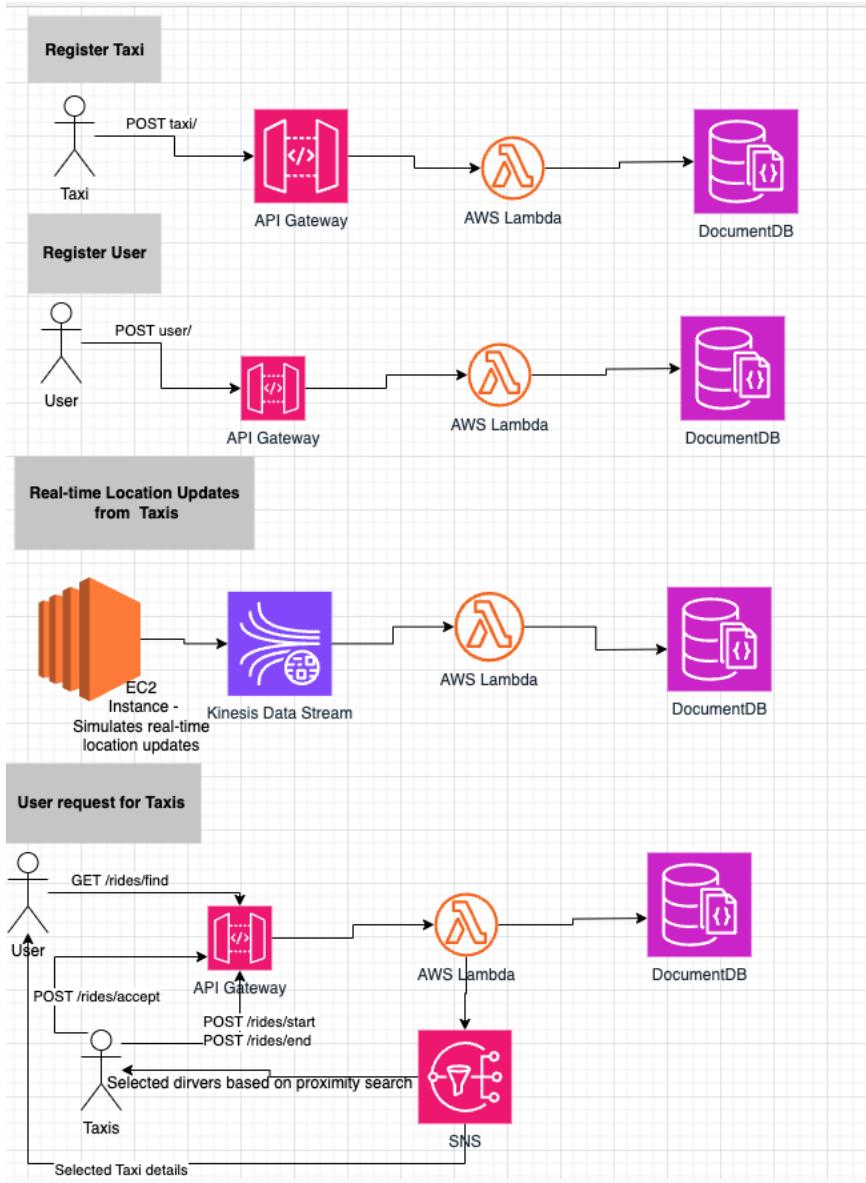


High level Architecture Design Diagram for the Location based Taxi Aggregator and Selector (Cloud)



Demo Presentation:

1. Delete existing DB if present

The screenshot shows the MongoDB Compass interface. The title bar reads "MongoDB Compass - ec2-3-143-22-60.us-east-2.compute.amazonaws.com:27018". The left sidebar has sections for "My Queries" and "Databases". Under "Databases", there is a search bar and a list of databases: "admin" (selected), "config", and "local". The main area displays the "admin" database details: Storage size, Collections, and Indexes. The "config" and "local" databases are also listed below.

