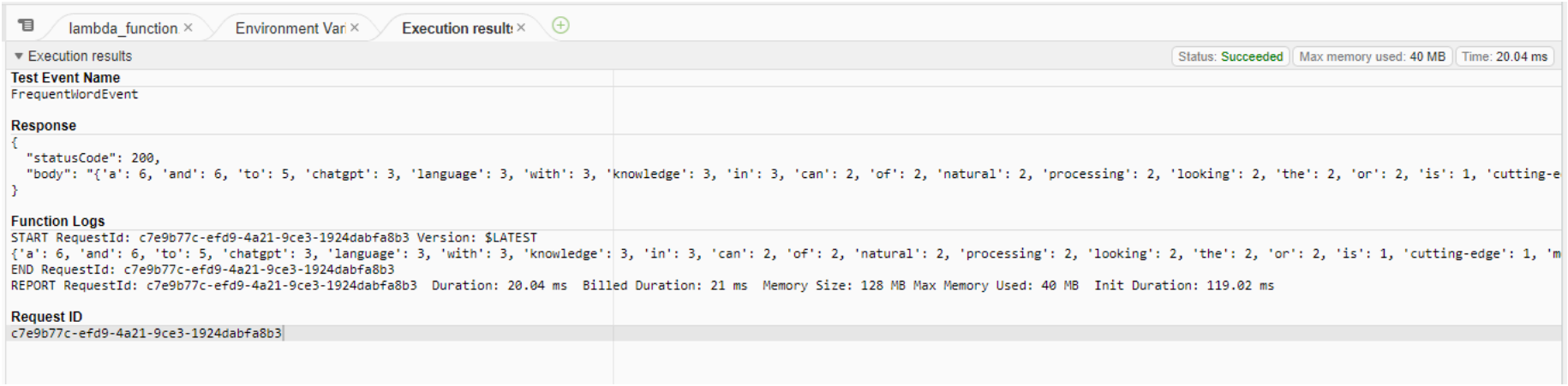
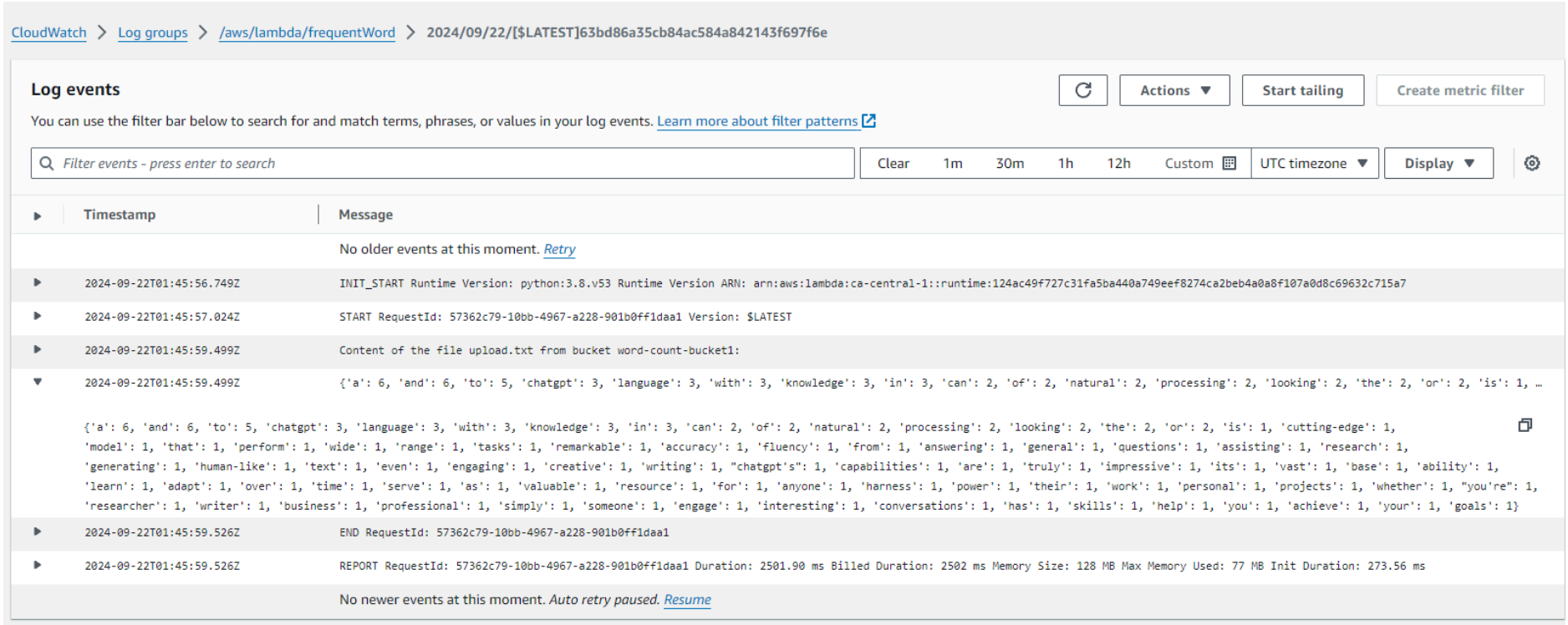
Step 6-7

