02/07/2020

My Solution for Home Exercise

Dynamic Yield

Author:

Jordan Yefet

Contact information:

Email | LinkedIn

02/07/2020

Contents

המטלה עצמה:

Home Exercise: A Story of API's and Cats

Goal

The following experiment is meant to (a) be fun, and (b) test your skills in rapidly utilizing and binding together a few state-of-the-art APIs to create new functionality, utilizing off-the-shelf managed components as possible.

If you get stuck or find that the exercise takes too much of your time, you can wrap it up by documenting what you've accomplished vs. what was left, and note how you would tackle everything not yet implemented/working properly. Also, you can contact us for questions, if you're stuck and it becomes a choice of either a small hint or giving up...

In Summary

At Dynamic Yield, we have a very hungry cat. The cat is being fed by uploading images of appropriate food (fish, milk or bread) into an S3 bucket. If the cat was not fed with a proper image for 15 minutes or more, an email should be automatically sent to an operator's address. If, following a warning email, the cat has then been fed again, a "back to normal" email should be sent. There should be only a single alert e-mail & back-to-normal e-mail, per each hunger period.

In more detail

1. Write a program which listens on a specific S3 bucket for new files. We strongly prefer that you use an AWS Lambda serverless function, which can be tied directly to S3 change notifications.

   If using a Lambda doesn't work for you, write a locally-running program which periodically checks for files it didn't yet process (or any other method you'd like).

2. When a new file is encountered, connect to Amazon Rekognition API to detect labels in the image.

3. If the labels returned from Rekognition include one of the "proper" food types (with 90% confidence or more), that means that suitable food has been given. In that case, we can update the last timestamp when valid food has been given. This value may be stored in

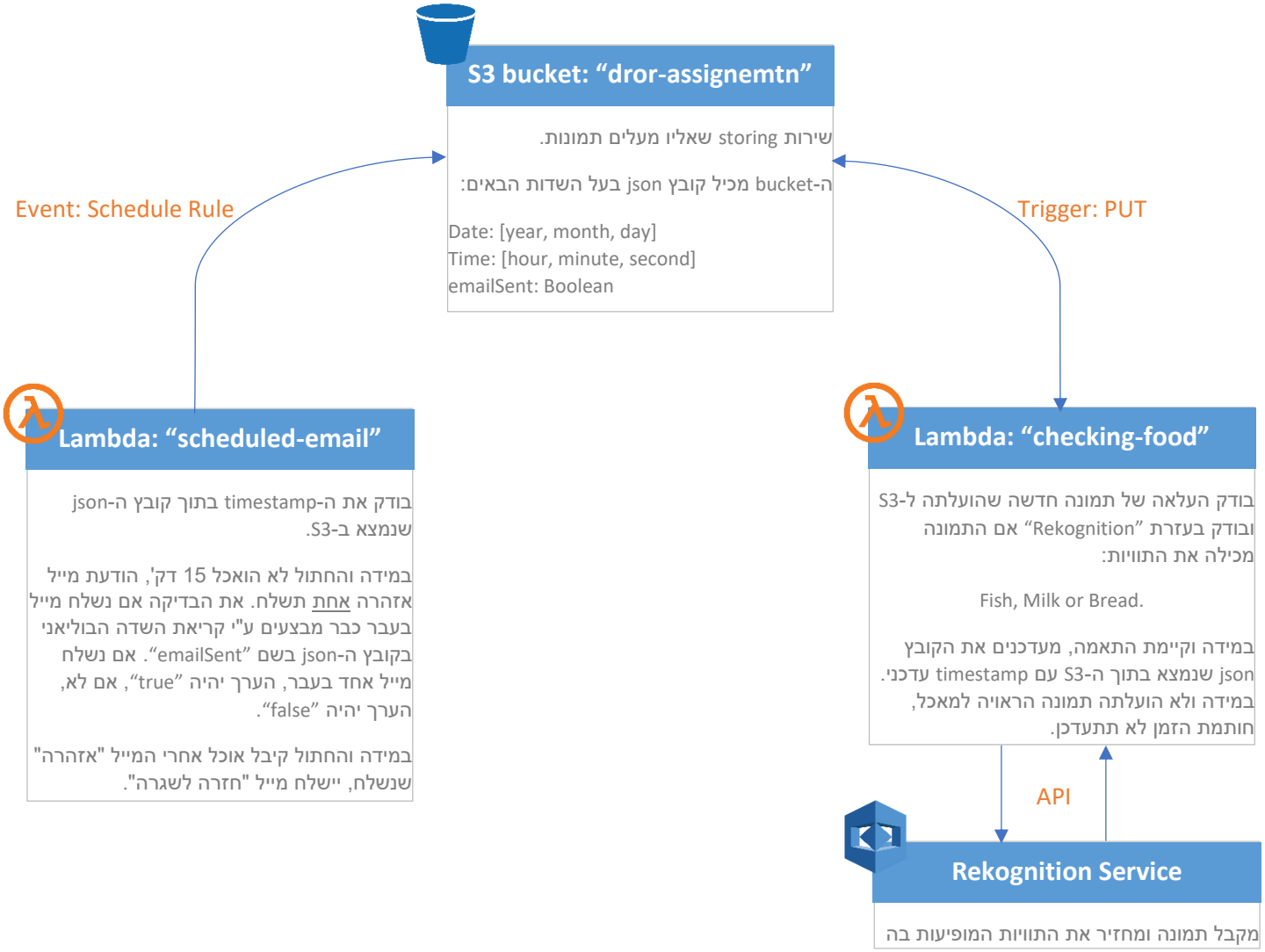
a simple file, Redis, ElastiCache or any super-simple storage.

