NXOS télémétrie avec Telegraf, InfluxDB et Grafana

Installation de la VM TIG

Virtual Machine

Prerequis

image : ubuntu-20.04.4-live-server-amd64.iso

2 voire 4 CPUs / 32G RAM / 200G Disk thin provisioning

Configuration NTP client

Ajouter l'IP du serveur NTP dans `/etc/systemd/timesyncd.conf` :

```
[Time]
NTP=192.168.123.254
```

Redémarrer le service NTP :

```
sudo systemctl restart systemd-timesyncd.service
```

Changer la timezone à Paris (CET +1) :

```
 sudo timedatectl set-timezone Europe/Paris
```

Vérification de la synchronisation NTP :

```
labuser@telemetry:~$ timedatectl
               Local time: Mon 2022-03-21 09:25:33 CET
           Universal time: Mon 2022-03-21 08:25:33 UTC
                 RTC time: Mon 2022-03-21 08:25:34
                Time zone: Europe/Paris (CET, +0100)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

Accès root en SSH

> Optionnel, permet d'éditer les fichiers de configuration (nécessitant sudo) de la VM TIG à distance en sftp.

Dans le fichier `/etc/ssh/sshd_config` , remplacer

```
#PermitRootLogin prohibit-password
```

par

```
PermitRootLogin yes
```

Redémarrer le service SSH :

```
sudo systemctl restart ssh
```

Assigner un mot de passe au compte `root` :

```
labuser@telemetry:~$ sudo passwd
New password:
Retype new password:
passwd: password updated successfully
```

InfluxDB

> Utilisation de InfluxDB 1.8 dans cet exemple. En effet, à partir de la version 2.0 il y a de nombreux changements (Web UI, authentification par token, databases remplacées par buckets, nouveau language de query flux). L'interfaçage de Grafana avec flux reste encore très limité.

Récupération du dernier package InfluxDB 1.8 :

```
wget https://dl.influxdata.com/influxdb/releases/influxdb_1.8.10_amd64.deb
```

Installation du package :

```
sudo dpkg –i influxdb_1.8.10_amd64.deb
```

Paramétrage du service pour qu'il démarre au boot :

```
sudo /bin/systemctl daemon-reload
sudo /bin/systemctl enable influxdb
sudo /bin/systemctl start influxdb
```

Vérification de l'état du service :

```
 labuser@telemetry:~$ sudo service influxdb status
● influxdb.service - InfluxDB is an open-source, distributed, time series database
     Loaded: loaded (/lib/systemd/system/influxdb.service; enabled; vendor preset:
enabled)
     Active: active (running) since Mon 2022-03-21 16:39:58 CET; 19h ago
       Docs: https://docs.influxdata.com/influxdb/
    Process: 2748 ExecStart=/usr/lib/influxdb/scripts/influxd-systemd-start.sh
(code=exited, status=0/SUCCESS)
   Main PID: 2754 (influxd)
      Tasks: 14 (limit: 38430)
     Memory: 97.9M
     CGroup: /system.slice/influxdb.service
             └─2754 /usr/bin/influxd –config /etc/influxdb/influxdb.conf
```

Telegraf

Configuration du repository pour télécharger le package :

```
wget -qO- https://repos.influxdata.com/influxdb.key | sudo tee /etc/apt/trusted.gpg.d/
influxdb.asc >/dev/null

source /etc/os-release

echo "deb https://repos.influxdata.com/${ID} ${VERSION_CODENAME} stable" | sudo tee /
etc/apt/sources.list.d/influxdb.list

sudo apt-get update
```

Installation du package :

```
sudo apt-get install telegraf
```

Vérification de l'état du service :

```
labuser@telemetry:~$ sudo service telegraf status
● telegraf.service - The plugin-driven server agent for reporting metrics into InfluxDB
     Loaded: loaded (/lib/systemd/system/telegraf.service; enabled; vendor preset:
enabled)
     Active: active (running) since Tue 2022-03-22 13:24:01 CET; 47s ago
       Docs: https://github.com/influxdata/telegraf
   Main PID: 3719 (telegraf)
      Tasks: 10 (limit: 38430)
     Memory: 35.2M
     CGroup: /system.slice/telegraf.service
             └─3719 /usr/bin/telegraf -config /etc/telegraf/telegraf.conf -config-
directory /etc/telegraf/telegraf.d
```

Grafana

Installation du package `libfontconfi1`, bibliothèque de configuration de polices générique :

```
sudo apt-get install -y adduser libfontconfig1
```

Récupération du dernier package Grafana :

```
wget https://dl.grafana.com/enterprise/release/grafana-enterprise_8.4.4_amd64.deb
```

Installation du package :

```
sudo dpkg -i grafana-enterprise_8.4.4_amd64.deb
```

Paramétrage du service pour qu'il démarre au boot :

```
sudo /bin/systemctl daemon-reload
sudo /bin/systemctl enable grafana-server
sudo /bin/systemctl start grafana-server
```

Vérification de l'état du service :

```
labuser@telemetry:~$ sudo service grafana-server status
● grafana-server.service - Grafana instance
     Loaded: loaded (/lib/systemd/system/grafana-server.service; enabled; vendor preset:
enabled)
     Active: active (running) since Tue 2022-03-22 12:38:02 CET; 25s ago
       Docs: http://docs.grafana.org
   Main PID: 45597 (grafana-server)
      Tasks: 9 (limit: 38430)
     Memory: 33.4M
     CGroup: /system.slice/grafana-server.service
             └─45597 /usr/sbin/grafana-server --config=/etc/grafana/grafana.ini --
pidfile=/run/grafana/grafana-server.pid --packaging=deb cfg:default.paths.logs=/var/log>
```

Configuration initiale

Création de comptes InfluxDB

Le compte admin sera utilisé pour les queries dans influxDB (notamment par Grafana). Le compte telegraf sera utilisé par Telegraf pour populer sa database.

```
labuser@telemetry:~$ influx
Connected to http://localhost:8086 version 1.8.10
InfluxDB shell version: 1.8.10
> create user admin with password 'cisco123' with all privileges
> create user telegraf with password 'cisco123' with all privileges
> show users
user     admin
----     -----
admin    true
telegraf true
> quit
```

Activation de l'authentification HTTP pour InfluxDB

Mofifier le fichier `/etc/influxdb/influxdb.conf` :

```
[http]
  # Determines whether HTTP endpoint is enabled.
  enabled = true

  # The bind address used by the HTTP service.
  bind-address = ":8086"

  # Determines whether user authentication is enabled over HTTP/HTTPS.
  auth-enabled = true
```

Simplifier le fichier de configuration Telegraf

Par défaut, le fichier de configuration /etc/telegraf/telegraf.conf contient plus de 9300 lignes. Afin de simplifier la lisibilité, utiliser le template suivant :

```
# Global tags can be specified here in key="value" format.
[global_tags]

# Configuration for telegraf agent
[agent]
  interval = "10s"
  round_interval = true
  metric_batch_size = 1000
  metric_buffer_limit = 10000
  collection_jitter = "0s"
  flush_interval = "10s"
  flush_jitter = "0s"
  precision = ""
  hostname = ""
  omit_hostname = false
  # debug = false
  # quiet = false

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

# Configuration for sending metrics to InfluxDB
[[outputs.influxdb]]
  ## The target database for metrics; will be created as needed.
  ## For UDP url endpoint database needs to be configured on server side.
  database = "telegraf"

  ## HTTP Basic Auth
  username = "telegraf"
  password = "cisco123"


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

# Read metrics about cpu usage
[[inputs.cpu]]
  ## Whether to report per-cpu stats or not
  percpu = true
  ## Whether to report total system cpu stats or not
  totalcpu = true
  ## If true, collect raw CPU time metrics
  collect_cpu_time = false
  ## If true, compute and report the sum of all non-idle CPU states
  report_active = false

# Read metrics about disk usage by mount point
[[inputs.disk]]
  ## Ignore mount points by filesystem type.
  ignore_fs = ["tmpfs", "devtmpfs", "devfs", "iso9660", "overlay", "aufs", "squashfs"]