To trigger my lambda function, I added a S3 trigger to the *frequentWord()* Lambda function, additionally, ".txt" is added to the optional suffix box in the trigger configuration, this means the Lambda function will trigger when a text file is uploaded to the S3 bucket. The *upload.txt* is uploaded to the S3 Bucket by first specifying the access key environments, and by running the *upload_file()* function. Once these steps are done, the Lambda function will run and parse the paragraph provided in the *upload.txt* file. The result can be found in "Monitor"->"logging"->"View logs in CloudWatch".

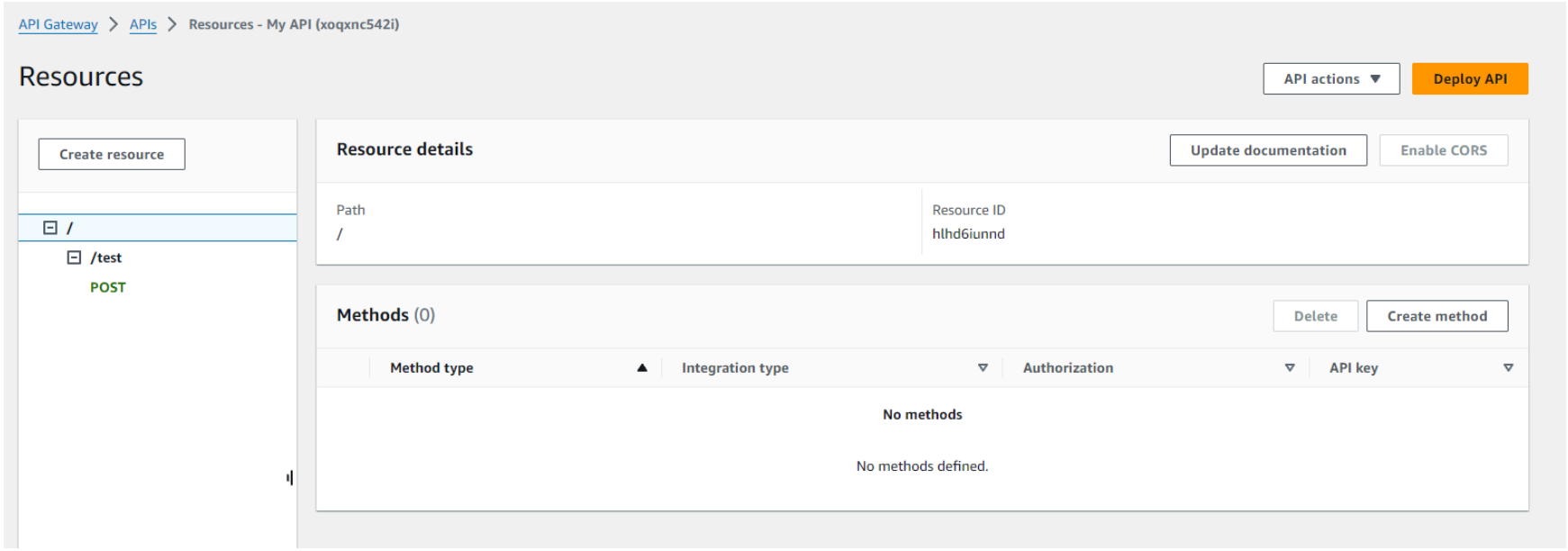
Execution result:
The below execution result shows the result of the *word_count* Lambda Function by passing a JSON event object as an argument. The JSON event object contains the input paragraph to be parsed.



