Wireless Inexpensive Bathymetric Logger (WIBL)
Cloud Frontend Integration

Thomas Ackerly,Timofei Nikshych, Chris Sullivan and Cole Glennon

Special thanks to Dr. Brian Calder (CCOM/JHC), Dr. Brian Miles (CCOM/JHC), and Dr. Matthew Plumlee(UNH CEPS)

Department of Computer Science, University of New Hampshire

Introduction

WIBL is part of a seafloor digitization project that has already accomplished making a cost effective logger to collect bathymetric data. Currently, uploading data into cloud involves the use of a command prompt. Because this can be lengthy and error-prone for both experienced and laymen users, our capstone is currently developing a Graphical User Interface (GUI) to automate the process. The goal is to cut the total time to handle files down by at least 10%.

This GUI is designed for use by *trusted nodes*, independent volunteer organizations that facilitate data collection from mariners and host data upload. This data is then uploaded to an international archive, and mariners are provided visual artifacts created from their data that illustrate the depth of the seafloor.

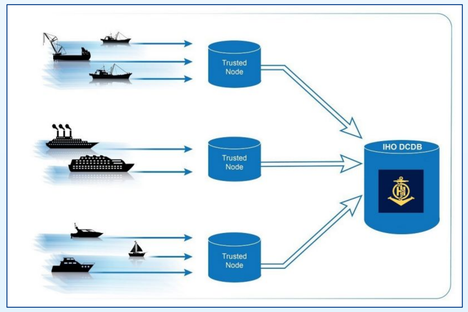


Figure 1: Data flow from Trusted Nodes to IHO Database

Requirements of the Web GUI

Functional Requirements:
- Upload logger data into cloud
- Handle processed data and return visual format
- Dashboard for viewing trends in data
- Protected by login, not accessible to everyone

Non-Functional Requirements:
- Intuitive flow and responsive navigation in GUI
- Querying data must be simple, statistics dashboard should be easily understandable
  - Emphasizing ease-of-use ensures anyone can operate UI.

Security Requirements (Minimal):
- Data is not sensitive, but potential for packet sniffing or man in the middle attack.
- Other considerations include unsanitized input.

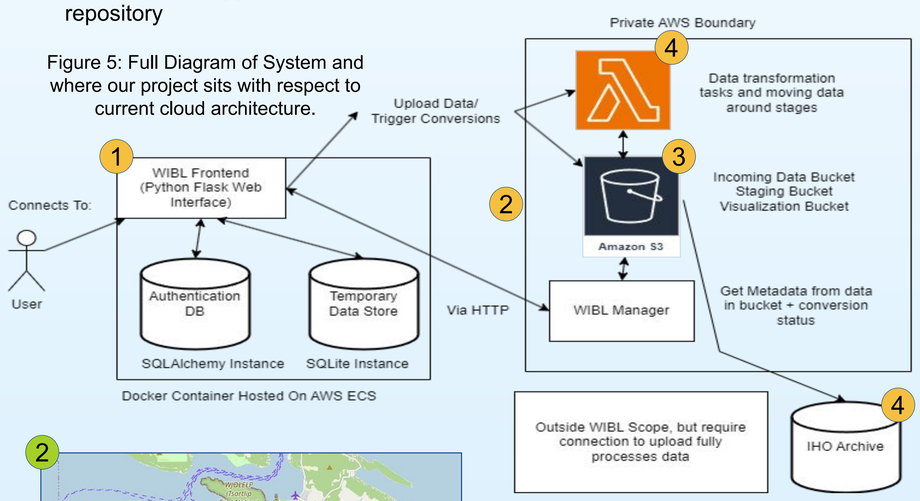
1

Figure 2: WIBL Web App Home Page

Architecture of System and Cloud

1. GUI sits in between user and cloud, ensures only interface talks directly with cloud
2. AWS houses backend data manager to communicate with the web GUI and represent current files in S3 buckets
3. Uploading data communicates with specific S3 buckets for different stages of data
4. Lambdas are triggered, converts and uploads files into an international repository

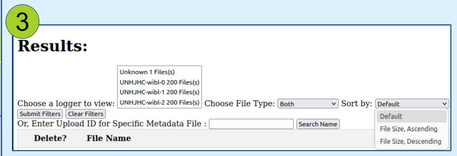
Figure 5: Full Diagram of System and where our project sits with respect to current cloud architecture.