# Read metrics about disk IO by device
[[inputs.diskio]]

# Get kernel statistics from /proc/stat
[[inputs.kernel]]

# Read metrics about memory usage
```

La section `outputs.influxdb` contient les paramètres d'authentification à InfluxDB. Les métriques envoyées seront poussées dans la database nommée `telegraf`.

Par défaut, plusieurs `inputs` sont activées. Telegraf parse les valeurs de cpu, mémoire, disque, etc ... de la machine sur laquelle il est installé et envoie les métrique associées à InfluxDB.

Relancer les services InfluxDB et Telegraf

```
labuser@telemetry:~$ sudo service influxdb restart
labuser@telemetry:~$ sudo service telegraf restart
```

Vue des métriques dans InfluxDB

Il est possible de se connecter à InfluxDB en CLI pour voir les différentes métriques stockées. Ces métriques sont stockées dans une database et classifiées par measurements (un measurement = 1 service input). Une métrique est une serie de fieldkeys marquée avec le temps de l'envoi.

Connection à InfluxDB en CLI :

```
labuser@telemetry:~$ influx –username 'admin' –password 'cisco123'
Connected to http://localhost:8086 version 1.8.10
InfluxDB shell version: 1.8.10
```

Visualisation des databases :

```
 > show databases
name: databases
name
----
_internal
telegraf
```

Accès à la database `telegraf` :

```
> use telegraf
Using database telegraf
```

Visualisation des measurements :

```
> show measurements
name: measurements
name
----
cpu
disk
diskio
kernel
mem
processes
swap
system
```

Visualisation des fieldkeys pour le measurement cpu :

```
> show field keys from cpu
name: cpu
fieldKey         fieldType
--------         ---------
usage_guest      float
usage_guest_nice float
usage_idle       float
usage_iowait     float
usage_irq        float
usage_nice       float
usage_softirq    float
usage_steal      float
usage_system     float
usage_user       float
```

Visualisation des métriques cpu reçues ces 30 dernières secondes :

