# DEVOPS CAPSTONE PROJECT

# DEPLOYING A MOVIE LISTING WEBSITE INTO AWS CLOUD INFRASTRUCTURE WITH PROPER SCALING


**Group-5**
**Team members:**

| | |
|---|---|
| 1) P. D S SIVA | 21MH5A0351 |
| 2) B. SIVA DURGA | 21P35A0371 |
| 3) K. VIVEK | 20A91A04L8 |
| 4) S. CHITTI | 20A91A0449 |
| 5) K. SUVARNA | 20P31A0484 |
| 6) A. BANGARRAJU | 20A91A0466 |
| 7) D. DEEPIKA | 20A91A0476 |
| 8) N. ANKITHA | 20A91A0496 |

## AIM OF THE PROJECT

The main Aim of the project is to deploy a website on an Amazon EC2 instance using Docker container technology.EC2 is a web service provided by AWS to easily deploy and manage virtual servers. It is highly scalable and available to use. Docker is a containerization tool which makes an application run on any platform. Mongo dB to replace the local database so that thedata is stored in our cluster and images are stored in S3.

We can host a web application on EC2 which is highlyreliable, can handle vast web traffic and runs smoothly. This application can be easily maintained andupdated because of docker with a load balancer attached to it.

## SCOPE OF THE PROJECT

The scope of the project is to deploy a website using Amazon Elastic Compute Cloud (EC2) and Docker container technology. The project includes developing a web application and modify it according to our requirements. Configuring and setting up an EC2 instance also plays a vital role in this project. Mongo dB is used instead of the local database.

The web application is containerized using docker for easier maintenance and deployment.
It involves the deployment of web application from our EC2 instance and also attaching a load balancer to reduce the web traffic load on an instance.

## TECHNOLOGIES USED


**AWS:** AWS is a cloud computing platform that provides a range of services for individuals and businesses to store, manage, and process data and applications in the cloud. With over 200 services, including computing, storage, databases, analytics, machine learning, IoT, and security, AWS is flexible, cost-effective, and reliable. Some popular AWS services include EC2 for computing, S3 for storage, RDS for databases, Lambda for serverless computing, DynamoDB for NoSQL databases, and ECS for container management.

**Docker:** Docker contains containers which are created when the image is starts running. Containers are self-contained, portable units of software that include all the components and dependencies an application needs to run. Docker simplifies application development and deployment by enabling developers to package their applications in containers and run them on any system with Docker installed. This provides greater flexibility, portability, and scalability for applications.

**GIT:** Git is a version control system that allows developers to track changes to their code and collaborate with others on a project. It is a free and open-source tool that provides features such as branching, merging, and version history. Git is a powerful tool for managing software development projects.

**Web Development:** Web development is the process of creating and maintaining websites and web applications. It involves a combination of programming, design, and content creation to create web pages that are functional and visually appealing. Web development can include both frontend development, which focuses on the user interface, and backend development, which focuses on the server-side programming that powers the website.

## TECHNOLOGIES STACK:

**Frontend**: ReactJS

**Backend**: NodeJS

**Database**: MongoDB

# IMPLEMENTATAION and OBSERVATIONS

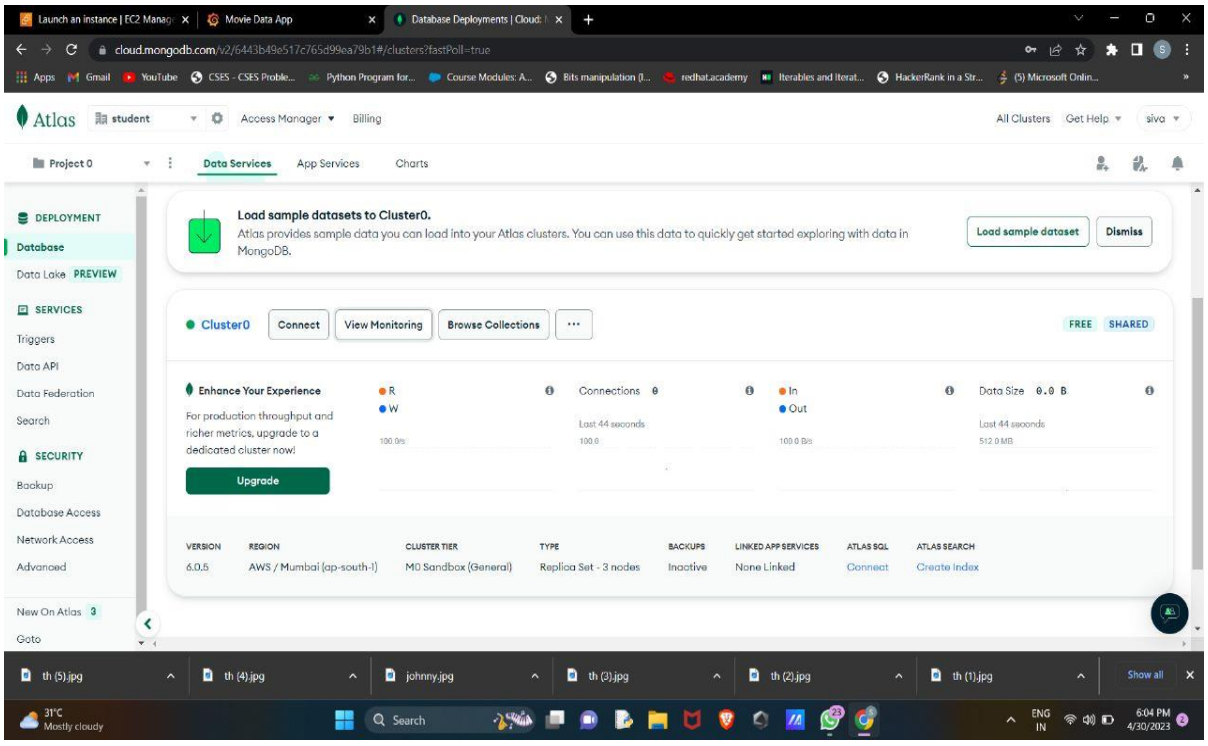
## Step 1: Connecting the Client with Atlas MongoDB

