Cloud Computing Project Design and Implementation (Group 7)

Ting Han Ko, Xiangxu Li, Yilin Yang

Data Source and Procurement

Data source link:

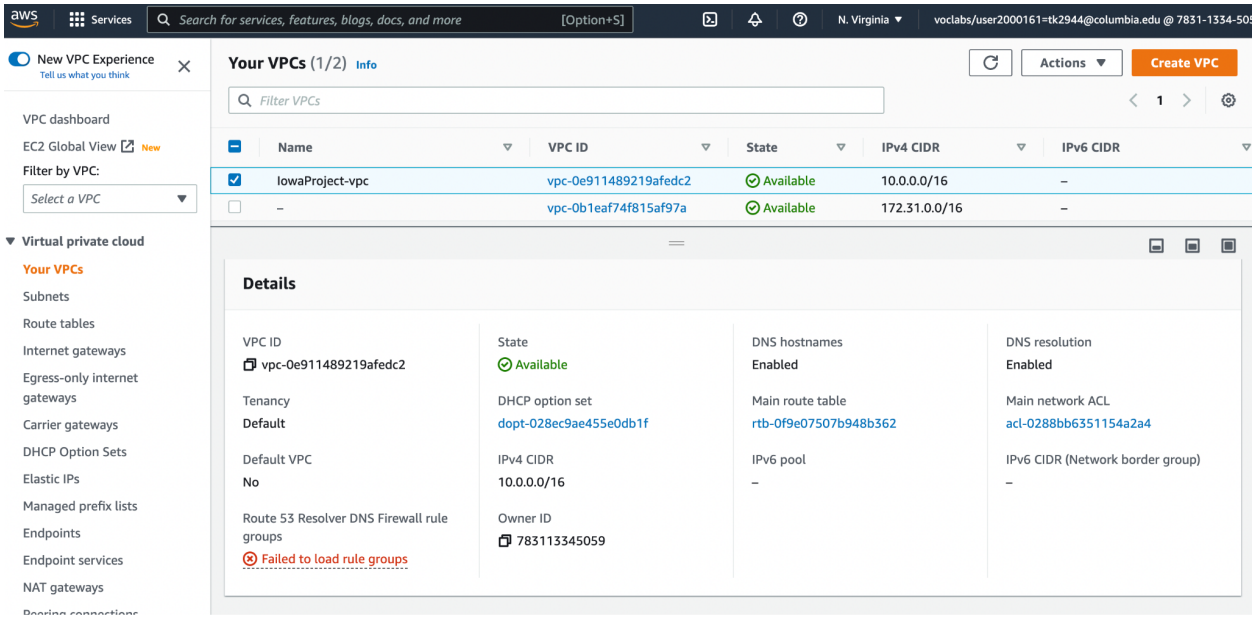
https://www.opendatanetwork.com/dataset/mydata.iowa.gov/m3tr-qhgy

This dataset contains the spirits purchase information of Iowa Class E liquor licensees by product and date of purchase from 2012 to 2022. Class E liquor is unopened liquor containers typically sold in grocery, liquor, and convenience stores.Data was extracted from the open data website, and it was uploaded to our S3 bucket through Amazon S3 console. A csv file originally 4.77 GB, resized to 1.2 GB for performance and efficiency purposes. It contains sales records in grocery stores, liquor stores, and convenience stores.

Amazon VPC

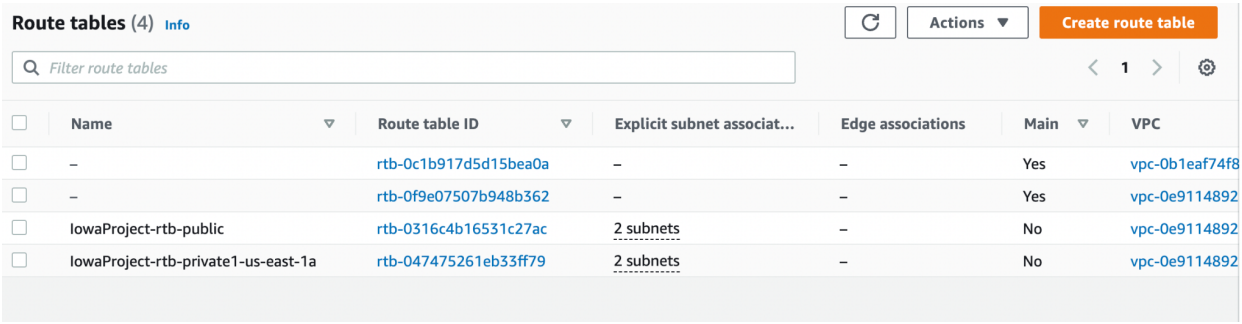
We wanted to set up 2 availability zones each with two subnets for private and public use. Therefore we also need to set up a NET gateway to connect those subnets together. Below is how we configure our VPC, subnets, route tables, NAT Gateway:

VPC Details Screenshot

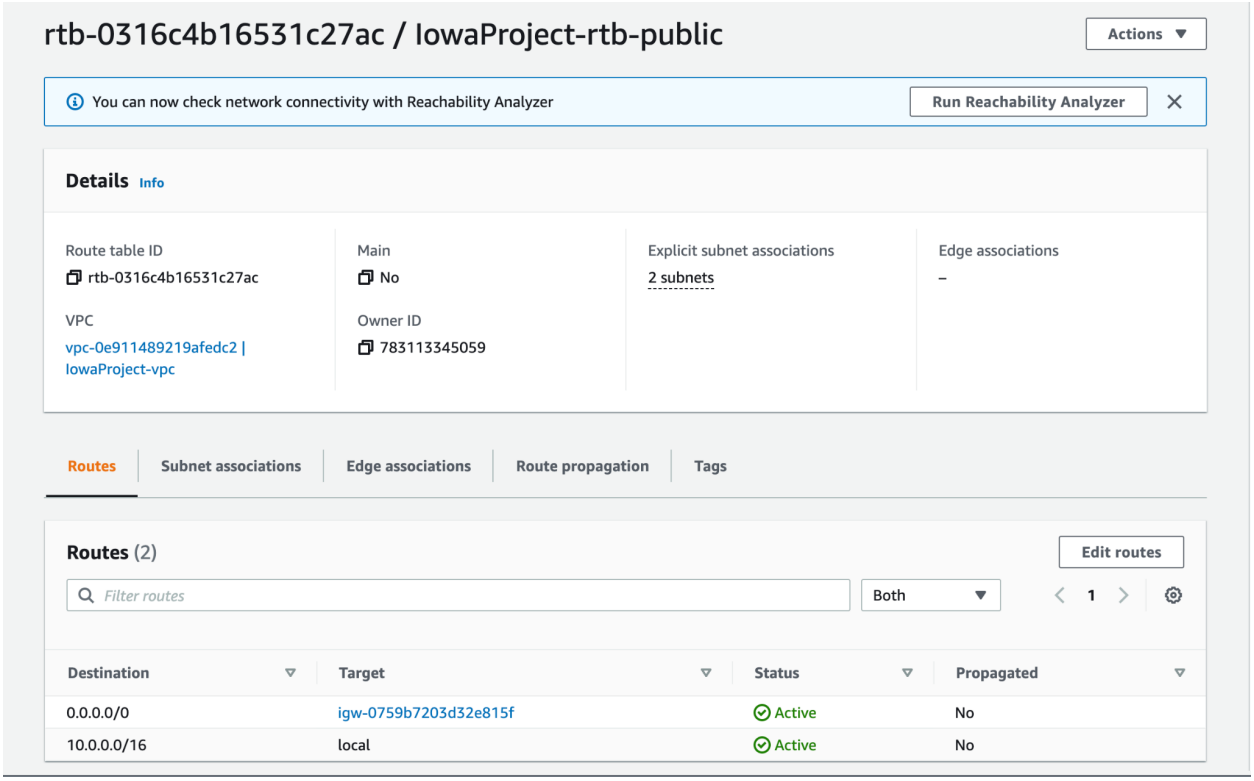


Two route tables

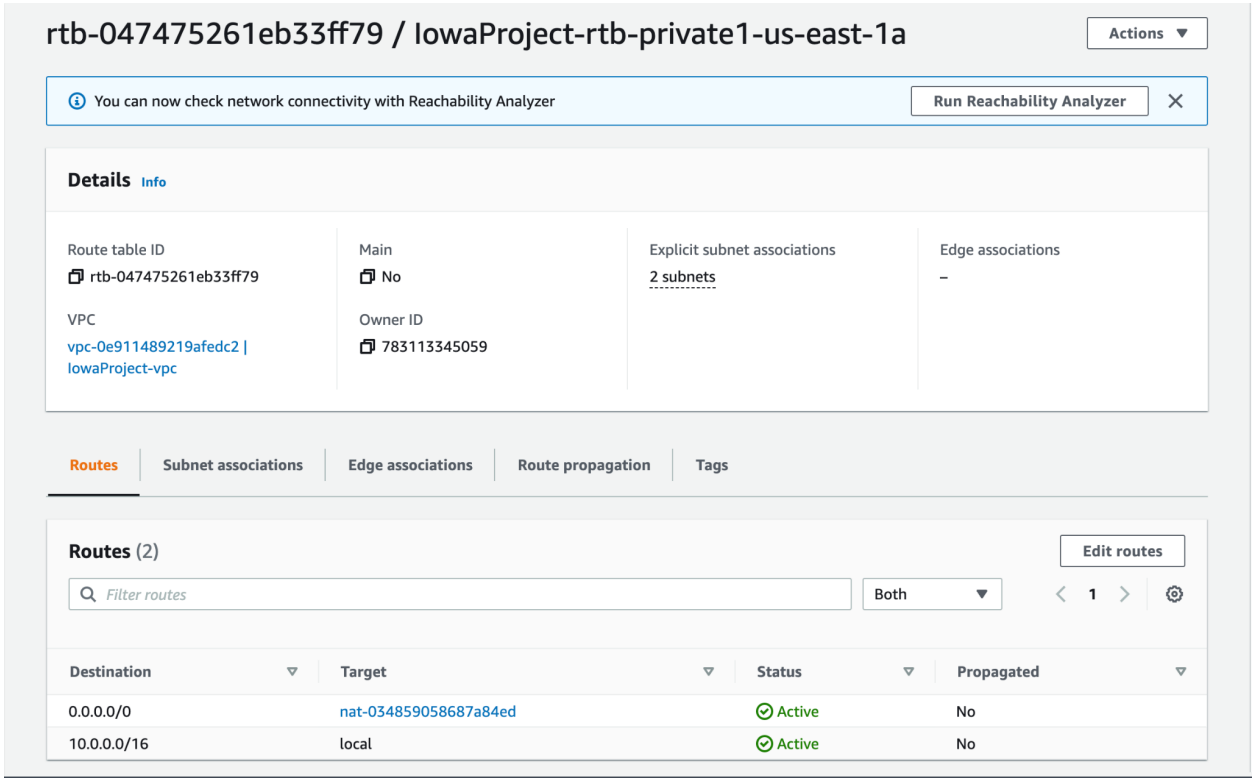
- IowaProject-rtb-public
- IowaProject-rtb-private1-us-east-1a



