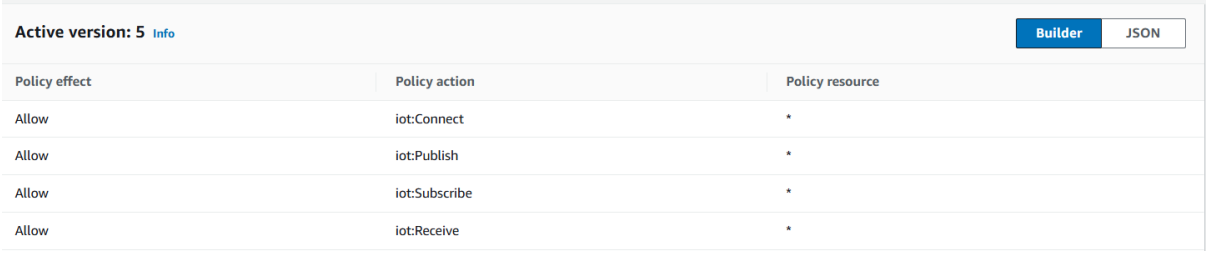
**Smart Pet Feeder with Facial Recognition Control**
**Cloud Configuration**

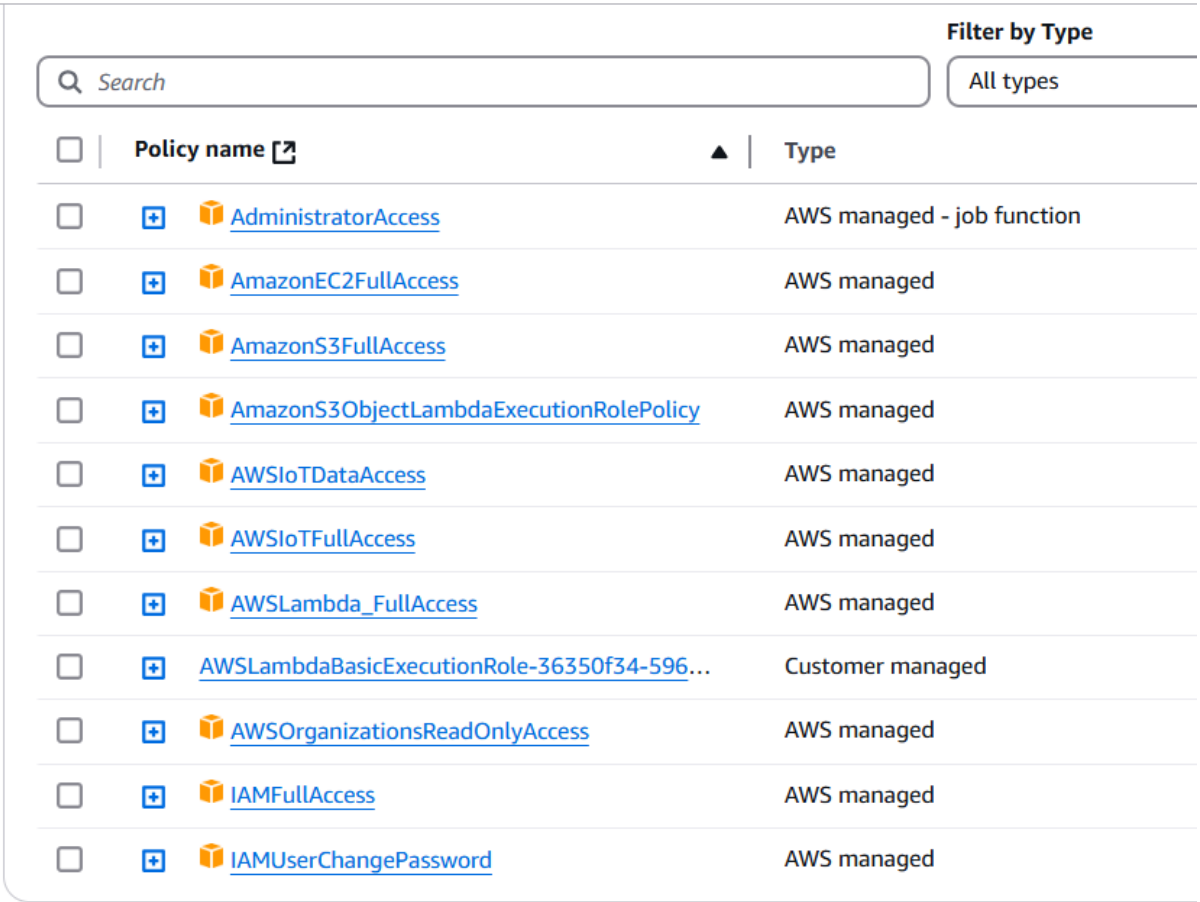
1. **Configure AWS IoT Core:**
    - Create an IoT Thing
        - Log in to the AWS Management Console and navigate to AWS IoT Core.
        - Create a new IoT Thing and download the device certificates (root CA, private key, and certificate).
        - Copy the root CA, private key, and certificate files to the Raspberry Pi.
    - Set Up a Policy: Attach an IoT policy to allow device communication.



