Vicente De Leon

Big Data Applications

November 19, 2023

HW7 – AWS

Summary about EC2, S3, RDS, and CloudFormation (purpose, key features, and benefits):

The Elastic Compute Cloud (in case the “EC2WebApp7”) provides secure, resizable compute capacity in the AWS cloud. It is a basic service interface that allows you to obtain and configure capacity with minimal frictions. This service gives you complete control of your computing resources and lets you run on Amazon’s proven computing environment. Some of the benefits I used for my web application was the scalability computing capacity and some security (even though I wish to have much more security for my project). EC2WebApp7 was the EC2 instance I created for my web application, which has a public IPV4 address (which was use for Streamlit) and was configured using a Load Balancer and a Target group along with an IAM role which had all permissions needed for the app’s success.

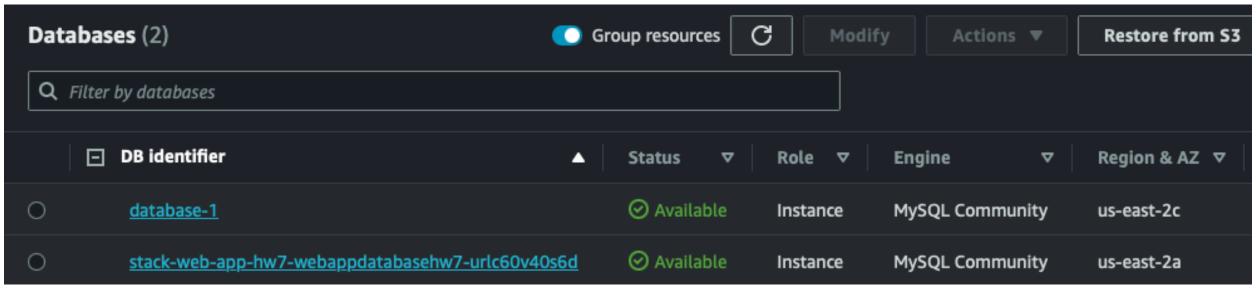
The S3 (in my case “s3-bucket-hw7-webapp”) is an object storage service that ensures scalability, availability security, and performance. Basically, this is where you store your files (within buckets) and retrieve them. This service was used to store csv data files I used to store my files for local testing and EC2 deployment testing functionalities.

The VPC (in my case “VPCWebApp7”) is basically a virtual private cloud used to launch AWS resources in a logically isolated virtual network that the user defines. It resembles how a data center works, with the benefits of using the scalable infrastructure of AWS. In my case, I created the VPCWebApp7 via CloudFormation stack creation to establish a secure, scalable, and highly available network (using three subnets for different availability zones) so I could host my web application on AWS (east-us-2a, east-us-2b, east-us-2c).

CloudFormation, is a service that helps you model and set up your AWS resources so that you can spend less time managing those resources and more time focusing on your application that runs within AWS. This is something I tried to take advantage during the beginning of this assignment. I tried creating my EC2 and RDS instances, as well as the S3 bucket and VPC using CloudFormation. Like I said, this was a huge mistake due to the fact I had no idea how to really work with AWS services. Even though it is a great and extremely useful tool, you need to know what you want and what you are really doing.

Creating RDS instance along with Database and table needed for web app:

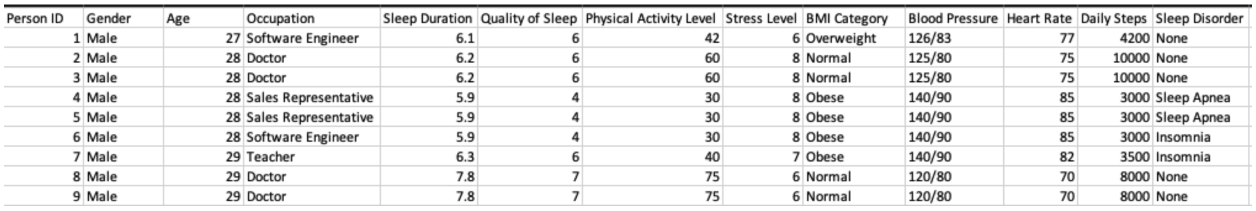
I initially created an RDS instance as well as EC2 using CloudFormation. However, after failing multiple times with the MySQL Workbench connection, the TA helped set up the RDS instance I needed. The RDS instance created. The image below shows my old RDS instance created via CloudFormation and the new RDS instance created thanks to the TA. For this assignment, we are going to be focusing on “database-1”.



Databases (2)

DB identifier	Status	Role	Engine	Region & AZ
database-1	Available	Instance	MySQL Community	us-east-2c
stack-web-app-hw7-webappdatabasehw7-urlc60v40s6d	Available	Instance	MySQL Community	us-east-2a

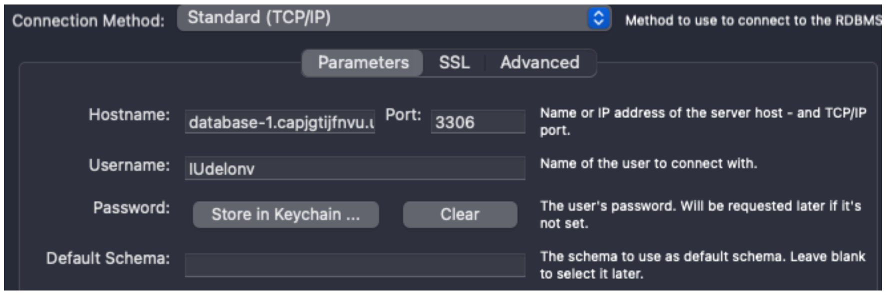
Why I wanted a MySQL connection? The idea behind my app is quite simple yet a little complex if you don’t master the AWS services. My app allows users to interact with S3 and RDS by uploading CSV files to the S3 bucket and then it inserts its content into the RDS table. For Database creation a connection to MySQL or any other MySQL client service was needed. The table within the database will be to store, for example, some sleep health data I found via Kaggle. The user will upload this csv file into S3 bucket, and this will migrate into RDS table:



Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Blood Pressure	Heart Rate	Daily Steps	Sleep Disorder
1	Male	27	Software Engineer	6.1	6	42	6	Overweight	126/83	77	4200	None
2	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75	10000	None
3	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75	10000	None
4	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85	3000	Sleep Apnea
5	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85	3000	Sleep Apnea
6	Male	28	Software Engineer	5.9	4	30	8	Obese	140/90	85	3000	Insomnia
7	Male	29	Teacher	6.3	6	40	7	Obese	140/90	82	3500	Insomnia
8	Male	29	Doctor	7.8	7	75	6	Normal	120/80	70	8000	None
9	Male	29	Doctor	7.8	7	75	6	Normal	120/80	70	8000	None

To set up a MySQL Workbench and RDS instance connection, I used the following resource:
https://aws.amazon.com/getting-started/hands-on/create-mysql-db/

The image below shows the Hostname (database-1 Endpoint), Username (the username I gave my old RDS stack instance via stack formation), as well as the password for old RDS instance. This step was crucial for me.



Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

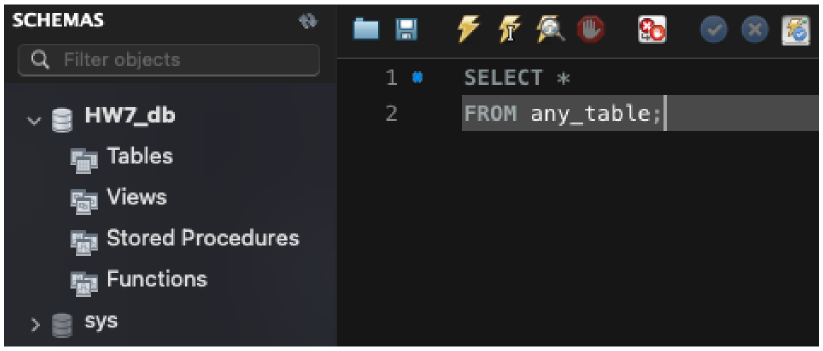
Hostname: database-1.capjgtijfnvu.u Port: 3306 — Name or IP address of the server host - and TCP/IP port.

Username: lUdelonv — Name of the user to connect with.

Password: Store in Keychain ... Clear — The user's password. Will be requested later if it's not set.

Default Schema: — The schema to use as default schema. Leave blank to select it later.

For database creation, I will be using what I learned from Professors Scrivner class "Applied Database Technologies". "HW7_db" database was set as default schema as the "container" for the table or any table that will be created based on the dataframe being used within the web application.



```
SELECT *
FROM any_table;
```

Important key points here to further continue with the process:

- The TA helped create a new security group called "WebAppSG".
- Database-1 resided within VPC: vpc-00cc3bf492505007e.
- The CIDR is 172.31.0.0/16 which is my address type IPv4.
- The security group ID associated with database-1 is: sg-09856ba64c63b9b1e.
- Next step will be modifying certain key points to match the new VPC I will be creating via CloudFormation. I need RDS and EC2 instance to reside within the same VPC for AWS services interactions. This is because after multiple attempts, I understood that you can't just simple change the VPC of an existing RDS instance ("database-1") directly. I will need to change the subnets the RDS instance is associated with.

VPC CloudFormation creation (JSON template):

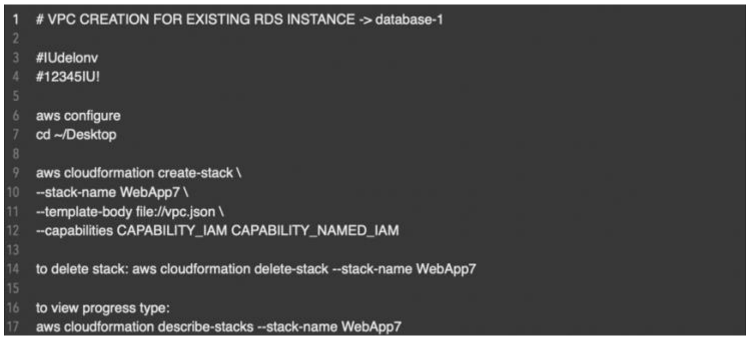
I hope this template contains the necessary components that will allow my web application to interact with MySQL and most importantly the internet. Since I don't really know AWS services that much, I know most of the things here are being kind of brute forced. However, for future projects I will ensure to make it more reliable in terms of security and development/production. The JSON template contains the following:

- VPC (WebAppVPCHW7): This is the VPC being crated, which my private network. I am defining CIDR block to "172.31.0.0/16" because that's the configuration I have on the VPC created for the RDS instance (database-1) the TA helped me set up. This CIDR defines the IP address range for the resources within this VPC.
- Subnets (WebAppSubnet1HW7, WebAppSubnet2HW7, WebAppSubnet3HW7): these are the 3 subnets created (to follow AWS console logic) where I can place the RDS database-1 instance and eventually the EC2 instances I will create. This just for high availability purposes. The idea is to have public subnets that can be access via internet (EC2 instance hosting the web server) and private subnets that can't be directly accessible from the internet (database-1).
- The Internet Gateway (WebAppInternetGatewayHW7): this is communication between the VPC and the internet.
- VPC Gateway Attachment (WebAppAttachVPCGatewayHW7): this will attach the internet gateway wot the VPC.

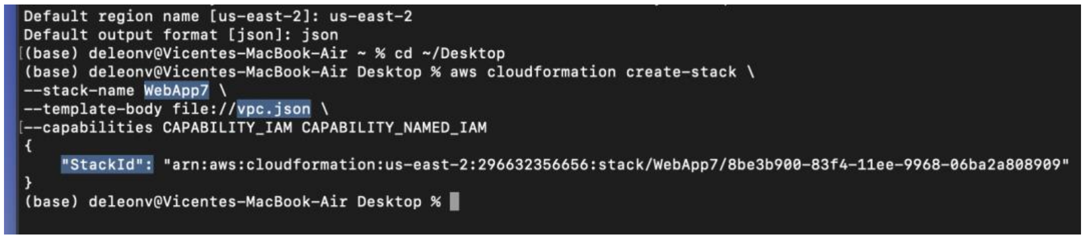
- Route Table (WebAppRouteTableHW7): this route table contains the set of rules to direct network traffic from the subnets to destinations outside the VPC. This includes the route WebAppRouteHW7, which directs all traffic for the internet (0.0.0.0/0) to the internet gateway, allowing instances in the subnets to access the internet.
- Route Associations (SubnetRouteTableAssociation1HW7, SubnetRouteTableAssociation2HW7, SubnetRouteTableAssociation3HW7): these associations will ensure that each subnet adheres to the routing defined in the Route Table.
- Security Group (WebAppSecurityGroupHW7): The ingress rule will allow MySQL traffic (TCP port 3306) from an IP address, which is necessary for my web app to communicate with RDS MySQL instances within the VPC. The egress rule allows all outbound traffic, allowing instances to initiate communication with any external service or endpoint.

