

Тема: Архитектура за класификация и съхранение на снимки, посредством ползване на услугите Amazon Rekognition, Amazon Lambda и Amazon CloudWatch

Предмет: Приложно-програмни интерфейси за работа с облачни архитектури с Амазон Уеб Услуги (AWS)

Изготвил: Венета Кирева, фн: 82184, имейл: fn82184@g.fmi.uni-sofia.bg

Лектор: проф. д-р Милен Петров, година: 2022/2023

Съдържание

1	Условие	3
2	Въведение	3
3	Теория	4
4	Използвани технологии	6
5	Инсталация и настройки	7
5.1	Създаване на Amazon Simple Storage Service (S3) Bucket	7
5.2	Създаване на Amazon Simple Queue Service (SQS) опашка	9
5.3	Създаване на Amazon DynamoDB таблица	10
5.4	Създаване на Amazon Simple Notification Service (SNS) Topic и Subscription	12
5.5	Създаване на роли чрез AWS Identity and Access Management (IAM)	15
5.6	Създаване на Amazon Lambda Function	22
5.7	Създаване на Amazon CloudWatch Rule	25
6	Кратко ръководство за потребителя	27
7	Примерни данни	29

8	Описание на програмния код	30
8.1	Обработка на получено изображение	30
8.2	Експортиране на Amazon DynamoDB таблица в CSV формат . . .	35
8.3	Уведомяване на потребителите за налични CSV файлове	37
9	Приноси на студента, ограничения и възможности за бъдещо развитие	39
10	Какво научих	40
11	Списък с фигури и таблици	41
12	Използвани източници	42

1 Условие

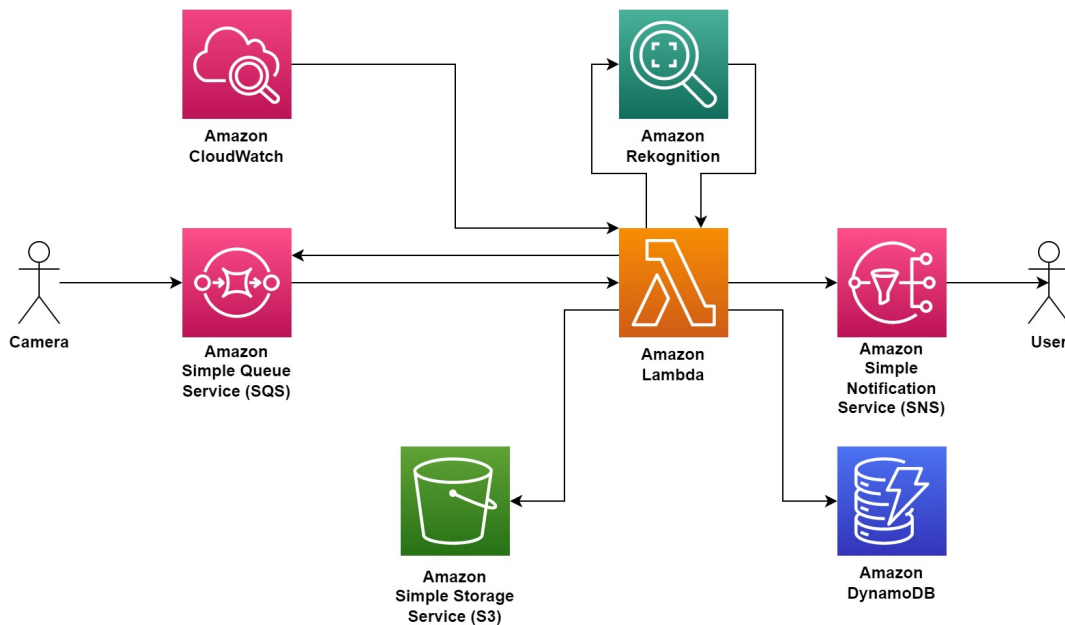
Създайте функционалност или архитектура, използвайки услугите на Amazon Web Services.

2 Въведение

Настоящият проект представя изграждането на архитектура в AWS за класификация и съхранение на изображения. За реализацията са използвани седем от услугите, предоставени от Amazon Web Services, като необходимите функционалности са в рамките на AWS Free Tier.

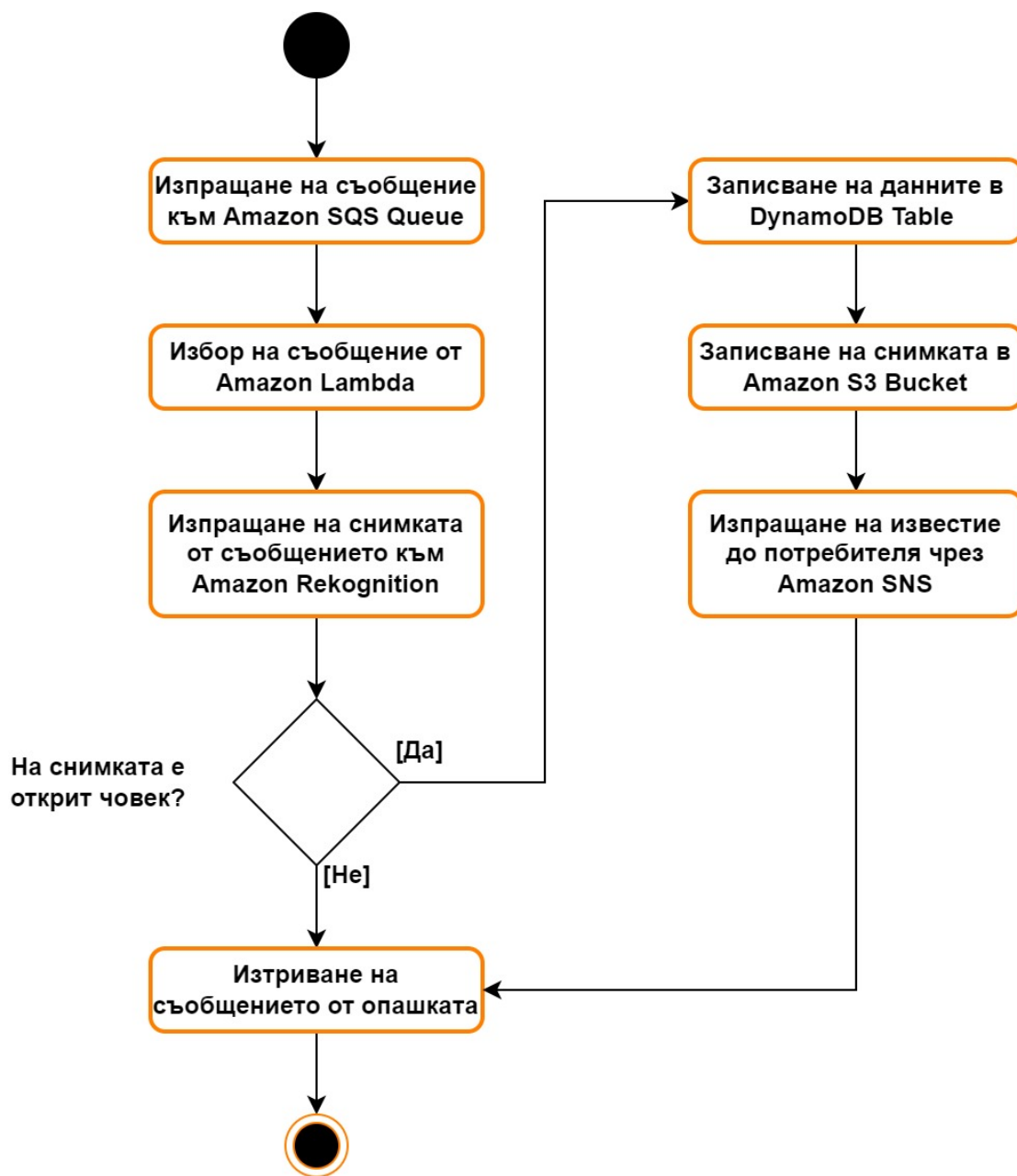
3 Теория

Изображенията биват получени в 64-битов формат чрез Amazon SQS Queue. Чрез график, зададен в Amazon CloudWatch Event, през определени интервали от време се извиква Amazon Lambda Function, която декодира снимката и извиква Amazon Rekognition, за да определи дали в нея е открит човек. Ако е открит, резултатот от изпълнението на Amazon Rekognition в Amazon DynamoDB Table, а снимката запазва в Amazon S3 Bucket. Изпраща връзка за достъп до снимката, валидна 24 часа, до имейл адресите на абонираните потребители.