4. A second program (or a scheduled AWS Lambda function) shall run periodically, and check the cat's feeding state:
    a. If the cat has not been fed for more than 15 minutes, send an email (but only once!)
    b. If, after a warning email was already sent, you detect that the cat was again fed, send a "back to normal" email - once per returning to normal state.

Good Luck!

02/07/2020

הסבר ארכיטקטורי:



02/07/2020

הסבר כללי:

לפני שהתחלתי לייצר את כל הרכיבים שתיארתי בארכיטקטורה, הייתי צריך להגדיר policies שונים ולשייך אותם ל-roles, כדי שכל רכיב יוכל לתקשר עם השאר.

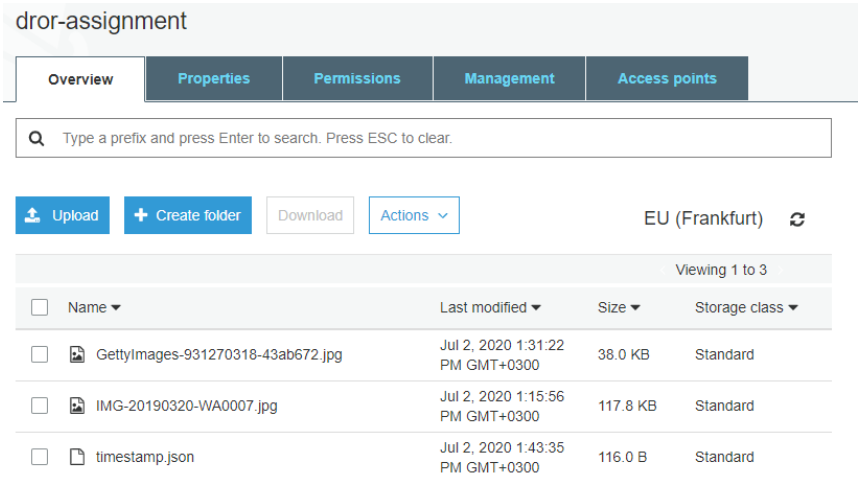
Roles:

- LambdaSendingEmail:
  - AmazonS3FullAccess
  - AWSLambdaBasicExecutionRole
  - LambdaSendingEmail (Custom. Allow lambda functions to send emails)
- S3PutObjectRole:
  - AmazonS3FullAccess
  - AmazonRekognitionFullAccess
  - AWSLambdaBasicExecutionRole

02/07/2020

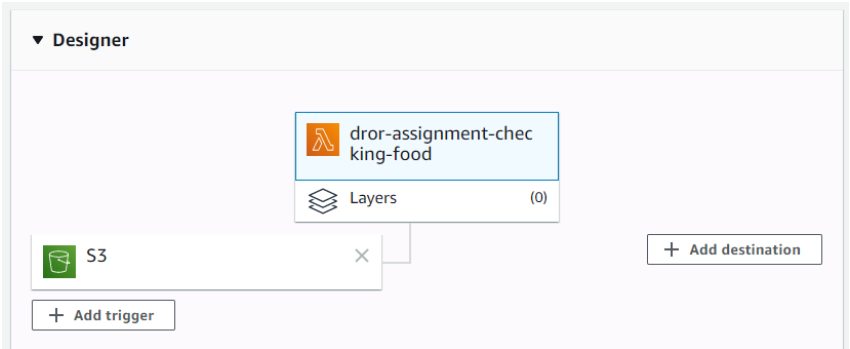
צילום הרכיבים והקוד:

S3 bucket: “dror-assignemnt”:



02/07/2020

Lambda function: “dror-assignment-checking-food”:




```
import os
import json
from datetime import *
import boto3

max_labels = 10
S3Bucket = "dror-assignment"
fileName = "timestamp.json"

def lambda_handler(event, context):
    # Filename of object (with path)
    photo_path = event['Records'][0]['s3']['object']['key']
    photo = os.path.basename(photo_path)

    client=boto3.client('rekognition')

    response = client.detect_labels(
        Image={'S3Object':{'Bucket':S3Bucket,'Name':photo}},
        MaxLabels=max_labels,
        MinConfidence = 90)


    print('\n')
    labels = []
    for label in response['Labels']:
        labels.append(label['Name'])

    print("labels: ")
    print(labels)
```

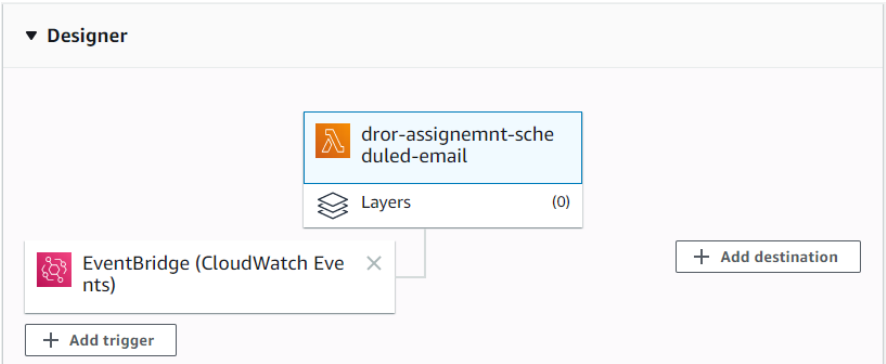
02/07/2020

```
    isFood = False
    for label in labels:
        if(label == "Fish" or label == "Bread" or label == "Milk"):
            isFood = True
            break

    if(isFood):
        print("Food has been uploaded.")
        print("Updating timestamp...")
        timestamp = timestampUpdate()
        print(f"timestamp has been updated: {timestamp}")
    else:
        print("The uploaded photo is not food!")




#######################################################
#writing to file
def timestampUpdate():
    #getting data from timestamp.json
    s3 = boto3.resource('s3')
    obj = s3.Object(S3Bucket, fileName)
    jsonData = json.load(obj.get()['Body'])

    #getting current datetime
    currentDate = date.today()
    currentTime = datetime.now().time()

    ##writing data to jsonData
    jsonData['date']['year'] = currentDate.year
    jsonData['date']['month'] = currentDate.month
    jsonData['date']['day'] = currentDate.day
    jsonData['time']['hour'] = currentTime.hour
    jsonData['time']['minute'] = currentTime.minute
    jsonData['time']['second'] = currentTime.second

    print("writing to data...")
    obj.put(Body=json.dumps(jsonData))

    return jsonData
```