This configuration will set up the necessary network infrastructure to support my web application's interaction with AWS services (RDS and EC2 instances). With this, I intent to host my web application on an EC2 instance in a public subnet, which allows it to be accesses from the internet. At the same time, it can interreact with RDS instance (database-1) that will also be hosted within the VPC. I made my database-1 instance public for the MySQL Workbench connection. However, it is important to try access this RDS instance only via VPC.

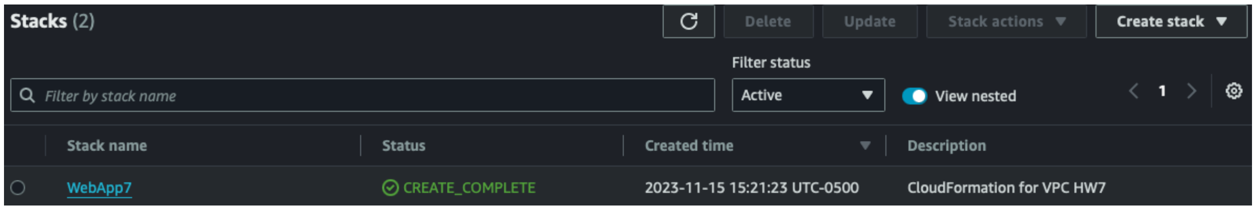
CloudFormation CLI command:



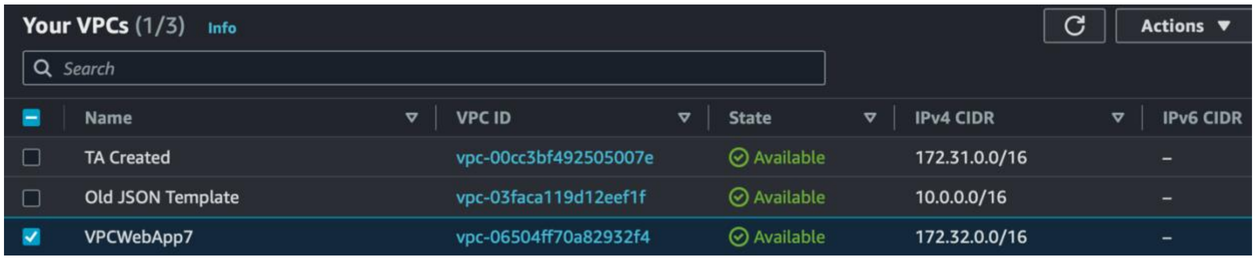
```
# VPC CREATION FOR EXISTING RDS INSTANCE -> database-1

#lUdelonv
#12345IU!

aws configure
cd ~/Desktop

aws cloudformation create-stack \
--stack-name WebApp7 \
--template-body file://vpc.json \
--capabilities CAPABILITY_IAM CAPABILITY_NAMED_IAM

to delete stack: aws cloudformation delete-stack --stack-name WebApp7

to view progress type:
aws cloudformation describe-stacks --stack-name WebApp7
```



```
Default region name [us-east-2]: us-east-2
Default output format [json]: json
(base) deleonv@Vicentes-MacBook-Air ~ % cd ~/Desktop
(base) deleonv@Vicentes-MacBook-Air Desktop % aws cloudformation create-stack \
--stack-name WebApp7 \
--template-body file://vpc.json \
--capabilities CAPABILITY_IAM CAPABILITY_NAMED_IAM
{
    "StackId": "arn:aws:cloudformation:us-east-2:296632356656:stack/WebApp7/8be3b900-83f4-11ee-9968-06ba2a808909"
}
(base) deleonv@Vicentes-MacBook-Air Desktop %
```



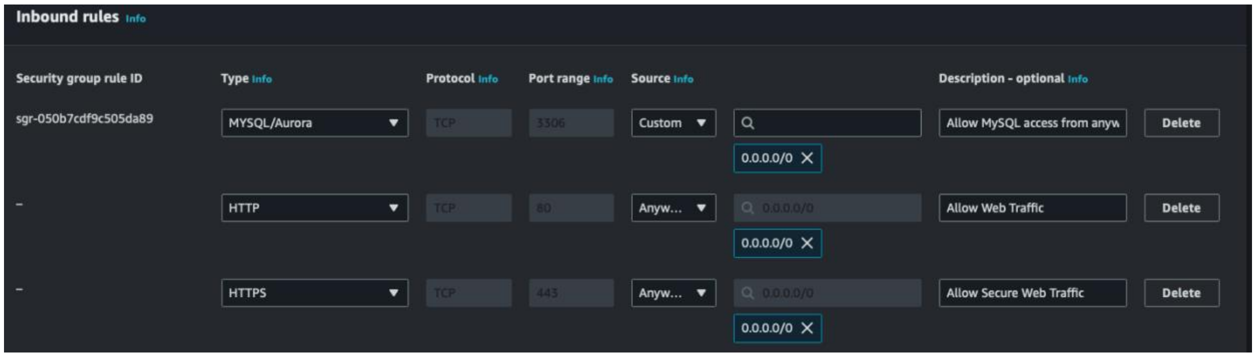
Stack name	Status	Created time	Description
WebApp7	CREATE_COMPLETE	2023-11-15 15:21:23 UTC-0500	CloudFormation for VPC HW7



Your VPCs (1/3)

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR
TA Created	vpc-00cc3bf492505007e	Available	172.31.0.0/16	–
Old JSON Template	vpc-03faca119d12eef1f	Available	10.0.0.0/16	–
✓ VPCWebApp7	vpc-06504ff70a82932f4	Available	172.32.0.0/16	–

Security Group: StreamlitWebAppSG (Created via CloudFormation):

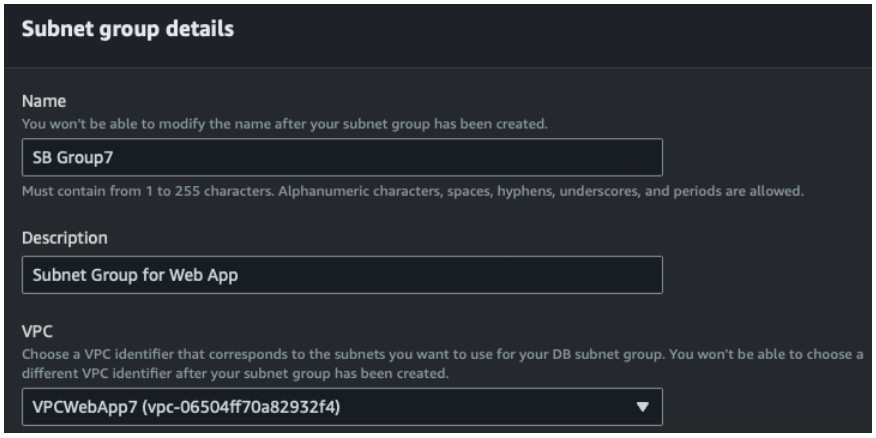


Inbound rules

Security group rule ID	Type	Protocol	Port range	Source		Description - optional
sgr-050b7cdf9c505da89	MYSQL/Aurora	TCP	3306	Custom	0.0.0.0/0	Allow MySQL access from anyw
–	HTTP	TCP	80	Anyw...	0.0.0.0/0	Allow Web Traffic
–	HTTPS	TCP	443	Anyw...	0.0.0.0/0	Allow Secure Web Traffic

Inbound rules explained: the first rule allows MySQL traffic on port 3306 from any IP address, which was needed for MySQL database connection. The second rule HTTP traffic on port 80 from any IP address (standard for web browsers). The third rule allows HTTPS traffic on port 443 from any IP address (secure web browser connection). I manually added the last two rules using the AWS console. Outbound rule will not be changed.

Now, let's go to the RDS Dashboard and select the Subnet Groups:

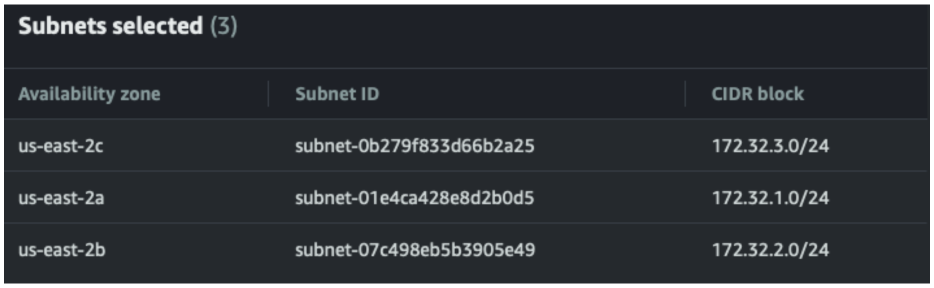


Subnet group details

Name
You won't be able to modify the name after your subnet group has been created.
SB Group7
Must contain from 1 to 255 characters. Alphanumeric characters, spaces, hyphens, underscores, and periods are allowed.

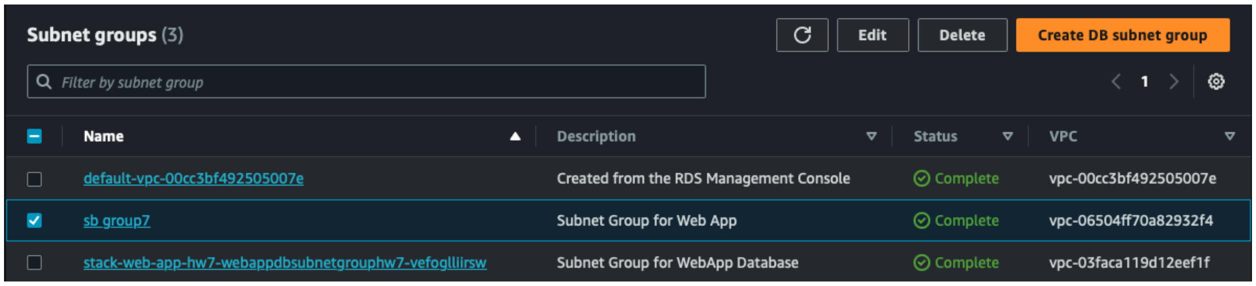
Description
Subnet Group for Web App

VPC
Choose a VPC identifier that corresponds to the subnets you want to use for your DB subnet group. You won't be able to choose a different VPC identifier after your subnet group has been created.
VPCWebApp7 (vpc-06504ff70a82932f4)



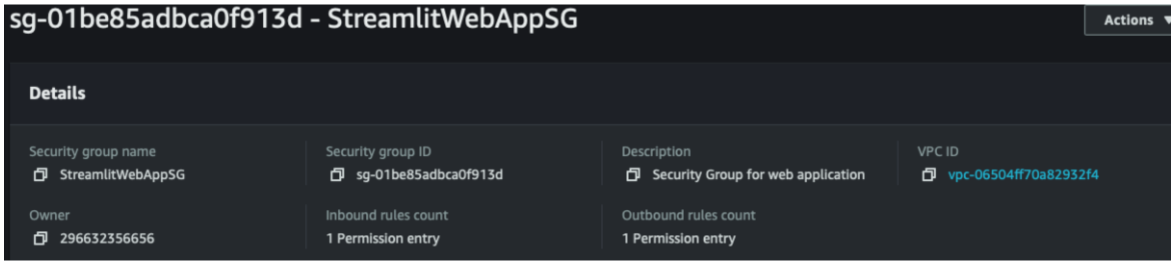
Subnets selected (3)

Availability zone	Subnet ID	CIDR block
us-east-2c	subnet-0b279f833d66b2a25	172.32.3.0/24
us-east-2a	subnet-01e4ca428e8d2b0d5	172.32.1.0/24
us-east-2b	subnet-07c498eb5b3905e49	172.32.2.0/24


Subnet groups (3)

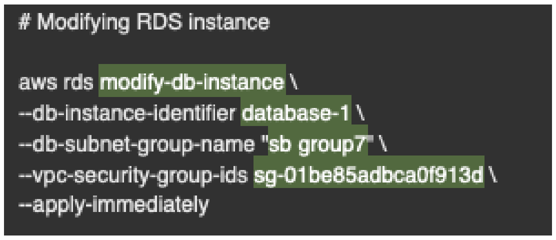
	Name	Description	Status	VPC
☐	default-vpc-00cc3bf492505007e	Created from the RDS Management Console	Complete	vpc-00cc3bf492505007e
☑	sb group7	Subnet Group for Web App	Complete	vpc-06504ff70a82932f4
☐	stack-web-app-hw7-webappdbsubnetgrouphw7-vefoglliirsw	Subnet Group for WebApp Database	Complete	vpc-03faca119d12eef1f

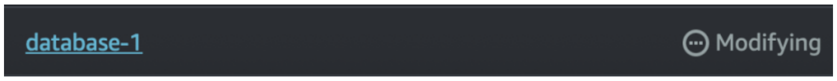
The non-selected subnets (image above) are just failed attempts from previous unfortunate experiments. After this, I went ahead to the RDS Dashboard to Modify "database-1". Within databse-1, I change the subnet group to "sb group7" and the security group to "StreamlitWebAppSG" (sg-01be85adbca0f913d). "StreamlitWebApp" is the name I gave to the security group via CloudFormation stack creation. Also, it is important to mentioned that the security group is associated to the VPC created via CloudFormation.

