**Asma Naeem**

**CSCI-E-90 Fall 2023**

**Final Project**

**Project Summary: https://youtu.be/2\_ZqRM0MVcc**

**Project Presentation: https://youtu.be/bNZpUJhQOBY**

1. **Introduction**

This project aims to use Amazon Kinesis Video Streams to ingest videos from camera devices and perform live face recognition and near real-time analysis using Amazon Rekognition Video. Amazon Kinesis Video Streams provides SDKs to securely stream video from connected devices to AWS for playback, storage, analytics, machine learning, and other processing.

We are trying to build a proof-of-concept pipeline that receives video streams from a Raspberry Pi camera device and leverages Amazon Rekognition Video to detect faces in videos.

This project demonstrates how to use Amazon Kinesis Video Streams APIs and SDKs to stream media to AWS Cloud securely. We will identify objects in video streams using Amazon Rekognition deep learning service.

1. **Architecture Diagram:**

A diagram of a video game

Description automatically generated with medium confidence

1. **Set Up**
   1. **Create Cloud formation Stack**

We will create the following resources:

* Kinesis Video Streams stream
* Kinesis Data Streams stream
* Rekognition Collection
* Rekognition StreamProcessor
* Lambda function
* OpenSearch domain
* IAM roles
* Cloud9 instance

3.1.1 Create a Lambda function (dstrm-srch-fn) to send face recognition results to the AWS OpenSearch service.

|  |
| --- |
| Lambda Function Python Code (dstrm-srch-fn.py): - |
|  |

- This function will process the Rekognition results from Kinesis Data Streams and send them to OpenSearch domain.

3.1.2 Create a .zip deployment package with dependencies for the Lambda function:

- Navigate to the project directory containing our lambda function.

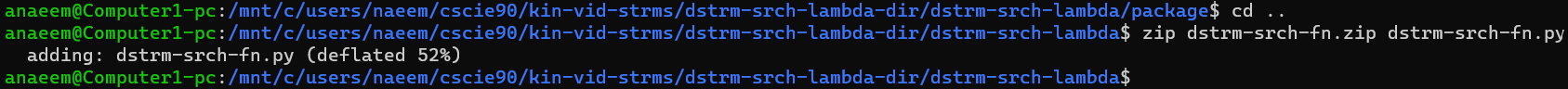
- Create a new directory named ‘package’ into which we will install our dependencies. We would keep our dependencies separate from our source code for easier navigating around our IDE.

- Install the Boto3 SDK from the Python Package Index using pip.

- Create a .zip file with the installed libraries at the root. This will create a .zip file in our project directory.

- Add the lambda function .py file (dstrm-srch-fn.py) to the root of the .zip file

- The .zip file should have a flat directory structure, with our function's handler code and all our dependency folders installed at the root. If the .py file containing our function’s handler code is not at the root of the .zip file, Lambda cannot run our code. A screenshot of a computer program

Description automatically generated   A screenshot of a computer

Description automatically generated

2.1.4 Create an S3 bucket (kin-vid-cloud-formation-bucket) and upload Lambda function code (dstrm-srch-fn.zip).

2.1.5 Create stack:

To Detect faces in streaming videos, Amazon Rekognition Video would require:

* + - 1. Kinesis Video Streams for sending Videos
      2. Amazon Rekognition Video stream processor to manage the streaming video analysis.
      3. Kinesis data stream consumer to read the analysis results that Amazon Rekognition Video sends to the Kinesis data stream.
* Use Face Search settings to search for faces from a collection
* Grant Kinesis permissions to StreamProcessor Role

Create a stack (using template file: kin-vid-cloud-formation.yml). Provide stack details as follows:

Stack name: kin-vid-cloud-formation

Parameters:

Allowed IP address: Your IP address.

S3Storage: Bucket name you created in the previous step.

