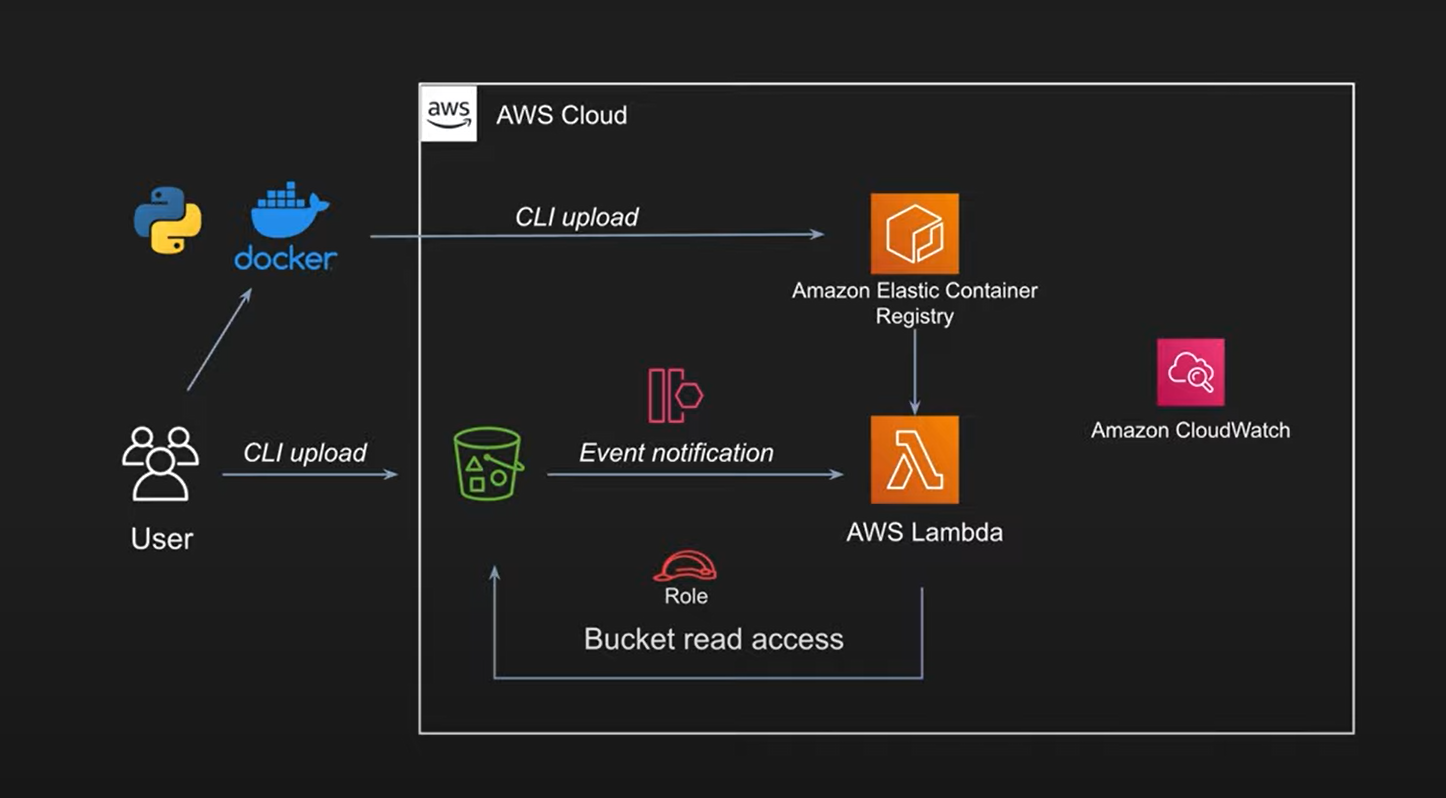
Assignment 2  
Name - Bhumika Punjabi, NUID- 001061256

# Data Ingestion Pipeline

**Data Ingestion Pipeline using Docker Containers on Lambda using ECR, EventBridge and S3**



## 

## **Data Pipeline key Steps-**

## Build Lambda function as a Docker Container Image

* sevir\_ingestion.py - Contains code for reading data from S3 doing some transformations and generating (x\_test and y\_test)Shuffle and split the data into training data and test data
* Build the Dockerfile and Tag the image

docker build -t data\_ingestion .

docker tag data\_ingestion:latest 711787209496.dkr.ecr.us-west-2.amazonaws.com/data\_ingestion:latest

* Create ECR repository and Push Lambda container image to ECR

aws ecr get-login-password --region us-west-2 | docker login --username AWS --password-stdin 711787209496.dkr.ecr.us-west-2.amazonaws.com

docker push 711787209496.dkr.ecr.us-west-2.amazonaws.com/data\_ingestion:latest

* Deploy Lambda container image with AWS Lambda console
* Sync the data with sevir open registry

aws s3 sync s3://sevir/ s3://seviringestion

# Detail Flow of the pipeline-

AWS Services used-

* S3 (To store sevir data)
* ECR(To store container image)
* IAM (For Permissions)
* AWS Lambda (To deploy image)