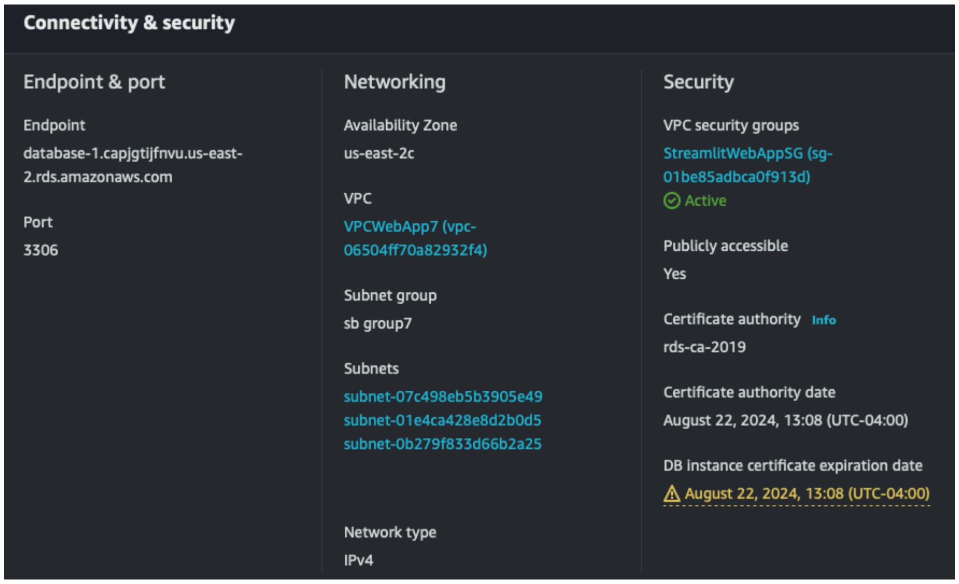

sg-01be85adbca0f913d - StreamlitWebAppSG

Details

Security group name: StreamlitWebAppSG	Security group ID: sg-01be85adbca0f913d	Description: Security Group for web application	VPC ID: vpc-06504ff70a82932f4
Owner: 296632356656	Inbound rules count: 1 Permission entry	Outbound rules count: 1 Permission entry	


```
# Modifying RDS instance

aws rds modify-db-instance \
--db-instance-identifier database-1 \
--db-subnet-group-name "sb group7" \
--vpc-security-group-ids sg-01be85adbca0f913d \
--apply-immediately
```


database-1 Modifying


Connectivity & security

Endpoint & port

Endpoint: database-1.capjgtijfnvu.us-east-2.rds.amazonaws.com

Port: 3306

Networking

Availability Zone: us-east-2c

VPC: VPCWebApp7 (vpc-06504ff70a82932f4)

Subnet group: sb group7

Subnets: subnet-07c498eb5b3905e49, subnet-01e4ca428e8d2b0d5, subnet-0b279f833d66b2a25

Network type: IPv4

Security

VPC security groups: StreamlitWebAppSG (sg-01be85adbca0f913d) Active

Publicly accessible: Yes

Certificate authority: rds-ca-2019

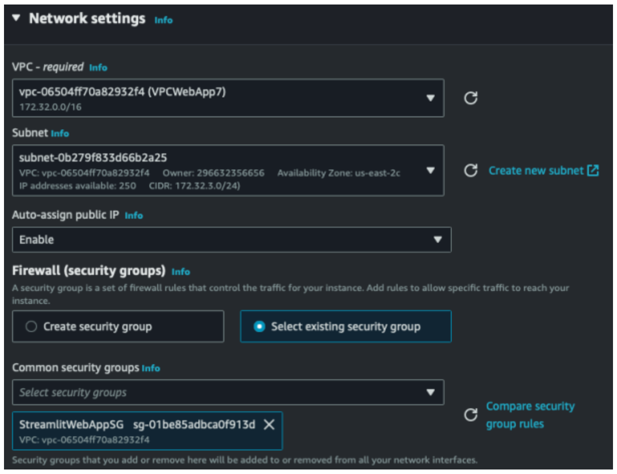
Certificate authority date: August 22, 2024, 13:08 (UTC-04:00)

DB instance certificate expiration date: August 22, 2024, 13:08 (UTC-04:00)

For the above RDS instance modification, I had to use the above CLI command, because when I tried to manually modify database-1 RDS instance, the security group that I needed (StreamlitWebAppSG ID: sg-01be85adbca0f913d) was not available within the options. I tried to refresh the console and web browser, but it didn't appear. So, I had to use the above CLI command to do it, which it worked.

Creating EC2 instance:

For the EC2 instance, I will be using the AWS console and not a CLI command because I need to see what I am selecting in terms of services. So, I launched a new Instance and made the following configuration. You can see that the VPC was set to the VPC created via CloudFormation (VPCWebApp7) along with its subnets, the StreamlitWebAppSG security group, and the IAM role (WebAppHW7) I created:



Network settings Info

VPC - *required* Info
vpc-06504ff70a82932f4 (VPCWebApp7)
172.32.0.0/16

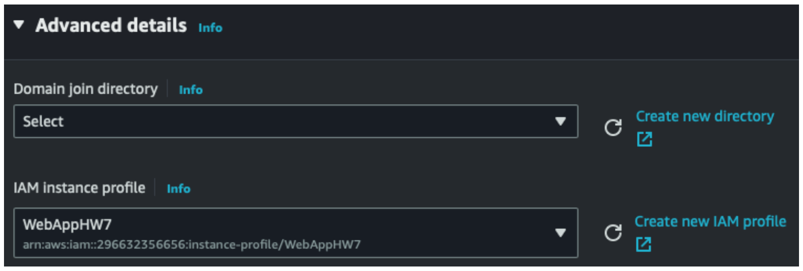
Subnet Info
subnet-0b279f833d66b2a25
VPC: vpc-06504ff70a82932f4   Owner: 296632356656   Availability Zone: us-east-2c
IP addresses available: 250   CIDR: 172.32.3.0/24)

Auto-assign public IP Info
Enable

Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group [✓ Select existing security group]

Common security groups Info
Select security groups
StreamlitWebAppSG sg-01be85adbca0f913d
VPC: vpc-06504ff70a82932f4
Security groups that you add or remove here will be added to or removed from all your network interfaces.

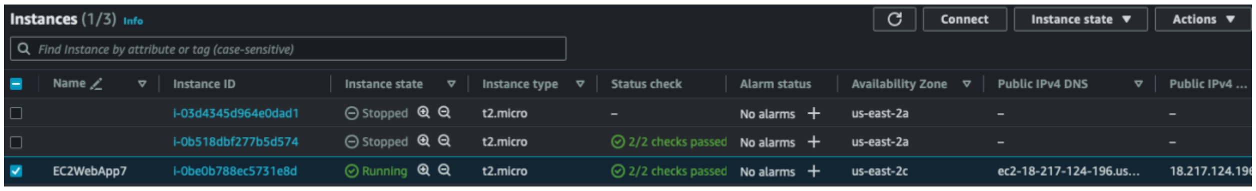


Advanced details Info

Domain join directory Info
Select

IAM instance profile Info
WebAppHW7
arn:aws:iam::296632356656:instance-profile/WebAppHW7

You can see EC2 instance "EC2WebApp7" created:



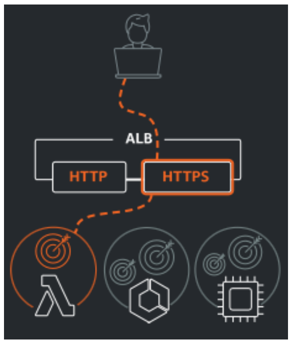
Instances (1/3) Info

Find Instance by attribute or tag (case-sensitive)

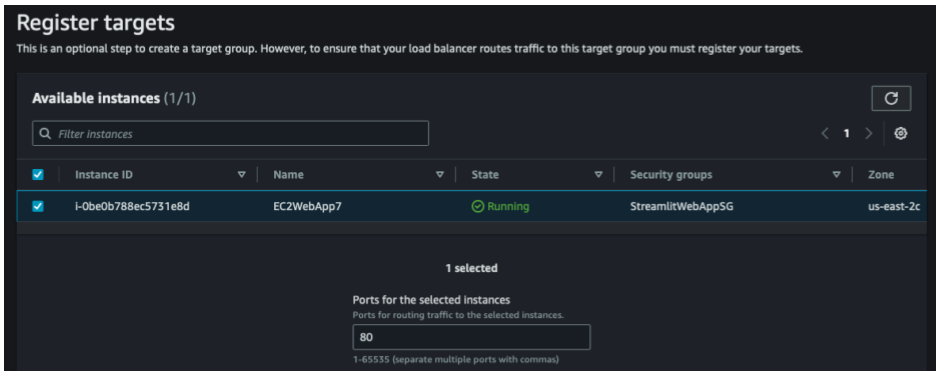
	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
☐		i-03d4345d964e0dad1	Stopped	t2.micro	–	No alarms	us-east-2a	–	–
☐		i-0b518dbf277b5d574	Stopped	t2.micro	2/2 checks passed	No alarms	us-east-2a	–	–
✓	EC2WebApp7	i-0be0b788ec5731e8d	Running	t2.micro	2/2 checks passed	No alarms	us-east-2c	ec2-18-217-124-196.us...	18.217.124.19

High traffic and Scalability:

- Application Load Balancer (LoadBalancerWebApp7): this service distributes incoming HTTP and HTTPS traffic across multiple targets such as the EC2 instances, containers, and other services, based on request attributes. When the Load Balancer receives a connection request, it evaluated the Listener rules in priority order to determine which rule to apply, and if applicable, it selects a target from the Target Group for the rule of actions. The load balancer will serve as an invisible intermediary between the user and the server group, ensuring that all resources servers are used equally.



- Target Group (TargetGroupWebApp7): this service will define a group of resources, such as the EC2 instance, that the Load Balancer should direct traffic to. It's set to target type Instances, to listen on port 80 (HTTP), and it is associated to VPCWebApp7 (created via CloudFormation). Then, I proceed to register my target and associate the EC2WebApp7 instance to the Target Group.



Register targets

This is an optional step to create a target group. However, to ensure that your load balancer routes traffic to this target group you must register your targets.

Available instances (1/1)

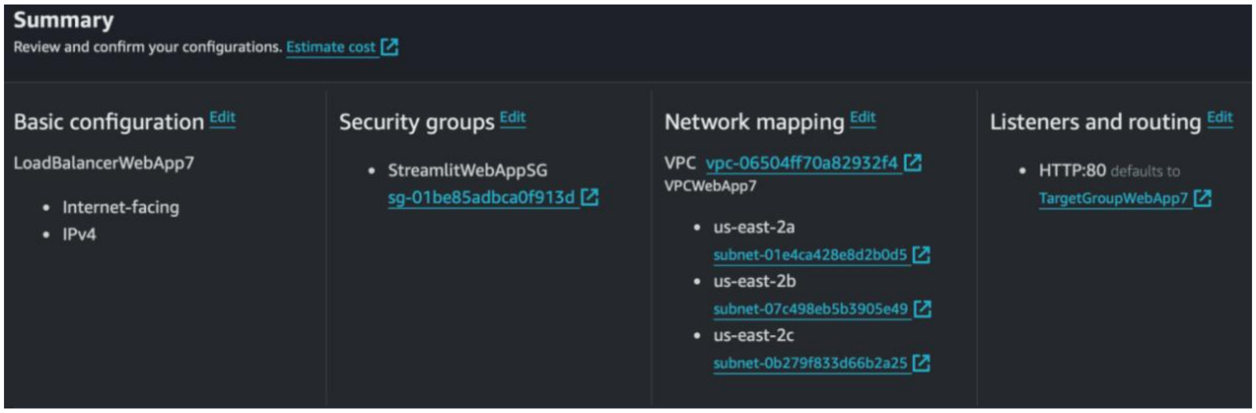
✓	Instance ID	Name	State	Security groups	Zone
✓	i-0be0b788ec5731e8d	EC2WebApp7	Running	StreamlitWebAppSG	us-east-2c

1 selected

Ports for the selected instances
Ports for routing traffic to the selected instances.
80
1-65535 (separate multiple ports with commas)

- Listener: the Listener is just a process that checks for connections requests, using the protocol and port that I just configured. In this case, it waits for incoming traffic on the Load Balancer (HTTP) and forwards it the Target Group.