LambdaSource: your Lambda function name

|  |
| --- |
| Cloud Formation Template: - |
| AWSTemplateFormatVersion: 2010-09-09  Parameters:    S3Storage:      Type: String    LambdaSource:      Type: String      Default: dstrm-srch-fn.zip    AllowedIP:      Type: String  Resources:    LambdaRole:      Type: "AWS::IAM::Role"      Properties:        AssumeRolePolicyDocument:          Version: "2012-10-17"          Statement:            - Effect: Allow              Principal:                Service:                  - lambda.amazonaws.com              Action:                - "sts:AssumeRole"        ManagedPolicyArns:          - arn:aws:iam::aws:policy/AmazonKinesisFullAccess        Policies:          - PolicyName: kin-vid-AWSLambdaBasicExecutionRole            PolicyDocument:              Version: "2012-10-17"              Statement:                - Effect: Allow                  Action: logs:CreateLogGroup                  Resource: !Sub "arn:aws:logs:${AWS::Region}:${AWS::AccountId}:\*"                - Effect: Allow                  Action:                    - logs:CreateLogStream                    - logs:PutLogEvents                  Resource: !Sub "arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-group:/aws/lambda/kin-dstrm-srch-lambda:\*"          - PolicyName: AWSLambdaElasticsearchExecutionRole-kin-vid            PolicyDocument:              Version: "2012-10-17"              Statement:                - Effect: Allow                  Action: es:ESHttpPost                  Resource: "\*"    OpenSearchDomain:      Type: AWS::OpenSearchService::Domain      Properties:        DomainName: kin-vid-domain        EngineVersion: "OpenSearch\_2.5"        AdvancedSecurityOptions:          Enabled: false        ClusterConfig:          InstanceCount: 1          InstanceType: "t3.small.search"        CognitoOptions:          Enabled: false        DomainEndpointOptions:          CustomEndpointEnabled: false          EnforceHTTPS: true          TLSSecurityPolicy: Policy-Min-TLS-1-2-2019-07        EBSOptions:          EBSEnabled: true          VolumeSize: 10          VolumeType: "gp2"        EncryptionAtRestOptions:          Enabled: false        NodeToNodeEncryptionOptions:          Enabled: false        AccessPolicies:          Version: "2012-10-17"          Statement:            - Effect: "Allow"              Principal:                AWS: "\*"              Action: "es:\*"              Resource: !Sub "arn:aws:es:${AWS::Region}:${AWS::AccountId}:domain/kin-vid-domain/\*"              Condition:                IpAddress:                  aws:SourceIp:                    Ref: AllowedIP            - Effect: "Allow"              Principal:                AWS: !GetAtt LambdaRole.Arn              Action: "es:\*"              Resource: !Sub "arn:aws:es:${AWS::Region}:${AWS::AccountId}:domain/kin-vid-domain/\*"    FaceCollection:      Type: AWS::Rekognition::Collection      Properties:        CollectionId: kin-vid-rekog-collection    KinesisVideoStream:      Type: AWS::KinesisVideo::Stream      Properties:        DataRetentionInHours: 1        Name: kin-vid-stream    KinesisDataStream:      Type: AWS::Kinesis::Stream      Properties:        Name: kin-vid-data-stream        ShardCount: 1    StreamProcessorRole:      Type: "AWS::IAM::Role"      Properties:        AssumeRolePolicyDocument:          Version: "2012-10-17"          Statement:            - Effect: Allow              Principal:                Service:                  - rekognition.amazonaws.com              Action:                - "sts:AssumeRole"        ManagedPolicyArns:          - arn:aws:iam::aws:policy/AmazonKinesisFullAccess          - arn:aws:iam::aws:policy/service-role/AmazonRekognitionServiceRole    StreamProcessor:      Type: AWS::Rekognition::StreamProcessor      Properties:        FaceSearchSettings:          CollectionId: kin-vid-rekog-collection          FaceMatchThreshold: 80        KinesisDataStream:          Arn: !GetAtt KinesisDataStream.Arn        KinesisVideoStream:          Arn: !GetAtt KinesisVideoStream.Arn        Name: kin-vid-stream-processor        RoleArn: !GetAtt StreamProcessorRole.Arn    LambdaFunction:      Type: AWS::Lambda::Function      Properties:        Architectures:          - x86\_64        Code:          S3Bucket: !Ref S3Storage          S3Key: !Ref LambdaSource        Environment:          Variables:            ES\_URL: !GetAtt OpenSearchDomain.DomainEndpoint            REGION: !Sub "${AWS::Region}"        FunctionName: kin-dstrm-srch-lambda        Handler: dstrm-srch-fn.on\_message        PackageType: Zip        Role: !GetAtt LambdaRole.Arn        Runtime: python3.10    LambdaInvocation:      Type: AWS::Lambda::EventSourceMapping      Properties:        BatchSize: 10        Enabled: True        EventSourceArn: !GetAtt KinesisDataStream.Arn        FunctionName: !GetAtt LambdaFunction.Arn        MaximumRetryAttempts: 1        StartingPosition: LATEST    Cloud9Instance:      Type: AWS::Cloud9::EnvironmentEC2      Properties:        AutomaticStopTimeMinutes: 240        ConnectionType: CONNECT\_SSH        Description: Dev environment        ImageId: resolve:ssm:/aws/service/cloud9/amis/ubuntu-18.04-x86\_64        InstanceType: m5.large        Name: kin-vid-test-env  Outputs:    OpenSearchURL:      Value: !GetAtt OpenSearchDomain.DomainEndpoint |