Active version: 5 Info — Builder | JSON

| Policy effect | Policy action | Policy resource |
|---|---|---|
| Allow | iot:Connect | * |
| Allow | iot:Publish | * |
| Allow | iot:Subscribe | * |
| Allow | iot:Receive | * |

- Policy for the IAM user as follows: (Added throughout the course of the project)



Filter by Type: All types

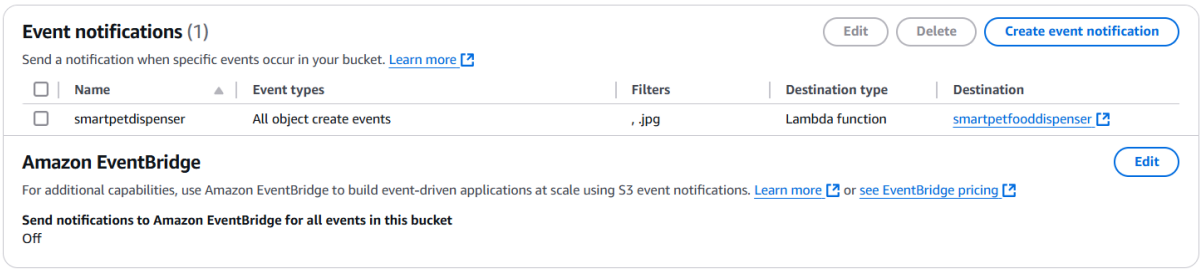
Search

| Policy name | Type |
|---|---|
| AdministratorAccess | AWS managed - job function |
| AmazonEC2FullAccess | AWS managed |
| AmazonS3FullAccess | AWS managed |
| AmazonS3ObjectLambdaExecutionRolePolicy | AWS managed |
| AWSIoTDataAccess | AWS managed |
| AWSIoTFullAccess | AWS managed |
| AWSLambda_FullAccess | AWS managed |
| AWSLambdaBasicExecutionRole-36350f34-596... | Customer managed |
| AWSOrganizationsReadOnlyAccess | AWS managed |
| IAMFullAccess | AWS managed |
| IAMUserChangePassword | AWS managed |

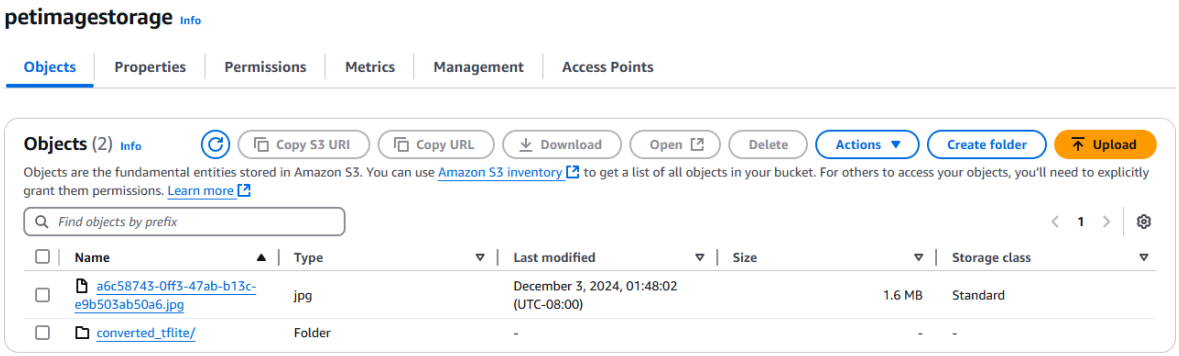
2. **Configure S3 buckets:**
    - **Create a Bucket**

