Note
/20

Compétences	Acquise	Non Acquise
C04 : Analyser un système informatique		
C05 : Concevoir un système informatique		

TP n°5

Stockage de Séries Temporelles : Ingestion par Telegraf et Base de Données InfluxDB v2

FICHE MISSION 2

I – PROBLEMATIQUE : Comment structurer, convertir et stocker un flux continu de métriques IIoT volatiles (JSON MQTT) dans une base de données de séries temporelles (TSDB) ?

II – OBJECTIFS :

- Comprendre le fonctionnement d'une base de données orientée séries temporelles (InfluxDB v2)
- Déployer et initialiser InfluxDB v2 sur la VM Ubuntu (**192.168.2.100**)
- Configurer le middleware Telegraf (plugin **inputs.mqtt_consumer** et parser **json_v2**)
- Valider l'ingestion des séries temporelles et exécuter des requêtes d'extraction en langage Flux

III – PRE-REQUIS :

- Avoir validé le TP4 IT (Broker **Mosquitto** opérationnel et publication **MQTT** active sur **192.168.2.100:1883**)
- Maîtrise de l'édition de fichiers de configuration sous Linux (**nano**, **vim**)
- Compréhension de la différence entre métadonnées textuelles (**Tags**) et valeurs numériques (**Fields**)

IV – MISE EN SITUATION : Lors du TP4 IT, vous avez validé la transmission des métriques de distance et de réflectivité du capteur **OGD592** sous forme de messages JSON vers le broker Mosquitto. Cependant, ces données restent volatiles : elles disparaissent dès leur affichage en console.

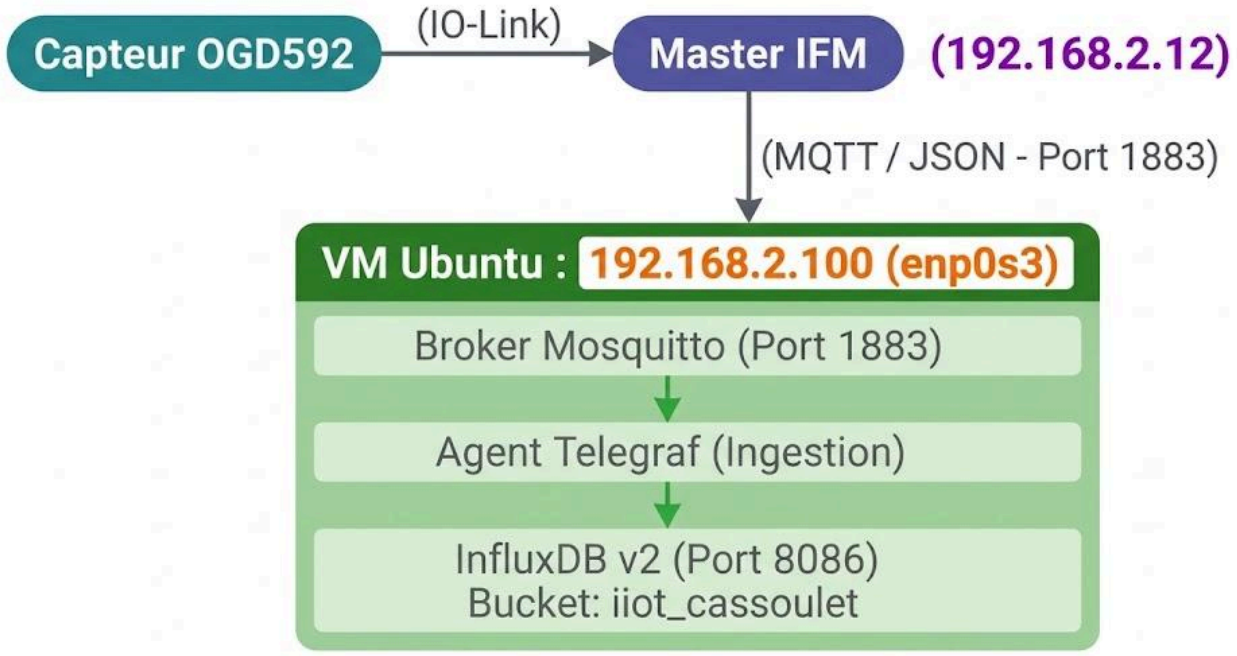
Dans cette mission, vous allez mettre en place la persistance des données sur la VM de supervision usine (**192.168.2.100**). Vous allez installer la base **InfluxDB v2** puis configurer l'agent middleware **Telegraf**. Ce dernier s'abonnera aux topics **MQTT** sur l'interface enp0s3, découpera le **JSON** IFM pour séparer les valeurs de mesure (**Fields**) des métadonnées (**Tags**) et enregistrera l'historique dans le bucket **iiot_cassoulet**.