```
> select * from cpu where time > now() - 30s
name: cpu
time                cpu       host      usage_guest usage_guest_nice usage_idle
usage_iowait        usage_irq usage_nice usage_softirq       usage_steal usage_system
usage_user
----                ---       ----      ------------ ----------------- -----------
-------------       --------- ---------- --------------       ------------ -------------
-----------
1647959430000000000 cpu-total telemetry 0           0                99.74962443664047 0
0         0          0                    0           0.1502253380069801
0.10015022533799858
1647959430000000000 cpu0      telemetry 0           0                99.69939879758861 0
0         0          0.10020040080160768 0           0.10020040080160546
0.10020040080158765
1647959430000000000 cpu1      telemetry 0           0                99.89979959919623 0
0         0          0                    0           0.10020040080160499 0
1647959430000000000 cpu2      telemetry 0           0                99.79959919841063 0
0         0          0                    0           0.10020040080162325
0.10020040080158765
1647959430000000000 cpu3      telemetry 0           0                99.69939879758867 0
0         0          0                    0           0.2004008016031744
0.10020040080160499
1647959440000000000 cpu-total telemetry 0           0                99.7995991984106
0.02505010020040593 0         0          0                   0
0.07515030060122224 0.10020040080160592
1647959440000000000 cpu0      telemetry 0           0                99.7993981945976
0.10030090270813105 0         0          0                   0           0
0.10030090270814664
1647959440000000000 cpu1      telemetry 0           0                99.7997997998136  0
0         0          0                    0           0
0.20020020020023677
1647959440000000000 cpu2      telemetry 0           0                99.89989989989772 0
0         0          0                    0           0.10010010010010015 0
1647959440000000000 cpu3      telemetry 0           0                99.69939879760679 0
0         0          0                    0           0.20040080160324744
0.10020040080164151
1647959450000000000 cpu-total telemetry 0           0                99.7994987468628  0
0         0          0                    0           0.05012531328319622
0.15037593984962427
1647959450000000000 cpu0      telemetry 0           0                99.69909729186904 0
0         0          0                    0           0.20060180541625763
0.10030090270811101
1647959450000000000 cpu1      telemetry 0           0                99.8997995991962  0
0         0          0                    0           0
0.10020040080162325
1647959450000000000 cpu2      telemetry 0           0                99.89989989989772 0
0         0          0                    0           0
0.10010010010011794
1647959450000000000 cpu3      telemetry 0           0                99.49899799598111 0
0         0          0                    0           0.20040080160319218
0.30060120240474375
```

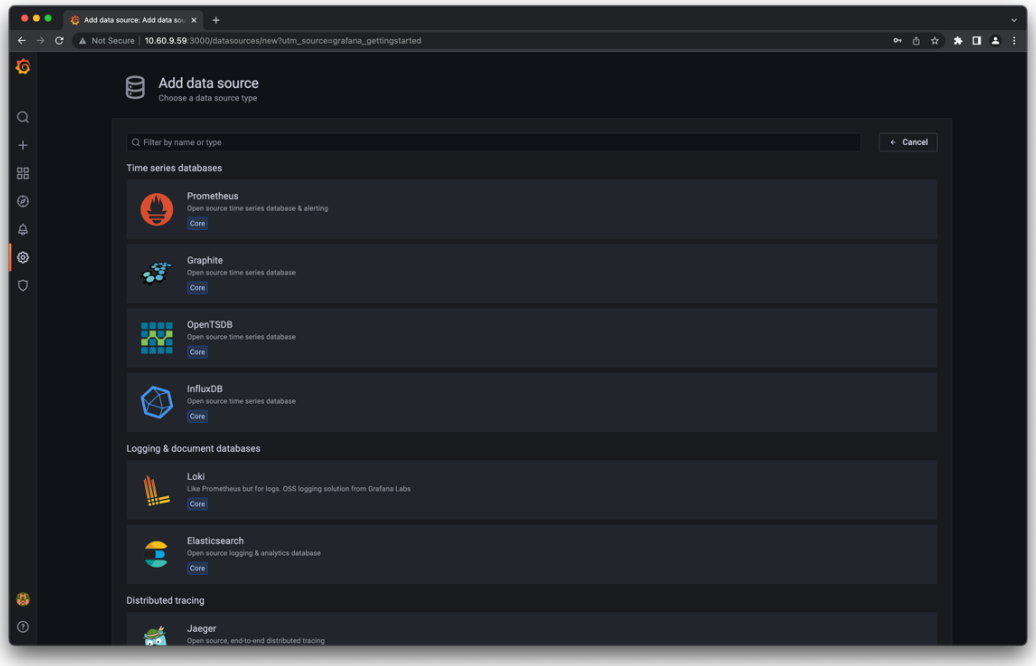
Configuration initiale de Grafana

Connection à Grafana

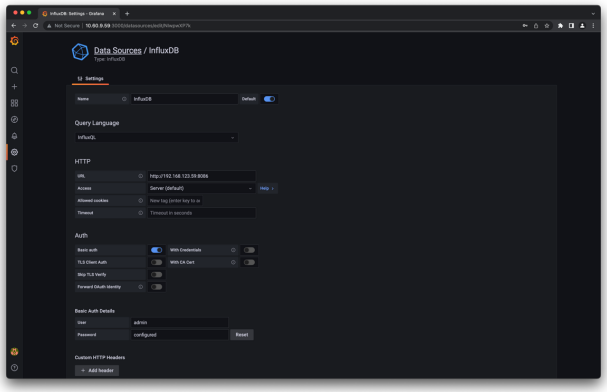
Se connecter en http sur la machine TIG avec le port TCP 3000 (port d'écoute par défaut de Grafana).
Les crédentials par défaut sont **admin** / **admin**.
Change le mot de passe par défaut du compte admin.

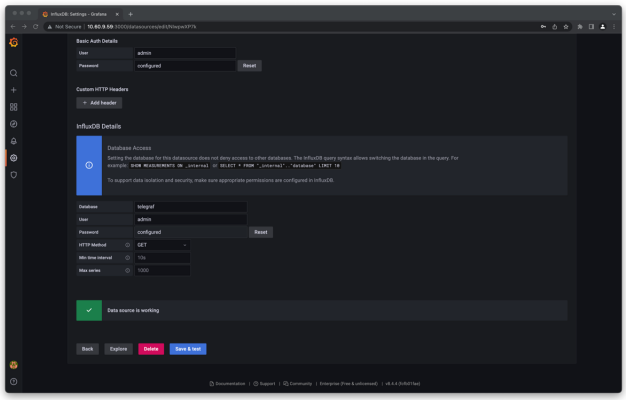
Ajout de la data source InfluxDB

Cliquer sur *Add data source* et chosiir *InfluxDB* :



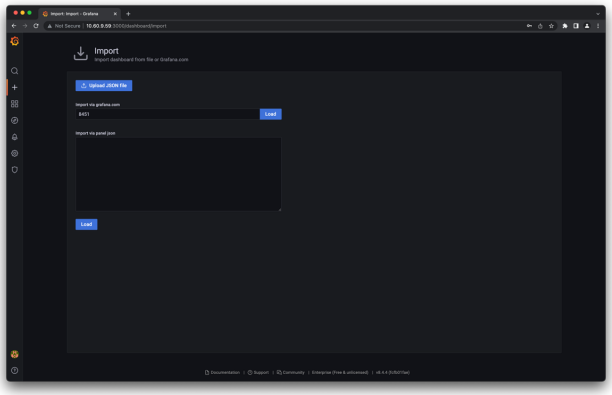
Remplir les settings pour l'accès à InfluxDB :

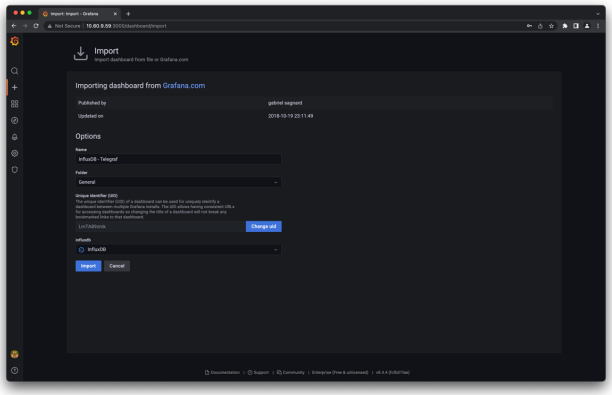




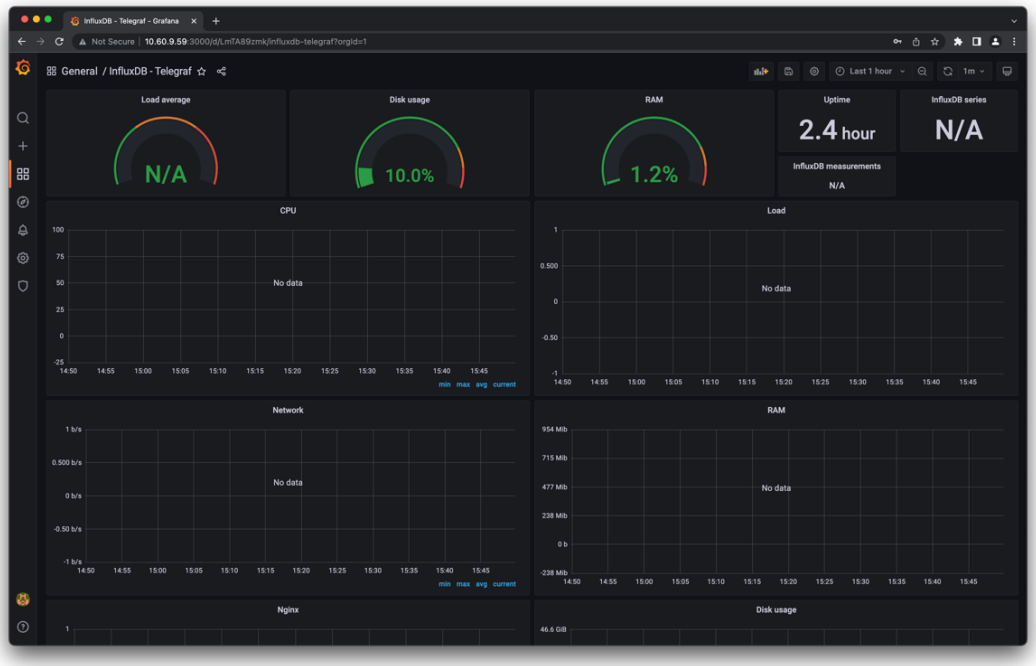
Finir par *Save & test*. Vérifier que la connection à la Data source est opérationnel.

Aller ensuite dans le menu *Import* à travers le bouton *+* de la barre latérale de gauche. Grafana propose l'import de dashboards prédéfinis. Nous allons en récupérer un qui met en valeur les métriques de cpu, disk, ... que Telegraf popule déjà. L'ID de ce dashboard est 8451. Préciser l'usage de la data source InfluxDB.

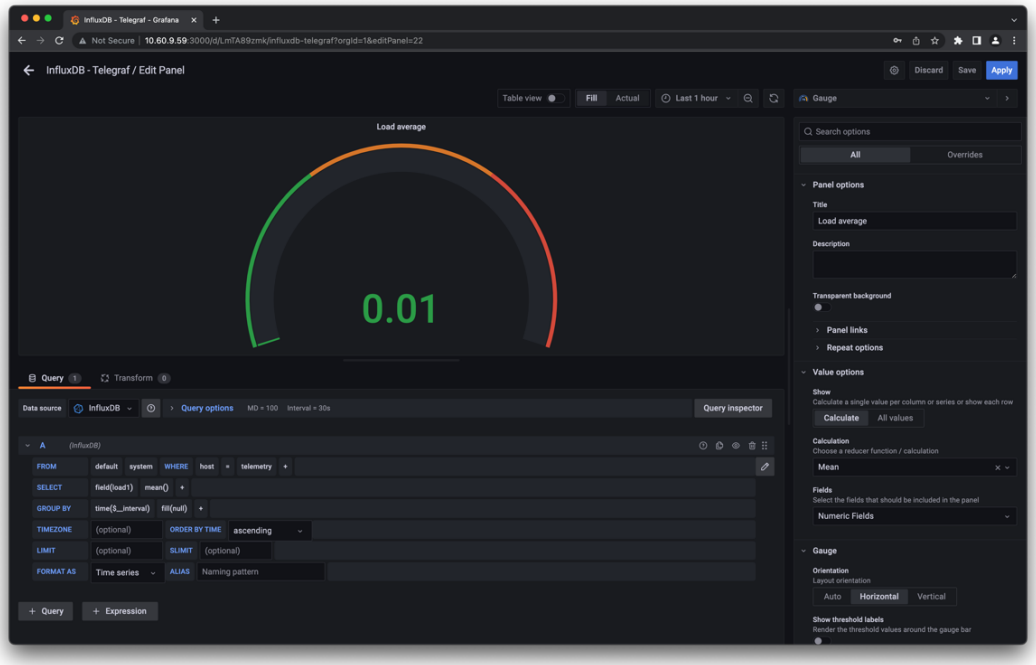




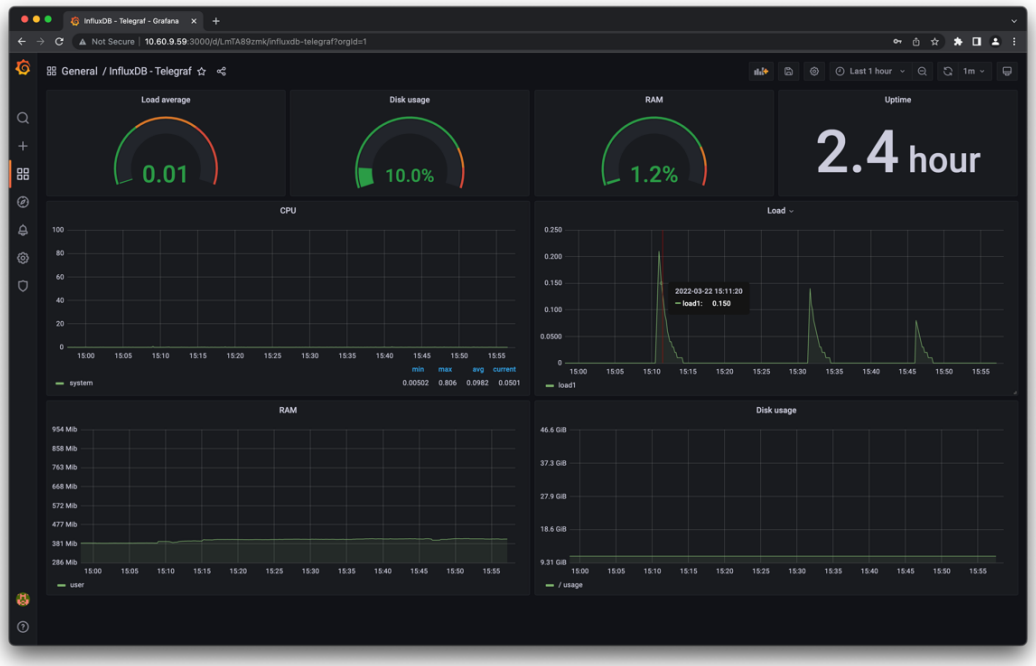
A ca stade, seuls quelques panels sont fonctionnels. Par exemple, le panel *RAM* fonctionne mais par le *panel Load average*.



Cliquer sur la barre de titre *Load average* pour editer le panel. Au niveau de la requête, on voit que le panel fait une requête sur le measurement `system` et le fieldkey `load1` mais sur le mauvais host `nagisa`. Cliquer sur `nagisa` pour changer le host (il va être automatiquement proposé car fait partie des fieldkeys du measurement `system`). Le rendu du panel va immediatement se mettre à jour :

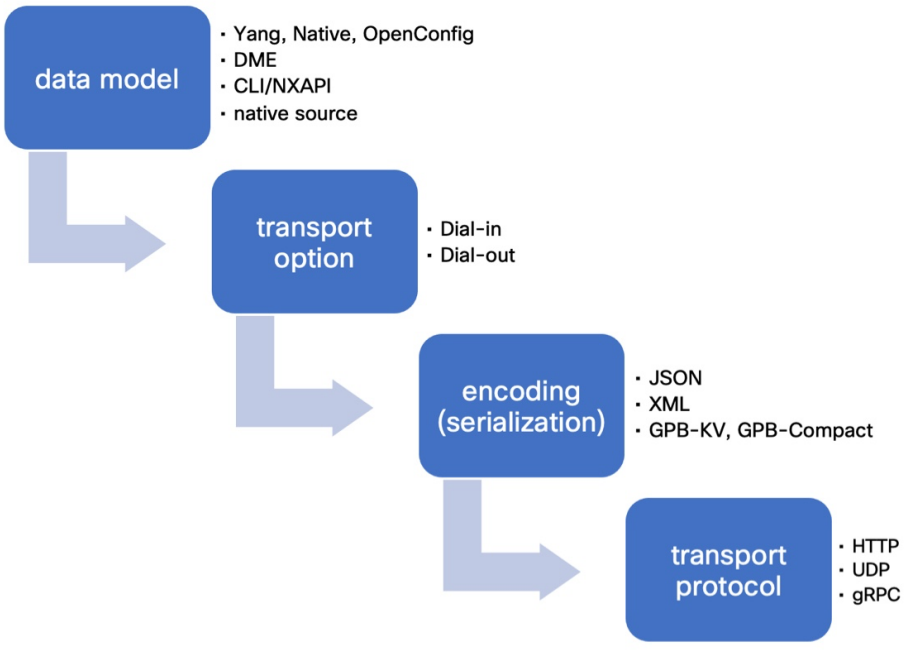


Répéter l'opération pour les autres panels. Retirer également les panels liés au réseau et à InfluxDB (Telegraf n'envoie aucune métrique par défaut).



Exemple de dashboarding en Dial Out

NX-OS sur Nexus 9000 supporte en grande variété de combinaisons pour la télémétrie, que ce soit en terme de data models, d'option de transport, d'encoding et de protocol de trabnsport :



Dans cet exemple, nous allons utiliser le protocol de transport gRPC en Dial-Out avec GPB comme encoding. Nous utiliserons également les data models DME, NXAPI-CLI et Yang OpenConfig pour le choix des métriques à envoyer à Telegraf.

Activation du plugin Cisco MDT

Telegraf inclut le plugin **Cisco Model-Driven Telemetry (MDT) Input**. Pas besoin de l'installer dans Telegraf, il est déjà présent à l'installation de Telegraf. Il faut juste l'activer. Ce plugin est capable d'écouter en TCP ou gRPC Dialout les informations envoyées par les plateformes NX-OS, IOS XR et IOS XE.

Pour activer le plugin, il faut rajouter sa section de configuration dans le fichier `/etc/telegraf/telegraf.conf` et relancer le service.

Section de configuration du plugin Cisco MDT :