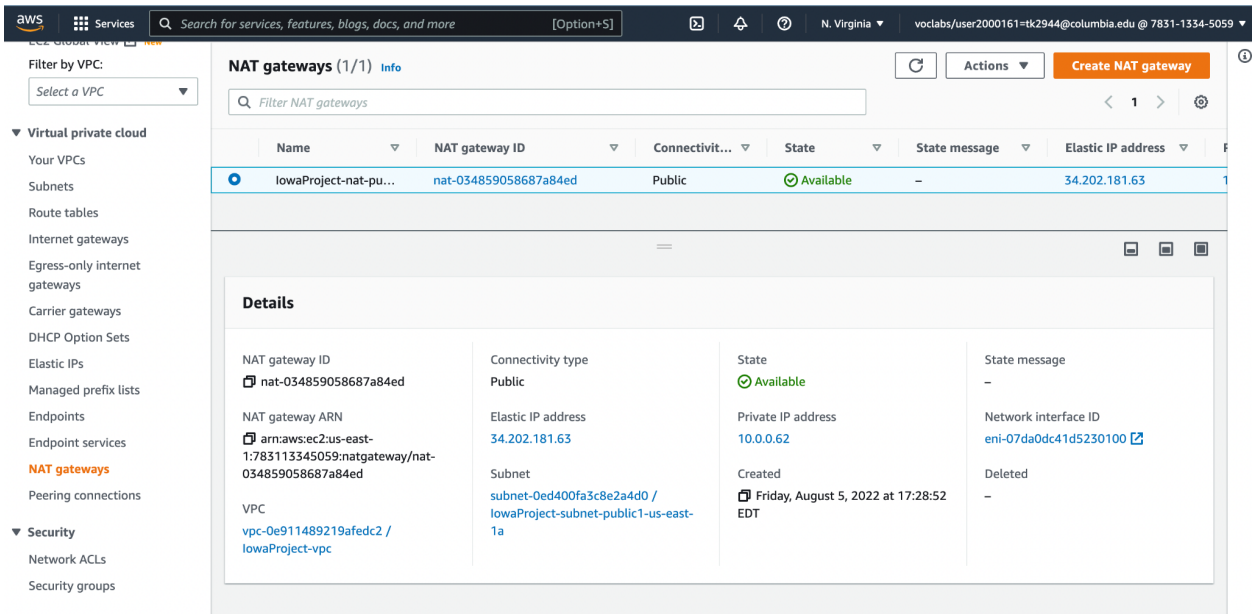
Public route table



Private route table



NAT Gateways



Four subnets

- IowaProject-subnet-private1-us-east-1a
- IowaProject-subnet-public1-us-east-1a
- IowaProject-subnet-private2
- IowaProject-subnet-public2

Subnets (10) Info

Actions ▼ | Create subnet

Filter subnets

1

Name	Subnet ID	State	VPC	IPv4 CIDR
IowaProject-subnet-private1-us-east-1a	subnet-07f613210f374031f	Available	vpc-0e911489219afedc2 | Iow...	10.0.1.0/24
IowaProject-subnet-public1-us-east-1a	subnet-0ed400fa3c8e2a4d0	Available	vpc-0e911489219afedc2 | Iow...	10.0.0.0/24
IowaProject-subnet-private2	subnet-01fb6e58f824a083c	Available	vpc-0e911489219afedc2 | Iow...	10.0.3.0/24
IowaProject-subnet-public2	subnet-0a57337500e8b447e	Available	vpc-0e911489219afedc2 | Iow...	10.0.2.0/24
–	subnet-0ca1e16ebffce13b4	Available	vpc-0b1eaf74f815af97a	172.31.0.0/20
–	subnet-079005b5febcfd101	Available	vpc-0b1eaf74f815af97a	172.31.80.0/20
–	subnet-0270c785aa77203f4	Available	vpc-0b1eaf74f815af97a	172.31.16.0/20
–	subnet-01f255e27f027e9be	Available	vpc-0b1eaf74f815af97a	172.31.64.0/20
–	subnet-0e41a98c38ceea648	Available	vpc-0b1eaf74f815af97a	172.31.32.0/20
–	subnet-01b8ba7514fad6f73	Available	vpc-0b1eaf74f815af97a	172.31.48.0/20

Private 1 us-east-1a subnet

subnet-07f613210f374031f / IowaProject-subnet-private1-us-east-1a

Actions ▼

Details

Subnet ID
subnet-07f613210f374031f

Available IPv4 addresses
251

Network border group
us-east-1

Default subnet
No

Customer-owned IPv4 pool
–

IPv6-only
No

DNS64
Disabled

Subnet ARN
arn:aws:ec2:us-east-1:783113345059:subnet/subnet-07f613210f374031f

IPv6 CIDR
–

VPC
vpc-0e911489219afedc2 | IowaProject-vpc

Auto-assign public IPv4 address
No

Outpost ID
–

Hostname type
IP name

Owner
783113345059

State
Available

Availability Zone
us-east-1a

Route table
rtb-047475261eb33ff79 | IowaProject-rtb-private1-us-east-1a

Auto-assign IPv6 address
No

IPv4 CIDR reservations
–

Resource name DNS A record
Disabled

IPv4 CIDR
10.0.1.0/24

Availability Zone ID
use1-az6

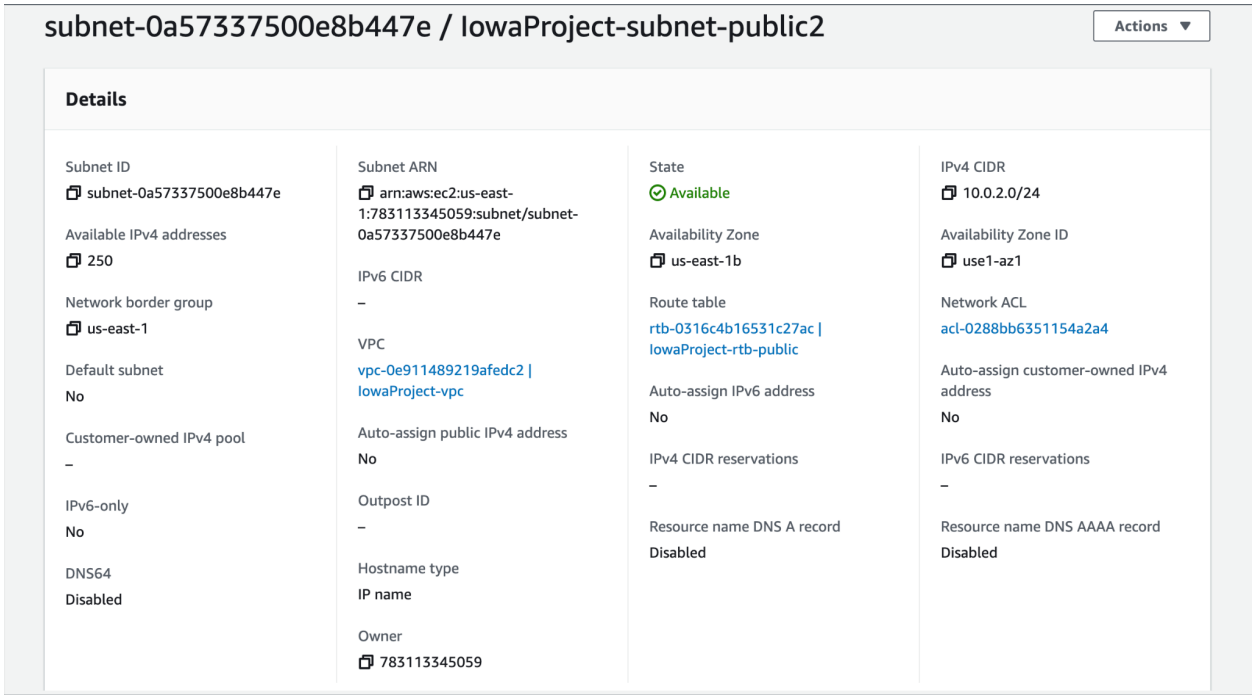
Network ACL
acl-0288bb6351154a2a4

Auto-assign customer-owned IPv4 address
No

IPv6 CIDR reservations
–

Resource name DNS AAAA record
Disabled

Public 2 us-east-1b subnet (associated with our EC2)



subnet-0a57337500e8b447e / IowaProject-subnet-public2

Actions

Details

Subnet ID subnet-0a57337500e8b447e	Subnet ARN arn:aws:ec2:us-east-1:783113345059:subnet/subnet-0a57337500e8b447e	State Available	IPv4 CIDR 10.0.2.0/24
Available IPv4 addresses 250	IPv6 CIDR –	Availability Zone us-east-1b	Availability Zone ID use1-az1
Network border group us-east-1	VPC vpc-0e911489219afedc2 | IowaProject-vpc	Route table rtb-0316c4b16531c27ac | IowaProject-rtb-public	Network ACL acl-0288bb6351154a2a4
Default subnet No	Auto-assign public IPv4 address No	Auto-assign IPv6 address No	Auto-assign customer-owned IPv4 address No
Customer-owned IPv4 pool –	Outpost ID –	IPv4 CIDR reservations –	IPv6 CIDR reservations –
IPv6-only No	Hostname type IP name	Resource name DNS A record Disabled	Resource name DNS AAAA record Disabled
DNS64 Disabled	Owner 783113345059		