V – <u>RESSOURCES MATERIELLES</u> : Vous réaliserez cette mission en travaillant avec un **PC** connecté à **Internet** et un accès à un environnement de travail de type **Workspace** avec les outils bureautiques :

- Traitement de texte.
- Tableur.
- Logiciel de diaporama.
- PC Étudiant (**192.168.2.20**) connecté au réseau IT
- Serveur sur une Machine Virtuelle **Ubuntu 26.04 LTS** (Réseau Interne **192.168.2.100** et Accès par Pont Carte Réseau Sans-Fil)
- Master **IO-Link** IFM (AL1304 / AL1306) avec double interface : IT (**192.168.2.12**) et OT (**192.168.1.12**)
- Capteur photoélectrique **IO-Link** OGD592
- Automate PLC Siemens S7-1200 : OT (**192.168.1.10**)
- Logiciel Moneo Configure
- Clients **MQTT** (mosquitto-clients, MQTT Explorer)
- Logiciels installés sur la VM : Broker **Mosquitto**, **InfluxDB v2**, **Agent Telegraf**

VI – <u>RESSOURCES PEDAGOGIQUES</u> : Vous utiliserez le dossier technique fourni ainsi que la documentation technique IFM du Master **IO-Link** et du capteur OGD592.

VII – <u>SCHEMA DU RESEAU</u> :



VIII – ETUDE PREALABLE DE LA PERSISTANCE DES DONNEES :

Dans cette nouvelle mission, nous allons nous intéresser à la persistance des données.

Quelle est la différence majeure entre une base de données relationnelle (**SQL**) et une base de données Time-Series (**InfluxDB**) ? Pourquoi InfluxDB est-il privilégié en **IIoT** ?

Les bases relationnelles **SQL** privilégient l'intégrité des transactions complexes (**ACID**) mais deviennent inefficaces face à l'écriture continue de volumes massifs de données horodatées. A l'inverse, **InfluxDB** est une base de séries temporelles conçue pour absorber des débits d'écriture très élevés et compresser l'historique tout en proposant des fonctions d'agrégation temporelle natives indispensables pour l'**IIoT**.

Quel est le rôle de l'agent middleware **Telegraf** ? Pourquoi est-il préférable d'utiliser cet agent intermédiaire plutôt que de faire écrire directement le Master **IO-Link** dans **InfluxDB** ?

L'agent **Telegraf** sert de pipeline d'infrastructures chargé d'extraire, de transformer et d'insérer les flux de données (**ETL**). Cela évite de surcharger le Master **IO-Link** : le Master transmet son flux **JSON** brut, puis **Telegraf** prend en charge le décodage, le typage des champs et la mise au format d'entrée d'**InfluxDB**.

Expliquer la notion de Parsing **JSON** et la distinction entre **Fields** (champs) et **Tags** (étiquettes) au sein d'**InfluxDB**.

Le parsing **JSON** correspond au découpage d'une chaîne de caractères structurée pour en extraire des variables exploitables. Dans **InfluxDB**, les **Fields** représentent les valeurs mesurées non indexées destinées aux calculs mathématiques (ex. la chaîne hexadécimale de données de processus pdin), alors que les Tags constituent des métadonnées textuelles indexées qui facilitent le filtrage rapide des requêtes (ex. identifiant de l'équipement ou localisation).

IX – INSTALLATION D'INFLUXDB V2 :

Nous allons commencer par installer **InfluxDB v2** sur notre machine virtuelle, taper les commandes suivantes pour l'installer :

```
wget https://dl.influxdata.com/influxdb/releases/influxdb2_2.7.5-1_amd64.deb
sudo dpkg -i influxdb2_2.7.5-1_amd64.deb
```

Activer **InfluxDB v2** et vérifier que le service est correctement lancé en tapant les commandes suivantes :

```
sudo systemctl daemon-reload
sudo systemctl enable --now influxdb
sudo systemctl status influxdb
```

