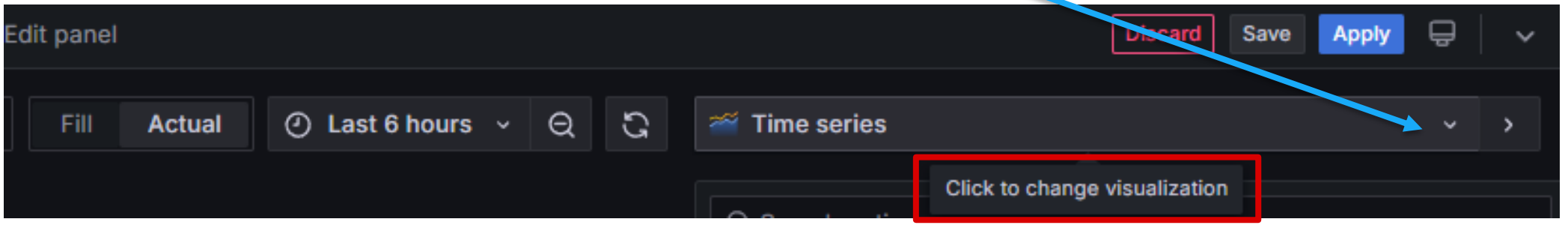



- Add visualization and make one using time series



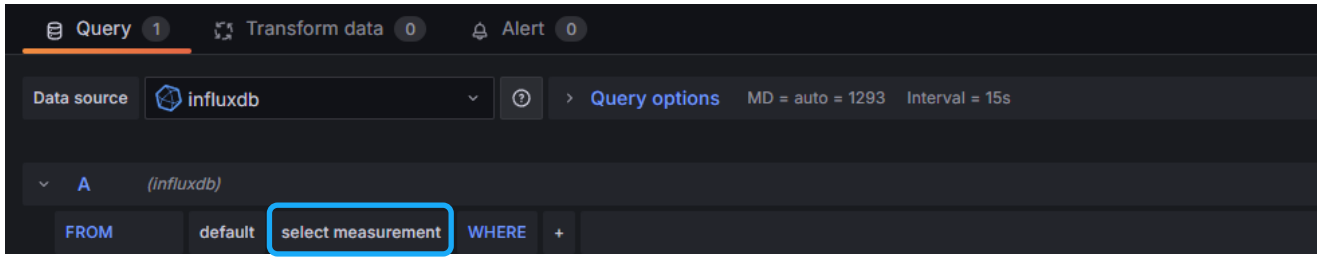




- "Time series" is automatically selected. If not, select it here

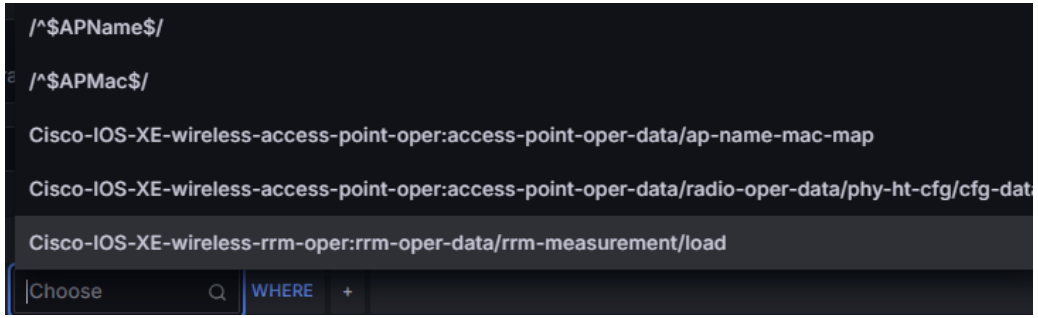


# Grafana

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

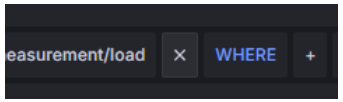


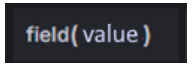

- Click "select measurement", you will get a menu of all measurements



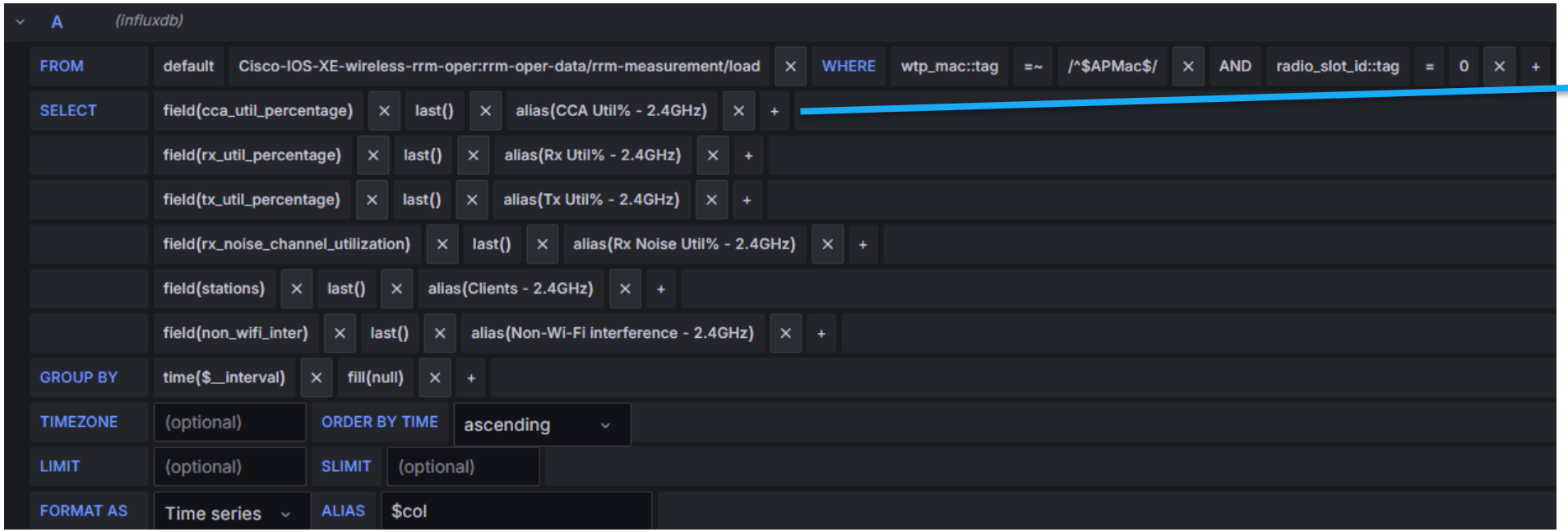

- Choose the *Cisco-IOS-XE-wireless-rrm-oper:rrm-oper-data/rrm-measurement/load*
- **(Optional)** You can enter YANG Suite, find the Cisco-IOS-XE-wireless-rrm-oper module and explore which values you could get from this YANG model

# Grafana

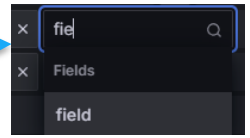
- To get the options, click the "+" signs, like here 

- You can search in the "Choose" field over the options
- To specify the value inside the "field(value)" click it 

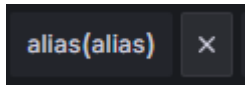
- Try to make the screen match this image




