**Technical Report – Group Project**



**COSC310 – Software Project Management**

**Team Crocodile**

**Joel Foster  
Daniel Hemmings  
David Newman**

**15/05/2020**

**Introduction**

**Project overview**

The time and effort required to manually sort and classify images is significant and painstaking. The collection of field data and the processing of said data needs to be automated and streamlined in such a way that the database of information can dynamically update as new data is collected. These efficiency gains would allow end users to easily access up to date data as they require it. The client has provided their predictive model for the automatic classification of drop bears across various camera trap deployments. The aim of this project is to deploy this model (via a Cloud based platform) and to develop a means for users to interact with the output prediction data.

**Goals of the project**

* Develop an AWS Cloud solution to retrieve raw data from deployed camera traps to be processed by a trained TensorFlow machine learning model and stored in an SQL database.
* Develop the required website and mobile phone application to inform the general public about dropbear sightings and assist in dropbear research.

**Project scope**

To meet the user requirements for this project the following software components will be developed.

* Cloud-based API using Amazon Web Services (AWS) to retrieve raw data from email generated by pre-deployed camera traps, to then be processed by a pre-trained TensorFlow machine learning model.
* A secondary Cloud-based API using AWS to retrieve various classification data from the TensorFlow machine learning model, to be stored in an SQL Cloud-based database on the same AWS platform.
* A two-facing website to facilitate the database information for the general public and researchers. On the public level, it will allow the retrieval of dropbear sightings based on a specified postcode. On the researcher level, it will provide an invite-only web portal for researchers to apply for, which will grant access to various information on dropbear sightings.
* A cross-platform smart phone app for Android and Apple iOS with identical functionality as the general public website.
* Email alerts and smart phone notification functionality for postcode sightings.

**Assumptions**

* The project team has the required experience to meet user requirements and deliver the project on time.
* The client will convene with the development team every 3 months.
* The budget allocated provides enough for additional human and computer resources if required.

**Constraints**

* Additional funding is not available for the project.
* Existing software components cannot be modified.
* Team members time on the project is limited to 40 hours per week, Monday to Friday, for the duration of the project.

**Key Milestones**

The milestones are set to be delivered every three months. These meetings will be used to showcase our current advancements in the project and allow Dr Client the ability to take part in the direction of the project. This will be achieved through up-to-date training of Dr Client, with the ability for him to ask questions and discuss relevant changes. Please note: Phase details, as seen in Table 1, can be found in the preliminary execution schedule.

Table 1: Milestones and timeframes

|  |  |  |
| --- | --- | --- |
| **Project Milestone** | **Project Artifact** | **Timeframe** |
| Project start |  | TBD |
| Milestone 1 | Phase 1 delivery | + 3 months |
| Milestone 2 | Phase 2 delivery | + 3 months |
| Milestone 3 | Phase 3 delivery | + 3 months |
| Milestone 4 | Phase 4 delivery, Product completion | + 3 months |

**Identification of Stakeholders**

We understand the following stakeholders as part of the project:

* Dr Client
* University of New England
* State and Federal level project funders
* Drop Bear Protection Society of Australia
* Software Project Manager
* Development Team
* 5 test users

**Technical Solution**

**Overview of Solution**

**Major Components**

* AWS Simple Email Service (SES):   
  AWS Cloud based Email service is capable of intercepting emails coming in from the field camera traps.
* AWS Lambda Function:   
  A script written in Python that is called to run based on a customisable trigger event, such as when AWS SES receives an email from the field. Custom script would be able to pull out any necessary metadata from email, such as timestamps, and retrieve the photo attachments. Script would then store this information into our S3 Bucket.
* AWS S3 Bucket:  
  AWS Cloud storage, would act as an intermediary storage option for our data. Raw data from the Camera Trap emails could be stored, and then passed directly into AWS SageMaker for processing. This data could then be moved to our AWS Aurora database for permanent storage.
* AWS SageMaker:   
  AWS machine learning service would allow us to deploy the client Tensorflow prediction model on a GPU instance (for greater processing requirements). The model would be fed with raw data from our S3 Bucket, and output prediction data in the form of a classification, and a confidence interval along with an output image, highlighting a potential Dropbear sighting (if one was detected).
* AWS DocumentDB:   
  A document database, using MongoDB, would be the most suitable to organise and store the images and their data. This service provides automatic scaling of both storage and computation and faster speeds than some of their other offerings, providing less maintenance in the future as the system grows.