2. Connect to Ec2 instance using below command from terminal

```
ssh -i "ec2cp2.pem" ubuntu@ec2-3-143-22-60.us-east-2.compute.amazonaws.com
```

```
[→ Downloads ssh -i "ec2cp2.pem" ubuntu@ec2-3-143-22-60.us-east-2.compute.amazonaws.com
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-1017-aws x86_64)
```

```
* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage
```

```
System information as of Sun Feb 25 21:54:56 UTC 2024
```

```
System load:  0.0          Processes:           111
Usage of /:   55.1% of 7.57GB  Users logged in:      0
Memory usage: 47%          IPv4 address for docker0: 172.17.0.1
Swap usage:   0%            IPv4 address for eth0:   172.31.9.133
```

```
* Ubuntu Pro delivers the most comprehensive open source security and
compliance features.
```

```
https://ubuntu.com/aws/pro
```

```
Expanded Security Maintenance for Applications is not enabled.
```

```
37 updates can be applied immediately.
To see these additional updates run: apt list --upgradable
```

```
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
```

```
*** System restart required ***
Last login: Sun Feb 25 20:41:34 2024 from 206.180.248.85
```

3. Run `InitializeTaxiAggregatorDB.py` to create DB, Collections and 2dsphere index and then insert default service area location.

```
[ubuntu@ip-172-31-9-133:~$ python3 InitializeTaxiAggregatorDB.py  
Created geospatial index for 'taxis' collection.  
Created geospatial index for 'customers' collection.  
Inserted default service area document into 'service_areas' collection.
```

4. Run RegisterTaxis.py to create 50 sample taxi data.

TaxiAggregator.taxis

50 DOCUMENTS 2 INDEXES

Documents Aggregations Schema Indexes Validation

Filter Type a query: { field: 'value' } Explain Reset Find Options ▾

+ ADD DATA EXPORT DATA 1 – 20 of 50

_id: ObjectId('65dbb8445e4eaedd47a33b74')
name: "Taxi_001"
type: "Luxury"
▶ location: Object

_id: ObjectId('65dbb8445e4eaedd47a33b75')
name: "Taxi_002"
type: "Luxury"
▶ location: Object

Luxury

_id: ObjectId('65dbb8445e4eaedd47a33b76')
name: "Taxi_003"
type: "Luxury"
▶ location: Object

_id: ObjectId('65dbb8445e4eaedd47a33b77')
name: "Taxi_004"
type: "Deluxe"
▶ location: Object

5. Run RegisterCustomers.py to create 5 sample Customer data.

```
[ubuntu@ip-172-31-9-133:~$ python3 RegisterCustomers.py
User Mohan registered successfully.
User Isaac registered successfully.
User Amir registered successfully.
User Ashok registered successfully.
User Leonard registered successfully.
```

TaxiAggregator.customers

5 DOCUMENTS 2 INDEXES

Documents Aggregations Schema Indexes Validation

Filter Type a query: { field: 'value' } Explain Reset Find Options ▾

+ ADD DATA EXPORT DATA 1 – 5 of 5

_id	name	location
ObjectId('65dbb8936eea60d19fc048e9')	"Mohan"	Object
ObjectId('65dbb8936eea60d19fc048eb')	"Isaac"	Object
ObjectId('65dbb8946eea60d19fc048ed')	"Amir"	Object
ObjectId('65dbb8946eea60d19fc048ef')	"Ashok"	Object
ObjectId('65dbb8946eea60d19fc048f1')	"Leonard"	Object