Load Balancer summary:



Summary
Review and confirm your configurations. Estimate cost

Basic configuration Edit
LoadBalancerWebApp7
- Internet-facing
- IPv4

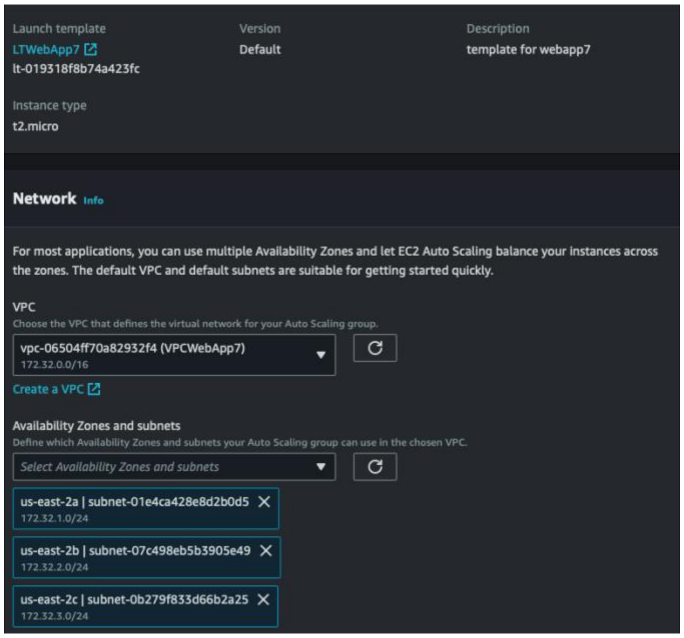
Security groups Edit
- StreamlitWebAppSG
  sg-01be85adbca0f913d

Network mapping Edit
VPC vpc-06504ff70a82932f4
VPCWebApp7
- us-east-2a
  subnet-01e4ca428e8d2b0d5
- us-east-2b
  subnet-07c498eb5b3905e49
- us-east-2c
  subnet-0b279f833d66b2a25

Listeners and routing Edit
- HTTP:80 defaults to
  TargetGroupWebApp7

Auto Scaling Group and Launch Template for Scalability and High traffic:

Launch template configurations within Auto Scaling Group:



Launch template	Version	Description
LTWebApp7 lt-019318f8b74a423fc	Default	template for webapp7

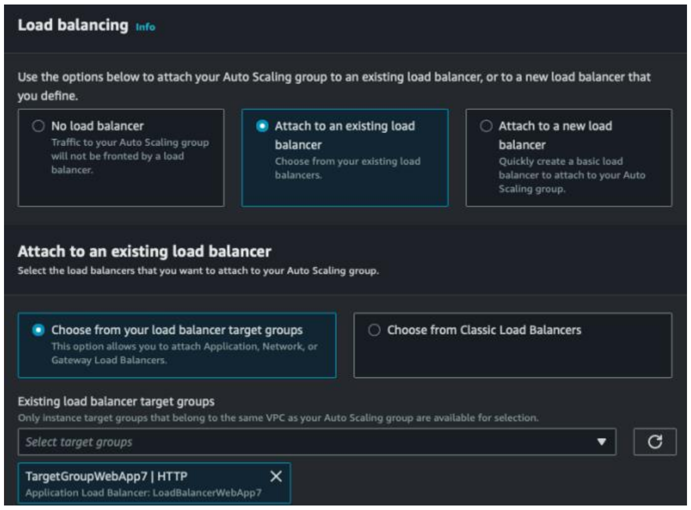
Instance type
t2.micro

Network Info

For most applications, you can use multiple Availability Zones and let EC2 Auto Scaling balance your instances across the zones. The default VPC and default subnets are suitable for getting started quickly.

VPC
Choose the VPC that defines the virtual network for your Auto Scaling group.
vpc-06504ff70a82932f4 (VPCWebApp7)
172.32.0.0/16
Create a VPC

Availability Zones and subnets
Define which Availability Zones and subnets your Auto Scaling group can use in the chosen VPC.
Select Availability Zones and subnets
- us-east-2a | subnet-01e4ca428e8d2b0d5 172.32.1.0/24
- us-east-2b | subnet-07c498eb5b3905e49 172.32.2.0/24
- us-east-2c | subnet-0b279f833d66b2a25 172.32.3.0/24



Load balancing Info

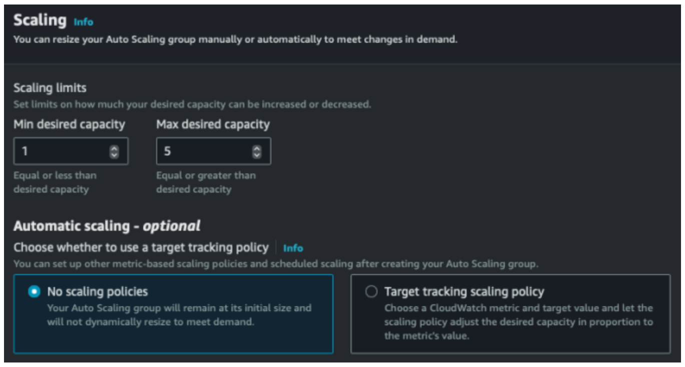
Use the options below to attach your Auto Scaling group to an existing load balancer, or to a new load balancer that you define.

- No load balancer — Traffic to your Auto Scaling group will not be fronted by a load balancer.
- [✓ Attach to an existing load balancer] — Choose from your existing load balancers.
- Attach to a new load balancer — Quickly create a basic load balancer to attach to your Auto Scaling group.

Attach to an existing load balancer
Select the load balancers that you want to attach to your Auto Scaling group.

- [✓ Choose from your load balancer target groups] — This option allows you to attach Application, Network, or Gateway Load Balancers.
- Choose from Classic Load Balancers

Existing load balancer target groups
Only instance target groups that belong to the same VPC as your Auto Scaling group are available for selection.
Select target groups
TargetGroupWebApp7 | HTTP
Application Load Balancer: LoadBalancerWebApp7



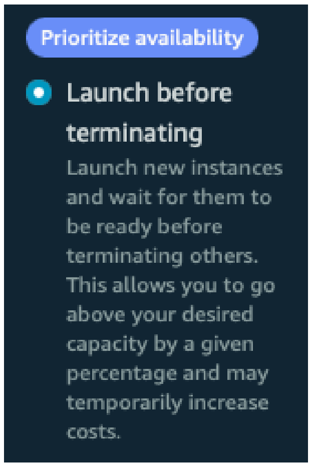
Scaling Info
You can resize your Auto Scaling group manually or automatically to meet changes in demand.

Scaling limits
Set limits on how much your desired capacity can be increased or decreased.

Min desired capacity	Max desired capacity
1	5
Equal or less than desired capacity	Equal or greater than desired capacity

Automatic scaling - *optional*
Choose whether to use a target tracking policy Info
You can set up other metric-based scaling policies and scheduled scaling after creating your Auto Scaling group.

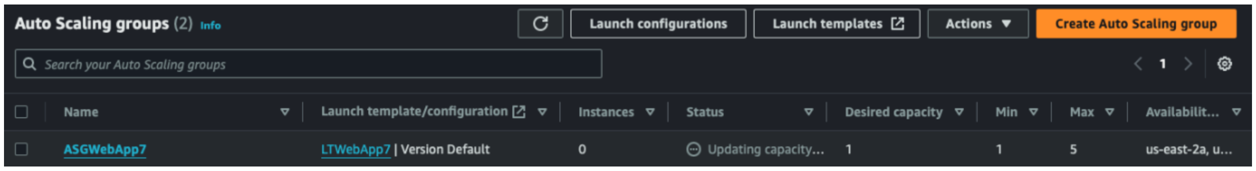
- [✓ No scaling policies] — Your Auto Scaling group will remain at its initial size and will not dynamically resize to meet demand.
- Target tracking scaling policy — Choose a CloudWatch metric and target value and let the scaling policy adjust the desired capacity in proportion to the metric's value.



Prioritize availability

[✓ Launch before terminating]
Launch new instances and wait for them to be ready before terminating others. This allows you to go above your desired capacity by a given percentage and may temporarily increase costs.

ASGWebApp7 (Auto Scaling Group):

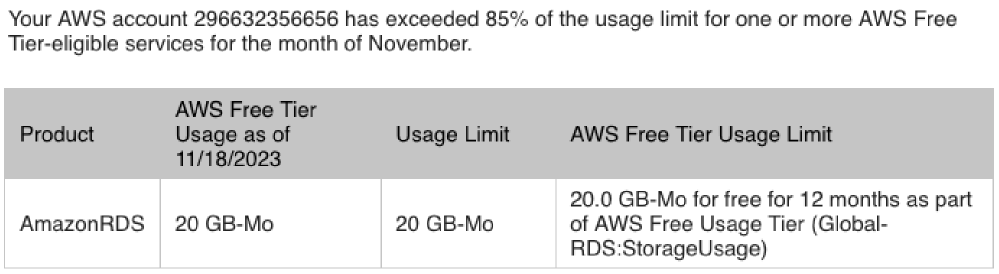


Auto Scaling groups (2) Info

Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availabilit...
ASGWebApp7	LTWebApp7 | Version Default	0	Updating capacity...	1	1	5	us-east-2a, u...

In Scaling capacities, the min desired capacity was scaled to 1 (ensures I have at least one instance always running) and max desired capacity to 5 instances to handle traffic. I set these numbers for high traffic and scalability. You could see all my configuration being related to the VPC created via CloudFormation. Even though I set min to 1 and max to 5, I change these setting to min 1 and max 1.

This is because I received and email warning from AWS which stated I had exceeded 85% of the usage limit for AWS free tier services as shown in the image below:


Your AWS account 296632356656 has exceeded 85% of the usage limit for one or more AWS Free Tier-eligible services for the month of November.

Product	AWS Free Tier Usage as of 11/18/2023	Usage Limit	AWS Free Tier Usage Limit
AmazonRDS	20 GB-Mo	20 GB-Mo	20.0 GB-Mo for free for 12 months as part of AWS Free Usage Tier (Global-RDS:StorageUsage)

I haven't tested my web application yet; however, I hope it works. I just hope I don't run into any issues with my free tier service.

Streamlit Web Application:

I decided to create a Streamlit Web Application because I have experience using Streamlit. I really like how easy and user friendly it is. Even my personal Data Science Portfolio was coded using Streamlit. I don't have to worry about the front-end coding, just the back-end coding. As always, I will provide the code I used to create this application within the final submission. The code structure is the following:

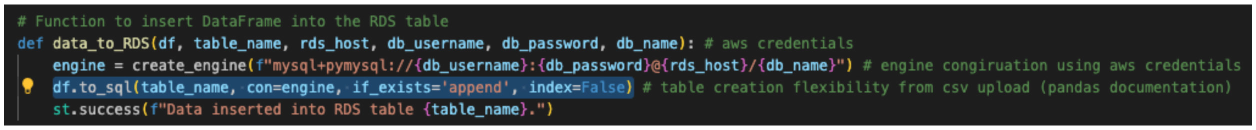
Functions:

- Data_to_RDS: inserts df into my RDS table (if table was created within databse).
- Upload_S3_RDS: handles CSV file upload to my S3 bucket, reads into DataFrame, and uses data_to_RDS function to insert data into my RDS table.
- S3_files: just the files within my S3 bucket.
- Download_S3_csv: function download data within my S3 bucket as CSV format.
- Display_RDS: takes data from my RDS table and displays it into DataFrame.
- Download_RDS_csv: function to download RDS data into csv format.