Donner la capture d'écran du bon fonctionnement d'**InfluxDB v2**.

```
root@ubuntu:/home/eleve# sudo systemctl status influxdb
● influxdb.service - InfluxDB is an open-source, distributed, time series database
     Loaded: loaded (/usr/lib/systemd/system/influxdb.service; enabled; preset: enabled)
     Active: active (running) since Sun 2026-09-20 12:54:18 UTC; 2min 21s ago
 Invocation: f7674c18d38449fea748f785f4949490
       Docs: https://docs.influxdata.com/influxdb/
    Process: 3030 ExecStart=/usr/lib/influxdb/scripts/influxd-systemd-start.sh (code=exited, status=0/SUCCESS)
   Main PID: 3034 (influxd)
      Tasks: 7 (limit: 1888)
     Memory: 47.4M (peak: 58.1M)
        CPU: 768ms
     CGroup: /system.slice/influxdb.service
             └─3034 /usr/bin/influxd

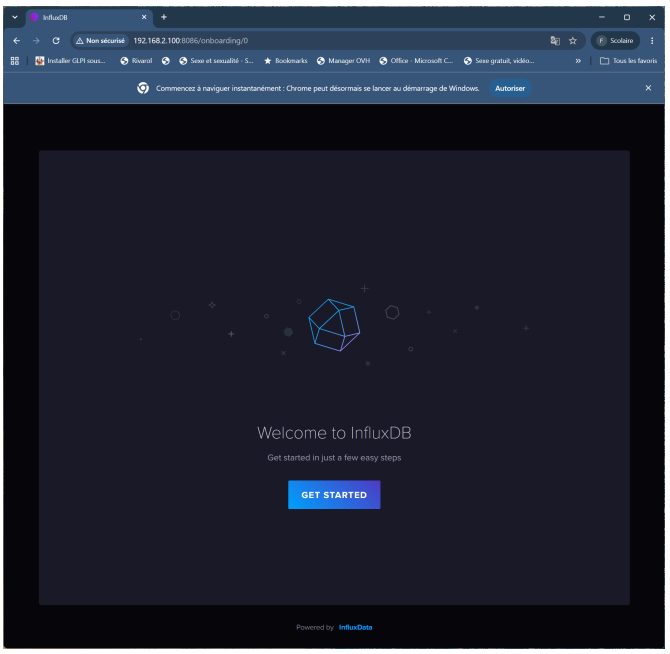
sept. 20 12:54:18 ubuntu influxd-systemd-start.sh[3034]: ts=2026-09-20T12:54:18.089097Z lvl=info msg="Open store (start)" log_id=15RixN2l000 service=storage-en>
sept. 20 12:54:18 ubuntu influxd-systemd-start.sh[3034]: ts=2026-09-20T12:54:18.089155Z lvl=info msg="Open store (end)" log_id=15RixN2l000 service=storage-engi>
sept. 20 12:54:18 ubuntu influxd-systemd-start.sh[3034]: ts=2026-09-20T12:54:18.089171Z lvl=info msg="Starting retention policy enforcement service" log_id=15R>
sept. 20 12:54:18 ubuntu influxd-systemd-start.sh[3034]: ts=2026-09-20T12:54:18.089177Z lvl=info msg="Starting precreation service" log_id=15RixN2l000 service=>
sept. 20 12:54:18 ubuntu influxd-systemd-start.sh[3034]: ts=2026-09-20T12:54:18.089755Z lvl=info msg="Starting query controller" log_id=15RixN2l000 service=sto>
sept. 20 12:54:18 ubuntu influxd-systemd-start.sh[3034]: ts=2026-09-20T12:54:18.092644Z lvl=info msg="Configuring InfluxQL statement executor (zeros indicate u>
sept. 20 12:54:18 ubuntu influxd-systemd-start.sh[3034]: ts=2026-09-20T12:54:18.105379Z lvl=info msg=Listening log_id=15RixN2l000 service=tcp-listener transpor>
sept. 20 12:54:18 ubuntu influxd-systemd-start.sh[3034]: ts=2026-09-20T12:54:18.105586Z lvl=info msg=Starting log_id=15RixN2l000 service=telemetry interval=8h
sept. 20 12:54:18 ubuntu influxd-systemd-start.sh[3030]: InfluxDB started
sept. 20 12:54:18 ubuntu systemd[1]: Started influxdb.service - InfluxDB is an open-source, distributed, time series database.
lines 1-23/23 (END)
```

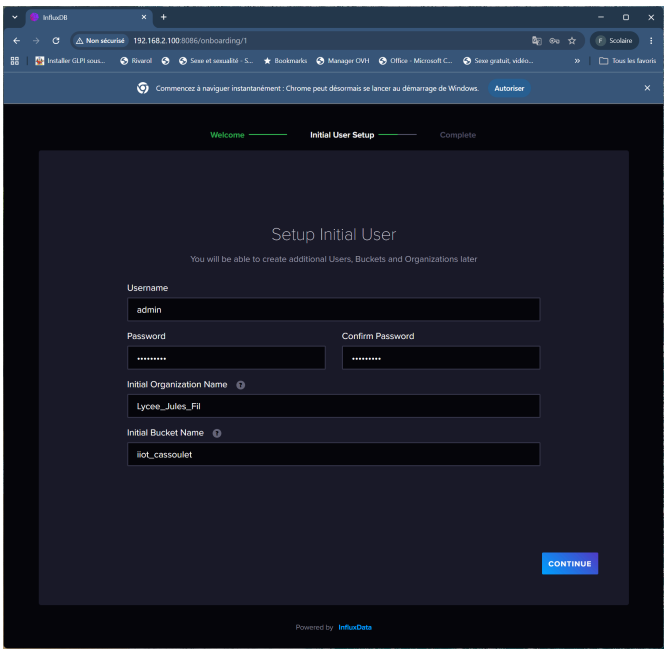
Vous pouvez accéder maintenant à l'interface de configuration d'**InfluxDB v2** en utilisant un navigateur Internet et en tapant l'**URL** suivante : **http://192.168.2.100:8086**