- Run TaxiLocationUpdateSimulator.py to send location data for different taxis at regular interval.

```
ubuntu@ip-172-31-9-133:~$ python TaxiLocationUpdateSimulator.py
Command 'python' not found, did you mean:
  command 'python3' from deb python3
  command 'python' from deb python-is-python3
[ubuntu@ip-172-31-9-133:~$ python3 TaxiLocationUpdateSimulator.py
Updating taxi Taxi_001 location to [77.97820711125355, 27.745265818907193]
Updating taxi Taxi_002 location to [77.58673984836355, 29.036456113994994]
Updating taxi Taxi_003 location to [76.49056633366784, 29.15019760064313]
Updating taxi Taxi_004 location to [77.34860418033428, 28.279482566945294]
Updating taxi Taxi_005 location to [76.2932339110117, 29.16262654214663]
Updating taxi Taxi_006 location to [78.03887525287256, 27.937017632738357]
Updating taxi Taxi_007 location to [76.5067643406846, 29.53966292281688]
Updating taxi Taxi_008 location to [76.60963986736485, 29.25120960128249]
Updating taxi Taxi_009 location to [76.27378677182081, 28.74841795335613]
Updating taxi Taxi_010 location to [77.6729285298968, 29.228750127313614]
Updating taxi Taxi_011 location to [77.18630057427484, 28.273989594099426]
Updating taxi Taxi_012 location to [76.7582422355116, 29.11730512483512]
Updating taxi Taxi_013 location to [77.65150779004566, 28.94162324128028]
Updating taxi Taxi_014 location to [77.1430552450643, 28.41919519291199]
Updating taxi Taxi_015 location to [77.13949438787712, 27.71132429275118]
Updating taxi Taxi_016 location to [77.8221742203139, 28.547789017663344]
Updating taxi Taxi_017 location to [77.80635258785628, 28.340690896219602]
Updating taxi Taxi_018 location to [76.92775677226199, 28.37555756848389]
Updating taxi Taxi_019 location to [76.76726326004464, 29.661682736044046]
Updating taxi Taxi_020 location to [77.43013006008695, 27.68382980774894]
Updating taxi Taxi_021 location to [77.4599212428373, 29.20846476430362]
Updating taxi Taxi_022 location to [77.20481079357265, 28.291306090568813]
Updating taxi Taxi_023 location to [77.6593070298245, 27.804897034634187]
Updating taxi Taxi_024 location to [76.88866000107448, 28.55892573260616]
Updating taxi Taxi_025 location to [78.31958717440725, 29.30523524524064]
Updating taxi Taxi_026 location to [78.27846161509976, 28.206082320706695]
Updating taxi Taxi_027 location to [76.79206993507886, 29.00431274173009]
Updating taxi Taxi_028 location to [77.83791845540817, 27.89976416757519]
Updating taxi Taxi_029 location to [77.79681239743678, 29.042001349464964]
Updating taxi Taxi_030 location to [78.19261212172515, 28.677594172172498]
Updating taxi Taxi_031 location to [77.61830013739012, 27.681872096632883]
Updating taxi Taxi_032 location to [77.87954030670697, 28.9824522545026]
Updating taxi Taxi_033 location to [77.12537631213397, 29.02144432726105]
Updating taxi Taxi_034 location to [76.413064713702, 29.055894099526903]
Updating taxi Taxi_035 location to [77.75771004258876, 29.520653784004924]
Updating taxi Taxi_036 location to [77.67801713380423, 29.340496146232038]
Updating taxi Taxi_037 location to [77.3272154627392, 28.305751974157452]
Updating taxi Taxi_038 location to [77.19925350705347, 29.389673442167016]
Updating taxi Taxi_039 location to [77.36966218193953, 27.971463255903895]
Updating taxi Taxi_040 location to [77.01391052136786, 29.577280989223492]
Updating taxi Taxi_041 location to [76.6294208635961, 28.302307282563486]
Updating taxi Taxi_042 location to [76.9017701160669, 27.981077202250884]
Updating taxi Taxi_043 location to [77.85403226949143, 29.172042044802016]
Updating taxi Taxi_044 location to [78.19219678946521, 28.097096766523855]
Updating taxi Taxi_045 location to [77.05806143965104, 27.763078404572727]
Updating taxi Taxi_046 location to [77.88284875519064, 28.926064171078806]
Updating taxi Taxi_047 location to [76.91559323537476, 29.480095222630915]
Updating taxi Taxi_048 location to [77.5033752148808, 27.722281668988742]