Public 1 us-east-1a subnet

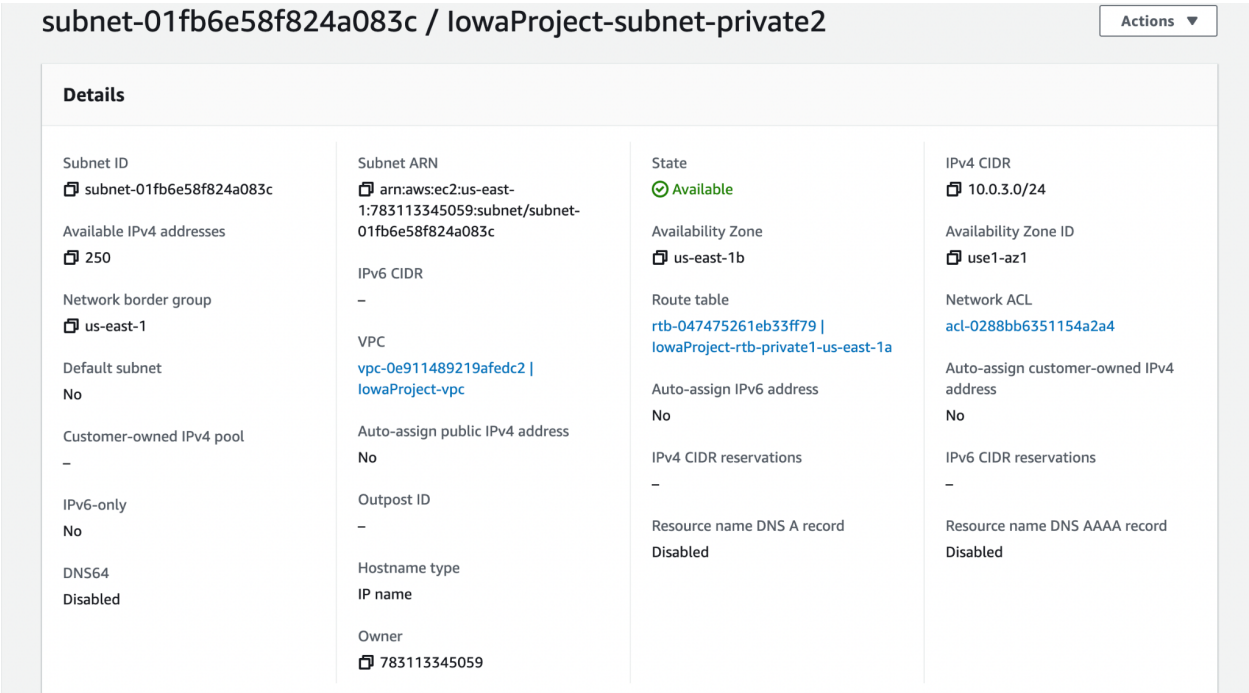
subnet-0ed400fa3c8e2a4d0 / IowaProject-subnet-public1-us-east-1a

Actions

Details

Subnet ID subnet-0ed400fa3c8e2a4d0	Subnet ARN arn:aws:ec2:us-east-1:783113345059:subnet/subnet-0ed400fa3c8e2a4d0	State Available	IPv4 CIDR 10.0.0.0/24
Available IPv4 addresses 250	IPv6 CIDR –	Availability Zone us-east-1a	Availability Zone ID use1-az6
Network border group us-east-1	VPC vpc-0e911489219afedc2 | IowaProject-vpc	Route table rtb-0316c4b16531c27ac | IowaProject-rtb-public	Network ACL acl-0288bb6351154a2a4
Default subnet No	Auto-assign public IPv4 address No	Auto-assign IPv6 address No	Auto-assign customer-owned IPv4 address No
Customer-owned IPv4 pool –	Outpost ID –	IPv4 CIDR reservations –	IPv6 CIDR reservations –
IPv6-only No	Hostname type IP name	Resource name DNS A record Disabled	Resource name DNS AAAA record Disabled
DNS64 Disabled	Owner 783113345059		

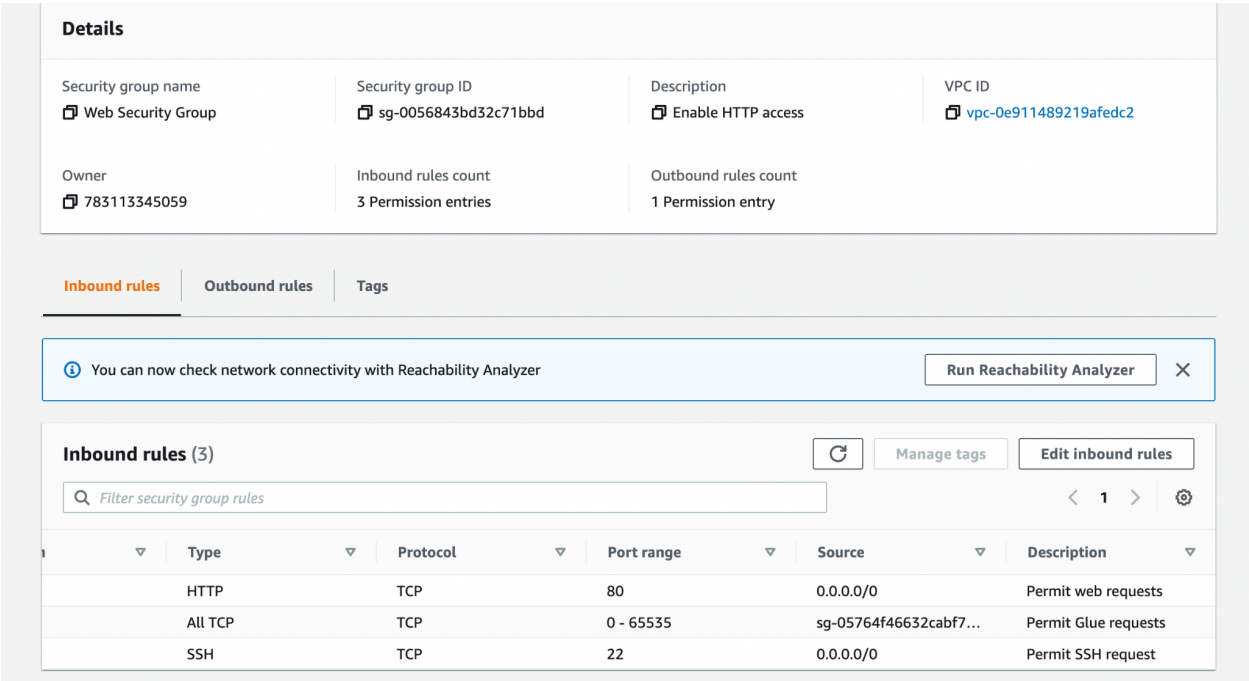
Private 2 us-east-1b subnet



Security Groups

We set up security groups to control access to our database, web server etc. They are mostly set to public for now for project sharing purposes. However, in the future we should especially modify Inbound Rules so we can have a better security outcome. Here's how we set up our security groups for now:

1. Web security group



Details

Security group name	Security group ID	Description	VPC ID
Web Security Group	sg-0056843bd32c71bbd	Enable HTTP access	vpc-0e911489219afedc2

Owner	Inbound rules count	Outbound rules count
783113345059	3 Permission entries	1 Permission entry

Inbound rules | Outbound rules | Tags

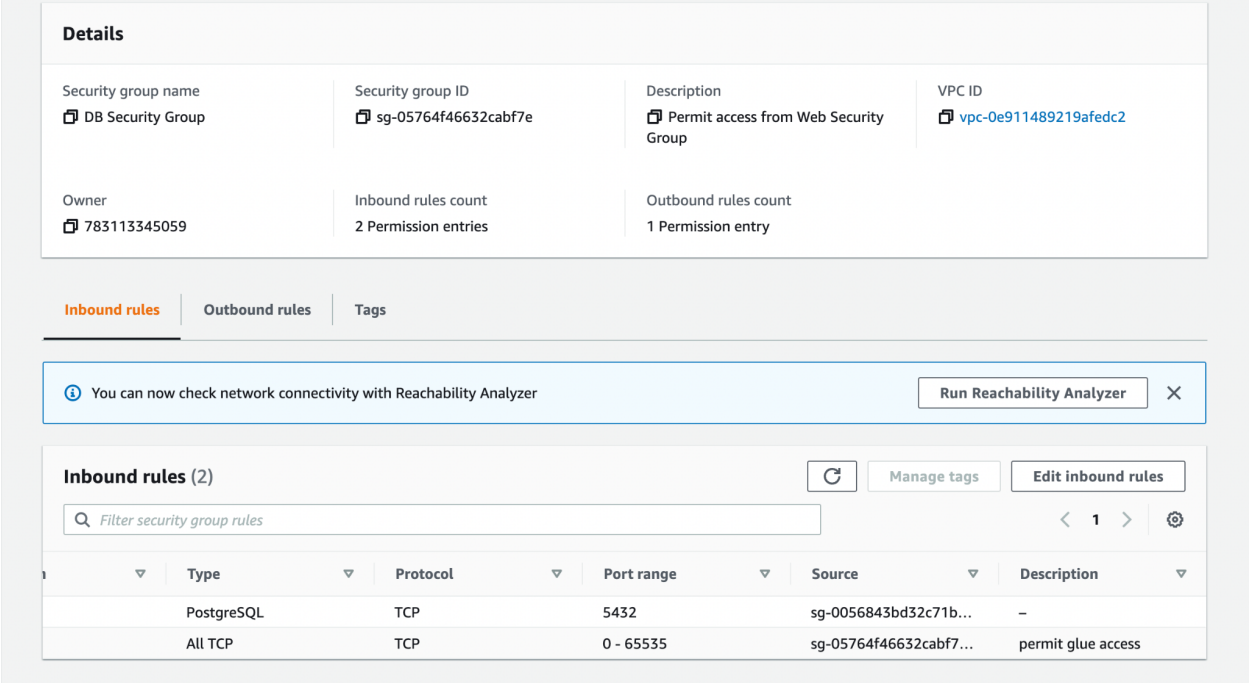
You can now check network connectivity with Reachability Analyzer — Run Reachability Analyzer

Inbound rules (3) — Manage tags | Edit inbound rules

Filter security group rules

Type	Protocol	Port range	Source	Description
HTTP	TCP	80	0.0.0.0/0	Permit web requests
All TCP	TCP	0 - 65535	sg-05764f46632cabf7…	Permit Glue requests
SSH	TCP	22	0.0.0.0/0	Permit SSH request

2. DB Security group (for our RDS - PostgresQL Database)



Details

Security group name	Security group ID	Description	VPC ID
DB Security Group	sg-05764f46632cabf7e	Permit access from Web Security Group	vpc-0e911489219afedc2

Owner	Inbound rules count	Outbound rules count
783113345059	2 Permission entries	1 Permission entry

Inbound rules | Outbound rules | Tags

You can now check network connectivity with Reachability Analyzer — Run Reachability Analyzer