Add another row



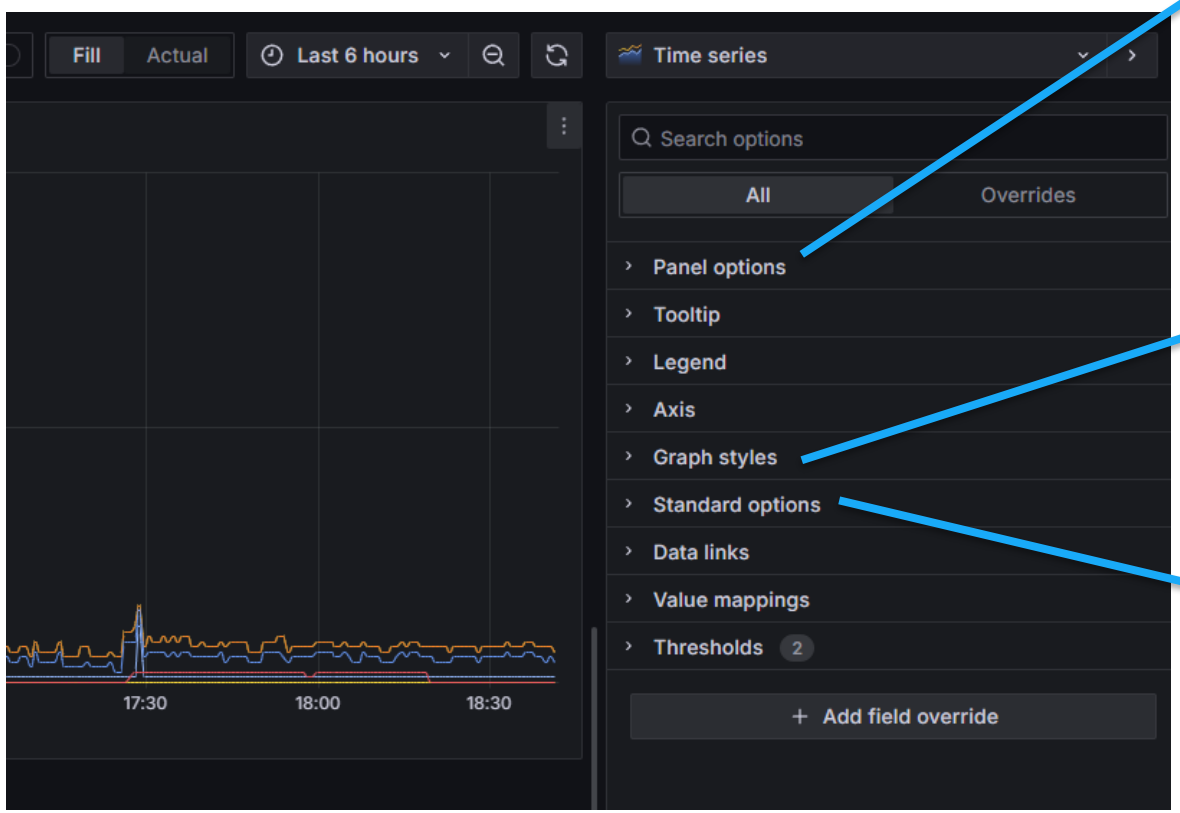

To create an alias, choose "alias" and click the alias(alias) button



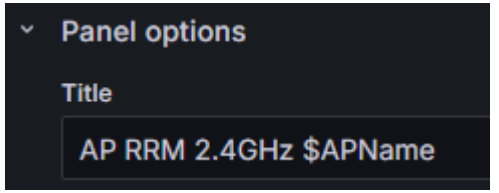

Type the name and press Enter

# Grafana

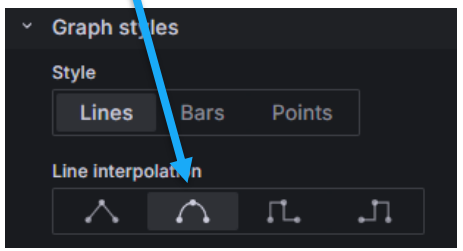
- In the field to the right, you have lots of options

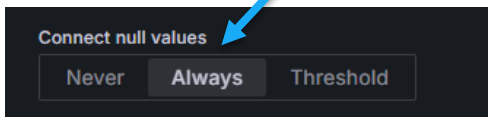



- Set the panel title

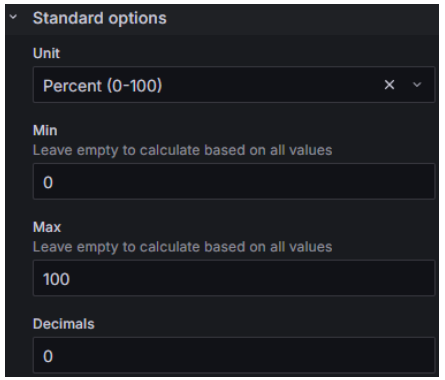



- Under graph styles you can make it look like you want ☺ Set Connect null values to always



