



**Biodiversity
Greece**

LIFE EL-BIOS
Hellenic Biodiversity Information System
www.biodiversity-greece.gr

Tel: +30 210 5241903 (int.: 129)
Email: info@biodiversity-greece.gr



PROJECT	LIFE18 GIE/GR/000899
PROJECT TITLE	Hellenic Biodiversity Information System: An innovative tool for biodiversity conservation.
ACRONYM	LIFE EL-BIOS
ACTION	B4
DELIVERABLE NUMBER	B4.2
DELIVERABLE TITLE	Replication Guide 2 [UTH-DECE]: Cloud data center system design, for archiving, processing and analyzing environmental data collected through wireless sensor nodes with AI algorithms
REPORT STATUS	Complete
COMPLETION DATE	31/12/2024
RESPONSIBLE	University of Thessaly



LIFE EL-BIOS (LIFE20 GIE/GR/001317) has received funding from the LIFE Programme of the European Union.
EU funding contribution: 1.354.524 € (52.68% of total eligible budget).



LIFE EL-BIOS
has received funding from GREEN FUND

COORDINATING
BENEFICIARY



ARISTOTLE
UNIVERSITY
OF THESSALONIKI



THE GOULANDRIS NATURAL HISTORY MUSEUM
GREEK BIOTOPE/WETLAND CENTRE

ASSOCIATED
BENEFICIARIES



AUTHORS:

- Pavlidis Effraim, Pavlidis Effraim, Stamopoulos Evaggelos, Psarouda Varvara, Maroulis Emanouil, Choumas Konstantinos, Oikonomou Fotini, Gabriilidis Orestis

WITH THE CONTRIBUTION OF:

- Korakis Athanasios, Professor UTH-DECE

DOCUMENT TITLE:

Cloud data center system design, for archiving, processing and analyzing environmental data collected through wireless sensor nodes with AI algorithms.

This publication reflects only the views of the authors. The Executive Agency for Climate, Infrastructure and Environment (CINEA) and the European Commission cannot be held responsible for any use which may be made of the information contained therein

SUGGESTED CITATION:

- Influx DB documentation
- MQTT protocol documentation
- HTTP and HTTPs protocols documentation
- Computer Networking, A Top-Down Analysis (Kurose|Rose)
- Python Data Analytics, Data Analysis and Science Using Pandas, matplotlib, and the Python Programming Language (Fabio Nelli)

TABLE OF CONTENTS

Acronyms and Abbreviations Lists.....	4
Executive Summary.....	5
1. Architecture.....	5
2. Key Features.....	5
3. System Requirements.....	6
4. Dependencies.....	7
5. Deployment Details.....	7

TABLE OF FIGURES

ACRONYMS & ABBREVIATIONS LIST

ENGLISH	
WSN	Wireless Sensor Network
3G	3rd Generation
IoT	Internet of Things
4G	4th Generation
5G	5th Generation
MQTT	Message Queuing Telemetry Transport
HTTP	Hypertext Transfer Protocol
DB	Database
API	Application Programming Interface

EXECUTIVE SUMMARY

The system consists of backend services that process data from distributed sensors deployed in wild environments, storing them along with metadata in a time-series database and exposing the data via an API. The project is divided into two distinct deployments.

Valia Kalda Deployment is a network of multiple distributed environmental, camera, and audio sensors deployed in a forested area. These sensors communicate their respective data through MQTT over a 4G connection.

Drakolimni Deployment is a standalone sensor package deployment that sends its data directly to its receiving backend service over HTTP using a 4G connection.

1. Architecture

Valia Kalda Deployment Components

InfluxDB v1.8: InfluxDB serves as the time-series database and metadata store for Valia Kalda Deployment. It is containerized for ease of deployment and configuration.

Docker Image: [InfluxDB v1.8](#)

elbios-mqtt-client: This service subscribes to special MQTT topics to collect data sent by the sensors. It processes the incoming data and writes it to the connected Influx DB instance in real-time.

elbios-backend-api: This service exposes the stored data and metadata in Influx DB through an API. It acts as the primary interface for external systems and consumers.

Drakolimni Deployment Components

elbios-influx-proxy: The elbios-influx-proxy service exposes an HTTP API endpoint to receive time-series data from the sensor package in Drakolimni Deployment. It connects directly to the Influx DB instance to store the incoming data.

2. Key Features

The architecture for sensor data management encompasses a robust, multi-faceted approach, tailored to the specific operational requirements of each deployment.

