SENG3011 - Y_2 - D2 Report

Shantanu Kulkarni

Contents

API Design Details
API Implementation

API Design Details

From an architectural standpoint, the goal was to harness the power of managed public cloud services to build an API which is lightweight, fully managed and performant. To do this, a micro-services architecture along with server-less computing and GraphQL was used to architect a solution which is decoupled, fully modularised and fully programmable.

Design Decisions

- **Database**
    - Choice
        - The choice of database is crucial for any API since it governs how data is stored, aggregated and queried in the back-end. By opting for a NoSQL database, ease of development is improved, as a schema is not required. Moreover, the scraper parses data into the known format of `articles` and `reports` in json, which directly map to a simple object store. This way, time is not wasted creating/querying/updating tables, as once an `article` or `report` structure is generated as a python dictionary, a simple `json.dumps()` can be executed to convert into json and directly put into the NoSQL database. Given the ease of integration when remaining AWS-native, **DynamoDB** seemed like a natural choice for a fully managed NoSQL database. The boto3 python sdk for AWS makes programmatic access to DynamoDB simple, with easy API calls such as `put_item()`.
        - Advantages
            - Fully managed service - no setup/maintenance required
            - No schema required
            - No parsing of data into tables
            - No knowledge of SQL required
        - Disadvantages
            - Inefficient querying
    - Challenges and findings
        - This is the first time using DynamoDB in a large scale project, and it has quite a lot of functionality which can be difficult to initially digest. For example, you can only query on `primary-key`, requiring the use of "Global Secondary Indexes" for more complex queries. As a workaround, querying on object attributes can be achieved with the `Scan()` boto3 API call, with a `FilterExpression` defined on an attribute. This works, however it is obviously inefficient since every `Scan()` operation

involves going over the entire table, making it unsuitable for larger data-sets. This can be seen in the following example:

```
table_reports.scan(FilterExpression=Attr("event_date").eq(date))
```

where `event_date` is an object attribute to be filtered on, and `table_reports` is the table containing all disease reports.

- **Compute**
  - Choice
    - Choosing compute can be at-times difficult, since it may involve setup, maintenance and extra cost. The decision to go server-less was motivated by the desire to abstract away as much responsibility/management of compute/networking infrastructure, allowing time to be maximised spent on writing code and not maintaining server infrastructure. AWS Lambda is an excellent server-less function-as-a-service compute option, which allows code to be directly run on an endpoint. It is then billed according to execution time i.e when the endpoint is actually hit, not for every second it is 'up' in the case of a traditional server. Moreover, the ability to attach Lambda functions as resolvers to the AWS AppSync API made developing the core API functionality very easy. Queries to the API endpoint invoke a single GraphQL query, which has a single Lambda function attached to it as a resolver (back-end).
    - Advantages
      - Speed of development
      - Cost efficiency
      - Seamless integration with other AWS services, particularly AppSync
      - No management of server infrastructure required at all - pure code
    - Disadvantages
      - Somewhat annoying debugging
  - Challenges and findings
    - Debugging is obviously not as easy as it would be when running code on a server, since there is no local machine or environment. However, **Amazon CloudWatch Events** logs all executions of a Lambda, making for some kind of debugging if required.
    - Unclear documentation at times made for frustrating debugging - i.e whether or not the `event_handler` python Lambda function is intended to return json or python object, or how the input parameters are stored in the `event` and `context` objects passed to the `event_handler` function as arguments when the Lambda is invoked.
- **Back-end**
  - Choice
    - As outlined in the D1 report, Golang was to be the programming language of choice for how the API back-end was implemented, however python3 was opted for in the end due to ease of development. The majority of group members are unfamiliar with Golang, therefore making its use unsuitable. I would still however like to re-implement the Lambda function that exists in Golang using the AWS sdk-for-go.
    - Advantages

            - Well-known and easy to develop in
            - boto3's popularity for developing using AWS makes for many blogs, stackoverflow threads, documentation and general support for different use-cases/issues
        - Disadvantages
            - Dynamically typed and interpreted makes for run-time errors which would otherwise be caught by a compiled language such as Golang
            - Vastly slower than a compiled language such as Golang
    - Challenges and findings
        - All in all a good choice given the power of boto3, however performance could be vastly optimised using Golang.
- **Framework**
    - Choice
        - GraphQL is a natural choice for this project, as it allows for queries to be defined on a set of types, which is useful given the known data format of `articles` and `reports`. Since the data format for how things like articles and reports is specified in the spec, it makes defining a GraphQL schema simple. AWS AppSync is a fully managed GraphQL service, completed with a querying tool in browser and an IDE for the schema and resolvers.
        - Advantages
            - Eliminates over-fetching or under-fetching problems thereby reducing network overhead
            - Reduced complexity in front-end development the development
            - AWS-native with seamless integration with AWS Lambda
            - Fully managed service - endpoint is setup and deployed already with customisable authentication - no code required for this
            - Write richer queries
        - Disadvantages
            - Added complexity in setting up a schema
    - Challenges and findings
        - AWS AppSync can be quite difficult to work with. Built-in resolver functions for queries on your schema are defined in VTL (Apache Velocity Templating Language) which is very complicated. Only until we figured out how to attach a Lambda function as a query resolver could we then begin debugging and testing the actual GraphQL endpoint. After this initial learning curve however things became more efficient, as doing things such as changing response content/fields was as easy as changing the return statement in the Lambda function, and then modifying the GraphQL schema accordingly. Also generating a schema from a DynamoDB table may seem like an easy option, but results in a range of types, mutations and queries being defined in your schema which are difficult to understand. This aspect of AppSync and GraphQL in general is the main drawback - learning and working with this idea of a schema and queries defined on types within the schema.
- **Machine Learning**
    - Choice
        - Given that fields such as `diseases` and `syndromes` can be fairly easily extracted from a given `article` `main_text` , NLP only really needs to be utilised for labelling other entities, such as dates and locations. Ideally, an intelligent model would actually distinguish between the mentioning of the disease in an article and

the actual outbreak/reporting of one, as well as group different occurrences of diseases as different reports within the same article. This is all rather difficult however, so is left as an exercise beyond this milestone, perhaps with the use of Amazon SageMaker and other Machine Learning services. Amazon Comprehend however can be quite effectively utilised for entity classification. This is done using the 'real-time-analysis' feature, which given an input text generates an entity report, labelling different words into types. A sample output of a entity report returned by Comprehend can be shown below:

```
{
"Entities": [
{
    "Score": 0.9985879063606262,
    "Type": "ORGANIZATION",
    "Text": "INOVIO",
    "BeginOffset": 23,
    "EndOffset": 29
},
{
    "Score": 0.9479143619537354,
    "Type": "LOCATION",
    "Text": "Ghana",
    "BeginOffset": 45,
    "EndOffset": 62
},
...
```