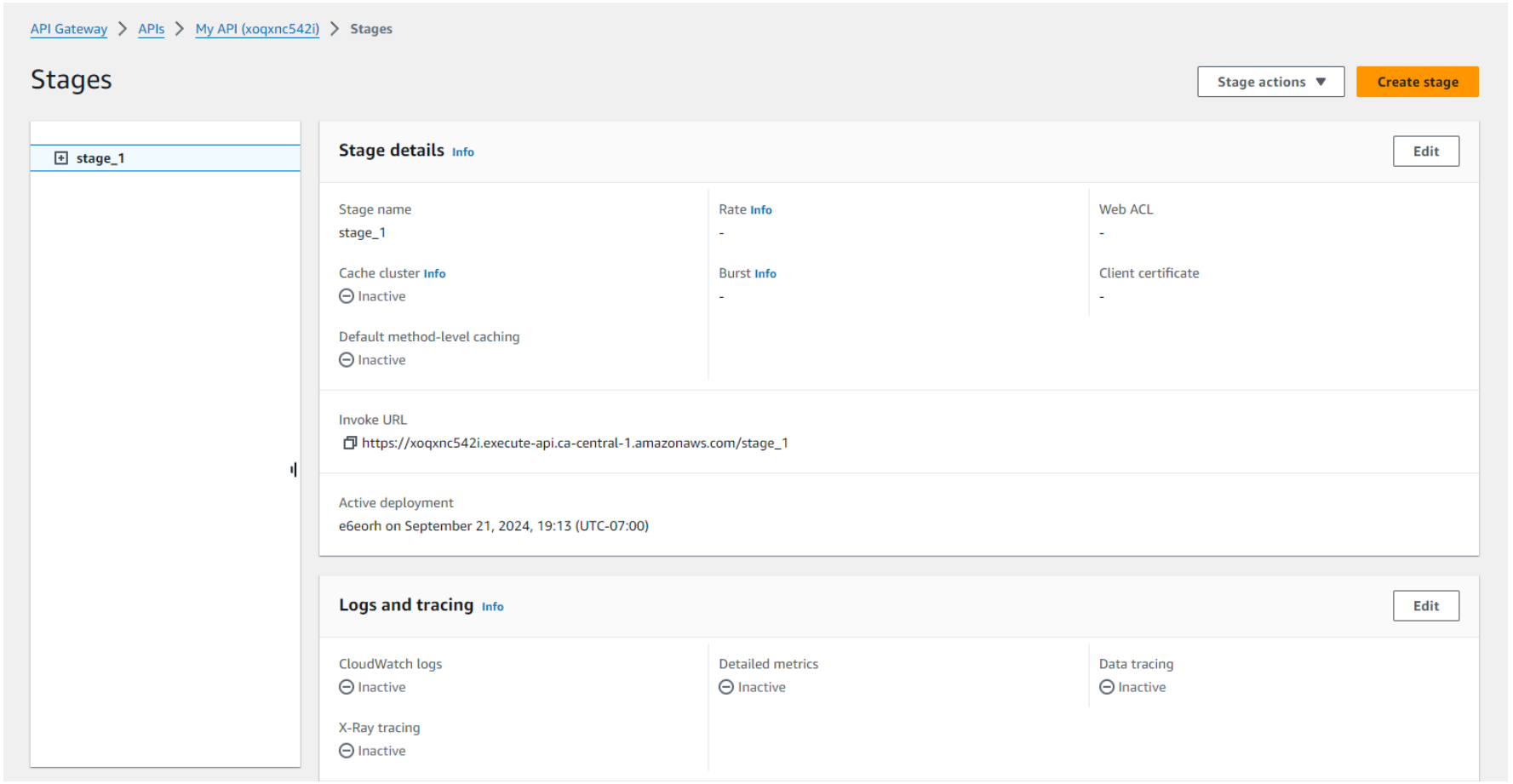

CloudWatch > Log groups > /aws/lambda/frequentWord > 2024/09/22/[$LATEST]63bd86a35cb84ac584a842143f697f6e
Log events
Actions
Start tailing
Create metric filter
You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns
Filter events - press enter to search
Clear
1m
30m
1h
12h
Custom
UTC timezone
Display
Timestamp
Message
No older events at this moment. Retry
2024-09-22T01:45:56.749Z INIT_START Runtime Version: python:3.8.v53 Runtime Version ARN: arn:aws:lambda:ca-central-1::runtime:124ac49f727c31fa5ba440a749eef8274ca2beb4a0a8f107a0d8c69632c715a7
2024-09-22T01:45:57.024Z START RequestId: 57362c79-10bb-4967-a228-901b0ff1daa1 Version: $LATEST
2024-09-22T01:45:59.499Z Content of the file upload.txt from bucket word-count-bucket1:
2024-09-22T01:45:59.499Z {'a': 6, 'and': 6, 'to': 5, 'chatgpt': 3, 'language': 3, 'with': 3, 'knowledge': 3, 'in': 3, 'can': 2, 'of': 2, 'natural': 2, 'processing': 2, 'looking': 2, 'the': 2, 'or': 2, 'is': 1, …
{'a': 6, 'and': 6, 'to': 5, 'chatgpt': 3, 'language': 3, 'with': 3, 'knowledge': 3, 'in': 3, 'can': 2, 'of': 2, 'natural': 2, 'processing': 2, 'looking': 2, 'the': 2, 'or': 2, 'is': 1, 'cutting-edge': 1,
'model': 1, 'that': 1, 'perform': 1, 'wide': 1, 'range': 1, 'tasks': 1, 'remarkable': 1, 'accuracy': 1, 'fluency': 1, 'from': 1, 'answering': 1, 'general': 1, 'questions': 1, 'assisting': 1, 'research': 1,
'generating': 1, 'human-like': 1, 'text': 1, 'even': 1, 'engaging': 1, 'creative': 1, 'writing': 1, "chatgpt's": 1, 'capabilities': 1, 'are': 1, 'truly': 1, 'impressive': 1, 'its': 1, 'vast': 1, 'base': 1, 'ability': 1,
'learn': 1, 'adapt': 1, 'over': 1, 'time': 1, 'serve': 1, 'as': 1, 'valuable': 1, 'resource': 1, 'for': 1, 'anyone': 1, 'harness': 1, 'power': 1, 'their': 1, 'work': 1, 'personal': 1, 'projects': 1, 'whether': 1, "you're": 1,
'researcher': 1, 'writer': 1, 'business': 1, 'professional': 1, 'simply': 1, 'someone': 1, 'engage': 1, 'interesting': 1, 'conversations': 1, 'has': 1, 'skills': 1, 'help': 1, 'you': 1, 'achieve': 1, 'your': 1, 'goals': 1}
2024-09-22T01:45:59.526Z END RequestId: 57362c79-10bb-4967-a228-901b0ff1daa1
2024-09-22T01:45:59.526Z REPORT RequestId: 57362c79-10bb-4967-a228-901b0ff1daa1 Duration: 2501.90 ms Billed Duration: 2502 ms Memory Size: 128 MB Max Memory Used: 77 MB Init Duration: 273.56 ms
No newer events at this moment. Auto retry paused. Resume