## Creating docker images-

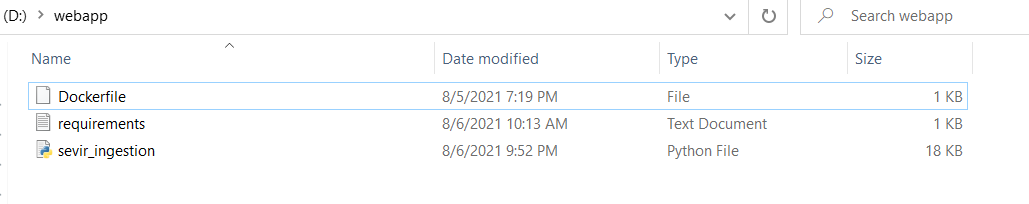


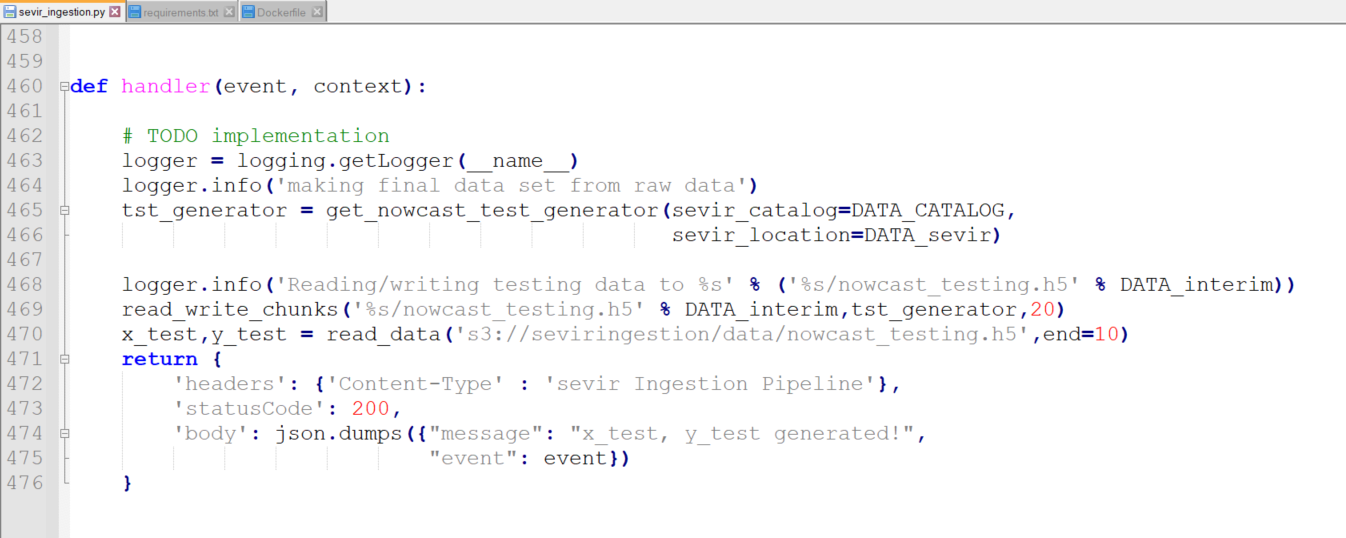
Container image includes the base operating system, the runtime, any Lambda extensions, your application code, and its dependencies

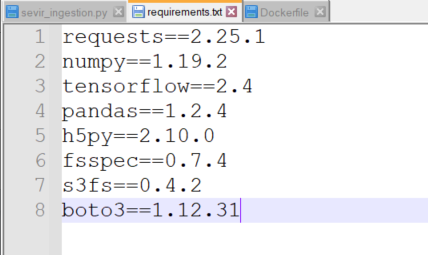
A container image is a packaged template built from a Dockerfile.

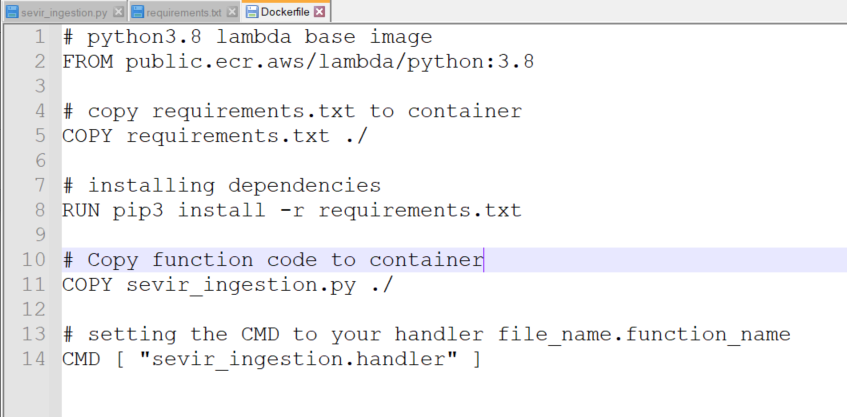
The image is assembled or built from commands in the Dockerfile, starting from a parent or base image, or from scratch. Each command then creates a new layer in the image, which is stacked in order on top of the previous layer.

