Smart Pet Feeder with Facial Recognition Control
Configuration

1. Raspberry Pi Configuration

- **Raspberry Pi Model**: Ensure Raspberry Pi has the appropriate specifications (*Raspberry Pi 4 Model B)* to run both the IoT-related services and machine learning components. Add in the necessary accessories such as:
    - Camera module (*Arducam 8MP*)
    - Servo motor for dispensing food (*SG90 Servo motor*)
- **Operating System**: Ensure you are using a compatible operating system : *(Raspberry Pi OS (Legacy) with desktop and recommended software: Debian version: 11 (bullseye))*
- **Network Configuration**:
    - Connect the Raspberry Pi to a stable Wi-Fi network for communication with AWS IoT Core and the S3 bucket.
    - Ensure the Raspberry Pi's network settings are configured for access to AWS IoT Core.
- **Hardware Setup**:
    - **Camera Module**: Set up the camera module and test it by capturing still images using commands such as libcamera-jpeg or any other camera library compatible with Raspberry Pi OS.
    - **Servo Motor**: Configure the GPIO pins for controlling the servo motor, which will dispense food based on certain conditions.

2. AWS Access Configuration

- **Environment Variables**: Store your AWS credentials securely.

  export AWS_ACCESS_KEY_ID="Your_Access_Key"
  export AWS_SECRET_ACCESS_KEY="Your_Secret_Key"
  export AWS_REGION="us-east-1"

- **IAM Role and Permissions**: The IAM role (for both your Lambda function and the Raspberry Pi) should have appropriate permissions for accessing AWS services like IoT Core, S3, and DynamoDB. Attach the following permissions to the role:

  iot:Connect, iot:Publish, iot:Subscribe, iot:Receive for IoT Core.
  s3:PutObject and s3:GetObject for S3 access.(or AmazonS3FullAccess)
  dynamodb:PutItem and dynamodb:GetItem for writing and retrieving pet feeding data from DynamoDB. (or AmazonDynamoDBFullAccess)

3. AWS IoT Core MQTT Configuration

- **Certificates**: Ensure your device has the correct certificates downloaded from AWS IoT Core (device certificate, private key, and root CA certificate) for secure communication.
- **MQTT Topics**: Set up MQTT topics in AWS IoT Core for communication between your device and cloud services.
    - pet/dispenser/image: For uploading images.
    - pet/dispenser/command: For receiving commands (to feed a pet or to query feeding status).
- **AWS IoT Rules**: Set up IoT rules to process incoming messages. Create a rule to invoke an AWS Lambda function whenever a new message is published to pet/dispenser/image topic.

4. S3 Bucket Configuration for Image Storage

- **Bucket Policy**: Configure the S3 bucket policy to control who can upload files to your S3 bucket.

5. DynamoDB Table Configuration

- **Table Structure**: Ensure that the DynamoDB table structure supports the necessary attributes to track pet feeding status. Table will have the following:
    - PetID (Primary Key): Name or ID of the pet.
    - LastFedTime: Timestamp of the last feeding.

6. AWS Lambda Function Configuration

- **Lambda Execution Role**: Attach the IAM role created with permissions to access S3, DynamoDB, and IoT Core.
- **Function Code**: Write the Lambda function to process IoT Core messages, interact with DynamoDB, and take actions. (Preprocess images, Upload image, Update feeding time and send last fed information)
- **Triggers**: Set up IoT Core rules to trigger this Lambda function whenever messages are received on topics like pet/dispenser/image.

7. Logging and Monitoring Configuration

- **CloudWatch Logs**: Enable logging for Lambda functions, IoT Core messages, and DynamoDB operations to monitor your application and troubleshoot errors.
- **CloudWatch Alarms**: Set up alarms for specific thresholds in your system (e.g., high number of failed image uploads or system downtime).