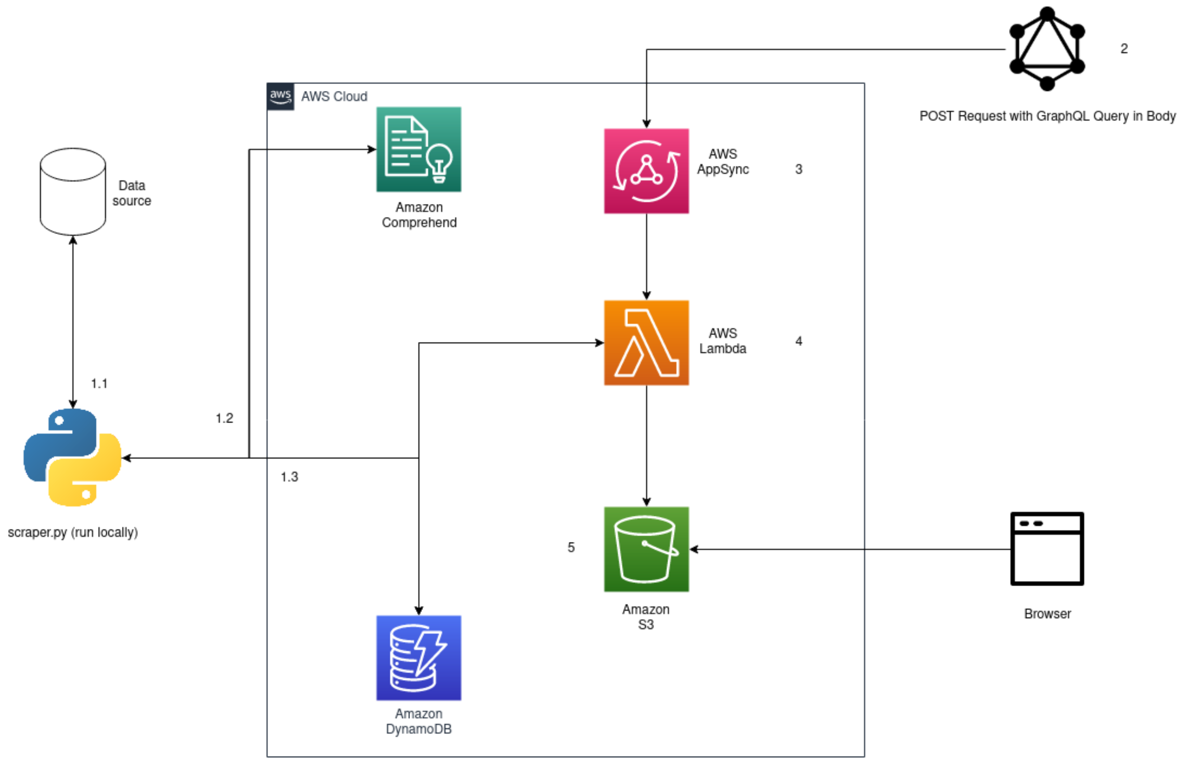
The "Score" indicates Comprehend's confidence in a particular classification, which is useful as we can sort by this field when generate reports to get the "highest confidence" location.

        - Advantages
            - Easy API call through boto3
            - Surprisingly cost efficient - free tier available for Amazon Comprehend
        - Disadvantages
            - Not yet very accurate
    - Challenges and findings
        - Comprehend also gives the ability to train custom classifiers, which we attempted with a list of 1000 diseases. The results were not satisfactory however, so I will be trying with a larger data-set of 17000 diseases which I found, along with a larger input text (validation data set). This also may however prove to incur cost.
- Tooling/DevOps/Misc
    - Amazon S3 was used for simple storage, i.e writing API call output files to a publically accessible bucket. I then inserted an `index.html` containing javascript which along with a permissive policy on the bucket, as well as a CORS policy makes for a website (directory listing) for the bucket.
    - Postman was used to document and test the API, creating a shareable Collection complete with a test suite and clear layout. This is fantastic as running the Collection can be automated, allowing for this to be injected into a CI/CD pipeline such as Travis CI or Jenkins, perhaps prior to a deployment stage. I would like to explore this, and CI/CD for Lambda in general, as this would make further streamline the Lambda development process.

API Implementation

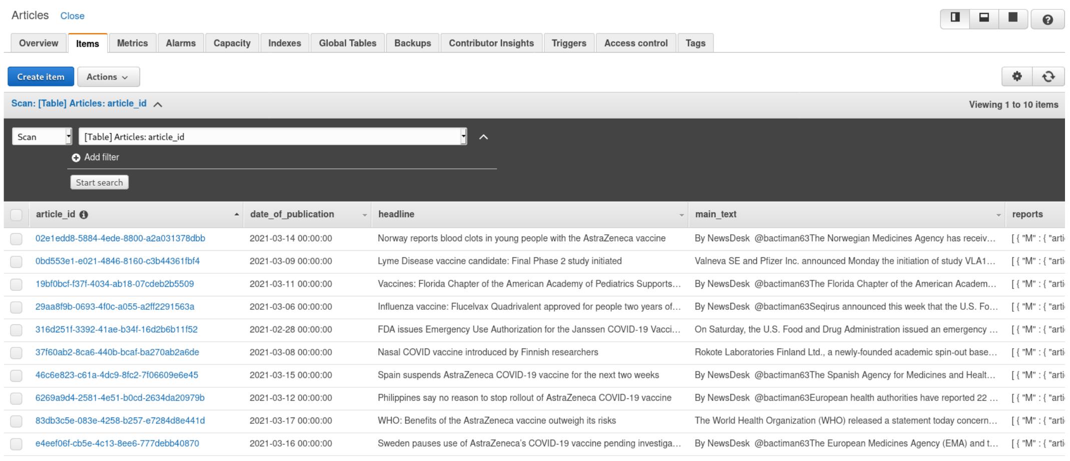
Architecture and implementation details

Our API utilises a broad range of managed services on Amazon Web Services to build an endpoint which can be queried after the scraper program is run. The flow can be summarised as such:



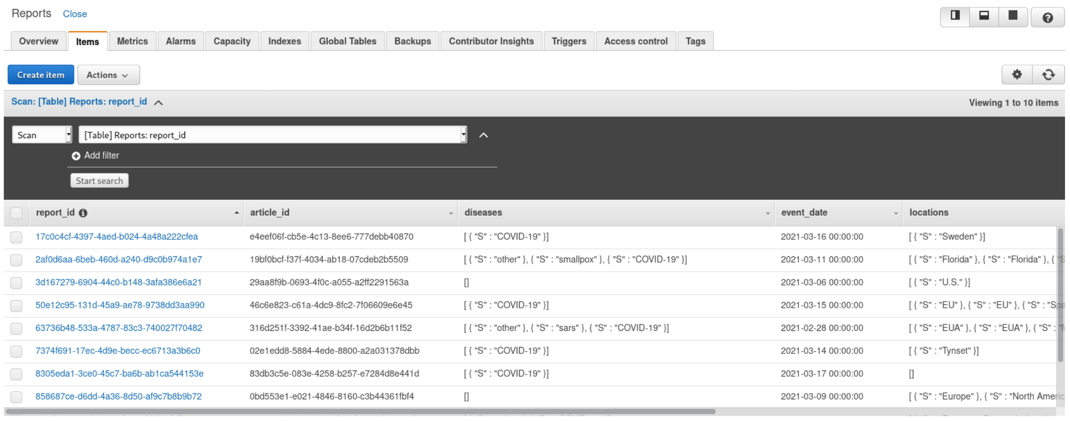
1. The initially flow on the left details how the database is intialised to be queried by the API.
    1. Run `scraper.py`. Currently this is done locally with `python3 scraper.py` but in the future will sit in it's own **AWS Lambda** function with a **CloudWatch Events Rate Expression** such as `rate(1 day)` which will invoke the Lambda function every day. The scraper scrapes the given data source of `http://outbreaknewstoday.com/` . Using the `BeautifulSoup` library, article objects are generated as python dictionaries in the desired format, with fields such as `date_of_publication`, `headline` and so forth being populated. In addition to these fields, we also add an `article-id` which is a uniquely generated `uuid` from a `uuid4()` call. Reports are stored as a children objects under the key `reports` in an article, with each report containing it's own unique `report-id` as well as the `article-id` of it's parent article.
    2. The scraper then begins to intelligently generate disease reports. This involves querying AWS's managed Natural Language Processing Service **Amazon Comprehend**. This and all programmatic access to AWS resources is done through the AWS python SDK **boto3**. The API call to the DetectEntities endpoint sends the `main_text` of a given `article` in the request, and returns a list of classified entities sorted by type and score (confidence) that the Amazon Comprehend real-time-analysis found in the article. From this, fields such as `location` in a report can be populated.
    3. The article object is now fully populated, complete with children `report` objects. This is then serialized into json with `json.dumps()` and then written to the `Articles` table in DynamoDB, with reports also being written but to a separate `Reports` table. Below are screenshots of both tables as well as sample objects within them. Note the `article-id`