Initially, cloned the “Movie listing” website which uses ReactJS as frontend, NodeJS as backend andMongoDB as database from the repo:

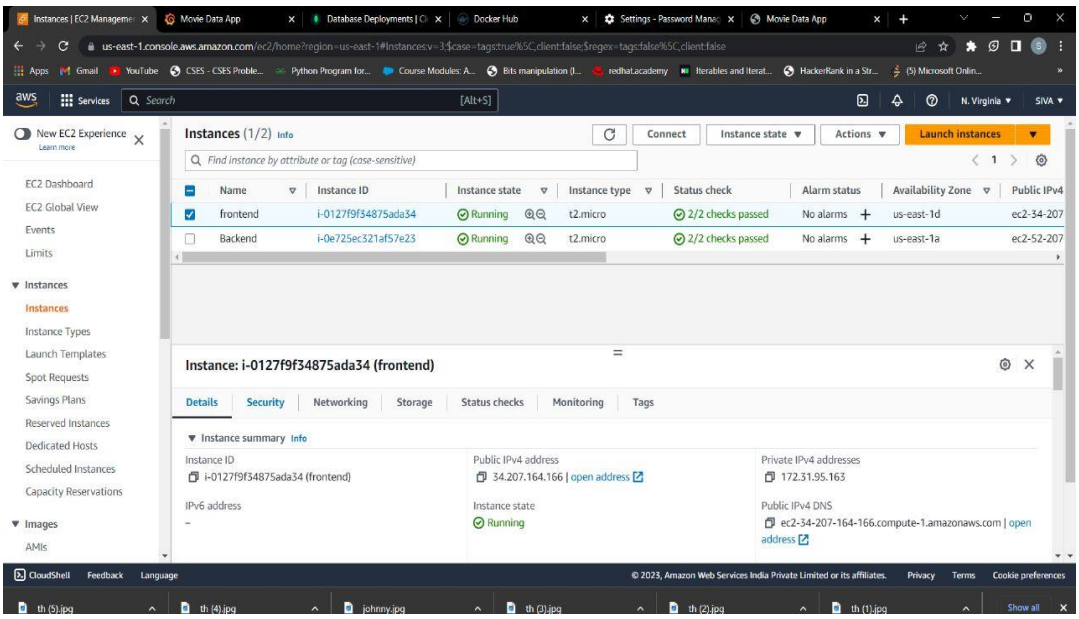
https://github.com/snehal-heíoviíed/DEVOPS_CAPS1ᵔONE

Here, we create MongoDB account and create a free tire cluster and given network access as allowaccess from anywhere and create the cluster.

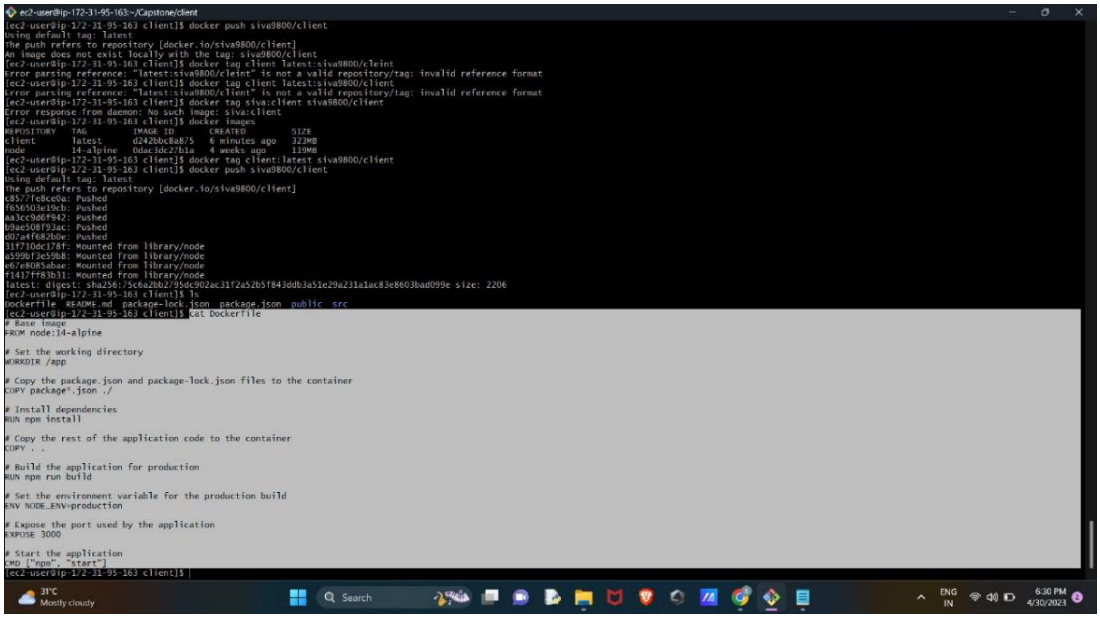
- After creation of the cluster then connect it. Here we able to see the connection link.