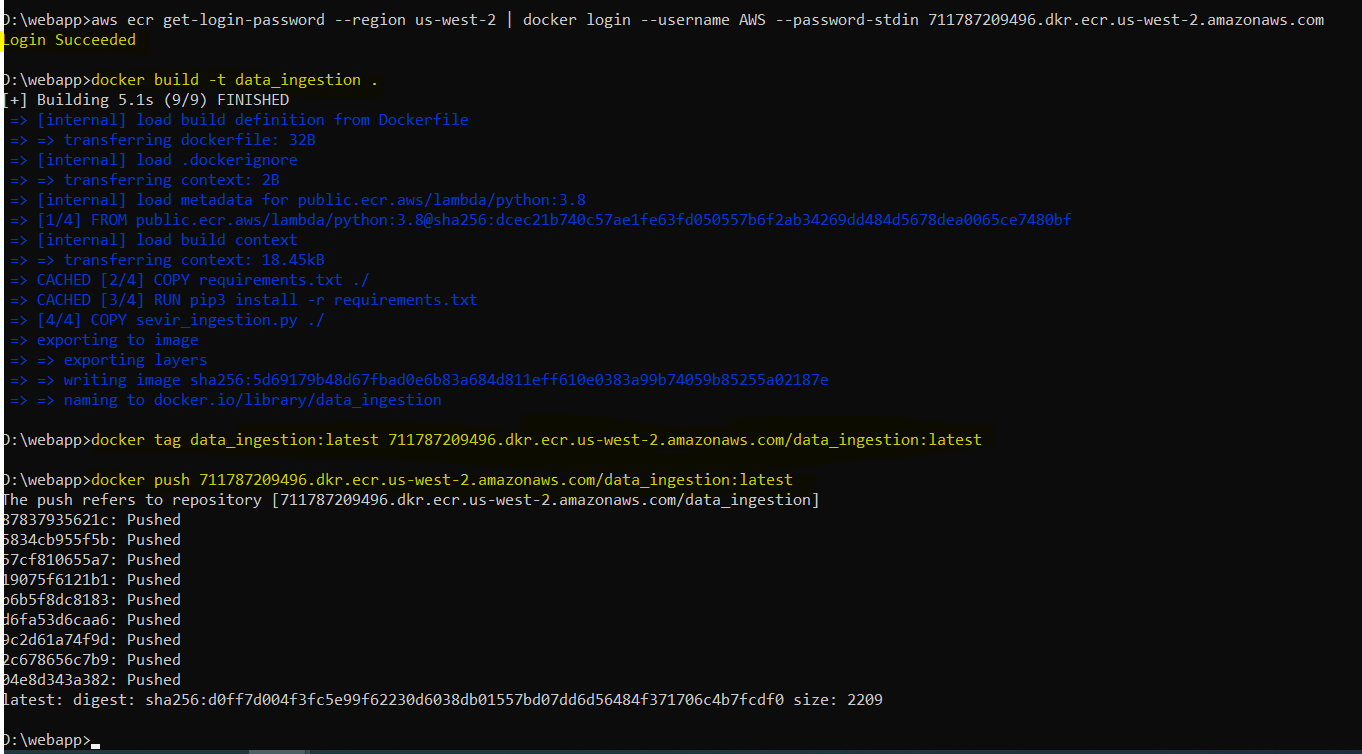
Once built from the packaged template, a container image is immutable and read-only.