- In the AWS Management Console, go to S3 and create a bucket (*petimagestorage* in this case) in the same region as IoT Core setup.
- Attach the **AmazonS3FullAccess** policy

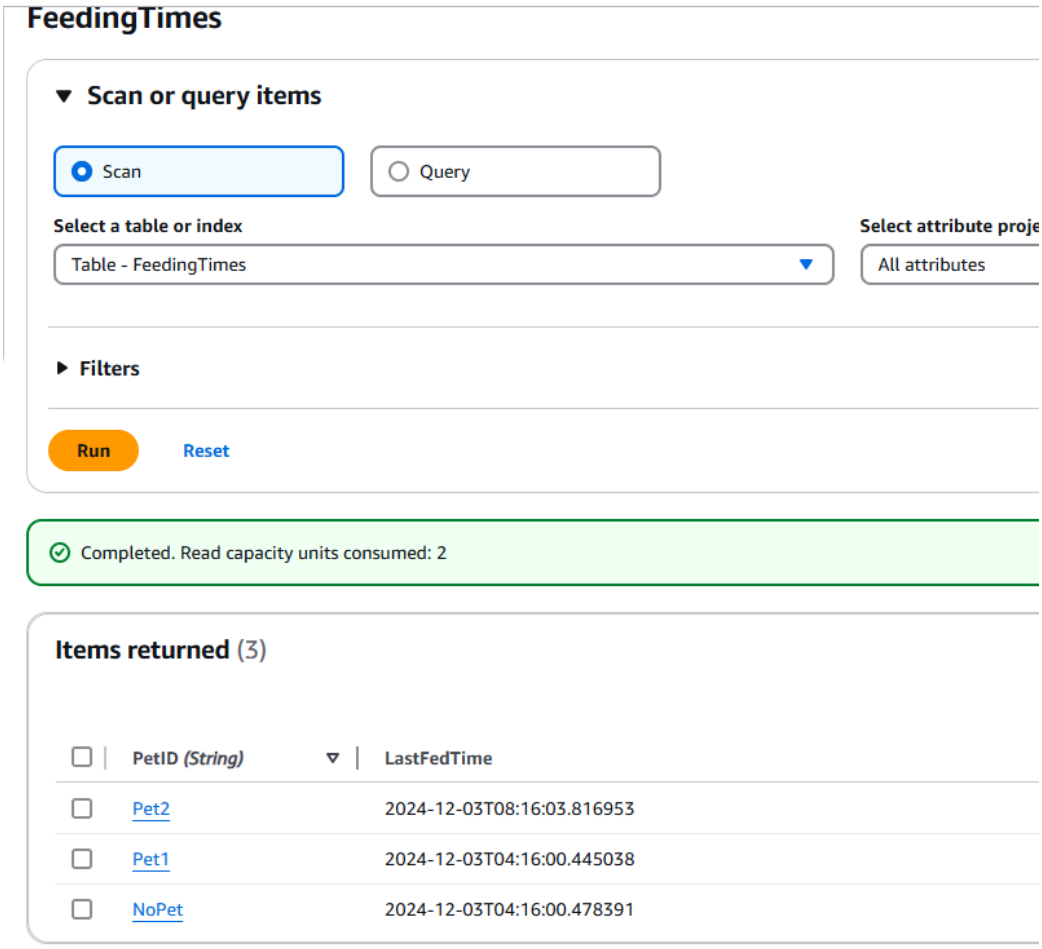
- **Event notifications**
  - Send a notification when specific events occur in the S3 bucket to the Lambda function



