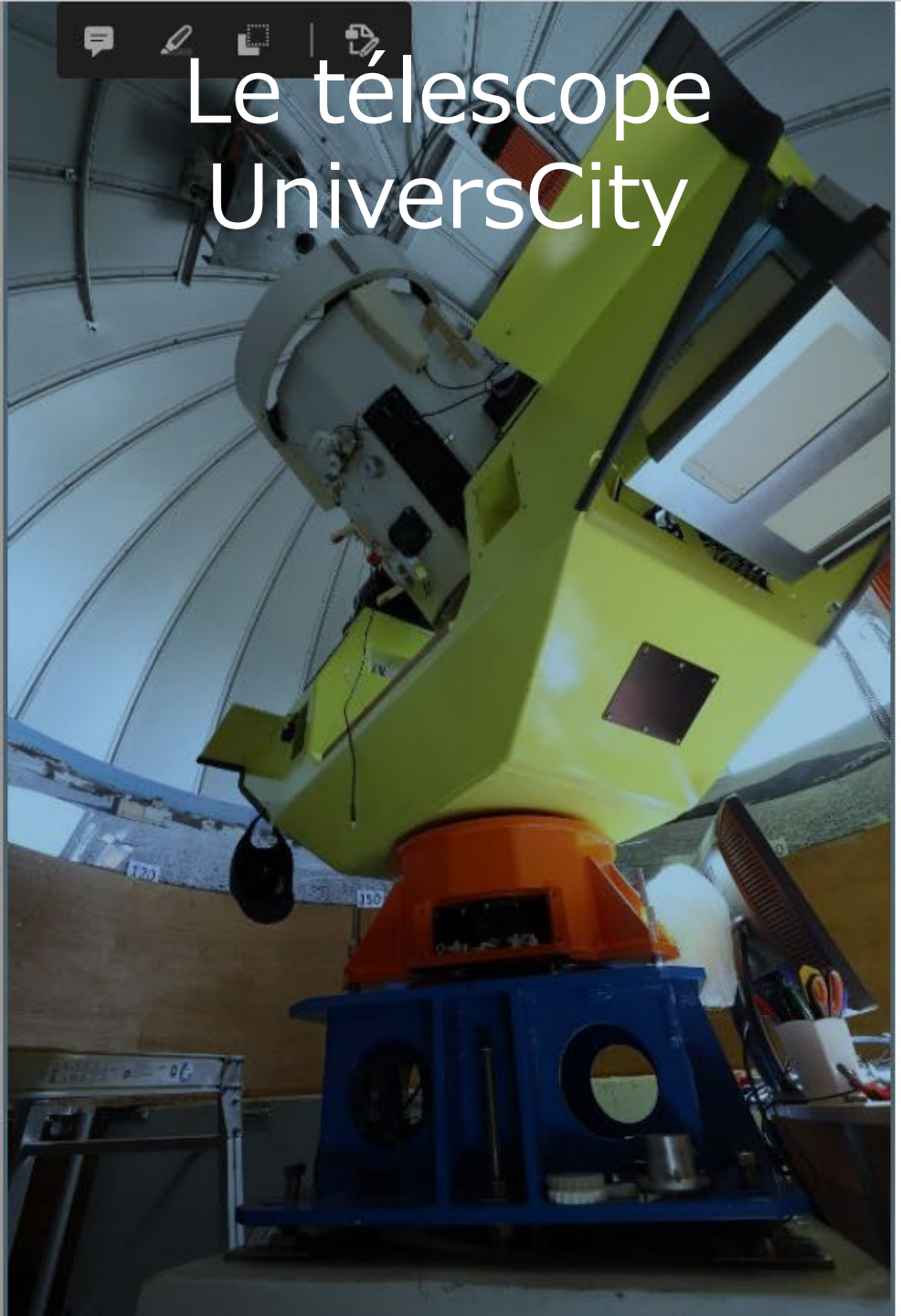
ANF IoT - REX Environnement logiciel pour projet d'automatisation de télescope

Olivier Boebion – olivier.boebion@oca.eu
Laboratoire Lagrange / GIR - OCA

27 octobre 2022

Contexte



Le télescope UniversCity

Caractéristiques techniques (I)

- Localisation
  - Plateau de Caussols (Calern) 43°45'9.15"N 6°55'18.42"E
- Monture (Alcor Systems)
  - Type : Alt-Azimutal
  - Moteur couple (direct drive) sur les 2 axes (poursuite jusqu'à ~15°/s)
  - Capacité charge : 630kg mobile, 50kg/plateforme
  - 2 plateformes Nasmyth (voie grand champ et voie Haute résolution)
- Coupole (AshDome)
  - 2 axes azimuth et hauteur (fenêtre d'observation)
- Bâtiment sur 2 niveaux
  - RdC : salle de contrôle
  - étage : instrumentation