Фигура 1: Общ преглед на връзките между услугите. [Diagrams.net]

Веднъж на 24 часа информацията от Amazon DynamoDB Table се експортира в CSV файл и се запазва в Amazon S3 Bucket. На всеки 7 дни се изпраща имейл до абонираните потребители с напомняне за последния генериран CSV файл и връзка за достъп към него.



Фигура 2: Диаграма, показваща действията за обработване на получена снимка.
[Diagrams.net]

4 Използвани технологии

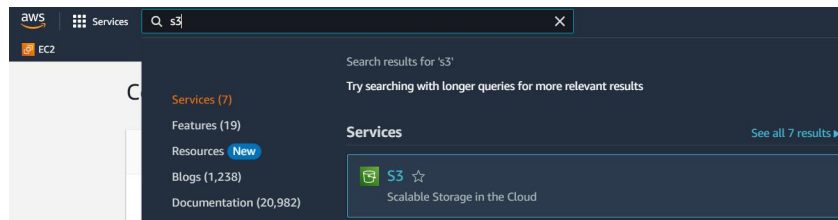
- **Amazon CloudWatch** - услуга, събираща и обработваща данни за състоянието на системата.
- **Amazon DynamoDB** - безсервърна NoSQL база от данни.
- **Amazon Lambda** - безсервърна изчислителна услуга.
- **Amazon Rekognition** - SaaS платформа, осигуряваща инструменти за използването на компютърно зрение за анализ на изображения и видео.
- **Amazon Simple Notification Service (SNS)** - уеб услуга, управляваща изпращането и получаването на съобщения до абонирани клиенти или точки на обслужване.
- **Amazon Simple Queue Service (SQS)** - услуга за опашка от съобщения.
- **Amazon Simple Storage Service (S3)** - уеб услуга, осигуряваща съхранение на обекти.

5 Инсталация и настройки

Преди да започнете с настройките на услугите е необходимо да сте влезли в AWS Free Tier акаунт.

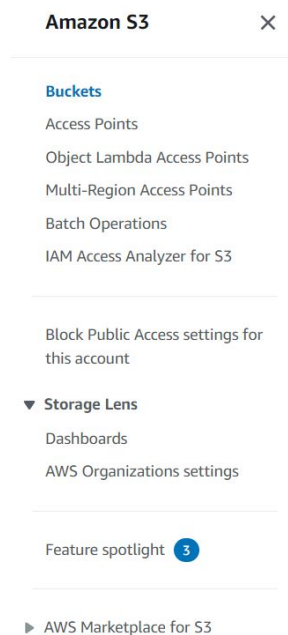
5.1 Създаване на Amazon Simple Storage Service (S3) Bucket

Стъпка 1. Напишете в търсачката "S3" и изберете *S3* от падащото меню с услуги.



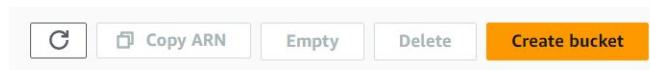
Фигура 3: Създаване на Amazon S3 Bucket - стъпка 1.

Стъпка 2. От менюто изберете *Buckets*.



Фигура 4: Създаване на Amazon S3 Bucket - стъпка 2.

Стъпка 3. Изберете бутона *Create bucket*.



Фигура 5: Създаване на Amazon S3 Bucket - стъпка 3.

Стъпка 4. Попълнете в полето Bucket name име. За целите на текущото ръководство ще използваме **main-bucket**.

The screenshot shows the 'Create bucket' page in the AWS Management Console. At the top, it says 'Create bucket' with an 'Info' link. Below that, a note states 'Buckets are containers for data stored in S3. Learn more' with a link. The main section is titled 'General configuration'. It contains a 'Bucket name' field with 'main-bucket' entered. Below the field is a note: 'Bucket name must be globally unique and must not contain spaces or uppercase letters. See rules for bucket naming' with a link. Below that is an 'AWS Region' dropdown menu currently set to 'US East (N. Virginia) us-east-1'. At the bottom, there is a section for 'Copy settings from existing bucket - optional' with a note 'Only the bucket settings in the following configuration are copied.' and a 'Choose bucket' button.

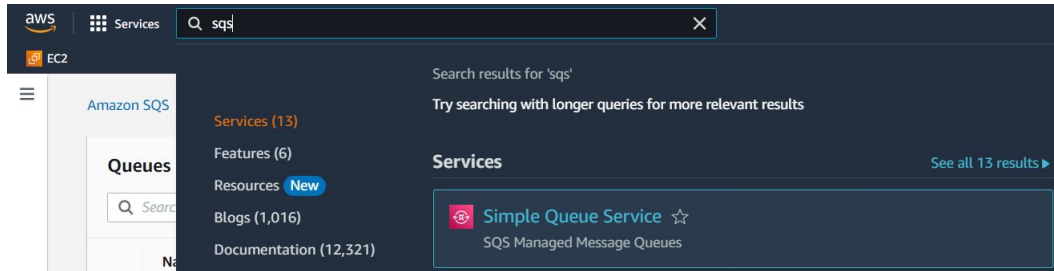
Фигура 6: Създаване на Amazon S3 Bucket - стъпка 4.

Стъпка 5. Оставете настройките по подразбиране на останалите опции и изберете бутона *Create bucket*.

Стъпка 6. Повторете отново стъпки 3. до 5. За целите на текущото ръководство за този AWS S3 Bucket ще използваме името **backup-bucket**.

5.2 Създаване на Amazon Simple Queue Service (SQS) опашка

Стъпка 1. Напишете в търсачката "SQS" и изберете *Simple Queue Service* от падащото меню с услуги.



Фигура 7: Създаване на Amazon SQS опашка - стъпка 1.

Стъпка 2. Изберете бутона *Create queue*.



Фигура 8: Създаване на Amazon SQS опашка - стъпка 2.

Стъпка 3. Попълнете в полето Name име. За целите на текущото ръководство ще използваме **queue**.

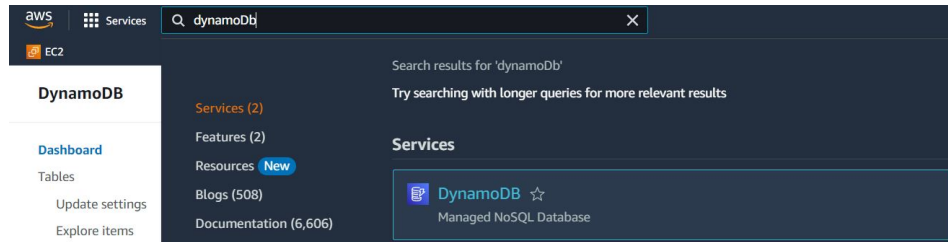
A screenshot of the 'Create queue' form in the AWS console. The form has a 'Details' section with two radio button options: 'Standard info' (selected) and 'FIFO info'. The 'Standard info' option includes sub-points: 'At-least-once delivery', 'At-least once delivery', and 'Best-effort ordering'. The 'FIFO info' option includes sub-points: 'First-in-first-out delivery, message ordering is preserved', 'First-in-first-out delivery', and 'Exactly-once processing'. Below these options is a warning message: 'You can't change the queue type after you create a queue.' At the bottom, there is a 'Name' field with the text 'MyQueue' entered. A note below the field states: 'A queue name is case-sensitive and can have up to 80 characters. You can use alphanumeric characters, hyphens (-), and underscores (_).'

Фигура 9: Създаване на Amazon SQS опашка - стъпка 3.

Стъпка 4. Оставете настройките по подразбиране на останалите опции и изберете бутона *Create queue*.

