CLOUD COMPUTING

/AI as a SERVICE (AIaaS)
/AWS REKOGNITION

Dr Mohammed Kaleem
m.kaleem@mmu.ac.uk

Manchester Metropolitan University

CONTENTS

AWS REKOGNITION – What is it?

Rekognition Image lets you easily build powerful applications to search, verify, and organize millions of images.

Rekognition Video lets you extract motion-based context from stored or live stream videos and helps you analyse them.

https://aws.amazon.com/rekognition/faqs/?nc=sn&loc=7

AWS REKOGNITION – What is it?

Amazon Rekognition is **a machine learning product by AWS** that helps in adding **image and video analysis to applications**. It uses deep learning to analyse images and videos.

Rekognition Image **lets you easily build powerful applications to search, verify, and organize millions of images.** Rekognition Video lets you extract motion-based context from stored or live stream videos and helps you analyze them.

Rekognition Image is an image recognition service that detects objects, scenes, activities, landmarks, faces, dominant colors, and image quality. Rekognition Image also extracts text, recognizes celebrities, and identifies inappropriate content in images. It also allows you to search and compare faces.

https://aws.amazon.com/rekognition/faqs/?nc=sn&loc=7

AWS REKOGNITION – How does it work?

Image & Video Input

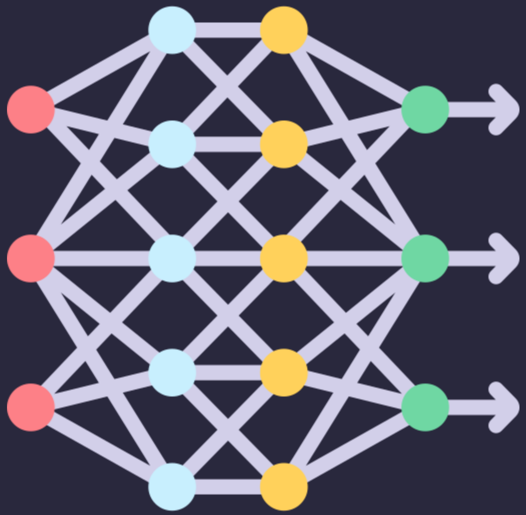
- **Users upload images** or videos to AWS Rekognition via the AWS Console, SDKs, or API.
- It can analyse stored images, real-time video streams, or videos stored in Amazon S3.

Processing & Analysis

- Rekognition **applies deep learning models trained on vast datasets** to detect and label objects, people, text, and more.
- For videos, it processes frames over time to track faces and detect activities.

Output & Results

- The service returns structured data (**JSON format**) with labels, confidence scores, bounding boxes (for objects and faces), or recognized text.



AWS REKOGNITION – Key features

Object & Scene Detection: Identifies objects (e.g., cars, animals, furniture) and scenes (e.g., beach, city, forest).

Facial Analysis: Detects faces and provides attributes (age range, emotions, glasses, beard, etc.).

Facial Recognition & Comparison: Matches faces against a database (e.g., security systems).

Text Detection (OCR): Extracts printed and handwritten text from images.

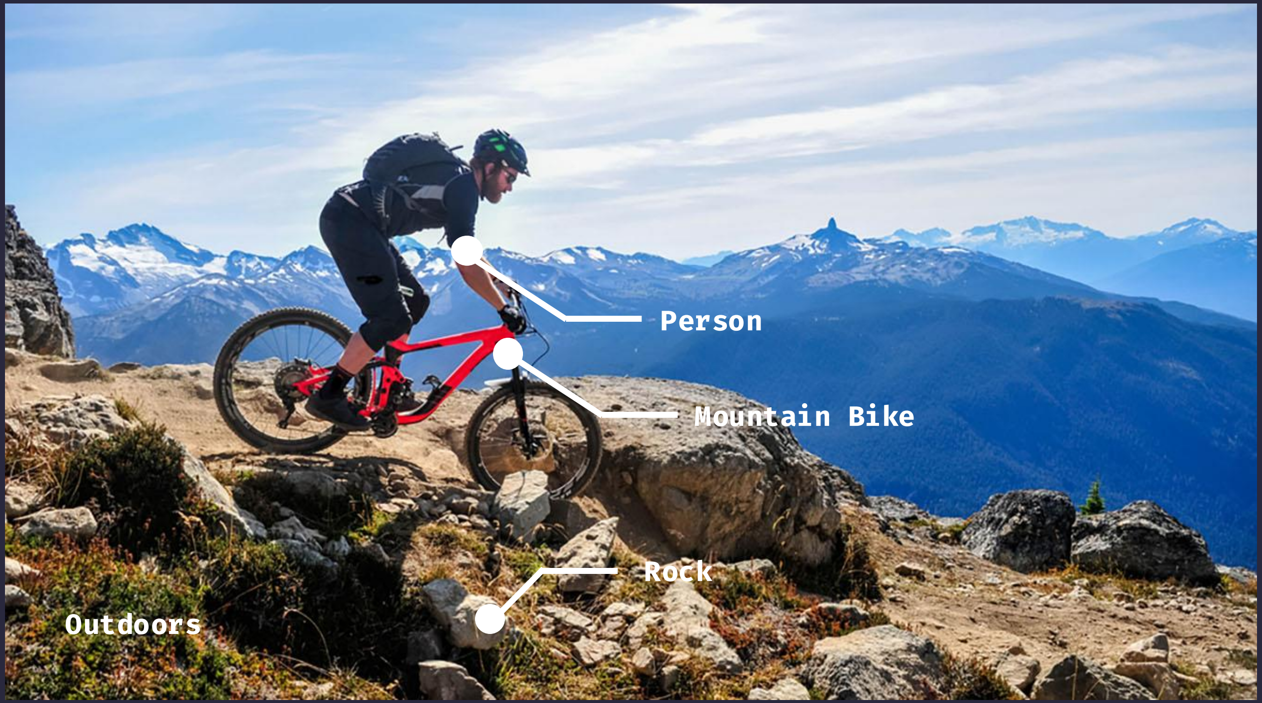
Celebrity Recognition: Identifies famous personalities in media.

Unsafe Content Detection: Flags nudity, violence, or other explicit content.

Custom Labels: Users can train Rekognition to identify specific objects not covered by default.

AWS REKOGNITION – What can it do?

Label images

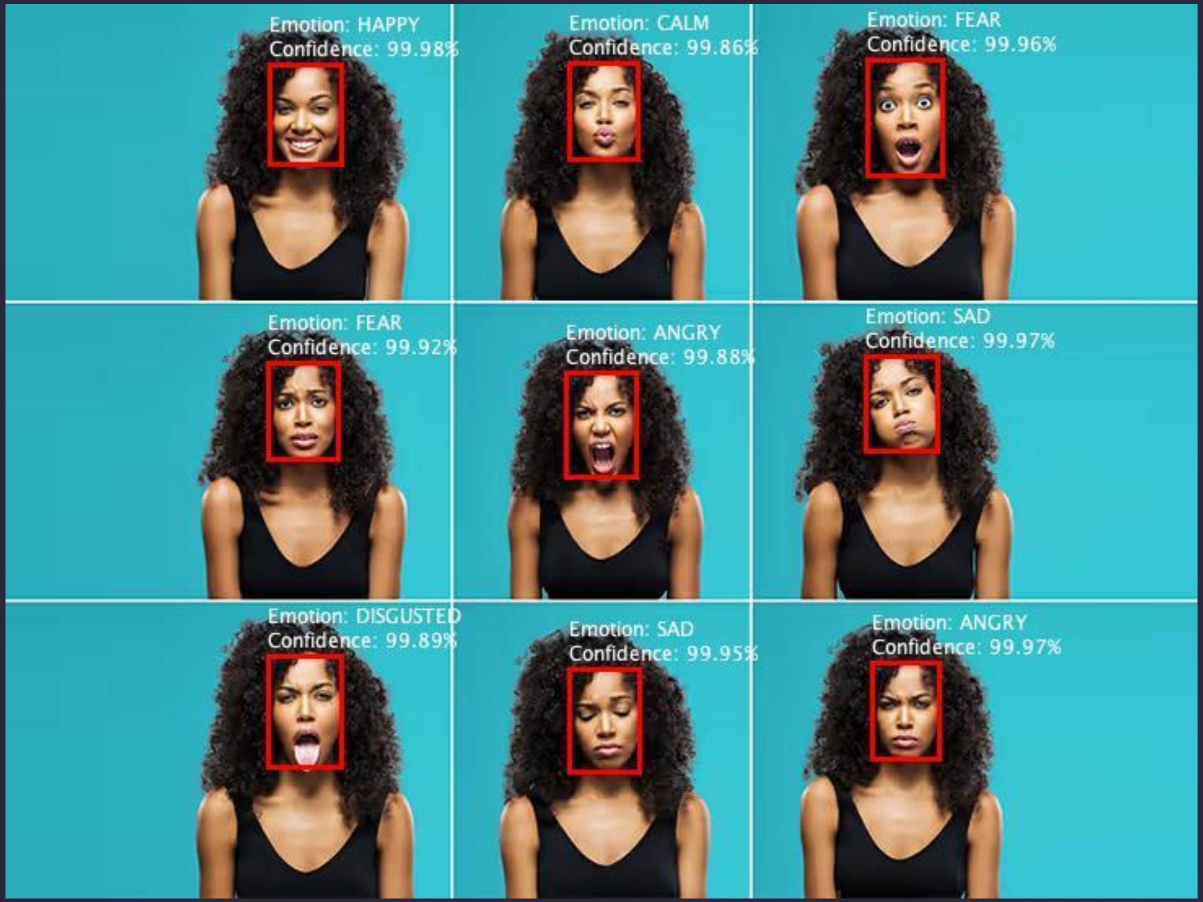


AWS REKOGNITION – What can it do?