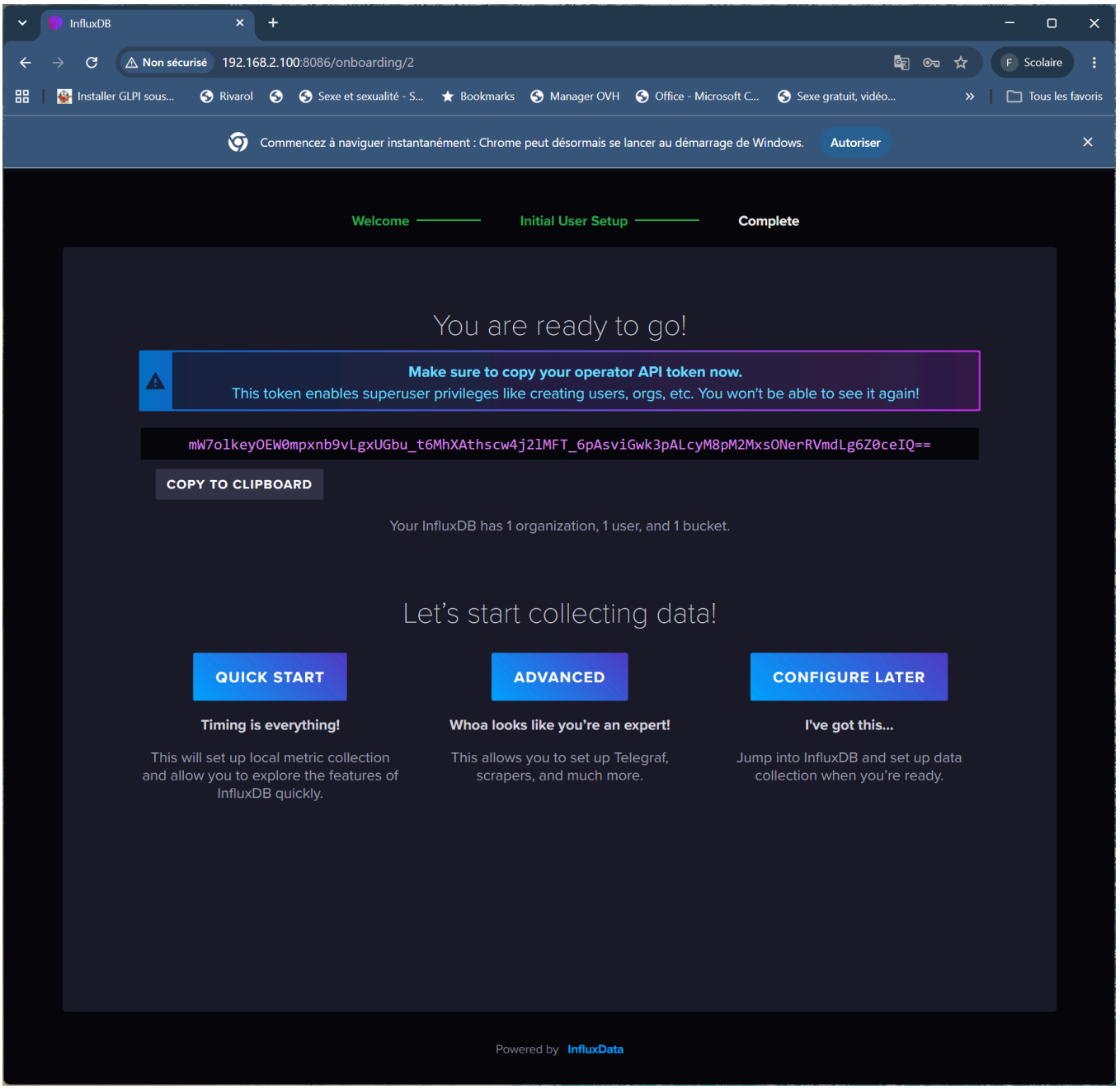
Remplir les informations suivantes et relever le Token généré :

- Organization : **Lycee_Jules_Fil**
- Bucket : **iiot_cassoulet**

Donner les captures d'écran de la configuration d'**InfluxDB v2** et relever le **Token** généré.







Le Token généré est :
mW7olkeyOEW0mpxnb9vLgxUGbu_t6MhXAthscw4j2lMFT_6pAsviGwk3pALcyM8pM2MxsONerRVmdLg6Z0ceIQ==

X – INSTALLATION DE TELEGRAF :

Nous allons maintenant installer **Telegraf**, utiliser la procédure suivante pour l'installer :

Procédure d'installation de Telegraf sur Ubuntu Server 26.04

1. Importer la clé GPG InfluxData dans le trousseau système :

Importer la clé publique officielle **DA61C26A0585BD3B** directement dans le trousseau système d'**APT** :

```
sudo gpg --keyserver keyserver.ubuntu.com --recv-keys DA61C26A0585BD3B
sudo gpg --export DA61C26A0585BD3B | sudo tee /etc/apt/trusted.gpg.d/influxdata.gpg >
/dev/null
```

2. Déclarer le dépôt APT InfluxData :

Configurer l'adresse du dépôt officiel dans le répertoire des sources d'**APT** :

```
echo 'deb [https://repos.influxdata.com/debian](https://repos.influxdata.com/debian) stable
main' | sudo tee /etc/apt/sources.list.d/influxdata.list
```

3. Mettre à jour la liste des paquets et installer Telegraf :

Recharger les dépôts et lancer l'installation :

```
sudo apt update && sudo apt install telegraf -y
```

Note : Si l'écran bleu de configuration d'APT apparaît, sélectionner l'option **« Garder la version actuellement installée »** et valider avec **<Ok>**.