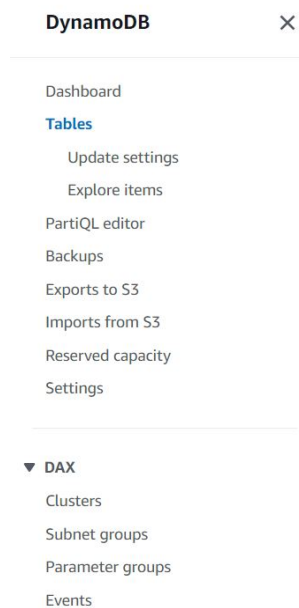
5.3 Създаване на Amazon DynamoDB таблица

Стъпка 1. Напишете в търсачката "DynamoDB" и изберете *DynamoDB* от падащото меню с услуги.



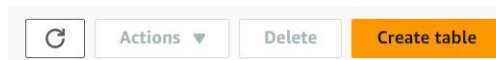
Фигура 10: Създаване на Amazon DynamoDB таблица - стъпка 1.

Стъпка 2. От менюто изберете *Tables*.



Фигура 11: Създаване на Amazon DynamoDB таблица - стъпка 2.

Стъпка 3. Изберете бутона *Create table*.



Фигура 12: Създаване на Amazon DynamoDB таблица - стъпка 3.

Стъпка 4. Попълнете в полето Name име. За целите на текущото ръководство ще използваме **table**. В полето Partition key въведете "picID".

DynamoDB > Tables > Create table

Create table

Table details [Info](#)

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name
This will be used to identify your table.

Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.).

Partition key
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

1 to 255 characters and case sensitive.

Sort key - optional
You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

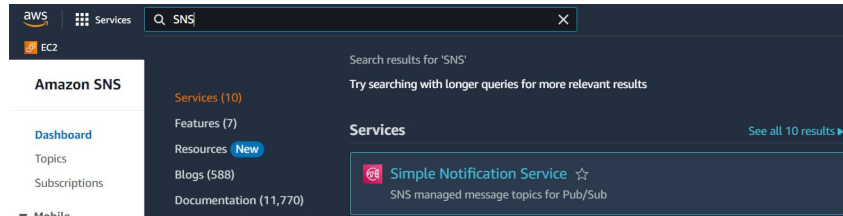
1 to 255 characters and case sensitive.

Фигура 13: Създаване на Amazon DynamoDB таблица - стъпка 4.

Стъпка 5. Оставете настройките по подразбиране на останалите опции и изберете бутона *Create table*.

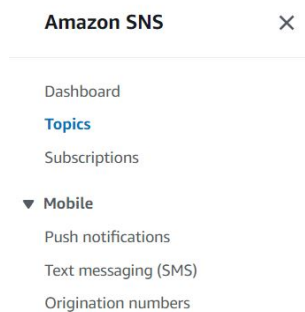
5.4 Създаване на Amazon Simple Notification Service (SNS) Topic и Subscription

Стъпка 1. Напишете в търсачката "SNS" и изберете *Simple Notification Service* от падащото меню с услуги.



Фигура 14: Създаване на Amazon SNS Topic - стъпка 1.

Стъпка 2. От менюто изберете *Topics*.



Фигура 15: Създаване на Amazon SNS Topic - стъпка 2.

Стъпка 3. Изберете бутона *Create topic*.



Фигура 16: Създаване на Amazon SNS Topic - стъпка 3.

Стъпка 4. В подменюто Type изберете Standard. Попълнете в полето Name име. За целите на текущото ръководство ще използваме **topic**.

Create topic

Details

Type [Info](#)
Topic type cannot be modified after topic is created

☐ FIFO (first-in, first-out)

- Strictly-preserved message ordering
- Exactly-once message delivery
- High throughput, up to 300 publishes/second
- Subscription protocols: SQS

☒ Standard

- Best-effort message ordering
- At-least once message delivery
- Highest throughput in publishes/second
- Subscription protocols: SQS, Lambda, HTTP, SMS, email, mobile application endpoints

Name

Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).

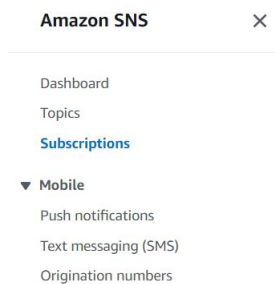
Display name - optional [Info](#)
To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message.

Maximum 100 characters.

Фигура 17: Създаване на Amazon SNS Topic - стъпка 4.

Стъпка 5. Оставете настройките по подразбиране на останалите опции и изберете бутона *Create topic*.

Стъпка 6. От менюто изберете *Subscriptions*.



Фигура 18: Създаване на Amazon SNS Topic - стъпка 6.

Стъпка 7. Изберете бутона *Create subscription*.



Фигура 19: Създаване на Amazon SNS Topic - стъпка 7.

Стъпка 8. В Topic ARN изберете ARN на създадения Amazon SNS Topic. В Protocol изберете Email. В полето Endpoint въведете имейл адреса, който да получава известията.

Create subscription

Details

Topic ARN
🔍 MyTopic

Protocol
The type of endpoint to subscribe
Email ▼

Endpoint
An email address that can receive notifications from Amazon SNS.
test@example.com

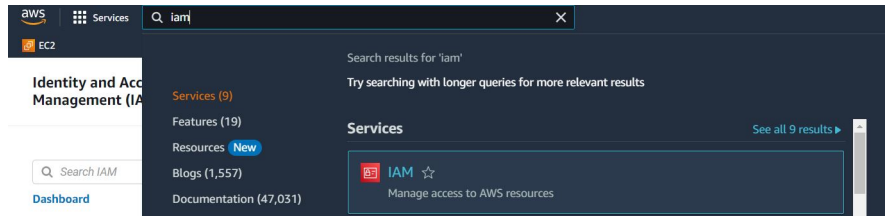
ⓘ After your subscription is created, you must confirm it. [Info](#)

Фигура 20: Създаване на Amazon SNS Topic - стъпка 8.

Стъпка 9. Оставете настройките по подразбиране на останалите опции и изберете бутона *Create subscription*.

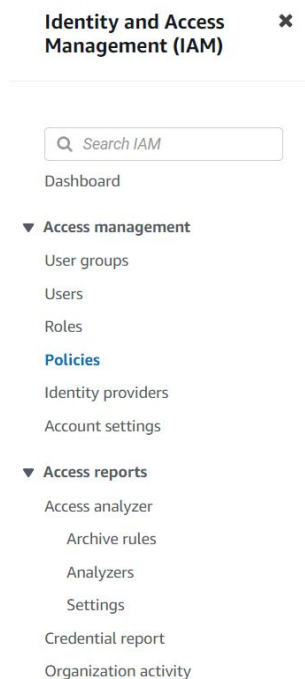
5.5 Създаване на роли чрез AWS Identity and Access Management (IAM)

Стъпка 1. От менюто изберете *Access management > Policies*.



Фигура 21: Създаване на роли чрез AWS IAM - стъпка 1.

Стъпка 2. От менюто изберете *Access management > Policies*.

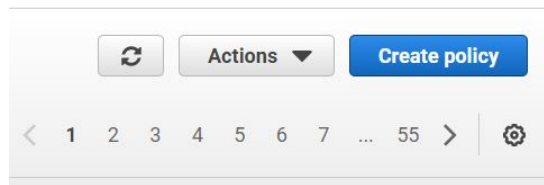


Фигура 22: Създаване на роли чрез AWS IAM - стъпка 2.

Стъпка 3. Изберете бутона *Create policy*.

Стъпка 4. Изберете бутона *JSON*.