```
 [[inputs.cisco_telemetry_mdt]]
#  ## Telemetry transport can be "tcp" or "grpc".  TLS is only supported when using the
grpc transport.
 transport = "grpc"
#
#  ## Address and port to host telemetry listener
 service_address = ":57000"
#
#  ## Enable TLS; grpc transport only.
#  # tls_cert = "/etc/telegraf/cert.pem"
#  # tls_key = "/etc/telegraf/key.pem"
#
#  ## Enable TLS client authentication and define allowed CA certificates; grpc
transport only.
#  # tls_allowed_cacerts = ["/etc/telegraf/clientca.pem"]
#
#  ## Define aliases to map telemetry encoding paths to simple measurement names
#  [inputs.cisco_telemetry_mdt.aliases]
```

Dans le snippet ci-dessus, il est précisé que le plugin va écouter le protocole gRPC sur le port TCP 57000. Une authenfication TLS est possible en gRPC.
L'envoi vers InfluxDB est automatique grâce à l'output configuré plus tôt.

Relancer le service Telegraf :

```
labuser@telemetry:~$ sudo service telegraf restart
```

Configuration des Nexus 9000

Pour configurer la télémétrie en NX-OS, il faut activer la feature telemetry :

```
feature telemetry
```

Dans la section telemetry, il faut ensuite configurer un **destination-profile**. Dans cette sous section, on retrouve les paramètres de configuration tels que la VRF à utiliser, l'interface source, la politique de retry, etc ...

```
telemetry
  destination-profile
    use-vrf management
```

Ensuite, il faut créer un **destination-group** qui va définir à qui envoyer les données, à savoir l'IP et le port de la VM TIG qui contient Telegraf ainsi que les caractéristiques de de protocol (gRPC) et d'encoding (GPB). Il est possible de configurer plusieurs destinations si nécessaire.

```
telemetry
  destination-group 1
    ip address 192.168.123.59 port 57000 protocol gRPC encoding GPB
```

Il faut maintenant créer un ou plusieurs **sensor-group**. Un sensor-group spécifie qu'elles sont les informations à envoyer et depuis quel data model. Il faut créer un sensor group par data model. Dans l'exemple suivant, 3 sensor-groups sont configurés. Un en DME, un autre en NXAPI-CLI et le dernier en Yang OpenConfig.

Le format du sensor path est le suivant :

- DME : DN de l'objet
- Yang : [nom du module]:[xpath du container ou de la leaf]
- NXAPI-CLI : nom de la show command

```
telemetry
  destination-profile
    use-vrf management
  destination-group 1
    ip address 192.168.123.59 port 57000 protocol gRPC encoding GPB
  sensor-group 1
    data-source DME
    path sys/bgp/inst depth unbounded
    path sys/intf depth unbounded
  sensor-group 2
    data-source NX-API
    path "show bgp l2vpn evpn summary" depth unbounded
    path "show environment power" depth unbounded
    path "show environment temperature" depth 0
    path "show interface status" depth unbounded
    path "show mac address-table count" depth unbounded
    path "show mac address-table dynamic" depth unbounded
    path "show nve vni summary" depth unbounded
    path "show system resources" depth unbounded
    path "show version" depth unbounded
  sensor-group 3
    data-source YANG
    path openconfig-interfaces:interfaces/interface/config
```

Le mot clé `unbounded` signifie que toute la profondeur de l'arbre depuis l'objet spécifié va être envoyée.

> Pour utiliser le Yang OpenConfig, le modèle doit être préalablement chargé sur les Nexus

```
copy http://10.60.7.9/bin/nx-os/n9000/mtx-openconfig-all-1.0.0.0-9.3.8.lib32_n9000.rpm
bootflash: vrf management

install add mtx-openconfig-all-1.0.0.0-9.3.8.lib32_n9000.rpm activate
```

Enfin, il faut définir une **subscription** qui va faire le lien entre les sensor-groups et le destination-group. C'est également dans la subscription que l'on précise la fréquence d'envoi en ms.