Inbound rules (2) — Manage tags | Edit inbound rules

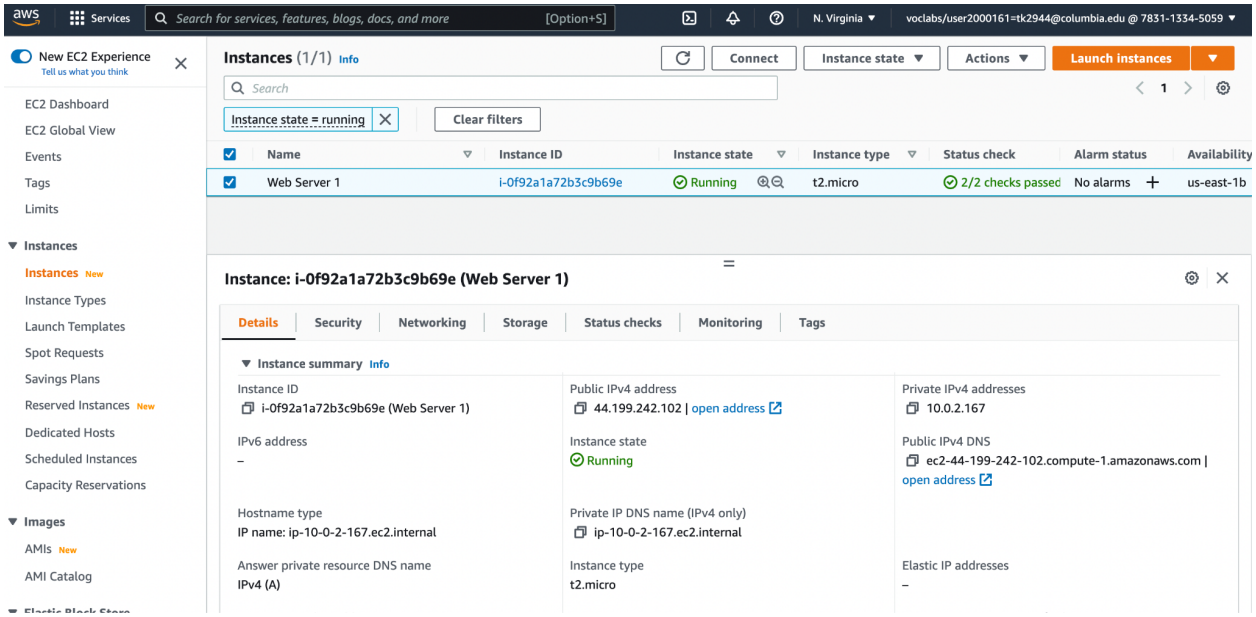
Filter security group rules

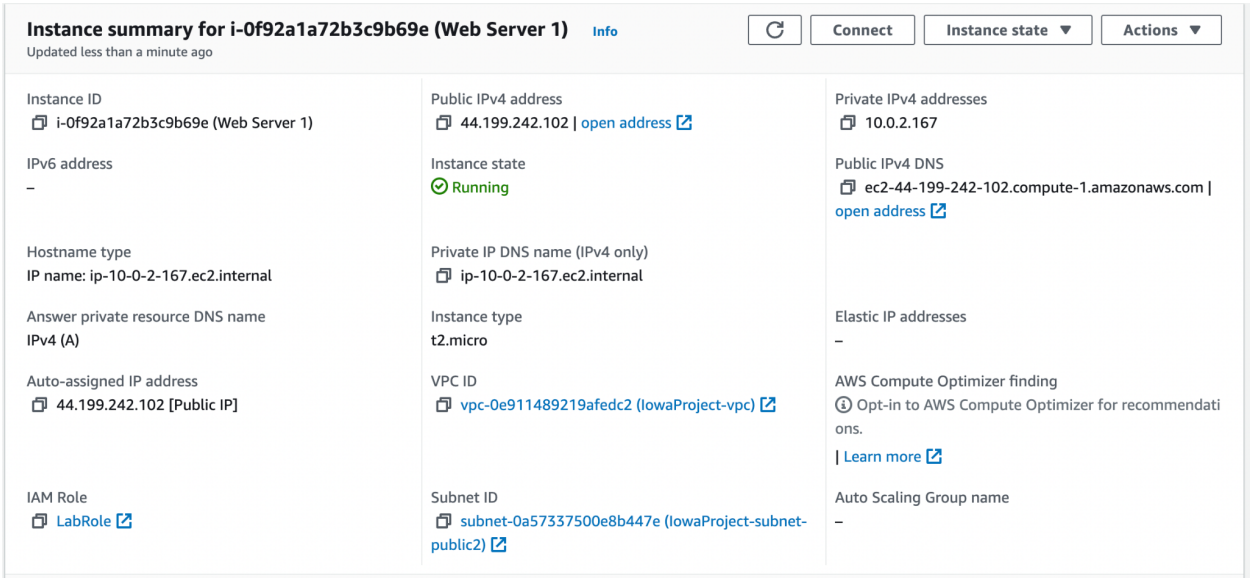
Type	Protocol	Port range	Source	Description
PostgreSQL	TCP	5432	sg-0056843bd32c71b…	–
All TCP	TCP	0 - 65535	sg-05764f46632cabf7…	permit glue access

Amazon EC2

We have to set up an EC2 instance in our VPC for web server usage and temporarily for our data pipeline transitioning data from S3 to our database. In an ideal scenario we will only have to use an EC2 instance solely for the web server most of the time..

Details and Summary Screenshots

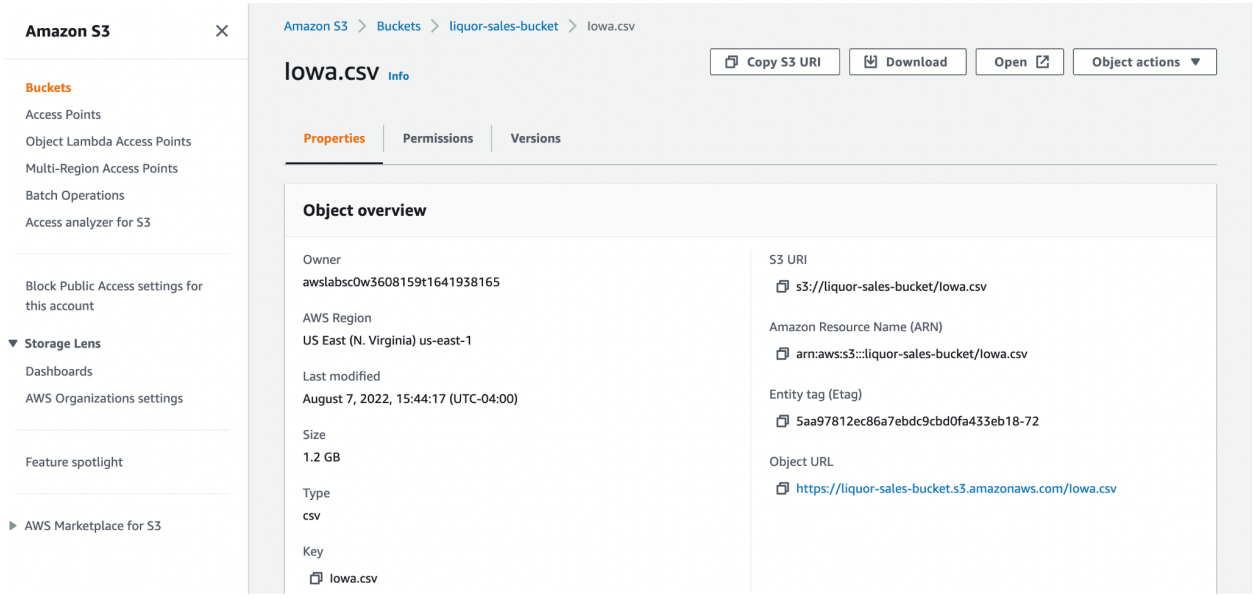




Amazon S3

S3 is scalable and reliable for our storage infrastructure. In the case of an extended project, we will be able to scale our storage seamlessly and in a long term project we can even categorize data into different storage trench/levels, to minimize cost. Here's how we set up our S3 bucket for Iowa dataset storage:

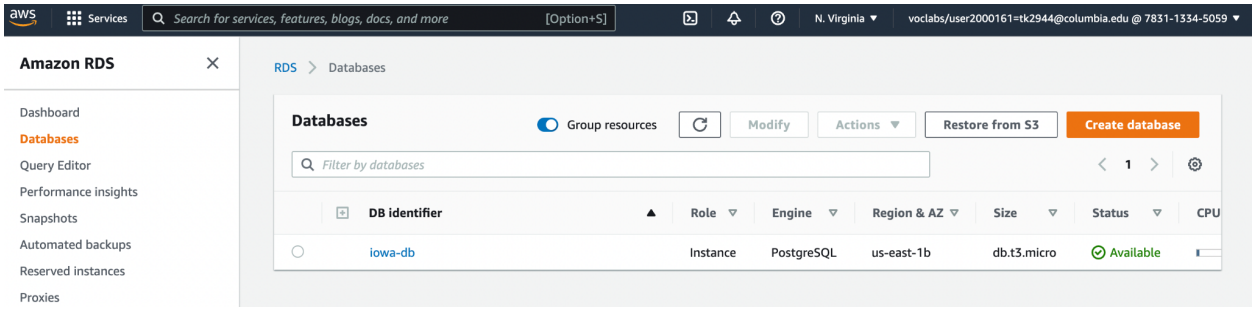
S3 Bucket created and the Iowa.csv file uploaded to the S3 bucket



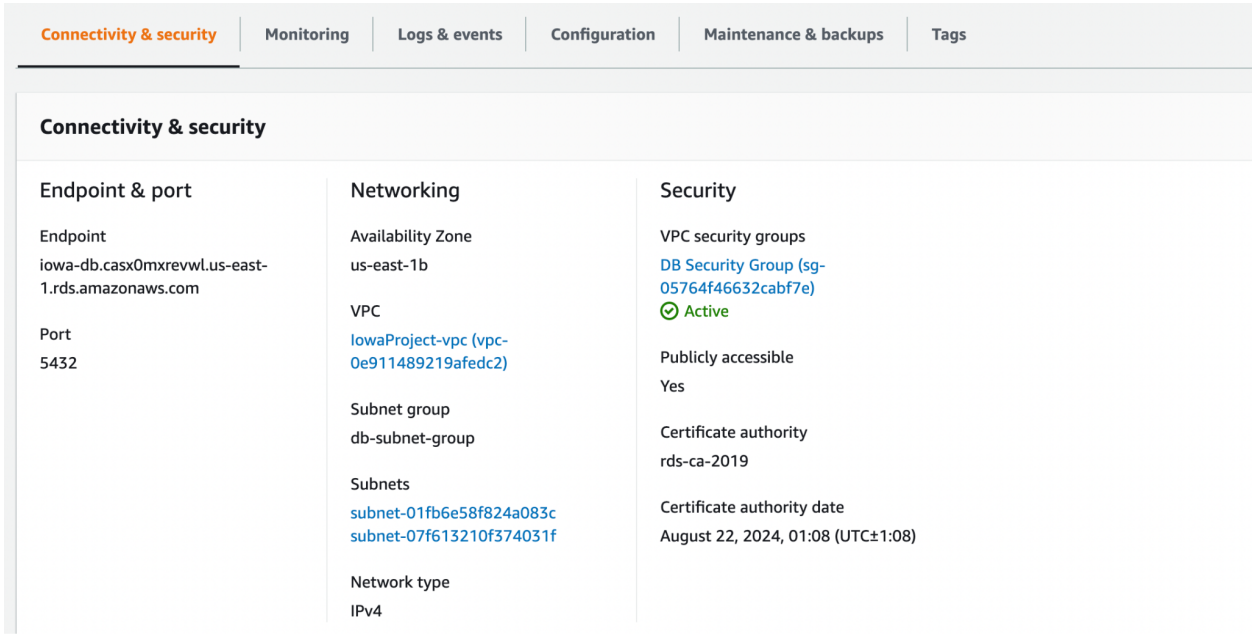
Amazon RDS

We wanted to set up a database to allow our query to directly access the data and show the query and analysis result, and we wanted to utilize the convenience and interface of Flask building on this database. Here's how we set up the database:

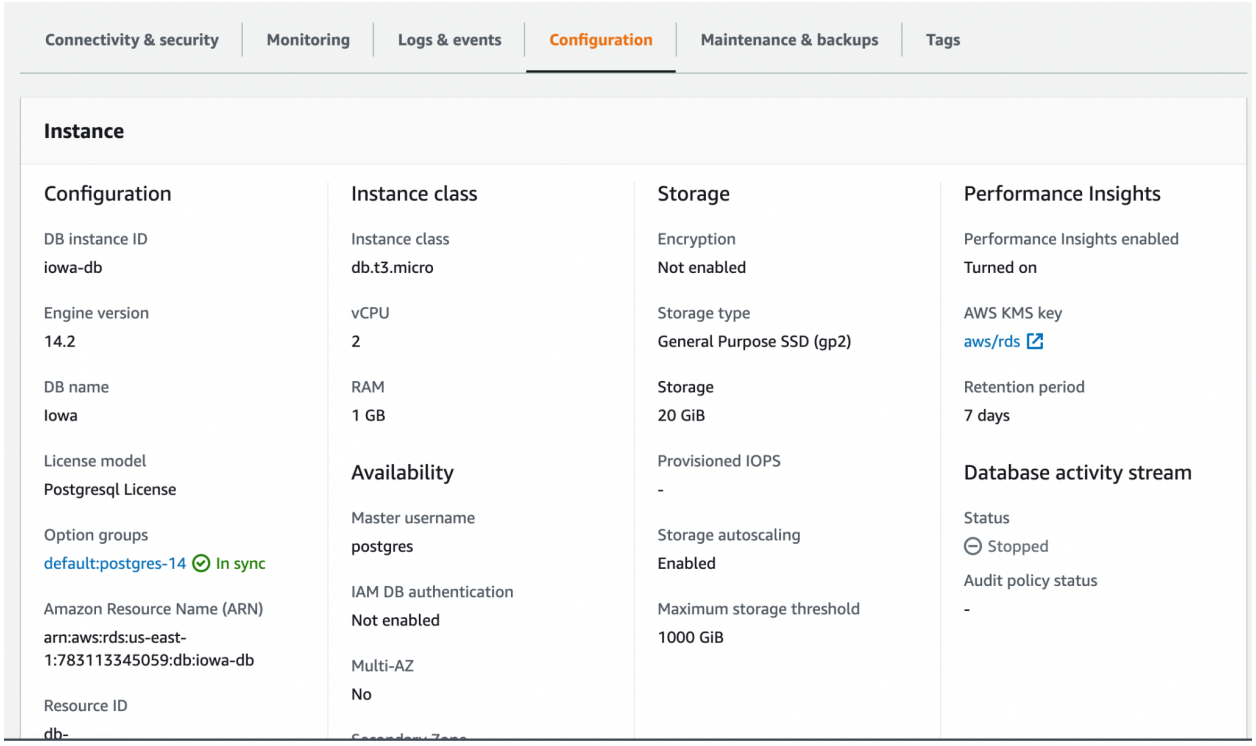
DB Instance



DB Connectivity and Security (Username: postgres; Password: postgres)

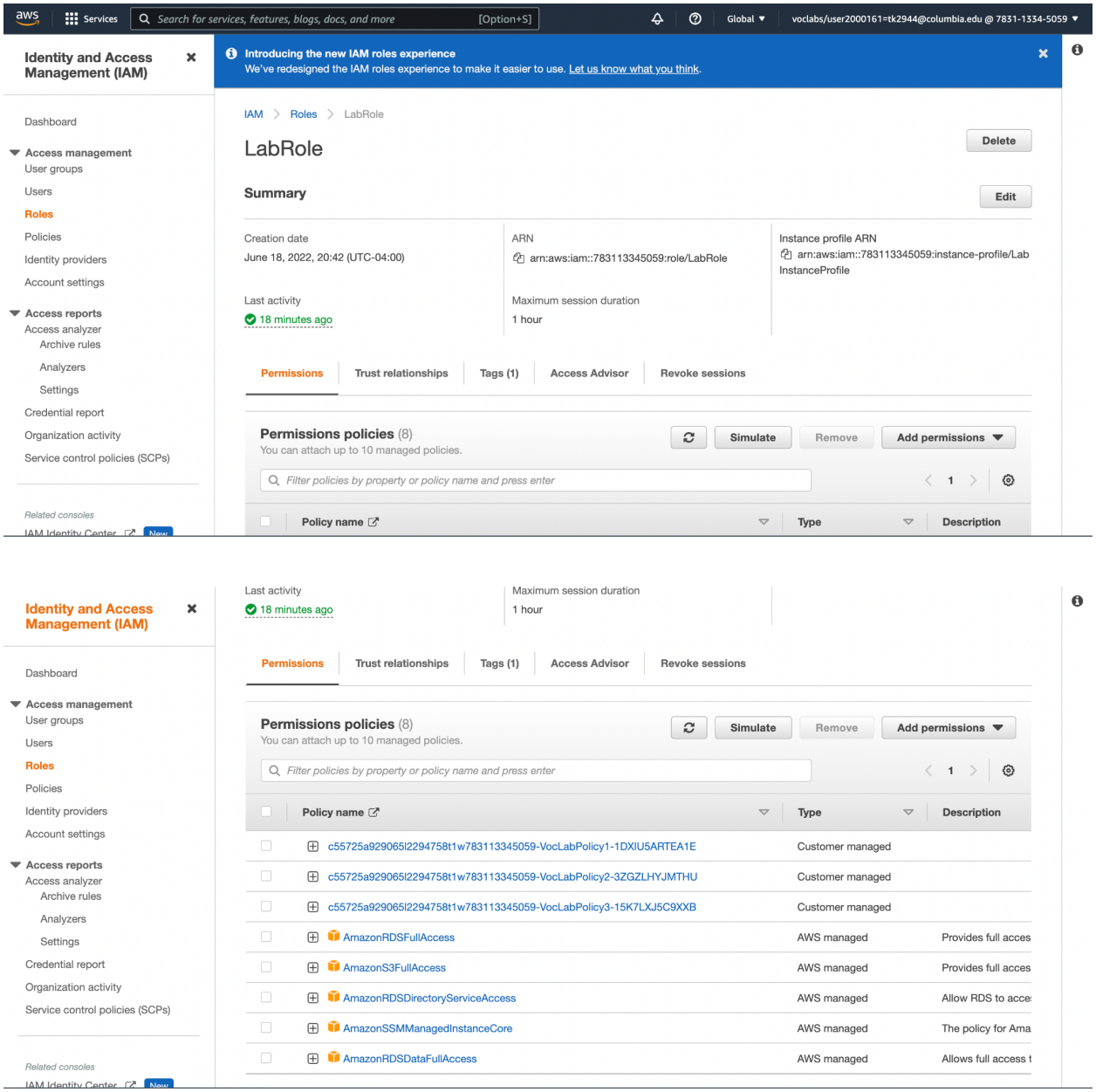


DB Configuration (database name: Iowa)



AWS IAM

We attached policies to LabRole in IAM to allow EC2 and RDS to access S3 so we can continue our ETL process. However, in the future we should set up more roles to ensure a better divided task access from different groups, enhancing our security. For now, this is how we set up our IAM role:



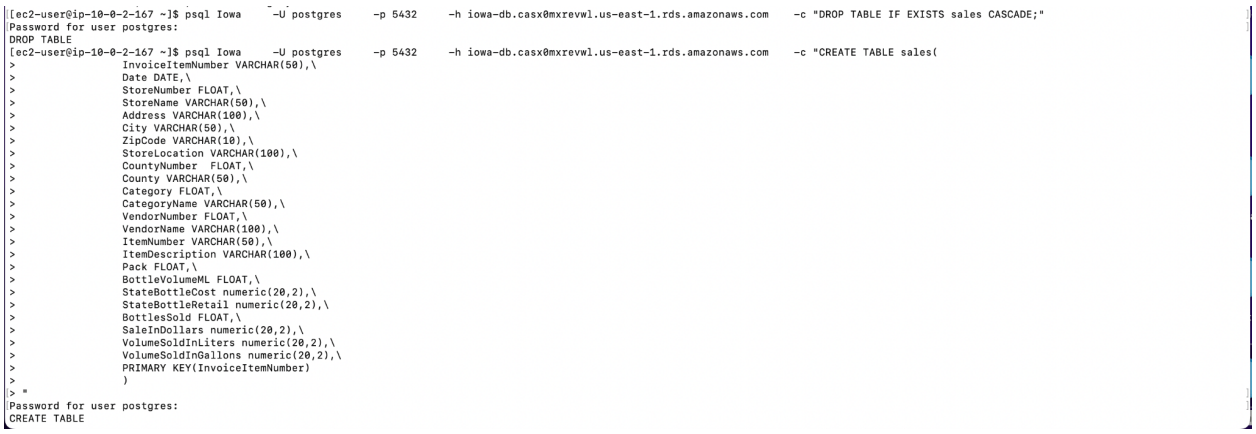
ETL Pipeline

We directly transfer the data through the command line interface from S3 Bucket to our EC2 Instance, and then from EC2 instance to our RDS DB Iowa database.