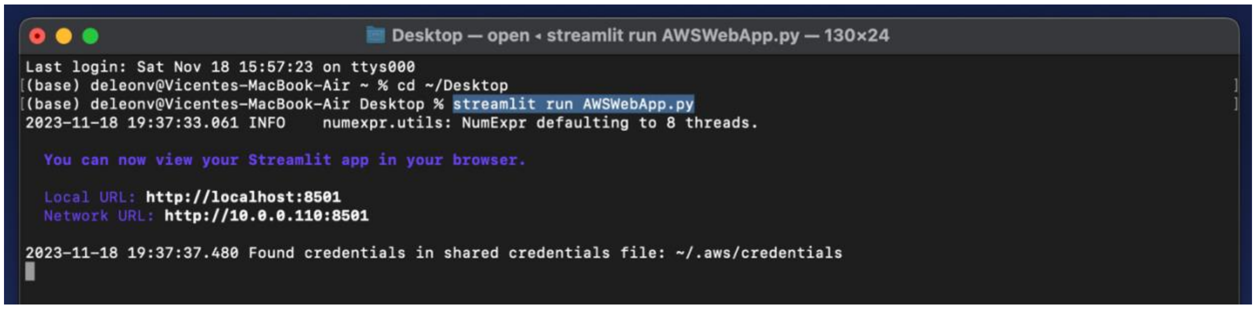
Streamlit App:

The app contains two tabs: S3 Operations (uploading data, populating database-1 table based on csv, downloads files within S3 bucket) and RDS Operations (Gets populated from S3 bucket files and has the operation to download located within databse-1 table). This app basically serves as a user-friendly interface that interacts with AWS services such as S3 and RDS, allowing the user to upload, manage, and download csv data. Even though AWS credentials are the heart of this web application, I don't think it is a good idea (because we are trying to make it as secure as possible) to code the AWS credentials within the Streamlit code (I will probably be sharing this app and code within my own Streamlit Cloud and GitHub).

The following code provides the flexibility of creating a new table based on the csv file the user uploads (if the table does not exist already). I could've created the table within MySQL Workbench, but I think it's more interesting to see a table being automatically created and populated by a csv upload.

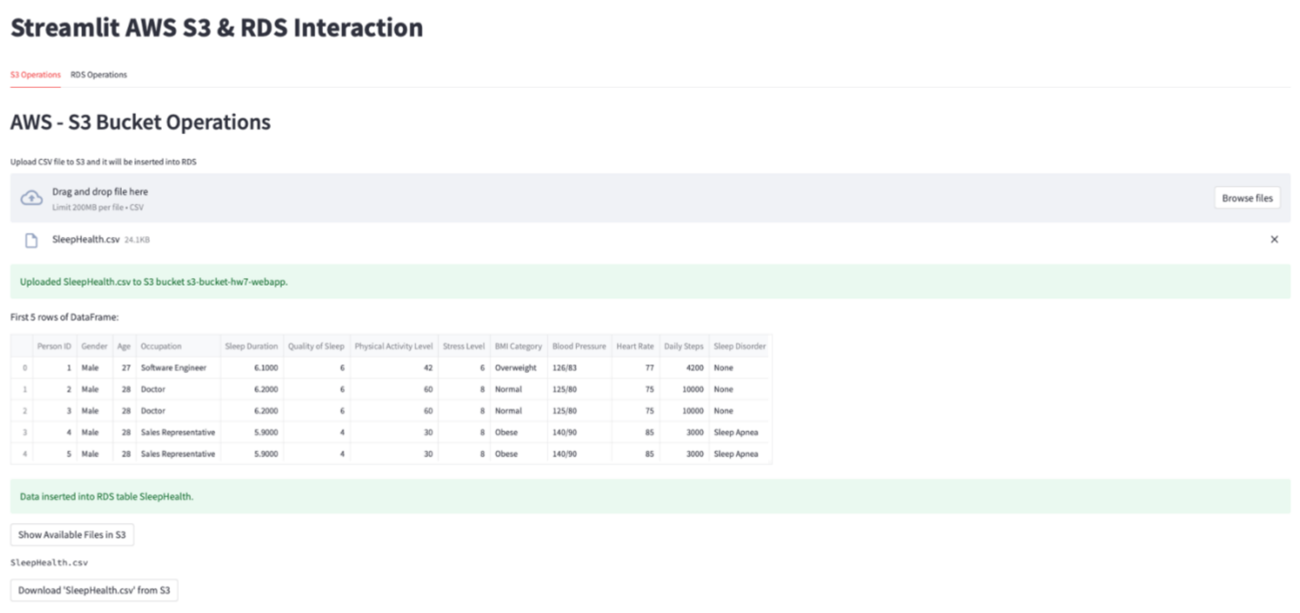

```python
# Function to insert DataFrame into the RDS table
def data_to_RDS(df, table_name, rds_host, db_username, db_password, db_name): # aws credentials
    engine = create_engine(f"mysql+pymysql://{db_username}:{db_password}@{rds_host}/{db_name}") # engine congiruation using aws credentials
    df.to_sql(table_name, con=engine, if_exists='append', index=False) # table creation flexibility from csv upload (pandas documentation)
    st.success(f"Data inserted into RDS table {table_name}.")
```

Testing AWSWebApp.py locally from Desktop:



```
Last login: Sat Nov 18 15:57:23 on ttys000
(base) deleonv@Vicentes-MacBook-Air ~ % cd ~/Desktop
(base) deleonv@Vicentes-MacBook-Air Desktop % streamlit run AWSWebApp.py
2023-11-18 19:37:33.061 INFO    numexpr.utils: NumExpr defaulting to 8 threads.

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://10.0.0.110:8501

2023-11-18 19:37:37.480 Found credentials in shared credentials file: ~/.aws/credentials
```

Tab 1: SE Operations



AWS - S3 Bucket Operations

Upload CSV file to S3 and it will be inserted into RDS

SleepHealth.csv 24.1KB

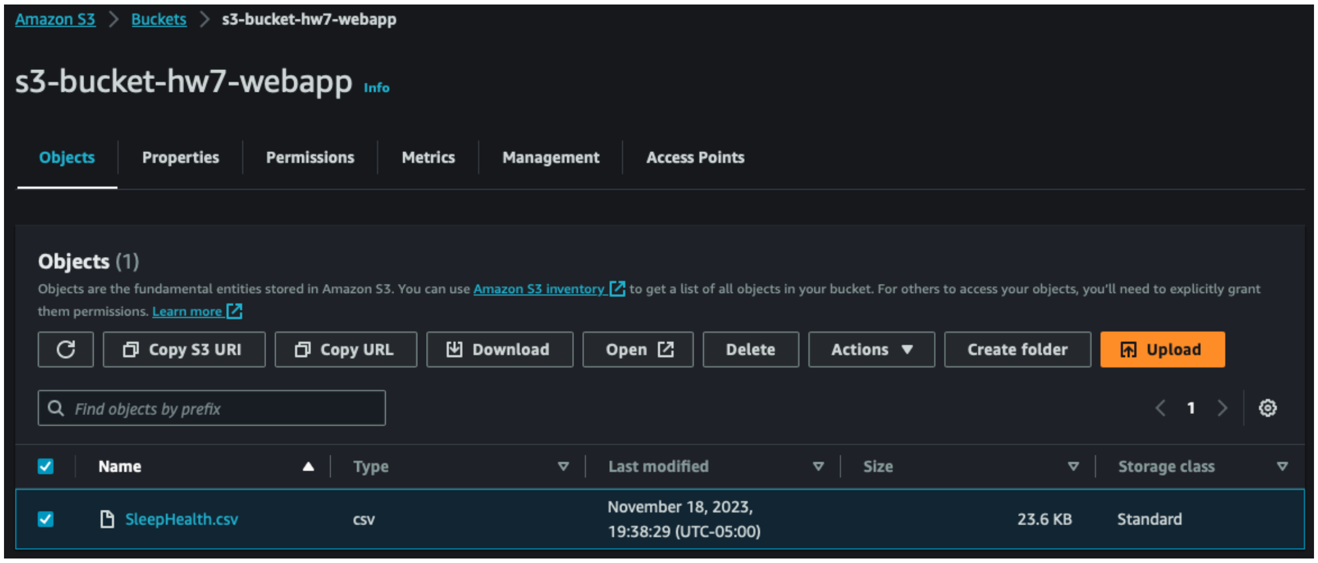
Uploaded SleepHealth.csv to S3 bucket s3-bucket-hw7-webapp.

First 5 rows of DataFrame:

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Blood Pressure	Heart Rate	Daily Steps	Sleep Disorder
0	1	Male	27	Software Engineer	6.1000	6	42	6	Overweight	126/83	77	4200	None
1	2	Male	28	Doctor	6.2000	6	60	8	Normal	125/80	75	10000	None
2	3	Male	28	Doctor	6.2000	6	60	8	Normal	125/80	75	10000	None
3	4	Male	28	Sales Representative	5.9000	4	30	8	Obese	140/90	85	3000	Sleep Apnea
4	5	Male	28	Sales Representative	5.9000	4	30	8	Obese	140/90	85	3000	Sleep Apnea

Data inserted into RDS table SleepHealth.

```
SleepHealth.csv
```



Amazon S3 > Buckets > s3-bucket-hw7-webapp
s3-bucket-hw7-webapp Info
Objects
Properties
Permissions
Metrics
Management
Access Points

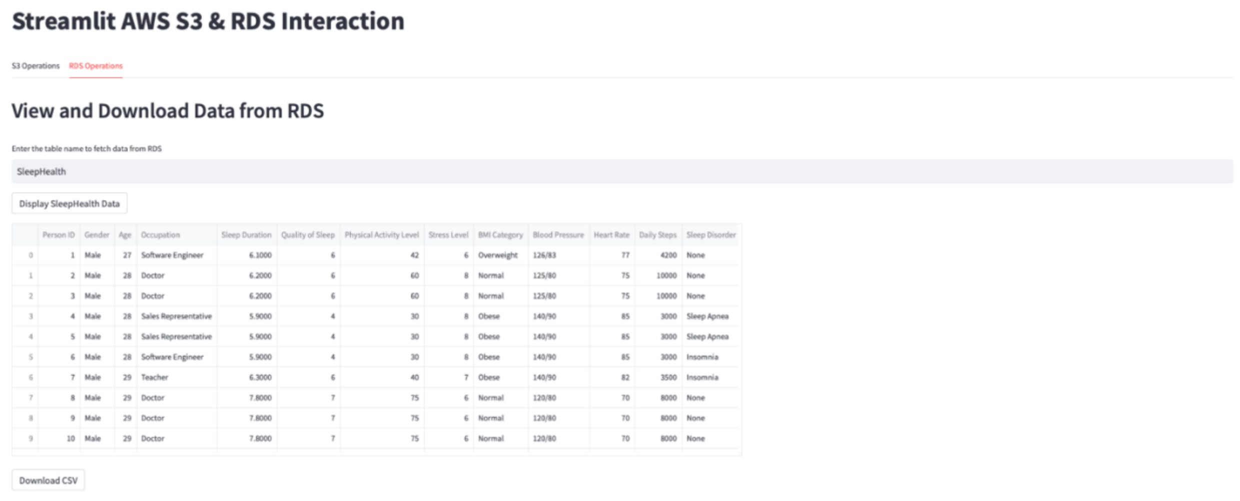
Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

	Name	Type	Last modified	Size	Storage class
✓	SleepHealth.csv	csv	November 18, 2023, 19:38:29 (UTC-05:00)	23.6 KB	Standard

You can see the CSV I uploaded in my application interface being displayed within S3 bucket AWS Console in the image above:

Tab 2: RDS Operations:



Streamlit AWS S3 & RDS Interaction

S3 Operations RDS Operations

View and Download Data from RDS

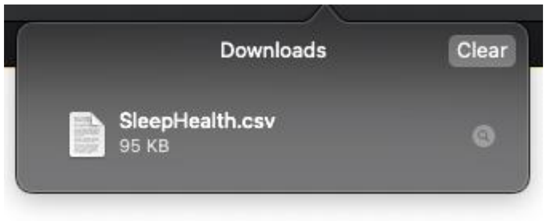
Enter the table name to fetch data from RDS