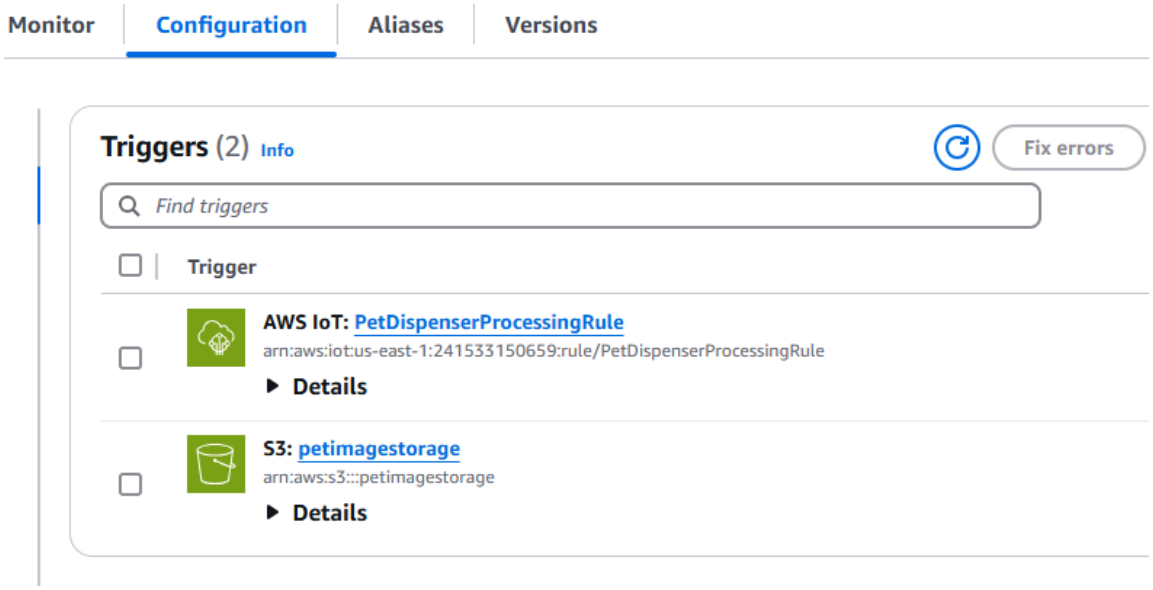

- **Upload machine learning model**
  - Upload the model into S3 for ease of access from Lambda function



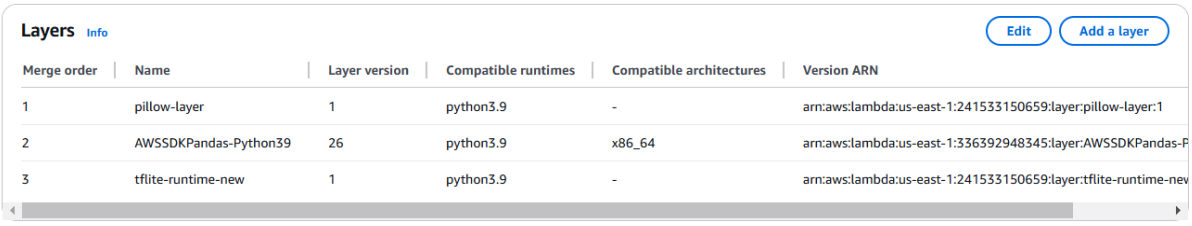

3. **Configure AWS DynamoDB**
   - **Create DynamoDB Table**
     - Go to Amazon DynamoDB in the AWS Console. Create Table. Set a table name (*FeedingTimes in this case*).
     - Set the primary key as PetID (String) and optionally add LastFedTime (Number) to track the feeding times. LastFedTime will be initialized in the table anyway.
     - Attach **AmazonDynamoDBFullAccess** to the IAM policy or you could also attach the policy to the lambda role.



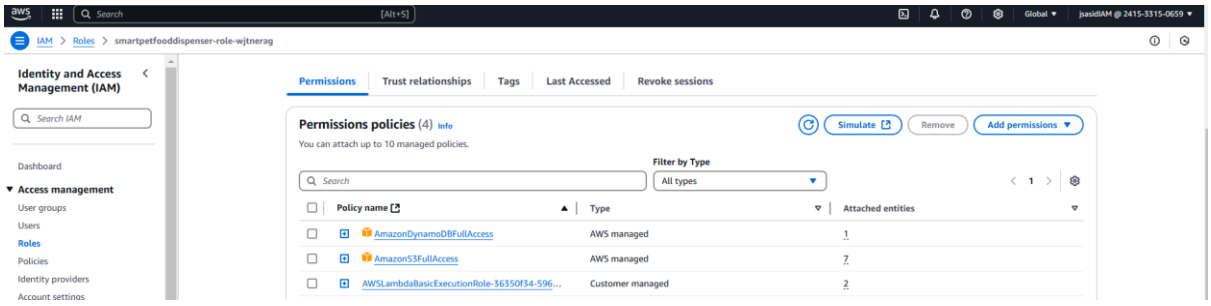

4. **Configire AWS Lambda**
    - Navigate to AWS Lambda in the console and create a new Lambda function (*smartpetfooddispenser* in this case). And write the lambda_handler code.
    - Trigger Lambda from IoT Core: Set up an IoT Rule in AWS IoT Core to trigger the Lambda function based on a message received in the IoT topic.
    - An S3 trigger is also added when setting the event notification from the previous step.