field present in every report, associating it with it's parent report for retrieval when resolving a request.



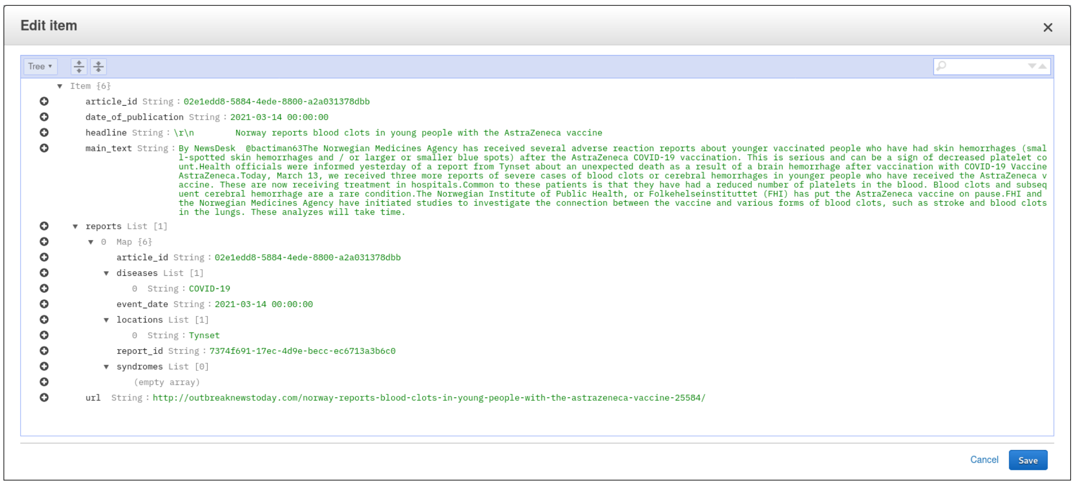
Articles Close
Overview Items Metrics Alarms Capacity Indexes Global Tables Backups Contributor Insights Triggers Access control Tags
Create item
Actions
Scan: [Table] Articles: article_id
Viewing 1 to 10 items
Scan
[Table] Articles: article_id
Add filter
Start search

article_id	date_of_publication	headline	main_text	reports
02e1edd8-5884-4ede-8800-a2a031378dbb	2021-03-14 00:00:00	Norway reports blood clots in young people with the AstraZeneca vaccine	By NewsDesk @bactiman63The Norwegian Medicines Agency has receiv…	[{ "M" : { "arti
0bd553e1-e021-4846-8160-c3b44361fbf4	2021-03-09 00:00:00	Lyme Disease vaccine candidate: Final Phase 2 study initiated	Valneva SE and Pfizer Inc. announced Monday the initiation of study VLA1…	[{ "M" : { "arti
19bf0bcf-f37f-4034-ab18-07cdeb2b5509	2021-03-11 00:00:00	Vaccines: Florida Chapter of the American Academy of Pediatrics Supports…	By NewsDesk @bactiman63The Florida Chapter of the American Academ…	[{ "M" : { "arti
29aa8f9b-0693-4f0c-a055-a2ff2291563a	2021-03-06 00:00:00	Influenza vaccine: Flucelvax Quadrivalent approved for people two years of…	By NewsDesk @bactiman63Seqirus announced this week that the U.S. Fo…	[{ "M" : { "arti
316d251f-3392-41ae-b34f-16d2b6b11f52	2021-02-28 00:00:00	FDA issues Emergency Use Authorization for the Janssen COVID-19 Vacci…	On Saturday, the U.S. Food and Drug Administration issued an emergency …	[{ "M" : { "arti
37f60ab2-8ca6-440b-bcaf-ba270ab2a6de	2021-03-08 00:00:00	Nasal COVID vaccine introduced by Finnish researchers	Rokote Laboratories Finland Ltd., a newly-founded academic spin-out base…	[{ "M" : { "arti
46c6e823-c61a-4dc9-8fc2-7f06609e6e45	2021-03-15 00:00:00	Spain suspends AstraZeneca COVID-19 vaccine for the next two weeks	By NewsDesk @bactiman63The Spanish Agency for Medicines and Healt…	[{ "M" : { "arti
6269a9d4-2581-4e51-b0cd-2634da20979b	2021-03-12 00:00:00	Philippines say no reason to stop rollout of AstraZeneca COVID-19 vaccine	By NewsDesk @bactiman63European health authorities have reported 22 …	[{ "M" : { "arti
83db3c5e-083e-4258-b257-e7284d8e441d	2021-03-17 00:00:00	WHO: Benefits of the AstraZeneca vaccine outweigh its risks	The World Health Organization (WHO) released a statement today concern…	[{ "M" : { "arti
e4eef06f-cb5e-4c13-8ee6-777debb40870	2021-03-16 00:00:00	Sweden pauses use of AstraZeneca's COVID-19 vaccine pending investiga…	By NewsDesk @bactiman63The European Medicines Agency (EMA) and t…	[{ "M" : { "arti

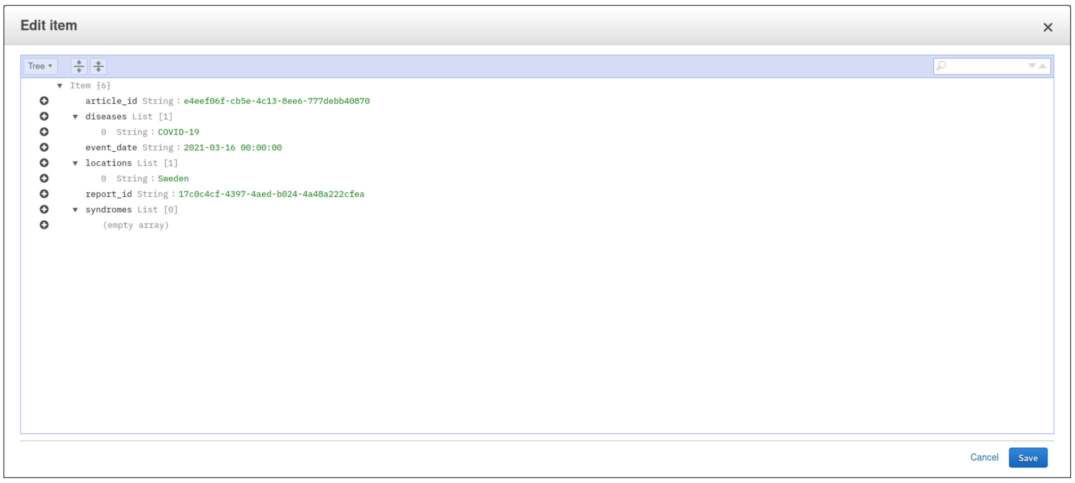


Reports Close
Overview Items Metrics Alarms Capacity Indexes Global Tables Backups Contributor Insights Triggers Access control Tags
Create item
Actions
Scan: [Table] Reports: report_id
Viewing 1 to 10 items
Scan
[Table] Reports: report_id
Add filter
Start search