```
telemetry
  subscription 1
    dst-grp 1
    snsr-grp 1 sample-interval 10000
    snsr-grp 2 sample-interval 10000
    snsr-grp 3 sample-interval 10000
```

Un sample-interval de 0 signifie que que la subscription est event-based.

En résumé, nous allons donc pousser la configuration suivante sur les leaves et spines du setup :

```
feature telemetry

telemetry
  destination-profile
    use-vrf management
  destination-group 1
    ip address 192.168.123.59 port 57000 protocol gRPC encoding GPB
  sensor-group 1
    data-source DME
    path sys/bgp/inst depth unbounded
    path sys/intf depth unbounded
  sensor-group 2
    data-source NX-API
    path "show bgp l2vpn evpn summary" depth unbounded
    path "show environment power" depth unbounded
    path "show environment temperature" depth unbounded
    path "show interface status" depth unbounded
    path "show mac address-table count" depth unbounded
    path "show mac address-table dynamic" depth unbounded
    path "show nve vni summary" depth unbounded
    path "show system resources" depth unbounded
    path "show version" depth unbounded
  sensor-group 3
    data-source YANG
    path openconfig-interfaces:interfaces/interface/config
  subscription 1
    dst-grp 1
    snsr-grp 1 sample-interval 10000
    snsr-grp 2 sample-interval 10000
    snsr-grp 3 sample-interval 10000
```

Vérification de la télémétrie sur les Nexus 9000

Le switch est bien connecté au collector :

```
Leaf-111# show telemetry transport

Session Id      IP Address      Port       Encoding     Transport  Status
--------------------------------------------------------------------------------
0               192.168.123.59  57000      GPB          gRPC       Connected
--------------------------------------------------------------------------------

Retry buffer Size:             10485760
Event Retry Messages (Bytes):  0
Timer Retry Messages (Bytes):  0
Total Retries sent:            0
Total Retries Dropped:         0
```

Tous les envois au collector sont sucessful :

```
Leaf-111# show telemetry data collector details

--------------------------------------------------------------------------------
Row ID         Successful     Failed         Skipped        Sensor Path(GroupId)
--------------------------------------------------------------------------------
1              25             0              0              show bgp l2vpn evpn
summary(2)
2              25             0              0              show mac address-table
count(2)
3              25             0              0              openconfig-
interfaces:interfaces/interface/config(3)
4              25             0              0              show interface status(2)
5              25             0              0              show nve vni summary(2)
6              25             0              0              show system resources(2)
7              25             0              0              show environment
temperature(2)
8              25             0              0              show environment power(2)
9              25             0              0              show mac address-table
dynamic(2)
10             26             0              0              sys/intf(1)
11             26             0              0              sys/bgp/inst(1)
12             25             0              0              show version(2)
```

Détails sur chacun des paths envoyés au collector :

```
Leaf-111# show telemetry control database sensor-paths
Sensor Path Database size = 12
------------------------------------------------------------------------------------------
------------
Row ID     Subscribed Linked Groups  Sec Groups  Retrieve level  Path(GroupId) : Query :
Filter
------------------------------------------------------------------------------------------
------------
1          No         1              0           Full            show mac address-table
dynamic(2) : NA : NA
GPB Encoded Data size in bytes (Cur/Min/Max): 802/801/802
JSON Encoded Data size in bytes (Cur/Min/Max): 0/0/0
CGPB Encoded Data size in bytes (Cur/Min/Max): 0/0/0
Collection Time in ms (Cur/Min/Max): 202/199/4941
Encoding Time in ms (Cur/Min/Max): 0/0/1
Transport Time in ms (Cur/Min/Max): 0/0/0
Streaming Time in ms (Cur/Min/Max): 203/200/4943

2          No         1              0           Full            sys/bgp/inst(1) : NA :
NA
GPB Encoded Data size in bytes (Cur/Min/Max): 18844/18843/18844
JSON Encoded Data size in bytes (Cur/Min/Max): 0/0/0
CGPB Encoded Data size in bytes (Cur/Min/Max): 0/0/0
Collection Time in ms (Cur/Min/Max): 9/8/579
Encoding Time in ms (Cur/Min/Max): 17/5/27
Transport Time in ms (Cur/Min/Max): 0/0/1
Streaming Time in ms (Cur/Min/Max): 26/25/585

3          No         1              0           Full            openconfig-
interfaces:interfaces/interface/config(3) : NA : NA
GPB Encoded Data size in bytes (Cur/Min/Max): 13269/13205/13269
JSON Encoded Data size in bytes (Cur/Min/Max): 0/0/0
CGPB Encoded Data size in bytes (Cur/Min/Max): 0/0/0
Collection Time in ms (Cur/Min/Max): 267/267/12441
Encoding Time in ms (Cur/Min/Max): 36/36/157
Transport Time in ms (Cur/Min/Max): 0/0/1
Streaming Time in ms (Cur/Min/Max): 305/303/12600

<SNIP>
```

Vérification de la télémétrie sur InfluxDB

Nouveaux measurements :