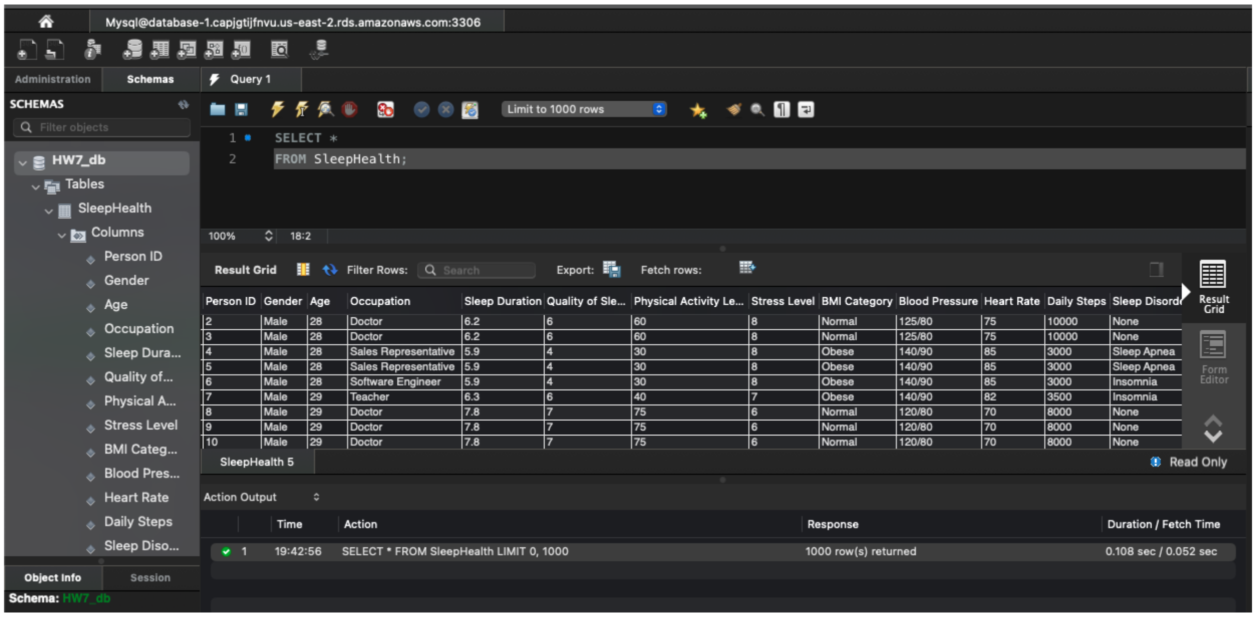
SleepHealth

Display SleepHealth Data

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Blood Pressure	Heart Rate	Daily Steps	Sleep Disorder
0	1	Male	27	Software Engineer	6.1000	6	42	6	Overweight	126/83	77	4200	None
1	2	Male	28	Doctor	6.2000	6	60	8	Normal	125/80	75	10000	None
2	3	Male	28	Doctor	6.2000	6	60	8	Normal	125/80	75	10000	None
3	4	Male	28	Sales Representative	5.9000	4	30	8	Obese	140/90	85	3000	Sleep Apnea
4	5	Male	28	Sales Representative	5.9000	4	30	8	Obese	140/90	85	3000	Sleep Apnea
5	6	Male	28	Software Engineer	5.9000	4	30	8	Obese	140/90	85	3000	Insomnia
6	7	Male	29	Teacher	6.3000	6	40	7	Obese	140/90	82	3500	Insomnia
7	8	Male	29	Doctor	7.8000	7	75	6	Normal	120/80	70	8000	None
8	9	Male	29	Doctor	7.8000	7	75	6	Normal	120/80	70	8000	None
9	10	Male	29	Doctor	7.8000	7	75	6	Normal	120/80	70	8000	None

Download CSV





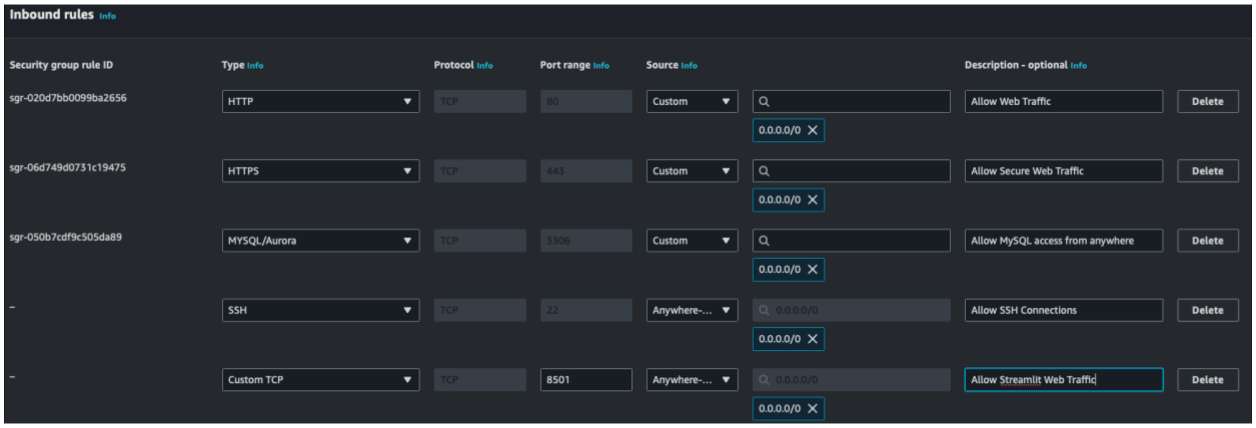
```
SELECT *
FROM SleepHealth;
```

Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sle...	Physical Activity Le...	Stress Level	BMI Category	Blood Pressure	Heart Rate	Daily Steps	Sleep Disord
2	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75	10000	None
3	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75	10000	None
4	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85	3000	Sleep Apnea
5	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85	3000	Sleep Apnea
6	Male	28	Software Engineer	5.9	4	30	8	Obese	140/90	85	3000	Insomnia
7	Male	29	Teacher	6.3	6	40	7	Obese	140/90	82	3500	Insomnia
8	Male	29	Doctor	7.8	7	75	6	Normal	120/80	70	8000	None
9	Male	29	Doctor	7.8	7	75	6	Normal	120/80	70	8000	None
10	Male	29	Doctor	7.8	7	75	6	Normal	120/80	70	8000	None

	Time	Action	Response	Duration / Fetch Time
✓ 1	19:42:56	SELECT * FROM SleepHealth LIMIT 0, 1000	1000 row(s) returned	0.108 sec / 0.052 sec

Deploying AWSWedApp.py using EC2 instance (EC2WebApp7):

After failing a couple of times, I decided to add two new inbound rules into my security group "StreamlitWebAppSG" to allow SSH (port 22) and Streamlit access (Streamlit defaults to port 8501) from any IPv4 address.

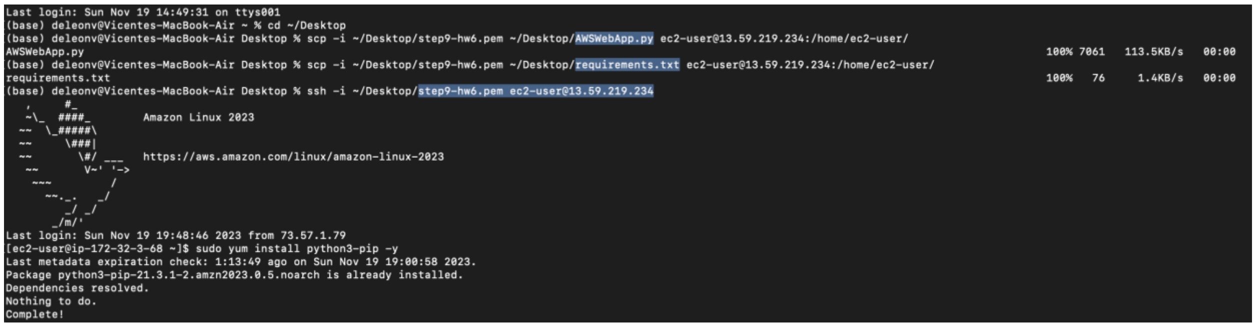


Inbound rules Info

Security group rule ID	Type	Protocol	Port range	Source		Description - optional	
sgr-020d7bb0099ba2656	HTTP	TCP	80	Custom	0.0.0.0/0	Allow Web Traffic	Delete
sgr-06d749d0731c19475	HTTPS	TCP	443	Custom	0.0.0.0/0	Allow Secure Web Traffic	Delete
sgr-050b7cdf9c505da89	MYSQL/Aurora	TCP	3306	Custom	0.0.0.0/0	Allow MySQL access from anywhere	Delete
–	SSH	TCP	22	Anywhere-...	0.0.0.0/0	Allow SSH Connections	Delete
–	Custom TCP	TCP	8501	Anywhere-...	0.0.0.0/0	Allow Streamlit Web Traffic	Delete

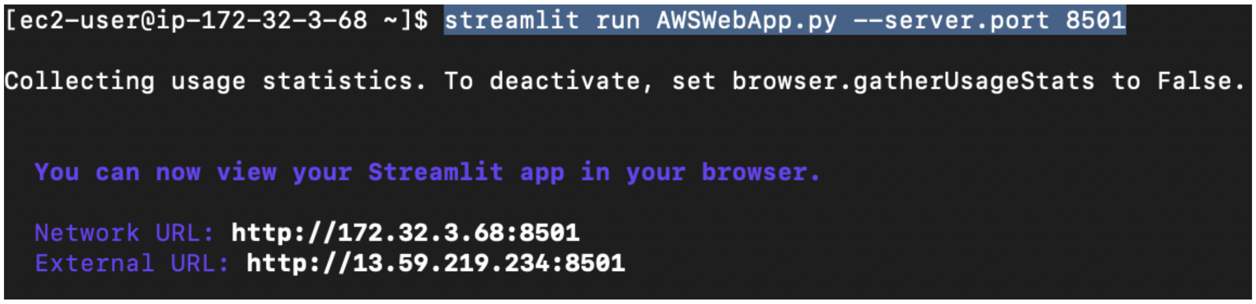
All files were stored within my Desktop for quick testing. So, I transfer files to my EC2 instance and initiated SSH connection using the following commands:

In the below image you could see my EC2-user (default for Amazon Linux) and EC2WebApp7 public IPv4 address (taken from AWS Console):



```
Last login: Sun Nov 19 14:49:31 on ttys001
(base) deleonv@Vicentes-MacBook-Air ~ % cd ~/Desktop
(base) deleonv@Vicentes-MacBook-Air Desktop % scp -i ~/Desktop/step9-hw6.pem ~/Desktop/AWSWebApp.py ec2-user@13.59.219.234:/home/ec2-user/
AWSWebApp.py                                                                                              100% 7061   113.5KB/s   00:00
(base) deleonv@Vicentes-MacBook-Air Desktop % scp -i ~/Desktop/step9-hw6.pem ~/Desktop/requirements.txt ec2-user@13.59.219.234:/home/ec2-user/
requirements.txt                                                                                          100%   76     1.4KB/s   00:00
(base) deleonv@Vicentes-MacBook-Air Desktop % ssh -i ~/Desktop/step9-hw6.pem ec2-user@13.59.219.234
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Sun Nov 19 19:48:46 2023 from 73.57.1.79
[ec2-user@ip-172-32-3-68 ~]$ sudo yum install python3-pip -y
Last metadata expiration check: 1:13:49 ago on Sun Nov 19 19:00:58 2023.
Package python3-pip-21.3.1-2.amzn2023.0.5.noarch is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

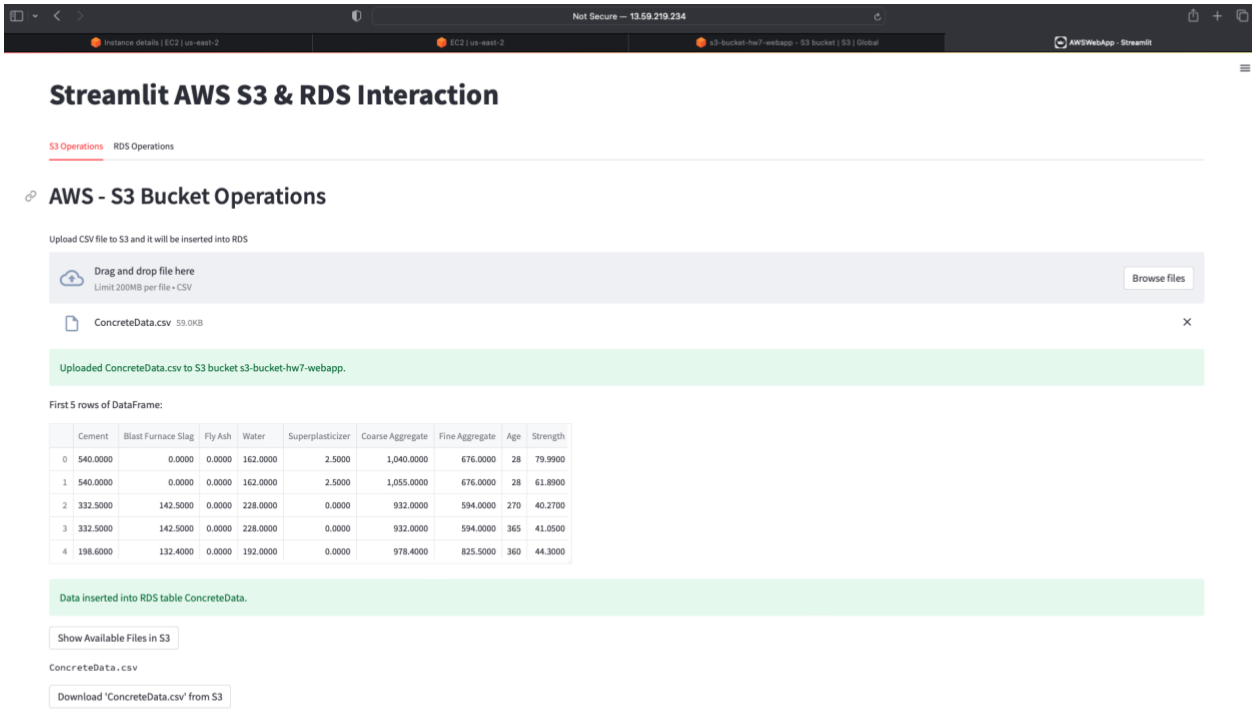
Running Streamlit App (AWSWebApp.py) using Streamlit default 8501 port:



```
[ec2-user@ip-172-32-3-68 ~]$ streamlit run AWSWebApp.py --server.port 8501

Collecting usage statistics. To deactivate, set browser.gatherUsageStats to False.