For the Valia Kalda deployment, an asynchronous Distributed Sensor Data Collection mechanism is implemented, leveraging the Message Queuing Telemetry Transport (MQTT) protocol. This choice

ensures highly efficient, low-bandwidth, and reliable communication, critical for environments with potentially intermittent network connectivity.

Conversely, the Drakolimni deployment employs a direct HTTP-based communication protocol for its data collection, prioritizing architectural simplicity and ease of integration.

Subsequent to data acquisition, both deployments utilize Influx DB version 1.8 for Time-Series Data Storage. This purpose-built database solution offers optimized storage, high ingestion rates, and efficient querying capabilities specifically designed for high-frequency time-series datasets, ensuring data integrity and accessibility.

Finally, an API-First Data Exposure strategy is employed, providing comprehensive access to both raw sensor data and associated metadata through a suite of well-documented RESTful Application Programming Interfaces (APIs). This standardized interface facilitates seamless integration with various external systems, third-party applications, and advanced analytical tools, promoting interoperability and data utilization across diverse platforms.

3. System Requirements

The foundational infrastructure supporting these deployments is designed for portability and robust connectivity. For the operational environment of the Influx DB instances, a container runtime environment (Docker) is utilized. This approach ensures consistent deployment, isolation, and efficient resource management for the database service across different host systems. Regarding network connectivity, the Valia Kalda deployment relies on MQTT over a 4G cellular network for its data transmission, capitalizing on the protocol's lightweight nature and the widespread availability of 4G infrastructure to ensure reliability, even in challenging remote conditions. Similarly, the Drakolimni deployment employs HTTP over a 4G cellular network for its communication needs, leveraging the ubiquitous nature of HTTP and the robust bandwidth provided by 4G to facilitate direct and straightforward data exchange.

4. Dependencies

The system's operational framework is built upon clearly defined dependencies and a strategic choice of programming languages.

The core time-series data management relies exclusively on Influx DB version 1.8, which serves as the primary data store. For consistent and reproducible deployment of Influx DB, the official Docker image for Influx DB v1.8 is utilized, ensuring environmental isolation and simplified orchestration.

Complementing this, the backend services, which manage data processing, API endpoints, and system logic, are robustly implemented using a combination of the Go programming language and JavaScript. Go's efficiency, concurrency features, and strong typing are leveraged for high-performance and scalable server-side operations, while JavaScript is employed for its versatility, particularly in areas requiring agile development and integration within modern web-centric ecosystems.

5. Deployment Architecture and Procedures

The deployment strategy for the sensor data collection infrastructure is meticulously defined for each distinct geographical location, ensuring optimal performance and reliable data flow. Both the Valia Kalda and Drakolimni deployments adhere to a containerized approach for core services, promoting consistency and scalability.

Valia Kalda Deployment Details:

For the Valia Kalda site, the Influx DB v1.8 instance is containerized and initiated using the standard Docker command:

```
docker run -d --name influxdb -p 8086:8086 influxdb:1.8.
```

This establishes a persistent, background-running container named influxdb, mapping the container's default HTTP API port (8086) to the host system. Post-deployment, the InfluxDB instance is rigorously configured to securely accept incoming time-series data from the elbios-mqtt-client service via its exposed HTTP interface, ensuring seamless data ingestion.



The elbios-mqtt-client service, responsible for interfacing with the distributed sensors, necessitates precise configuration. Critical MQTT broker connection parameters, including subscription topics, authentication credentials, and endpoint addresses, are meticulously defined. These configurations are typically managed through robust environment variables or dedicated configuration files, adhering to best practices for secure and adaptable service deployment.

Finally, the elbios-backend-api service is deployed and configured to establish a persistent connection with the aforementioned InfluxDB instance. Its primary function is to expose a comprehensive suite of RESTful API endpoints, enabling secure and efficient querying of the stored time-series data for external consumption, analysis, and visualization.

Drakolimni Deployment Details:

The Drakolimni deployment mirrors the containerization strategy for its database component. The InfluxDB v1.8 setup is executed in a manner identical to the Valia Kalda deployment, utilizing the official Docker image to ensure a consistent and isolated database environment.

The distinct component for the Drakolimni site is the elbios-influx-proxy service. Upon deployment, this service is configured to actively listen on a designated HTTP port, acting as an intermediary for data ingress. It is subsequently configured to establish a secure and reliable connection with the local InfluxDB instance, facilitating the storage of incoming HTTP-based sensor data. This proxy layer ensures that all data traffic is appropriately handled and directed to the time-series database.