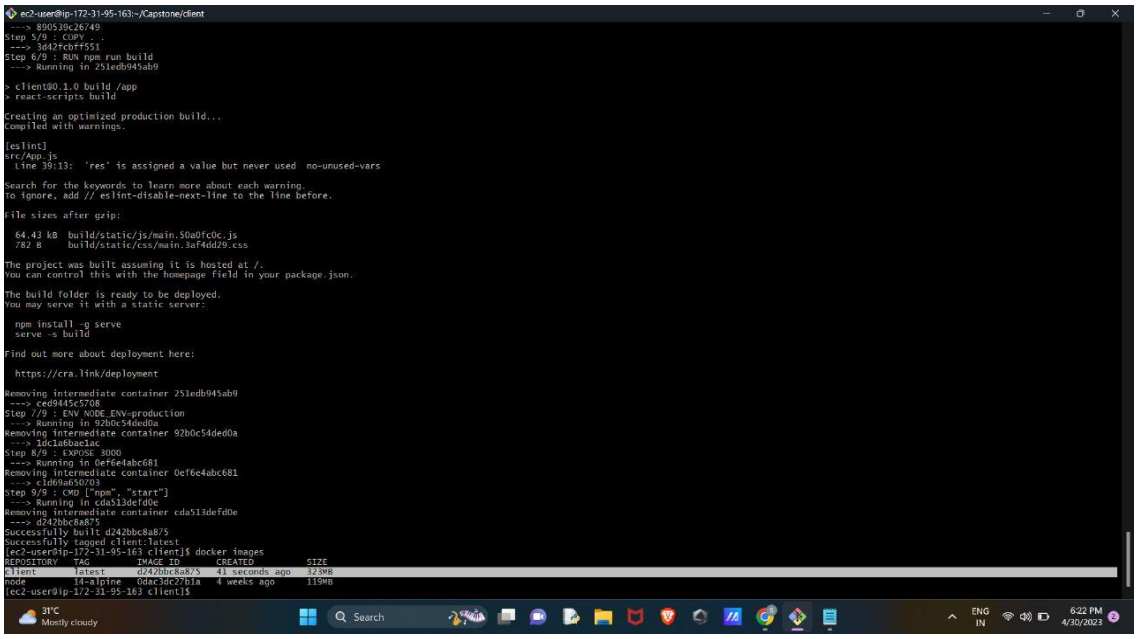


Creation of EC2 instance.

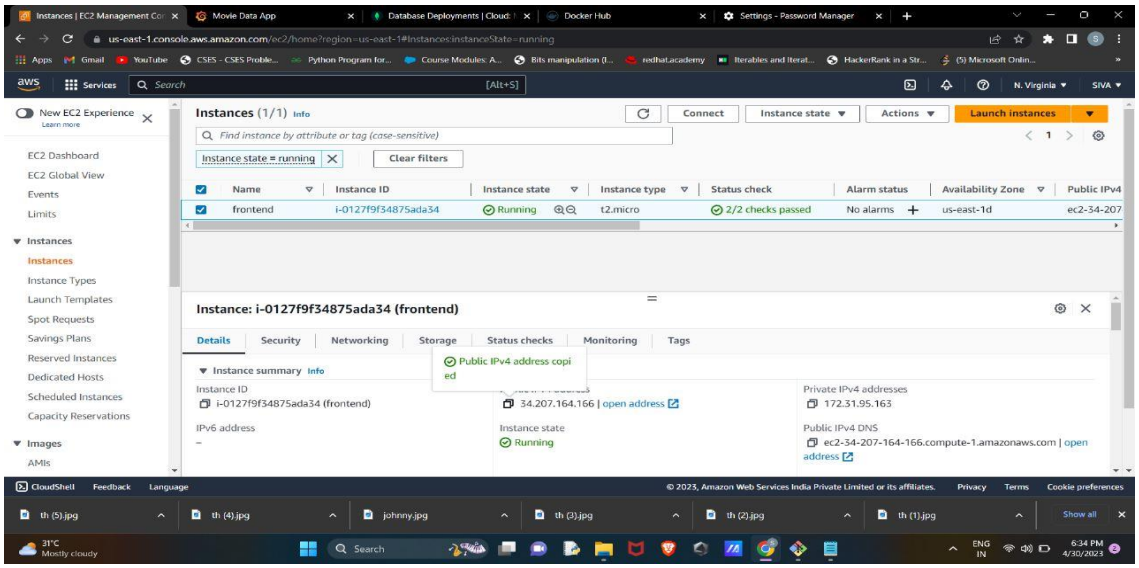



Creating dockerfile and image in frontend(client)