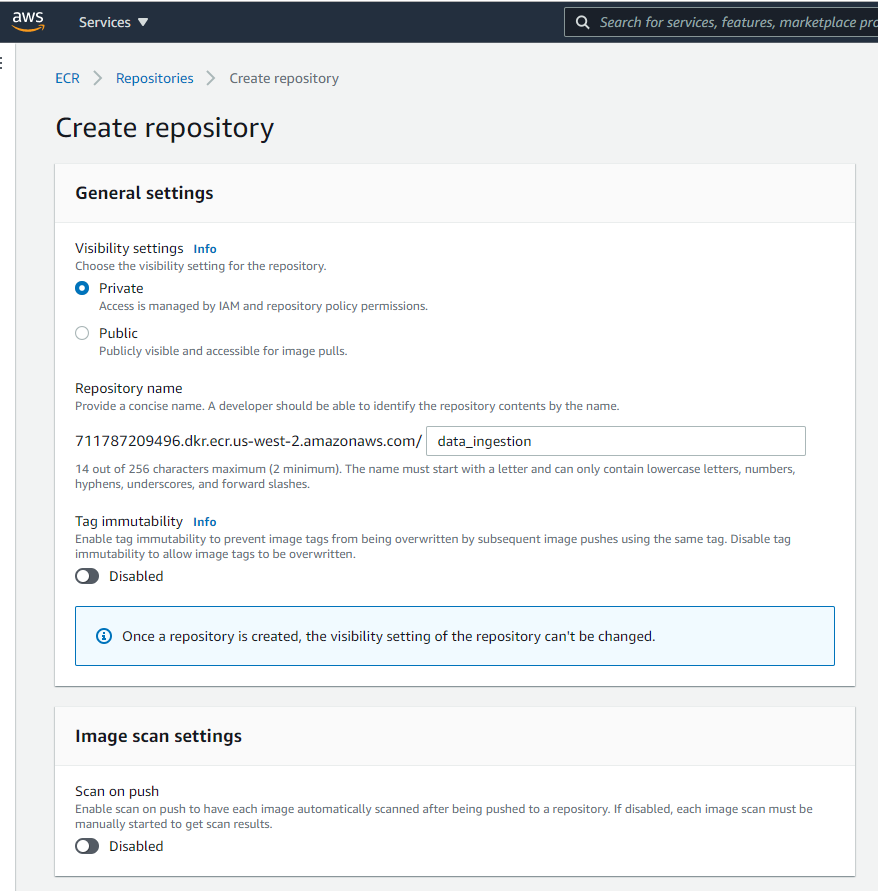


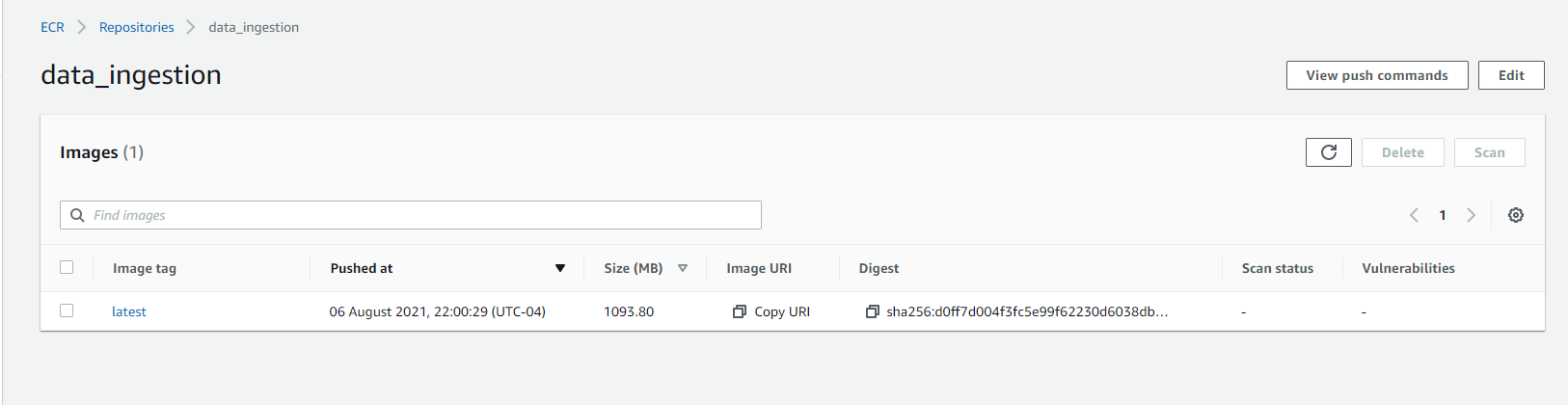


## Elastic Container Registry

Push images to an Amazon Elastic Container Registry (ECR) repository, a managed AWS container image registry service.

We create Lambda function, specifying the source code as the ECR image URL from the registry.