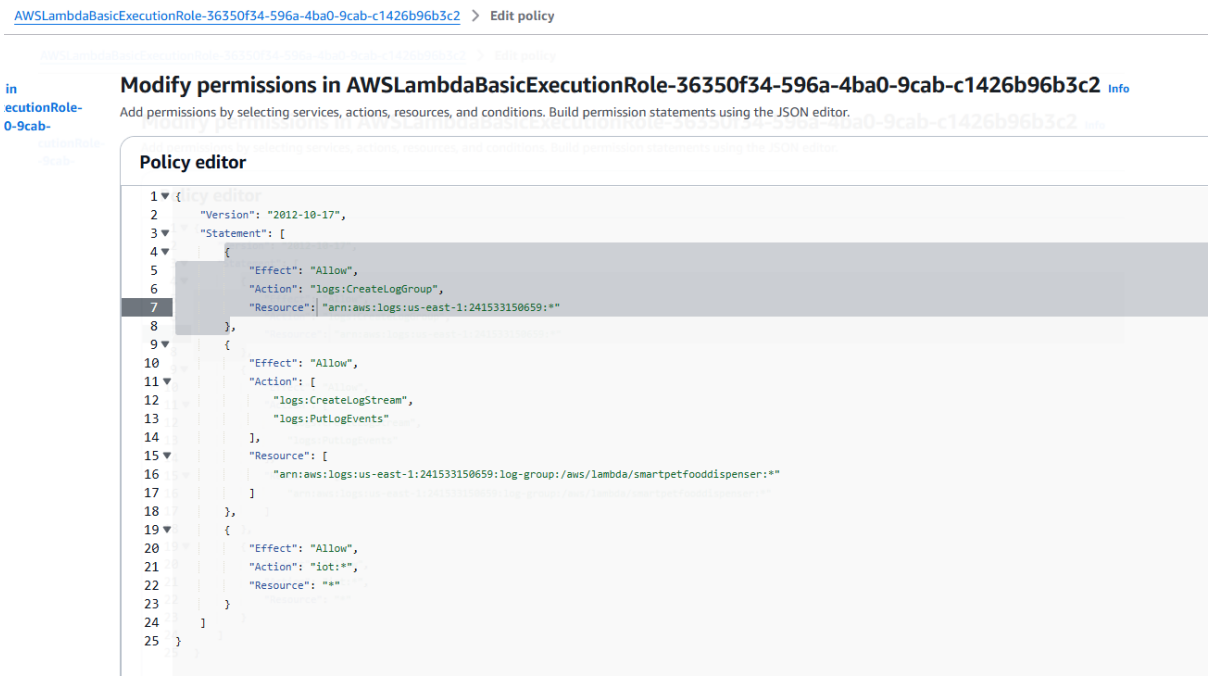

- Add the following layers to resolve dependencies issues.
  - AWSSDKPandas-Python39
  - Add layer → AWS layer → AWSSDKPandas-Python39
    - Note: Make sure the python version is Python 3.9
  - pillow-layer
  - tflite-runtime
    - Create a layer from AWS CLI and attach the layer to the lambda function.




- Set an appropriate execution role for the Lambda function that grants permissions to interact with DynamoDB, S3, and IoT Core.




- Add the following permissions to the Lambda role




5. **Configure AWS CloudWatch Monitoring**
    - Set up CloudWatch to monitor logs for the Lambda functions, MQTT messages, or device status.

**Overview of the Cloud Configuration**

1. IoT Core: Device securely connects using certificates and MQTT, sending data to topics like pet/dispenser/image.
2. S3: Device uploads images captured by the Raspberry Pi camera to S3.
3. DynamoDB: Stores the recognition data, i.e. Pet ID and last fed time
4. Lambda: Processes data from IoT Core and updates DynamoDB and publish messages back to Raspberry pi device
5. IAM Roles and Policies: Secure permissions for IoT Core, S3, DynamoDB, and Lambda.