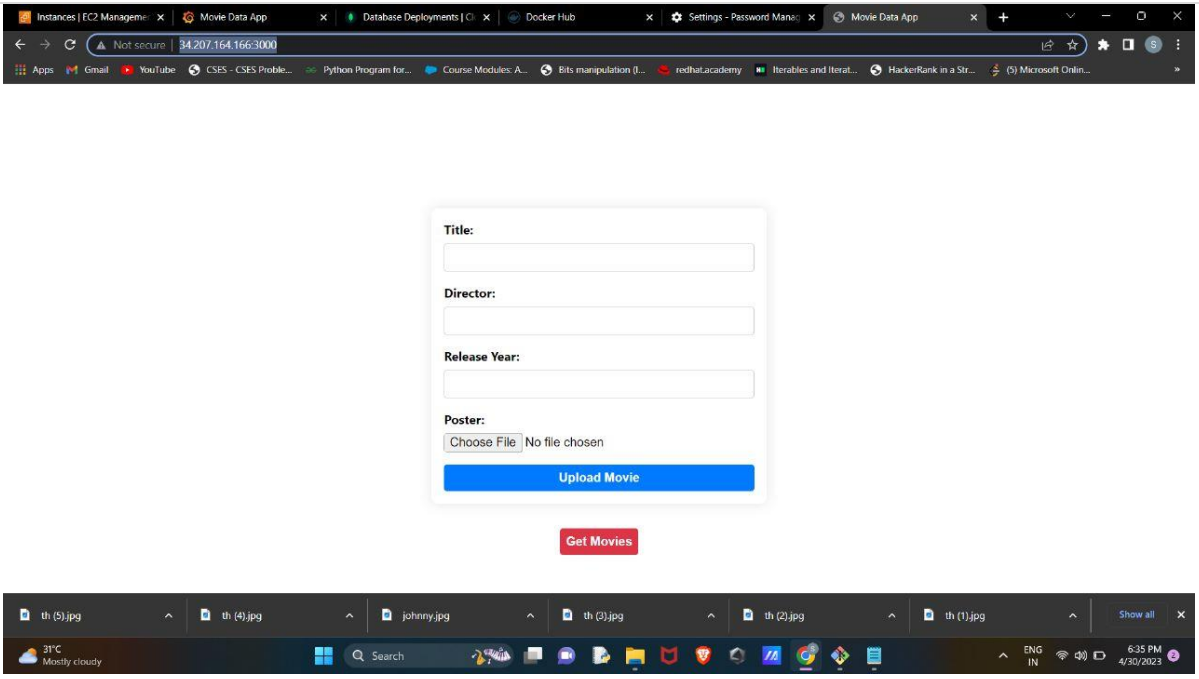




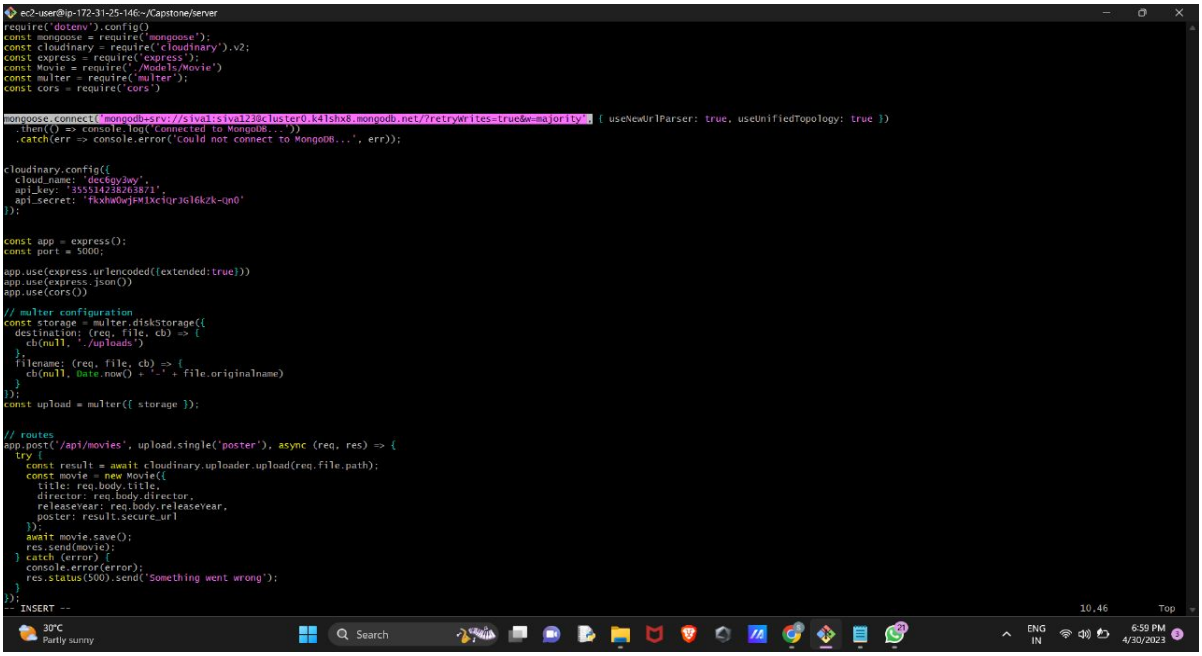

Front end public Ip




Front end running successfully with 34.207.164.166:3000 PUBLIC IP.

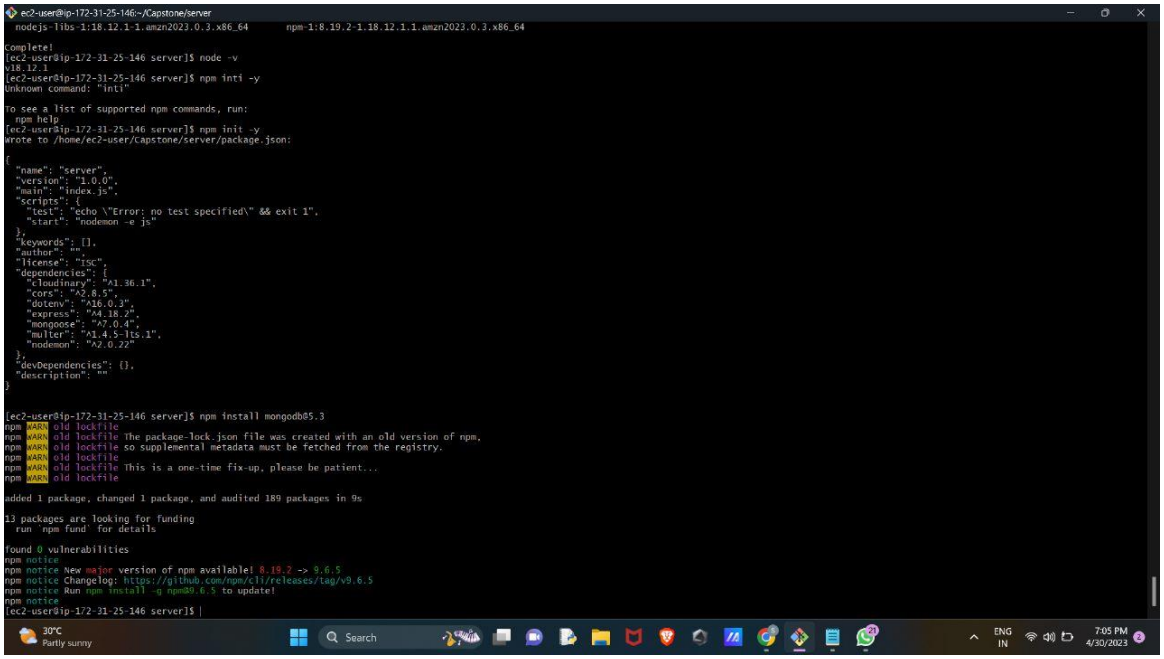



MongoDB connection and installing mongoDB in Frontend.