* An Open Search Domain is created to visualize the face recognition results.
* Grant Kinesis and Open Search Domain permissions to the Lambda Role.
  1. **Setup Raspberry Pi**

Burn Raspberry Pi 64-bit OS image into the microSD card. Insert the microSD card into the Raspberry Pi and connect the power supply to the Raspberry Pi. Once the Raspberry Pi has booted, run a terminal on Raspberry Pi and execute the following commands to install dependencies: -

|  |
| --- |
| sudo apt update  sudo apt install -y \  automake \  build-essential \  cmake \  git \  gstreamer1.0-plugins-base-apps \  gstreamer1.0-plugins-bad \  gstreamer1.0-plugins-good \  gstreamer1.0-plugins-ugly \  gstreamer1.0-tools \  gstreamer1.0-omx-generic \  libcurl4-openssl-dev \  libgstreamer1.0-dev \  libgstreamer-plugins-base1.0-dev \  liblog4cplus-dev \  libssl-dev \  pkg-config |

* 1. **Configure AWS credentials**
* Using AWS CLI, run the following command to get temporary credentials ("AccessKeyId", "SecretAccessKey", "SessionToken”).

|  |
| --- |
| aws sts get-session-token |

* Open the **Raspberry Pi terminal** and execute the following commands.

|  |
| --- |
| export AWS\_DEFAULT\_REGION="The region you use (e.g. us-east-1)"  export AWS\_ACCESS\_KEY\_ID="The AccessKeyId value of the result above"  export AWS\_SECRET\_ACCESS\_KEY="The SecretAccessKey value of the result above"  export AWS\_SESSION\_TOKEN="The SessionToken value of the result above" |

1. **Kinesis Video Streams**
   1. **Set up Kinesis Video Streams Producer SDK C++**

* Run the following command in the Raspberry Pi terminal to get the Kinesis Video Streams Producer SDK C++ from the GitHub repository.

|  |
| --- |
| cd  git clone https://github.com/awslabs/amazon-kinesis-video-streams-producer-sdk-cpp.git |

* Execute the following commands to download and build the dependent libraries.

|  |
| --- |
| mkdir -p ~/amazon-kinesis-video-streams-producer-sdk-cpp/build  cd ~/amazon-kinesis-video-streams-producer-sdk-cpp/build  cmake -DBUILD\_GSTREAMER\_PLUGIN=ON .. |

* Run the following command to build the SDK.

|  |
| --- |
| make |

* Use Pi Camera to capture video and upload with Amazon Kinesis Video Streams Producer SDK C++ sample application.

|  |
| --- |
| cd ~/amazon-kinesis-video-streams-producer-sdk-cpp/build  while true; do libcamera-vid -t 15000 -o ~/kin-vid-raw.h264 && ffmpeg -y -r 30 -i ~/kin-vid-raw.h264 -vcodec copy ~/kin-vid.mp4 && ./kvs\_gstreamer\_sample kin-vid-stream ~/kin-vid.mp4 && sleep 15s; done |

Or upload any video for analysis.

|  |
| --- |
| while true; do ./kvs\_gstreamer\_sample kvs-workshop-stream ~/<YOUR-VIDEO-NAME.mp4> && sleep 20s; done |

A screenshot of a computer screen

Description automatically generated

Video streams can be sent from any device. For example, I also created a Cloud9 instance resource with the Cloud Formation and used the same steps as above to build Amazon Kinesis Video Streams Producer SDK C++ and upload videos.A screenshot of a computer

Description automatically generated

* 1. **Check the Video Stream**
* Open the Amazon Kinesis Video Streams console.
* Select Video streams in the left menu.
* Select kin-vid-stream in the list of streams.
* CloudFormation has created the stream.
* Select the Media playback part to expand it, and you will see the video. A screenshot of a computer

  Description automatically generated

1. **Analyze video with Amazon Rekognition**
   1. **Setup a collection**

* Save photos of the faces to be recognized in Amazon Rekognition on your PC. Register at least 2 or 3 people for the following steps.
* Prepare a PNG or JPEG image file of 15MB or less, with the face covering most of the image.
* The name of the image file should be something like name.jpg (e.g. katniss.jpg). The name is used as the image ID when the image is added.
* Create a bucket (kin-vid-rekog-bucket) in the same region as the cloud formation. Block all public access. Upload your images into the bucket.
* Register faces to the collection:
  + Set the S3 bucket name you just created and Collection ID as environment variables.