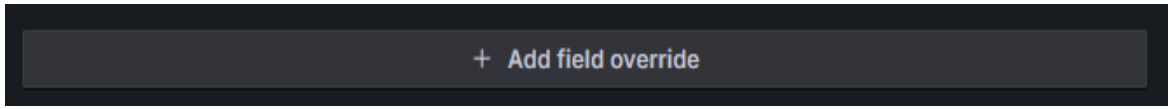




- Fix values to percent

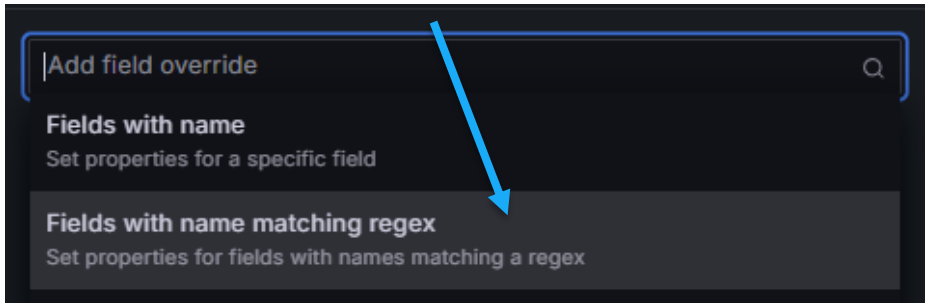


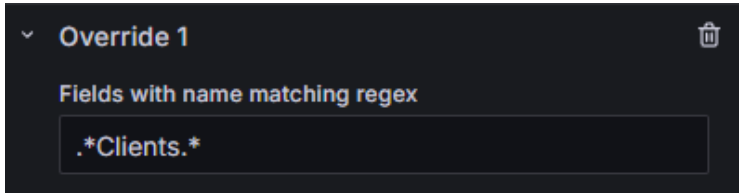
# Grafana

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

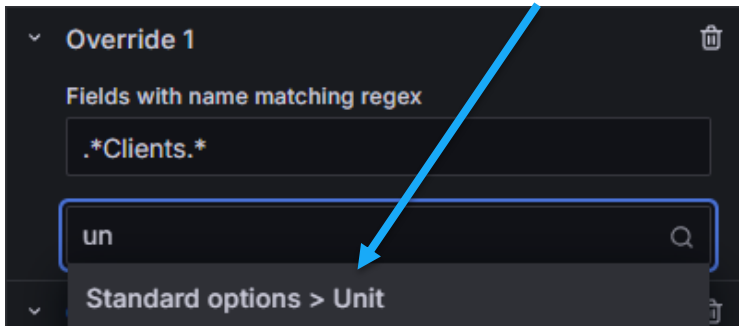




- Select the regex-override

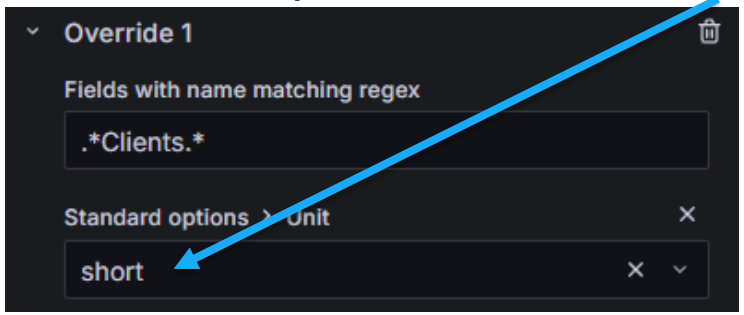



Standard options > Unit

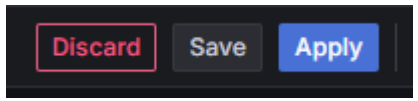



Standard options > Unit = short

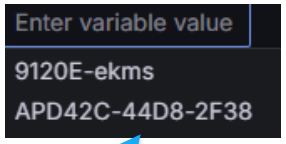


# Grafana

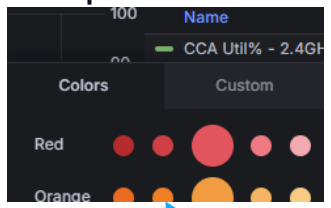
- **(Optional)** Tooltip -> Tooltip mode: All
- **(Optional)** Legend
  - Mode: Table
  - Placement: Right
  - Values: Last*, Min & Max
- **(Optional)** Thresholds
  - 35% (Red)
  - Percentage
  - As lines (dashed)
- **(Optional)** Graph styles
  - Fill opacity: 10
- Remember to apply changes