Point de départ

Télescope en fonctionnement manuel et à distance depuis l’été 2021

Automatisation des observations souhaitée

- Gestion des risques et des pannes
- Contrôle de la coupole, du télescope et des caméras
- Planification dynamique des observations
- Traitement et mise à disposition des données d’observation

Volonté de définir un environnement logiciel reproductible

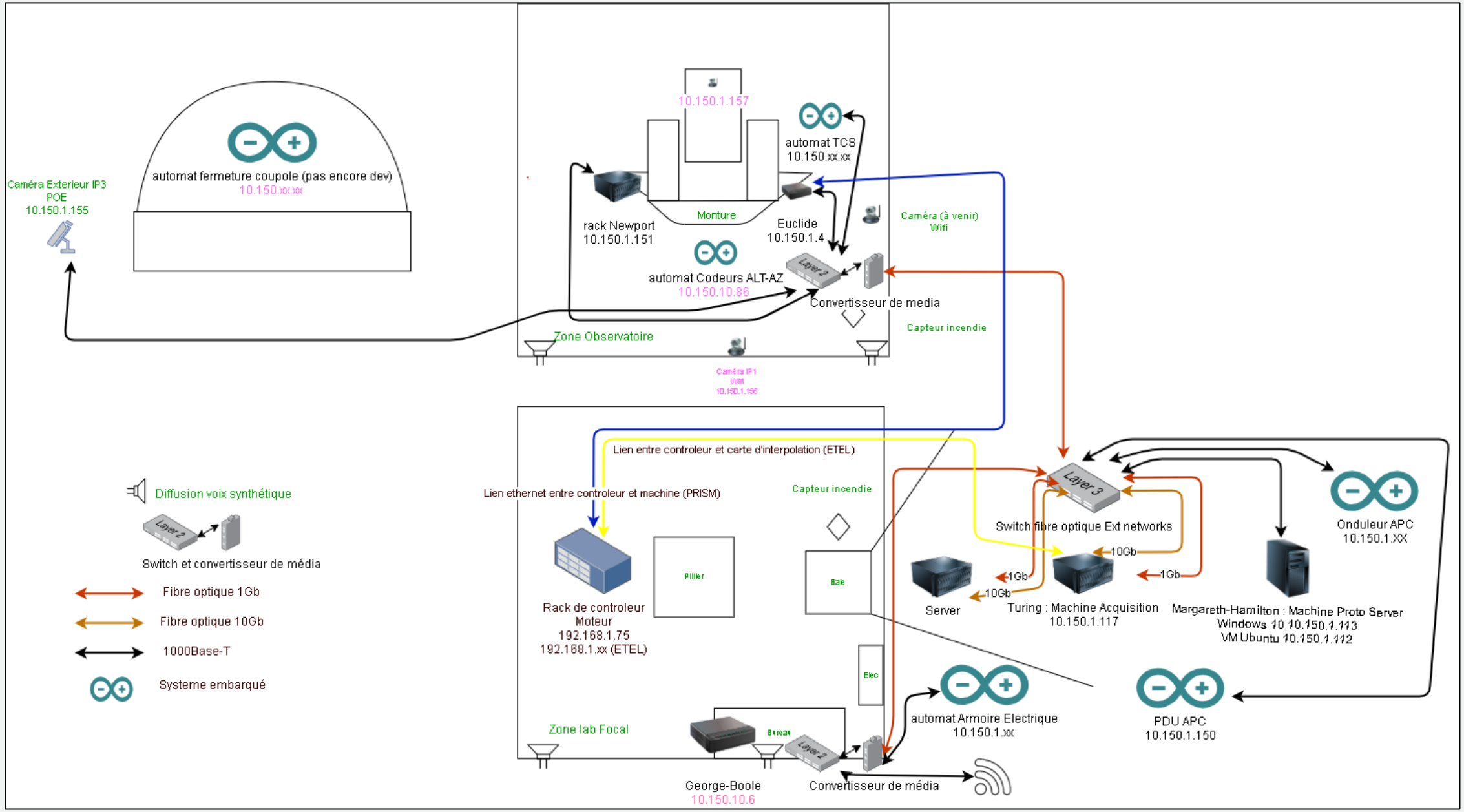


Schéma des connexions réseaux d'UniversCity (Erick Bondoux – OGS Technologies)