Screenshot of Importing the “Iowa.csv” file from S3 to EC2

```
[ec2-user@ip-10-0-2-167 ~]$ aws s3 cp "s3://liquor-sales-bucket/Iowa.csv" Iowa.csv
 download: s3://liquor-sales-bucket/Iowa.csv to ./Iowa.csv
[ec2-user@ip-10-0-2-167 ~]$ ls
 Iowa.csv  Iowa_Liquor_Sales.csv  out.csv  testingImport.py  testing.py  tiny_file.csv
```

Create an empty table in RDS DB- Iowa through EC2



```
[ec2-user@ip-10-0-2-167 ~]$ psql Iowa    -U postgres    -p 5432    -h iowa-db.casx0mxrevwl.us-east-1.rds.amazonaws.com    -c "DROP TABLE IF EXISTS sales CASCADE;"
Password for user postgres:
DROP TABLE
[ec2-user@ip-10-0-2-167 ~]$ psql Iowa    -U postgres    -p 5432    -h iowa-db.casx0mxrevwl.us-east-1.rds.amazonaws.com    -c "CREATE TABLE sales(
>               InvoiceItemNumber VARCHAR(50),\
>               Date DATE,\
>               StoreNumber FLOAT,\
>               StoreName VARCHAR(50),\
>               Address VARCHAR(100),\
>               City VARCHAR(50),\
>               ZipCode VARCHAR(10),\
>               StoreLocation VARCHAR(100),\
>               CountyNumber  FLOAT,\
>               County VARCHAR(50),\
>               Category FLOAT,\
>               CategoryName VARCHAR(50),\
>               VendorNumber FLOAT,\
>               VendorName VARCHAR(100),\
>               ItemNumber VARCHAR(50),\
>               ItemDescription VARCHAR(100),\
>               Pack FLOAT,\
>               BottleVolumeML FLOAT,\
>               StateBottleCost numeric(20,2),\
>               StateBottleRetail numeric(20,2),\
>               BottlesSold FLOAT,\
>               SaleInDollars numeric(20,2),\
>               VolumeSoldInLiters numeric(20,2),\
>               VolumeSoldInGallons numeric(20,2),\
>               PRIMARY KEY(InvoiceItemNumber)
>               )
> "
Password for user postgres:
CREATE TABLE
```

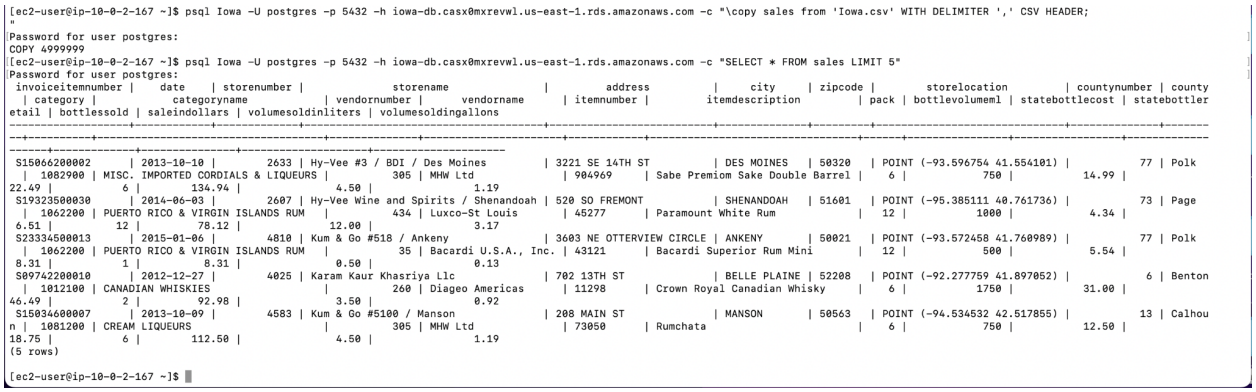
Code reference:

psql Iowa -U postgres -p 5432 -h iowa-db.casx0mxrevwl.us-east-1.rds.amazonaws.com -c "DROP TABLE IF EXISTS sales CASCADE;”

psql Iowa -U postgres -p 5432 -h iowa-db.casx0mxrevwl.us-east-1.rds.amazonaws.com -c "CREATE TABLE sales(
InvoiceItemNumber VARCHAR(50),\
Date DATE,\
StoreNumber FLOAT,\
StoreName VARCHAR(50),\
Address VARCHAR(100),\
City VARCHAR(50),\
ZipCode VARCHAR(10),\
StoreLocation VARCHAR(100),\
CountyNumber FLOAT,\

```
    County VARCHAR(50),\
    Category FLOAT,\
    CategoryName VARCHAR(50),\
    VendorNumber FLOAT,\
    VendorName VARCHAR(100),\
    ItemNumber VARCHAR(50),\
    ItemDescription VARCHAR(100),\
    Pack FLOAT,\
    BottleVolumeML FLOAT,\
    StateBottleCost numeric(20,2),\
    StateBottleRetail numeric(20,2),\
    BottlesSold FLOAT,\
    SaleInDollars numeric(20,2),\
    VolumeSoldInLiters numeric(20,2),\
    VolumeSoldInGallons numeric(20,2),\
    PRIMARY KEY(InvoiceItemNumber)
    )
"
```

Loaded the csv file from S3 to RDS DB- Iowa and read table (first 5 rows)



```
[ec2-user@ip-10-0-2-167 ~]$ psql Iowa -U postgres -p 5432 -h iowa-db.casx0mxrevwl.us-east-1.rds.amazonaws.com -c "\copy sales from 'Iowa.csv' WITH DELIMITER ',' CSV HEADER;
"
Password for user postgres:
COPY 4999999
[ec2-user@ip-10-0-2-167 ~]$ psql Iowa -U postgres -p 5432 -h iowa-db.casx0mxrevwl.us-east-1.rds.amazonaws.com -c "SELECT * FROM sales LIMIT 5"
Password for user postgres:
 invoiceitemnumber |    date    | storenumber |              storename              |         address          |     city     | zipcode |        storelocation         | countynumber | county
   | category |           categoryname           | vendornumber |      vendorname      | itemnumber |        itemdescription         | pack | bottlevolumeml | statebottlecost | statebottler
etail | bottlessold | saleindollars | volumesoldinliters | volumesoldingallons
--------------------+------------+-------------+-------------------------------------+--------------------------+--------------+---------+------------------------------+--------------+--------
--+----------+----------------------------------+--------------+----------------------+------------+--------------------------------+------+----------------+-----------------+--------------
------+-------------+---------------+--------------------+----------------------
 S15066200002       | 2013-10-10 |        2633 | Hy-Vee #3 / BDI / Des Moines        | 3221 SE 14TH ST          | DES MOINES   | 50320   | POINT (-93.596754 41.554101) |           77 | Polk
   |  1082900 | MISC. IMPORTED CORDIALS & LIQUEURS |          305 | MHW Ltd              |     904969 | Sabe Premiom Sake Double Barrel |    6 |            750 |           14.99 |
22.49 |           6 |        134.94 |               4.50 |                 1.19
 S19323500030       | 2014-06-03 |        2607 | Hy-Vee Wine and Spirits / Shenandoah | 520 SO FREMONT           | SHENANDOAH   | 51601   | POINT (-95.385111 40.761736) |           73 | Page
   |  1062200 | PUERTO RICO & VIRGIN ISLANDS RUM   |          434 | Luxco-St Louis       |      45277 | Paramount White Rum             |   12 |           1000 |            4.34 |
 6.51 |          12 |         78.12 |              12.00 |                 3.17
 S23334500013       | 2015-01-06 |        4810 | Kum & Go #518 / Ankeny              | 3603 NE OTTERVIEW CIRCLE | ANKENY       | 50021   | POINT (-93.572458 41.760989) |           77 | Polk
   |  1062200 | PUERTO RICO & VIRGIN ISLANDS RUM   |           35 | Bacardi U.S.A., Inc. |      43121 | Bacardi Superior Rum Mini       |   12 |            500 |            5.54 |
 8.31 |           1 |          8.31 |               0.50 |                 0.13
 S09742200010       | 2012-12-27 |        4025 | Karam Kaur Khasriya Llc             | 702 13TH ST              | BELLE PLAINE | 52208   | POINT (-92.277759 41.897052) |            6 | Benton
   |  1012100 | CANADIAN WHISKIES                  |          260 | Diageo Americas      |      11298 | Crown Royal Canadian Whisky     |    6 |           1750 |           31.00 |
46.49 |           2 |         92.98 |               3.50 |                 0.92
 S15034600007       | 2013-10-09 |        4583 | Kum & Go #5100 / Manson             | 208 MAIN ST              | MANSON       | 50563   | POINT (-94.534532 42.517855) |           13 | Calhou
n |  1081200 | CREAM LIQUEURS                     |          305 | MHW Ltd              |      73050 | Rumchata                        |    6 |            750 |           12.50 |
18.75 |           6 |        112.50 |               4.50 |                 1.19
(5 rows)

[ec2-user@ip-10-0-2-167 ~]$
```

Code reference:

```
psql Iowa -U postgres -p 5432 -h
iowa-db.casx0mxrevwl.us-east-1.rds.amazonaws.com\
 -c "\COPY sales FROM 'Iowa.csv' WITH DELIMITER ',' CSV HEADER; "
```

psql Iowa -U postgres -p 5432 -h iowa-db.casx0mxrevwl.us-east-1.rds.amazonaws.com -c "SELECT * FROM sales LIMIT 5"

Data Analysis Example

Code reference:

psql Iowa -U postgres -p 5432 -h iowa-db.casx0mxrevwl.us-east-1.rds.amazonaws.com -c "select county, sum(SaleinDollars) as total_sales_in_dollars

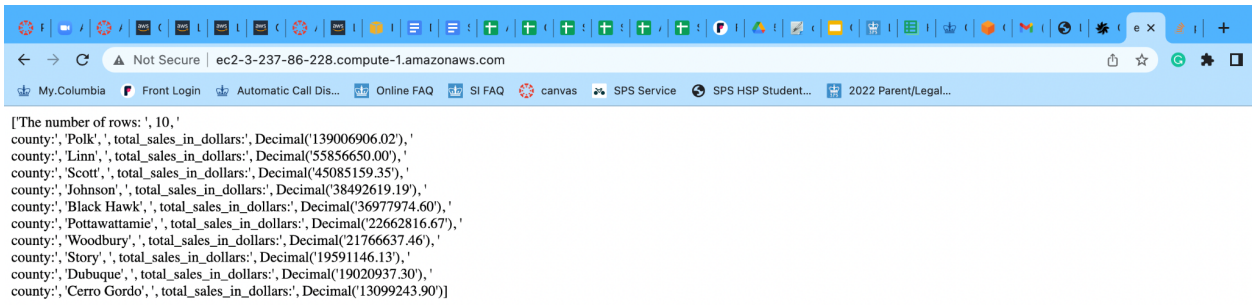
from sales
group by County
order by total_sales_in_dollars desc
limit 10;

Flask Server

We wanted to build a flask server to show query and simple analysis results.
Example of using python file to activate flask and query/ analyze data:

```
^C[ec2-user@ip-10-0-2-167 ~]$ sudo nano query_analysis_server.py
[ec2-user@ip-10-0-2-167 ~]$ sudo python3 query_analysis_server.py
WARNING: This is a development server. Do not use it in a production deployment. Use a productio
n WSGI server instead.
 * Serving Flask app 'APAN5450App'
 * Debug mode: off
 * Running on all addresses (0.0.0.0)
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://127.0.0.1:80
 * Running on http://10.0.2.167:80 (Press CTRL+C to quit)
158.222.128.225 - - [10/Aug/2022 01:56:25] "GET / HTTP/1.1" 200 -
^C[ec2-user@ip-10-0-2-167 ~]$ sudo nano query_analysis_server.py
[ec2-user@ip-10-0-2-167 ~]$ sudo python3 query_analysis_server.py
WARNING: This is a development server. Do not use it in a production deployment. Use a productio
n WSGI server instead.
 * Serving Flask app 'APAN5450App'
 * Debug mode: off
 * Running on all addresses (0.0.0.0)
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://127.0.0.1:80
 * Running on http://10.0.2.167:80 (Press CTRL+C to quit)
158.222.128.225 - - [10/Aug/2022 01:57:29] "GET / HTTP/1.1" 200 -
```