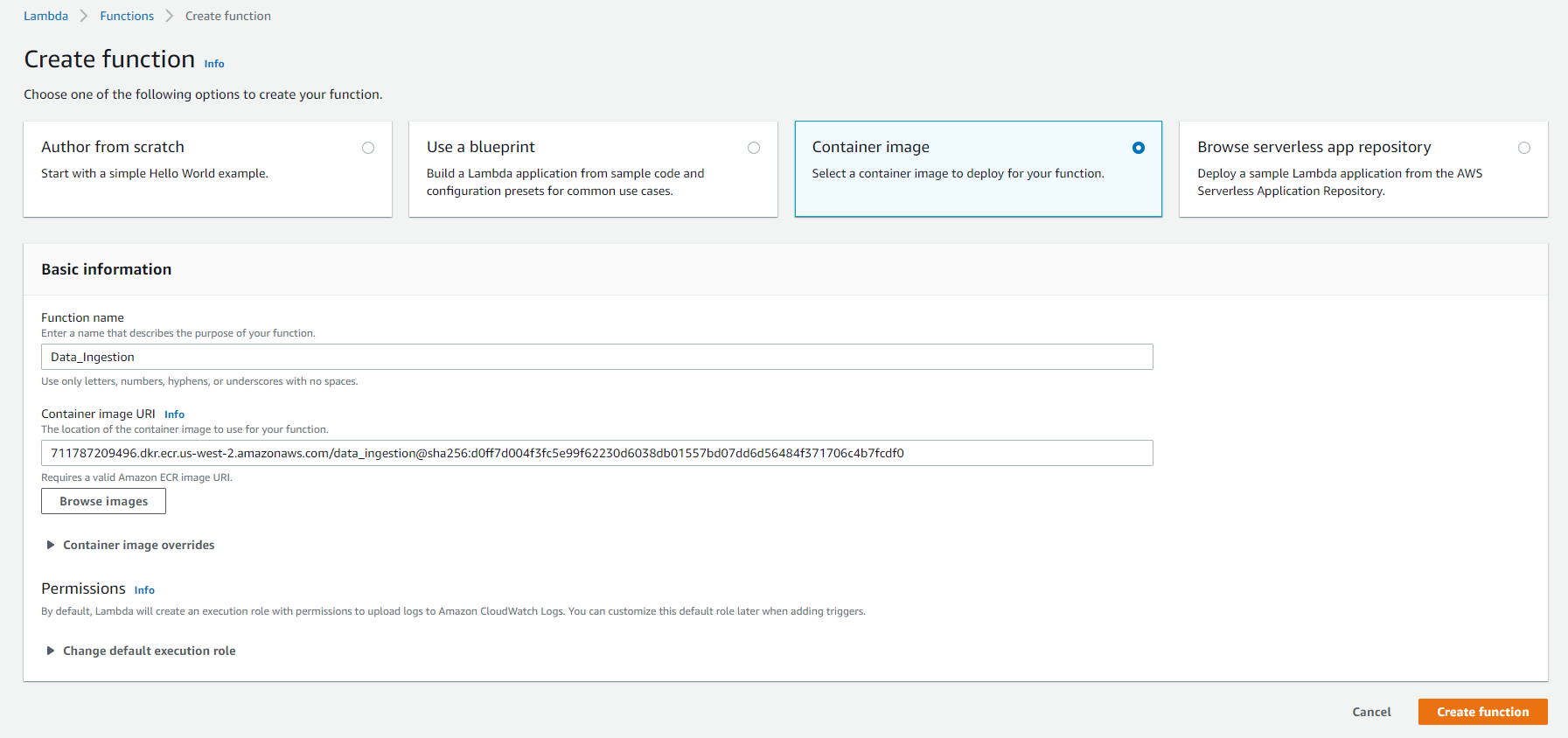


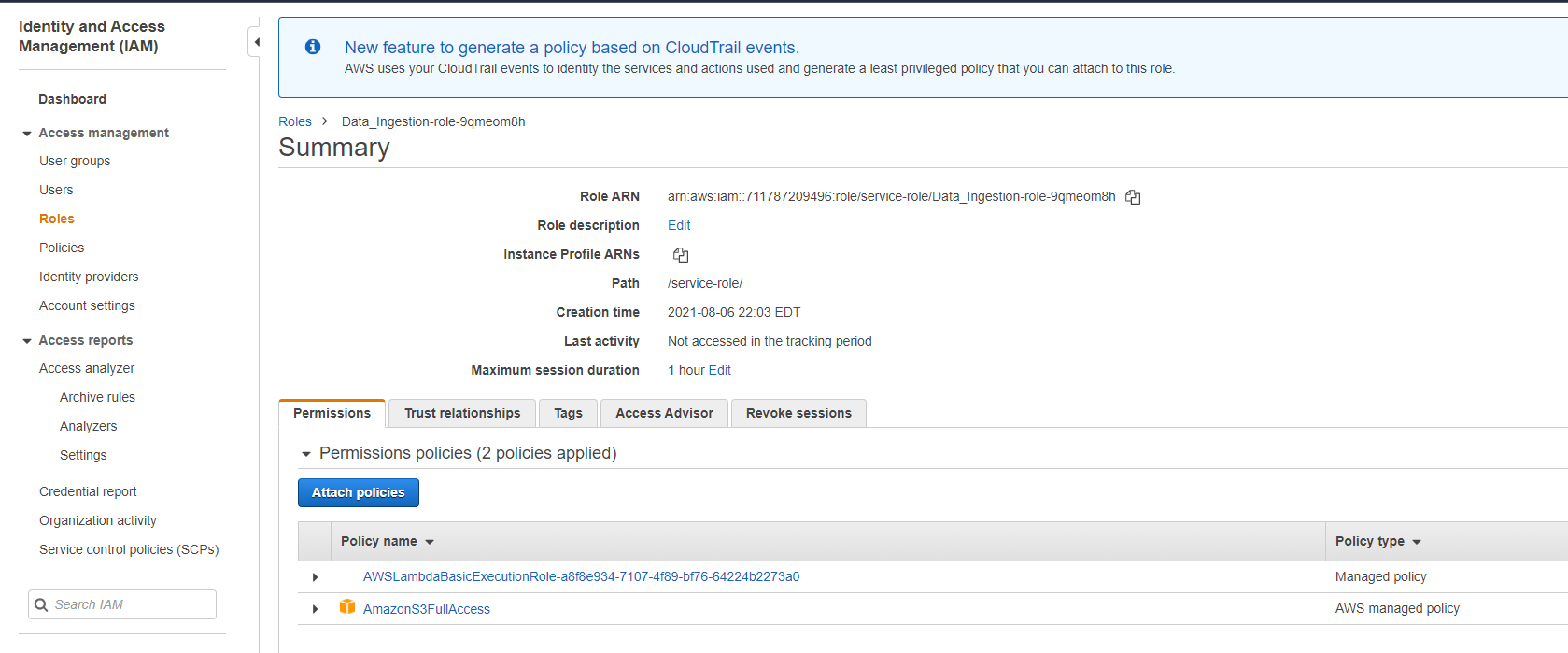
## Lambda Function:

Lambda provides a set of open-source base images that you can use to build your container image.

Lambda uses the image to construct the execution environment during function initialization.

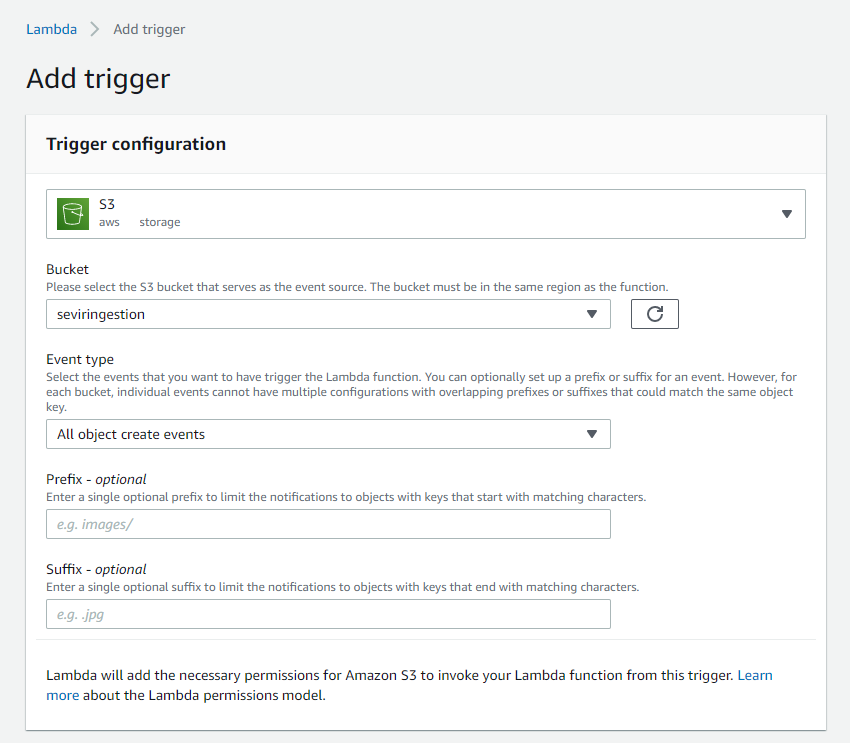
Lambda also optimizes the image and caches it close to where the functions runs