02/07/2020

Lambda function: “dror-assignemnt-scheduled-email”:

Schedule Rule:
15 minutes




```
import json
import os
import boto3
from datetime import *
from botocore.exceptions import ClientError

S3Bucket = "dror-assignment"
fileName = "timestamp.json"
sender = "jordan.yefet90@gmail.com"
recipient = "jordan.yefet90@gmail.com"
emailSubject = "A Message from your Cat!"

def lambda_handler(event, context):
    timeLimitForFood = timedelta(minutes=15)


    #getting data from timestamp.json
    s3 = boto3.resource('s3')
    obj = s3.Object(S3Bucket, fileName)
    jsonData = json.load(obj.get()['Body'])

    #getting the date inside of the json file
    timestamp_year = jsonData['date']['year']
    timestamp_month = jsonData['date']['month']
    timestamp_day = jsonData['date']['day']

    #getting the time inside of the json file
    timestamp_hour = jsonData['time']['hour']
    timestamp_minute = jsonData['time']['minute']
    timestamp_second = jsonData['time']['second']
```

```
    #getting the email status Boolean
    timestamp_emailSent= jsonData['emailSent']


    #building Date and Time objects of the json data
    timestampDate = date(timestamp_year, timestamp_month, timestamp_day)
    timestampTimedelta = timedelta(hours=timestamp_hour, minutes=timestamp
_minute, seconds=timestamp_second)
    #building Date and Time objects of the current Date and Time
    currentDate = date.today()
    currentTime = datetime.now().time()
    currentTimedelta = timedelta(hours=currentTime.hour, minutes=currentTi
me.minute, seconds=currentTime.second)

    #conditional time delta - (given in the home-work)
    limit_timedelata = timedelta(minutes=15)


    if(timestamp_emailSent == False):
        if((currentDate==timestampDate) and ((currentTimedelta - timestamp
Timedelta) < limit_timedelata)): #the first condition checks if it's the s
ame date, the second condition checks if the timedelta is lower then 15 mi
nutes
            print("All good, the cat already ate!")
        else:
            body = "Warning! Feed the cat!"
            print(f"Sending an Email: {body}")
            emailFunction(sender, recipient, emailSubject, body)
            jsonData['emailSent'] = True
            #writing to jsonData
            obj.put(Body=json.dumps(jsonData))
    else:
        if((currentDate==timestampDate) and ((currentTimedelta - timestamp
Timedelta) < limit_timedelata)):
            body = "Back to normal..."
            print(f"Sending an Email: {body}")
            emailFunction(sender, recipient, emailSubject, body)
            jsonData['emailSent'] = False
            #writing to jsonData
            obj.put(Body=json.dumps(jsonData))
        else:
            print("The Email has already been sent. Check your inbox!")
```

```
def emailFunction(sender, recipient, emailSubject, emailBody):
    # Replace sender@example.com with your "From" address.
    # This address must be verified with Amazon SES.
    SENDER = sender

    # Replace recipient@example.com with a "To" address. If your account
    # is still in the sandbox, this address must be verified.
    RECIPIENT = recipient

    # Specify a configuration set. If you do not want to use a configurati
on
    # set, comment the following variable, and the
    # ConfigurationSetName=CONFIGURATION_SET argument below.
    CONFIGURATION_SET = "ConfigSet"

    # If necessary, replace us-west-
2 with the AWS Region you're using for Amazon SES.
    AWS_REGION = "eu-central-1"

    # The subject line for the email.
    SUBJECT = emailSubject

    # The email body for recipients with non-HTML email clients.
    BODY_TEXT = (#"A Message from your Cat!\r\n"
                 "{emailBody}"
                )

    # The HTML body of the email.
    BODY_HTML = """<html>
    <head></head>
    <body>
      <!-- <h1>A Message from your Cat!</h1> -->
      <p>""" + str(emailBody) + """</p>
    </body>
    </html>
                 """

    # The character encoding for the email.
    CHARSET = "UTF-8"

    # Create a new SES resource and specify a region.
    client = boto3.client('ses',region_name=AWS_REGION)
```

```
    # Try to send the email.
    try:
        #Provide the contents of the email.
        response = client.send_email(
            Destination={
                'ToAddresses': [
                    RECIPIENT,
                ],
            },
            Message={
                'Body': {
                    'Html': {
                        'Charset': CHARSET,
                        'Data': BODY_HTML,
                    },
                    'Text': {
                        'Charset': CHARSET,
                        'Data': BODY_TEXT,
                    },
                },
                'Subject': {
                    'Charset': CHARSET,
                    'Data': SUBJECT,
                },
            },
            Source=SENDER,
            # If you are not using a configuration set, comment or delete
the
            # following line
            # ConfigurationSetName=CONFIGURATION_SET,
        )
    # Display an error if something goes wrong.
    except ClientError as e:
        print(e.response['Error']['Message'])
    else:
        print("Email sent! Message ID:"),
        print(response['MessageId'])
```