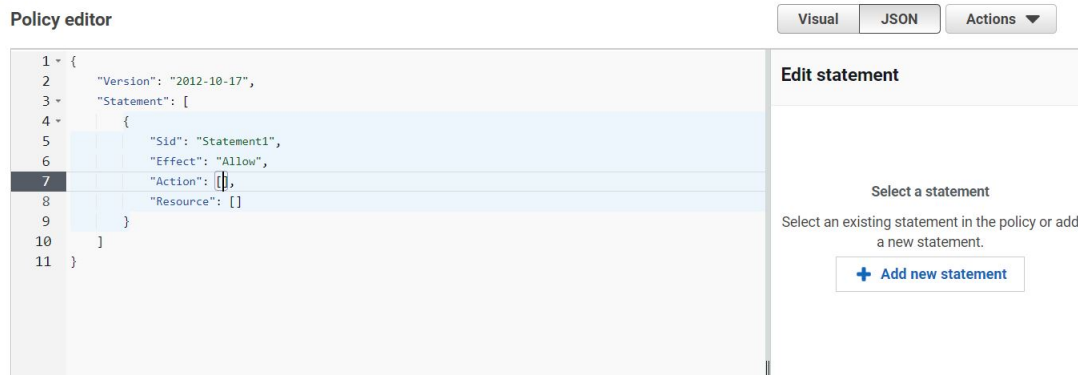
Стъпка 5. Заменете съдържанието на Policy editor полето със следния текст, като на местата, означени с коментар, заместите с необходимите данни на създадените в предните стъпки услуги.



Фигура 23: Създаване на роли чрез AWS IAM - стъпка 3.

Specify permissions [Info](#)

Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.



Фигура 24: Създаване на роли чрез AWS IAM - стъпка 4.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "sqs:DeleteMessage",
        "sqs:ListQueues",
        "sqs:ReceiveMessage"
      ],
      "Effect": "Allow",
      "Resource": "" # ARN на Amazon SQS Queue
    },
    {
      "Action": [
        "sns:Subscribe",
        "sns:Publish"
      ],
```



```

    "Effect": "Allow",
    "Resource": "" # ARN на Amazon SNS Topic
  },
  {
    "Action": [
      "s3:GetObject",
      "s3:PutObject",
      "s3:ListBucket",
      "s3:PutObjectAcl"
    ],
    "Effect": "Allow",
    "Resource": [
      "", # ARN на основния Amazon S3 Bucket
      "" # ARN на основния Amazon S3 Bucket + "/*"
    ]
  },
  {
    "Action": [
      "dynamodb:CreateTable",
      "dynamodb:PutItem",
      "dynamodb:Update*"
    ],
    "Effect": "Allow",
    "Resource": "" # ARN на Amazon DynamoDB Table
  },
  {
    "Action": [
      "rekognition:DetectLabels"
    ],
    "Effect": "Allow",
    "Resource": "*"
  },
  {
    "Action": [
      "logs:PutLogEvents"
    ],
    "Effect": "Allow",
    "Resource": "*"
  },
  {
    "Effect": "Allow",

```

```

        "Action": "logs:CreateLogGroup",
        "Resource": "" # "arn:aws:logs:" + AWS регион + ":" + Account ID + ":*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "logs:CreateLogStream",
            "logs:PutLogEvents"
        ],
        "Resource": [
            "*"
        ]
    }
]
}

```

Стъпка 6. Изберете бутона *Next*.

Стъпка 7. Попълнете в полето Policy име. За целите на текущото ръководство ще използваме **main-policy**.

Review and create

Review the permissions, specify details, and tags.

Policy details

Policy name

Enter a meaningful name to identify this policy.

Maximum 128 characters. Use alphanumeric and '+', '@', '-' characters.

Description - optional

Add a short explanation for this policy.

Maximum 1,000 characters. Use alphanumeric and '+', '@', '-' characters.

Фигура 25: Създаване на роли чрез AWS IAM - стъпка 7.

Стъпка 8. За да го запазите изберете бутона **Create policy**.

Стъпка 9. Повторете отново стъпки 3. до 7., като замените текста с приложения. За целите на текущото ръководство за следващата политика ще използваме името **dynamodb-s3-policy**.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "dynamodb:CreateTable",
        "dynamodb:PutItem",
        "dynamodb:Update*",
        "dynamodb:Scan"
      ],
      "Effect": "Allow",
      "Resource": "" # ARN на Amazon DynamoDB таблица
    },
    {
      "Action": [
        "s3:PutObject",
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": [
        "", # ARN на резервния Amazon S3 Bucket
        "" # ARN на резервния Amazon S3 Bucket + "/*"
      ]
    },
    {
      "Action": [
        "logs:PutLogEvents"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "logs:CreateLogGroup",
      "Resource": "" # "arn:aws:logs:" + AWS регион + ":" + Account ID + ":*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ]
    }
  ]
}

```

```

    ],
    "Resource": [
        "*"
    ]
}
]
}

```

Стъпка 10. Повторете отново стъпки 3. до 7., като замените текста с приложения. За целите на текущото ръководство за следващата политика ще използваме името **dynamodb-sns-policy**.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:ListBucket",
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": [
        "", # ARN на резервния Amazon S3 Bucket
        "" # ARN на резервния Amazon S3 Bucket + "/"
      ]
    },
    {
      "Action": [
        "sns:Subscribe",
        "sns:Publish"
      ],
      "Effect": "Allow",
      "Resource": # ARN на Amazon SNS Topic
    },
    {
      "Action": [
        "logs:PutLogEvents"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}

```

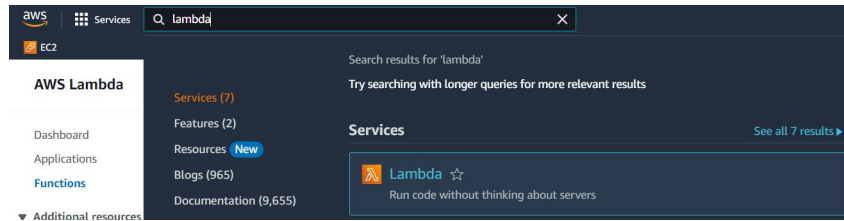
```