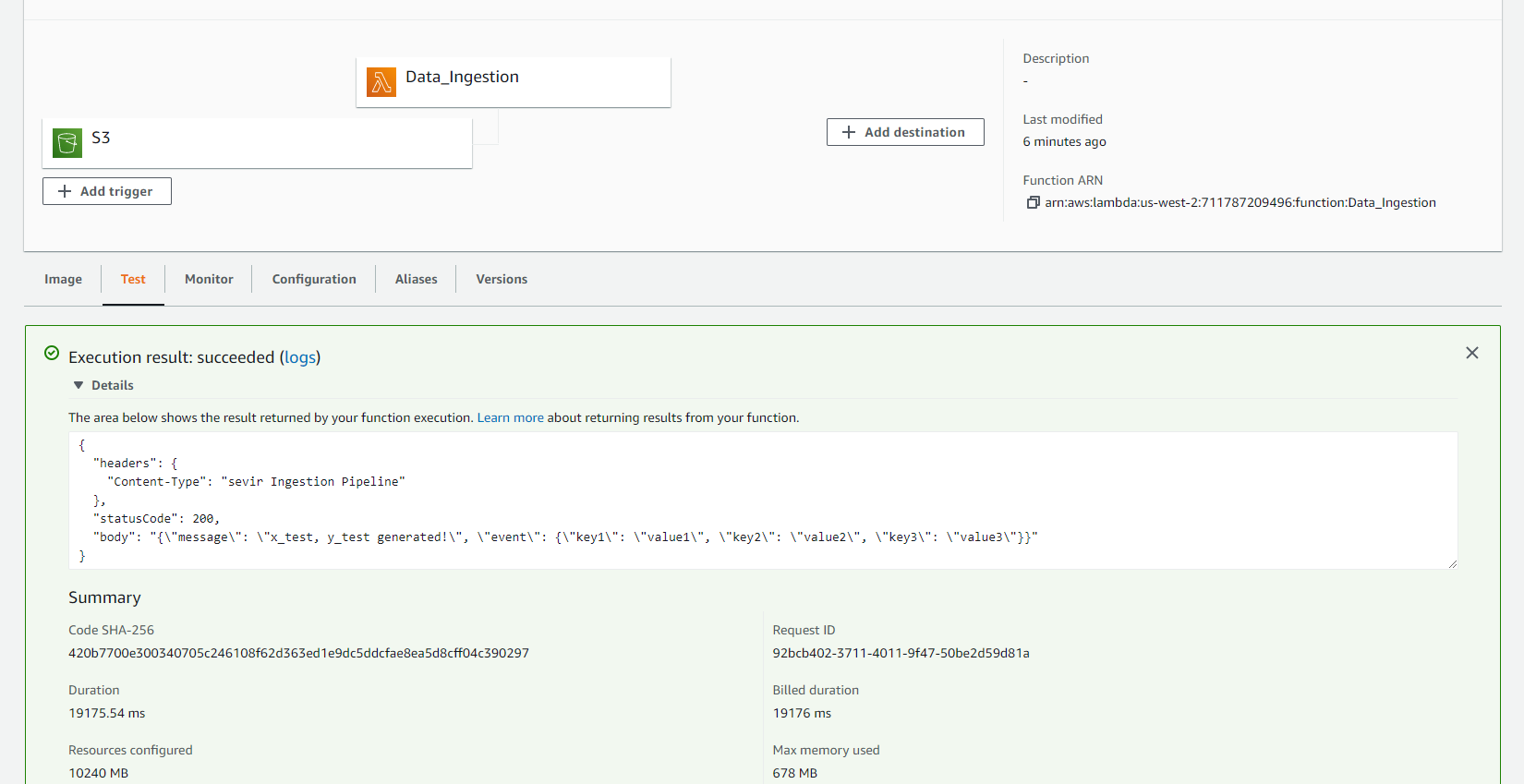


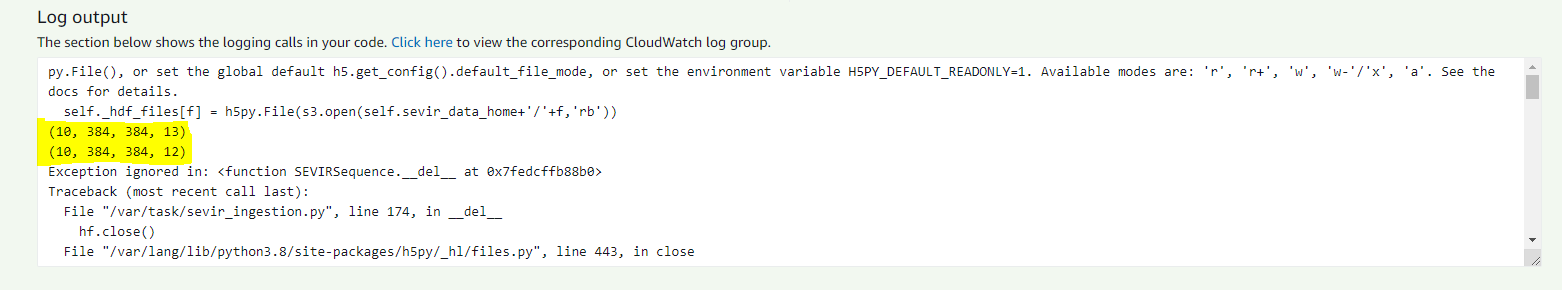
## Trigger-

To Invoke Lambda Function, as and when data is added to the S3 bucket.