4. Démarrer et vérifier le service Telegraf :

Activer le démarrage automatique au boot de la VM et lance le service immédiatement :

```
sudo systemctl enable --now telegraf
```

Pour vérifier le bon fonctionnement du service :

```
sudo systemctl status telegraf
```

(Le statut doit indiquer **active (running)**).

5. Renommer le fichier de configuration en sauvegarde :

```
sudo mv /etc/telegraf/telegraf.conf /etc/telegraf/telegraf.conf.bak
```

Nous allons configurer **Telegraf** avec la structure réelle du JSON IFM dans le fichier.

➔ Créer le fichier et complétez-le en remplaçant le Token par celui généré lors de l'installation d'**InfluxDB v2** : **/etc/telegraf/telegraf.conf**

```
[[outputs.influxdb_v2]]
 urls = ["http://127.0.0.1:8086"]
 token =
"mW7olkeyOEW0mpxnb9vLgxUGbu_t6MhXAthscw4j2lMFT_6pAsviGwk3pALcyM8pM2MxsONerRVmdLg6Z0ceIQ=="
 organization = "Lycee_Jules_Fil"
 bucket = "iiot_cassoulet"

[[inputs.mqtt_consumer]]
 servers = ["tcp://192.168.2.100:1883"]
 topics = [
   "usine/castelnaudary/ligne1/#"
 ]
 client_id = "telegraf_iiot_client"
 data_format = "json_v2"

 [[inputs.mqtt_consumer.json_v2]]
   [[inputs.mqtt_consumer.json_v2.field]]
     path = "data.payload.*pdin.data"
     rename = "pdin_hex"
     type = "string"

   [[inputs.mqtt_consumer.json_v2.field]]
     path = "data.payload.*counter.data"
     rename = "compteur_event"
     type = "int"
```

Quelle est la commande pour relancer le service **Telegraf** ? Donner la capture d'écran pour vérifier qu'il a bien redémarré et qu'aucune erreur de configuration n'apparaît.

La commande pour relancer le service **Telegraf** est : **sudo systemctl restart telegraf**.

La commande pour vérifier que **Telegraf** a bien redémarré et qu'aucune erreur de configuration n'apparaît est : **sudo systemctl status telegraf**.

```
root@ubuntu:/home/eleve# sudo systemctl restart telegraf
root@ubuntu:/home/eleve# sudo systemctl status telegraf
● telegraf.service - Telegraf
     Loaded: loaded (/usr/lib/systemd/system/telegraf.service; enabled; preset: enabled)
     Active: active (running) since Sun 2026-09-20 15:01:42 UTC; 11s ago
 Invocation: fd2a901b43ab4bcaa770b5be3b511f5c
       Docs: https://github.com/influxdata/telegraf
   Main PID: 5312 (telegraf)
      Tasks: 6 (limit: 1888)
     Memory: 24.3M (peak: 24.3M)
        CPU: 105ms
     CGroup: /system.slice/telegraf.service
             └─5312 /usr/bin/telegraf -config /etc/telegraf/telegraf.conf -config-directory /etc/telegraf/telegraf.d

sept. 20 15:01:42 ubuntu telegraf[5312]: 2026-09-20T15:01:42Z I! Loaded inputs: mqtt_consumer
sept. 20 15:01:42 ubuntu telegraf[5312]: 2026-09-20T15:01:42Z I! Loaded aggregators:
sept. 20 15:01:42 ubuntu telegraf[5312]: 2026-09-20T15:01:42Z I! Loaded processors:
sept. 20 15:01:42 ubuntu telegraf[5312]: 2026-09-20T15:01:42Z I! Loaded secretstores:
sept. 20 15:01:42 ubuntu telegraf[5312]: 2026-09-20T15:01:42Z I! Loaded outputs: influxdb_v2
sept. 20 15:01:42 ubuntu telegraf[5312]: 2026-09-20T15:01:42Z I! Tags enabled: host=ubuntu
sept. 20 15:01:42 ubuntu systemd[1]: Started telegraf.service - Telegraf.
sept. 20 15:01:42 ubuntu telegraf[5312]: 2026-09-20T15:01:42Z I! [agent] Config: Interval:10s, Quiet:false, Hostname:"ubuntu", Flush Interval:10s
sept. 20 15:01:42 ubuntu telegraf[5312]: 2026-09-20T15:01:42Z W! [agent] The default value of 'skip_processors_after_aggregators' will change to 'true' with Te
sept. 20 15:01:42 ubuntu telegraf[5312]: 2026-09-20T15:01:42Z I! [inputs.mqtt_consumer] Connected [tcp://192.168.2.100:1883]
lines 1-22/22 (END)
```

XI – VERIFICATION ET TRAITEMENT DES DONNEES AVEC FLUX :

Nous allons maintenant vérifier que **Telegraf** récupère bien les données et les envoie à **InfluxDB**.

