Automated Image Labeling Using AWS Rekognition

1. Objective

The goal of this project is to develop an image label detection application using AWS Rekognition. This application automatically detects objects, animals, and other entities in a dog image stored in Amazon S3, processes them using AWS Rekognition, and displays detected labels with high confidence levels. This solution leverages cloud services to demonstrate the power of server-less image recognition using Python.

---

2. Pre-requisites

1. **AWS Account**: Set up an AWS free-tier account.
2. **Basic Python Knowledge**: Understanding Python fundamentals.
3. **Software**:
    - PuTTY (or an SSH client) to connect to an AWS EC2 instance.
    - GitHub account for project sharing and version control.
4. **AWS IAM Role**: Configure an IAM role with permissions for Amazon Rekognition and S3.
5. **Amazon S3 Bucket**: Store the dog image or any file in an S3 bucket.

---

3. Technology Stack

- **Python**: Programming language used for scripting.
- **AWS Rekognition**: Is an image recognition service that detects objects, scenes, activities, landmarks, faces, dominant colors, and image quality.
- **Amazon S3**: Cloud storage used to store image files.
- **Amazon EC2**: Virtual server instance for running the script.
- **Libraries**:
    - **Boto3**: AWS SDK for Python.
    - **Pillow (PIL)**: For image manipulation.
    - **Matplotlib**: For displaying images.
- **PuTTY**: SSH client for EC2 instance connection.

---

4. Step-by-Step Guide

Step 1: Set Up AWS S3 and EC2 Resources

1. **Create an S3 Bucket**:
    - Navigate to the Amazon S3 console and create a new bucket.
    - Name the bucket (e.g., image-labels-bucket).

- Upload your dog image (e.g., Dog Image.jpg) to this bucket.

2. **Launch an EC2 Instance**:
   - Open the EC2 dashboard, select "Launch Instance," and choose an Amazon Linux 2 AMI.
   - Select an instance type (e.g., t3.micro for free-tier eligibility).
   - Configure an IAM role with S3 and Rekognition permissions and attach it to the instance.
   - Create a key pair for SSH access and download the .pem file.

Step 2: Connect to the EC2 Instance Using PuTTY

1. **Convert the Key File**:
   - Use PuTTYgen to convert your .pem file to a .ppk format for PuTTY.
   - Save the .ppk file.
2. **Launch PuTTY**:
   - Open PuTTY and load your .ppk key in the SSH > Auth section.
   - Enter your EC2 instance's public IP address in the "Session" section and connect.

Step 3: Set Up the Python Environment on EC2

1. **Install Python 3.8**:
   - Run commands to update packages and install Python 3.8:

     sudo yum update -y
     sudo yum install python3.8

2. **Install Python Libraries**:
   - Install Boto3, Pillow, and Matplotlib:

     pip3.8 install boto3 pillow matplotlib

Step 4: Create and Upload the Script

1. **Create a New Python Script**:
   - Open a text editor, such as nano, to create a Python file (label_generator.py):

     nano label_generator.py

Paste the Python Code:

- Copy and paste the following code into label_generator.py:

```
import boto3
import matplotlib.pyplot as plt
from PIL import Image, ImageDraw

def detect_labels(photo, bucket):
    client = boto3.client('rekognition', region_name='ap-south-1')
```

```
    # Detect labels with a maximum of 10 labels
    response = client.detect_labels(Image={'S3Object': {'Bucket': bucket, 'Name': photo}}, MaxLabels=10)
    return response['Labels']

def show_bounding_boxes(photo, bucket, labels):
    s3_client = boto3.client('s3')

    # Download the image from S3
    s3_client.download_file(bucket, photo, 'downloaded_image.jpg')
    image = Image.open('downloaded_image.jpg')

    img_width, img_height = image.size
    draw = ImageDraw.Draw(image)

    # Draw bounding boxes for each label with instances
    for label in labels:
        for instance in label.get('Instances', []):  # Use .get() to avoid KeyError if 'Instances' key is missing
            box = instance['BoundingBox']
            left = img_width * box['Left']
            top = img_height * box['Top']
            width = img_width * box['Width']
            height = img_height * box['Height']
            points = (
                (left, top),
                (left + width, top),
                (left + width, top + height),
                (left, top + height),
                (left, top)
            )
            draw.line(points, fill='#00d400', width=2)

    # Display the image with bounding boxes
    plt.imshow(image)
    plt.axis('off')
    plt.show()

def main():
    bucket = 'image-labels-bucket'  # Your S3 bucket name
    photo = 'Dog Image.jpg'         # Your image file name

    # Detect labels
    labels = detect_labels(photo, bucket)

    # Filter labels with high confidence
```

```
    high_confidence_labels = [label for label in labels if
label['Confidence'] > 80]  # Adjust confidence threshold as needed

    # Print detected labels with high confidence
    print("Detected Labels with High Confidence:")
    for label in high_confidence_labels:
        print(f"Label: {label['Name']}, Confidence:
{label['Confidence']}")

    # Show bounding boxes for high confidence labels
    show_bounding_boxes(photo, bucket, high_confidence_labels)

if __name__ == "__main__":
    main()
```

1. **Save and Exit**:
    - Press Ctrl + X, then Y to save and exit nano.

Step 5: Run the Script

1. **Execute the Script**:
    - Run the script by executing:

      python3.8 label_generator.py

2. **View the Output**:
    - The output should show labels with confidence values. A window should display the image with bounding boxes drawn around detected objects.

6. Conclusion

This project demonstrates a practical application of AWS Rekognition for automated image labeling and object detection. The solution uses serverless and managed services on AWS, allowing easy scalability and seamless integration. By deploying this project on GitHub, others can learn from and build on it, showcasing AWS Rekognition's potential for real-world applications.