Detect Faces

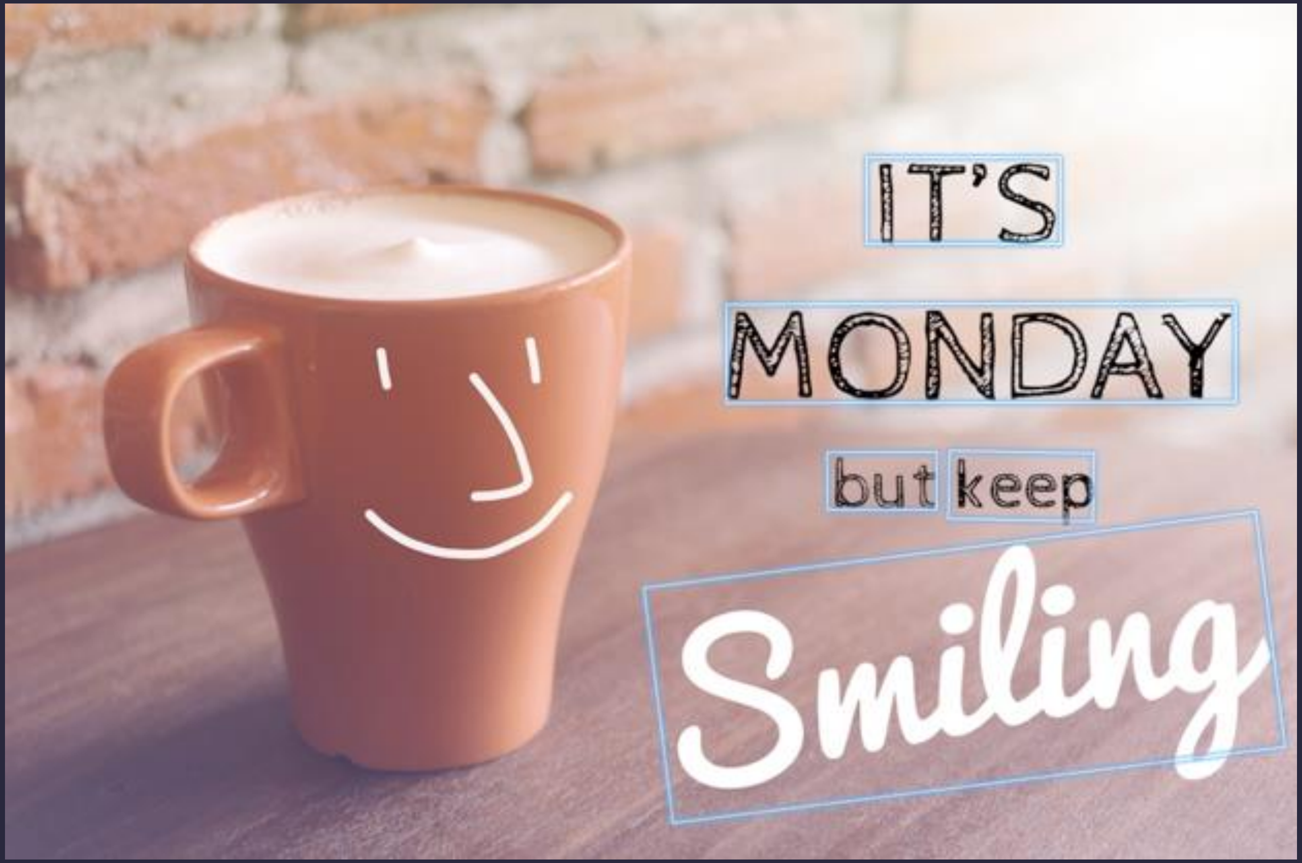
AWS REKOGNITION – What can it do?

Detect Emotion



AWS REKOGNITION – What can it do?

Detect Text



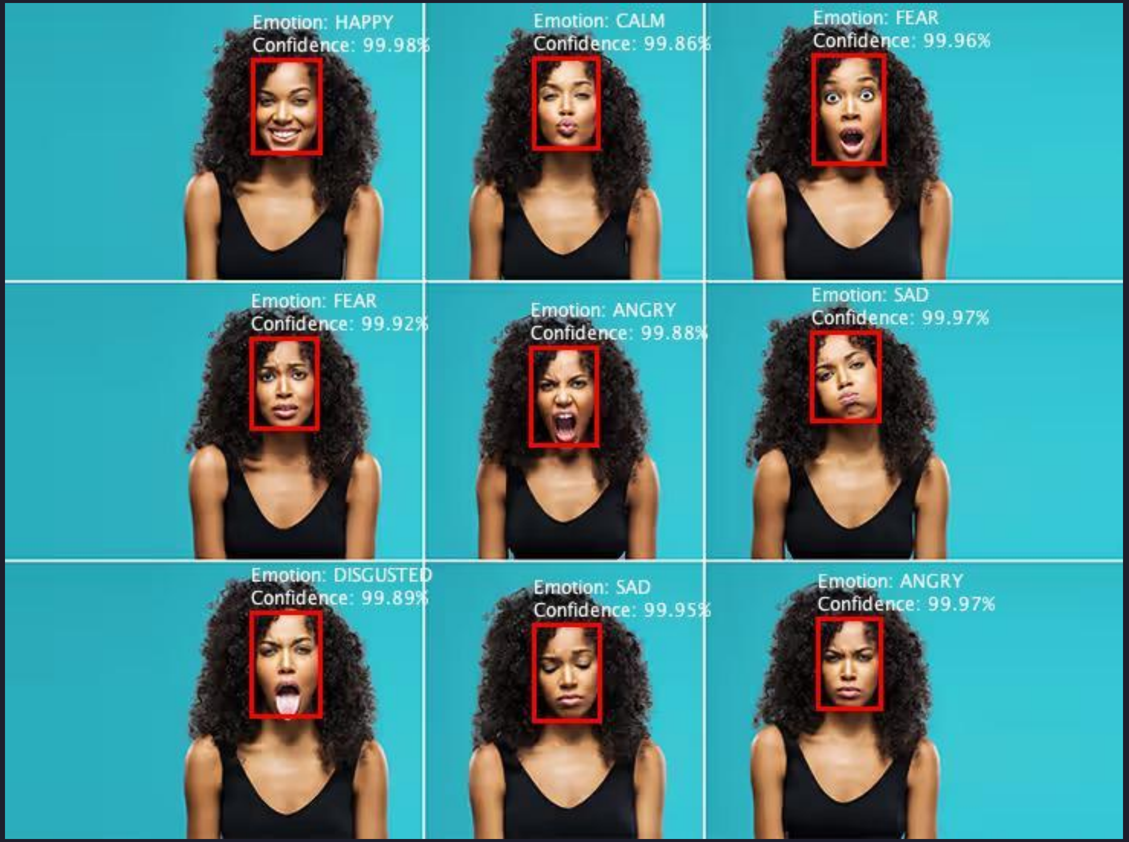
AWS REKOGNITION – What can it do?

Face analysis, detection and labeling

Amazon Rekognition allows us to detect faces in a video or image.

It can also detect things such as gender, glasses, facial hair, age range, etc., of each of those faces.

It can also help us in tracing the changes in these features over time in a video.



AWS REKOGNITION – USE CASES

Security & Authentication

✓ **Face-Based Access Control**: Authenticate employees or customers using facial recognition.

✓ **Surveillance & Intrusion Detection**: Identify unauthorized individuals in security camera footage.

✓ **Law Enforcement & Missing Persons**: Compare suspect images with criminal databases or locate missing persons.

Content Moderation & Compliance

✓ **Social Media & Streaming Platforms**: Detect inappropriate, violent, or explicit content in user-uploaded media.

✓ **E-commerce & Ads**: Ensure compliance with advertising policies by filtering out restricted content.

✓ **Child Safety**: Identify harmful imagery and prevent exploitation.

Retail & Customer Engagement

✓ **Personalized Shopping Experiences**: Recognize VIP customers and offer personalized services.

✓ **Automated Checkout Systems**: Identify products using object detection for cashier-less stores.

✓ **Sentiment Analysis**: Analyze customer emotions and engagement in retail spaces.

AWS REKOGNITION – USE CASES

Healthcare & Medical Imaging

- ✓ **Patient Identification**: Verify patient identity for secure access to medical records.
- ✓ **Skin Disease Detection**: Recognize patterns in dermatological conditions for early diagnosis.
- ✓ **Medical Research & Analysis**: Detect anomalies in X-rays, MRIs, or other medical images.

Automotive & Transportation

- ✓ **Driver Monitoring Systems**: Detect drowsiness or distracted driving in real time.
- ✓ **Traffic & Accident Analysis**: Identify vehicle types, license plates, or traffic violations.
- ✓ **Automated Toll Systems**: Use license plate recognition for seamless toll collection.

Document Processing & OCR

- **Automated Invoice & Receipt Processing**: Extract text from scanned documents for financial processing.
- **ID Verification**: Validate driver's licenses, passports, and other IDs.
- **Digitizing Handwritten Records**: Convert handwritten notes into searchable digital text.

AWS REKOGNITION – USE CASES

Entertainment & Media

- ✓ **Celebrity Recognition**: Identify famous personalities in movies, sports events, or public appearances.
- ✓ **Video Indexing & Search**: Automatically tag and categorize video content for easier searchability.
- ✓ **Sports Analytics**: Track players and detect key moments in sports footage.