report_id	article_id	diseases	event_date	locations
17c0c4cf-4397-4aed-b024-4a48a222cfea	e4eef06f-cb5e-4c13-8ee6-777debb40870	[{ "S" : "COVID-19" }]	2021-03-16 00:00:00	[{ "S" : "Sweden" }]
2af0d6aa-6beb-460d-a240-d9c0b974a1e7	19bf0bcf-f37f-4034-ab18-07cdeb2b5509	[{ "S" : "other" }, { "S" : "smallpox" }, { "S" : "COVID-19" }]	2021-03-11 00:00:00	[{ "S" : "Florida" }, { "S" : "Florida" }, { "S
3d167279-6904-44c0-b148-3afa386e6a21	29aa8f9b-0693-4f0c-a055-a2ff2291563a	[]	2021-03-06 00:00:00	[{ "S" : "U.S." }]
50e12c95-131d-45a9-ae78-9738dd3aa990	46c6e823-c61a-4dc9-8fc2-7f06609e6e45	[{ "S" : "COVID-19" }]	2021-03-15 00:00:00	[{ "S" : "EU" }, { "S" : "EU" }, { "S" : "Sp
63736b48-533a-4787-83c3-740027f70482	316d251f-3392-41ae-b34f-16d2b6b11f52	[{ "S" : "other" }, { "S" : "sars" }, { "S" : "COVID-19" }]	2021-02-28 00:00:00	[{ "S" : "EUA" }, { "S" : "EUA" }, { "S" : "N
7374f691-17ec-4d9e-becc-ec6713a3b6c0	02e1edd8-5884-4ede-8800-a2a031378dbb	[{ "S" : "COVID-19" }]	2021-03-14 00:00:00	[{ "S" : "Tynset" }]
8305eda1-3ce0-45c7-ba6b-ab1ca544153e	83db3c5e-083e-4258-b257-e7284d8e441d	[{ "S" : "COVID-19" }]	2021-03-17 00:00:00	[]
858687ce-d6dd-4a36-8d50-af9c7b8b9b72	0bd553e1-e021-4846-8160-c3b44361fbf4	[]	2021-03-09 00:00:00	[{ "S" : "Europe" }, { "S" : "North Americ



```
▼ Item {6}
    article_id String : 02e1edd8-5884-4ede-8800-a2a031378dbb
    date_of_publication String : 2021-03-14 00:00:00
    headline String : \r\n        Norway reports blood clots in young people with the AstraZeneca vaccine
    main_text String : By NewsDesk  @bactiman63The Norwegian Medicines Agency has received several adverse reaction reports about younger vaccinated people who have had skin hemorrhages (smal
                       l-spotted skin hemorrhages and / or larger or smaller blue spots) after the AstraZeneca COVID-19 vaccination. This is serious and can be a sign of decreased platelet co
                       unt.Health officials were informed yesterday of a report from Tynset about an unexpected death as a result of a brain hemorrhage after vaccination with COVID-19 Vaccine
                       AstraZeneca.Today, March 13, we received three more reports of severe cases of blood clots or cerebral hemorrhages in younger people who have received the AstraZeneca v
                       accine. These are now receiving treatment in hospitals.Common to these patients is that they have had a reduced number of platelets in the blood. Blood clots and subseq
                       uent cerebral hemorrhage are a rare condition.The Norwegian Institute of Public Health, or Folkehelseinstituttet (FHI) has put the AstraZeneca vaccine on pause.FHI and
                       the Norwegian Medicines Agency have initiated studies to investigate the connection between the vaccine and various forms of blood clots, such as stroke and blood clots
                       in the lungs. These analyzes will take time.
  ▼ reports List [1]
    ▼ 0  Map {6}
        article_id String : 02e1edd8-5884-4ede-8800-a2a031378dbb
      ▼ diseases List [1]
          0  String : COVID-19
        event_date String : 2021-03-14 00:00:00
      ▼ locations List [1]
          0  String : Tynset
        report_id String : 7374f691-17ec-4d9e-becc-ec6713a3b6c0
      ▼ syndromes List [0]
          (empty array)
    url  String : http://outbreaknewstoday.com/norway-reports-blood-clots-in-young-people-with-the-astrazeneca-vaccine-25584/
```



```
▼ Item {6}
    article_id String : e4eef06f-cb5e-4c13-8ee6-777debb40870
  ▼ diseases List [1]
      0  String : COVID-19
    event_date String : 2021-03-16 00:00:00
  ▼ locations List [1]
      0  String : Sweden
    report_id String : 17c0c4cf-4397-4aed-b024-4a48a222cfea
  ▼ syndromes List [0]
      (empty array)
```

2. Now that the DynamoDB table is populated, a user can now query the API endpoint to make a request. To resolve requests to the endpoint, we have described a single GraphQL query type called `getArticles()` , which takes in a combination of `period_of_interest`, `location` and `key_terms` . Currently, the API only supports date ranges and not singular/exact dates. `period_of_interest` is a required parameter, but `location` and `key_terms` are not.
3. Below is the GraphQL schema defining the various types and queries.

```
type Articles {
    url: String!
    headline: String!
    main_text: String!
    reports: [Reports]!
    article_id: String!
    date_of_publication: String!
}

type Mutation {
    putArticle(url: String!, headline: String!): Articles
}

type Query {
    getArticles(period_of_interest: String, key_terms: String, location:
String): Response!
}

type Reports {
    article_id: String!
    report_id: String!
    diseases: [String]!
    syndromes: [String]!
    locations: [String]!
    event_date: String!
}

type Response {
    body: [Articles]
    statusCode: Int!
    statusMessage: String!
}

type Schema {
    query: Query
    mutation: Mutation
}
```

`putArticle()` is on defined as a placeholder since the API currently has no mutation/write requirements. Across the whole schema `!` denotes compulsory for a field or parameter. As seen in the `getArticles()` definition, none of the input fields are compulsory as parameters, as this logic is handled by the back-end. This is because making query parameters required results in poor error messages being sent to the user upon a request where a required parameter is not given. The `Reponse` type is defined for easy return statements in the Lambda function resolver, with the `body` field being a nested type containing a list of `Articles`. This maps very nicely to python as evident in the `lambda.py` source-code where return statements are simply:

```
return {
    "body": results,
    "statusCode": 200,
    "statusMessage": "Query successful."
}
```

For error responses or responses with no body, body can simply be left empty or not at all returned:

```
return {
    "statusCode": 200,
    "statusMessage": "No articles matching the given query."
}
```

This seamless integration of services is extremely powerful and a main advantage of using GraphQL and particularly AWS AppSync. All data from the scraper, to the database, and finally to the GraphQL schema/Lambda function are serialized in the exact same way.