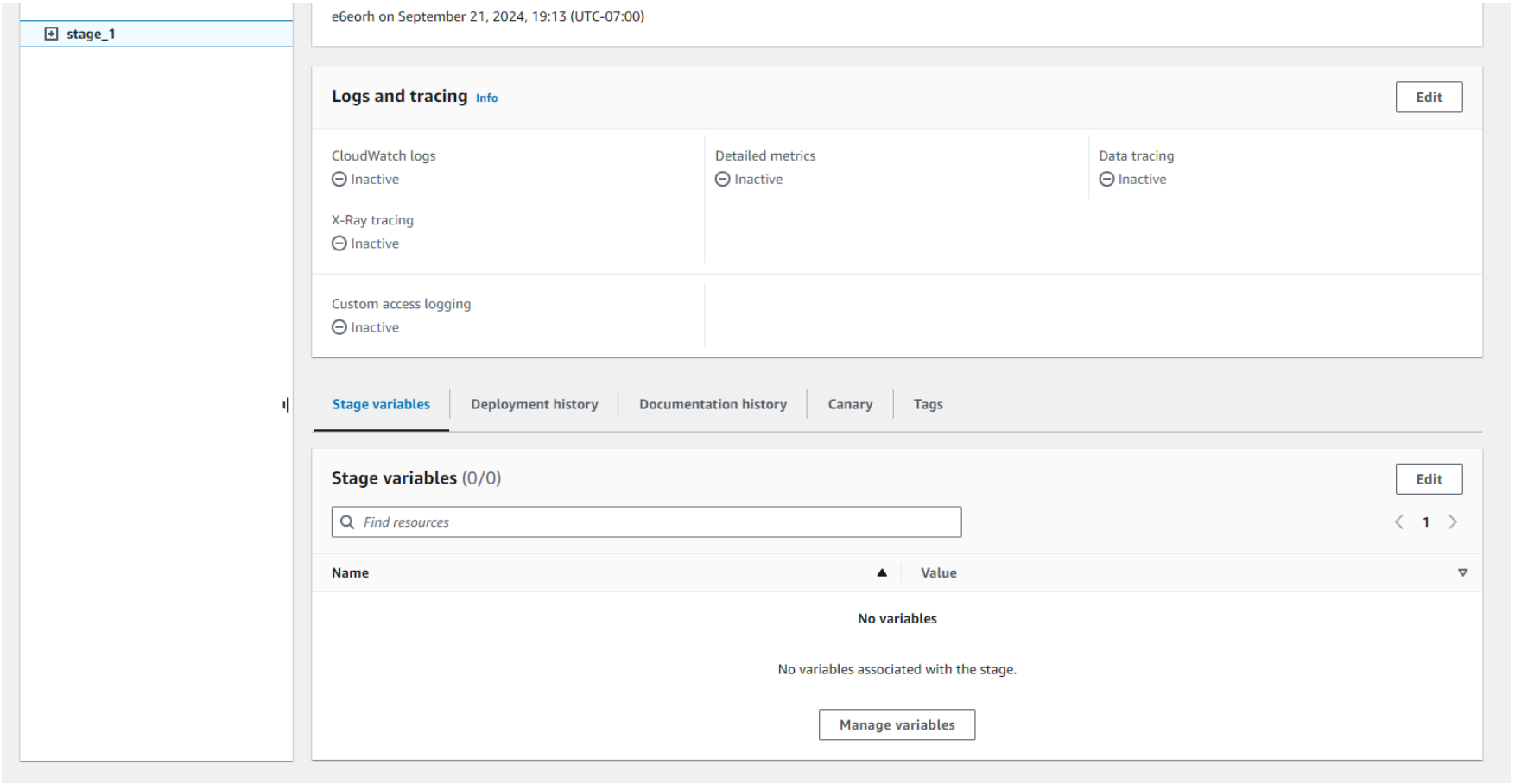
Step 8

API Resources Page:

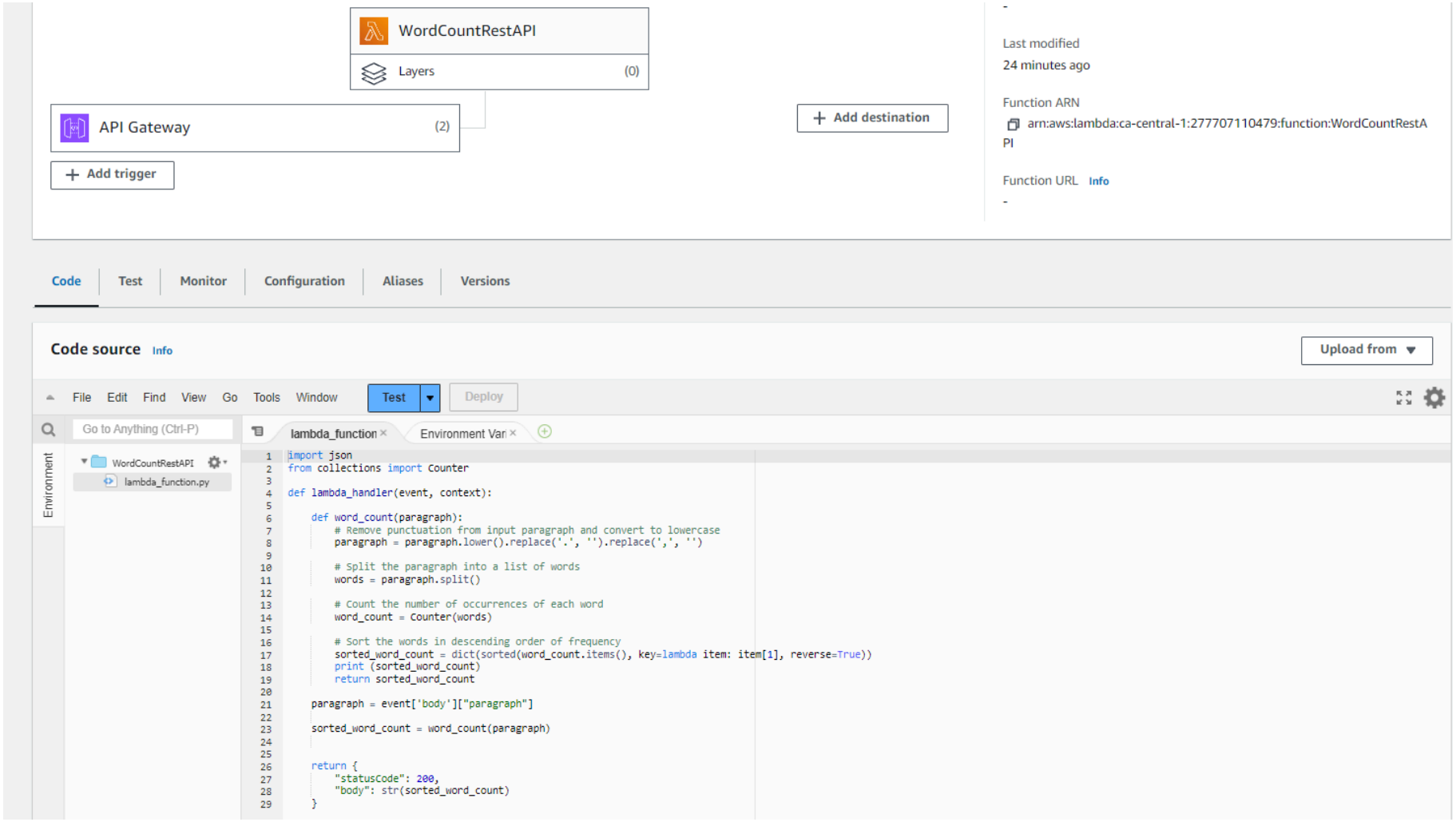


API Stages Page:

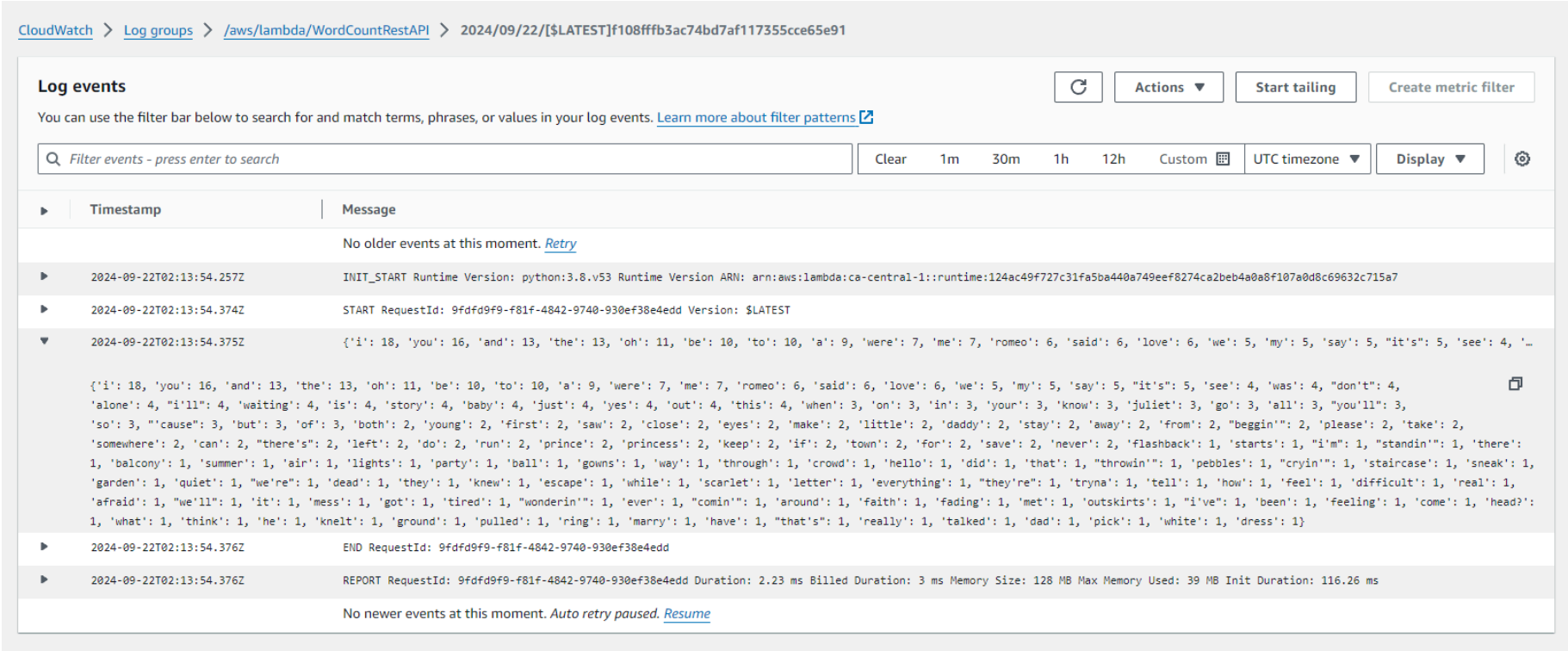



stage_1
e6eorh on September 21, 2024, 19:13 (UTC-07:00)
Logs and tracing Info
Edit
CloudWatch logs
Inactive
Detailed metrics
Inactive
Data tracing
Inactive
X-Ray tracing
Inactive
Custom access logging
Inactive
Stage variables
Deployment history
Documentation history
Canary
Tags
Stage variables (0/0)
Edit
Find resources
1
Name
Value
No variables
No variables associated with the stage.
Manage variables

Serverless Function:



Execution result:



URL:

invoke_url_and_resource_path = "https://xoqxnc542i.execute-api.ca-central-1.amazonaws.com/stage_1/test"

Step 9

Similar to Step 7, I created a S3 bucket to trigger my *goat_latin()* Lamba Function. The trigger code is the *upload_file()* provided in the assignment, with some minor changes to the environment variables and bucket name. Once *upload_file()* is ran, the *upload.txt* file containing a sentence will be uploaded to the S3 bucket, thereby triggering the *goat_latin()* Lambda function, which will parse the sentence provided in the *upload.txt* file into goat latin.

Serverless Function (goat_latin):

Goat Latin rules:
You are given a string sentence that consist of words separated by spaces. Each word consists of lowercase and uppercase letters only.
We would like to convert the sentence to "Goat Latin" (a made-up language similar to Pig Latin.) The rules of Goat Latin are as follows:
If a word begins with a vowel ('a', 'e', 'i', 'o', or 'u'), append "ma" to the end of the word.

- For example, the word "apple" becomes "applema".

If a word begins with a consonant (i.e., not a vowel), remove the first letter and append it to the end, then add "ma".

- For example, the word "goat" becomes "oatgma".

Add one letter 'a' to the end of each word per its word index in the sentence, starting with 1.

- For example, the first word gets "a" added to the end, the second word gets "aa" added to the end, and so on.

