Documentation for AWS

I. Introduction

This document contains the documentation for the various services and systems that are either *implemented* or *partially implemented* as a suggestion for future deployment. Some components are *partially implemented* because there is either insufficient time to finish them or because there isn't enough related infrastructure to fully visualize the capability.

II. Components

A. Setup

There are various processes and systems that require authorization of some form for security reasons and permissions reasons.

I. IAM

All permissions and security actions are mainly centered around IAM.

1. User Groups
    a. User Groups are clusters that can contain multiple users. Groups can also contain permissions that apply to all users. For instance, currently the group `CIS-II` contains users for terminal access.
    b. One can specify the degree of access a user can have For instance, if one desires that a given group only have programmatic access through AWS CLI.
2. Users
    a. Individually users can both have credential access to console or programmatically.
    b. **Access keys can only be accessed once for programmatic access.**
3. Permissions and Policies

a. Permissions and policies vary by service and action. See AWS documentation for specific use cases for your application.

B. Sagemaker

Sagemaker is a tool used to either to automate or make more efficient various machine learning processes.

I. Sagemaker Labeling - Fully Implemented

Documentation for creating a job can be found on AWS websites. Do note, ensure that that any job must have the data setup in the first place.

1. Terms:
    a. Clone Job - Used to copy a job's actions, but "old data" cannot be reused.
    b. Chained Job - Use previous job's actions and "old data" to continue on a given job.
    c. Input Manifest - Is a `.manifest` file that contains the setup of the job. It specifies where the input files are located.
    d. Output Manifest - Is a `.manifest` file that contains the output of the job. The output specifies where the files of each annotated frames are stored.
2. Extraction scripts:
    a. Under the github repository, under the `/AWS/ExtractionScripts` folder contains scripts to extract files and images from AWS.
3. Automated vs Manual Setup
    a. Automated Setup creates jobs for you and setup the file structure for you. However, this is only mostly useful when trying to create, chain, or clone a job.
    b. Manual Setup creates jobs based on the `.manifest` file that you feed it. This is useful if you would like to create a new job with a diminished data set that has already been created.

II. Training

Training can be done via `.jyp` notebooks in the `notebooks` section of the Sagemaker dashboard. If one plans to user GPU however, ensure that the proper quota amount

ensures an EC2 instance for training.

III. Sagemaker Inference - Not Started

Eventually it could be potentially more beneficial to deploy ML models to sagemaker.

C. AWS Cognito - Partially Implemented

Cognito is a service to create new customer users for an application. There is no way to visualize this services without using an additional web application that is connected to cognito.

I. Identity Pools

Identity pools are created for a given application or user case. Documentation for identity pools can be found on AWS.

i. Login - A login system using SSO from other services eg. Google, Facebook, etc. can be created or an email + SMS system as well.

ii. Recovery - A recovery system for users to retrieve passwords or accounts.

iii. Fields - Usernames, names, and other information about the patient can be associated with a user.

Currently, there is a CIS Test Identity Pool

II. User Groups

User groups are partitioned groups that contain users. Users can be part of multiple groups. Currently, there is an user pools of sample doctors office within the identity pool.

III. Code

For an automated method of creating new users and their related components, a sample system can be found under `/AWS/PipelineFiles/TestPipeline` and use the `SampleRun.py` file to test. Files can be found via github.