Updating taxi Taxi_049 location to [77.72628514594892, 28.499725586722526]
Updating taxi Taxi_050 location to [77.64095957470933, 29.19474357593197]
Updated taxi locations at 2024-02-25 22:02:42
```

```

Updating taxi Taxi_001 location to [77.97372778543281, 27.744944606490023]
Updating taxi Taxi_002 location to [77.58873820204091, 29.027871233851766]
Updating taxi Taxi_003 location to [76.4822766988495, 29.150795822895]
Updating taxi Taxi_004 location to [77.34936944537577, 28.28435069250968]
Updating taxi Taxi_005 location to [76.2842864154463, 29.16381089109697]
Updating taxi Taxi_006 location to [78.03497099859592, 27.927321379525875]
Updating taxi Taxi_007 location to [76.50756974327204, 29.544378081597095]
Updating taxi Taxi_008 location to [76.61657786335856, 29.242242339582567]
Updating taxi Taxi_009 location to [76.27551126808359, 28.756792477992423]
Updating taxi Taxi_010 location to [77.67781705916757, 29.236220888871323]
Updating taxi Taxi_011 location to [77.1902621843461, 28.26722560925496]
Updating taxi Taxi_012 location to [76.75502940546608, 29.121903904295824]
Updating taxi Taxi_013 location to [77.64778281519139, 28.93836482299896]
Updating taxi Taxi_014 location to [77.15202760097463, 28.42315584840432]
Updating taxi Taxi_015 location to [77.14764195247777, 27.708180586957656]
Updating taxi Taxi_016 location to [77.81688028029885, 28.542693495777513]
Updating taxi Taxi_017 location to [77.80089490360108, 28.331915573971198]
Updating taxi Taxi_018 location to [76.93062073364865, 28.373620489083788]
Updating taxi Taxi_019 location to [76.7607562575273, 29.65526848658319]
Updating taxi Taxi_020 location to [77.43805191385675, 27.680155525787566]
Updating taxi Taxi_021 location to [77.45264969472467, 29.21347102673014]
Updating taxi Taxi_022 location to [77.21390393747745, 28.299277310886065]
Updating taxi Taxi_023 location to [77.65291042273277, 27.809195697139035]
Updating taxi Taxi_024 location to [76.89372683906592, 28.568597094368766]
Updating taxi Taxi_025 location to [78.32729337010969, 29.30772157588192]
Updating taxi Taxi_026 location to [78.28197600993002, 28.20267698568134]
Updating taxi Taxi_027 location to [76.79604072861137, 29.012469713225045]
Updating taxi Taxi_028 location to [77.8339293623563, 27.896952317232763]
Updating taxi Taxi_029 location to [77.78958939293365, 29.044473255449613]
Updating taxi Taxi_030 location to [78.19080901654158, 28.6844723339578]
Updating taxi Taxi_031 location to [77.61435287966823, 27.67707125963407]
Updating taxi Taxi_032 location to [77.88411577177119, 28.972782752266838]
Updating taxi Taxi_033 location to [77.12907736389359, 29.01828997664848]
Updating taxi Taxi_034 location to [76.40542225977988, 29.06147032987683]
Updating taxi Taxi_035 location to [77.75271056662496, 29.515600883974084]
Updating taxi Taxi_036 location to [77.67419300349455, 29.35020626598568]
Updating taxi Taxi_037 location to [77.31909470338209, 28.303690398486463]
Updating taxi Taxi_038 location to [77.19283686587157, 29.394882462303222]
Updating taxi Taxi_039 location to [77.37648970004281, 27.9721571776232]
Updating taxi Taxi_040 location to [77.01810602562762, 29.584837093654812]
Updating taxi Taxi_041 location to [76.62971404448729, 28.299047375315784]
Updating taxi Taxi_042 location to [76.8997084286199, 27.972430072440787]
Updating taxi Taxi_043 location to [77.8627018070846, 29.169022205480612]
Updating taxi Taxi_044 location to [78.2013120951041, 28.10539012592195]
Updating taxi Taxi_045 location to [77.05986373755377, 27.77092408938565]
Updating taxi Taxi_046 location to [77.87843559581597, 28.934623985351543]
Updating taxi Taxi_047 location to [76.92248887541562, 29.47936213617439]
Updating taxi Taxi_048 location to [77.50791015470277, 27.727112547820564]
Updating taxi Taxi_049 location to [77.7296887372189, 28.498669842212074]
Updating taxi Taxi_050 location to [77.64672177833478, 29.188779435778795]
Updated taxi locations at 2024-02-25 22:03:43
-----
```