All endpoints are `POST`, with the user inputting the GraphQL query in the body of the request. After specifying the input parameters in the query, the user can choose to specify which fields it wishes to retrieve, whether that is all of them:

```
query {
  getArticles(key_terms: "COVID-19", location: "United States",
period_of_interest: "2021-01-01 17:00:xx to 2021-03-22 00:00:00") {
    statusMessage
    body {
      article_id
      date_of_publication
      main_text
      headline
      url
      reports {
        article_id
        event_date
        diseases
        locations
        report_id
        syndromes
      }
    }
    statusCode
  }
}
```

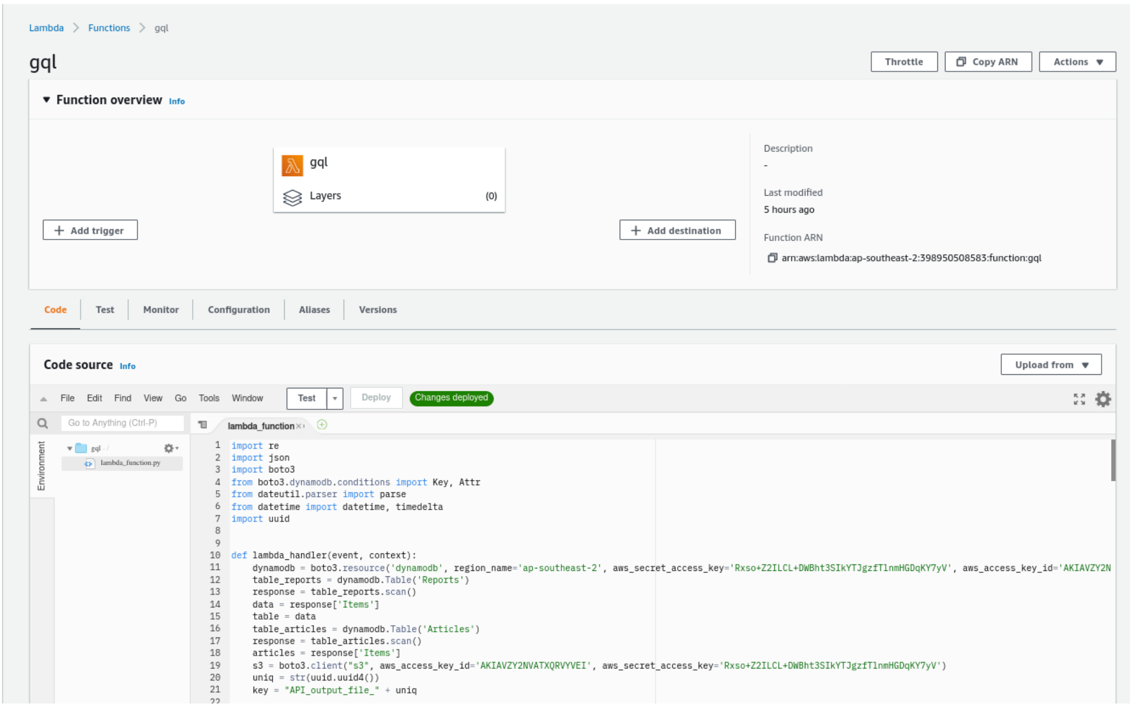
```
{
    "data": {
        "getArticles": {
            "statusMessage": "Query successful",
            "body": [
                {
                    "article_id": "316d251f-3392-41ae-b34f-16d2b6b11f52",
                    "date_of_publication": "2021-02-28 00:00:00",
                    "main_text": "On Saturday, the U.S. Food and Drug
Administration issued an emergency use authorization (EUA) for the third
vaccine for the prevention of coronavirus disease 2019 (COVID-19) caused by
severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). The EUA allows
the Janssen COVID-19 Vaccine to be distributed in the U.S for use in
individuals 18 years of age and older.“The authorization of this vaccine
expands the availability of vaccines, the best medical prevention method for
COVID-19, to help us in the fight against this pandemic, which has claimed
over half a million lives in the United States,” said Acting FDA Commissioner
Janet Woodcock, M.D. “The FDA, through our open and transparent scientific
review process, has now authorized three COVID-19 vaccines with the urgency
called for during this pandemic, using the agency’s rigorous standards for
safety, effectiveness and manufacturing quality needed to support emergency
use eporting System (VAERS) for Janssen COVID-19 Vaccine: serious adverse
events, cases of Multisystem Inflammatory Syndrome and cases of COVID-19 that
result in hospitalization or death......",
                    "headline": "\r\n        FDA issues Emergency Use
Authorization for the Janssen COVID-19 Vaccine, the 3rd to date        ",
                    "url": "http://outbreaknewstoday.com/fda-issues-
emergency-use-authorization-for-the-janssen-covid-19-vaccine-the-3rd-to-date-
76713/",
                    "reports": [
                        {
                            "article_id": "316d251f-3392-41ae-b34f-
16d2b6b11f52",
                            "event_date": "2021-02-28 00:00:00",
                            "diseases": [
                                "other",
                                "sars",
                                "COVID-19"
                            ],
                            "locations": [
                                "EUA",
                                "EUA",
                                "Mexico",
                                "U.S",
                                "U.S.",
                                "U.S.",
                                "United States",
                                "Mexico",
                                "South America",
                                "South America",
                                "South Africa",
                                "South Africa"
                            ],
                            "report_id": "63736b48-533a-4787-83c3-
740027f70482",
                            "syndromes": [
                                "Acute respiratory syndrome"
                            ]
                        }
                    ]
                }
            ],
            "statusCode": 200
        }
    }
}
```

or simply a few:

```
query {
  getArticles(key_terms: "COVID-19", location: "United States",
period_of_interest: "2021-01-01 17:00:xx to 2021-03-22 00:00:00") {
    statusMessage
    body {
      date_of_publication
      headline
      url
      reports {
        event_date
        diseases
        locations
      }
    }
    statusCode
  }
}
```

```
{
    "data": {
        "getArticles": {
            "statusMessage": "Query successful",
            "body": [
                {
                    "date_of_publication": "2021-02-28 00:00:00",
                    "headline": "\r\n        FDA issues Emergency Use
Authorization for the Janssen COVID-19 Vaccine, the 3rd to date        ",
                    "url": "http://outbreaknewstoday.com/fda-issues-
emergency-use-authorization-for-the-janssen-covid-19-vaccine-the-3rd-to-date-
76713/",
                    "reports": [
                        {
                            "event_date": "2021-02-28 00:00:00",
                            "diseases": [
                                "other",
                                "sars",
                                "COVID-19"
                            ],
                            "locations": [
                                "EUA",
                                "EUA",
                                "Mexico",
                                "U.S",
                                "U.S.",
                                "U.S.",
                                "United States",
                                "Mexico",
                                "South America",
                                "South America",
                                "South Africa",
                                "South Africa"
                            ]
                        }
                    ]
                }
            ],
            "statusCode": 200
        }
    }
}
```

The resolver Lambda function "gql" is shown below on the console, as well as how it is "attached" to the `getArticles` query and configured in VTL to return data.