Result showing on the web page:



Flask Code Reference:

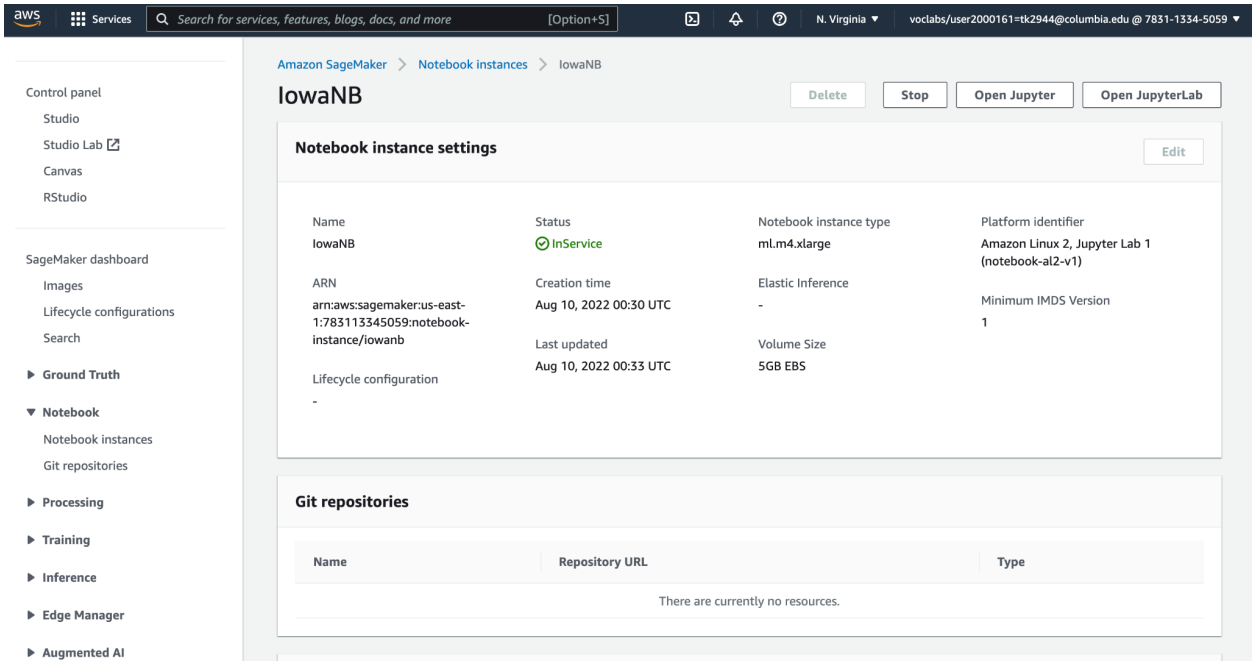
```
import sys
import psycopg
sys.path.append("/home/ec2-user/.local/lib/python3.7/site-packages")
from flask import Flask

app = Flask("APAN5450App")

@app.route('/')
def hello_world():
```

```
    output = []
# Connect to an existing database
    with psycopg.connect(
        host="iowa-db.casx0mxrevwl.us-east-1.rds.amazonaws.com",
        port="5432",
        dbname="Iowa",
        user="postgres",
        password="postgres") as conn:
            # Open a cursor to perform database operations
            with conn.cursor() as cur:
            # Query the database to verify the inserted data
                queryCmd = '''SELECT county, sum(SaleinDollars) AS
total_sales_in_dollars FROM sales GROUP BY County ORDER BY
total_sales_in_dollars DESC LIMIT 10;'''
                cur.execute(queryCmd)
                output.append("The number of rows: ")
                output.append(cur.rowcount)

                row = cur.fetchone()
                while row is not None:
                    output.append("<br/>county:")
                    output.append(row[0])
                    output.append(", total_sales_in_dollars:")
                    output.append(row[1])
                    row = cur.fetchone()

                # Close the cursor
                cur.close()
    return str(output)

app.run(host='0.0.0.0', port=80)
```

Amazon SageMaker

We wanted to include further analysis possibilities in the future, especially machine learning models for time series analysis to predict sales performance. Therefore, we wanted to set up SageMaker, different from Flask, we will conduct deeper model training on the SageMaker Instance that requires a heavier workload.

We built a notebook instance in AWS Sagemaker, and we could open the Jupyter Lab in this instance. We successfully imported data and used Python to analyze the data in Sagemaker.



Analysis Examples:

Here are some analytical results.

First, we want to figure out which cities sell the most liquor. This could help us decide demands, target cities and amounts we can supply. We find that Des Moines, Cedar Rapids, and Davenport are the top 3 cities in terms of bottles sold. And these cities will be our potential market to enter.

Top 3 cities in terms of amount sold.

```
result1=pd.value_counts(df.City)
print(result1.head(3))
```

```
DES MOINES      441709
CEDAR RAPIDS    325014
DAVENPORT       222359
```

We also analyze the sales of each category. According to the result below, we find that Vodka and Whiskies are popular in the Iowa market. We could sell these liquor to Iowa.

Top categories in terms of amount sold

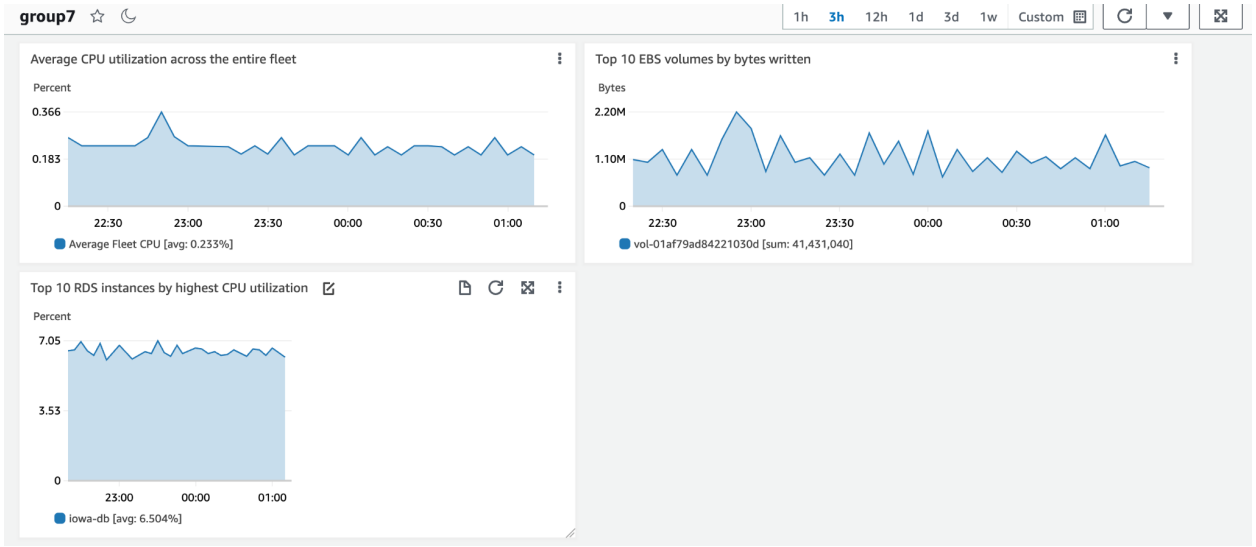
```
pd.value_counts(np.array(df.Category))
```

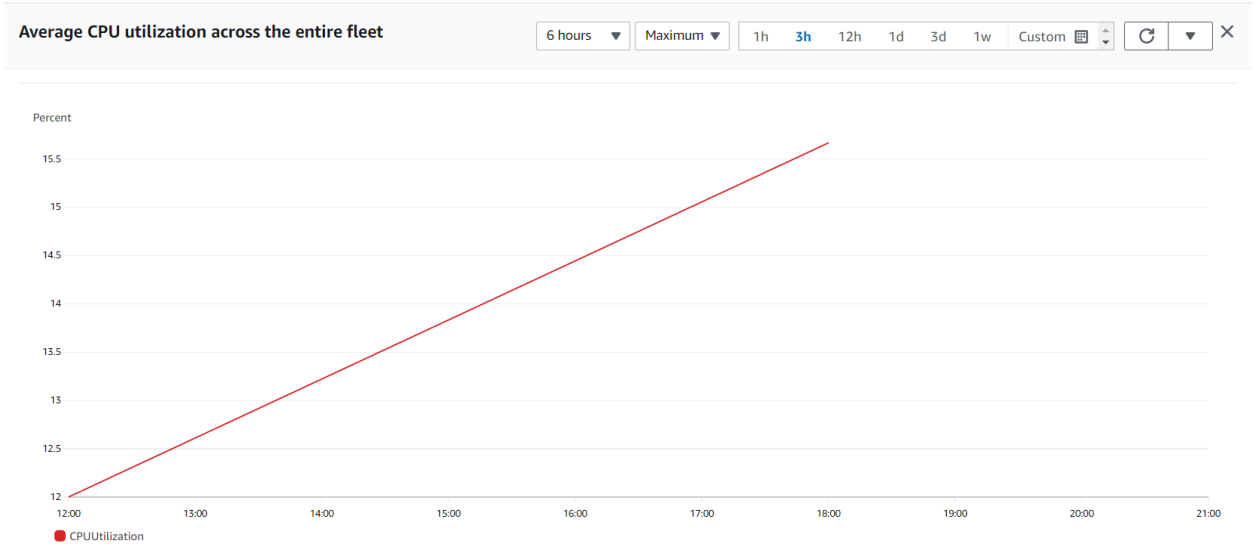
```
1031080.0    625775
1012100.0    469118
1011200.0    273401
1062310.0    267232
1031200.0    251262
              ...
1092100.0         4
1092000.0         4
101220.0          4
1082300.0         1
1089290.0         1
Length: 104, dtype: int64
```