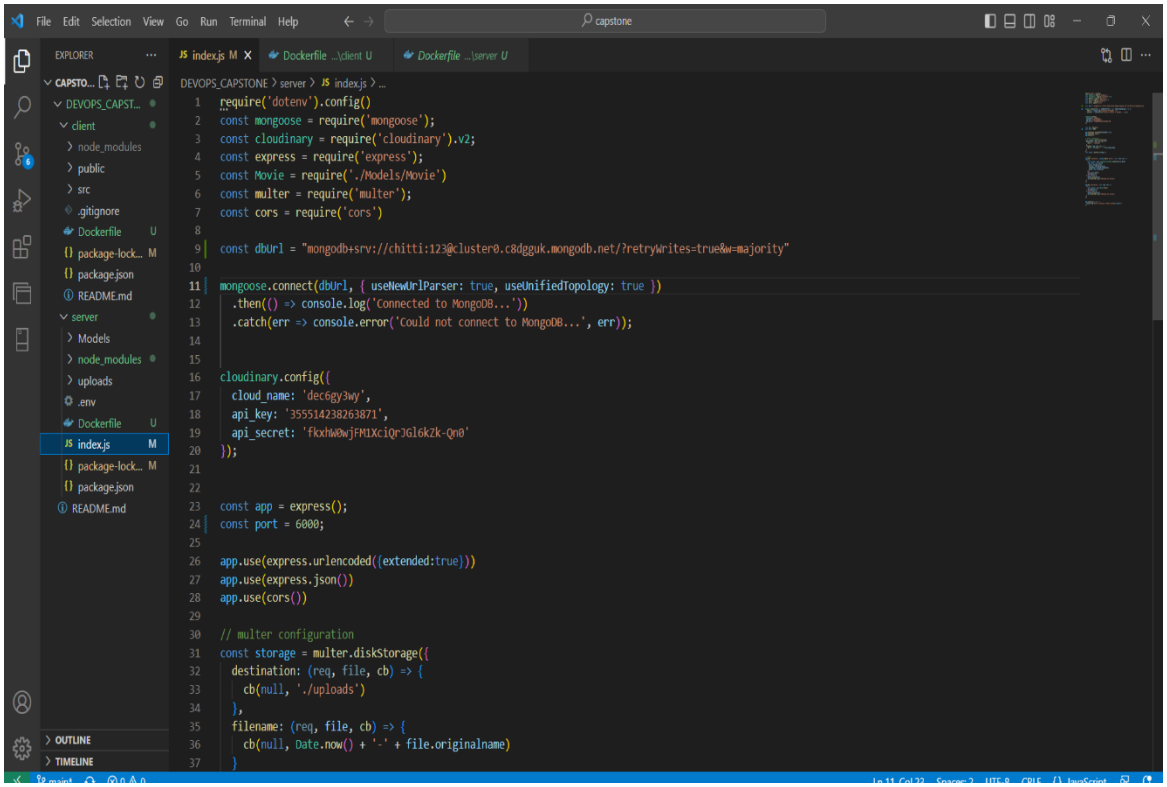





2

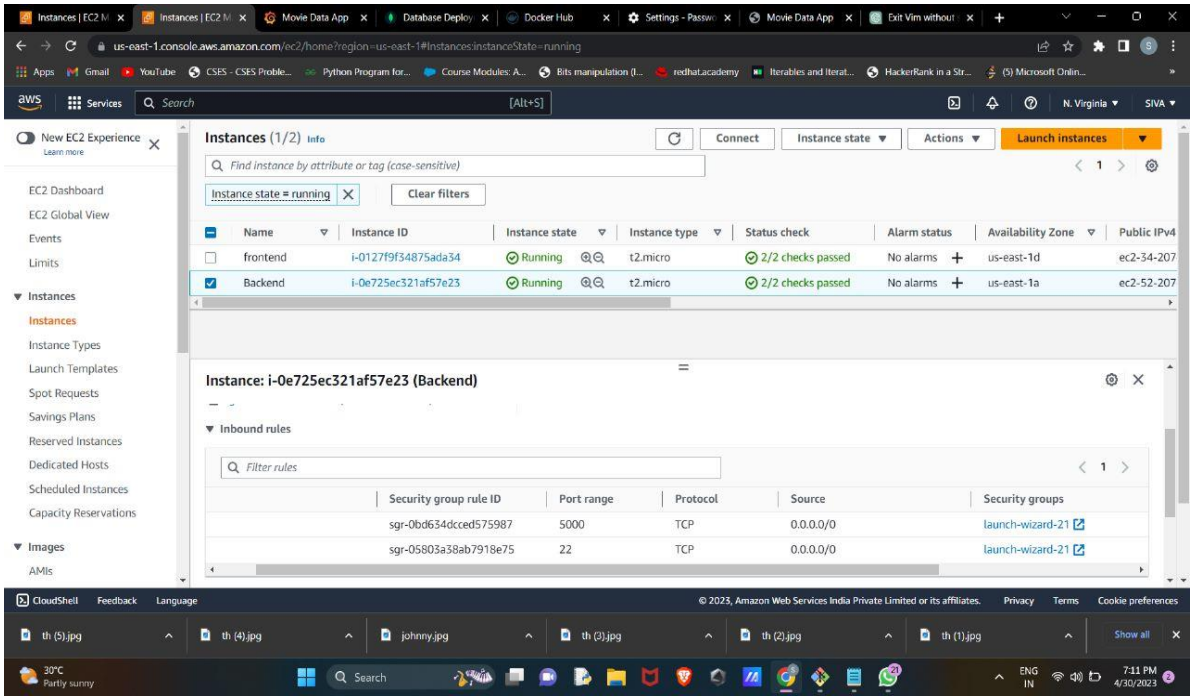


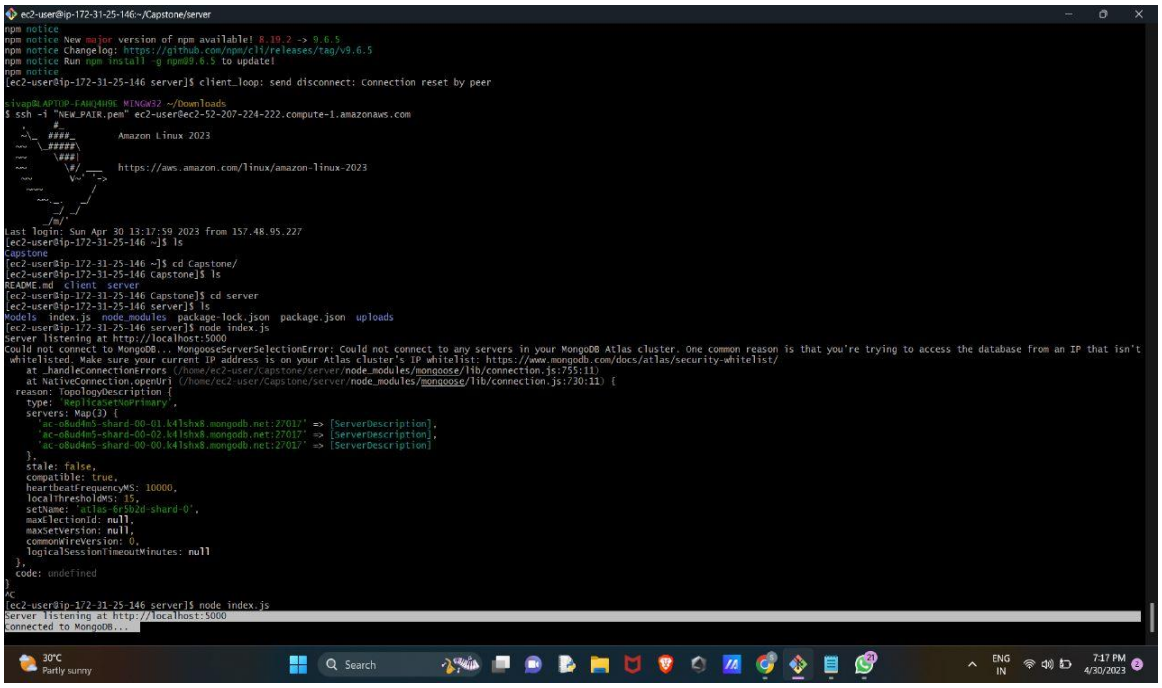

## Step 2: Connecting the Server with Atlas MongoDB.