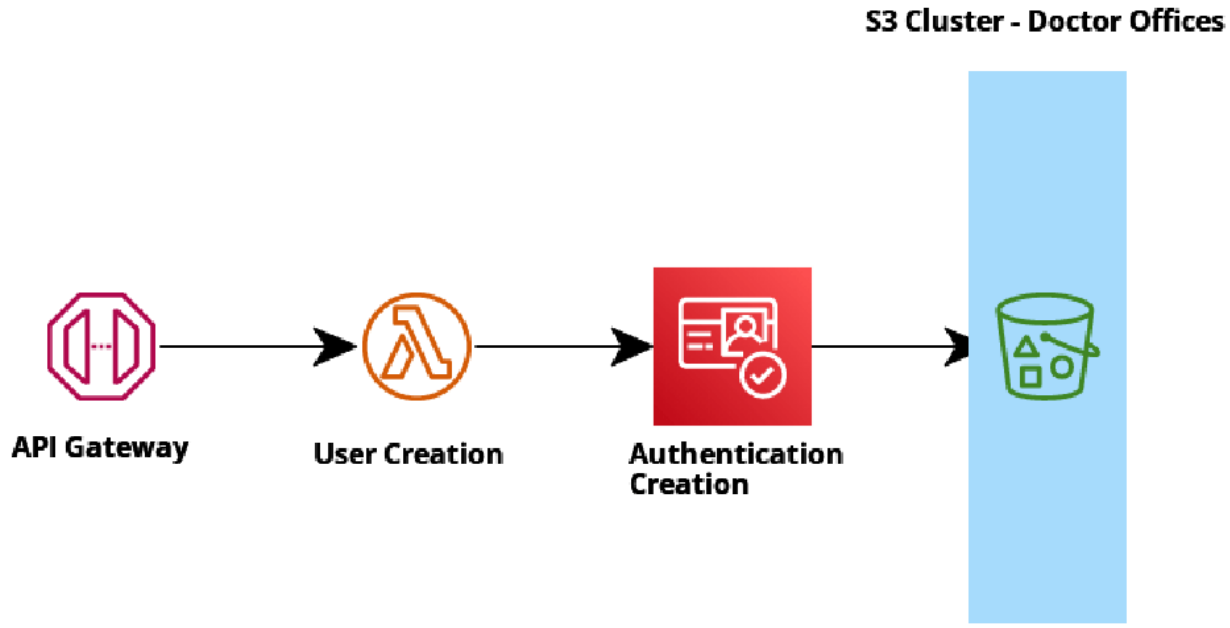
Note, the current code is in python. It is suggested that the the app developer use AWS javascript SDK for user and app creation.

IV. Future Suggestions

An app developer should interface with Cognito to allow easier access of AWS components.

A noted limitation of Cogntio is that there is no elegant method of hierarchy of users, similar to inheritance in classes and objects.

V. Diagram

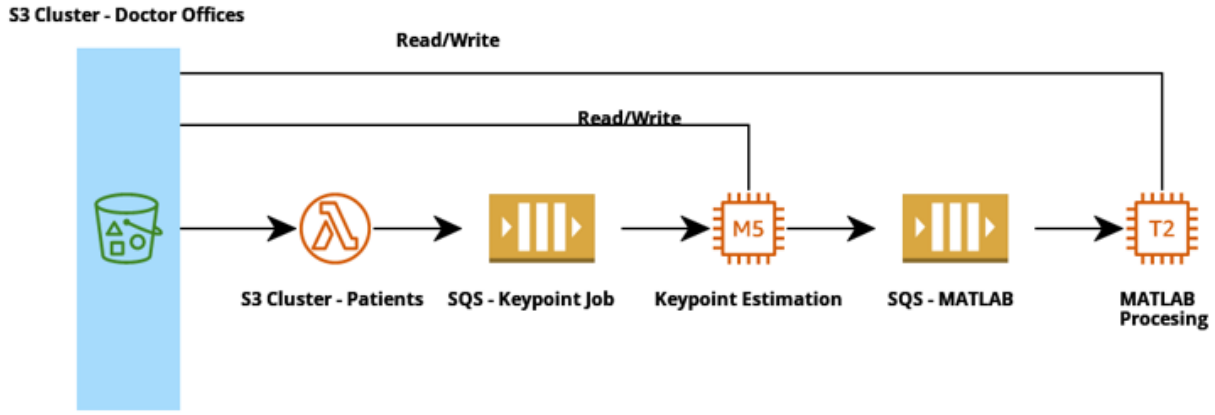


The above diagram describes a pipeline for new user creation alongside the resources needed for that new user.

Lambda functions can be implemented from `/AWS/PipelineFiles/TestPipeline` . The API Gateway has not be been implemented. (Everything right of User Creation in the diagram above has been implemented)

D. Pipeline

The pipeline is the system for start to finish for processing files.



S3 Buckets

Per requirements, buckets are automatically provisioned and are attached to a lambda function through bucket notifications. code to provision new buckets via cognito can be found in the github repository under the `AWS\Pipeline_Files\Cognito` folder. This applies to both input and output.

Note, that these buckets are attached to lambda functions via bucket notifications, and that the proper policy for attaching these buckets to the lambda functions must be in place. See IAM roles for lambda.

Lambda Functions

In the above diagram, S3 Cluster - Patients, is the lambda function that checks if an `.mp4` and `.hpf` file exist. If both exist, then SQS is notified and contains a new job ready to be processed. The file for the lambda function can be found here: `AWS\Pipeline_Files\lambdaFunctions`

Queue wise, the files are loaded onto `cis-ii-workload-queue`

EC2 Instances

Once jobs are put on the `cis-ii-workload-queue` , an EC2 then grabs the jobs through repeatedly checking if the file is there. The file that performs this operation can be found here: `AWS\Pipeline_Files\launchFile.py` . The instances are configured to delete the job once everything is completed. Do note, SQS has a limited read time. Meaning that once an instance acquires that job, other instances cannot read it. However, if the instance that has acquired that job fails to process the information in time, another instance can

take over and access that job. This prevents duplications while also ensuring all jobs are processed even if a failure occurs.