Gestion des risques et des pannes

- Collecte des données depuis des capteurs environnementaux (température, pression, humidité, détecteur de foudre à faible et grande distance, positions de l'instrument, consommations électriques…)
- Stockage des données
- Supervision/visualisation des données
- Envoi/réception d'alarmes ou d'états
- Contrôle/commande

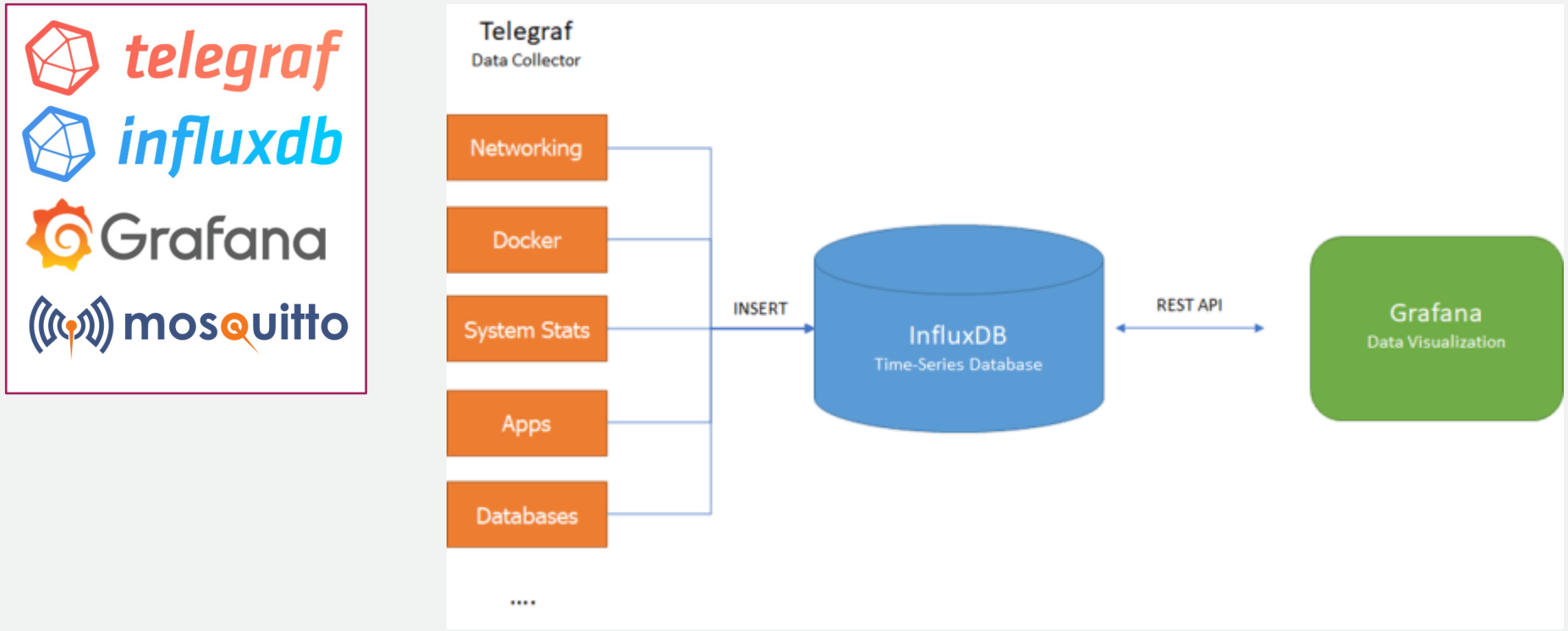
Quelques matériels à noter

Controllino MEGA pure 100-200-10

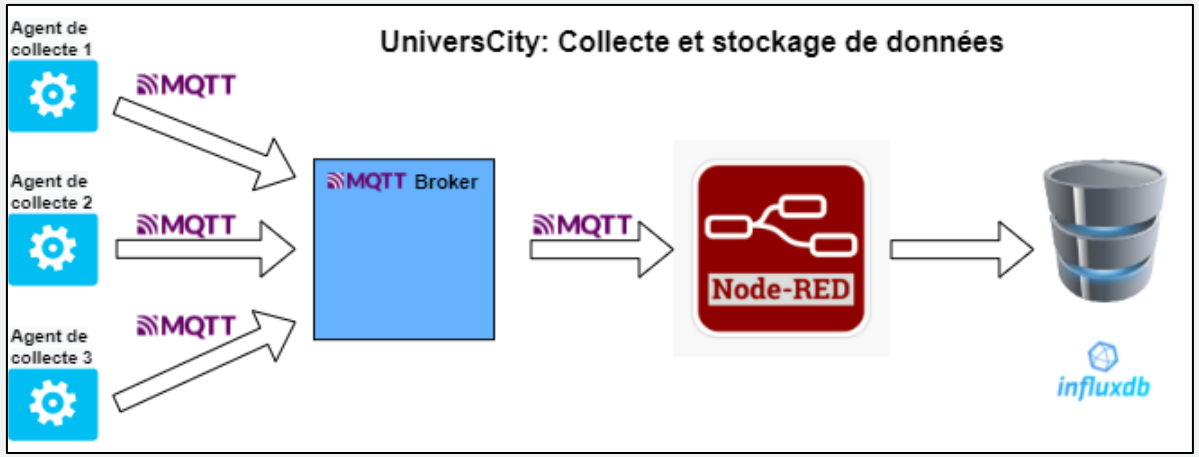
Capteurs météo Vaisala WXT 530 reliées en liaison radio par modules DIGI Xbee 868Mhz