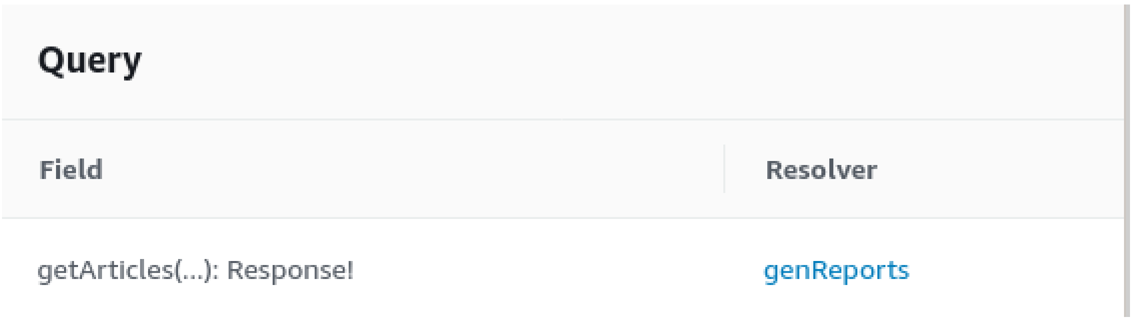
Description
-

Last modified
5 hours ago

Function ARN
arn:aws:lambda:ap-southeast-2:398950508583:function:gql

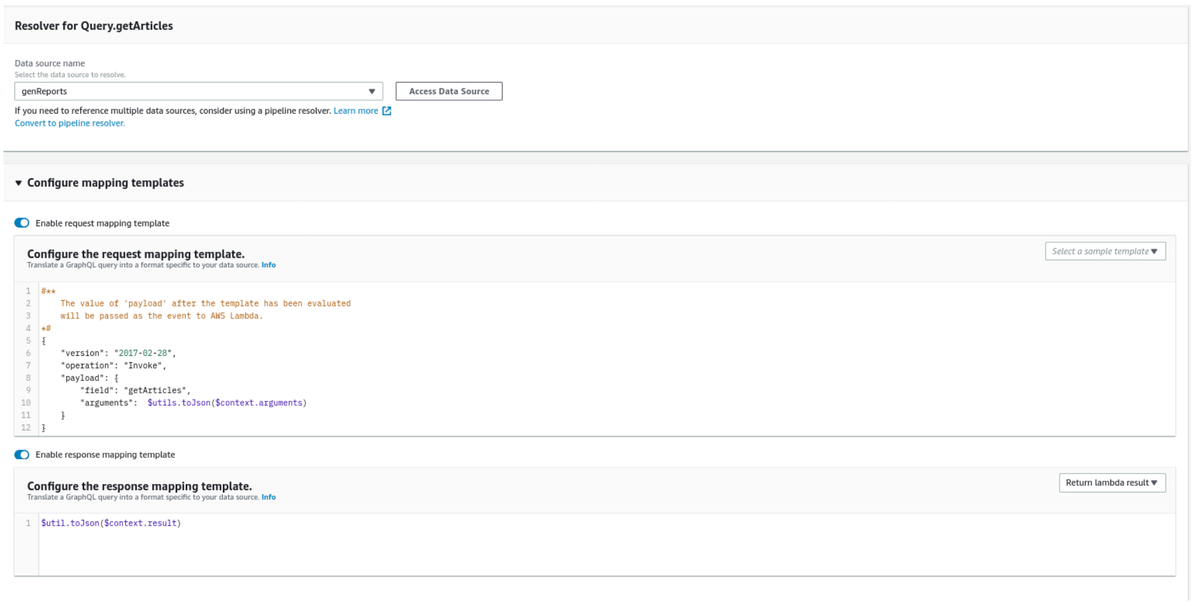
```
import re
import json
import boto3
from boto3.dynamodb.conditions import Key, Attr
from dateutil.parser import parse
from datetime import datetime, timedelta
import uuid


def lambda_handler(event, context):
    dynamodb = boto3.resource('dynamodb', region_name='ap-southeast-2', aws_secret_access_key='Rxso+Z2ILCL+DWBht3SIkYTJgzfTlnmHGDqKY7yV', aws_access_key_id='AKIAVZY2N
    table_reports = dynamodb.Table('Reports')
    response = table_reports.scan()
    data = response['Items']
    table = data
    table_articles = dynamodb.Table('Articles')
    response = table_articles.scan()
    articles = response['Items']
    s3 = boto3.client("s3", aws_access_key_id='AKIAVZY2NVATXQRVYVEI', aws_secret_access_key='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')
    uniq = str(uuid.uuid4())
    key = "API_output_file_" + uniq
```



Query

Field	Resolver
getArticles(...): Response!	genReports


Resolver for Query.getArticles

Data source name
Select the data source to resolve.
genReports

If you need to reference multiple data sources, consider using a pipeline resolver. Learn more
Convert to pipeline resolver.

Configure mapping templates

Enable request mapping template

Configure the request mapping template.
Translate a GraphQL query into a format specific to your data source. Info

```
#**
    The value of 'payload' after the template has been evaluated
    will be passed as the event to AWS Lambda.
*#
{
    "version": "2017-02-28",
    "operation": "Invoke",
    "payload": {
        "field": "getArticles",
        "arguments":  $utils.toJson($context.arguments)
    }
}
```

Enable response mapping template

Configure the response mapping template.
Translate a GraphQL query into a format specific to your data source. Info

```
$util.toJson($context.result)
```

4. The "gql" Lambda function can now resolve the the request. The `lambda_handler(event,context)` function is passed the query parameters in the `event` object as such:

```
{
    'field': 'getArticles',
    'arguments': {
        'period_of_interest': ['2021-01-01 17:00:xx to 2021-03-22 00:00:00'],
        'key_terms': 'COVID-19,sars',
        'location': 'Sweden'
    }
}
```

So query parameters like `location` can be accessed as such: `event['arguments']['location']`. The code then retrieves all entries in the `Reports` DynamoDB table. Then, it filters this table first by `period_of_interest`, then by `key_terms` or `location` depending on whether they are provided or not. This is done through various list comprehensions, minimising the number of queries made to the DynamoDB table, given that filtering on attributes like this: `table_reports.scan(FilterExpression=Attr("event_date").eq(date))` is performance expensive as it scans through the entire table. The results are returned in the previously specified format, with error messages for incorrectly formatted inputs (e.g bad date range) or no results for the given query.