|  |
| --- |
| export COLLECTION\_ID=kin-vid-rekog-collection  export BUCKET\_NAME=" kin-vid-rekog-bucket " |

* + Run the following commands on the PC terminal to add all the images you uploaded to the S3 bucket to the collection.

|  |
| --- |
| for key in $(aws s3 ls s3://$BUCKET\_NAME | awk '{print $4}'); do  name=$(echo $key | sed 's/\.[^\.]\*$//')  echo "index: $key"  aws rekognition index-faces --collection-id $COLLECTION\_ID \  --image "S3Object={Bucket=$BUCKET\_NAME,Name=$key}" \  --external-image-id $name \  --max-faces=1  done |

* Confirm the number of faces registered in the collection.

|  |
| --- |
| aws rekognition describe-collection --collection-id $COLLECTION\_ID |

* You will get the following output and check the value of FaceCount.

|  |
| --- |
| {  "FaceCount": 5,  "FaceModelVersion": "6.0",  "CollectionARN": "arn:aws:rekognition:your-region:012345678901:collection/kin-vid-rekog-collection",  "CreationTimestamp": 1588237022.79  } |

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

* 1. **Configure stream processor**
* CloudFormation has created a stream processor. Run the following command to confirm the configuration of the stream processor.

|  |
| --- |
| aws rekognition describe-stream-processor --name kin-vid-stream-processor |

* You should see the input as kin-vid-stream stream and the output as kin-vid-data-stream stream. The video will be analyzed for face search with kin-vid-rekog-collection collection.
* Run the following command to start the stream processor.

|  |
| --- |
| aws rekognition start-stream-processor --name kin-vid-stream-processor |

* Confirm the stream processor is running by checking the faces recognized on Amazon Rekognition Video.
  + Open Amazon Rekognition Console.
  + Click on the Metrics on the left menu.
  + Select the 1 hour at the top right and make sure the call is successful and faces are detected. A screenshot of a computer

    Description automatically generated A screenshot of a computer

    Description automatically generated
* Confirm that the recognition results have been sent to Amazon Kinesis Data Streams.
  + Open Amazon Kinesis Data Streams Console.
  + Select Data streams from the menu on the left and select kin-vid-data-stream.
  + Open the Monitoring tab and make sure that data is being put. A screenshot of a computer

    Description automatically generated A screenshot of a computer

    Description automatically generated A screenshot of a computer

    Description automatically generated
  1. **Send results to Open Search**
* CloudFormation has created a Lambda function and an OpenSearch domain. The Lambda function sends the recognition result to the OpenSearch Service.
* Check the Lambda function from the AWS Lambda Console by selecting the kin-dstrm-srch-lambda function.
* Open the Amazon OpenSearch Service console.
* Select Domains from the left-hand menu and select kin-vid-domain
* Select the URL for General information -> OpenSearch Dashboards URL
* After opening the OpenSearch dashboard, set the index pattern.
  + Open left menu
  + Select Discover > Index Patterns on the left, and select Create index pattern on the right
  + Select index pattern: ‘face' and click Next.
  + Select the Time Filter field name: timestamp.
  + Select Create index pattern.
* Check the records registered in Amazon OpenSearch Service on OpenSearch Dashboards.
  + Select the compass icon on the left menu.
  + You can check the name and confidence at each time on the screen. A screenshot of a computer

    Description automatically generated
* Visualize the results on OpenSearch Dashboard.
  + Select the chart icon on the left, then select Create New Visualization.
  + Select the Vertical Bar.
  + In New Data Table / Choose a source, select face\*
  + In the Metrics section on the right, set the following.
  + Repeat the following process for each person in your collection.
  + Select Metric > + Add and select Y-Axis.
  + Aggregation: Average Bucket
  + bucket
    - Aggregation: Filters
    - Filter 1: name.keyword == the name of the person in the collection (e.g., name.keyword == katniss)
  + Metric
* Aggregation: Count
  + Custom label: Your name (e.g. katniss)
  + Select Buckets > + Add and select X-Axis.
* Configure the following settings.
  + Aggregation: Date histogram
* Select the Update button to apply the changes.
* The graph will be displayed to show how many times each person is recognized at each time. A screenshot of a computer

  Description automatically generated

1. **Clean Up**

* Delete CloudFormation (kin-vid-cloud kin-vid-cloud-formation-bucket).
* Delete Amazon S3 Buckets created (kin-vid-cloud-form and kin-vid-rekog-bucket)
* Delete CloudWatch logs for lambda function (aws/lambda/ kin-dstrm-srch-lambda).