Web Platform for Management of Data Science as a Service

Emi Harada, Alex Castro, Haobo Zhang
Faculty Adviser: Payman Arabshahi | paymana@uw.edu
Company sponsor: Sonos
Industry Mentors: Devon Lazarus, Natalie Remedios, Aaron Augsburger, Christian Woodruff, Chris Dolan, Jason Bliss, Ben Hodson

SONOS

Development of a data science web platform to enable data analytics and data categorization

When large amounts of data enter the company's data ecosystem, determining where all of the data is located gets complicated. With many tools and resources available for analysts, tedious time is spent searching for the right information and sorting it out. This takes time away from employees to analyze the useful data and report their insights. That's when our project comes into play. It will enable analysts to query the metadata information that is held on a centralized web platform for easy access.

OVERALL MAIN COMPONENTS

We used a few cloud components from the Amazon Web Services (AWS) in our project. Not only is AWS reliable and scalable to fit our problem, it can integrate into the company's network smoothly.

> Data Zoo (S3): stores raw data that contains information about Sonos products, finances, user surveys, etc.

> Lambda: allows scalable code to run without having to worry about server management. This function is triggered hourly from another AWS service called Cloudwatch.

> Catalog Database (RDS): provides a web service that can scale a relational database. We used MySQL database to hold our metadata information.

>Client Website (Elastic Beanstalk): deploys a web application so end-users can interact with finding the metadata information catalog.

DIAGRAM OF TECHNOLOGY STACK

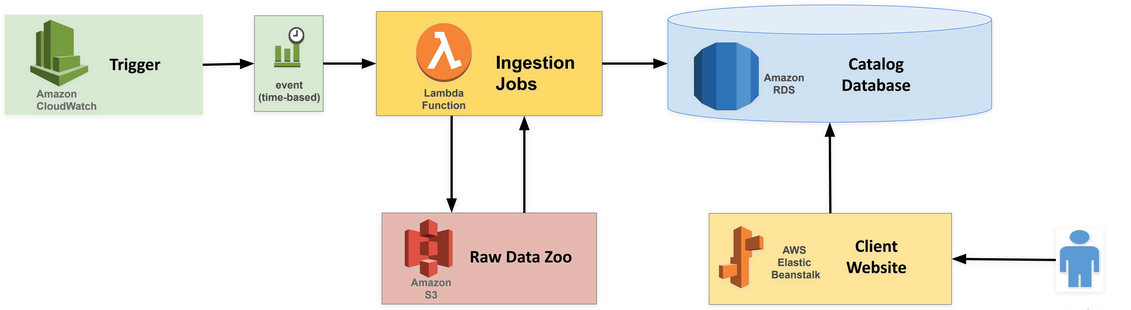

Figure 1: AWS components used in the project

FLOW OF THE DIAGRAM

Back End:

Whole project is controlled by the AWS CloudWatch service
> Creates events that trigger our AWS lambda function hourly
  > 1. First reads through the S3 buckets
  > 2. Extracts all the metadata from files
  > 3. Generates SQL query that inserts metadata into our relational database

> Implemented our relational database with a MySQL server hosted by RDS

> Performs different queries for further analysis with metadata relations

Front End:

> Interactive website with Django framework

> Allows the client to perform queries on our database display the data users are interested in

> Enables clients to help organize the data by adding custom tags to them

TABLE MODELS

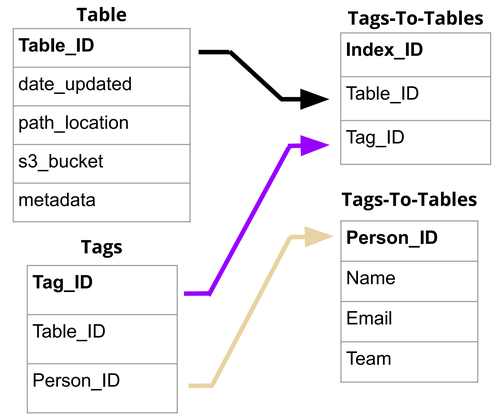

Figure 2: Connection of the tables inside the database

USE CASES

Our system was designed to address a variety of issues that Sonos currently has with their data. Here are the issues that we focused on and how our tool addresses them.

> **Cataloging Metadata:** Our system ingests the most recently updated files in the data lake and then stores the metadata in the system's RDS. In our design, we also create new metadata by allowing users to make their own categories.

> **Searching Tables:** All the information that we store on the tables is searchable via the client website.

> **Connecting Users:** When a tag is searched the interface also shows the contact information of the person who created the tag.

Development Console

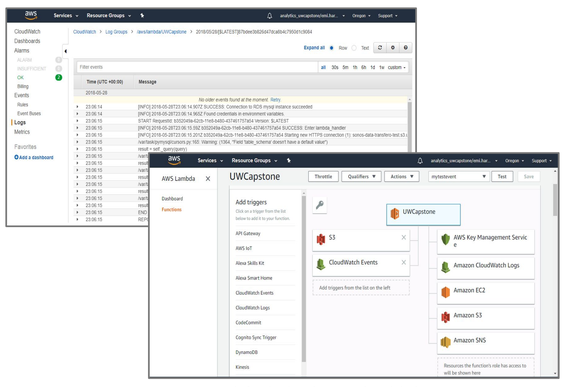

Figure 3: Management console for Cloudwatch [top] and Lambda [bottom]

TOOLS

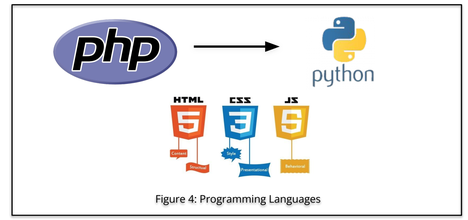

Figure 4: Programming Languages

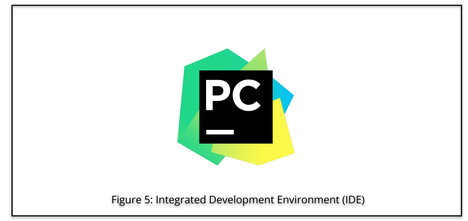

Figure 5: Integrated Development Environment (IDE)

FUTURE DEVELOPMENT

Examples of features that could be implemented to better support the needs of the business:

> Django App to display descriptions of how the keywords are used in each table, in expanded form after hovering over

> Django App to search by "fields" in the fields table after extracting the XML tags (i.e. column headers) in each file

> Django App to be more fluid by relating keywords that can associate with one another (i.e. "stereo pairing" also shows results for "bonding")

CONCLUSION

> Built a high fidelity metadata catalog for SONOS with AWS

> Metadata catalog consists of 3 main parts:
  > data injection by AWS lambda
  > data management by AWS RDS database
  > website as user interface implemented by Django

> With metadata catalog, easier and more precise data analysis can be performed on trillions of raw data points

> Further improvements can be done. For example:
  > more detailed data injection
  > smarter search functionality

References

[1] AWS Documentation [online]. Available: https://aws.amazon.com/documentation/. [Accessed May 26, 2018].