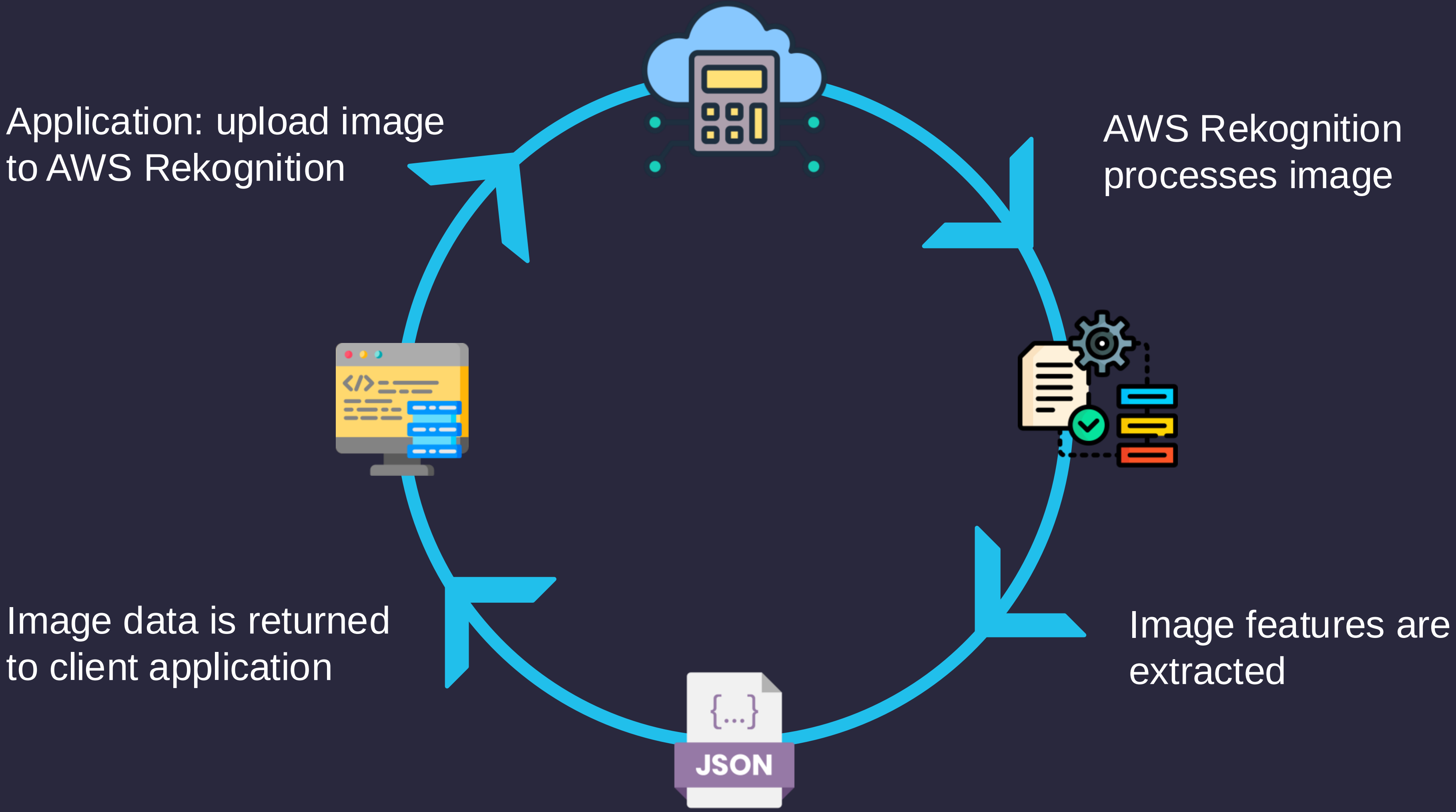
Manufacturing & Quality Control

- ✓ **Defect Detection**: Identify damaged or defective products on assembly lines.
- ✓ **Inventory Management**: Track and categorize items using object detection.
- ✓ **Workplace Safety Monitoring**: Ensure workers wear safety gear (helmets, gloves, vests).

Smart Cities & Public Safety

- ✓ **Crowd Monitoring**: Detect large gatherings for public event management.
- ✓ **Traffic Management**: Analyse road congestion and optimize traffic flow.
- ✓ **Environmental Monitoring**: Identify pollution levels, wildfires, or illegal waste dumping.

AWS REKOGNITION – HIGH LEVEL OVERVIEW


Application: upload image to AWS Rekognition
AWS Rekognition processes image
Image features are extracted
Image data is returned to client application
{...}
JSON

AWS REKOGNITION API CALL RESULT

When you call the Rekognition API, AWS Rekognition analyzes an image and **returns a JSON formatted object** that contains data on **objects**, **scenes**, and **concepts** detected in the image, along with **confidence scores** and **bounding box data** (if applicable).

```
{
 "Labels": [
  {
   "Name": "Dog",
   "Confidence": 98.5,
   "Instances": [
    {
     "BoundingBox": {
      "Width": 0.3,
      "Height": 0.4,
      "Left": 0.2,
      "Top": 0.3
     },
     "Confidence": 98.5
    }
   ],
   "Parents": []
  },
  {
   "Name": "Animal",
   "Confidence": 99.2,
   "Instances": [],
   "Parents": ["Dog"]
  },
  {
   "Name": "Park",
   "Confidence": 95.4,
   "Instances": [],
   "Parents": []
  },
  {
   "Name": "Outdoors",
   "Confidence": 97.8,
   "Instances": [],
   "Parents": []
  }
 ],
 "LabelModelVersion": "3.0"
}
```

AWS REKOGNITION REPONSE KEY ELEMENTS

Each object represents a recognized feature (e.g., **Dog**, **Animal**, **Park**, **Outdoors**). This differs slightly depending on the API called (i.e. text, face, label etc).

Field	**Description**
Name	The label describing the detected object or scene.
Confidence	The confidence score (0-100) indicating the accuracy of detection.
Instances	If the label is an object (e.g., "Dog"), this contains **bounding boxes** for detected instances.
Parents	A hierarchy showing broader categories for the label (e.g., "Animal" is a parent of "Dog").
BoundingBox	Specifies where the object is in the image (values between 0-1, relative to image size).
Confidence	Confidence score for that specific instance.

HOW TO USE THE RESPONSE DATA

Filter Results by Confidence Level

- ➢ Ignore labels with confidence < 80% to avoid false detections.

Use Labels to Categorize Images

- ➢ Example: If "Beach" is detected, tag the image in a travel app.

Locate Objects in Images

- ➢ If "BoundingBox" exists, use it to draw boxes around detected objects.

Group Labels Using Parent Categories

- ➢ Example: If a "Golden Retriever" is detected, look at "Parents" to classify it under "Dog".

ACCESS KEYS

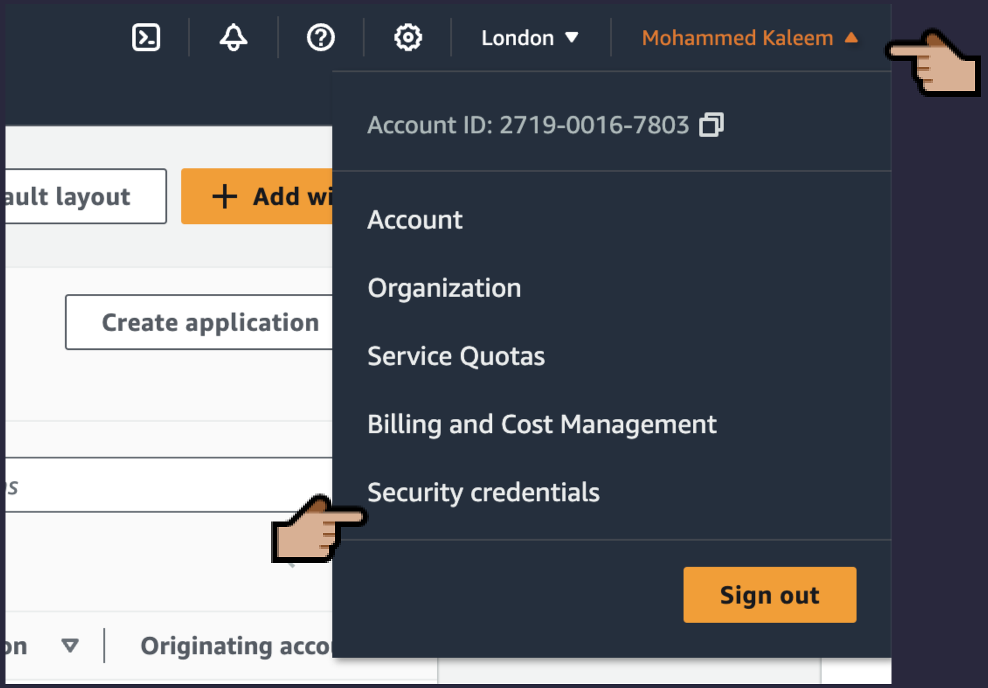
Before we start coding, we need to create access keys.

Access keys are alpha numeric strings that are unique to your account and allow you to control certain AWS services through code.

You are allowed a maximum of 2 access keys.

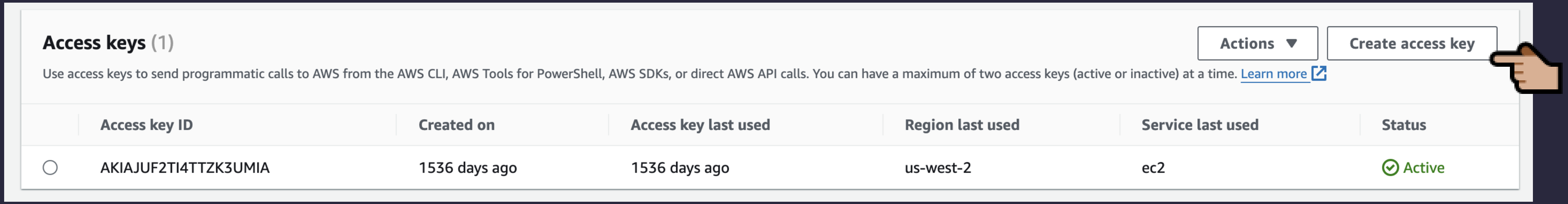
GENERATING ACCESS KEYS

Once you've logged in, click on your name (top right of screen) and select "Security credentials".



GENERATING ACCESS KEYS – continued

Scroll down till you see the access key section and select “Create access key”.

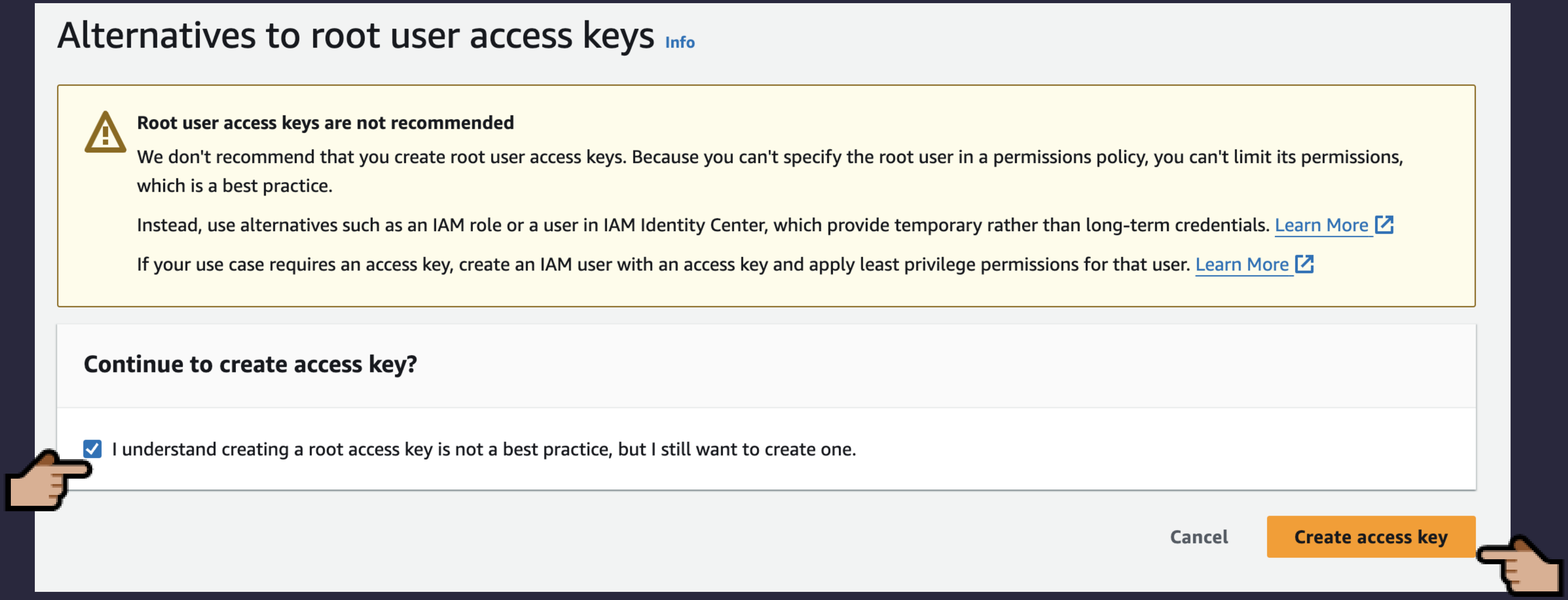


Access keys (1)