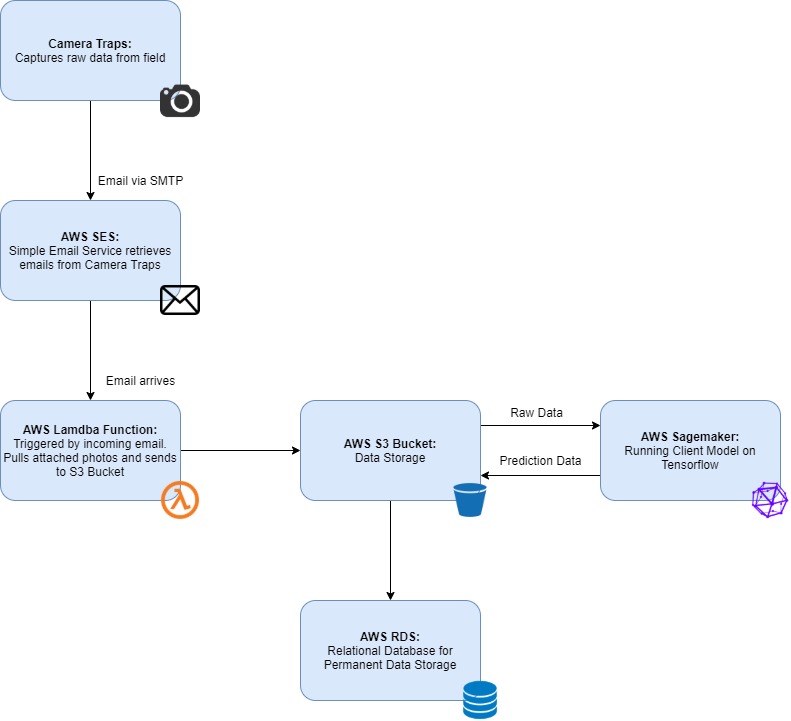


Figure 1 - Overview of Proposed Solution - Cloud Processing



Figure 2- Client Side Interaction

**Justification of Technical Solution**

Utilising only AWS components allows for simple and straightforward integration between the various components of the proposed solution. For example:

* AWS SES and Lambda Functions allow for easier photo retrieval.
* AWS SageMaker contains pre-defined containers for running a Tensorflow model.
* AWS has the ability to dynamically scale various services are required.
* Lambda calls and AWS SageMaker are only costing money when they are running, no need to create an instance and pay for it unless it’s actually working.
* AWS DocumentDB allows a fast and efficient document storage database and would be more effective than using a relational database.

Table 2: Estimated first year pricing

|  |  |  |
| --- | --- | --- |
| **Service** | **Details (Estimated)** | **AUD/Month ($/m)** |
| AWS Simple Email Service (SES) | 100 triggers per camera per day = 1200 emails/day @ 1.5 MB | $30.21 |
| AWS Lambda Function | 36,000 emails a month @ 10 second processing time | $17.30 |
| AWS S3 Bucket | ~2 GB storage per day = 1 TB a year | $41.71 |
| AWS SageMaker | 2 seconds processing per image, 2 hours needed per day @ $0.32/hr | $19.31 |
| AWS DocumentDB | With 1 TB storage | $1659.58 |
|  | **TOTAL (initial year)** | **$21,217.32** |

Note: Prices will grow dynamically with more or less use, dependant on added camera traps and access to servers by users/researchers.