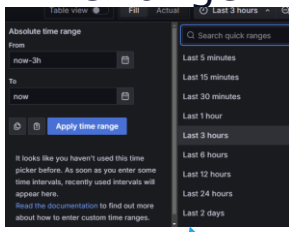

Select variable




Graph colors

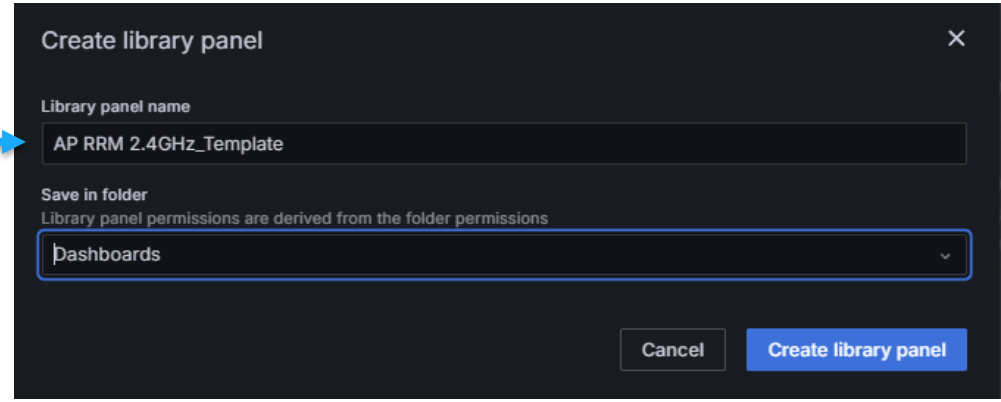


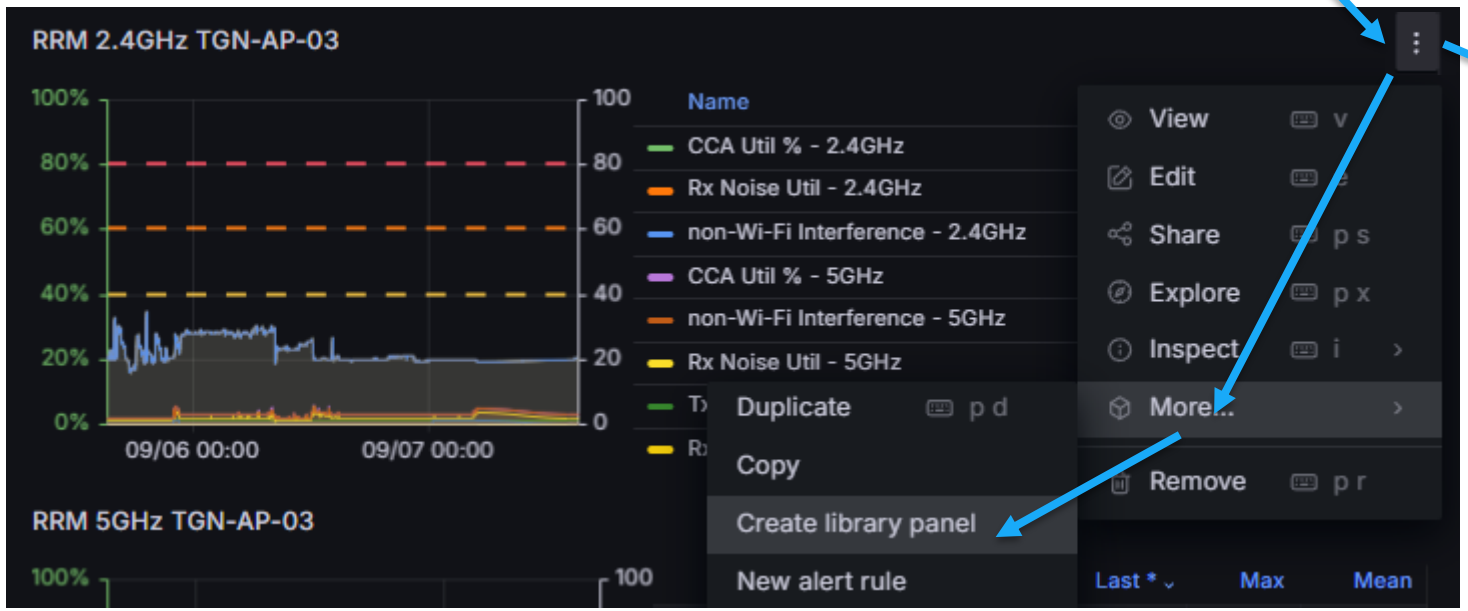

Time range



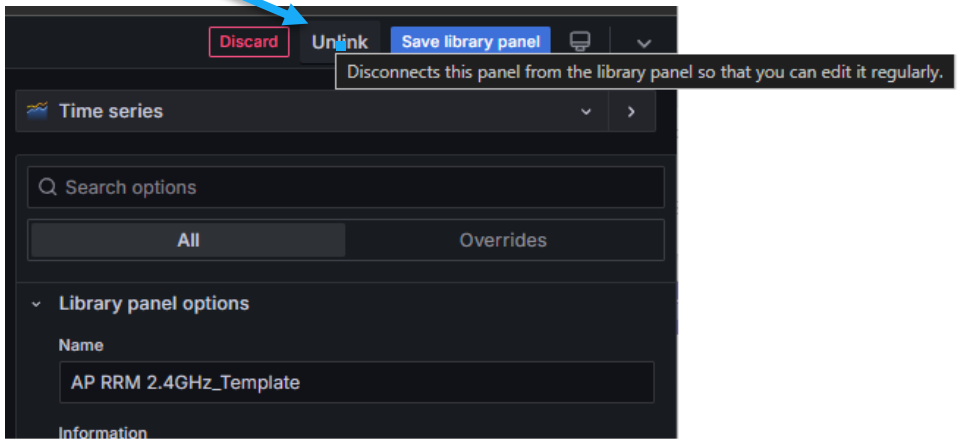
# Grafana

- If you want to save your work as a «template»
  - Great when you make new dashboards.
- **WARNING!**
  - Unlink the panel because from the template (#2)



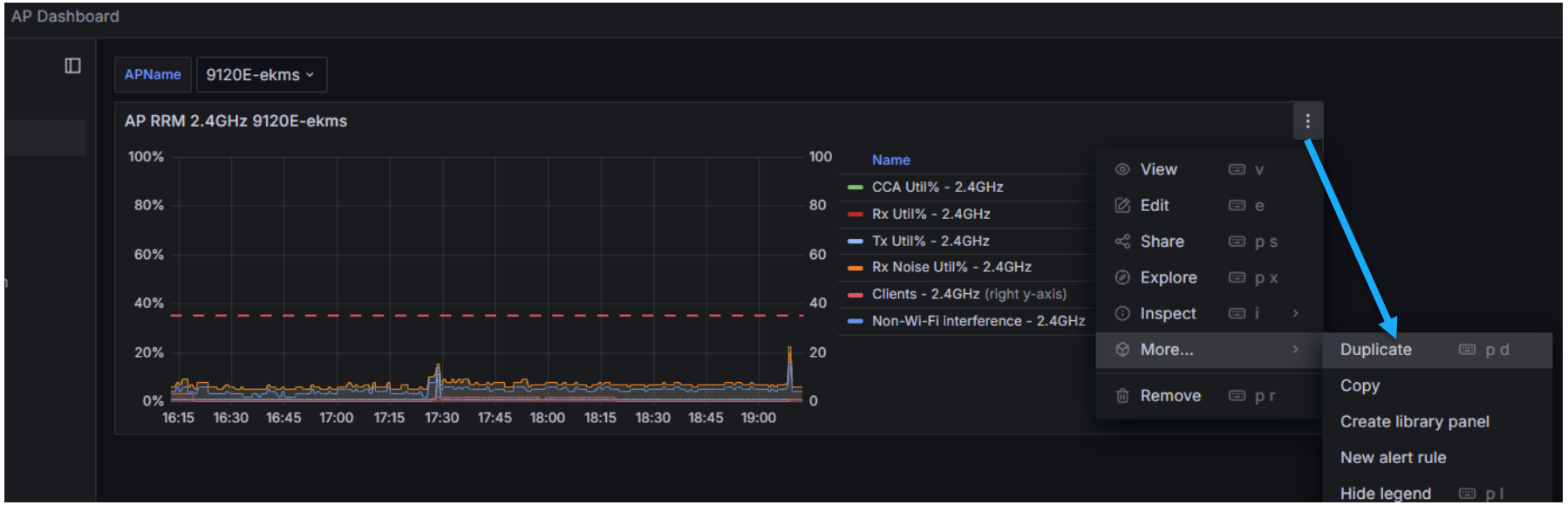




#2

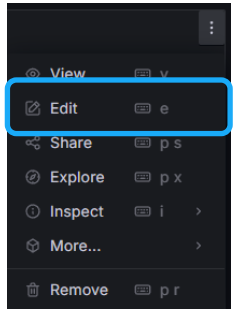


# Duplicate dashboards

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

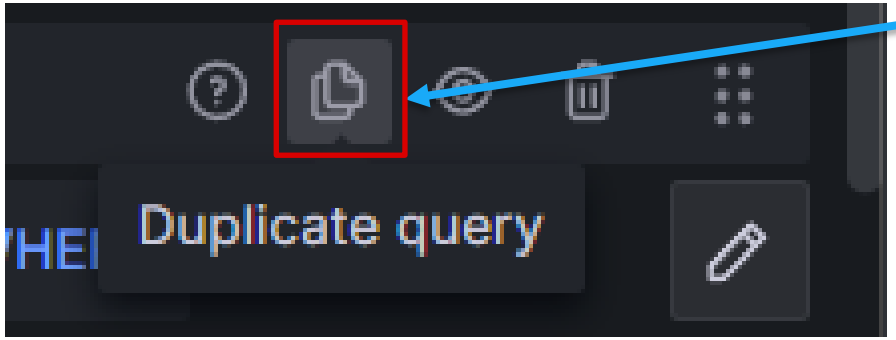



- Edit the duplicated visualization

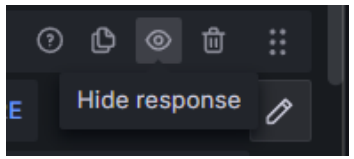


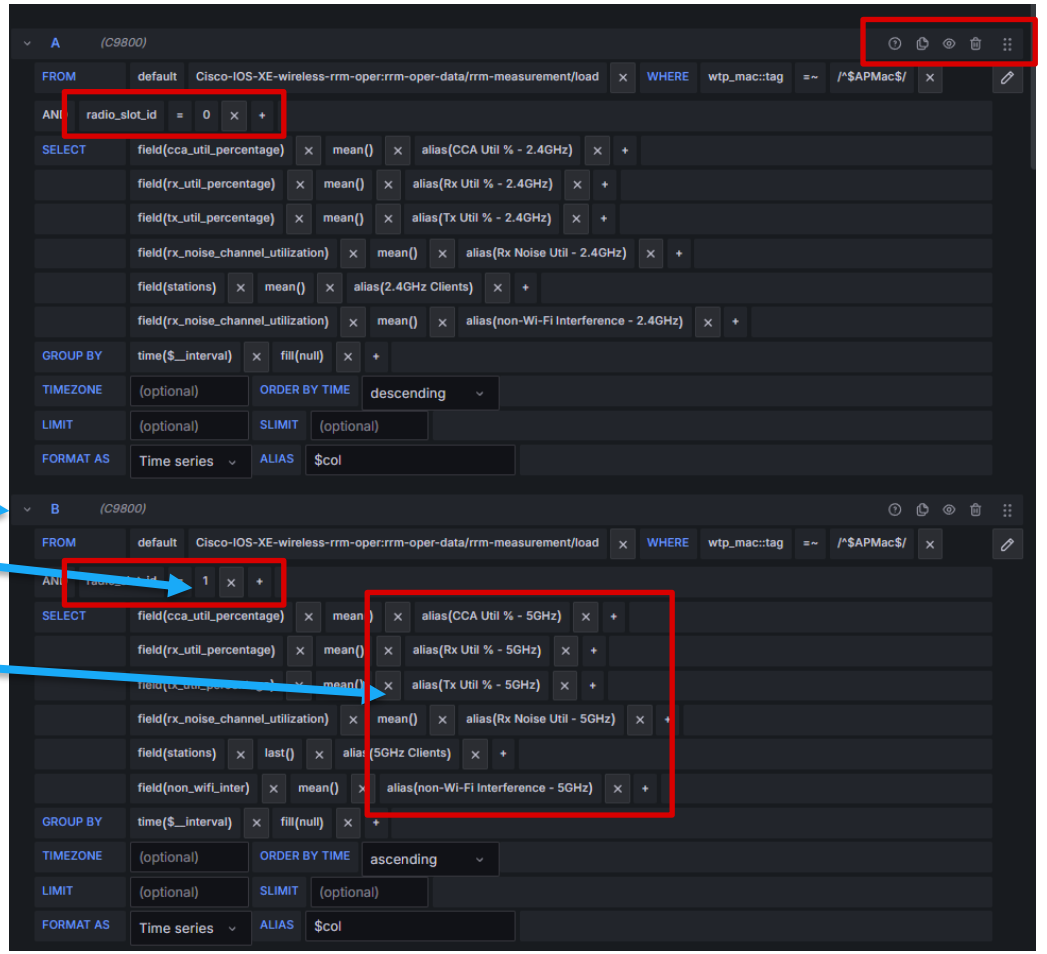
# Or add everything in one panel?