02/07/2020

ביצוע בדיקות:

Disclaimer:

את הבדיקות עם ה-"scheduled-email" :Lambda אבצע on-demand ולא אחכה כל 15 דקות לבדיקה אוטומטית.

בדיקה 1:

ה-S3 ריק (מלבד הקובץ json). מעלה תמונה של דג (נכלל תחת אוכל).

תמונה:

Log events עבור "checking-food":

Log events

Filter events | Clear | 1m | 30

	Timestamp	Message
		There are older events to load. *Load more.*
▶	2020-07-02T…	START RequestId: 7e6982bd-2768-489c-8102-0f11d22d1311 Version: $LATEST
▶	2020-07-02T…	labels:
▶	2020-07-02T…	['Fish', 'Animal', 'Goldfish']
▶	2020-07-02T…	Food has been uploaded.
▶	2020-07-02T…	Updating timestamp...
▶	2020-07-02T…	writing to data...
▶	2020-07-02T…	timestamp has been updated: {'date': {'year': 2020, 'month': 7, 'day': 2}, 'time': {'hour': 11, 'minute': 35, 'second': 20}, 'emailSent': False}
▶	2020-07-02T…	END RequestId: 7e6982bd-2768-489c-8102-0f11d22d1311
▶	2020-07-02T…	REPORT RequestId: 7e6982bd-2768-489c-8102-0f11d22d1311 Duration: 2797.85 ms Billed Duration: 2800 ms Memory Size: 128 MB Max Memory Used: 72 MB Init Duration: 168.22 ms

חותמת הזמן עודכנה בתוך ה-S3 והיא מוצגת ל-log events.

02/07/2020

Log events עבור "scheduled-email":

כשאנחנו עדיין בתחום הזמן הרצוי (עברו פחות מ-15 דקות מהארוחה הקודמת) –

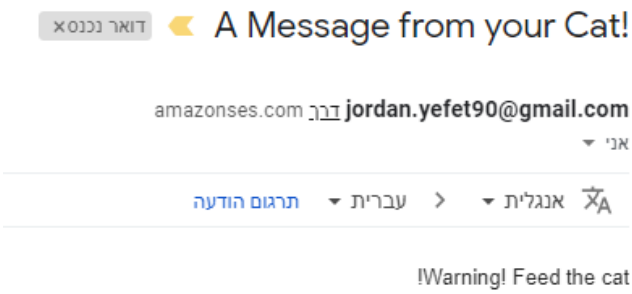
	Timestamp	Message
		There are older events to load. *Load more.*
▶	2020-07-02T…	START RequestId: 26e58233-e646-42f4-8f74-24df1ee9ee91 Version: $LATEST
▶	2020-07-02T…	All good, the cat already ate!
▶	2020-07-02T…	END RequestId: 26e58233-e646-42f4-8f74-24df1ee9ee91