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. Learn more

	Access key ID	Created on	Access key last used	Region last used	Service last used	Status
○	AKIAJUF2TI4TTZK3UMIA	1536 days ago	1536 days ago	us-west-2	ec2	✓ Active

GENERATING ACCESS KEYS – continued



Alternatives to root user access keys Info

Root user access keys are not recommended

We don't recommend that you create root user access keys. Because you can't specify the root user in a permissions policy, you can't limit its permissions, which is a best practice.

Instead, use alternatives such as an IAM role or a user in IAM Identity Center, which provide temporary rather than long-term credentials. Learn More

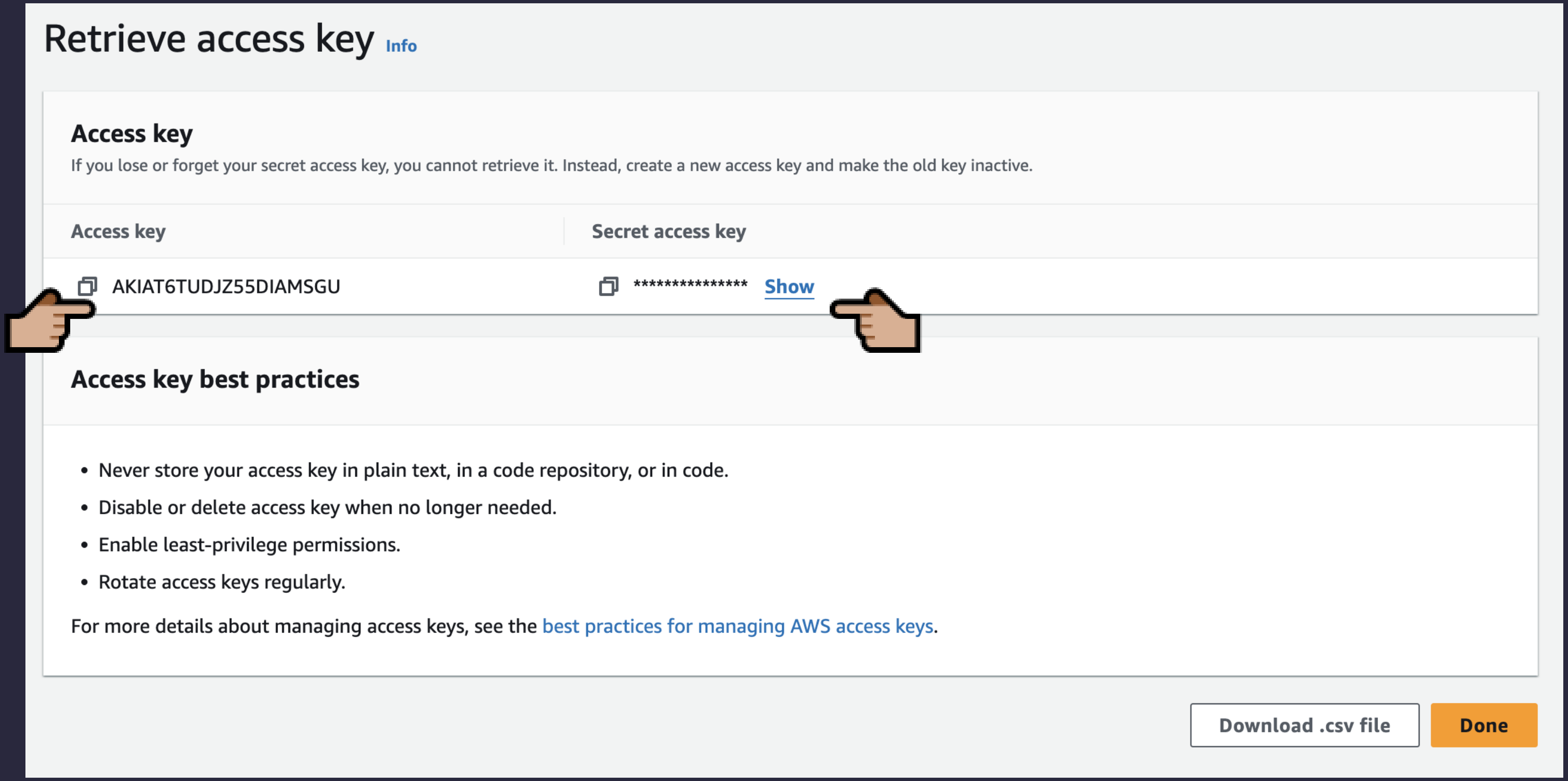
If your use case requires an access key, create an IAM user with an access key and apply least privilege permissions for that user. Learn More

Continue to create access key?

[✓] I understand creating a root access key is not a best practice, but I still want to create one.

GENERATING ACCESS KEYS – continued

Make a note of both keys (or just download them in a csv file).



Retrieve access key Info

Access key

If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
AKIAT6TUDJZ55DIAMSGU	*************** Show

Access key best practices

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the best practices for managing AWS access keys.

REQUIRED MAVEN DEPENDENCIES

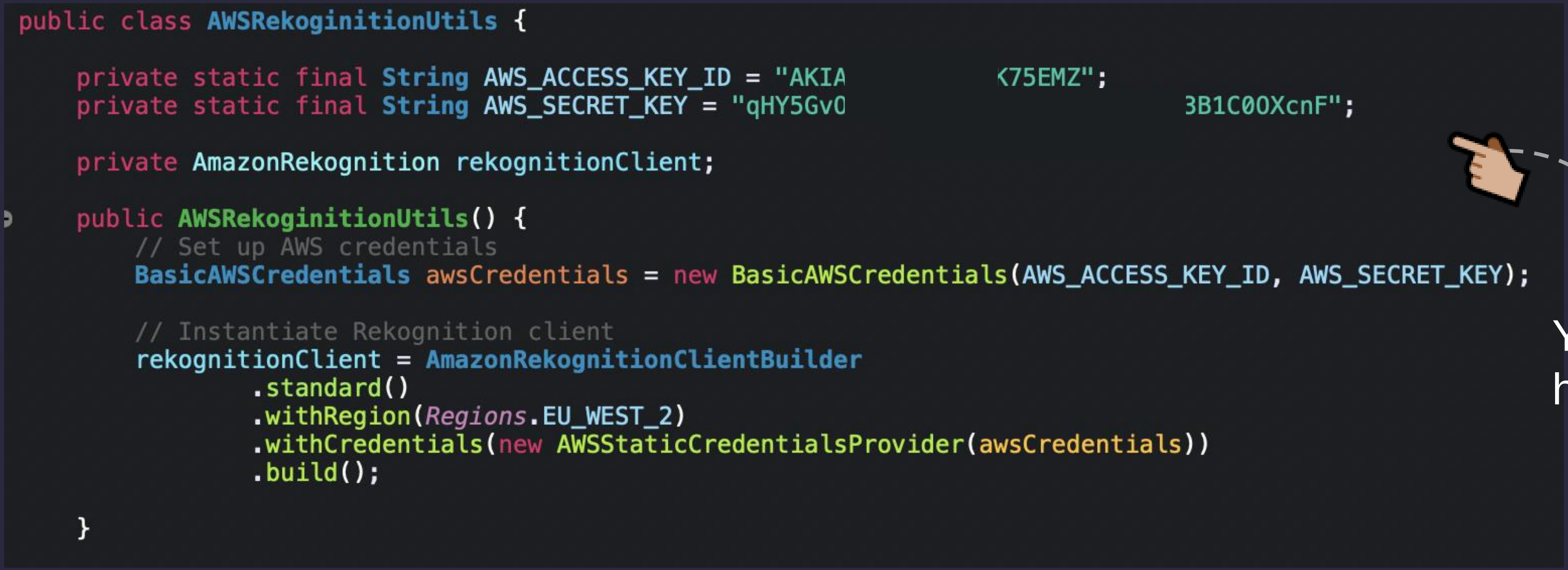
Create a new maven project (*skip archetype selection*) and add the following dependency to the pom.xml file.

```xml
<dependency>
  <groupId>com.amazonaws</groupId>
  <artifactId>aws-java-sdk-rekognition</artifactId>
  <version>1.12.114</version>
</dependency>
```

Alternatively use the starter project on eclipse.