1. Vérification des Logs système :

Pour vérifier que **Telegraf** fonctionne convenablement, taper la commande suivante :

```
sudo journalctl -u telegraf -f -n 20
```

Donner une capture d'écran des Logs de **Telegraf** et analyser ces Logs.

```
root@ubuntu:/home/eleve# sudo journalctl -u telegraf -f -n 20
sept. 20 15:01:42 ubuntu systemd[1]: Stopping telegraf.service - Telegraf...
sept. 20 15:01:42 ubuntu telegraf[5254]: 2026-09-20T15:01:42Z I! [agent] Stopping running outputs
sept. 20 15:01:42 ubuntu systemd[1]: telegraf.service: Deactivated successfully.
sept. 20 15:01:42 ubuntu systemd[1]: Stopped telegraf.service - Telegraf.
sept. 20 15:01:42 ubuntu systemd[1]: Starting telegraf.service - Telegraf...
sept. 20 15:01:42 ubuntu (telegraf)[5312]: telegraf.service: Referenced but unset environment variable evaluates to an empty string: TELEGRAF_OPTS
sept. 20 15:01:42 ubuntu telegraf[5312]: 2026-09-20T15:01:42Z W! Strict environment variable handling is the new default starting with v1.38.0! If your configur
ation does not work with strict handling please explicitly add the --non-strict-env-handling flag to switch to the previous behavior!
sept. 20 15:01:42 ubuntu telegraf[5312]: 2026-09-20T15:01:42Z I! Loading config: /etc/telegraf/telegraf.conf
sept. 20 15:01:42 ubuntu telegraf[5312]: 2026-09-20T15:01:42Z I! Starting Telegraf 1.40.0 brought to you by InfluxData the makers of InfluxDB
sept. 20 15:01:42 ubuntu telegraf[5312]: 2026-09-20T15:01:42Z I! Available plugins: 244 inputs, 9 aggregators, 35 processors, 26 parsers, 68 outputs, 8 secret s
tores
sept. 20 15:01:42 ubuntu telegraf[5312]: 2026-09-20T15:01:42Z I! Loaded inputs: mqtt_consumer
sept. 20 15:01:42 ubuntu telegraf[5312]: 2026-09-20T15:01:42Z I! Loaded aggregators:
sept. 20 15:01:42 ubuntu telegraf[5312]: 2026-09-20T15:01:42Z I! Loaded processors:
sept. 20 15:01:42 ubuntu telegraf[5312]: 2026-09-20T15:01:42Z I! Loaded secretstores:
sept. 20 15:01:42 ubuntu telegraf[5312]: 2026-09-20T15:01:42Z I! Loaded outputs: influxdb_v2
sept. 20 15:01:42 ubuntu telegraf[5312]: 2026-09-20T15:01:42Z I! Tags enabled: host=ubuntu
sept. 20 15:01:42 ubuntu systemd[1]: Started telegraf.service - Telegraf.
sept. 20 15:01:42 ubuntu telegraf[5312]: 2026-09-20T15:01:42Z I! [agent] Config: Interval:10s, Quiet:false, Hostname:"ubuntu", Flush Interval:10s
sept. 20 15:01:42 ubuntu telegraf[5312]: 2026-09-20T15:01:42Z W! [agent] The default value of 'skip_processors_after_aggregators' will change to 'true' with Tel
egraf v1.40.0! If you need the current default behavior, please explicitly set the option to 'false'!
sept. 20 15:01:42 ubuntu telegraf[5312]: 2026-09-20T15:01:42Z I! [inputs.mqtt_consumer] Connected [tcp://192.168.2.100:1883]
```

L'analyse de la capture d'écran confirme le bon fonctionnement du pipeline d'ingestion :

1. Lecture de la configuration : **Telegraf** charge bien le fichier personnalisé **/etc/telegraf/telegraf.conf**.
2. Chargement des plugins (**ETL**) :
   - Input : **mqtt_consumer** est chargé pour intercepter les messages **MQTT** de la ligne de production.
   - Output : **influxdb_v2** est chargé pour envoyer les métriques traitées vers la base de données temporelle.

3. Démarrage du service : **Systemd** valide le lancement du daemon (**Started telegraf.service**).

4. Connexion au Broker **MQTT** : La ligne finale **[inputs.mqtt_consumer] Connected [tcp://192.168.2.100:1883]** confirme que Telegraf communique avec le broker **Mosquitto** sur l'adresse IP 192.168.2.100.

Remarque sur les avertissements (**W!**) : Les lignes marquées d'un **W!** sont simplement des avertissements concernant les évolutions futures des variables d'environnement dans les prochaines versions de **Telegraf**. Elles n'impactent en rien le fonctionnement de l'infrastructure.