כשעברו 15 דקות והחתול עדיין לא אכל –

▶	2020-07-02T14:53:19.248+03:00	START RequestId: c3a4cfa7-86b8-4e1d-a6cb-2619e4405762 Version: $LATEST
▶	2020-07-02T14:53:19.506+03:00	Sending an Email: Warning! Feed the cat!
▶	2020-07-02T14:53:20.132+03:00	Email sent! Message ID:
▶	2020-07-02T14:53:20.132+03:00	010701730f5fbf88-89641468-2351-4d66-afb7-0df68ead8c0f-000000
▶	2020-07-02T14:53:20.203+03:00	END RequestId: c3a4cfa7-86b8-4e1d-a6cb-2619e4405762

המשתנה emailSent שונה ל-"True", וזה יימנע שליחת X מיילים מתמשכים במידה והחתול לא יקבל כל X*15[minutes].

Email received:



02/07/2020

בדיקה 2:

ה-S3 מכיל בתוכו את התמונה של הדג. מעלה תמונה שהיא בוודאות לא אוכל.

תמונה:

(תמונה של הגיטרה שלי)

Log events עבור "checking-food":

▶	Timestamp	Message
		There are older events to load. *Load more.*
▶	2020-07-02T…	START RequestId: a3abc9be-97c9-44c7-800c-616458bc4ded Version: $LATEST
▶	2020-07-02T…	labels:
▶	2020-07-02T…	['Guitar', 'Leisure Activities', 'Musical Instrument', 'Bass Guitar', 'Electric Guitar']
▶	2020-07-02T…	The uploaded photo is not food!
▶	2020-07-02T…	END RequestId: a3abc9be-97c9-44c7-800c-616458bc4ded

חותמת הזמן לא עודכנה בתוך ה-S3, ולכן אינה מוצגת ב-log events.

02/07/2020

Log events עבור "scheduled-email":

לאחר "בדיקה 1", הזמן הרצוי עבר כבר, ונשלח מייל בעבר. לכן מה שנקבל:

	Timestamp	Message
		There are older events to load. *Load more.*
▶	2020-07-02T…	START RequestId: 6f140f81-351d-41b6-93b0-c46688b0e19d Version: $LATEST
▶	2020-07-02T…	The Email has already been sent. Check your inbox!
▶	2020-07-02T…	END RequestId: 6f140f81-351d-41b6-93b0-c46688b0e19d

Email received:

לא קיבלנו מייל מאחר והוא כבר נשלח בעבר.

02/07/2020

בדיקה 3:

ה-S3 מכיל בתוכו את התמונה של הדג ושל הגיטרה חשמלית. מעלה תמונה שהיא בוודאות אוכל.

תמונה:

Log events עבור "checking-food":

	Timestamp	Message
		There are older events to load. *Load more.*
▶	2020-07-02T…	START RequestId: 826c37ae-e3d3-4347-9136-eeaa4ac9e05a Version: $LATEST
▶	2020-07-02T…	labels:
▶	2020-07-02T…	['Beverage', 'Drink', 'Milk']
▶	2020-07-02T…	Food has been uploaded.
▶	2020-07-02T…	Updating timestamp...
▶	2020-07-02T…	writing to data...
▶	2020-07-02T…	timestamp has been updated: {'date': {'year': 2020, 'month': 7, 'day': 2}, 'time': {'hour': 12, 'minute': 27, 'second': 17}, 'emailSent': True}
▶	2020-07-02T…	END RequestId: 826c37ae-e3d3-4347-9136-eeaa4ac9e05a

*יש לשים לב שהמשתנה הבוליאני "emailSent" עדיין לא השתנה, וזה מאחר שהוא משתנה רק כשהפונקציה "scheduled-email" תרוץ.