In our current API , we tested for a variety of combination of inputs to check if they work correctly,

1. Check where the request provided contains all the fields and the format for all fields is correct. In this case the API returns a `200` response with articles as JSON objects
2. Check where only the `period_of_interest` and `key_terms` is provided in the request and the `period_of_interest` is provided in the correct format. The API returns a `200` response with the articles as JSON objects
3. Check where only the `period_of_interest` and `location` is provided in the request and the `period_of_interest` is provided in the correct format. The API returns a `200` response with the articles as JSON objects

4. Check where if `period_of_interest` field provided in the request does not follow the correct format, the API returns a `400` response with the status message indicating the error to user
5. Check where if `period_of_interest` field is not provided in the request , the API returns a `400` response with the status message indicating the error to user
6. Check for the case where the start date provided in the request is greater then the end date for the request, where the API returns a response with the status message indicating the error to the user
7. Check for the case where the request contains all the necessary fields in the correct format , but no articles match the request parameters. In this case the API returns a `200` response , but also the status message which indicates that the response is empty

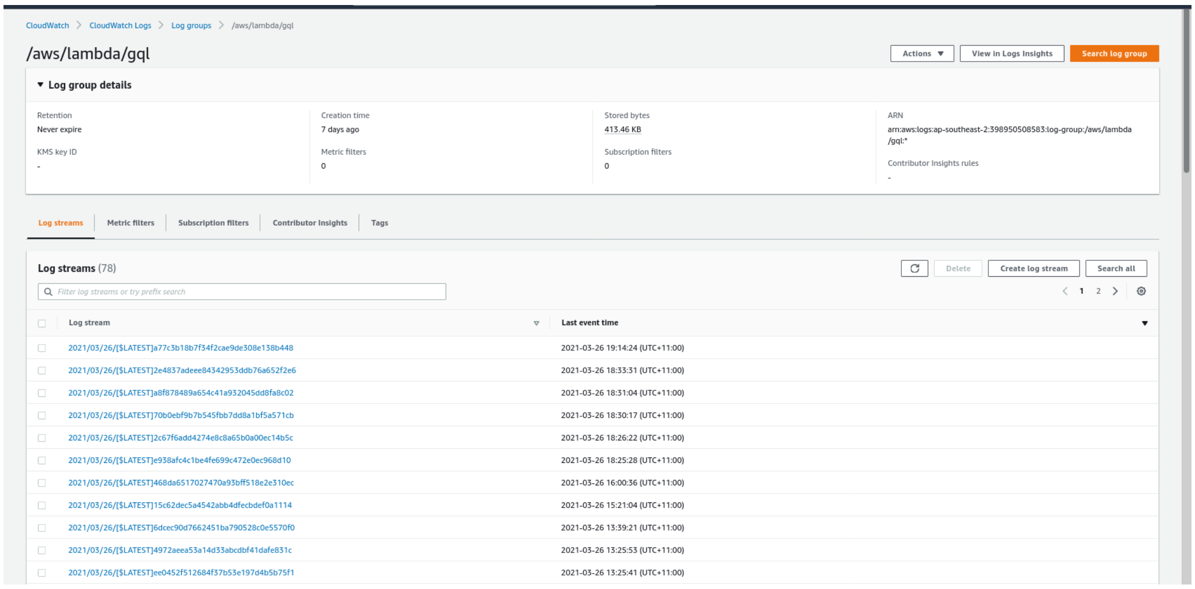
The above mentioned cases have been checked for and thoroughly verified. These use cases follow the specification given quite closely and takes care of most types of request.

There is one particular type of use case for which our API fails and doesn't work properly. Whenever an exact date is mentioned in the request instead of a date range , our API fails and returns an error. We plan on fixing this issue with our API on further releases , so our application can work for most cases.

This is detailed in the Postman Collection tests as API Documentation. Prior to returning the response message to the AppSync endpoint, the Lambda function will write the response message to a publicly accessible s3 bucket at https://y-2.s3-ap-southeast-2.amazonaws.com/index.html, appending a unique `uuid` to the output file name. This is done through the s3 boto3 client, with an API call as such:

```
uniq = str(uuid.uuid4())
key = "API_output_file_" + uniq
response = s3.put_object(
            Bucket='y-2',
            Key=key,
            Body=str(ret),
            ACL='public-read'
         )
```

Logs for the Lambda function's execution (for debugging purposes) can be viewed in CloudWatch Logs:



/aws/lambda/gql

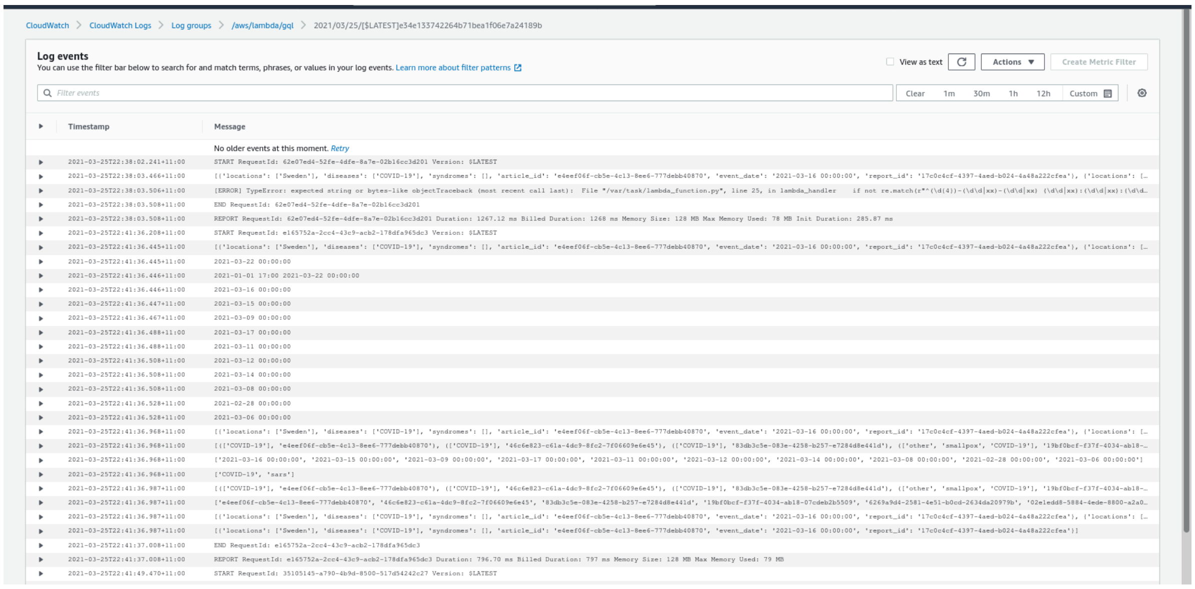
Log group details

Retention Never expire	Creation time 7 days ago	Stored bytes 413.46 KB	ARN arn:aws:logs:ap-southeast-2:398950508583:log-group:/aws/lambda/gql:*
KMS key ID -	Metric filters 0	Subscription filters 0	Contributor Insights rules -

Log streams (78)

Log stream	Last event time
2021/03/26/[$LATEST]a77c3b18b7f34f2cae9de308e138b448	2021-03-26 19:14:24 (UTC+11:00)
2021/03/26/[$LATEST]2e4837adeee84342953ddb76a652f2e6	2021-03-26 18:33:31 (UTC+11:00)
2021/03/26/[$LATEST]a8f878489a654c41a932045dd8fa8c02	2021-03-26 18:31:04 (UTC+11:00)
2021/03/26/[$LATEST]70b0ebf9b7b545fbb7dd8a1bf5a571cb	2021-03-26 18:30:17 (UTC+11:00)
2021/03/26/[$LATEST]2c67f6add4274e8c8a65b0a00ec14b5c	2021-03-26 18:26:22 (UTC+11:00)
2021/03/26/[$LATEST]e938afc4c1be4fe699c472e0ec968d10	2021-03-26 18:25:28 (UTC+11:00)
2021/03/26/[$LATEST]468da6517027470a93bff518e2e310ec	2021-03-26 16:00:36 (UTC+11:00)
2021/03/26/[$LATEST]15c62dec5a4542abb4dfecbdef0a1114	2021-03-26 15:21:04 (UTC+11:00)
2021/03/26/[$LATEST]6dcec90d7662451ba790528c0e5570f0	2021-03-26 13:39:21 (UTC+11:00)
2021/03/26/[$LATEST]4972aeea53a14d33abcdbf41dafe831c	2021-03-26 13:25:53 (UTC+11:00)
2021/03/26/[$LATEST]ee0452f512684f37b53e197d4b5b75f1	2021-03-26 13:25:41 (UTC+11:00)