Code source Info

File Edit Find View Go Tools Window Test Deploy Changes not deployed

Go to Anything (Ctrl-P)

Environment

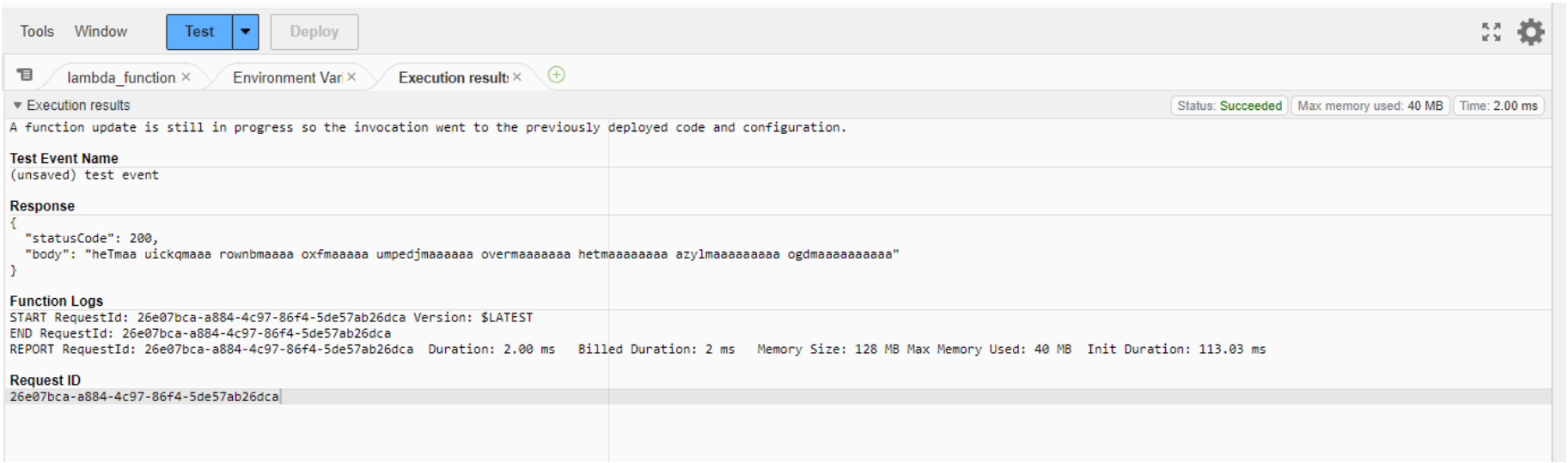
goatLatin /

lambda_function.py

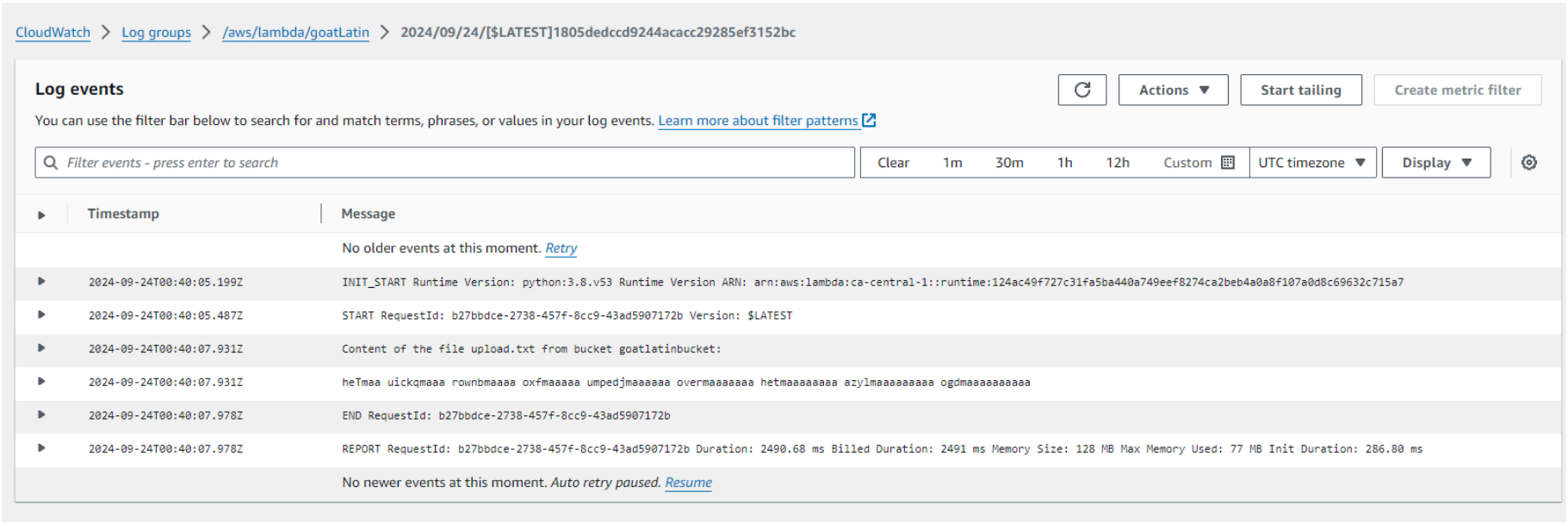
lambda_function × Environment Vari × Execution results ×

```python
import json
import boto3

def lambda_handler(event, context):
    def goat_latin(sentence):
        res = ""
        vowel="aeiouAEIOU"
        for idx, word in enumerate(sentence.split()):
            if word[0] not in vowel:
                word = word[1:] + word[0]
            word += "ma" + "a"*(idx+1)
            res += word + " "

        res = res.strip()
        print(res)
        return res

    s3 = boto3.client('s3')

    # Get bucket name and file key from the S3 event
    bucket_name = event['Records'][0]['s3']['bucket']['name']
    file_key = event['Records'][0]['s3']['object']['key']

    # Get the file object from S3
    file_obj = s3.get_object(Bucket=bucket_name, Key=file_key)

    # Read the content of the file
    file_content = file_obj['Body'].read().decode('utf-8')

    print(f'Content of the file {file_key} from bucket {bucket_name}:')

    res = goat_latin(file_content)

    return {
        'statusCode': 200,
        'body': str(res)
    }
```