AWSRekognition Client Object

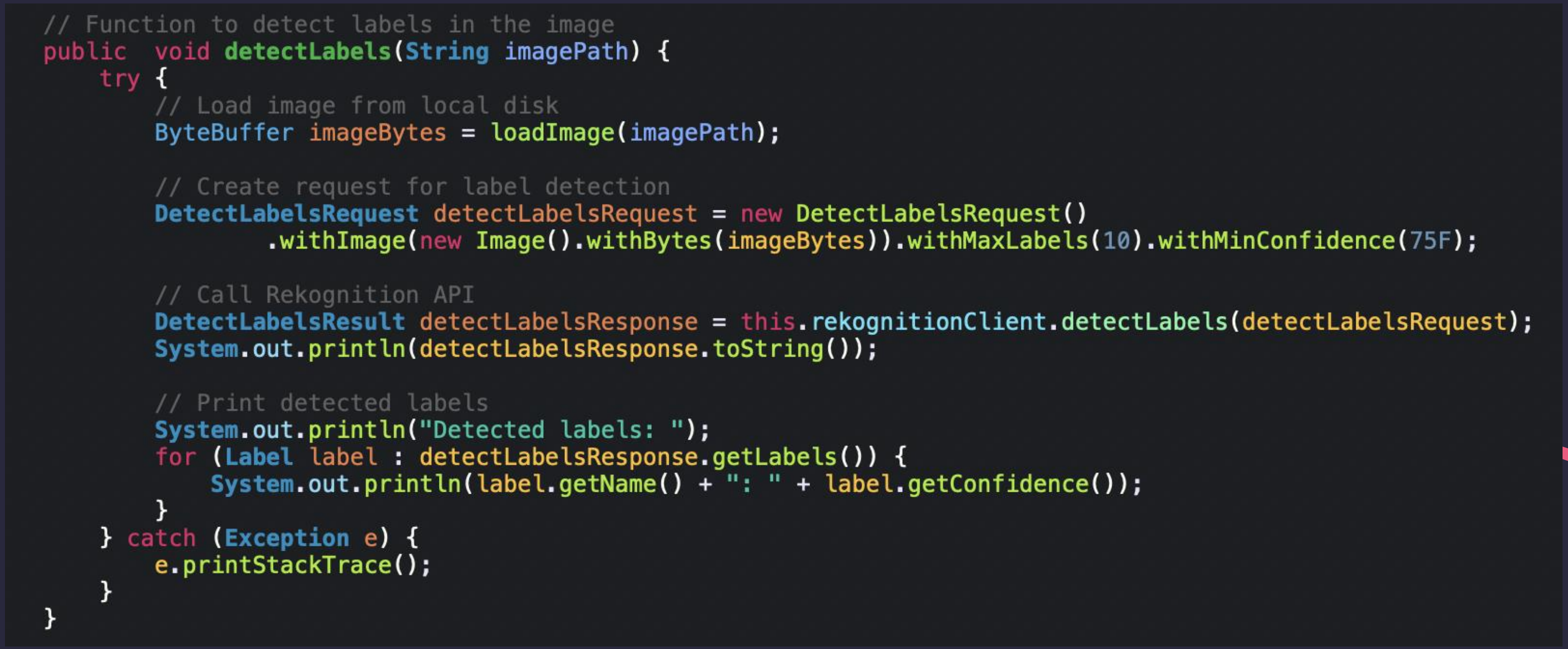
Start by instantiating a global Amazon Rekognition client object.
We will use this object in all our Rekognition methods. It is therefore, recommended that you implemented all this in a "utils" class (as shown below).



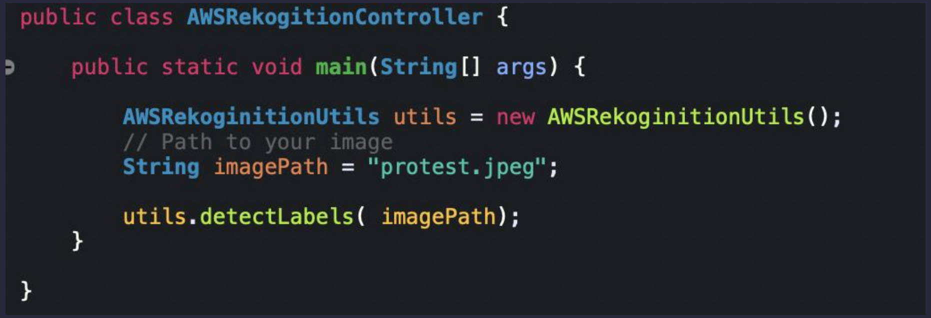
```java
public class AWSRekoginitionUtils {

    private static final String AWS_ACCESS_KEY_ID = "AKIA[redacted]K75EMZ";
    private static final String AWS_SECRET_KEY = "qHY5Gv0[redacted]3B1C0OXcnF";

    private AmazonRekognition rekognitionClient;

    public AWSRekoginitionUtils() {
        // Set up AWS credentials
        BasicAWSCredentials awsCredentials = new BasicAWSCredentials(AWS_ACCESS_KEY_ID, AWS_SECRET_KEY);

        // Instantiate Rekognition client
        rekognitionClient = AmazonRekognitionClientBuilder
                .standard()
                .withRegion(Regions.EU_WEST_2)
                .withCredentials(new AWSStaticCredentialsProvider(awsCredentials))
                .build();

    }
```

Your keys here

AWSRekognition – DETECT LABELS

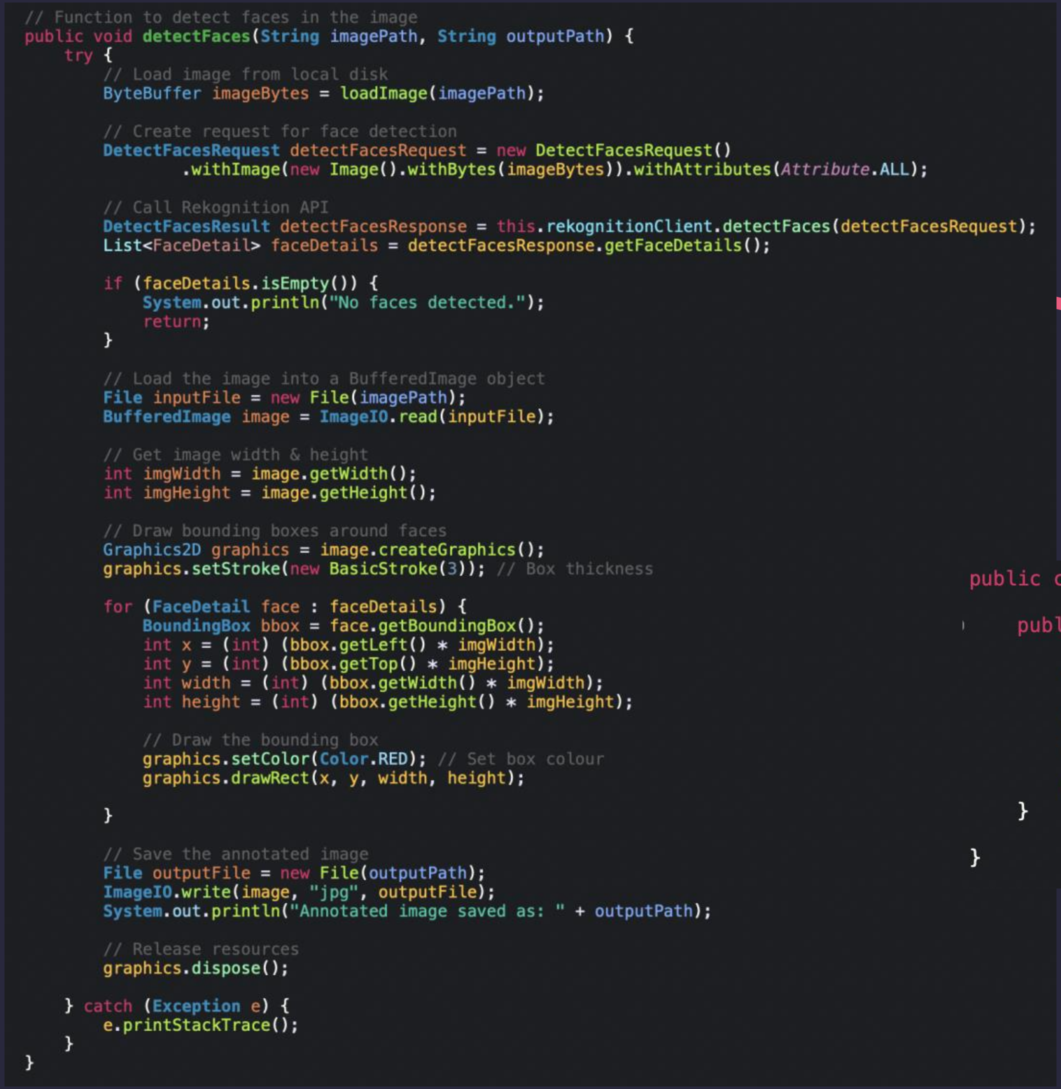


```java
// Function to detect labels in the image
public  void detectLabels(String imagePath) {
    try {
        // Load image from local disk
        ByteBuffer imageBytes = loadImage(imagePath);

        // Create request for label detection
        DetectLabelsRequest detectLabelsRequest = new DetectLabelsRequest()
                .withImage(new Image().withBytes(imageBytes)).withMaxLabels(10).withMinConfidence(75F);

        // Call Rekognition API
        DetectLabelsResult detectLabelsResponse = this.rekognitionClient.detectLabels(detectLabelsRequest);
        System.out.println(detectLabelsResponse.toString());

        // Print detected labels
        System.out.println("Detected labels: ");
        for (Label label : detectLabelsResponse.getLabels()) {
            System.out.println(label.getName() + ": " + label.getConfidence());
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

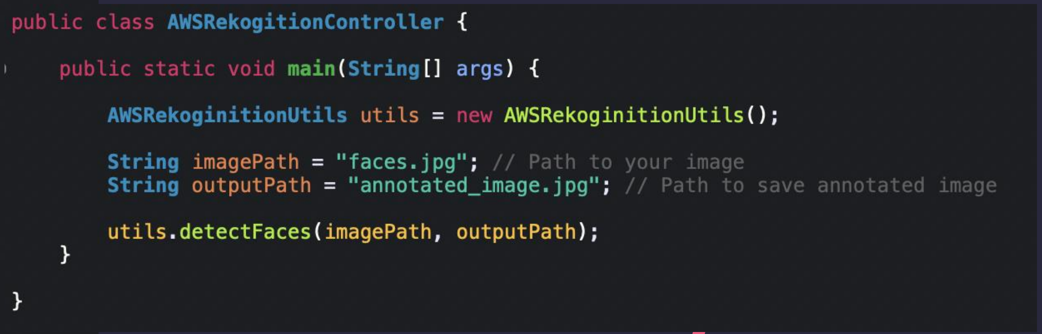