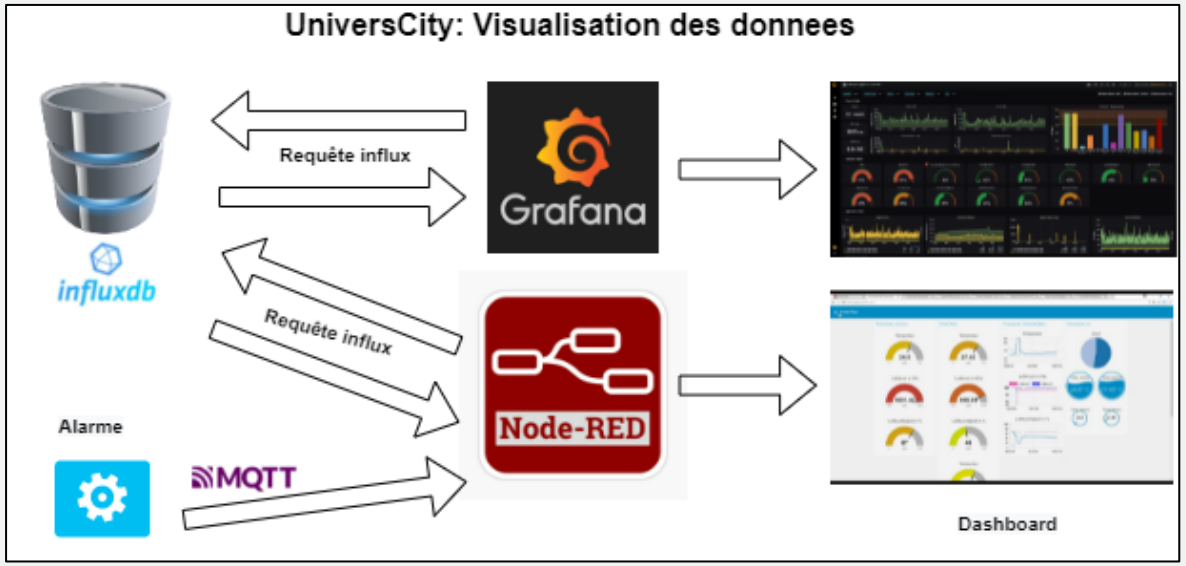
Environnement logiciel mis en place

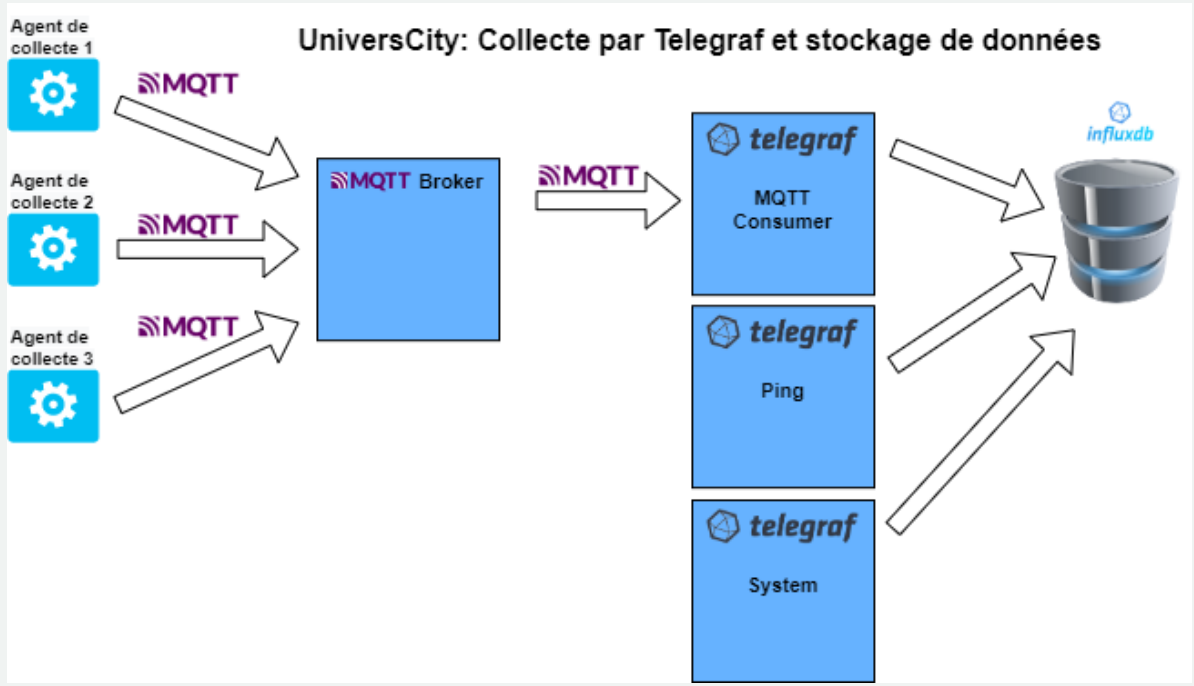
TIG + Mosquitto MQTT

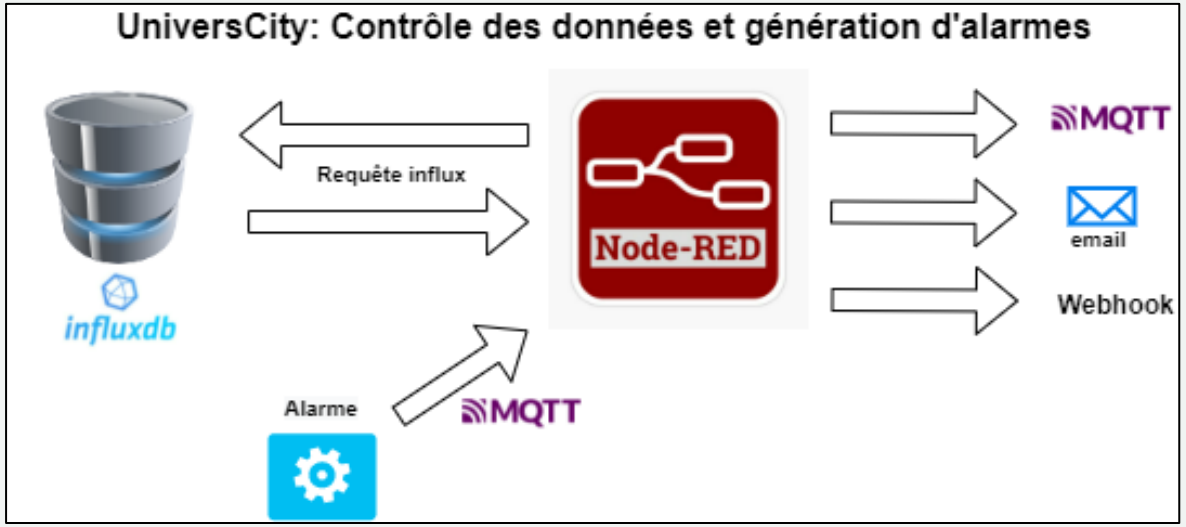


Collecte, contrôle et visualisation des données environnementales d'UniversCity

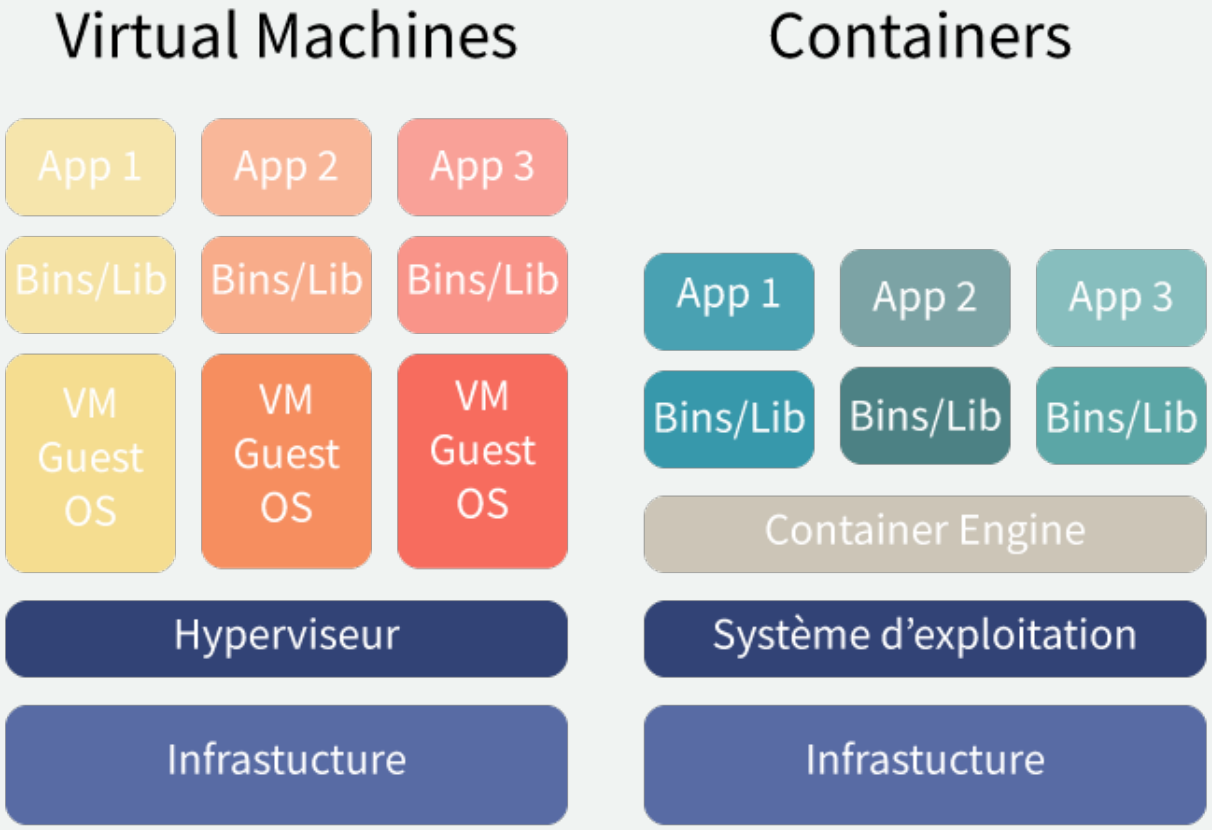








Containers ?


Virtual Machines
App 1
App 2
App 3
Bins/Lib
Bins/Lib
Bins/Lib
VM Guest OS
VM Guest OS
VM Guest OS
Hyperviseur
Infrastucture
Containers
App 1
App 2
App 3
Bins/Lib
Bins/Lib
Bins/Lib
Container Engine
Système d'exploitation
Infrastucture

Déploiement

Mosquitto MQTT Broker

Version 3.1

Sans TLS

QOS 0

Avec authentification

ACL simple

- seul les clients MQTT authentifiés peuvent publier