  You can now view your Streamlit app in your browser.

  Network URL: http://172.32.3.68:8501
  External URL: http://13.59.219.234:8501
```

How the App looks:

Tab 1: The image shows how the app is being hosted via web server, interactions with a new csv file (ConcreteData.csv) and the button below shows the SleepHealth.csv that was stored within S3 bucket from local testing. The user also can retrieve data from S3 bucket if he or she wishes to.



Not Secure — 13.59.219.234
Instance details | EC2 | us-east-2
EC2 | us-east-2
s3-bucket-hw7-webapp - S3 bucket | S3 | Global
AWSWebApp · Streamlit

Streamlit AWS S3 & RDS Interaction

S3 Operations
RDS Operations

AWS - S3 Bucket Operations

Upload CSV file to S3 and it will be inserted into RDS

Drag and drop file here
Limit 200MB per file • CSV
Browse files

ConcreteData.csv 59.0KB

Uploaded ConcreteData.csv to S3 bucket s3-bucket-hw7-webapp.

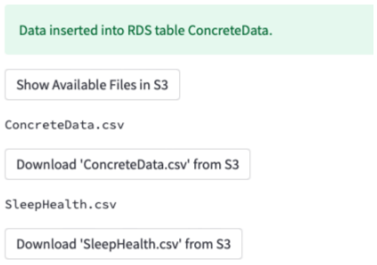
First 5 rows of DataFrame:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age	Strength
0	540.0000	0.0000	0.0000	162.0000	2.5000	1,040.0000	676.0000	28	79.9900
1	540.0000	0.0000	0.0000	162.0000	2.5000	1,055.0000	676.0000	28	61.8900
2	332.5000	142.5000	0.0000	228.0000	0.0000	932.0000	594.0000	270	40.2700
3	332.5000	142.5000	0.0000	228.0000	0.0000	932.0000	594.0000	365	41.0500
4	198.6000	132.4000	0.0000	192.0000	0.0000	978.4000	825.5000	360	44.3000

Data inserted into RDS table ConcreteData.

ConcreteData.csv

Download 'ConcreteData.csv' from S3



Data inserted into RDS table ConcreteData.

Show Available Files in S3

ConcreteData.csv

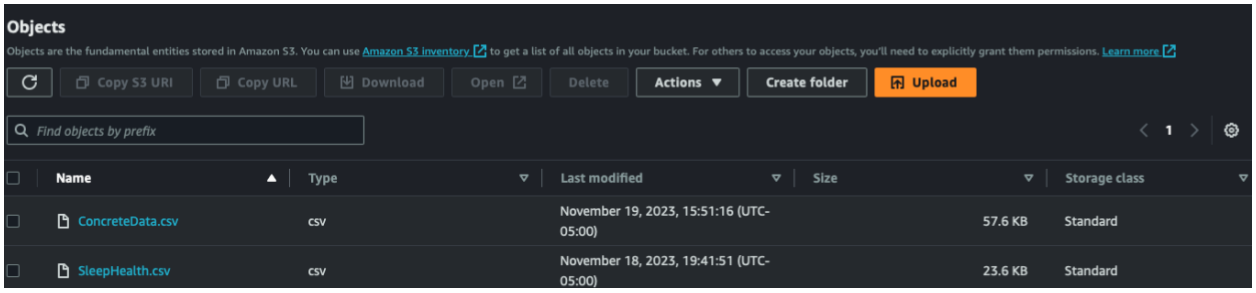
Download 'ConcreteData.csv' from S3

SleepHealth.csv

Download 'SleepHealth.csv' from S3

Two csv files being stored within S3 bucket:

- ConcreteData.csv: New file used for EC2 interaction.
- SleepHealth.csv: old file from local testing.

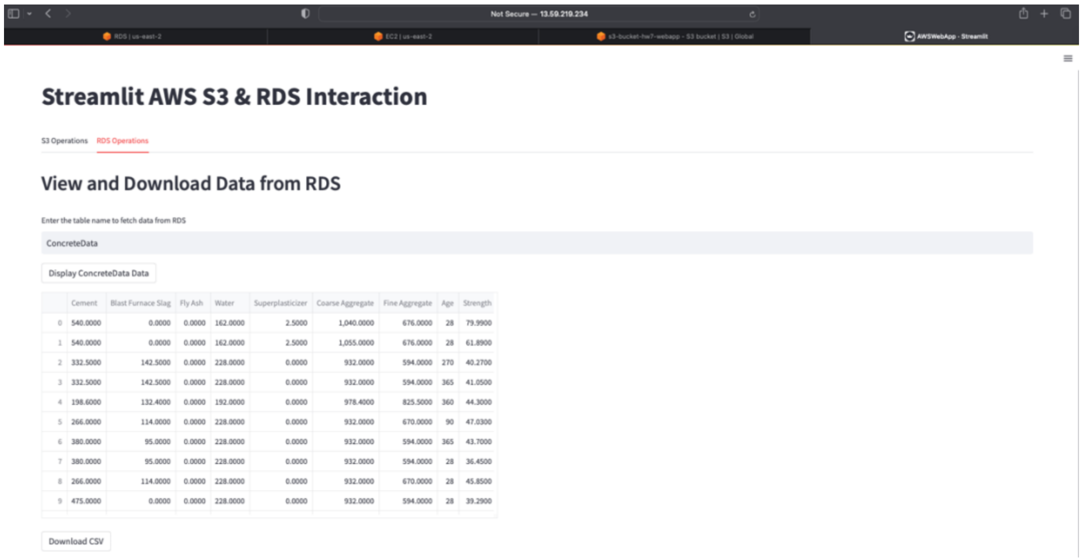


Objects

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
ConcreteData.csv	csv	November 19, 2023, 15:51:16 (UTC-05:00)	57.6 KB	Standard
SleepHealth.csv	csv	November 18, 2023, 19:41:51 (UTC-05:00)	23.6 KB	Standard

Tab 2: The image below shows the procedure tested using data from ConcreteData.csv. As we know, the user uploads a csv file, in this case ConcreteData.csv, into the S3 bucket and the RDS Instance database table is automatically populated using the uploaded csv data. The user doesn't have to worry about tables name due to the "to_sql" line code. The table name will be automatically assigned using the csv file name, which I think its way more interesting thanks to the flexibility being implemented. I wish I could of have more time to experiment with more stuff and to clean my Streamlit code a bit more.


Not Secure — 13.59.219.234
RDS | us-east-2
EC2 | us-east-2
s3-bucket-hw7-webapp - S3 bucket | S3 | Global
AWSWebApp · Streamlit

Streamlit AWS S3 & RDS Interaction

S3 Operations | RDS Operations

View and Download Data from RDS

Enter the table name to fetch data from RDS

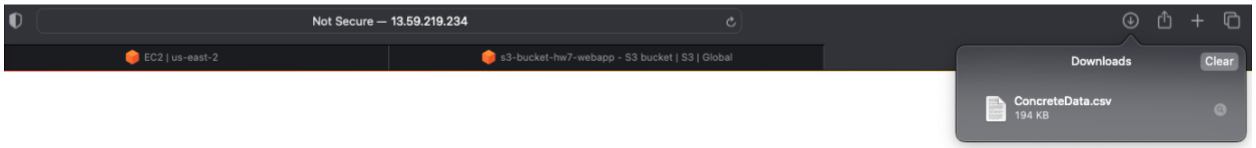
ConcreteData

Display ConcreteData Data

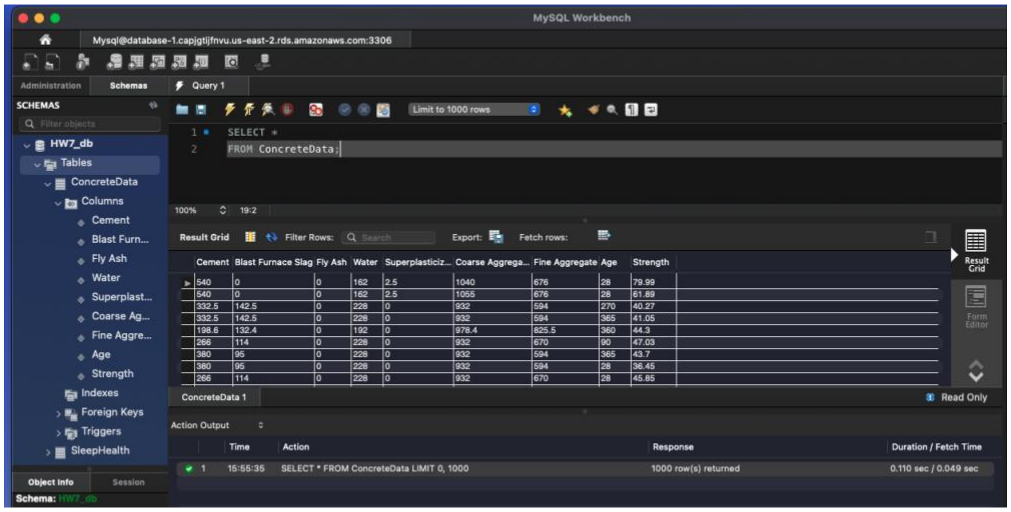
	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age	Strength
0	540.0000	0.0000	0.0000	162.0000	2.5000	1,040.0000	676.0000	28	79.9900
1	540.0000	0.0000	0.0000	162.0000	2.5000	1,055.0000	676.0000	28	61.8900
2	332.5000	142.5000	0.0000	228.0000	0.0000	932.0000	594.0000	270	40.2700
3	332.5000	142.5000	0.0000	228.0000	0.0000	932.0000	594.0000	365	41.0500
4	198.6000	132.4000	0.0000	192.0000	0.0000	978.4000	825.5000	360	44.3000
5	266.0000	114.0000	0.0000	228.0000	0.0000	932.0000	670.0000	90	47.0300
6	380.0000	95.0000	0.0000	228.0000	0.0000	932.0000	594.0000	365	43.7000
7	380.0000	95.0000	0.0000	228.0000	0.0000	932.0000	594.0000	28	36.4500
8	266.0000	114.0000	0.0000	228.0000	0.0000	932.0000	670.0000	28	45.8500
9	475.0000	0.0000	0.0000	228.0000	0.0000	932.0000	594.0000	28	39.2900

Download CSV

Download option from RDS to download database data as a csv file:


Not Secure — 13.59.219.234
EC2 | us-east-2
s3-bucket-hw7-webapp - S3 bucket | S3 | Global
Downloads
Clear
ConcreteData.csv
194 KB

If you create a connection via MYSQL Workbench to see if the data really exists, just repeat the process I mentioned earlier for MySQL Workbench connection. As you see in the image below, both tables from local testing and EC2WebApp7 instance deploying exist:



```
SELECT *
FROM ConcreteData;
```

Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticiz...	Coarse Aggrega...	Fine Aggregate	Age	Strength
540	0	0	162	2.5	1040	676	28	79.99
540	0	0	162	2.5	1055	676	28	61.89
332.5	142.5	0	228	0	932	594	270	40.27
332.5	142.5	0	228	0	932	594	365	41.05
198.6	132.4	0	192	0	978.4	825.5	360	44.3
266	114	0	228	0	932	670	90	47.03
380	95	0	228	0	932	594	365	43.7
380	95	0	228	0	932	594	28	36.45
266	114	0	228	0	932	670	28	45.85

	Time	Action	Response	Duration / Fetch Time
✓ 1	15:55:35	SELECT * FROM ConcreteData LIMIT 0, 1000	1000 row(s) returned	0.110 sec / 0.049 sec

Observations and Problems:

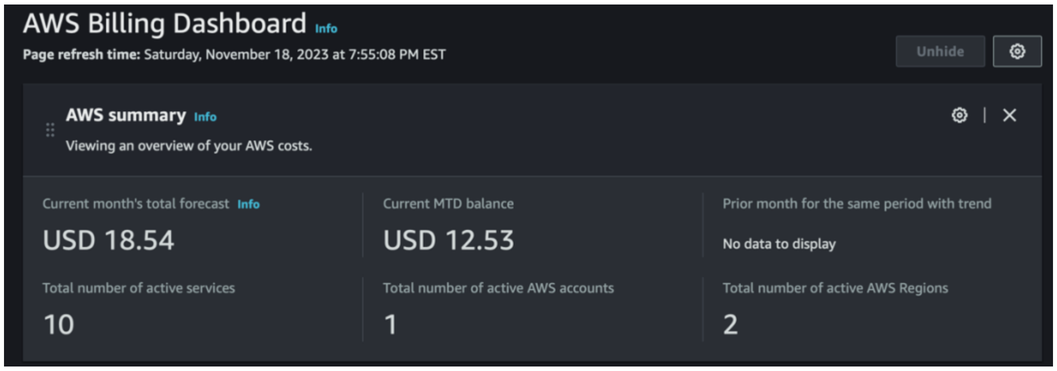
- The problem that took me days to figure out was the MySQL Workbench connection with RDS instance. I initially created a RDS instance, EC2 instance, VPC and subnets all using CloudFormation. This was a huge mistake because everything was completely wrong, and I had to use the AWS Console to set up most of the services except for the VPC. The TA helped a lot during this process. However, I had to create EC2 instance and other configurations services based on the VPC created using CloudFormation. At the end, I just assigned this new VPC to the existing RDS instance the TA helped me create. After this, MySQL Workbench connection was a success.

- I had problems with Streamlit's altair and had to manually modify the reqruirments.txt created via pipreqs to match the "altair" configuration that will allow to use Streamlit.

- I had to create environmental variables to try and provide more security and I was running into issues regarding the EC2 deploying because I forgot we also (if using these types of variables) must update the "zshrc" within the SSH and EC2 connection for AWSWebApp.py could access my AWS credentials. This is something than can be easily forget, due to the number of details AWS services require.

- The Auto Scaling group setting were change from max 5 to max 1 because of the number of instances were being launched. This gave me a problem regarding my free tier option, reason why I change it. since I do not master AWS yet, I am worried about the billing. I will just delete the Auto Scaling Group to avoid scaling and launching new instances and I will just stop the Auto Scaling process and the EC2WebApp7 EC2 instance (just like I did for HW6 to avoid incurring costs).

- Understanding AWS Services was quite challenging and confusing for me. I came across multiple failing attempts and errors that cost me a lot of time for this homework. However, I do feel I understood this homework's mission. I personally think that the only way to address AWS material and challenges, I just by trying and failing just like I did. I had to do this assignment 3 times to fully understand what was going on**. Many of the benefits were explained during the creation of this report specially on the things I want to implement regarding scalability, high traffic, storage, management, and architecture. A mix of CLI commands helped me figure out things that I couldn't see in my AWS console. Also, I used many codes and knowledge form other classes to create this project.**


AWS Billing Dashboard Info

Page refresh time: Saturday, November 18, 2023 at 7:55:08 PM EST

AWS summary Info

Viewing an overview of your AWS costs.

Current month's total forecast Info	Current MTD balance	Prior month for the same period with trend
USD 18.54	USD 12.53	No data to display
Total number of active services	Total number of active AWS accounts	Total number of active AWS Regions
10	1	2

AWS References:

EC2: https://aws.amazon.com/ec2/ec2-get-started/#:~:text=Amazon%20Elastic%20Compute%20Cloud%20(Amazon,configure%20capacity%20with%20minimal%20friction.

S3: https://aws.amazon.com/s3/getting-started/#:~:text=Amazon%20Simple%20Storage%20Service%20(Amazon,at%20any%20time%2C%20from%20anywhere.

VPC: https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html

CF: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html

Parameters: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html

Creating AMI role: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create_for-service.html

Getting caller identity: https://docs.aws.amazon.com/cli/latest/reference/sts/get-caller-identity.html

Creating bucket: https://docs.aws.amazon.com/cli/latest/reference/s3api/create-bucket.html

S3 best practices: https://docs.aws.amazon.com/AmazonS3/latest/userguide/security-best-practices.html

Bucket Policy: https://docs.aws.amazon.com/AmazonS3/latest/userguide/S3OutpostsBucketPolicyEdit.html

Bucket Policies: https://docs.aws.amazon.com/AmazonS3/latest/userguide/example-bucket-policies.html

app running on EC2 instances - AMI: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2.html

GetObject: https://docs.aws.amazon.com/AmazonS3/latest/API/API_GetObject.html

PutObject: https://docs.aws.amazon.com/AmazonS3/latest/API/API_PutObject.html

DeleteObject: https://docs.aws.amazon.com/AmazonS3/latest/API/API_DeleteObject.html

Template basics: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/gettingstarted.templatebasics.html

CloudFormation: https://aws.amazon.com/blogs/mt/automate-account-creation-and-resource-provisioning-using-aws-service-catalog-aws-organizations-and-aws-lambda/#:~:text=AWS%20CloudFormation%20allows%20you%20to,the%20compute%20time%20you%20consume.

Template version (only valid value): https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/format-version-structure.html

Stack Creation: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-cli-creating-stack.html

Creating web application using VPC, EC2, and RDS: https://medium.com/@awseng12/building-a-three-tier-web-application-in-aws-with-vpc-alb-ec2-and-rds-23af0c0e610d#:~:text=Setting%20up%20the%20Networking%20Infrastructure,subnets%20for%20the%20database%20layer.

High-Availability Web Application on AWS using VPC, Auto Scaling, ALB: https://medium.com/@stephanietabares/creating-a-high-availability-web-application-infrastructure-on-aws-with-vpc-auto-scaling-and-62a4f0f0b17b

Web App AutoScaling: https://medium.com/@Deshone_Henry/how-to-create-a-highly-available-website-using-auto-scaling-with-aws-ec07a83db69e

CloudFormation LoadBalancer: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-elasticloadbalancingv2-loadbalancer.html

Target Groups explained: https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-target-groups.html

CloudFormation TargetGroup: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-elasticloadbalancingv2-targetgroup.html

Listener explained: https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-listeners.html

CloudFormation Listener: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-elasticloadbalancingv2-listener.html

Load Balancer Algorithm: https://aws.amazon.com/what-is/load-balancing/#:~:text=Load%20balancers%20improve%20application%20performance,closer%20server%20to%20reduce%20latency

Load Balancer: https://docs.aws.amazon.com/AmazonECS/latest/APIReference/API_LoadBalancer.html

Auto Scaling explained: https://docs.aws.amazon.com/autoscaling/ec2/userguide/auto-scaling-groups.html

AWS AutoScaling: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-autoscaling-autoscalinggroup.html

Auto Scaling Web App: https://medium.com/@Deshone_Henry/how-to-create-a-highly-available-website-using-auto-scaling-with-aws-ec07a83db69e

Min vs Max configuration:
https://docs.aws.amazon.com/apprunner/latest/api/API_CreateAutoScalingConfiguration.html

AWS LaunchConfiguration: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-autoscaling-launchconfiguration.html

CloudFormation EC2: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-instance.html

CloudFormation RDS: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-rds-dbinstance.html

RDS: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-rds-dbinstance.html

DB instances:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.DBInstanceClass.html

AWS RDS DB instance storage:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Storage.html

CloudFormation S3 Bucket: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-s3-bucket.html

AWS::EC2:VPC: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-vpc.html

Subnets: https://docs.aws.amazon.com/vpc/latest/userguide/configure-subnets.html#

Subnets: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-rds-dbsubnetgroup.html

DB Instance and Subnets:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_VPC.WorkingWithRDSInstanceinaVPC.html#

CloudFormation Subnets: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-subnet.html

Stack: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-cli-creating-stack.html

Other Stack Parameters:
https://docs.aws.amazon.com/AWSCloudFormation/latest/APIReference/API_CreateStack.html

To view stack progress: https://docs.aws.amazon.com/cli/latest/reference/cloudformation/describe-stacks.html

Creating stack: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-cli-creating-stack.html

Parameter: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html

Other Parameters for custom names (which I have): https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html

To describe stack and see progress: https://docs.aws.amazon.com/cli/latest/reference/cloudformation/describe-stacks.html

Delete stack: https://docs.aws.amazon.com/cli/latest/reference/cloudformation/delete-stack.html

Alphanumeric character problems: https://stackoverflow.com/questions/53626203/cloudformation-parameter-template-error-parameter-is-non-alphanumeric

Internet Gateway: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-internetgateway.html

VPC Gateway attachment: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-vpcgatewayattachment.html

DependsOn attribute: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-attribute-dependson.html

MySQL Workbench RDS instance connection: https://aws.amazon.com/getting-started/hands-on/create-mysql-db/

Environmental Variables: https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/guide_credentials_environment.html

Environmental Variables: https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-envvars.html

Route Tables explained: https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Route_Tables.html

Route Tables: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-routetable.html

Route Table Associations: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-subnetroutetableassociation.html

Route Table: https://docs.aws.amazon.com/vpc/latest/userguide/WorkWithRouteTables.html

Routes: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-route.html

Security Groups CF: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-securitygroup.html

Sg Ingress: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-securitygroupingress.html

Sg Egress: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-securitygroupegress.html

Modify DB instance: https://docs.aws.amazon.com/cli/latest/reference/docdb/modify-db-instance.html

SSH: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/authorizing-access-to-an-instance.html

Transferring files to EC2: https://docs.aws.amazon.com/managedservices/latest/appguide/qs-file-transfer.html#

Transferring files to EC2: https://www.bornfight.com/blog/transferring-files-between-local-machine-and-aws-instance/

EC2 user (default): https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/connect-to-linux-instance.html

Connect to SSH: https://docs.aws.amazon.com/lightsail/latest/userguide/amazon-lightsail-ssh-using-terminal.html

Installing python within Amazon Linux: https://docs.vmware.com/en/VMware-vSphere-Bitfusion/4.0/Example-Guide/GUID-D3A136D1-0EA1-4CD0-AFCF-7FD88299A53A.html

Streamlit Web App References:

Documentation: https://docs.streamlit.io/library/get-started

Text elements: https://docs.streamlit.io/library/api-reference/text

Boto3 Documentation: https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3.html

Boto3: https://medium.com/featurepreneur/boto3-aws-sdk-for-python-e7391b9901c5

Uploading files to S3: https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3/client/upload_fileobj.html

Engine documentation: https://docs.sqlalchemy.org/en/20/core/engines.html

To_sql: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_sql.html

To_sql: https://www.w3resource.com/pandas/dataframe/dataframe-to_sql.php

Progress and status: https://docs.streamlit.io/library/api-reference/status

Stream: https://docs.python.org/3/library/io.html

Stream: https://www.digitalocean.com/community/tutorials/python-io-bytesio-stringio

Download files: https://towardsdatascience.com/how-to-upload-and-download-files-from-aws-s3-using-python-2022-4c9b787b15f2

Boto3 list_objects_v2: https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3/client/list_objects_v2.html

Getting object: https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3/client/get_object.html

Getting object: https://stackoverflow.com/questions/31976273/open-s3-object-as-a-string-with-boto3

St.download: https://docs.streamlit.io/library/api-reference/widgets/st.download_button

Dataframe to csv: https://www.freecodecamp.org/news/dataframe-to-csv-how-to-save-pandas-dataframes-by-exporting/

Os.getenv: https://www.geeksforgeeks.org/python-os-getenv-method/

St.file_uploader: https://docs.streamlit.io/library/api-reference/widgets/st.file_uploader

St.file_uploader: https://andymcdonaldgeo.medium.com/uploading-and-reading-files-with-streamlit-92885ac3a1b6

Stinput and button: https://docs.streamlit.io/library/api-reference/widgets/st.button

Streamlit port issues; https://discuss.streamlit.io/t/running-streamlit-on-aws-ec2-handling-port-issues-port-80-port-8501-nginx/29902

Altair error: https://discuss.streamlit.io/t/modulenotfounderror-no-module-named-altair-vegalite-v4/42921/2

Streamlit configuration: https://docs.streamlit.io/library/advanced-features/configuration