```java
public class AWSRekogitionController {

    public static void main(String[] args) {

        AWSRekoginitionUtils utils = new AWSRekoginitionUtils();
        // Path to your image
        String imagePath = "protest.jpeg";

        utils.detectLabels( imagePath);
    }

}
```

AWSRekognition – DETECT FACES

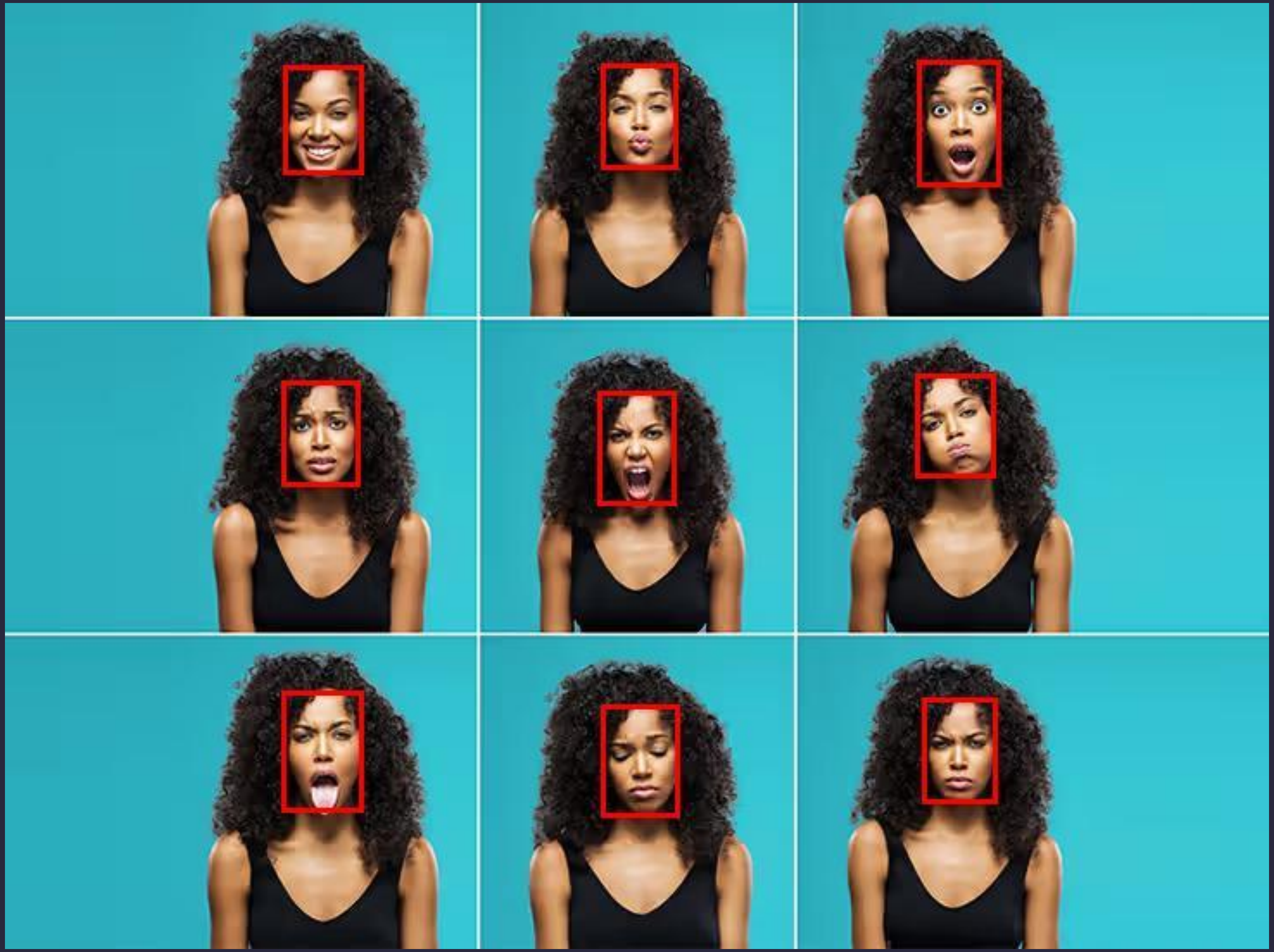


```java
// Function to detect faces in the image
public void detectFaces(String imagePath, String outputPath) {
    try {
        // Load image from local disk
        ByteBuffer imageBytes = loadImage(imagePath);

        // Create request for face detection
        DetectFacesRequest detectFacesRequest = new DetectFacesRequest()
                .withImage(new Image().withBytes(imageBytes)).withAttributes(Attribute.ALL);

        // Call Rekognition API
        DetectFacesResult detectFacesResponse = this.rekognitionClient.detectFaces(detectFacesRequest);
        List<FaceDetail> faceDetails = detectFacesResponse.getFaceDetails();

        if (faceDetails.isEmpty()) {
            System.out.println("No faces detected.");
            return;
        }

        // Load the image into a BufferedImage object
        File inputFile = new File(imagePath);
        BufferedImage image = ImageIO.read(inputFile);

        // Get image width & height
        int imgWidth = image.getWidth();
        int imgHeight = image.getHeight();

        // Draw bounding boxes around faces
        Graphics2D graphics = image.createGraphics();
        graphics.setStroke(new BasicStroke(3)); // Box thickness

        for (FaceDetail face : faceDetails) {
            BoundingBox bbox = face.getBoundingBox();
            int x = (int) (bbox.getLeft() * imgWidth);
            int y = (int) (bbox.getTop() * imgHeight);
            int width = (int) (bbox.getWidth() * imgWidth);
            int height = (int) (bbox.getHeight() * imgHeight);

            // Draw the bounding box
            graphics.setColor(Color.RED); // Set box colour
            graphics.drawRect(x, y, width, height);

        }

        // Save the annotated image
        File outputFile = new File(outputPath);
        ImageIO.write(image, "jpg", outputFile);
        System.out.println("Annotated image saved as: " + outputPath);

        // Release resources
        graphics.dispose();

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

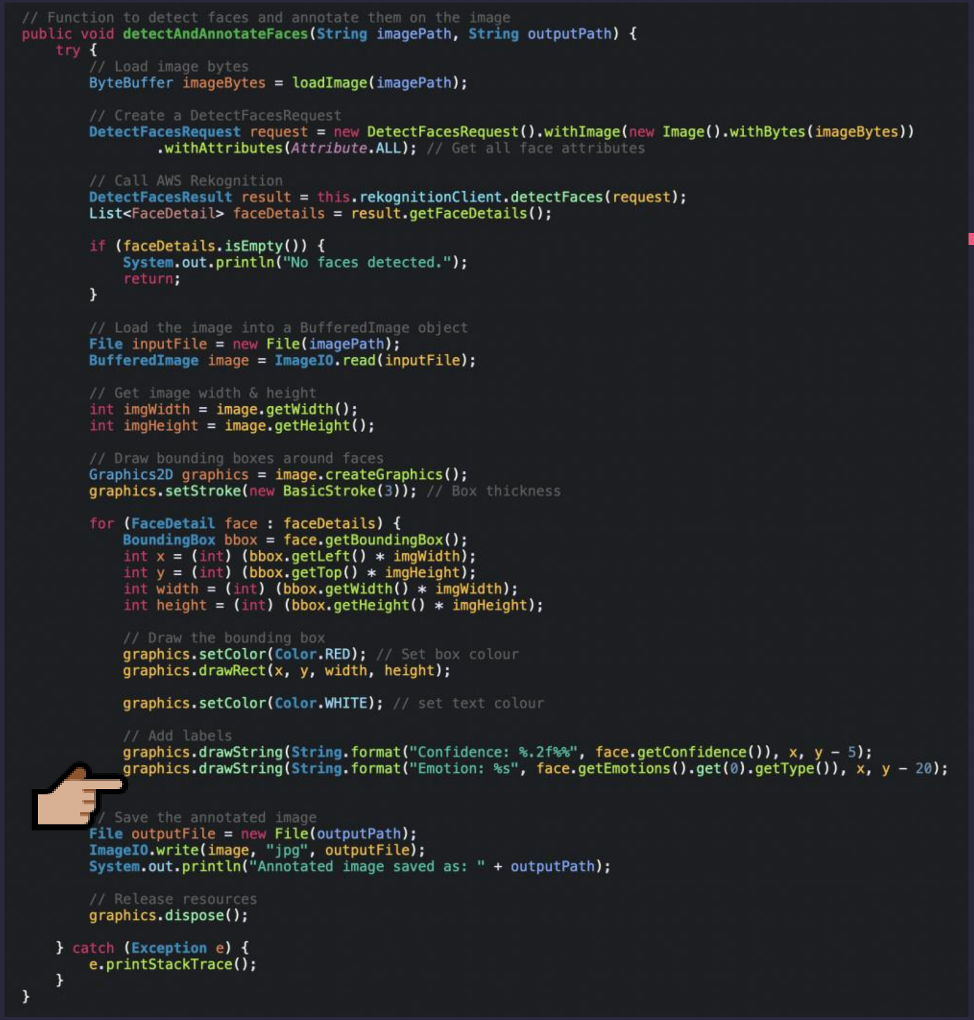


```java
public class AWSRekogitionController {

    public static void main(String[] args) {

        AWSRekoginitionUtils utils = new AWSRekoginitionUtils();

        String imagePath = "faces.jpg"; // Path to your image
        String outputPath = "annotated_image.jpg"; // Path to save annotated image

        utils.detectFaces(imagePath, outputPath);
    }

}
```