- souscription autorisées pour les connexions en anonyme

InfluxDB

InfluxDB existe en 2 versions majeures: 1.x and 2.x depuis novembre 2020

Attention aux anciennes pages web, en particulier:

L’identification des utilisateurs et de leur droits associés: API Tokens (v2)

La syntaxe influxdb CLI

Le langage de requêtes InfluxDB: influx (v2) / influxQL.

Référence :

https://www.sqlpac.com/fr/documents/influxdb-migration-v1-v2-procedure.html

Node-RED et Grafana

Node-RED

- Sécurisation de l’accès à Node-Red
- Possibilité de déposer les versions des flows sur un dépôt GIT

Grafana

- Pas de soucis particulier à déployer

Travail réalisé

- Services MQTT Broker, InfluxDB, Telegraf, Grafana et application Node-RED fonctionnels
- Collecte et affichage de données du Dome, de la Monture, système et réseaux
- Collecte de données du PDU APC par application container Python (à finaliser)
- Déploiement de l’environnement par docker-compose

Et maintenant ?

Finaliser l’environnement logiciel

- Ecriture d’un guide d’administration (Arrêt/Démarrage des containers, Logs, modification d’une configuration, usage de Rundeck …)
- Sauvegardes à mettre en place
- Ecriture d'un plan de reprise d'activité

Finaliser la collecte de données

- Collecte de données météo, postes informatiques par Telegraf, automates, onduleur APC, ⋯
- Accès à la caméra All Sky et aux données météos du plateau de Calern

- Suite à la formation : Passage en connexion TLS ? Messages MQTT en QOS 2 ? Utilisation du mode Testament ?