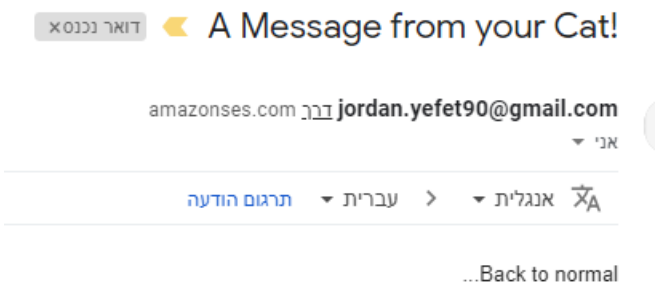
02/07/2020

Log events עבור "scheduled-email":

נשלח מייל חזרה "Back to normal..."

▶	2020-07-02T15:39:16.510+03:00	START RequestId: 040ac938-a878-4897-b25f-b95f82f3406f Version: $LATEST
▶	2020-07-02T15:39:18.484+03:00	Sending an Email: Back to normal...
▶	2020-07-02T15:39:18.949+03:00	Email sent! Message ID:
▶	2020-07-02T15:39:18.949+03:00	010701730f89d832-38e2d085-f506-49a3-8c24-c50086733045-000000
▶	2020-07-02T15:39:19.055+03:00	END RequestId: 040ac938-a878-4897-b25f-b95f82f3406f

Email received:

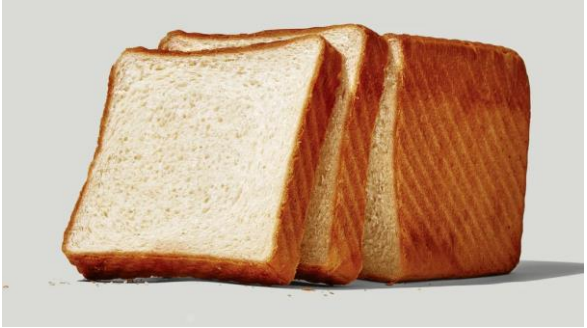


02/07/2020

בדיקה 4:

נעלה תמונה שהיא בוודאות אוכל, כאשר הזמן לא עבר את ה-15 דקות (ז"א החתול עדיין שבע).

תמונה:



Log events עבור "checking-food":

```
2020-07-02T…    START RequestId: 59da1f0c-9b21-422b-9610-3114ea9a79f8 Version: $LATEST
2020-07-02T…    labels:
2020-07-02T…    ['Bread', 'Food', 'French Toast', 'Toast']
2020-07-02T…    Food has been uploaded.
2020-07-02T…    Updating timestamp...
2020-07-02T…    writing to data...
2020-07-02T…    timestamp has been updated: {'date': {'year': 2020, 'month': 7, 'day': 2}, 'time': {'hour': 12, 'minute': 45, 'second': 9}, 'emailSent': False}
```

במקרה של הפונקציה "checking-food" היא לא תציג הודעה שונה. היא פשוט תעדכן את ה-timestamp.
ניתן לראות שהמשתנה הבוליאני השתנה ל-"False" לאחר שהפונקציה "scheduled-email" רצה בבדיקה 3.

Log events עבור "scheduled-email":

```
2020-07-02T…    START RequestId: 1a3298fa-c1e2-4fa0-b0a4-e16f4e07eec1 Version: $LATEST
2020-07-02T…    All good, the cat already ate!
2020-07-02T…    END RequestId: 1a3298fa-c1e2-4fa0-b0a4-e16f4e07eec1
```

Email received:

לא קיבלנו מייל מאחר והחתול כבר שבע (לא עברו 15 דקות מהפעם האחרונה שהוא אכל).