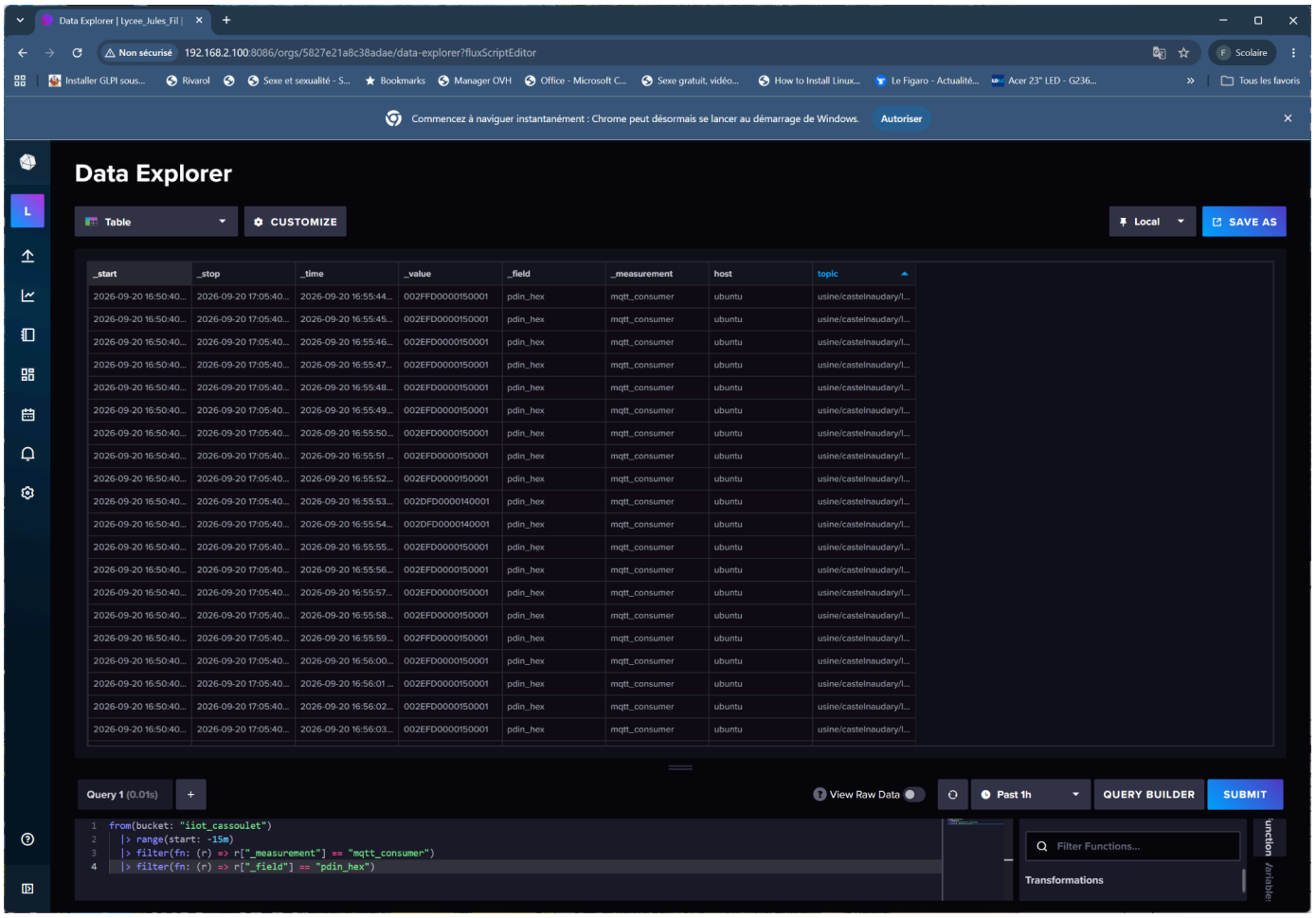
2. Visualisation directe des données dans l'interface Web d'InfluxDB v2 :

Il est maintenant possible de visualiser les données enregistrées dans **InfluxDB v2**. Pour cela, vous allez devoir choisir votre Bucket puis la sélection de la mesure. Le mieux est d'utiliser une requête de type script, taper cette requête dans Script Editor :

```
from(bucket: "iiot_cassoulet")
 |> range(start: -15m)
 |> filter(fn: (r) => r["_measurement"] == "mqtt_consumer")
 |> filter(fn: (r) => r["_field"] == "pdin_hex")
```

Expliquer la procédure permettant de contrôler et de visualiser les données ingérées directement depuis l'interface Web d'**InfluxDB v2** sans passer par un outil tiers. Donner une capture d'écran des données enregistrées dans **InfluxDB v2** sous forme de **Table**.