    },
    {
      "Effect": "Allow",
      "Action": "logs:CreateLogGroup",
      "Resource": "" # "arn:aws:logs:" + AWS регион + ":" + Account ID + ":*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}

```

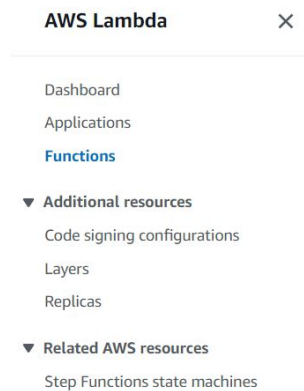
5.6 Създаване на Amazon Lambda Function

Стъпка 1. Напишете в търсачката "Lambda" и изберете *Lambda* от падащото меню с услуги.



Фигура 26: Създаване на Amazon Lambda Function - стъпка 1.

Стъпка 2. От менюто изберете *Functions*.



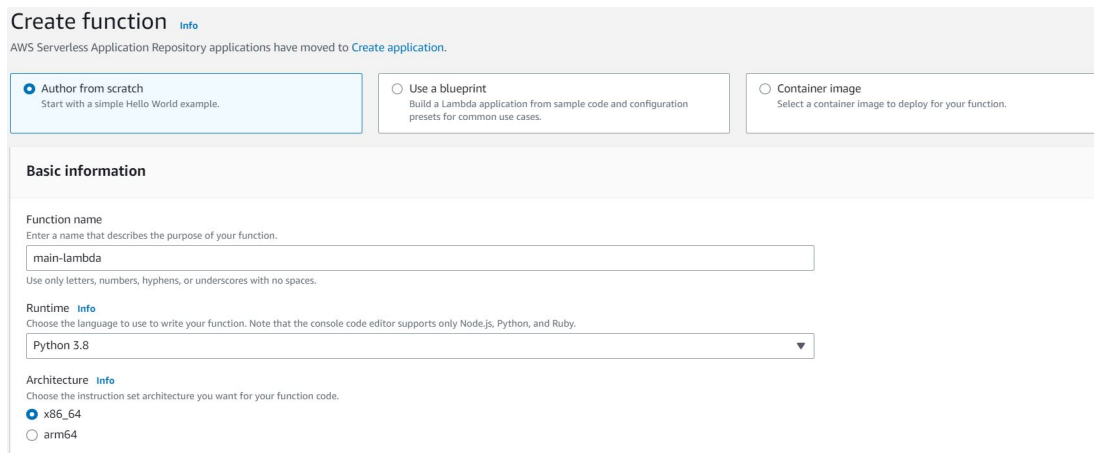
Фигура 27: Създаване на Amazon Lambda Function - стъпка 2.

Стъпка 3. Изберете бутона *Create function*.



Фигура 28: Създаване на Amazon Lambda Function - стъпка 3.

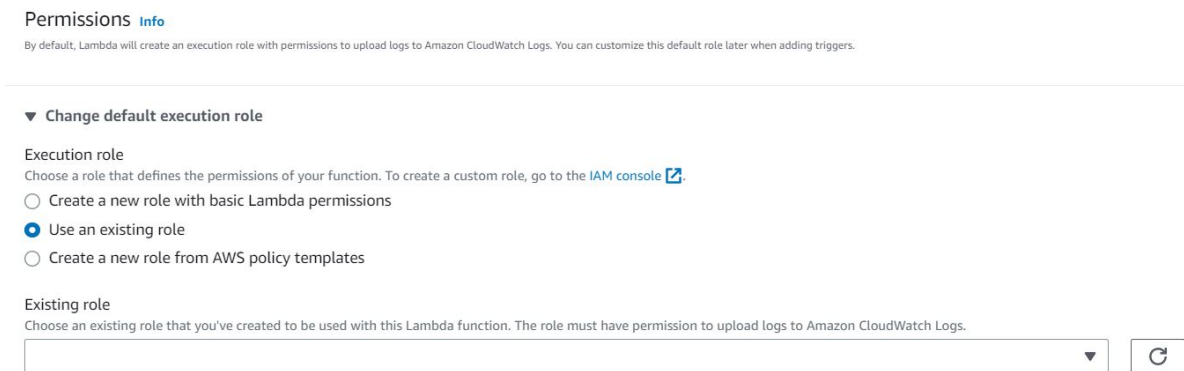
Стъпка 4. Изберете Author from scratch. Попълнете в полето Function name име. За целите на текущото ръководство ще използваме **main-lambda**. В Runtime изберете Python 3.8.



The screenshot shows the 'Create function' page in the AWS Lambda console. At the top, there are three tabs: 'Author from scratch' (selected), 'Use a blueprint', and 'Container image'. Below the tabs is the 'Basic information' section. It contains three fields: 'Function name' with the value 'main-lambda', 'Runtime' set to 'Python 3.8', and 'Architecture' set to 'x86_64'.

Фигура 29: Създаване на Amazon Lambda Function - стъпка 4.

Стъпка 5. Изберете Change default execution role. В Execution role изберете Use an existing role. От падащото меню Existing role изберете main-policy.



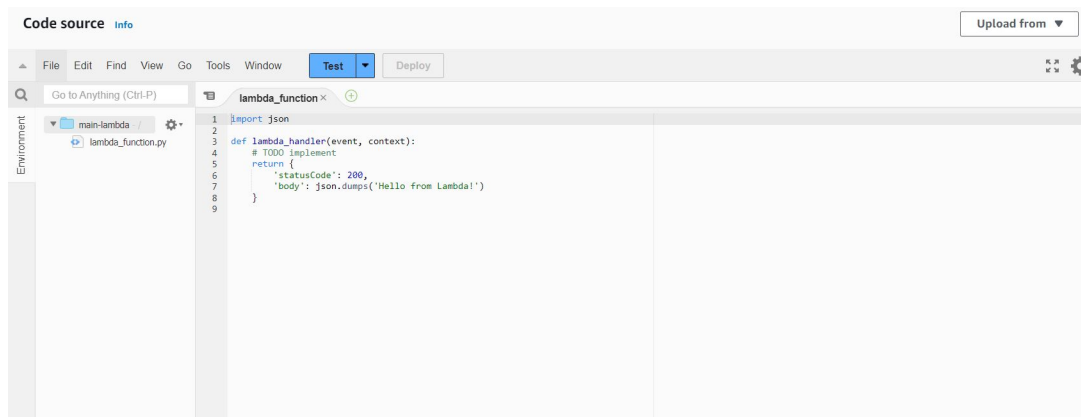
The screenshot shows the 'Permissions' section of the AWS Lambda console. It features a section titled 'Change default execution role'. Under 'Execution role', the option 'Use an existing role' is selected. Below this, there is a dropdown menu for 'Existing role' and a 'Create' button.

Фигура 30: Създаване на Amazon Lambda Function - стъпка 5.

Стъпка 6. Оставете настройките по подразбиране на останалите опции и изберете бутона *Create function*.

Стъпка 7. В полето Code source въведете кода от 8.1, като на указаните места в декларацията на променливи добавите данните за създадените ресурси. За Amazon S3 Bucket посочете main-bucket.

Стъпка 8. Изберете бутона *Deploy*.



Фигура 31: Създаване на Amazon Lambda Function - стъпка 7.



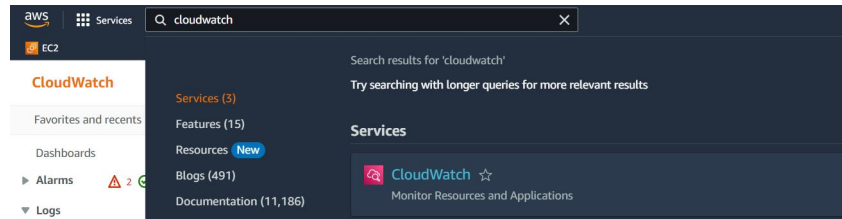
Фигура 32: Създаване на Amazon Lambda Function - стъпка 8.

Стъпка 9. Повторете отново стъпки 3. до 8., като използвате dynamodb-s3-policy. В полето Code source въведете кода от 8.2 с Amazon S3 Bucket backup-bucket. За целите на текущото ръководство ще зададем името **dynamodb-s3-lambda**.

Стъпка 10. Повторете отново стъпки 3. до 8., като използвате dynamodb-sns-policy. В полето Code source въведете кода от 8.3 с Amazon S3 Bucket backup-bucket. За целите на текущото ръководство ще зададем името **dynamodb-sns-lambda**.

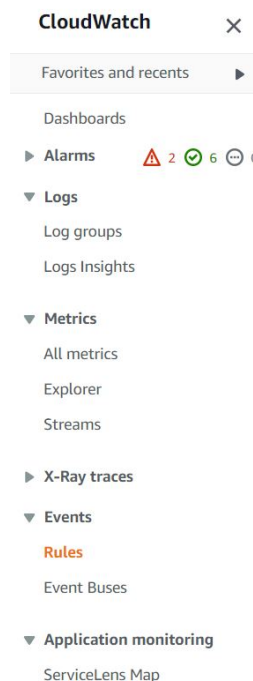
5.7 Създаване на Amazon CloudWatch Rule