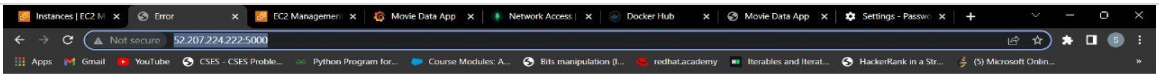


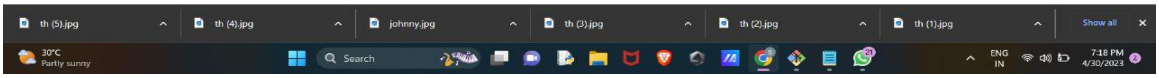

2



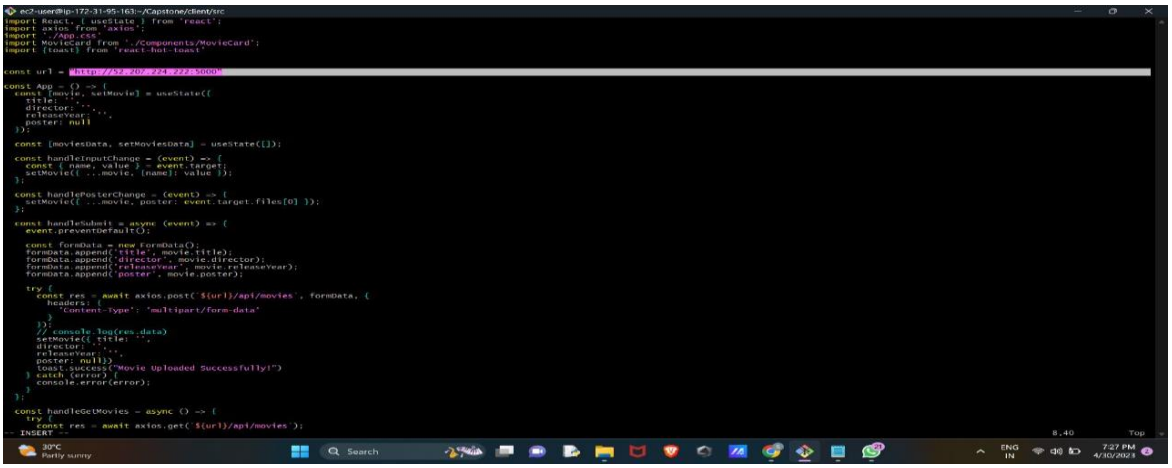




Backend accessible IP 52.207.224.222:5000