- You can add everything in one huge panel by duplicate the query inside the pane
  - Useful when you need multiple variables in a single panel



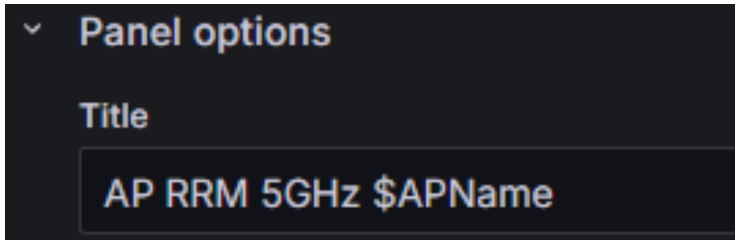

- Now we have query B for 5GHz
  - Change radio-slot to 1 (or 2 if it is dual-5GHz)
  - Rename alias to 5GHz
- Repeat for 6GHz
- Hide queries with the eye.



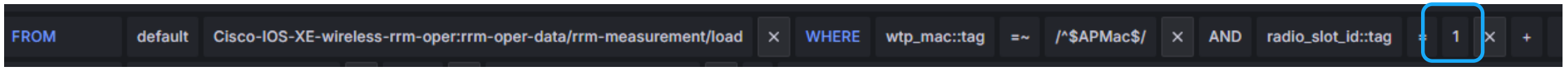



# Edit the values for the 5GHz visualization

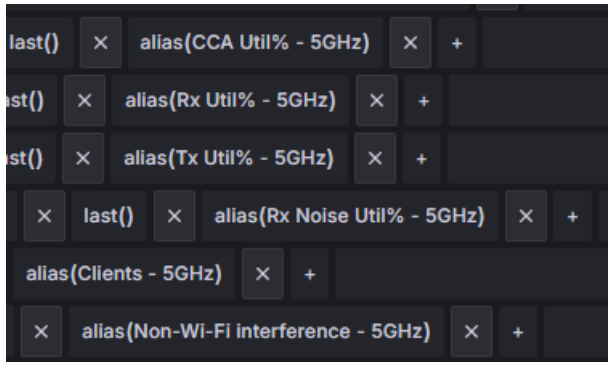
- Panel Options: Title



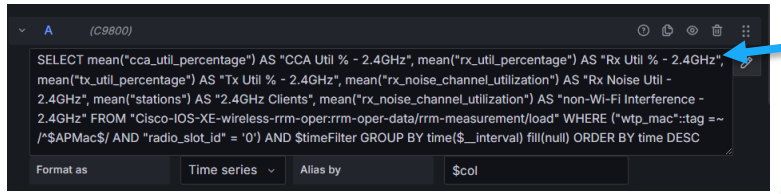

- FROM in Query: radio_slot_id::tag =1
  - If the AP is dual 5GHz (for example 9136, 9166) 5GHz is at 1 and 2