Стъпка 1. Напишете в търсачката "CloudWatch" и изберете *CloudWatch* от падащото меню с услуги.



Фигура 33: Създаване на Amazon CloudWatch Rule - стъпка 1.

Стъпка 2. От менюто изберете *Events > Rules*.



Фигура 34: Създаване на Amazon CloudWatch Rule - стъпка 2.

Стъпка 3. Изберете бутона *Create rule*.



Фигура 35: Създаване на Amazon CloudWatch Rule - стъпка 3.

Стъпка 4. Изберете Schedule и в полето Fixed rate of въведете интервала от време, през който искате да се проверява опашката за получени съобщения.

Step 1: Create rule

Create rules to invoke Targets based on Events happening in your AWS environment.

Event Source

Build or customize an Event Pattern or set a Schedule to invoke Targets.

☐ Event Pattern  ☒ Schedule 

☒ Fixed rate of Minutes 

☐ Cron expression

[Learn more about CloudWatch Events schedules.](#)

► Show sample event(s)

Фигура 36: Създаване на Amazon CloudWatch Rule - стъпка 4.

Стъпка 5. В Targets изберете бутона Add target. От падащото меню изберете Lambda function, след което от Function посочете main-lambda.

Targets

Select Target to invoke when an event matches your Event Pattern or when schedule is triggered.

Lambda function  

Function* 

► Configure version/alias

► Configure input

 Add target*

Фигура 37: Създаване на Amazon CloudWatch Rule - стъпка 5.

Стъпка 6. Оставете настройките по подразбиране на останалите опции и изберете бутона *Configure details*.

Стъпка 7. В Name въведете име и изберете бутона *Create rule*.

Step 2: Configure rule details

Rule definition

Name*

Description

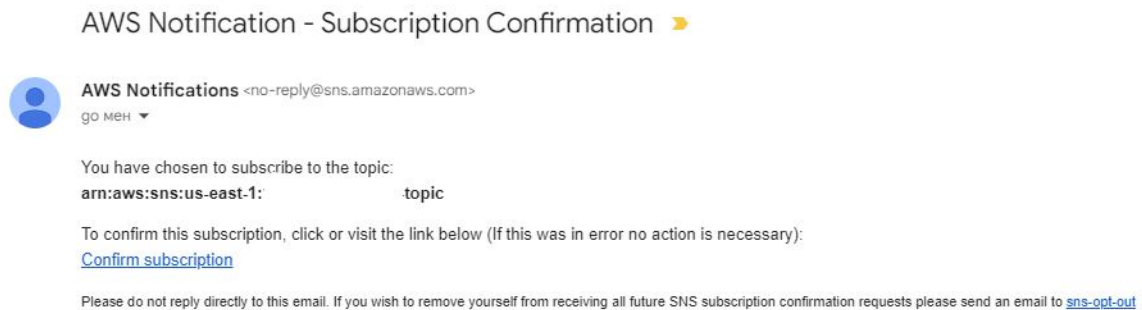
State ☒ Enabled

Фигура 38: Създаване на Amazon CloudWatch Rule - стъпка 7.

Стъпка 8. Повторете отново стъпки 3. до 7., като в Target посочвате всяка от останалите функции.

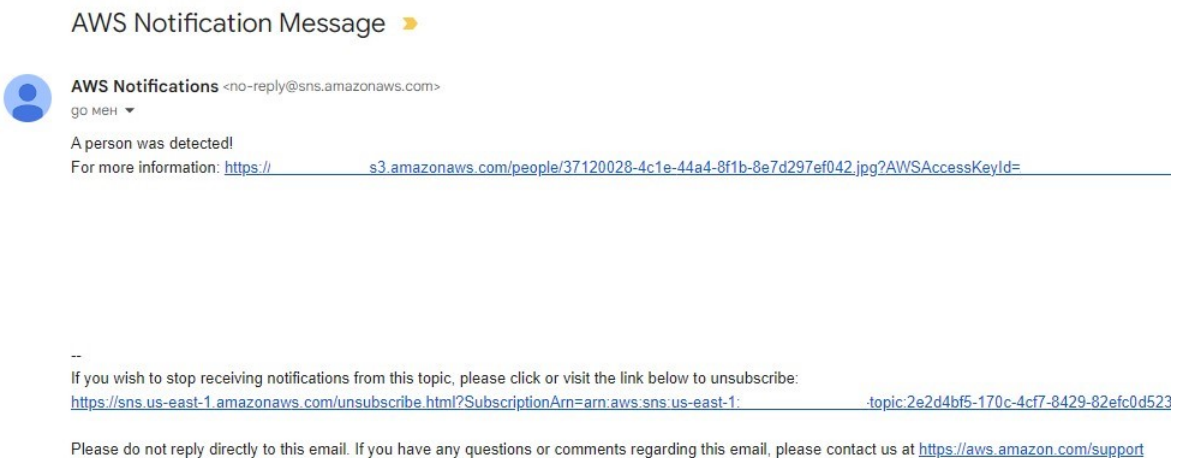
6 Кратко ръководство за потребителя

След добавянето на имейл адрес в Amazon SNS Subscription, до него ще бъде изпратено съобщение за потвърждение. За потвърждение е необходимо да се избере *Confirm subscription*.



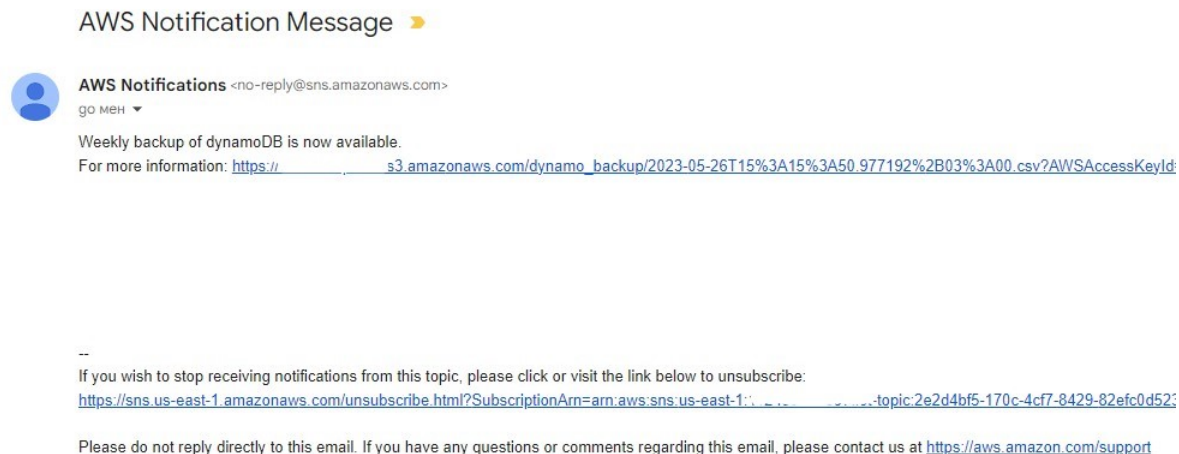
Фигура 39: Потвърждаване на абонамент към Amazon SNS Subscription.

След като абонаментът е потвърден, към този адрес ще започнат да се изпращат съобщения.

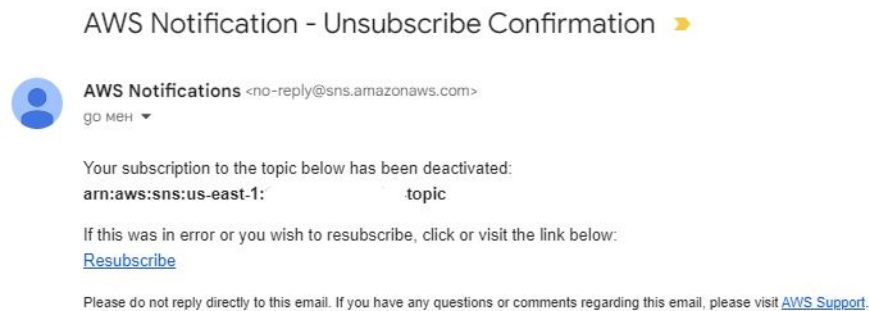


Фигура 40: Съобщение за откриване на човек на снимката.

За прекратяване на абонамента се използва връзката, предоставена в долната част на получените имейли, като при спиране бива изпратено писмо с потвърждение.



Фигура 41: Съобщение за готово седмично резервно копие на Amazon DynamoDB Table.



Фигура 42: Потвърждаване на прекратяването на абонамент към Amazon SNS Subscription.

7 Примерни данни

Демонстрация на работата на системата върху следното изображение:

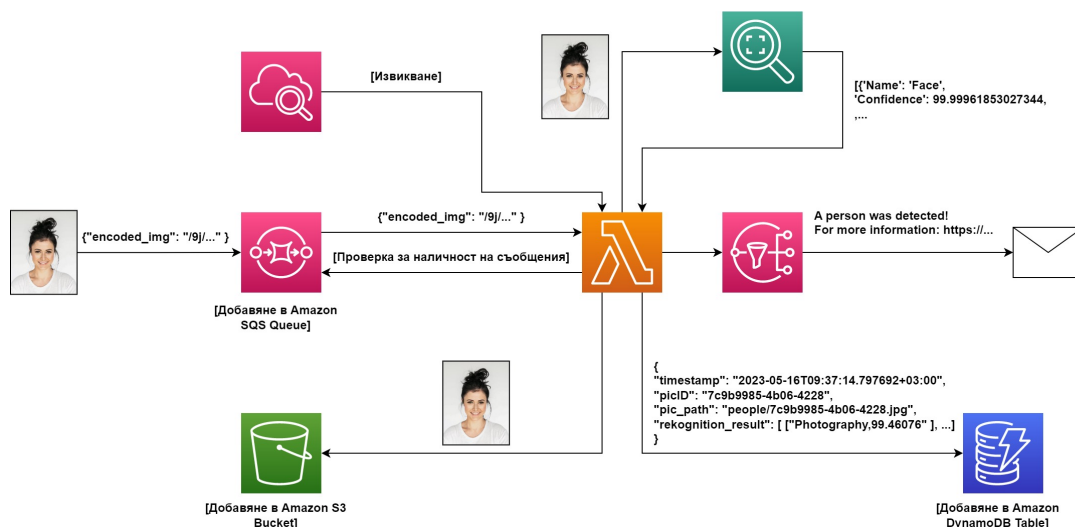


Фигура 43: Примерно изображение. [Jake Nackos, Unsplash]

Преобразуваме изображението в 64-битов формат. Създаваме JSON файл, в който срещу ключа "encoded_image" записваме кодираната снимка.