Execution result

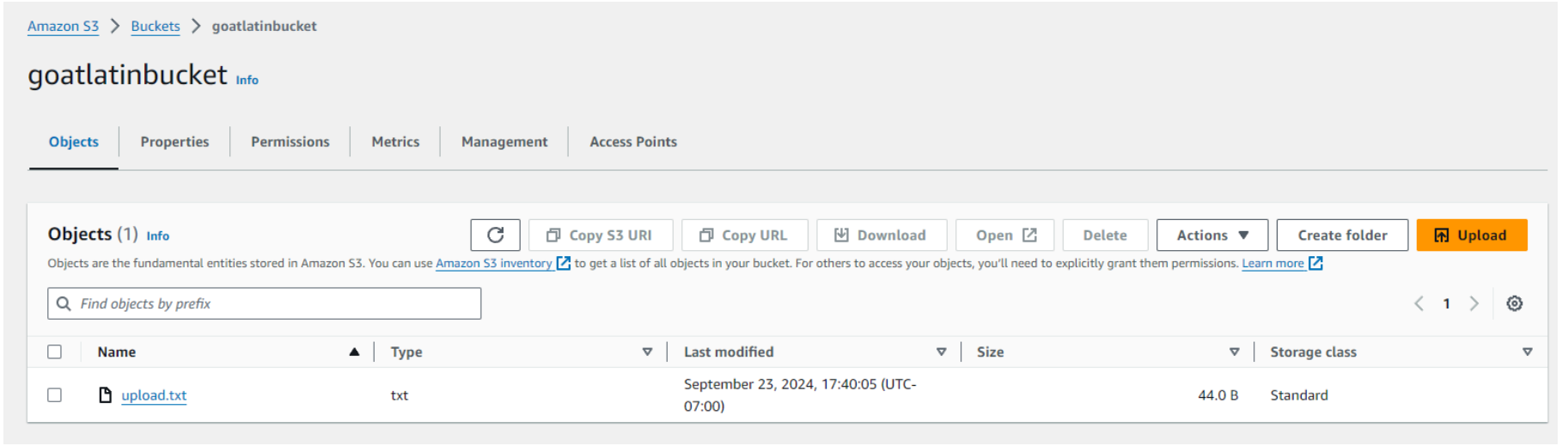


CloudWatch Log:



Trigger Event:

```python
import logging
import boto3
from botocore.exceptions import ClientError
import os
from dotenv import load_dotenv

load_dotenv(".env")

for key, value in os.environ.items():
    print(f"{key}: {value}")

def upload_file(file_name, bucket, object_name=None):
    # Verify env file is setup correctly
    print("AWS Access Key:", os.getenv("AWS_ACCESS_KEY_ID"))

    """Upload a file to an S3 bucket


    :param file_name: File to upload
    :param bucket: Bucket to upload to
    :param object_name: S3 object name. If not specified then file_name is used
    :return: True if file was uploaded, else False
    """
    # If S3 object_name was not specified, use file_name
    if object_name is None:
        object_name = os.path.basename(file_name)
    # Upload the file
    s3_client = boto3.client('s3', aws_access_key_id=os.getenv("AWS_ACCESS_KEY_ID"), aws_secret_access_key=os.getenv("AWS_SECRET_ACCESS_KEY"), aws_session_token=os.getenv
    ("AWS_SESSION_TOKEN"))
    try:
        response = s3_client.upload_file(file_name, bucket, object_name)
        print(response)
    except ClientError as e:
        logging.error(e)
        return False
    return True
upload_file('upload.txt', "goatlatinbucket" , 'upload.txt')
```


Amazon S3 > Buckets > goatlatinbucket
goatlatinbucket Info
Objects
Properties
Permissions
Metrics
Management
Access Points
Objects (1) Info
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Find objects by prefix
1
Name
Type
Last modified
Size
Storage class
upload.txt
txt
September 23, 2024, 17:40:05 (UTC-07:00)
44.0 B
Standard