Files are read from the S3 bucket that is signified from the SQS. Files are written back to the same S3 bucket as the input, as signified from the SQS.

AMI's

Note that for each instances, an AMI - image, can be extracted from that image. AMI's are useful because one can boot up new instances (copies) of an existing instance. There are currently two AMIs.

- CIS-II-Demonstrator-B: This template is the base template to simulate updating and running functions.
- CIS-II-Inference: This template is the inference template. Note: not working as of 5/8. Must re-clone the repository

AMI's can be configured to dynamically scale a task. Note the version of the AMI when using it.

MATLAB EC2

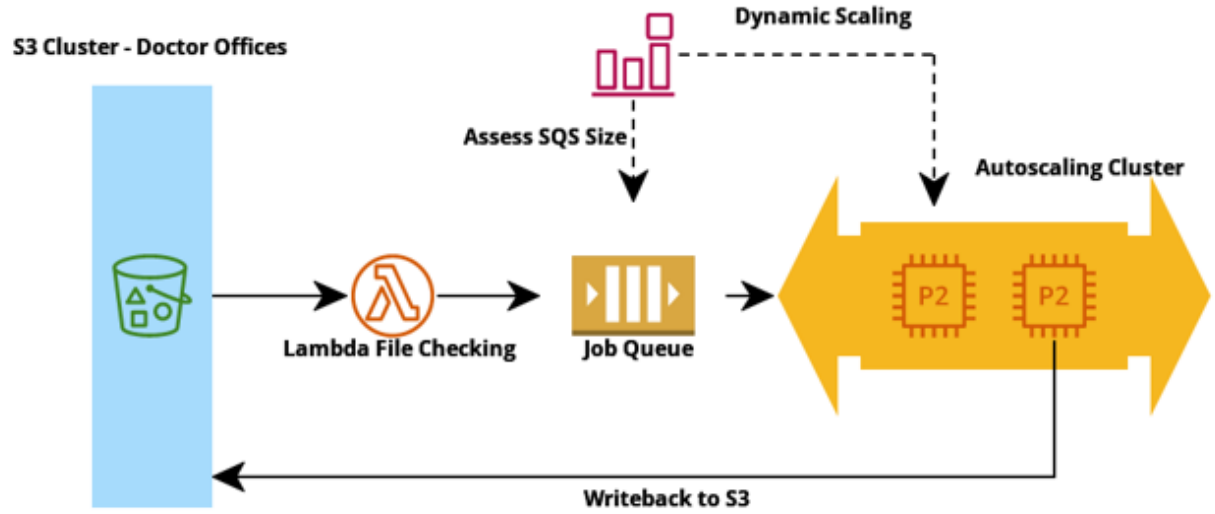
In the example pipeline above, the `workload-queue` has its jobs read by the MATLAB instance. This instance acquires the load, runs MATLAB on a JHU license (expires when user graduates), and has a desktop setting. The only method of accessing this instance is through remote desktop or through ssh. This machine then writes the file to `preprocessing-queue` . Reading and writing files is the same as what is mentioned above.

INFERENCE EC2

In the example pipeline above, the `preprocessing-queue` has its jobs read by the inference instance. This instance acquires the load and runs the inference model. Reading and writing files is the same as what is mentioned above.

Ideal Pipeline

The above pipeline is not scaleable. Below is the ideal deployment, with a majority of components implemented because of the transfer function and keypoint prediction not finished as of May 5th.



Cloudwatch and SQS

The principle of the pipeline remains the same up until Job Queue. A cloudwatch alarm is activated when it detects on the condition that there is 10 or more jobs per consecutive minutes. When this alarm is activated, it requests that the auto-scaling component to expand. Cloudwatch gets this information from the SQS queue. A similar function occurs when descaling the autoscaling group.

Custom metrics for the cloudwatch alarms must be implemented. They are not implemented as of May 5th because of the incomplete individual components.

Autoscaling Cluster

The current autoscaling cluster `CIS-II-Demonstrator` utilizes the cloudwatch system listed above. The AMI images are based on the CIS-II-Demonstrator and the instances are currently t2.micro instances in order to save cost for a demonstrator.

Metrics of the cluster, along the SQS, can be found in the cloudwatch dashboard.

Old Versions of the pipeline:

AWS Pipeline for Modeling

AWS Pipeline Full Design Documentation

MISC. Docs

Archived Old Documentation

AWS Web-Based Processes

Github AWS Docs