```
{  
  "encoded_img": "/9j/4AAQSkZJRgABAQEASABIAAD<...>KYH/9k="
```

Резултатът от обработката на изображението е показан на приложената диаграма.



Фигура 44: Работа на системата с примерни данни. [Diagrams.net]

8 Описание на програмния код

За реализирането на логиката в Amazon Lambda Function е използван езикът Python (версия 3.8) и библиотеката boto3.

8.1 Обработка на получено изображение

Включване на използваните библиотеки:

```
import base64
import boto3
from datetime import datetime
import dateutil
import json
import uuid
```

Деклариране на променливи, съдържащи използваните ресурси:

```
QUEUE = '' # URL на Amazon SQS Queue
BUCKET = '' # име на Amazon S3 Bucket
TOPIC = '' # ARN на Amazon SNS Topic
TABLE = '' # име на Amazon DynamoDB Table
REGION = '' # име на Amazon регион
```

Функция, връщаща получените от Amazon SQS опашката с URL стойността на QUEUE съобщения:

```
def get_messages():
    return boto3.client(
        'sqs',
        region_name=REGION
    ).receive_message(
        QueueUrl=QUEUE,
        MaxNumberOfMessages=1
    )
```

Функция, премахваща от опашката съобщение:

```
def delete_message(receipt_handle):
    return boto3.client(
        'sqs',
        region_name=REGION
    ).delete_message(
```

```

        QueueUrl=QUEUE,
        ReceiptHandle=receipt_handle
    )

```

Функция, създаваща временен файл със съдържание снимката, получена в опашката:

```

def write_picture(sqs_messages):
    pic_uuid = str(uuid.uuid4())
    pic_path = f'/tmp/{pic_uuid}.jpg'

    decoded_image = base64.b64decode(
        json.loads(
            sqs_messages['Messages'][0]['Body']
        )['encoded_img']
    )

    with open(pic_path, 'wb') as f:
        f.write(decoded_image)

    return pic_uuid, pic_path

```

Функция, извикваща Amazon Recognition върху подадена снимка, и връщаща 10-те етикета, получени с най-голяма вероятност:

```

def run_rekognition(pic_path):
    with open(pic_path, 'rb') as image:
        return boto3.client(
            'rekognition',
            region_name=REGION
        ).detect_labels(
            Image={
                'Bytes': image.read()
            },
            MaxLabels=10,
        )['Labels']

```

Функция, проверяваща дали Amazon Rekognition е поставил на етикета "човек" вероятност, по-голяма от 90%, и той е сред десетте разпознати с най-голяма вероятност обекти на снимката:

```

def detect_person(pic_path):
    rekognition_labels = run_rekognition(pic_path)

```

```

is_person = False
for label in rekognition_labels:
    label_name = label['Name'].lower()
    label_confidence = label['Confidence']

    if label_name in ['human', 'person'] and label_confidence >= 90:
        is_person = True
return is_person, rekognition_labels

```

Функция, добавяща към Amazon DynamoDB таблицата TABLE нов запис, съдържащ информация за снимката и резултата, получен от Amazon Rekognition:

```

def add_to_dynamoDB(pic_uuid, labels, pic_path):
    table = boto3.resource(
        'dynamodb',
        region_name=REGION
    ).Table(TABLE)

    table.put_item(
        Item={
            'timestamp': str(datetime.now(
                dateutil.tz.gettz('Europe/Sofia')
            ).isoformat()),
            'picID': pic_uuid,
            'pic_path': pic_path,
            'rekognition_result': [
                ','.join([label['Name'], str(label['Confidence'])])
                for label
                in labels
            ]
        }
    )

```

Функция, добавяща в Amazon S3 Bucket с име BUCKET подаденото като параметър изображение:

```

def add_to_bucket(img, pic_path):
    boto3.client(
        's3',
        region_name=REGION
    ).upload_file(

```



```

        img,
        BUCKET,
        pic_path
    )

```

Функция, генерираща валиден 24 часа URL към обект, съхранен в Amazon S3:

```

def get_public_url(file):
    return boto3.client(
        's3',
        region_name=REGION
    ).generate_presigned_url(
        ClientMethod='get_object',
        Params={
            'Bucket': BUCKET,
            'Key': file
        },
        ExpiresIn=86400
    )

```

Функция, публикуваща в TOPIC ново съобщение, съдържащо публичен линк към снимката, подадена като параметър:

```

def publish_to_topic(img_path):
    boto3.client(
        'sns',
        region_name=REGION
    ).publish(
        TopicArn=TOPIC,
        Message=f'A person was detected!'
                f'\nFor more information: {get_public_url(img_path)}'
    )

```

Основна функция, която получава всички съобщения в Amazon SQS опашката. Ако такива са налични, започва обработката на едно от тях. Проверява дали на снимката от текущото съобщение има човек, и, ако това е така, запазва снимката в Amazon S3 пространството за съхранение, добавя информацията за нея в Amazon DynamoDB таблицата и изпраща съобщение към Amazon SNS услугата. При завършване на работата с текущото изображение го премахва от опашката:

```

def lambda_handler(event, context):

```

```

sqs_messages = get_messages()

if "Messages" in sqs_messages:
    pic_uuid, pic_path = write_picture(sqs_messages)
    is_person, rekognition_labels = detect_person(pic_path)

    if is_person:
        saved_pic_path = pic_path.replace("/tmp/", "people/")
        add_to_dynamoDB(
            pic_uuid=pic_uuid,
            pic_path=saved_pic_path,
            labels=rekognition_labels
        )
        add_to_bucket(
            img=pic_path,
            pic_path=saved_pic_path
        )
        publish_to_topic(
            saved_pic_path
        )
        delete_message(sqs_messages['Messages'][0]['ReceiptHandle'])
    else:
        print("No pictures in queue")

```

8.2 Експортиране на Amazon DynamoDB таблица в CSV формат

Включване на използваните библиотеки:

```
import boto3
import csv
from datetime import datetime
import dateutil
```

Деклариране на променливи, съдържащи използваните ресурси:

```
BUCKET = '' # име на Amazon S3 Bucket
TABLE = '' # име на Amazon DynamoDB Table
TEMP_LOCATION = '' # адрес на временния CSV файл
REGION = '' # име на Amazon регион
```

Функция, извличаща данните от Amazon DynamoDB таблицата:

```
def scan_table(previous_response=None):
    table = boto3.resource(
        'dynamodb',
        region_name=REGION
    ).Table(
        TABLE
    )

    if previous_response is not None:
        return table.scan(
            ExclusiveStartKey=previous_response['LastEvaluatedKey']
        )
    else:
        return table.scan()
```

Функция, добавяща експортираната в CSV формат Amazon DynamoDB таблица в Amazon S3, като за име на файла се използва текущото време:

```
def add_to_bucket():
    boto3.resource(
        's3',
        region_name=REGION
    ).Bucket(
        BUCKET
```

```

).upload_file(
    TEMP_LOCATION,
    'dynamo_backup/' + str(
        datetime.now(
            dateutil.tz.gettz('Europe/Sofia')
        ).isoformat()
    ) + '.csv'
)

```

Основна функция, изличаща и записваща в CSV файл записите от Amazon DynamoDB таблицата, и добавяща получения файл в Amazon S3:

```

def lambda_handler(event, context):
    with open(TEMP_LOCATION, 'w') as output_file:
        writer = csv.writer(output_file)
        is_start = True

        response = {'LastEvaluatedKey': []}

        while 'LastEvaluatedKey' in response:
            if is_start:
                response = scan_table()
            else:
                response = scan_table(response)
            for item in response['Items']:
                if is_start:
                    writer.writerow(item.keys())
                is_start = False
                writer.writerow(item.values())

        add_to_bucket()

```

8.3 Уведомяване на потребителите за налични CSV файлове

Включване на използваните библиотеки:

```
import boto3
```

Деклариране на променливи, съдържащи използваните ресурси:

```
BUCKET = '' # име на Amazon S3 Bucket
TOPIC = '' # ARN на Amazon SNS Topic
REGION = '' # име на Amazon регион
```

Получаване на последния добавен в Amazon S3 пространството за съхранение CSV файл:

```
def last_updated_file():
    paginator = boto3.client(
        's3'
    ).get_paginator(
        "list_objects_v2"
    ).paginate(
        Bucket=BUCKET
    )

    result = None
    for page in paginator:
        if "Contents" in page:
            obj = sorted(
                page['Contents'],
                key=lambda x: x['LastModified']
            )[-1]
            if result is None or obj['LastModified'] > result['LastModified']:
                result = obj
    return result['Key']
```

Функция, генерираща валиден 24 часа URL към обект, съхранен в Amazon S3:

```
def get_public_url(file):
    return boto3.client(
        's3',
        region_name=REGION
```

```

).generate_presigned_url(
    ClientMethod='get_object',
    Params={
        'Bucket': BUCKET,
        'Key': file
    },
    ExpiresIn=86400
)

```

Функция, публикуваща в TOPIC ново съобщение, съдържащо публичен линк към снимката, подадена като параметър:

```

def publish_to_topic(img_path):
    boto3.client(
        'sns',
        region_name=REGION
    ).publish(
        TopicArn=TOPIC,
        Message=f'Weekly backup of dynamoDB is now available.'
                f'\nFor more information: {get_public_url(img_path)}'
    )

```

Основна функция, изпращаща съобщение към Amazon SNS Topic. Съобщението съдържа валиден 24 часа URL за достъп до последния обновен файл в Amazon S3 Bucket:

```

def lambda_handler(event, context):
    file_key = last_updated_file()
    publish_to_topic(file_key)

```

9 Приноси на студента, ограничения и възможности за бъдещо развитие

Поради ограничения при правата за достъп, някои от функционалностите не са налични чрез Amazon Academy. В следствие на това, използваните ресурси са съобразени с квотите в AWS Free Tier.

Потенциална посока за бъдещо развитие на проекта е добавяне на възможност за лицево разпознаване - потребителите могат да добавят своя снимка. При получаване на изображение с човек, лицата да биват сравнявани (напр. чрез използването на инструмент като DeepFace или чрез Amazon Rekognition) и на потребителя да се изпраща съобщение само ако на снимката не е бил разпознат той. Подобно разширение би могло да намери приложение при охранителните камери.

Възможност за разширяване на функционалността на проекта е използване на инструментите, предоставени от Amazon SageMaker, за автоматизация на анализа на събраните данни.

10 Какво научих

Работата по текущия проект е първият ми опит със създаването Cloud-базирано решение. По време на работата успях да се запозная по-обстойно с разгледаните по време на лекциите Amazon услуги.

Създаването на потребителски роли, които отговарят на Least Privilege принципа, и които осигуряват необходимото ниво на сигурност с оглед на съхранението на лични снимки, ми помогна да разбере една от по-непознатите за мен услуги - AWS IAM.

В процеса на работа имах възможността да се запозная с някои от услугите, предлагани от AWS, които не попадат в обхвата на курса - Amazon Rekognition.

11 Списък с фигури и таблици

Списък на фигурите

1	Общ преглед на връзките между услугите. [Diagrams.net]	4
2	Диаграма, показваща действията за обработване на получена снимка. [Diagrams.net]	5
3	Създаване на Amazon S3 Bucket - стъпка 1.	7
4	Създаване на Amazon S3 Bucket - стъпка 2.	7
5	Създаване на Amazon S3 Bucket - стъпка 3.	8
6	Създаване на Amazon S3 Bucket - стъпка 4.	8
7	Създаване на Amazon SQS опашка - стъпка 1.	9
8	Създаване на Amazon SQS опашка - стъпка 2.	9
9	Създаване на Amazon SQS опашка - стъпка 3.	9
10	Създаване на Amazon DynamoDB таблица - стъпка 1.	10
11	Създаване на Amazon DynamoDB таблица - стъпка 2.	10
12	Създаване на Amazon DynamoDB таблица - стъпка 3.	10
13	Създаване на Amazon DynamoDB таблица - стъпка 4.	11
14	Създаване на Amazon SNS Topic - стъпка 1.	12
15	Създаване на Amazon SNS Topic - стъпка 2.	12
16	Създаване на Amazon SNS Topic - стъпка 3.	12
17	Създаване на Amazon SNS Topic - стъпка 4.	13
18	Създаване на Amazon SNS Topic - стъпка 6.	13
19	Създаване на Amazon SNS Topic - стъпка 7.	14
20	Създаване на Amazon SNS Topic - стъпка 8.	14
21	Създаване на роли чрез AWS IAM - стъпка 1.	15
22	Създаване на роли чрез AWS IAM - стъпка 2.	15
23	Създаване на роли чрез AWS IAM - стъпка 3.	16
24	Създаване на роли чрез AWS IAM - стъпка 4.	16
25	Създаване на роли чрез AWS IAM - стъпка 7.	18
26	Създаване на Amazon Lambda Function - стъпка 1.	22
27	Създаване на Amazon Lambda Function - стъпка 2.	22
28	Създаване на Amazon Lambda Function - стъпка 3.	22
29	Създаване на Amazon Lambda Function - стъпка 4.	23
30	Създаване на Amazon Lambda Function - стъпка 5.	23
31	Създаване на Amazon Lambda Function - стъпка 7.	24
32	Създаване на Amazon Lambda Function - стъпка 8.	24
33	Създаване на Amazon CloudWatch Rule - стъпка 1.	25
34	Създаване на Amazon CloudWatch Rule - стъпка 2.	25
35	Създаване на Amazon CloudWatch Rule - стъпка 3.	25

36	Създаване на Amazon CloudWatch Rule - стъпка 4.	26
37	Създаване на Amazon CloudWatch Rule - стъпка 5.	26
38	Създаване на Amazon CloudWatch Rule - стъпка 7.	26
39	Потвърждаване на абонамент към Amazon SNS Subscription.	27
40	Съобщение за откриване на човек на снимката.	27
41	Съобщение за готово седмично резервно копие на Amazon DynamoDB Table.	28
42	Потвърждаване на прекратяването на абонамент към Amazon SNS Subscription.	28
43	Примерно изображение. [Jake Nackos, Unsplash]	29
44	Работа на системата с примерни данни. [Diagrams.net]	29

12 Използвани източници

- [1] Boto3 documentation
- [2] DynamoDB, explained.
- [3] Analyzing images stored in an Amazon S3 bucket
- [4] AWS Rekognition using Lambda with SQS and SNS. Part 1 (step by step tutorial)
- [5] Building a Simple Scheduled Task with AWS using Lambda Function and Amazon Cloudwatch Event
- [6] Sharing objects using presigned URLs