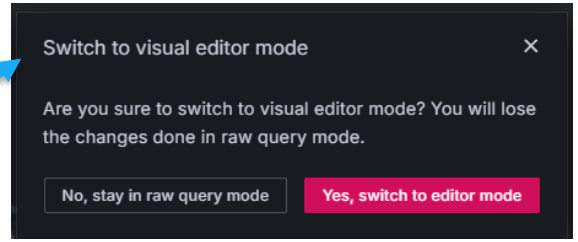

- alias fields for all SELECT statements




- You can edit the query manually by pressing the pencil

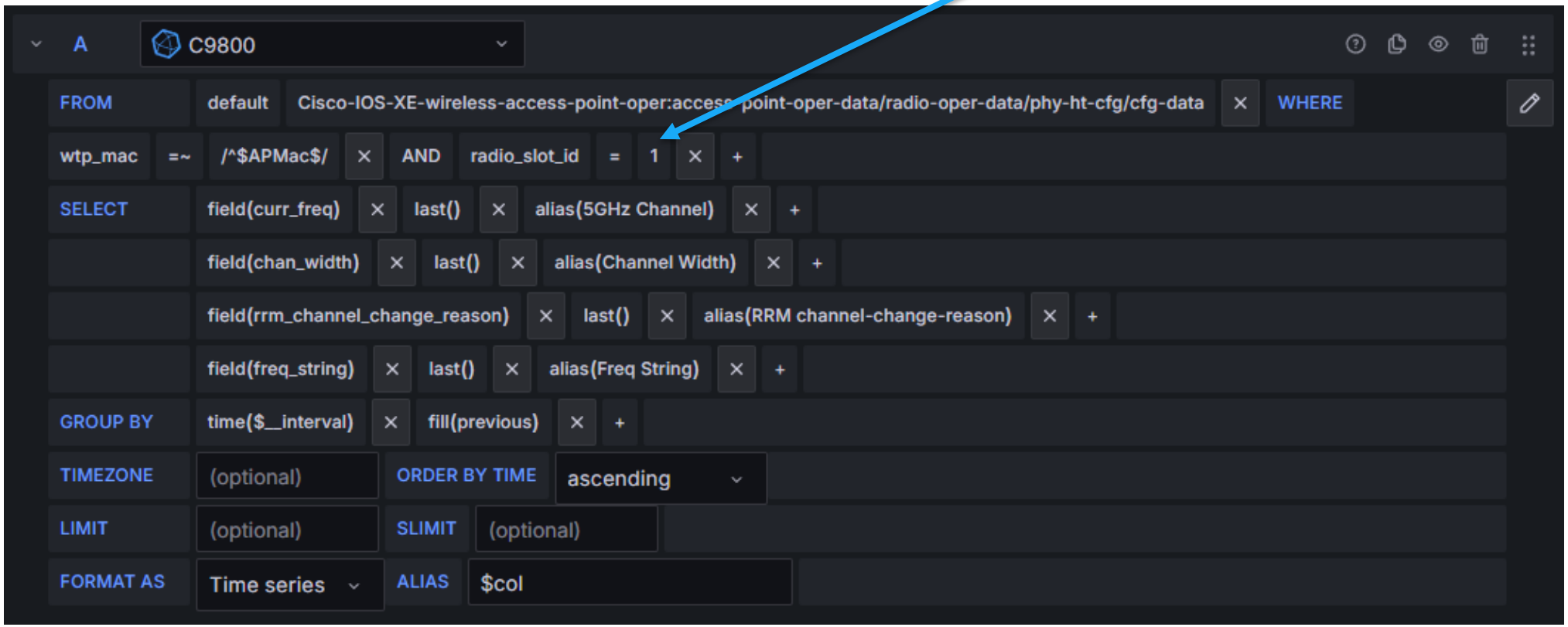



  - If you edit query manually you cannot go back to edit visual editor mode



# Grafana

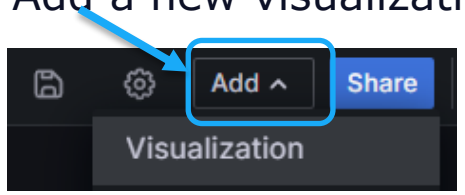
- Add the following data for 2.4, 5 and 6GHz (hint, radio_slot_id)



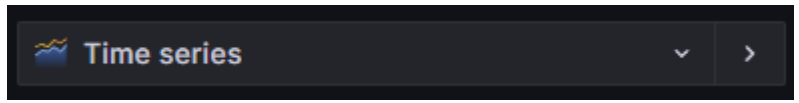

```
SELECT last("curr_freq") AS "Channel",
last("chan_width") AS "Ch Width",
last("rrm_channel_change_reason") AS "RRM channel change
reason", last("freq_string") AS "Freq string" FROM
"Cisco-IOS-XE-wireless-access-point-oper:access-point-
oper-data/radio-oper-data/phy-ht-cfg/cfg-data" WHERE
("wtp_mac"::tag =~ /^$APMac$/ AND "radio_slot_id"::tag =
'0') AND $timeFilter GROUP BY time($__interval)
fill(null)
```

# Grafana

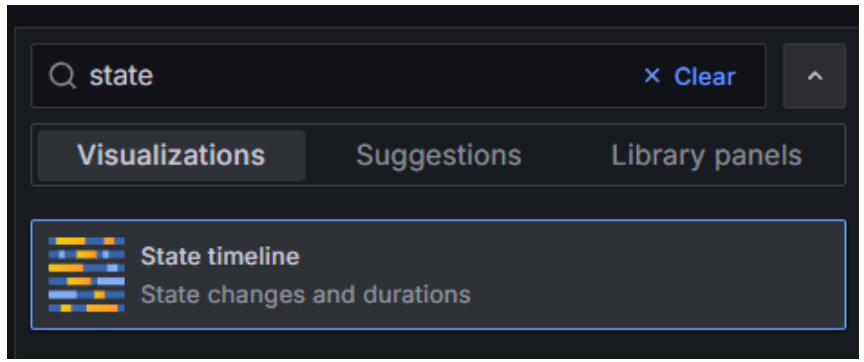
- Add a new visualization to your AP Dashboard




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

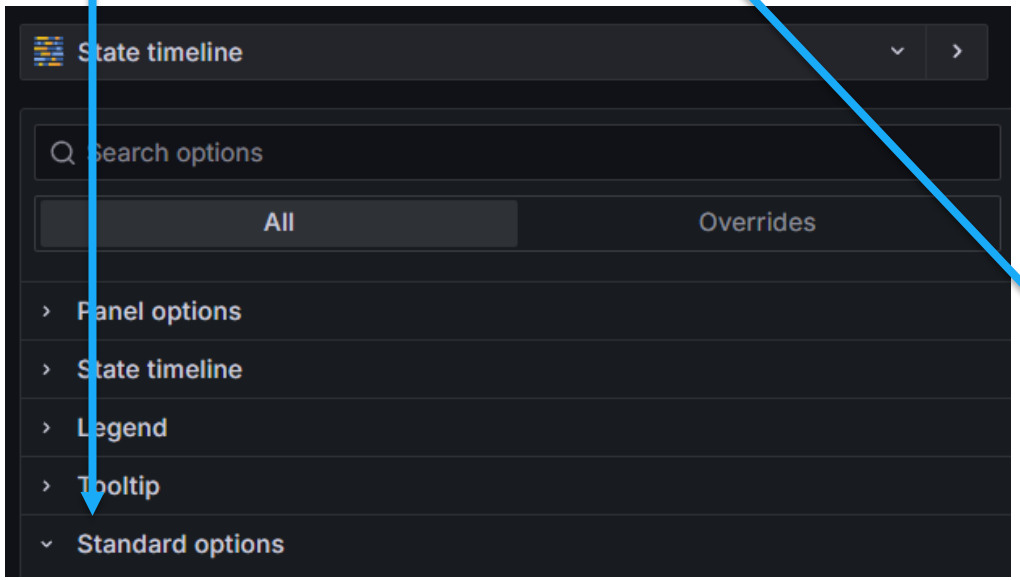


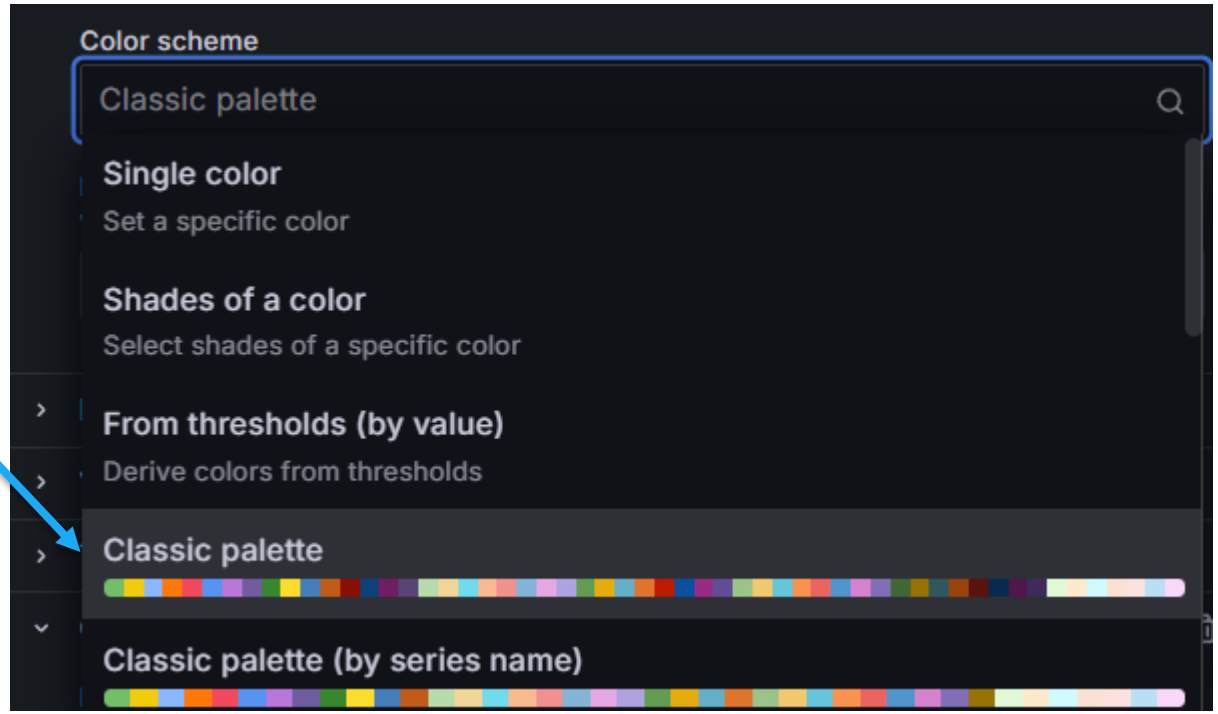

- Search for and select "State timeline "