```
> show measurements
name: measurements
name
----
cpu
disk
diskio
kernel
mem
openconfig-interfaces:interfaces/interface/config
processes
show bgp l2vpn evpn summary
show environment power
show environment temperature
show interface status
show mac address-table count
show mac address-table dynamic
show nve vni summary
show system resources
show version
swap
sys/bgp/inst
sys/intf
sys/tm-connection-hello
system
```

Fieldkeys pour le path `openconfig-interfaces:interfaces/interface/config` :

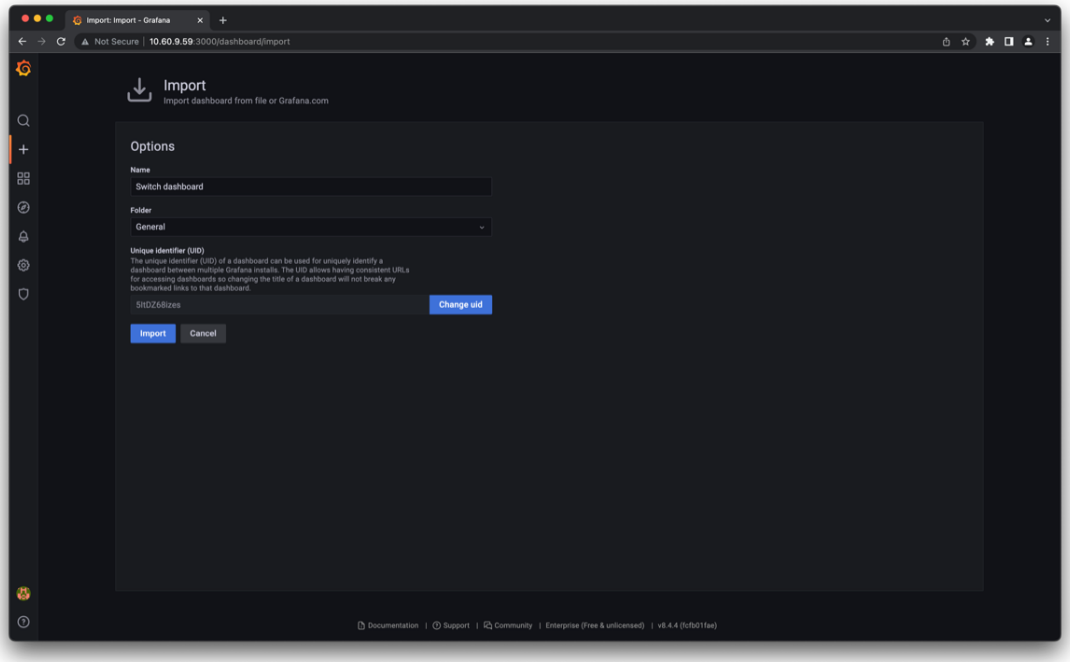
```
> show field keys from "openconfig-interfaces:interfaces/interface/config"
name: openconfig-interfaces:interfaces/interface/config
fieldKey    fieldType
--------    ---------
description string
enabled     boolean
mtu         integer
name        string
tpid        string
type        string
```

Dashboarding Grafana

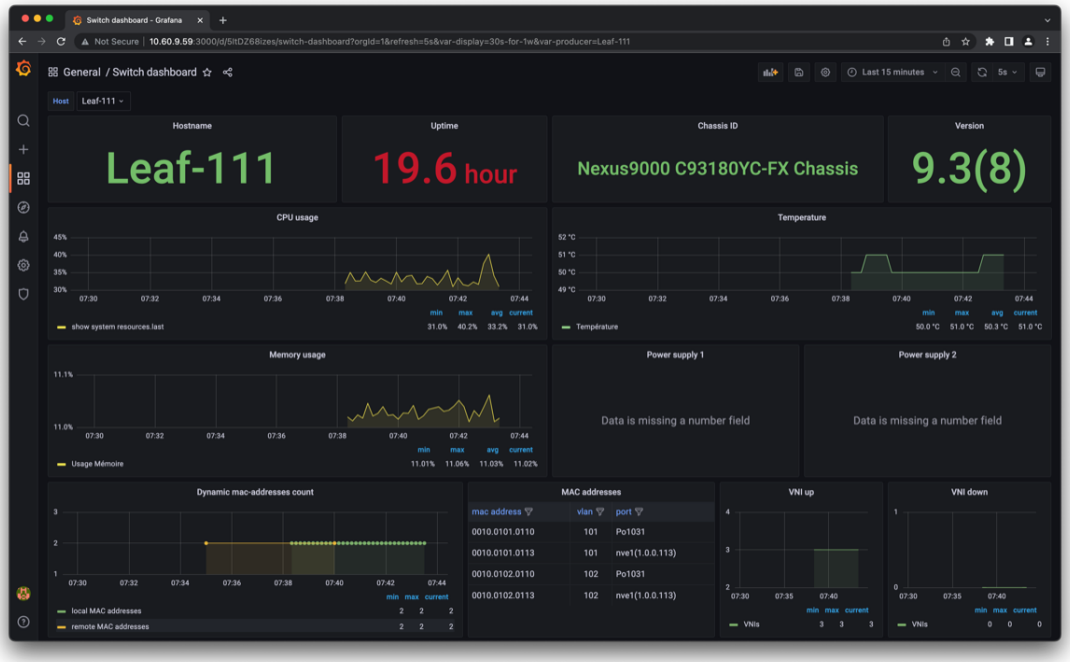
Grafana est déjà connecté à InfluxDB. Il nous reste donc juste à créer un dashboard utilisant les métriques poussées par les Nexus 9000 via Telegraf. Pour ce faire, nous allons importer un Dashboard existant. Ce dashboard est un fichier JSON reprenant les caractéristiques de tous les panels et des queries associées.

Switch dashboard.json
JSON File
63.0 KB

Importer ce fichier dans Grafana en utilisant le même menu que précédemment avec l'option *Upload JSON file*.



Le Dashboard apparait alors à l'écran, avec les différents KPIs (panels) en activité :



A noter que ce dashboard est utilisable pour tous les switches du setup. Il suffit de changer de nom de host en haut à gauche du dashboard. Cet ensemble de hosts est automatiquement détecté et consolidé par Grafana (fieldkey source dans les measurements).

Les panels *Power supply 1 & 2* ne fonctionnent pas actuellement.

Si l'on zoom sur le panel Power Supply 2, on voit qu'il y a 2 queries vers InfluxDB pour extraire les champs `actual_int` et `actual_out` dans le measurement `show environment power` quand la valeur `powersup/TABLE_psinfo` est égale 2 et que la valeur `source` est égale à la variable `/^$Producer$/` à savoir le host selectionné dans le dashboard (Leaf-111 dans notre exemple).

Si on lance le *Query inspector*, on s'aperçoit que les valeurs collectées ne sont pas des integers mais des strings se terminant par *W* pour Watts. Du coup, Grafana ne peut pas grapher ces valeurs.

Coté Nexus 9000, ces valeurs sont le résultat NXAPI-CLI de la commande `show environment power` :

```
Leaf-111# show environment power | json-pretty
{
    "powersup": {
        "voltage_level": "12",
        "TABLE_psinfo": {
            "ROW_psinfo": [
                {
                    "psnum": "1",
                    "psmodel": "------------",
                    "actual_out": "0 W",
                    "actual_input": "0 W",
                    "tot_capa": "0 W",
                    "ps_status": "Absent"
                },
                {
                    "psnum": "2",
                    "psmodel": "NXA-PAC-500W-PE",
                    "actual_out": "131 W",
                    "actual_input": "142 W",
                    "tot_capa": "500 W",
                    "ps_status": "Ok"
                }
            ]
        },
        "power_summary": {
            "ps_redun_mode": "PS-Redundant",
            "ps_oper_mode": "Non-Redundant",
            "tot_pow_capacity": "500.00 W",
            "tot_gridA_capacity": "0.00 W",
            "tot_gridB_capacity": "500.00 W",
            "cumulative_power": "500.00 W",
            "tot_pow_out_actual_draw": "131.00 W",
            "tot_pow_input_actual_draw": "142.00 W",
            "tot_pow_alloc_budgeted": "N/A",
            "available_pow": "N/A"
        }
    }
}
```

Coté InfluxDB, les deux fields actual_int et actual_out sont bien vus en tant que string avec le suffixe W :

```
> select last(actual_input) from "show environment power" where "source" = 'Leaf-111'
and "powersup/TABLE_psinfo" = '2'
name: show environment power
time                last
----                ----
1648019901569000000 143 W
> select last(actual_out) from "show environment power" where "source" = 'Leaf-111' and
"powersup/TABLE_psinfo" = '2'
name: show environment power
time                last
----                ----
1648019921598000000 131 W
> show field keys from "show environment power"
name: show environment power
fieldKey                                        fieldType
--------                                        ---------
actual_input                                    string
actual_out                                      string
powersup/power_summary/available_pow            string
powersup/power_summary/cumulative_power         string
powersup/power_summary/ps_oper_mode             string
powersup/power_summary/ps_redun_mode            string
powersup/power_summary/tot_gridA_capacity       string
powersup/power_summary/tot_gridB_capacity       string
powersup/power_summary/tot_pow_alloc_budgeted   string
powersup/power_summary/tot_pow_capacity         string
powersup/power_summary/tot_pow_input_actual_draw string
powersup/power_summary/tot_pow_out_actual_draw  string
powersup/voltage_level                          integer
ps_status                                       string
psmodel                                         string
tot_capa                                        string
```

Nous allons utiliser Telegraf pour transformer à la fois ces valeurs (en suppriment W à la fin) et changer le type de string à integer grâce a 2 plugins intégrés `processors.strings` et `processors.converter` .

Rajouter la section suivante à `/etc/telegraf/telegraf.conf` :

```
[[processors.strings]]
  [[processors.strings.replace]]
    field = "actual_input"
    old = " W"
    new = ""
  [[processors.strings.replace]]
    field = "actual_out"
    old = " W"
    new = ""

[[processors.converter]]
  [processors.converter.fields]
    integer = ["actual_*"]
```

Puis redémarrer le service Telegraf :