7. Post below request in postman to find rides based on customer location proximity search. API -
<https://udmqfurkie.execute-api.us-east-2.amazonaws.com/dev/ride/find>

```
{
  "location": {
```

```

    "type": "Point",
    "coordinates": [77.23149, 28.61123]
},
"type": "Luxury"
}

```

ACSECapstone / Ride find

POST <https://udmqfurfie.execute-api.us-east-2.amazonaws.com/dev/ride/find>

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL **JSON** Beautify

```

1 {
2   "location": {
3     "type": "Point",
4     "coordinates": [77.23149, 28.61123]
5   },
6   "type": "Luxury"
7 }
8

```

Body Cookies Headers (7) Test Results [Save Response](#)

Pretty Raw Preview Visualize **JSON**

```

1 {
2   "statusCode": 200,
3   "headers": {
4     "Content-Type": "application/json"
5   },
6   "body": "[{"_id": "\\"65dbb8445e4eaedd47a33b81\\", "name": \"Taxi_014\", "type": \"Luxury\",
7     "location": {"type": "Point", "coordinates": [77.15628988904251, 28.42747545782674]}, {
8       "_id": "\\"65dbb8445e4eaedd47a33ba4\\", "name": \"Taxi_049\", "type": \"Luxury\",
9       "location": {"type": "Point", "coordinates": [77.72787385050535, 28.49286001162497]}"}]"
10 }
11

```

- Post below json request to create a taxi data using API - <https://udmqfurfie.execute-api.us-east-2.amazonaws.com/dev/taxi>

```

{
  "name": "Taxi_052",
  "type": "Luxury",
  "location": {
    "type": "Point",
    "coordinates": [
      78.3368142118244,
      27.66077927262116
    ]
  }
}

```

ACSECapstone / Taxi

POST <https://udmqfurkie.execute-api.us-east-2.amazonaws.com/dev/taxi> Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL **JSON** Beautify

```

1  {
2    "name": "Taxi_052",
3    "type": "Luxury",
4    "location": {
5      "type": "Point",
6      "coordinates": [
7        78.3368142118244,
8        27.66077927262116
9      ]
10    }
11  }

```

Body Cookies Headers (7) Test Results

200 OK 835 ms 539 B Save Response

Pretty Raw Preview Visualize JSON

```

1  {
2    "statusCode": 200,
3    "headers": {
4      "Content-Type": "application/json"
5    },
6    "body": "{\"message\": \"Successfully registered taxis within the service area\", \"registered_ids\": [\"65dbba3d37878068ca2c1e85\"]}"
7  }

```

{"name": "Taxi_052"} find result:

TaxiAggregator.taxis

50 DOCUMENTS 2 INDEXES

Documents Aggregations Schema Indexes Validation

Filter [{"name": "Taxi_052"}](#)

Explain Reset Find Options

[ADD DATA](#)

[EXPORT DATA](#)

1-1 of 1

```

_id: ObjectId('65dbbec5c2ac7d7d5653a185')
name: "Taxi_052"
type: "Luxury"
location: Object

```

9. Post below json request to create a customer data using API - <https://udmqfurkie.execute-api.us-east-2.amazonaws.com/dev/customer>

```

{
  "name": "Manju",
  "location": {

```

"type": "Point",
 "coordinates": [
 77.24098,
 28.65454
]
 }
 }

ACSECapstone / Customer

Save

POST https://udmqfurifie.execute-api.us-east-2.amazonaws.com/dev/customer **Send**