With data such as execution time, input parameters and debug output (e.g things from stdout) available:


CloudWatch > CloudWatch Logs > Log groups > /aws/lambda/gql > 2021/03/25/[$LATEST]e34e133742264b71bea1f06e7a24189b
Log events
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
View as text
Actions
Create Metric Filter
Filter events
Clear
1m
30m
1h
12h
Custom
Timestamp
Message
No older events at this moment. Retry

```
2021-03-25T22:38:02.241+11:00  START RequestId: 62e07ed4-52fe-4dfe-8a7e-02b16cc3d201 Version: $LATEST
2021-03-25T22:38:03.466+11:00  [{'locations': ['Sweden'], 'diseases': ['COVID-19'], 'syndromes': [], 'article_id': 'e4eef06f-cb5e-4c13-8ee6-777debb40870', 'event_date': '2021-03-16 00:00:00', 'report_id': '17c0c4cf-4397-4aed-b024-4a48a222cfea'}, {'locations': [...
2021-03-25T22:38:03.506+11:00  [ERROR] TypeError: expected string or bytes-like objectTraceback (most recent call last):  File "/var/task/lambda_function.py", line 25, in lambda_handler    if not re.match(r"^(\d{4})-(\d\d|xx)-(\d\d|xx) (\d\d|xx):(\d\d|xx):(\d\d...
2021-03-25T22:38:03.508+11:00  END RequestId: 62e07ed4-52fe-4dfe-8a7e-02b16cc3d201
2021-03-25T22:38:03.508+11:00  REPORT RequestId: 62e07ed4-52fe-4dfe-8a7e-02b16cc3d201 Duration: 1267.12 ms Billed Duration: 1268 ms Memory Size: 128 MB Max Memory Used: 78 MB Init Duration: 285.87 ms
2021-03-25T22:41:36.208+11:00  START RequestId: e165752a-2cc4-43c9-acb2-178dfa965dc3 Version: $LATEST
2021-03-25T22:41:36.445+11:00  [{'locations': ['Sweden'], 'diseases': ['COVID-19'], 'syndromes': [], 'article_id': 'e4eef06f-cb5e-4c13-8ee6-777debb40870', 'event_date': '2021-03-16 00:00:00', 'report_id': '17c0c4cf-4397-4aed-b024-4a48a222cfea'}, {'locations': [...
2021-03-25T22:41:36.445+11:00  2021-03-22 00:00:00
2021-03-25T22:41:36.446+11:00  2021-01-01 17:00 2021-03-22 00:00:00
2021-03-25T22:41:36.446+11:00  2021-03-16 00:00:00
2021-03-25T22:41:36.447+11:00  2021-03-15 00:00:00
2021-03-25T22:41:36.467+11:00  2021-03-09 00:00:00
2021-03-25T22:41:36.488+11:00  2021-03-17 00:00:00
2021-03-25T22:41:36.488+11:00  2021-03-11 00:00:00
2021-03-25T22:41:36.508+11:00  2021-03-12 00:00:00
2021-03-25T22:41:36.508+11:00  2021-03-14 00:00:00
2021-03-25T22:41:36.508+11:00  2021-03-08 00:00:00
2021-03-25T22:41:36.528+11:00  2021-02-28 00:00:00
2021-03-25T22:41:36.528+11:00  2021-03-06 00:00:00
2021-03-25T22:41:36.968+11:00  [{'locations': ['Sweden'], 'diseases': ['COVID-19'], 'syndromes': [], 'article_id': 'e4eef06f-cb5e-4c13-8ee6-777debb40870', 'event_date': '2021-03-16 00:00:00', 'report_id': '17c0c4cf-4397-4aed-b024-4a48a222cfea'}, {'locations': [...
2021-03-25T22:41:36.968+11:00  [(['COVID-19'], 'e4eef06f-cb5e-4c13-8ee6-777debb40870'), (['COVID-19'], '46c6e823-c61a-4dc9-8fc2-7f06609e6e45'), (['COVID-19'], '83db3c5e-083e-4258-b257-e7284d8e441d'), (['other', 'smallpox', 'COVID-19'], '19bf0bcf-f37f-4034-ab18-...
2021-03-25T22:41:36.968+11:00  ['2021-03-16 00:00:00', '2021-03-15 00:00:00', '2021-03-09 00:00:00', '2021-03-17 00:00:00', '2021-03-11 00:00:00', '2021-03-12 00:00:00', '2021-03-14 00:00:00', '2021-03-08 00:00:00', '2021-02-28 00:00:00', '2021-03-06 00:00:00']
2021-03-25T22:41:36.968+11:00  ['COVID-19', 'sars']
2021-03-25T22:41:36.987+11:00  [(['COVID-19'], 'e4eef06f-cb5e-4c13-8ee6-777debb40870'), (['COVID-19'], '46c6e823-c61a-4dc9-8fc2-7f06609e6e45'), (['COVID-19'], '83db3c5e-083e-4258-b257-e7284d8e441d'), (['other', 'smallpox', 'COVID-19'], '19bf0bcf-f37f-4034-ab18-...
2021-03-25T22:41:36.987+11:00  ['e4eef06f-cb5e-4c13-8ee6-777debb40870', '46c6e823-c61a-4dc9-8fc2-7f06609e6e45', '83db3c5e-083e-4258-b257-e7284d8e441d', '19bf0bcf-f37f-4034-ab18-07cdeb2b5509', '6269a9d4-2581-4e51-b0cd-2634da20979b', '02e1edd8-5884-4ede-8800-a2a0...
2021-03-25T22:41:36.987+11:00  [{'locations': ['Sweden'], 'diseases': ['COVID-19'], 'syndromes': [], 'article_id': 'e4eef06f-cb5e-4c13-8ee6-777debb40870', 'event_date': '2021-03-16 00:00:00', 'report_id': '17c0c4cf-4397-4aed-b024-4a48a222cfea'}, {'locations': [...
2021-03-25T22:41:36.987+11:00  [{'locations': ['Sweden'], 'diseases': ['COVID-19'], 'syndromes': [], 'article_id': 'e4eef06f-cb5e-4c13-8ee6-777debb40870', 'event_date': '2021-03-16 00:00:00', 'report_id': '17c0c4cf-4397-4aed-b024-4a48a222cfea'}]
2021-03-25T22:41:37.008+11:00  END RequestId: e165752a-2cc4-43c9-acb2-178dfa965dc3
2021-03-25T22:41:37.008+11:00  REPORT RequestId: e165752a-2cc4-43c9-acb2-178dfa965dc3 Duration: 796.70 ms Billed Duration: 797 ms Memory Size: 128 MB Max Memory Used: 79 MB
2021-03-25T22:41:49.470+11:00  START RequestId: 35105145-a790-4b9d-8500-517d54242c27 Version: $LATEST
```

5. Finally, the output files from API calls are available on a publicly accessible s3 bucket at https://y-2.s3-ap-southeast-2.amazonaws.com/index.html, which contains an `index.html` file serves a simple directory listing of the bucket's contents.