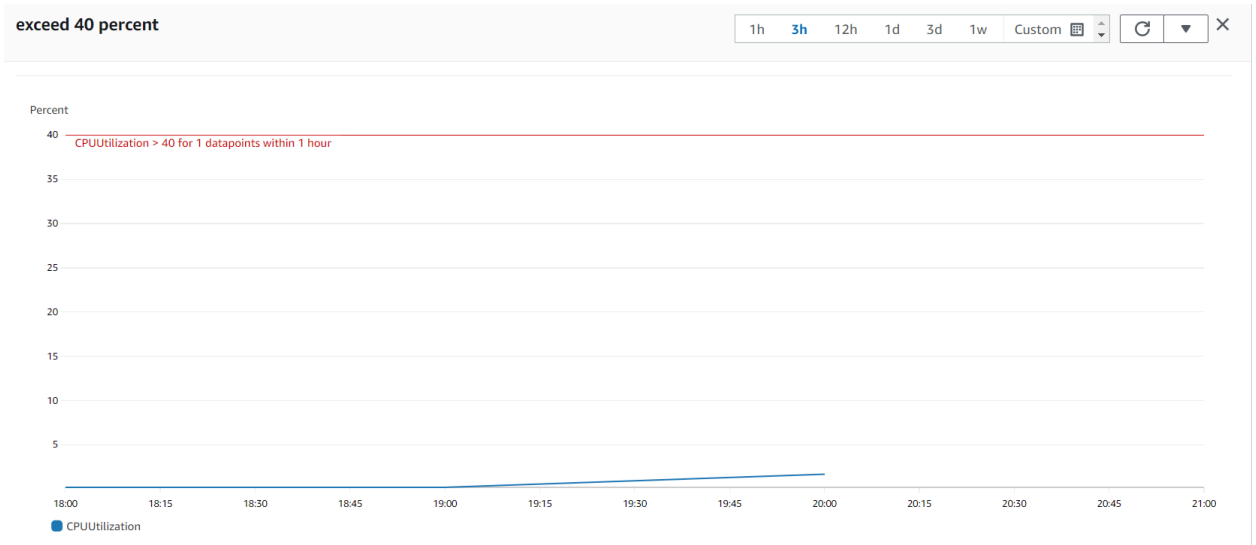
Category Number	Category Name
1031080	VODKA 80 PROOF
1012100	CANADIAN WHISKIES
1011200	STRAIGHT BOURBON WHISKIES
1062310	SPICED RUM
1031200	VODKA FLAVORED

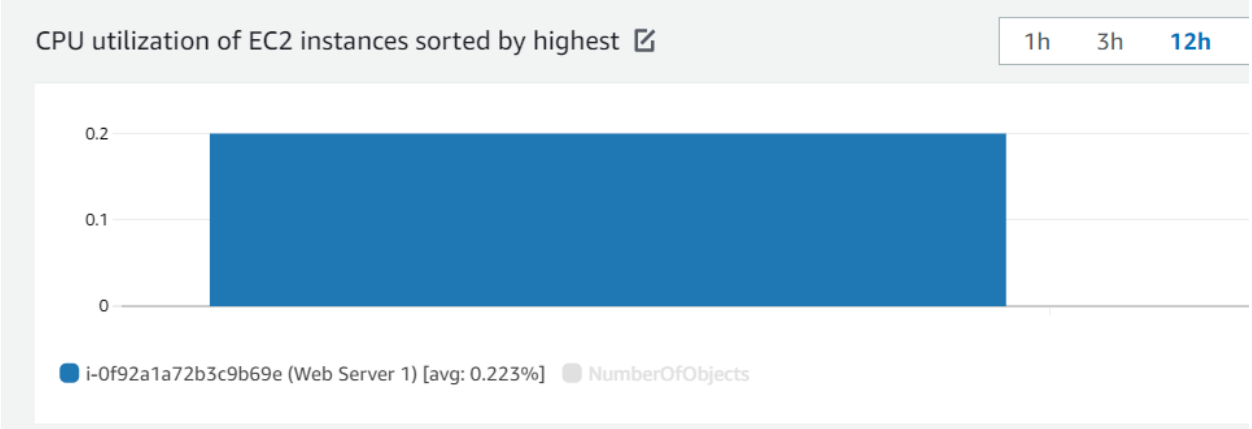
Cloudwatch dashboard and alert:

We can monitor the CPU utilizations of EC2, EBS volumes, etc. on the Cloudwatch dashboard. An alert is also set for when CPU utilization is higher than expected.(We will receive the alert email if CPU utilization is high)


group7
1h 3h 12h 1d 3d 1w Custom
Average CPU utilization across the entire fleet
Percent
0.366
0.183
0
22:30 23:00 23:30 00:00 00:30 01:00
Average Fleet CPU [avg: 0.233%]
Top 10 EBS volumes by bytes written
Bytes
2.20M
1.10M
0
22:30 23:00 23:30 00:00 00:30 01:00
vol-01af79ad84221030d [sum: 41,431,040]
Top 10 RDS instances by highest CPU utilization
Percent
7.05
3.53
0
23:00 00:00 01:00
iowa-db [avg: 6.504%]


Average CPU utilization across the entire fleet
6 hours
Maximum
1h 3h 12h 1d 3d 1w Custom
Percent
15.5
15
14.5
14
13.5
13
12.5
12
12:00 13:00 14:00 15:00 16:00 17:00 18:00 19:00 20:00 21:00
CPUUtilization

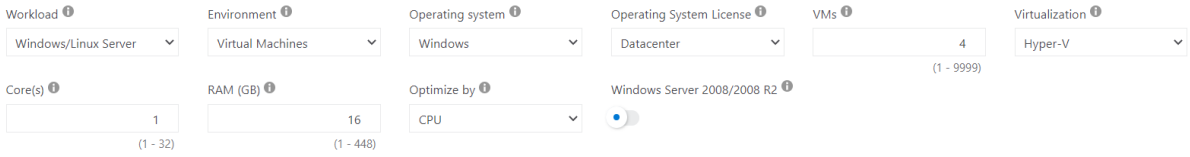

exceed 40 percent
1h 3h 12h 1d 3d 1w Custom
Percent
40
CPUUtilization > 40 for 1 datapoints within 1 hour
35
30
25
20
15
10
5
18:00 18:15 18:30 18:45 19:00 19:15 19:30 19:45 20:00 20:15 20:30 20:45 21:00
CPUUtilization



Cost analysis(1 year)

We choose cloud computing for our project and based on the requirements of analysis, we need these settings and configurations: 1TB Storage, vCPUs: 4, Memory: 16GB, Storage: 1TB. And then we compared Azure with AWS to figure out which one could save money for us.

Azure 1 year Total cost of ownership:



Storage

Enter the details of your on-premises storage infrastructure. After adding storage, select the storage type and enter the remaining details.

Storage 1

Storage type	Disk type	Capacity	Backup	Archive
Local Disk/SAN	HDD	1 TB (1 - 5000)	1 TB (0 - 5000)	1 TB (0 - 5000)

Add storage

Networking

Enter the amount of network bandwidth you currently consume in your on-premises environment.

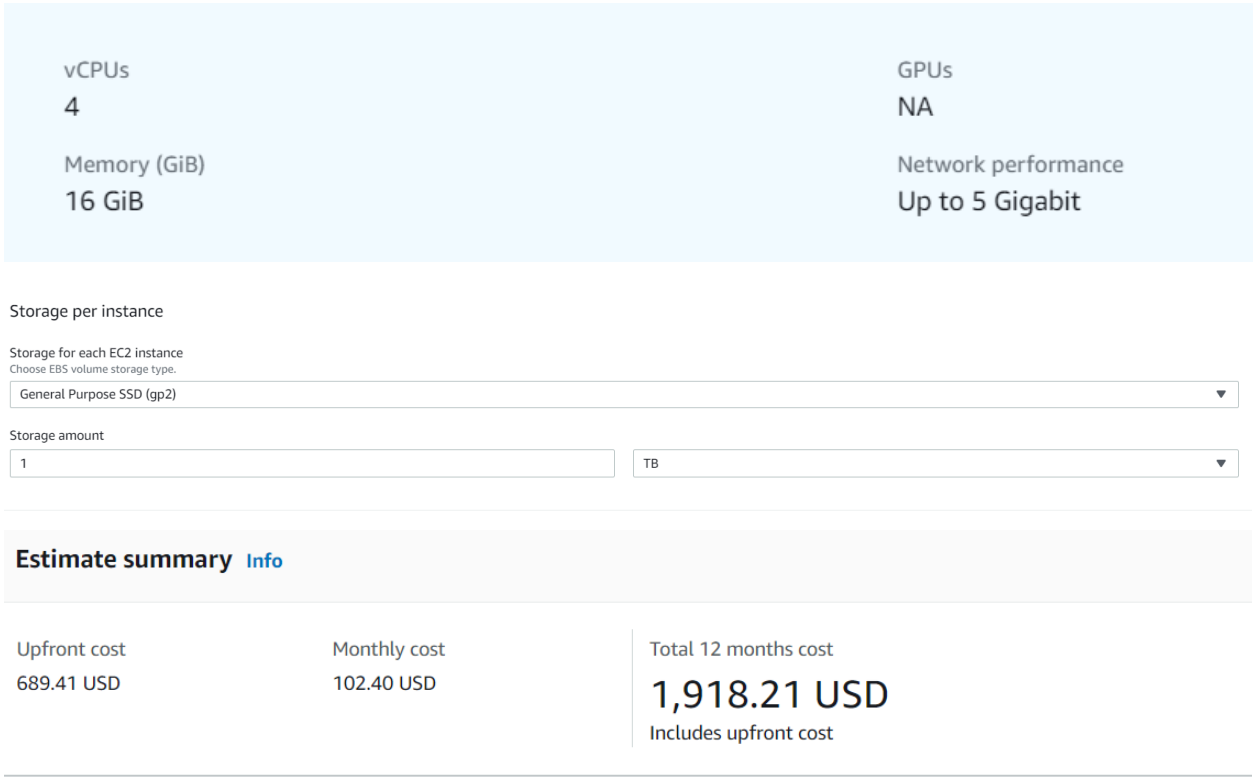
Outbound bandwidth: 1 GB (1 - 2000000)

Cost over 1 year(s)

Azure cost breakdown summary

Category	Cost
Compute	$1,948.56
Data Center	$0.00
Networking	$0.12
Storage	$831.12
IT Labor	$1,533.41
Total	**$4,313.00**

AWS 1 year Total cost of ownership:



Based on calculations, we find AWS could save more than 50% cost for us if we join EC2 1-Year Instance Savings Plans. That's why we chose AWS for our project.

Success criteria

1. Set up VPC
2. Set up security groups
3. Load data into an S3 bucket
4. Data processed and flask running in EC2
5. Modify a role in IAM to allow EC2 and RDS to access S3
6. RDS configured and loaded with processed data
7. Activate flask server that query data and display analytic results
8. Set up Cloudwatch to monitor CPU utilization and send alarms if CPU utilization is high
9. Set up a notebook instance in Sagemaker to analyze data

Summary

Based on the screenshots and descriptions above, we have completed our clouds computing project and met all of the success criteria. We set up a VPC first, loaded data into an S3 bucket and built a flask server running in EC2 to process and analyze data.

After that, we set up a database on RDS and access the data from both the CLI and flask server to display the data and analysis result. The analysis results could help our client better understand the Iowa liquor market. Besides, we also set up a Python notebook instance in Sagemaker to analyze data. To ensure security and privacy, we modified a role in IAM to allow EC2 and RDS to access S3 . We also set up security groups accordingly.

Also, we set up Cloudwatch to monitor CPU utilizations and send alarms if CPU utilization is higher than we expected. For cost analysis, we compared our cost of cloud computing on AWS and Microsoft Azure. We found that AWS's cost is significantly lower than Azure's cost.