```
labuser@telemetry:~$ sudo service telegraf restart
```

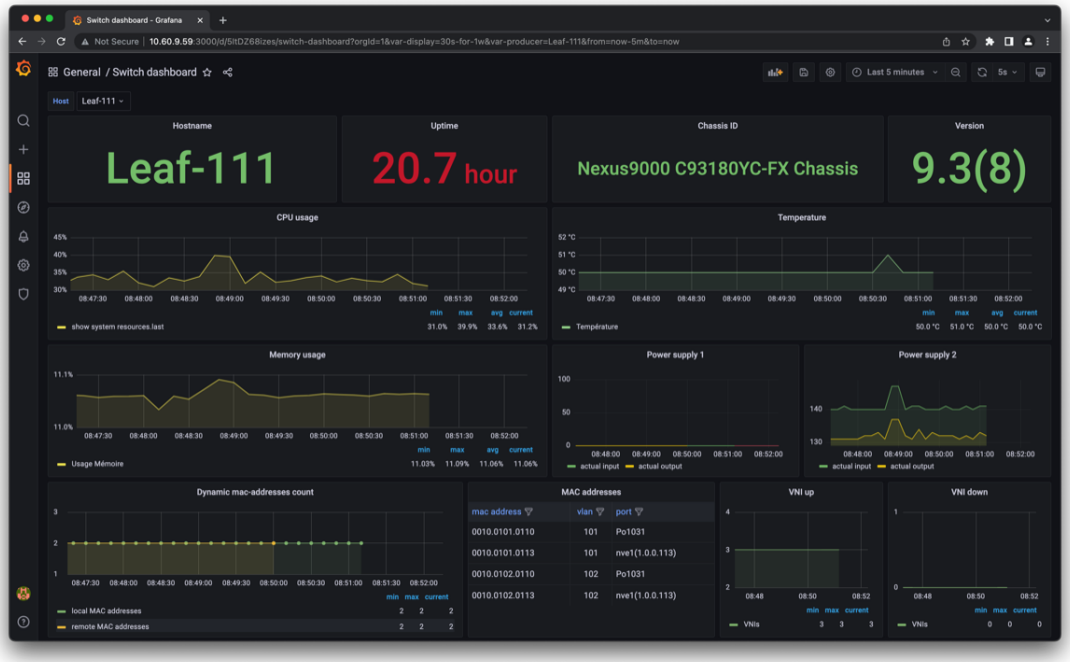
Enfin, optionnel, nous allons détruire dans InfluxDB les métriques existantes associées au measurement `show environment power` :

```
labuser@telemetry:~$ influx -username 'admin' -password 'cisco123'
Connected to http://localhost:8086 version 1.8.10
InfluxDB shell version: 1.8.10
> use telegraf
Using database telegraf
> drop measurement "show environment power"
```

Quelques secondes plus tard, les métriques associés à `show environment power` réapparaissent et dans le bon format :

```
> show field keys from "show environment power"
name: show environment power
fieldKey                                        fieldType
--------                                        ---------
actual_input                                    integer
actual_out                                      integer
powersup/power_summary/available_pow            string
powersup/power_summary/cumulative_power         string
powersup/power_summary/ps_oper_mode             string
powersup/power_summary/ps_redun_mode            string
powersup/power_summary/tot_gridA_capacity       string
powersup/power_summary/tot_gridB_capacity       string
powersup/power_summary/tot_pow_alloc_budgeted   string
powersup/power_summary/tot_pow_capacity         string
powersup/power_summary/tot_pow_input_actual_draw string
powersup/power_summary/tot_pow_out_actual_draw  string
powersup/voltage_level                          integer
ps_status                                       string
psmodel                                         string
tot_capa                                        string
> select last(actual_input) from "show environment power" where "source" = 'Leaf-111'
and "powersup/TABLE_psinfo" = '2'
name: show environment power
time                last
----                ----
1648021752427000000 140
> select last(actual_out) from "show environment power" where "source" = 'Leaf-111' and
"powersup/TABLE_psinfo" = '2'
name: show environment power
time                last
----                ----
1648021762024000000 131
```

Coté Grafana, les panels *Power Supply* sont maintenant fonctionnels (Poser supply 1 est à 0 car il n'y a pas d'alimentation dans ce slot) :



Exemple de dashboarding en gNMI Dial In

Il y a différents protocols de configuration réseau disponibles sur les Nexus 9000, incluant NETCONF, RESTCONF et gNMI. Tous ces protocoles utilisent des data models de type YANG pour manipuler les informations de configuration et d'état. Ils peuvent utiliser des protocoles de transport et d'encoding différents. Dans cet exemple, nous allons nous focaliser sur gRPC Network Management Interface (gNMI) qui s'appuie sur le framework Remote Procedure Call initialement développé par Google (gRPC). gNMI est un protocole unifié à la fois pour la gestion de la configuration (provisioning) et la streaming telemetry.

Tandis que NETCONF et RESTCONF sont spécifiés par l'IETF, les spécifications gNMI sont disponibles librement sur le compte GitHub OpenConfig (https://github.com/openconfig/gnmi).

A partir du train 9.3, un agent gNMI est supporté sur les switches. Il offre notamment un mode de subscription dial-in pour la télémétrie. Dans ce mode dial in, le collector (Telegraf) va spécifier qu'elles sont les métriques qu'il souhaite recevoir (subscribe). Les switches vont alors pousser au collector les informations demandées utilisant soit le data model DME soit YANG (native ou OpenConfig).

L'intérêt de ce mode dial in est qu'il n'y a plus à configurer de manière distribuée sur chacun des switches les paths des sensor-groups. La configuration est centralisée sur le collector Telegraf.

Activation de l'agent gGRC

Pour tourner gRPC et gNMI sur les switches, il faut activer la feature `grpc` :

```
Leaf-111# conf t
Enter configuration commands, one per line. End with CNTL/Z.
Leaf-111(config)# feature grpc
```

Par défaut, les switches écoutent sur le port TCP 50051 dans la VRF de management :

```
Leaf-111(config)# show grpc gnmi service vrf management statistics

==============
gRPC Endpoint
==============

Vrf            : management
Server address : [::]:50051

Cert notBefore : Mar 23 08:41:31 2022 GMT
Cert notAfter  : Mar 24 08:41:31 2022 GMT

Max concurrent calls            :  8
Listen calls                    :  1
Active calls                    :  0

Number of created calls         :  1
Number of bad calls             :  0

Subscription stream/once/poll   :  0/0/0

Max gNMI::Get concurrent        :  5
Max grpc message size           :  8388608
gNMI Synchronous calls          :  0
gNMI Synchronous errors         :  0
gNMI Adapter errors             :  0
gNMI Dtx errors                 :  0
```

On peut voir qu'un self-signed certificat d'une durée d'une journée a été créé automatiquement.

Authentification TLS

La télémétrie avec gNMI utilise des certificats TLS pour valider la communication client (Telegraf) / Serveur (Nexus 9000). Nous allons donc créer pour commencer un certificat self-signed et le charger à la fois sur la VM TIG et les switches.

Génération du certificat self-signed depuis la VM TIG :

```
labuser@telemetry:~$ openssl req -newkey rsa:2048 -nodes -keyout gnmi.key -x509 -days
1000 -out gnmi.pem
Generating a RSA private key
...+++++
..................+++++
writing new private key to 'gnmi.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:Issy
Organization Name (eg, company) [Internet Widgits Pty Ltd]:DIY
Organizational Unit Name (eg, section) []:Webinar
Common Name (e.g. server FQDN or YOUR name) []:gnmi
Email Address []:fcouderc@cisco.com
```

Deux fichiers ont été créés :

```
labuser@telemetry:~$ ls
gnmi.key  gnmi.pem
```

Pour exporter le certificat et la private key sur les Nexus 9000, il faut d'abord exporter au format pkcs12. Cette opération est password protected.