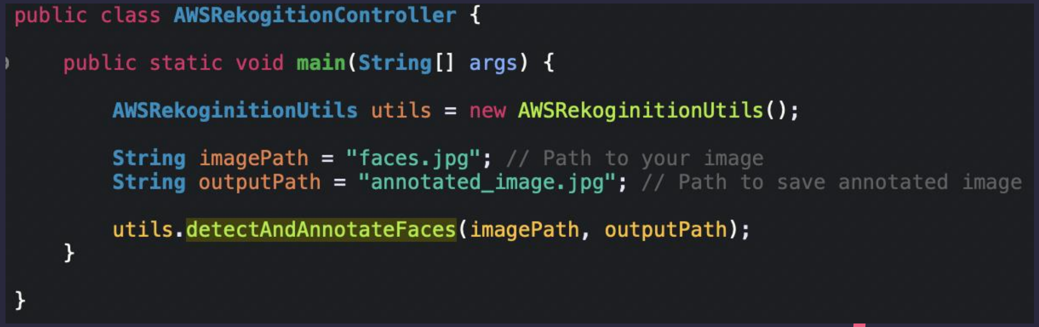
AWSRekognition – DETECT FACES RESULT



AWSRekognition – DETECT EMOTION + ANNOTATE

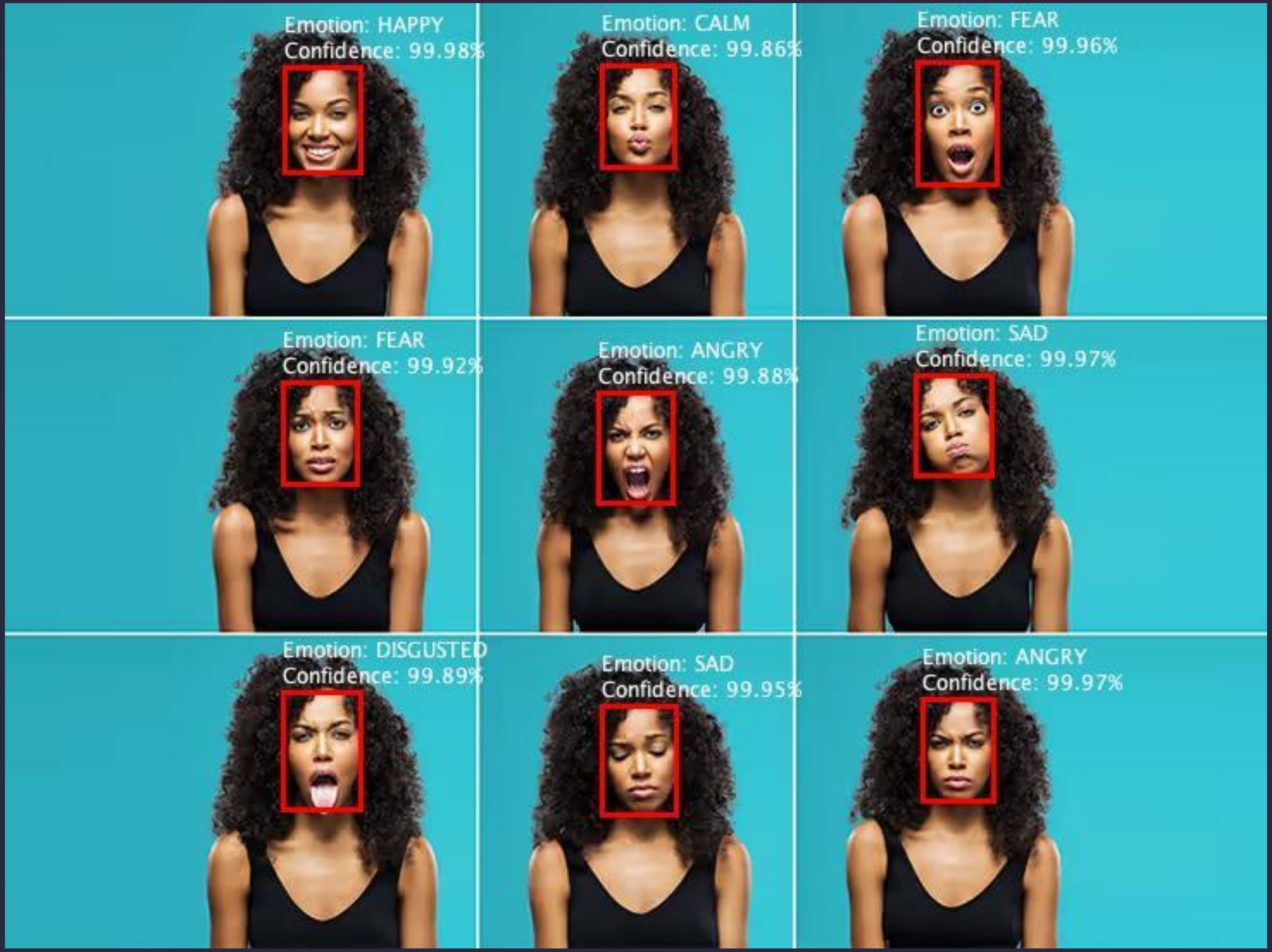


```java
// Function to detect faces and annotate them on the image
public void detectAndAnnotateFaces(String imagePath, String outputPath) {
    try {
        // Load image bytes
        ByteBuffer imageBytes = loadImage(imagePath);

        // Create a DetectFacesRequest
        DetectFacesRequest request = new DetectFacesRequest().withImage(new Image().withBytes(imageBytes))
                .withAttributes(Attribute.ALL); // Get all face attributes

        // Call AWS Rekognition
        DetectFacesResult result = this.rekognitionClient.detectFaces(request);
        List<FaceDetail> faceDetails = result.getFaceDetails();

        if (faceDetails.isEmpty()) {
            System.out.println("No faces detected.");
            return;
        }

        // Load the image into a BufferedImage object
        File inputFile = new File(imagePath);
        BufferedImage image = ImageIO.read(inputFile);

        // Get image width & height
        int imgWidth = image.getWidth();
        int imgHeight = image.getHeight();

        // Draw bounding boxes around faces
        Graphics2D graphics = image.createGraphics();
        graphics.setStroke(new BasicStroke(3)); // Box thickness

        for (FaceDetail face : faceDetails) {
            BoundingBox bbox = face.getBoundingBox();
            int x = (int) (bbox.getLeft() * imgWidth);
            int y = (int) (bbox.getTop() * imgHeight);
            int width = (int) (bbox.getWidth() * imgWidth);
            int height = (int) (bbox.getHeight() * imgHeight);

            // Draw the bounding box
            graphics.setColor(Color.RED); // Set box colour
            graphics.drawRect(x, y, width, height);

            graphics.setColor(Color.WHITE); // set text colour

            // Add labels
            graphics.drawString(String.format("Confidence: %.2f%%", face.getConfidence()), x, y - 5);
            graphics.drawString(String.format("Emotion: %s", face.getEmotions().get(0).getType()), x, y - 20);

        // Save the annotated image
        File outputFile = new File(outputPath);
        ImageIO.write(image, "jpg", outputFile);
        System.out.println("Annotated image saved as: " + outputPath);

        // Release resources
        graphics.dispose();

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

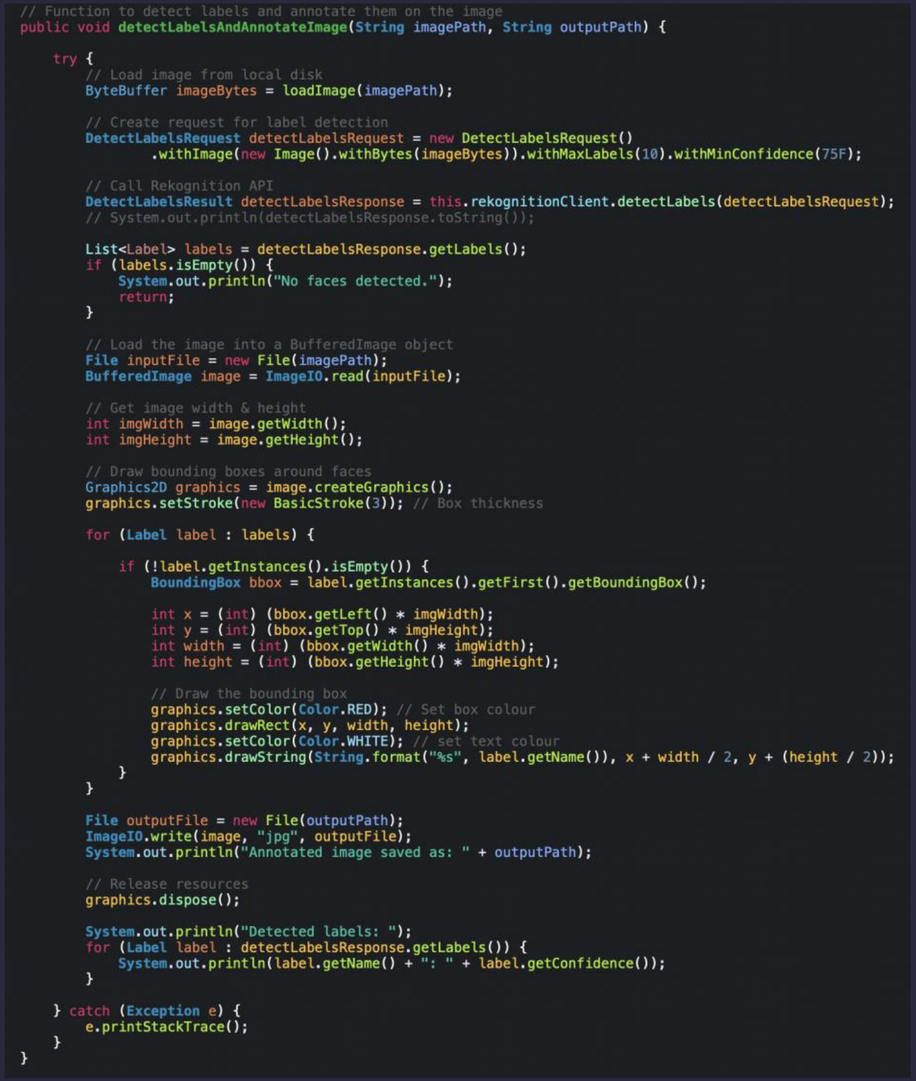


```java
public class AWSRekogitionController {

    public static void main(String[] args) {

        AWSRekoginitionUtils utils = new AWSRekoginitionUtils();