2

Figure 3: Example of an Artifact Map given to the Mariners (Go to Artifact Page)

3



Filter Functionality via Logger, part of Figure 3

3

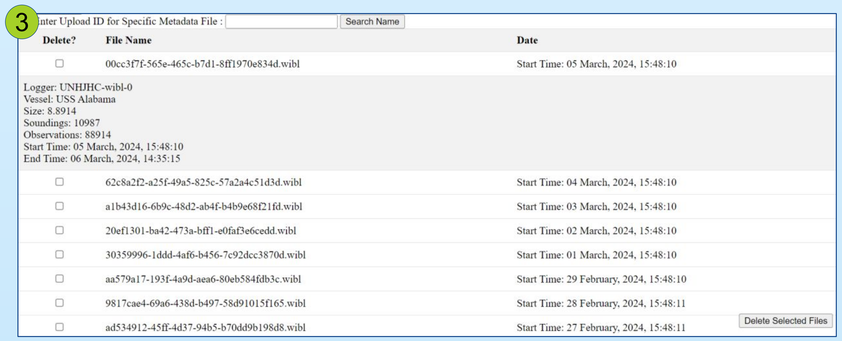


Figure 4: Result of query of manager (View Data and Results)

Testing

Testing Methods:
- Docker to initialize both Manager and Frontend one machine
- Heartbeat checks on themselves and between containers to ensure connectivity
- Python scripts to load manager with simulated metadata:
  - Used for testing output, filtering, and sorting
- Tests being implemented:
  - Incorporate libraries that mock Amazon Web Services to verify upload, modification, and retrieval from S3 buckets.
  - Simulate triggering lambdas to generate visual artifacts from data.

Discussion

As stated previously, a command line interface was used for interacting with the system and data upload. The user needed to initialize an S3 bucket, edit configurations, create a virtual environment and install multiple packages before attempting file upload. The whole process required manual input, which is prone to error.

The project success criteria is measured by accessibility, ease of use, and the granularity of data. Ease of use is quantified with time spent on the GUI and the amount of mouse clicks spent throughout the upload process. The result is a 10% decrease in time to initialize the upload environment. Compared to the previous method, we can reduce the time it takes for a new user to reach a state of upload to 3 clicks. File upload/query would only take an additional 4 clicks, due to the addition of parameter filtering.

Current Status of Project

Completed Items and Features:
- Secure and Persistent Authentication
- Presentable and Easy-to-use Web GUI, hosted on the cloud
- Autonomous execution of shell commands via Web GUI
- Connections to cloud endpoints, providing file upload, download, and querying

Future Work:
- Generation of Visual Artifacts
  - Project relies on mariners providing logger data, returning visual outputs will enable consumer retention.
- Requires interfacing with AWS Lambdas and S3 Buckets dedicated to visualization
- View status of file in cloud
- Statistics dashboard for tracking various metrics

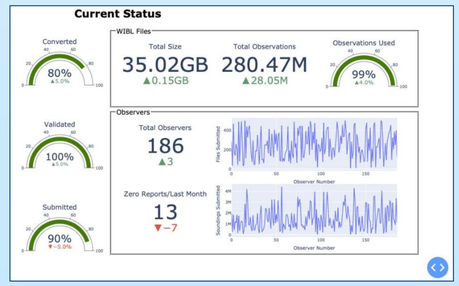


Figure 6: Concept of Completed Dashboard