1. Accès au Data Explorer :
   - Connexion à l'interface Web via l'adresse : **http://192.168.2.100:8086**
   - Sélection du menu **Data Explorer** : Icône d'histogramme à gauche
2. Filtrage via le constructeur visuel (**Query Builder**) :
   - Sélection du bucket : **iiot_cassoulet**
   - Sélection de la mesure (**_measurement**) : **mqtt_consumer**
   - Sélection du champ (**_field**) : **pdin_hex** ou **compteur_event**
3. Modes de représentation :
   - Vue Graphique (**Graph**) : Permet de suivre l'évolution temporelle de la réception des messages
   - Vue Tableau (**Table View**) : Permet de lire directement la valeur brute des trames hexadécimales enregistrées horodatées
4. Création d'un Dashboard d'appoint :
   - En cliquant sur « **Save As** » ➔ « **Dashboard Cell** », il est possible de créer un tableau de bord de contrôle rapide dans **InfluxDB** pour valider la continuité de l'ingestion



Expliquer pourquoi les données peuvent s'afficher dans une **Table** mais pas dans un **Graph**.

L'impossibilité d'afficher les données dans un **Graph** alors qu'elles apparaissent parfaitement dans une **Table** est liée au type de données traité par InfluxDB :

- Le type de données (**String** vs **Numérique**) : Le champ **pdin_hex** contient des chaînes de caractères (du texte brut comme "**002EFD0000150001**").
- Le fonctionnement du **Graph** (Graphique) : Un graphique trace des points dans un repère orthogonal. Il requiert impérativement des valeurs numériques (**int** ou **float**) sur l'axe vertical (**Y**) pour calculer des coordonnées, échelonner une échelle et relier des points. Il ne peut pas positionner du texte sur un axe quantitatif.
- Le fonctionnement de la **Table** : Un tableau est un simple conteneur de restitution brute. Il affiche n'importe quel type de donnée (texte, nombre, booléen, horodatage) sous forme de lignes et de colonnes, sans imposer de représentation géométrique ni de calcul mathématique.

En résumé : La **Table** restitue du texte brut, le **Graph** exige des nombres pour convertir des mesures en coordonnées. Pour tracer une courbe sur le graphique, **InfluxDB** doit recevoir une donnée numérique (comme le champ **compteur_event** ou la distance convertie en millimètres).

3. Simulation d'encrassement de la lentille et maintenance prédictive :

Nous avons simulé un encrassement de l'optique du capteur IFM OGD592 par la pose d'un ruban adhésif translucide (scotch) sur la fenêtre optique. Ce dispositif atténue le faisceau laser émis et réduit l'intensité du signal lumineux retourné au récepteur, sans couper intégralement la mesure.

Nous relevons ces données dans la table **InfluxDB** :

2026-09-20 17:23:54...	2026-09-20 17:38:54...	2026-09-20 17:38:35...	0021FD00004A0001	pdin_hex	mqtt_consumer	ubuntu	usine/castelnaudary/l...
2026-09-20 17:23:54...	2026-09-20 17:38:54...	2026-09-20 17:38:36...	0066FD0000050001	pdin_hex	mqtt_consumer	ubuntu	usine/castelnaudary/l...

Expliquer comment ces données mettent en évidence l'encrassement du capteur et conclure sur l'intérêt d'avoir une maintenance prédictive avec ce type de données.

Ces données mettent en évidence l'encrassement du capteur à travers trois indicateurs clés :

1. La chute brutale du taux de rémission (Octets 5 et 6) : La réflectivité de la cible est codée sur les caractères 9 à 12 de la trame **pdin_hex**
   - A 17:38:35, les octets **5–6** affichent **004A**, soit une réflectivité nominale de **74 %** (4×16+10)
   - A 17:38:36, suite à la pose du scotch, ces octets chutent à **0005**, soit **5 %** de rémission
   - Cette perte de 69 points de rémission prouve que le ruban adhésif atténue fortement l'intensité du faisceau laser retourné au récepteur
2. La persistance de la mesure de distance : Malgré l'atténuation du signal optique, le capteur continue de renvoyer une mesure de distance (Octets 1 et 2 passant de **0021** / **33 mm** à **0066** / **102 mm**). Le faisceau n'est donc pas totalement coupé, ce qui confirme un phénomène d'encrassement et non une obstruction totale
3. Indicateur de maintenance prédictive : Sur une ligne d'industrialisation, l'analyse en temps réel de ces deux octets dans **InfluxDB** permet de détecter la dégradation progressive de l'optique. La définition d'un seuil d'alerte (par exemple dès que la rémission descend en dessous de **20 %**) permet de planifier un nettoyage préventif du capteur avant que la mesure de distance ne devienne impossible.

Grille de Notation			Validation Compétences
Etude préalable de la persistance des données		**/4**	
Installation et Configuration d'**InfluxDB v2**		**/4**	
Installation et Déploiement de **Telegraf**		**/3**	**Valide C05**
Ingestion **MQTT**, Parsing **JSON** et Requêtes **Flux**		**/3**	
Visualisation **Table**/**Graph** et Maintenance Prédictive		**/6**	**Valide C04**
Barème de notation total		**/20**	