        String imagePath = "faces.jpg"; // Path to your image
        String outputPath = "annotated_image.jpg"; // Path to save annotated image

        utils.detectAndAnnotateFaces(imagePath, outputPath);
    }

}
```

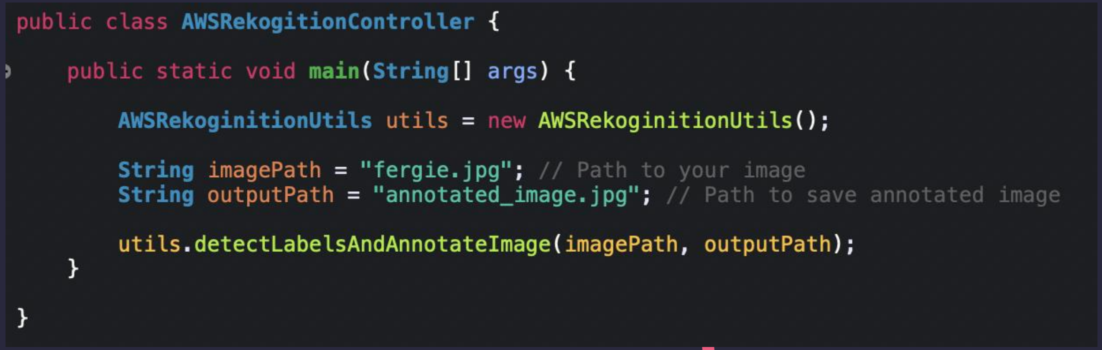
AWSRekognition – DETECT FACES + ANNOTATE RESULT



AWSRekognition – DETECT LABELS + ANNOTATE

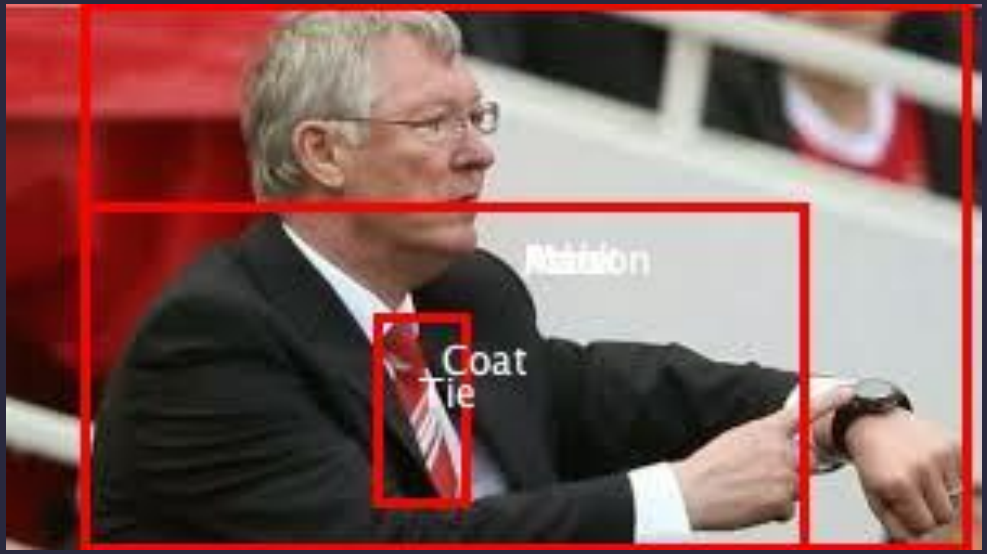


```java
// Function to detect labels and annotate them on the image
public void detectLabelsAndAnnotateImage(String imagePath, String outputPath) {

    try {
        // Load image from local disk
        ByteBuffer imageBytes = loadImage(imagePath);

        // Create request for label detection
        DetectLabelsRequest detectLabelsRequest = new DetectLabelsRequest()
                .withImage(new Image().withBytes(imageBytes)).withMaxLabels(10).withMinConfidence(75F);

        // Call Rekognition API
        DetectLabelsResult detectLabelsResponse = this.rekognitionClient.detectLabels(detectLabelsRequest);
        // System.out.println(detectLabelsResponse.toString());

        List<Label> labels = detectLabelsResponse.getLabels();
        if (labels.isEmpty()) {
            System.out.println("No faces detected.");
            return;
        }

        // Load the image into a BufferedImage object
        File inputFile = new File(imagePath);
        BufferedImage image = ImageIO.read(inputFile);

        // Get image width & height
        int imgWidth = image.getWidth();
        int imgHeight = image.getHeight();

        // Draw bounding boxes around faces
        Graphics2D graphics = image.createGraphics();
        graphics.setStroke(new BasicStroke(3)); // Box thickness

        for (Label label : labels) {

            if (!label.getInstances().isEmpty()) {
                BoundingBox bbox = label.getInstances().getFirst().getBoundingBox();

                int x = (int) (bbox.getLeft() * imgWidth);
                int y = (int) (bbox.getTop() * imgHeight);
                int width = (int) (bbox.getWidth() * imgWidth);
                int height = (int) (bbox.getHeight() * imgHeight);

                // Draw the bounding box
                graphics.setColor(Color.RED); // Set box colour
                graphics.drawRect(x, y, width, height);
                graphics.setColor(Color.WHITE); // set text colour
                graphics.drawString(String.format("%s", label.getName()), x + width / 2, y + (height / 2));
            }
        }

        File outputFile = new File(outputPath);
        ImageIO.write(image, "jpg", outputFile);
        System.out.println("Annotated image saved as: " + outputPath);

        // Release resources
        graphics.dispose();

        System.out.println("Detected labels: ");
        for (Label label : detectLabelsResponse.getLabels()) {
            System.out.println(label.getName() + ": " + label.getConfidence());
        }

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

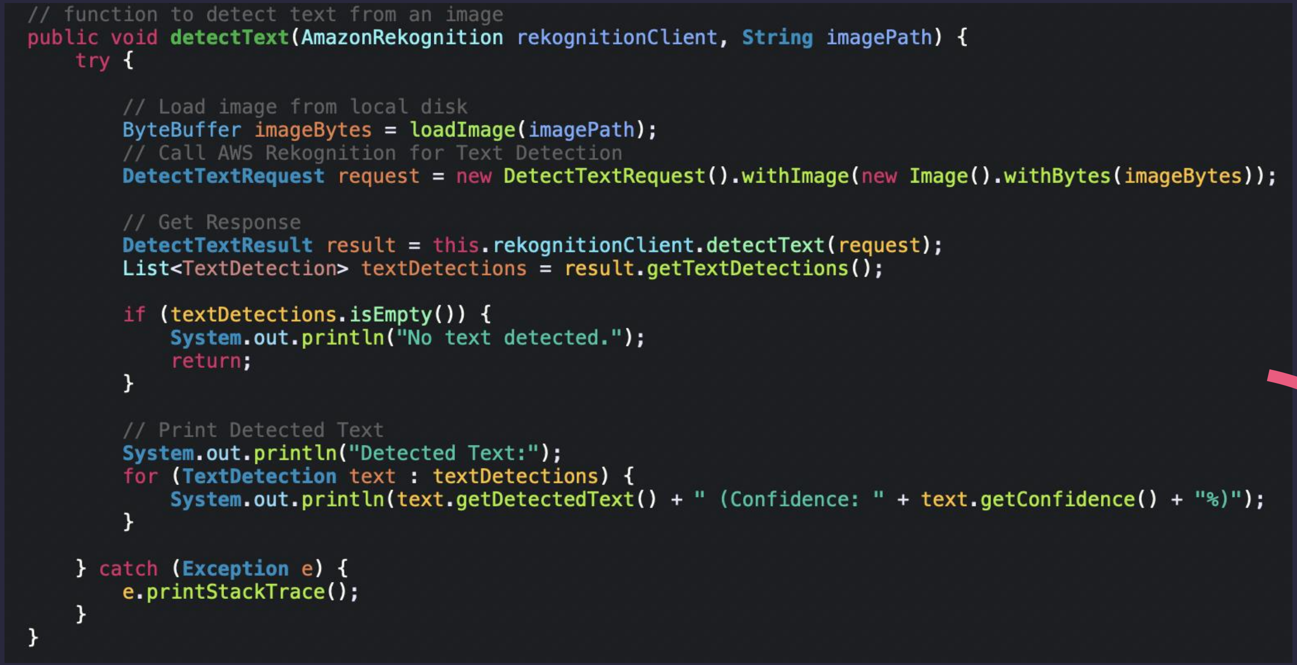


```java
public class AWSRekogitionController {

    public static void main(String[] args) {

        AWSRekoginitionUtils utils = new AWSRekoginitionUtils();

        String imagePath = "fergie.jpg"; // Path to your image
        String outputPath = "annotated_image.jpg"; // Path to save annotated image

        utils.detectLabelsAndAnnotateImage(imagePath, outputPath);
    }

}
```

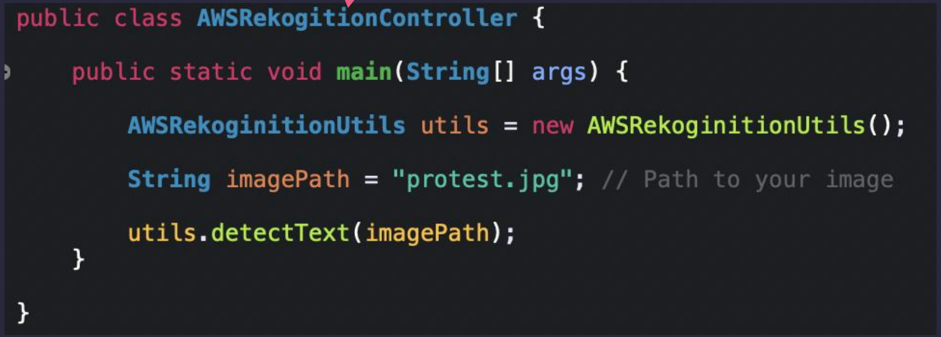
AWSRekognition – DETECT LABLES + ANNOTATE RESULT



AWSRekognition – DETECT TEXT



```java
// function to detect text from an image
public void detectText(AmazonRekognition rekognitionClient, String imagePath) {
    try {

        // Load image from local disk
        ByteBuffer imageBytes = loadImage(imagePath);
        // Call AWS Rekognition for Text Detection
        DetectTextRequest request = new DetectTextRequest().withImage(new Image().withBytes(imageBytes));

        // Get Response
        DetectTextResult result = this.rekognitionClient.detectText(request);
        List<TextDetection> textDetections = result.getTextDetections();

        if (textDetections.isEmpty()) {
            System.out.println("No text detected.");
            return;
        }

        // Print Detected Text
        System.out.println("Detected Text:");
        for (TextDetection text : textDetections) {
            System.out.println(text.getDetectedText() + " (Confidence: " + text.getConfidence() + "%)");
        }

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```



```java
public class AWSRekogitionController {

    public static void main(String[] args) {

        AWSRekoginitionUtils utils = new AWSRekoginitionUtils();

        String imagePath = "protest.jpg"; // Path to your image

        utils.detectText(imagePath);
    }

}
```

Summary

- AWS Rekognition is an excellent API for image analysis.
- It has pretrained AI models for image analysis.
- Very easy to set up and use within existing applications that require image analysis.
- Has SDKs for Java, C++, JavaScript, PHP and more

- Use maven for dependency management and start using AWS Rekognition for image analysis.