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies
 none form-data x-www-form-urlencoded raw binary GraphQL **JSON** Beautify

```

1 {
2   "name": "Manju",
3   "location": {
4     "type": "Point",
5     "coordinates": [
6       77.24098,
7       28.65454
8     ]
9   }
10 }

```

Body Cookies Headers (7) Test Results 200 OK 1037 ms 539 B

Pretty Raw Preview Visualize JSON

```

1 {
2   "statusCode": 200,
3   "headers": {
4     "Content-Type": "application/json"
5   },
6   "body": "{\"message\": \"Successfully registered customer within the service area\", \"registered_id\": \"65dbbf52269d6cbdbe185bf8\"}"
7 }

```

{"name": "Manju"} find result:

TaxiAggregator.customers

5 DOCUMENTS 2 INDEXES

Documents Aggregations Schema Indexes Validation

Filter Explain Reset Find Options

ADD DATA EXPORT DATA

1-1 of 1

```

_id: ObjectId('65dbbf52269d6cbdbe185bf8')
name: "Manju"
location: Object

```

9. Service area validation for customer. Post below request to API (<https://udmqfurifie.execute-api.us-east-2.amazonaws.com/dev/customer>) to test this.

```
{
  "name": "Manju",
  "location": {
    "type": "Point",
    "coordinates": [
      87.24098,
      28.65454
    ]
  }
}
```

The screenshot shows the Postman interface with the following details:

- Method:** POST
- URL:** <https://udmqfurkie.execute-api.us-east-2.amazonaws.com/dev/customer>
- Body (JSON):**

```

1  {
2   ... "name": "Manju",
3   ... "location": {
4     ... "type": "Point",
5     ... "coordinates": [
6       ... 87.24098,
7       ... 28.65454
8     ...
9   ...
10 }

```
- Response Status:** 200 OK
- Response Headers:**
 - Content-Type: application/json
- Response Body:**

```

1 {
2   "statusCode": 200,
3   "headers": {
4     "Content-Type": "application/json"
5   },
6   "body": "{\"message\": \"Customer not within the service area\"}"
7 }

```

10. Service area validation for taxi. Post below request to API - <https://udmqfurkie.execute-api.us-east-2.amazonaws.com/dev/taxi> to test this.

```
{
  "name": "Taxi_052",
  "type": "Luxury",
  "location": {
    "type": "Point",
    "coordinates": [
      78.3368142118244,
```

```
    37.66077927262116  
]  
}  
}
```

ACSECapstone / Taxi

Save ⌂ ⌄ ⌅

POST https://udmqfurifie.execute-api.us-east-2.amazonaws.com/dev/taxi **Send**

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL **JSON** Beautify

```
1 {  
2   "name": "Taxi_052",  
3   "type": "Luxury",  
4   "location": {  
5     "type": "Point",  
6     "coordinates": [  
7       78.3368142118244,  
8       37.66077927262116  
9     ]  
10   }  
11 }
```

Body Cookies Headers (7) Test Results 200 OK 879 ms 493 B Save Response

Pretty Raw Preview Visualize JSON **JSON**

```
1 {  
2   "statusCode": 200,  
3   "headers": {  
4     "Content-Type": "application/json"  
5   },  
6   "body": "{\"message\": \"No taxis were registered; none were within the service area\"}"  
7 }
```

EC2 instance:

The screenshot shows the AWS EC2 Instances page. The left sidebar includes links for EC2 Dashboard, EC2 Global View, Events, Instances (with sub-links for Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, and Capacity Reservations), Images (AMIs, AMI Catalog), Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), Network & Security (Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces), and Load Balancing. The main content area displays a table for 'Instances (1/1)'. The table has columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone. One row is selected for 'cp2' (Instance ID: i-0cf3f75984444cf08), which is 'Running' on an 't2.micro' instance type in the 'us-east-2a' availability zone. Below the table, a detailed view for 'Instance: i-0cf3f75984444cf08 (cp2)' is shown, with tabs for Details, Status and alarms, Monitoring, Security, Networking, Storage, and Tags. The 'Details' tab is active, displaying information such as Instance ID (i-0cf3f75984444cf08 (cp2)), Public IPv4 address (3.143.22.60), Private IPv4 addresses (172.31.9.133), Public IPv4 DNS (ec2-3-143-22-60.us-east-2.compute.amazonaws.com), Instance state (Running), Hostname type (IP name: ip-172-31-9-133.us-east-2.compute.internal), Private IP DNS name (ip-172-31-9-133.us-east-2.compute.internal), Answer private resource DNS name (IPv4 (A)), Instance type (t2.micro), VPC ID, and Elastic IP addresses (none listed). The bottom of the page includes CloudShell, Feedback, and copyright information: © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences.

API Gateway:

Lambda Functions:

Taxi Registration Lambda:

TaxiAggregator

Function overview

Description: -

Last modified: 3 hours ago

Function ARN: arn:aws:lambda:us-east-2:820919553617:function:TaxiAggregator

Function URL: Info

Code source

```

1 import json
2 import pymongo
3
4 # Connect to your MongoDB (replace with your actual connection details)
5 MONGO_URL = "mongodb://ec2-3-143-22-60.us-east-2.compute.amazonaws.com:27018/"
6 DB_NAME = "TaxiAggregator"
7
8 def fetch_service_area_boundary(db):
9     """
10         Fetch the service area boundary from MongoDB.
11     """

```

Code | Test | Monitor | Configuration | Aliases | Versions

Create a simple web app

In this tutorial you will learn how to:

- Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
- Invoke your function through its function URL

[Learn more](#)

[Start tutorial](#)

Customer/User registration Lambda:

CustomerRegistration

Function overview

Description: -

Last modified: 3 hours ago

Function ARN: arn:aws:lambda:us-east-2:820919553617:function:CustomerRegistration

Function URL: Info

Code source

```

1 import json
2 import pymongo
3
4 # Connect to your MongoDB (replace with your actual connection details)
5 MONGO_URL = "mongodb://ec2-3-143-22-60.us-east-2.compute.amazonaws.com:27018/"
6 DB_NAME = "TaxiAggregator"
7
8 def fetch_service_area_boundary(db):
9     """
10         Fetching service area boundary...
11         service_area = db.service_areas.find_one({"name": "Default Service Area"})
12     """

```

Code | Test | Monitor | Configuration | Aliases | Versions

Create a simple web app

In this tutorial you will learn how to:

- Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
- Invoke your function through its function URL

[Learn more](#)

[Start tutorial](#)

Ride Find Lambda:

The screenshot shows the AWS Lambda console for the 'RideFind' function. The top navigation bar includes links for Services, Search, Option+S, Ohio, and greatlearningco170888192858@8209-1955-3617. The main interface displays the function overview with tabs for Diagram, Template, and Code. The Diagram tab shows a basic architecture with an API Gateway triggering a Lambda function named 'RideFind'. The Lambda function has no layers. The 'Code source' tab is active, showing the code editor with the following Python script:

```
1 import json
2 from pymongo import MongoClient, GEOSPHERE
3
4 # Environment variables for MongoDB connection
5 MONGO_URI = "mongodb://ec2-3-143-22-60.us-east-2.compute.amazonaws.com:27018/"
6 DB_NAME = "TaxiAggregator"
7
8 def fetch_nearest_taxis(customer_loc, taxi_type=None, limit=2):
9     client = MongoClient(MONGO_URI)
10    db = client[DB_NAME]
11    taxis = db['taxis']
```

The right sidebar contains sections for 'Info' (with a note about common use cases), 'Tutorials' (with a 'Create a simple web app' section), and 'Learn more' (with a 'Start tutorial' button).

Docker instance running in EC2:

```
[ubuntu@ip-172-31-9-133:~]$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                         NAMES
f5ed6430b154        mongo:latest       "docker-entrypoint.s..."   7 days ago         Up 7 days          0.0.0.0:27018->27017/tcp, ::27018->27017/tcp   mongodb
```