Surveiller, contrôler et piloter

Dashboard Grafana et Node-RED

Contrôle alimentation électrique (PDU, onduleur, disjonteur, informatique)

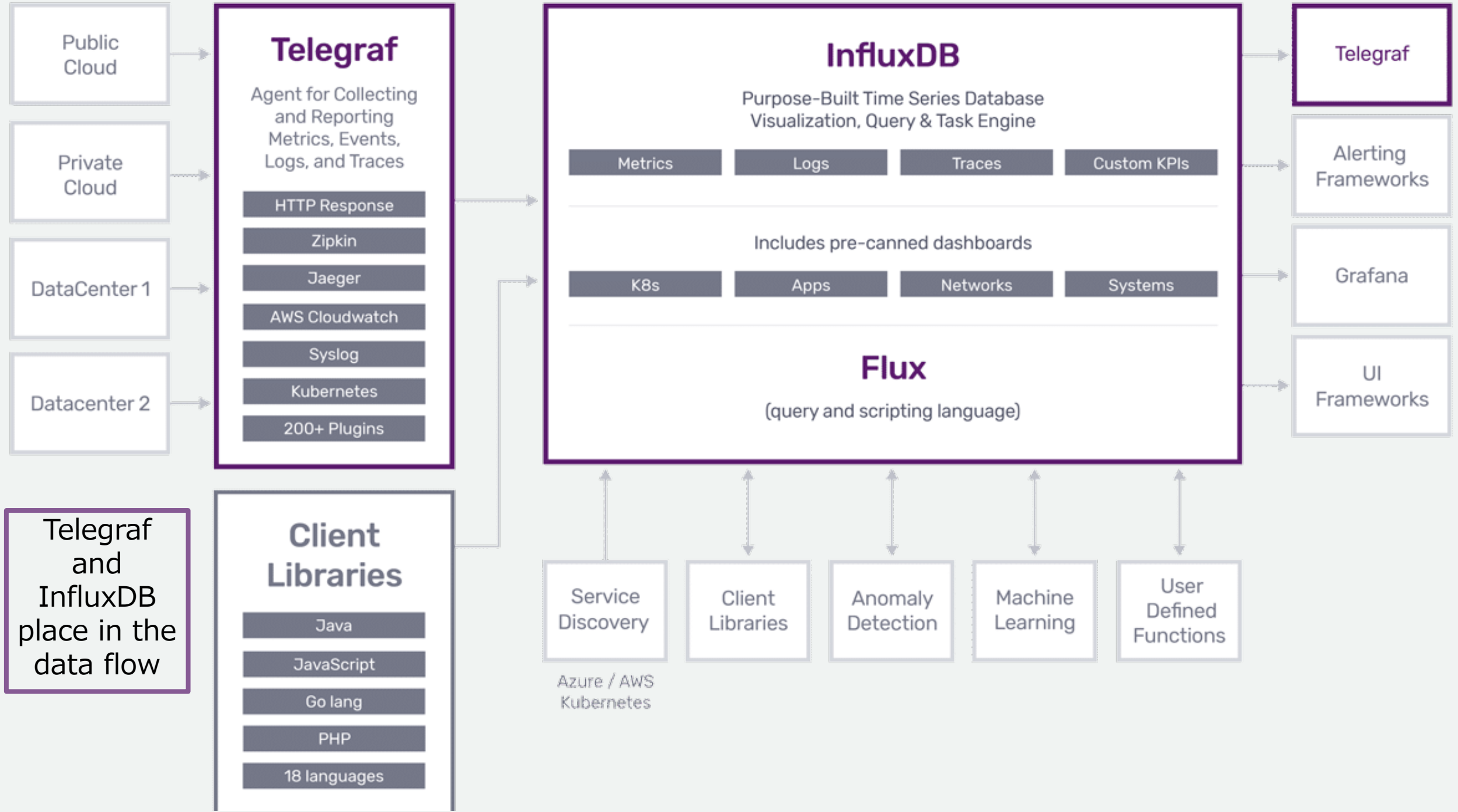
Formaliser les seuils, les alarmes et l’interface avec les logiciels d’observation

InfluxDB

What is InfluxDB ?

"A platform for building time series applications"

- Developed by the company InfluxData (initiated by Errplane company in 2013) https://www.influxdata.com/
- Written in Go programming language
- Optimized for storage and retrieval of time series data in fields such as operations monitoring, application metrics, IoT sensor data, and real-time analytics


Public Cloud
Private Cloud
DataCenter 1
Datacenter 2
Telegraf
Agent for Collecting and Reporting Metrics, Events, Logs, and Traces
HTTP Response
Zipkin
Jaeger
AWS Cloudwatch
Syslog
Kubernetes
200+ Plugins
Client Libraries
Java
JavaScript
Go lang
PHP
18 languages
InfluxDB
Purpose-Built Time Series Database
Visualization, Query & Task Engine
Metrics
Logs
Traces
Custom KPIs
Includes pre-canned dashboards
K8s
Apps
Networks
Systems
Flux
(query and scripting language)
Telegraf
Alerting Frameworks
Grafana
UI Frameworks
Service Discovery
Client Libraries
Anomaly Detection
Machine Learning
User Defined Functions
Azure / AWS Kubernetes
Telegraf and InfluxDB place in the data flow