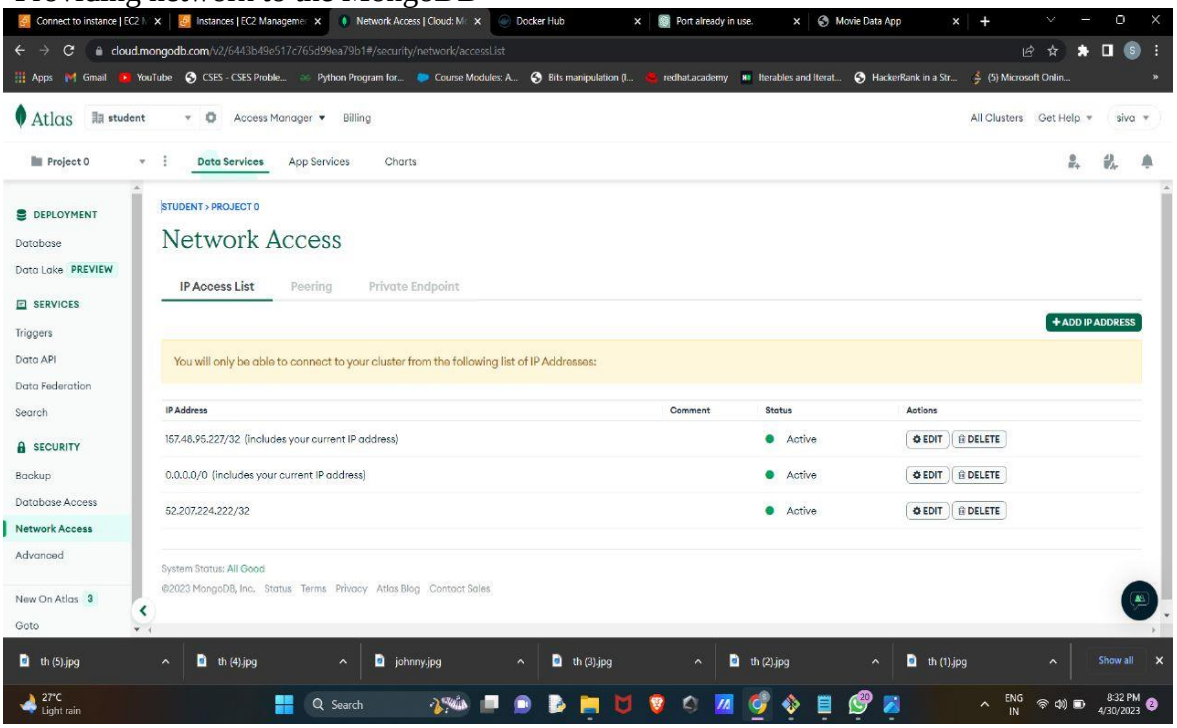


Added Backend URL to Frontend

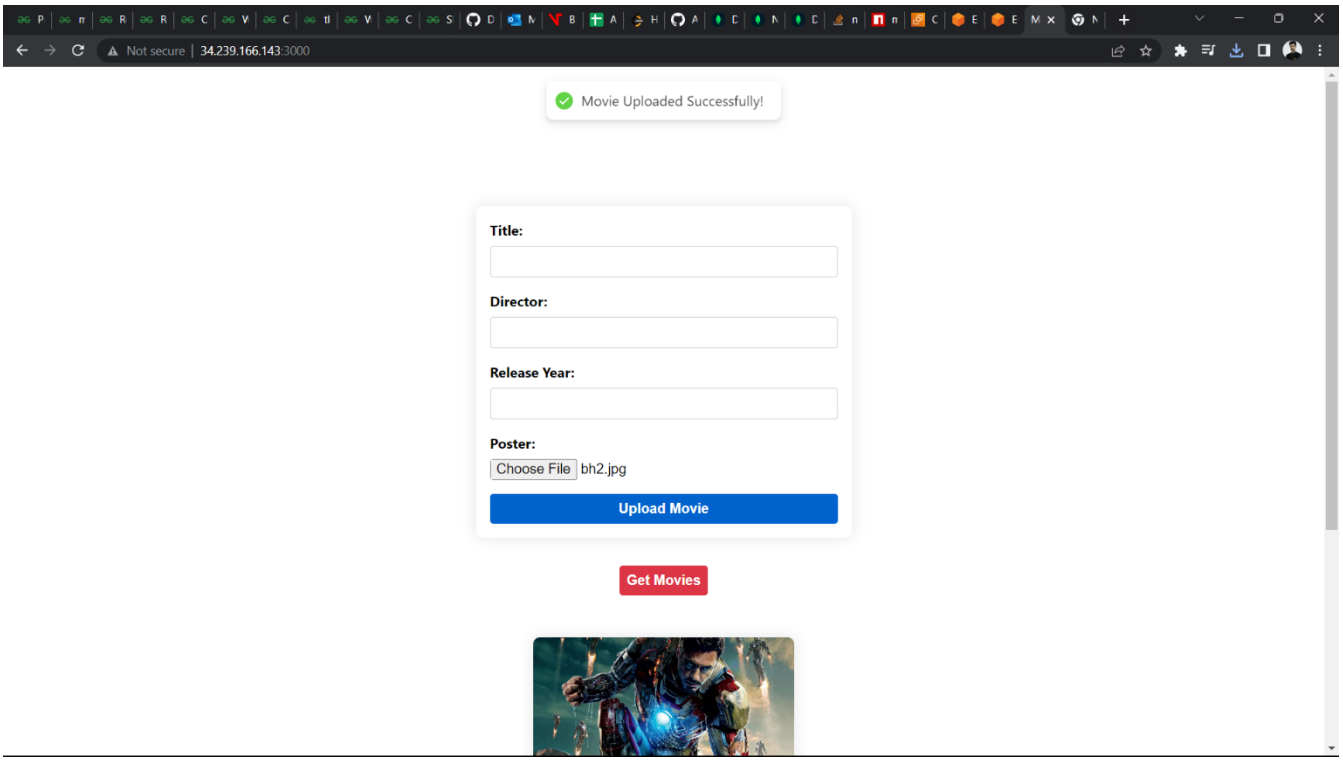


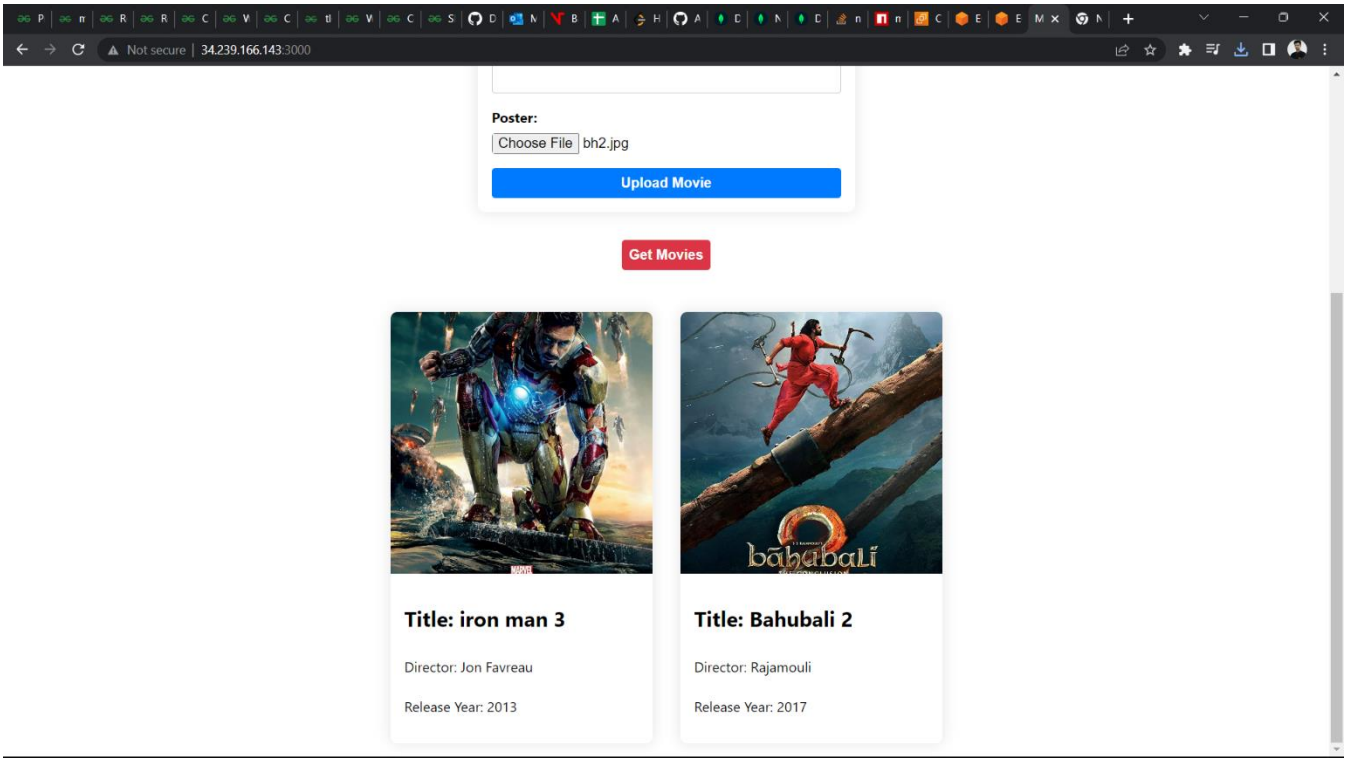

Providing network to the MongoDB






2