```
labuser@telemetry:~$ openssl pkcs12 -export -out gnmi.pfx -inkey gnmi.key -in gnmi.pem
-certfile gnmi.pem -password pass:cisco123
```

Un nouveau fichier gnmi.pfx a été créé :

```
labuser@telemetry:~$ ls
gnmi.key  gnmi.pem  gnmi.pfx
```

Il faut ensuite télécharger le fichier pfx dans la bootflash des switches :

```
Leaf-111# copy scp://labuser@192.168.123.59/home/labuser/gnmi.pfx bootflash: vrf
management
labuser@192.168.123.59's password:
gnmi.pfx
100% 3653     3.0MB/s   00:00
Copy complete, now saving to disk (please wait)...
Copy complete.

Leaf-111# dir bootflash:///gnmi.pfx
       3653    Mar 23 10:45:16 2022  gnmi.pfx

Usage for bootflash://sup-local
 2465153024 bytes used
114123698176 bytes free
116588851200 bytes total
```

Sur chacun des switches, créer un trustpoint s'appuyant sur le pfx ainsi que le mot de passe qui a servi à l'export :

```
Leaf-111(config)# crypto ca trustpoint gnmicert
Leaf-111(config-trustpoint)# crypto ca import gnmicert pkcs12 gnmi.pfx cisco123
```

Enfin, configurer l'agent gRPC pour qu'il utilise le certificat du trustpoint que l'on vient de créer :

```
Leaf-111(config)# grpc certificate gnmicert
```

On voit maintenant que l'agent utilise le nouveau certificat qui a une durée de 1000 jours :

```
Leaf-111(config)# show grpc gnmi service statistics

=============
gRPC Endpoint
=============

Vrf            : management
Server address : [::]:50051

Cert notBefore : Mar 23 08:35:02 2022 GMT
Cert notAfter  : Dec 17 08:35:02 2024 GMT

Max concurrent calls            :  8
Listen calls                    :  1
Active calls                    :  0

Number of created calls         :  1
Number of bad calls             :  0

Subscription stream/once/poll   :  0/0/0

Max gNMI::Get concurrent        :  5
Max grpc message size           :  8388608
gNMI Synchronous calls          :  0
gNMI Synchronous errors         :  0
gNMI Adapter errors             :  0
gNMI Dtx errors                 :  0
```

Enfin, coté Télégraf, il faut mettre le fichier pem à disposition. Nous allons le copier dans le répertoire `/etc/telegraf/telegraf.conf` :

```
labuser@telemetry:~$ sudo cp /home/labuser/gnmi.pem /etc/telegraf/gnmi.pem
```

Configuration du plugin gNMI

Telegraf inclut le plugin **gNMI input**. Pas besoin de l'installer dans Telegraf, il est déjà présent à l'installation de Telegraf.

Pour activer le plugin, il faut rajouter sa section de configuration dans le fichier `/etc/telegraf/telegraf.conf` et relancer le service.

Section de configuration du plugin gNMI :

```
[[inputs.gnmi]]
  ### Address and port of the gNMI gRPC server
  addresses =
["192.168.123.111:50051","192.168.123.112:50051","192.168.123.115:50051","192.168.123.12
1:50051","192.168.123.122:50051"]

  ### define credentials
  username = "admin"
  password = "cisco123"

  ## GNMI encoding requested (one of: "proto", "json", "json_ietf")
  encoding = "proto"

  ### enable client-side TLS and define CA to authenticate the device
  enable_tls = true
  tls_ca = "/etc/telegraf/gnmi.pem"
  insecure_skip_verify = true

[inputs.gnmi.tags]
  tag1 = "nxos_gnmi"

[[inputs.gnmi.subscription]]
  origin = "device"
  path = "/System/intf-items/phys-items/PhysIf-list/dbgIfIn-items"
  name = "native_ingress_rate"
  subscription_mode = "sample"
  sample_interval = "10s"

[[inputs.gnmi.subscription]]
  origin = "device"
  path = "/System/intf-items/phys-items/PhysIf-list/dbgIfOut-items"
  name = "native_egress_rate"
  subscription_mode = "sample"
  sample_interval = "10s
```

Dans cet exemple, nous demandons à travers le subscribe gRPC l'envoi des métriques correspondant aux paths Yang natifs `/System/intf-items/phys-items/PhysIf-list/dbgIfIn-items` et `/System/intf-items/phys-items/PhysIf-list/dbgIfOut-items` .

Pour info, on aurait pu également demander un xpath OpenConfig comme dans l'exemple suivant :

```
[[inputs.gnmi.subscription]]
  ## Name of the measurement that will be emitted
  name = "openconfig-counters"

  ## Origin and path of the subscription
  origin = "openconfig-interfaces-counters"
  path = "/interfaces/interface/state/counters"
  sample_interval = "10s"
  subscription_mode = "sample"
```

Vérification de la télémétrie sur les Nexus 9000

Au niveau des Nexus 9000, il est possible de visualiser le statut des demandes du subscribe de Telegraf :

```
Leaf-111# show grpc internal gnmi subscription statistics | grep -B 1 -A 8 "Data
Collector Details"
------------------------------------------------------------------------------
Data Collector Details
------------------------------------------------------------------------------


------------------------------------------------------------------------------
Row ID         Successful     Failed         Skipped        Sensor Path(GroupId)
------------------------------------------------------------------------------
1              51             0              0              Cisco-NX-OS-device:System/
intf-items/phys-items/PhysIf-list/dbgIfOut-items(2739)
2              51             0              0              Cisco-NX-OS-device:System/
intf-items/phys-items/PhysIf-list/dbgIfIn-items(2739)
```

Vérification de la télémétrie sur InfluxDB

Nouveaux measurements :

```
> show measurements
name: measurements
name
----
device:/System/intf-items/phys-items
native_egress_rate
native_ingress_rate
```

Fieldkeys pour le path `openconfig-interfaces:interfaces/interface/config` :

```
> show field keys from "native_ingress_rate"
name: native_ingress_rate
fieldKey      fieldType
--------      ---------
broadcastPkts integer
discards      integer
errors        integer
multicastPkts integer
nUcastPkts    integer
noBuffer      integer
octetRate     integer
octets        integer
packetRate    integer
rateInterval  integer
ucastPkts     integer
unknownEtype  integer
unknownProtos integer
> show field keys from "native_egress_rate"
name: native_egress_rate
fieldKey      fieldType
--------      ---------
broadcastPkts integer
discards      integer
errors        integer
multicastPkts integer
nUcastPkts    integer
octetRate     integer
octets        integer
packetRate    integer
qLen          integer
rateInterval  integer
ucastPkts     integer
```

Visualisation des entrées pour l'interface e1/51 du switch 192.168.123.111 :

```
> select * from native_egress_rate where (time > now() - 30s and source =
'192.168.123.111' and id = 'eth1/51')
name: native_egress_rate
time                broadcastPkts discards errors host      id      multicastPkts
nUcastPkts octetRate octets  packetRate path                                  qLen
rateInterval source          tag1      ucastPkts
----                ------------- -------- ------ ----      --      -------------
---------- --------- ------  ---------- ----                                  ----
---------- ------          ----      ---------
1648043044284948048 3             0        0      telemetry eth1/51 13356         13359
26        2901941 0          device:/System/intf-items/phys-items 0    300
192.168.123.111 nxos_gnmi 3939
1648043054287827486 3             0        0      telemetry eth1/51 13357         13360
26        2902027 0          device:/System/intf-items/phys-items 0    300
192.168.123.111 nxos_gnmi 3939
1648043064289414677 3             0        0      telemetry eth1/51 13359         13362
26        2902358 0          device:/System/intf-items/phys-items 0    300
192.168.123.111 nxos_gnmi 3941
```

Dashboarding Grafana

Grafana est déjà connecté à InfluxDB. Il nous reste donc juste à créer un dashboard utilisant les métriques poussées par les Nexus 9000 via Telegraf. Pour ce faire, nous allons importer un Dashboard existant. Ce dashboard est un fichier JSON reprenant les caractéristiques de tous les panels et des queries associées.

Switches interfaces.json
JSON File
12.0 KB

Choisir son switch et les interfaces à monitorer. Le dashboard montre alors les débits entrants et sortant ainsi que les potentielles erreurs sur les interfaces :