InfluxDB terminology (1/2)

Organization : a workspace for a group of users. All dashboards, tasks, buckets, members, etc., belong to an organization

Bucket : a named location where time series data is stored. All buckets have a **retention period**

Timestamp : all data stored has a timestamp in UTC format store in a _time column

Measurement : the name of the measurement in the _measurement column

InfluxDB terminology (2/2)

Field : a field includes a field key stored in the _field column and a field value stored in the _value columnSerie

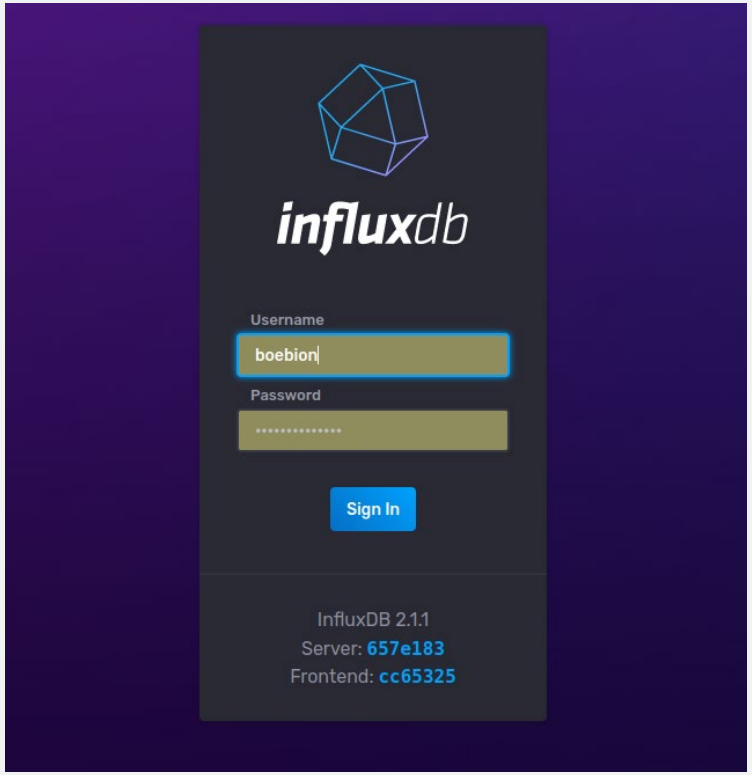
Tag : a tag include tag keys and tag values that are stored as strings and metadata.

Serie : includes timestamps and field values for a given series key

Point : includes the series key, a field value, and a timestamp.

InfluxDB UI overview

Demo



Write data with Python

```
#!/usr/bin/env python3

import logging
from datetime import datetime, timezone
import time

import influxdb_client
from influxdb_client.client.influxdb_client import InfluxDBClient
from influxdb_client.client.write_api import SYNCHRONOUS
from influxdb_client.domain.write_precision import WritePrecision

bucket = "Python"
org = "GIR"
token = "PXEP4XZMrp_AzLRkSk6Ja3dWI5_STC7yvVTJCRoMKBTX_PP1NThLxI37WBKgU4jiFmyp6q9sqm57Nw6OL3xCdw=="
# Store the URL of your InfluxDB instance
url = "http://localhost:8086"

try:
    with influxdb_client.InfluxDBClient(url=url, token=token, org=org) as client:
        write_api = client.write_api(write_options=SYNCHRONOUS)

        while True:
            p = influxdb_client.Point("meteo").tag("location", "Nice").field("temperature", 22.87)
            write_api.write(bucket=bucket, org=org, record=p)

            write_api.flush()

            time.sleep(1)

            p = influxdb_client.Point("meteo").tag("location", "Marseille").field("temperature", 19.4)
            write_api.write(bucket=bucket, org=org, record=p)

            write_api.flush()

            time.sleep(1)
            sequence = ["meteo,location=Nice temperature=23.43234543", "meteo,location=Marseille temperature=24", "meteo,location=Paris
temperature=25.0"]
            write_api.write(bucket=bucket, org=org, record=sequence)

            write_api.flush()
            time.sleep(10)

except KeyboardInterrupt:
    print("Program stopped by keyboard interrupt")
finally:
    print("Program ending")
    write_api.close()
    client.close()
```