# Grafana

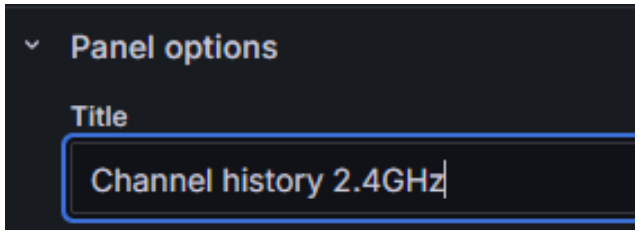
- You probably see that your timeline variables looks a bit weird?
  - That is why we use a color scheme that derive colors from thresholds
- Naviagate to Standard Options
  - At the bottom you will find Color scheme
  - Change it to f.ex classic palette



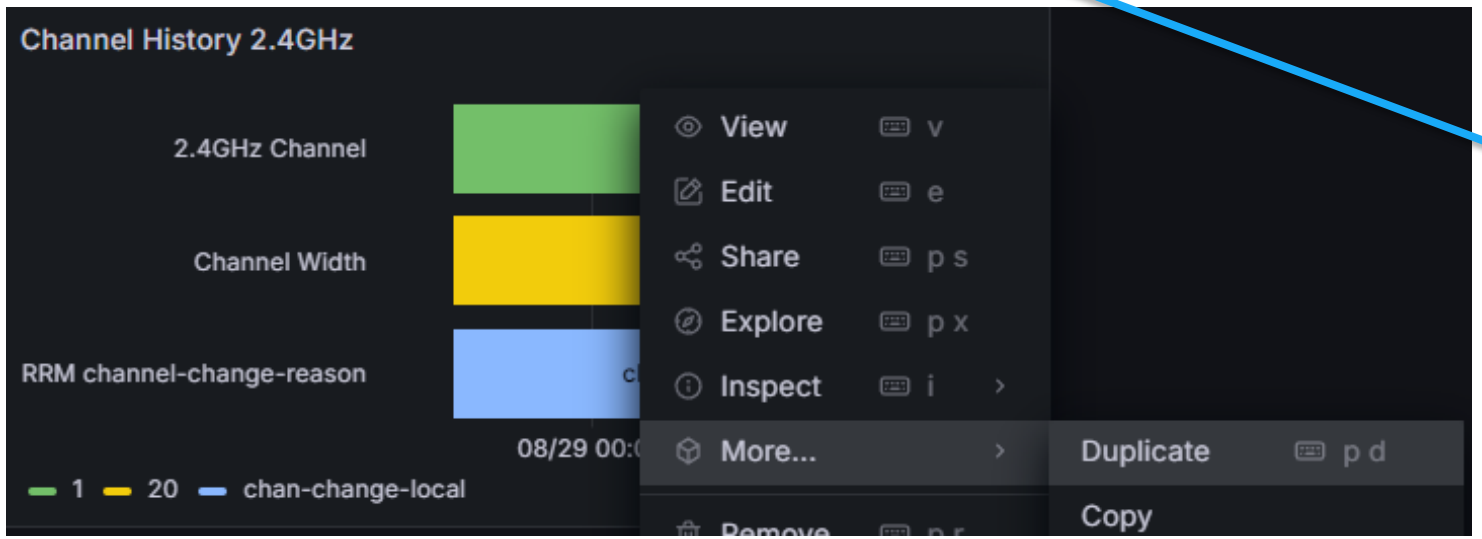



# Grafana

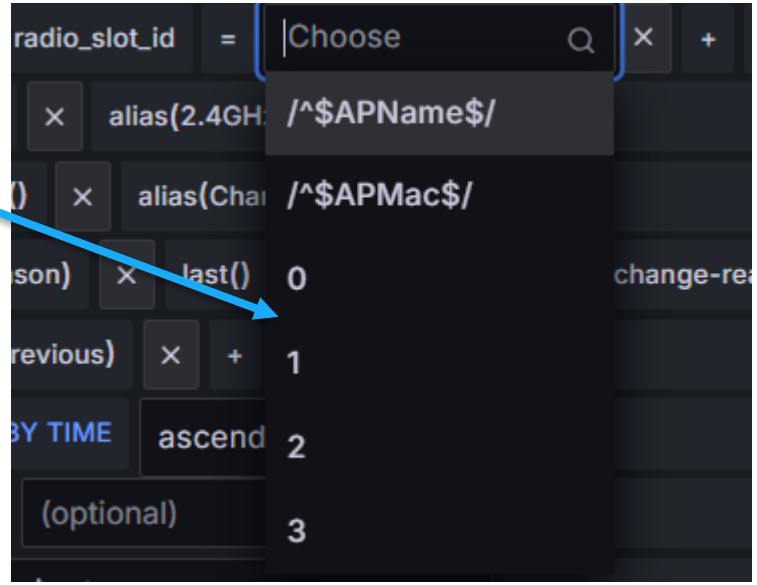
- Name the panel




- Edit the panel (top right corner and duplicate) and make one for 2.4, 5 and 6GHz (radio-slot)

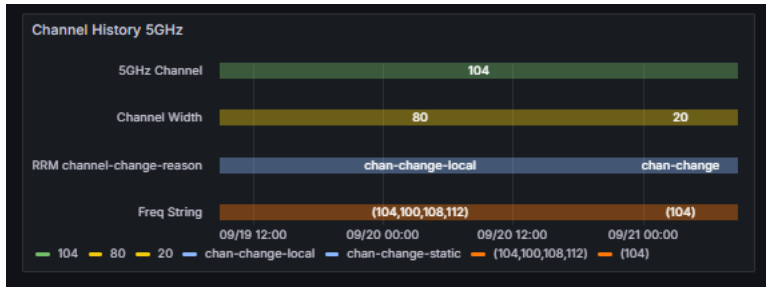


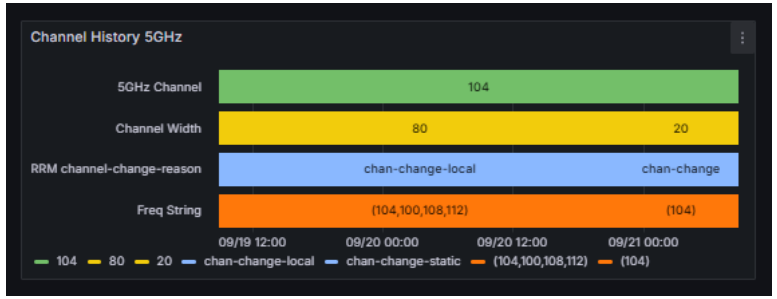

- Slot 0 = 2.4GHz
- Slot 1 = 5GHz
- Slot 2 (if AP is dual 5GHz) = 5GHz
- Slot 3 = 6GHz