## OUTPUT-

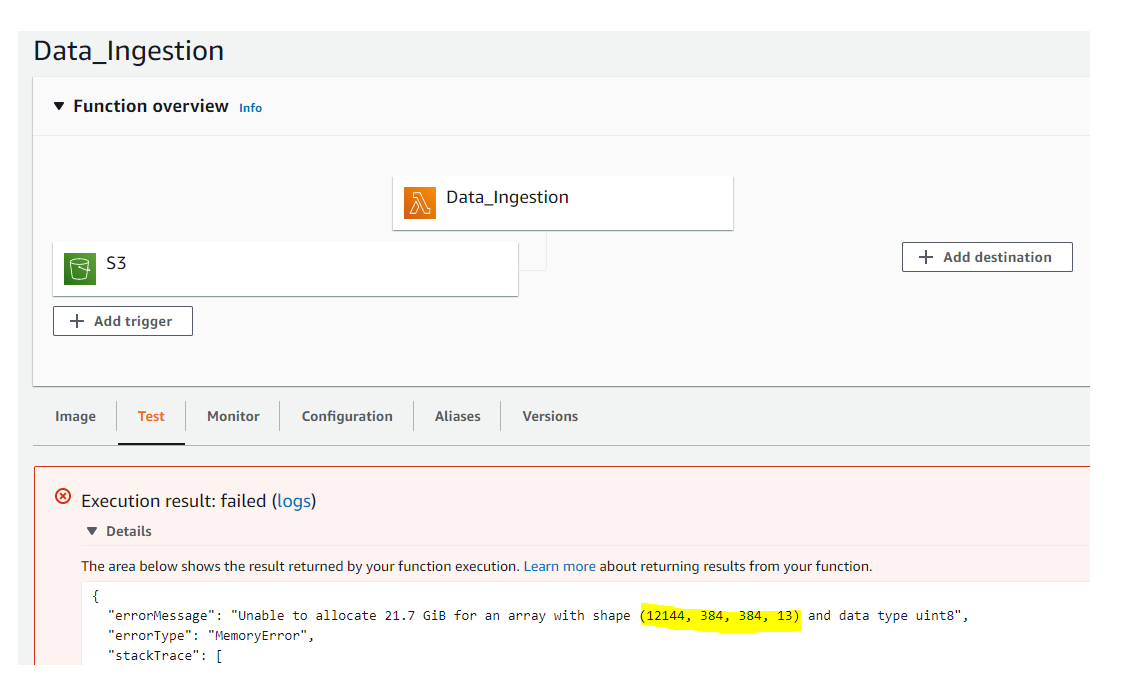


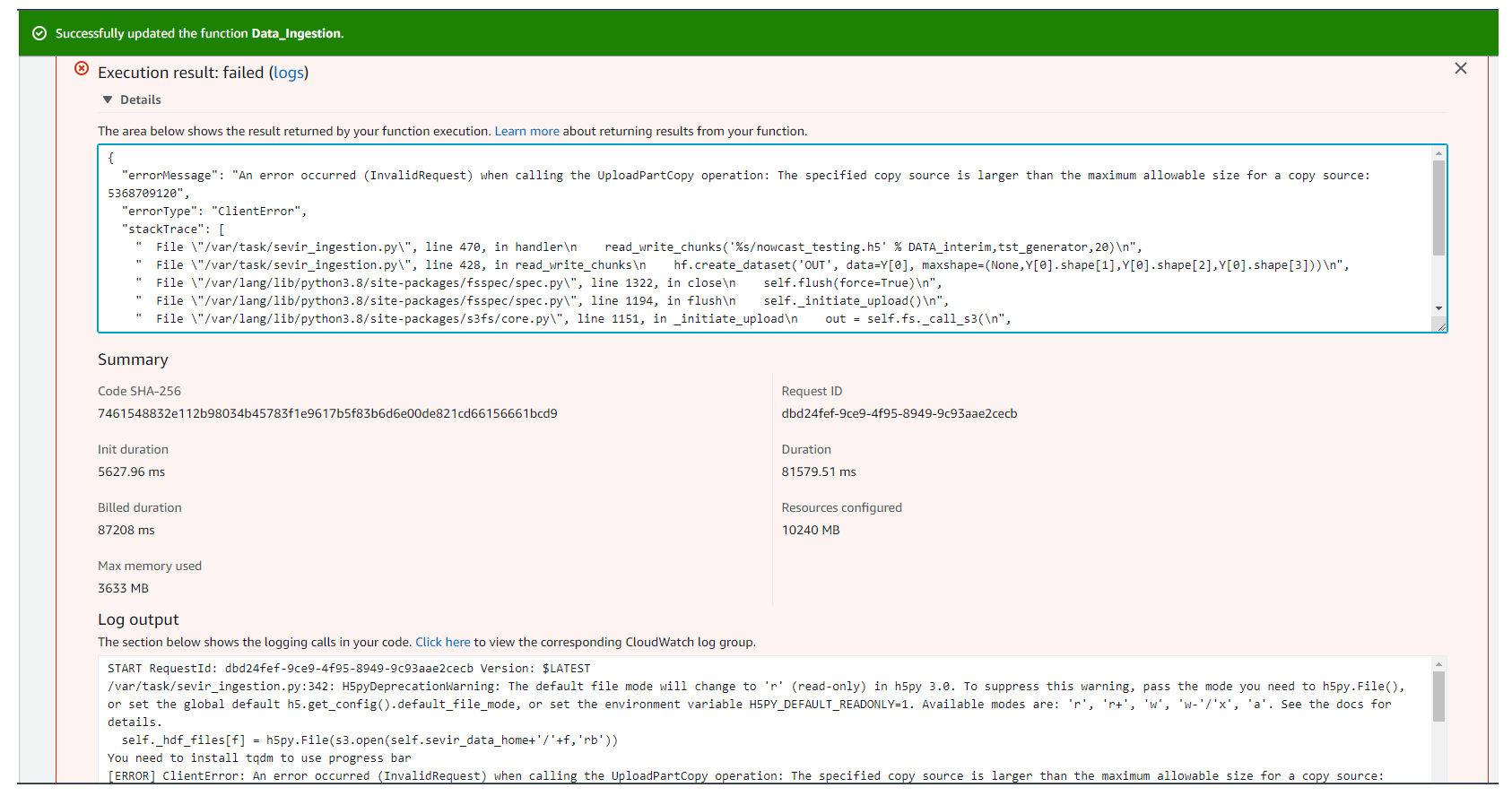


# Challenges

**Tool Based:**

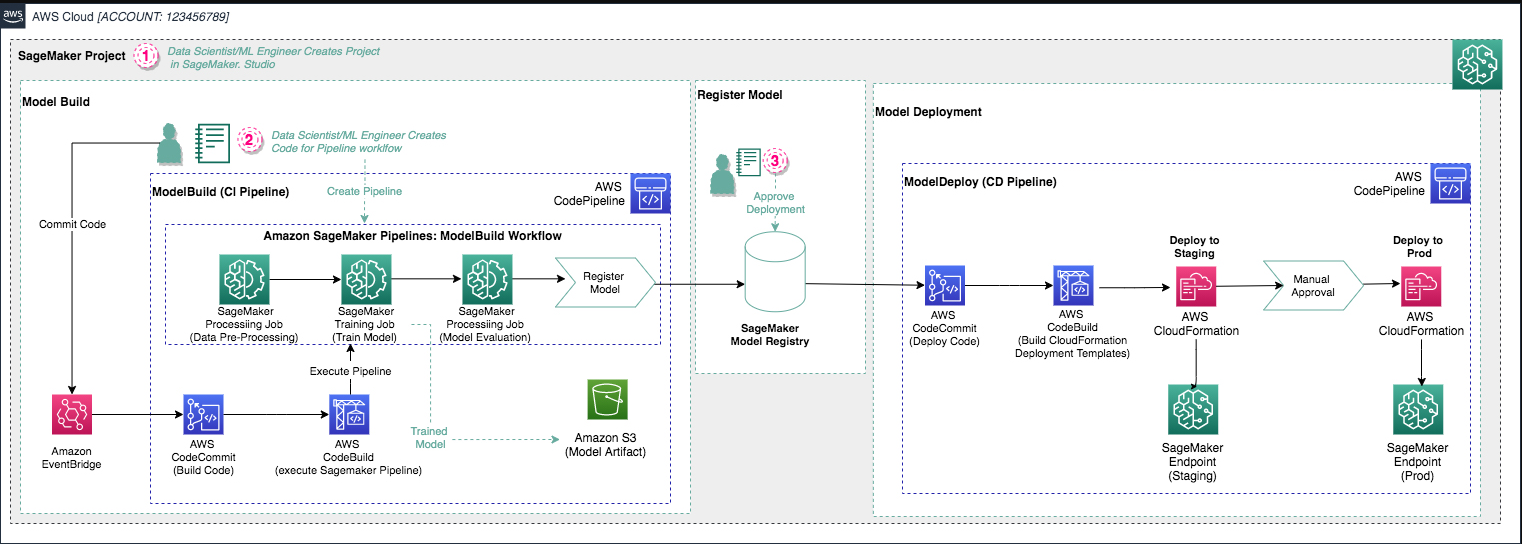
* Setting up environment on the lambda function was not allowing the zip folder to be greater than 250 mb
* Switched to Docker, further faced lot of issue with setting up the docker desktop and making it up and running
* Memory issues when deploying the container to lambda
* Also though the code was running locally, when pushing it to the cloud had to adjust the version of all the libraries to make the execution of the code successful.





# Model Building Pipeline

## Amazon SageMaker Pipeline:



* Set up the Amazon SageMaker session, create an instance of the model (an estimator), and define the model’s hyperparameters
* Start the training job and trains the model using gradient optimization on large instances
* Further deploy the model on a server and create a SageMaker endpoint that can be accessed.

# Deploy/inference pipeline

## Triggering a pipeline run:

* Committing changes to the CodeCommit repository triggers a new pipeline run, as an [Amazon EventBridge](https://aws.amazon.com/eventbridge/) event monitors for commits.
* Further, we can monitor the run by choosing the pipeline inside the SageMaker project.
* The pipeline run displays the steps of the pipeline, which can be monitored.
* When the pipeline is complete, inside the SageMaker project we can inspect the metadata attached to the model artifacts and manually approve the model
* This approval triggers the ModelDeploy pipeline and exposes an endpoint for real-time inference