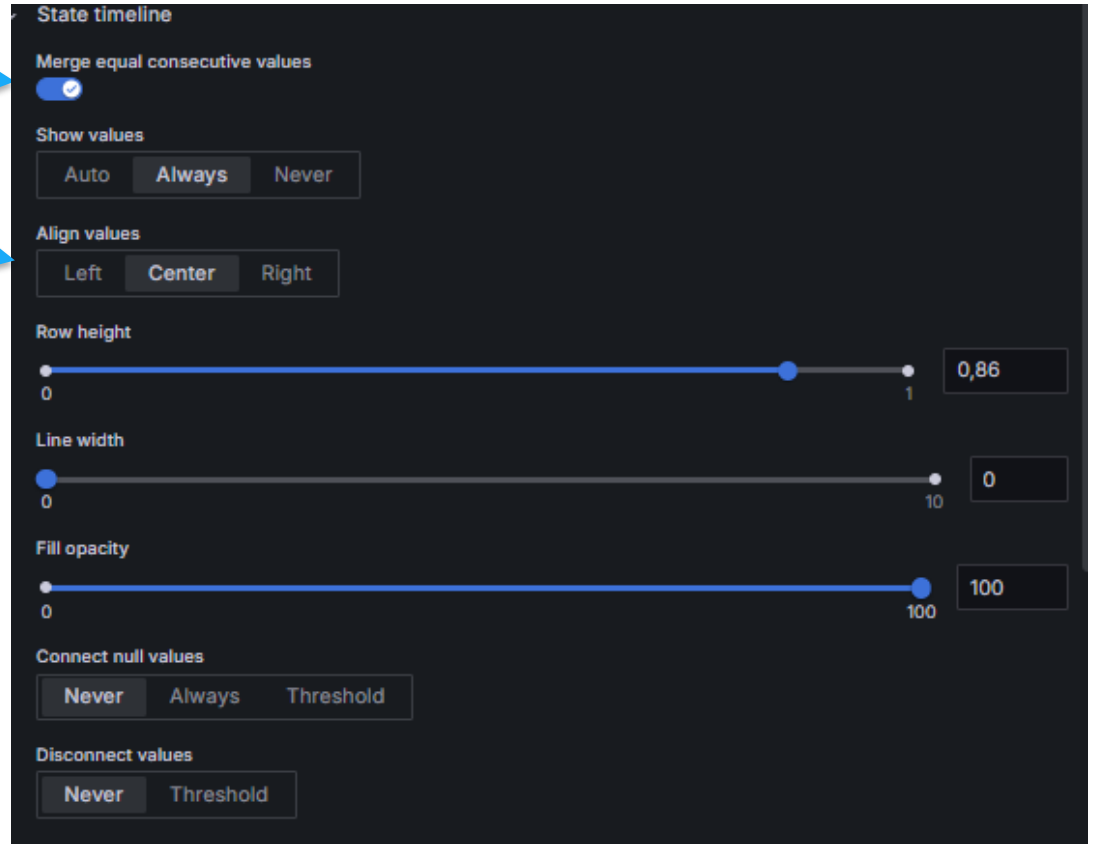


# Grafana (optional)

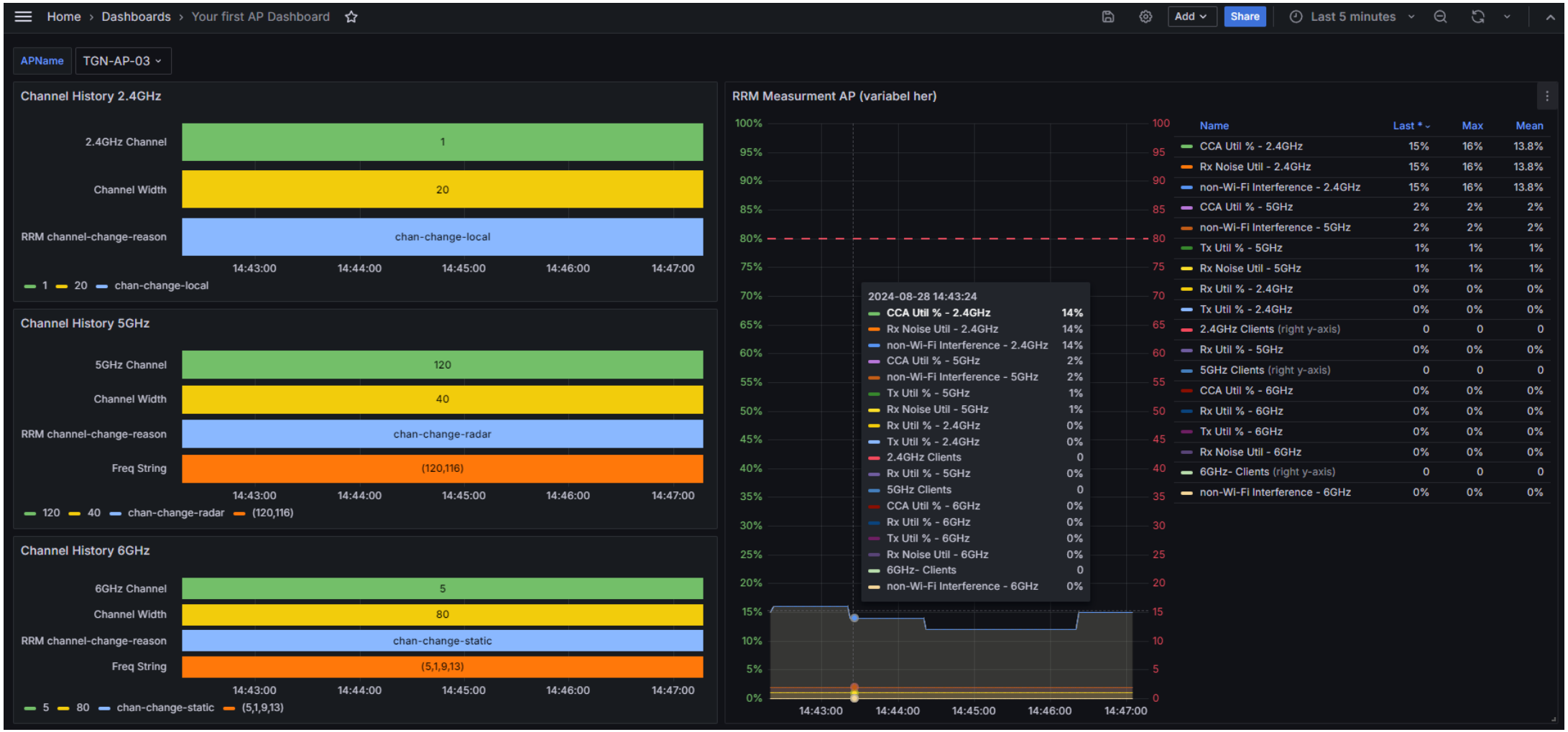
- Merge consecutive values
- Align to text to center.
- Changer row hight and fill opacity



# My first AP Dashboard (example)



# My first AP Dashboard (example 2)

# Save your dashboard to JSON