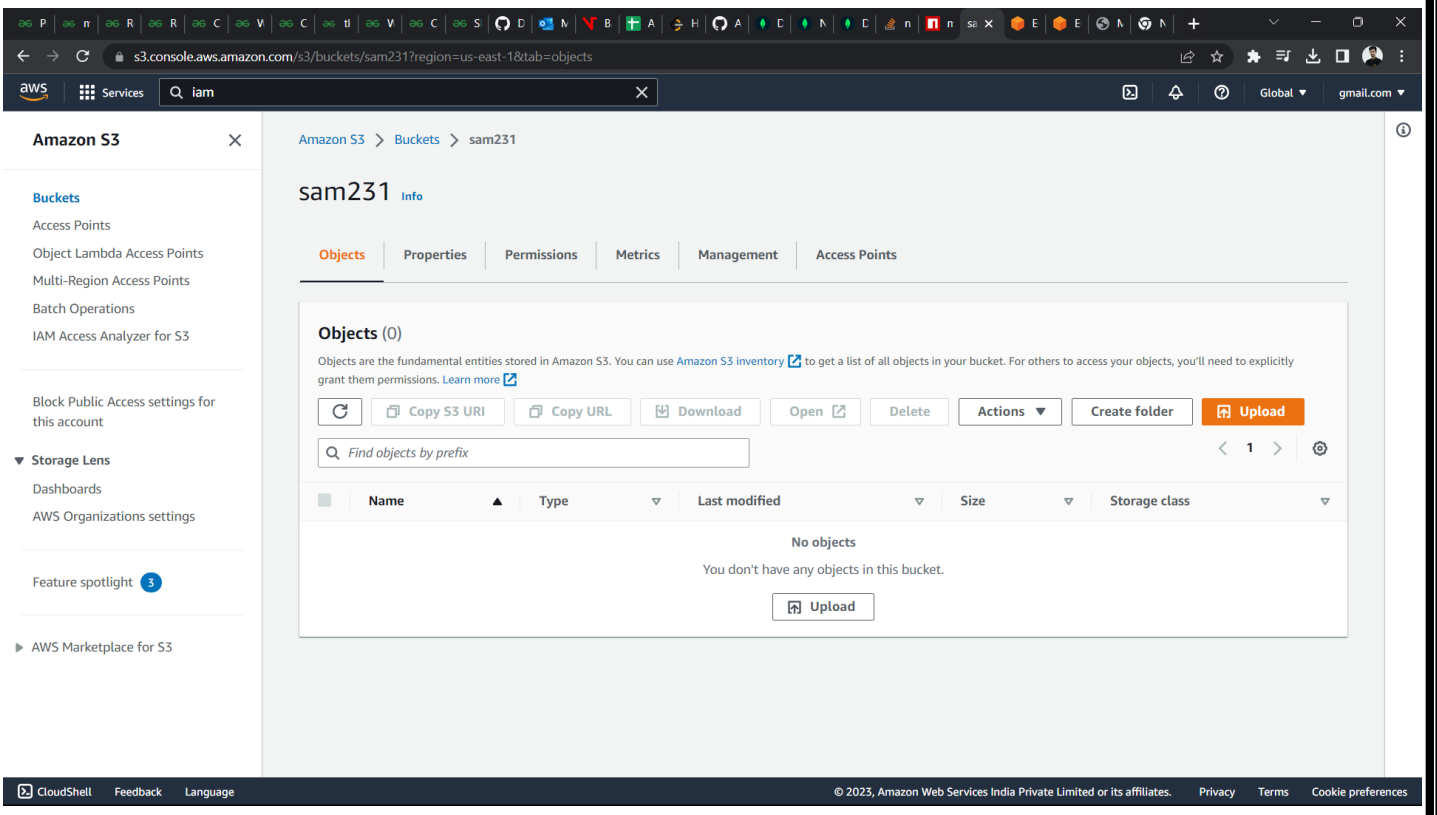

Not secure | 34.239.166.143:3000
Movie Uploaded Successfully!
Title:
Director:
Release Year:
Poster:
Choose File bh2.jpg
Upload Movie
Get Movies

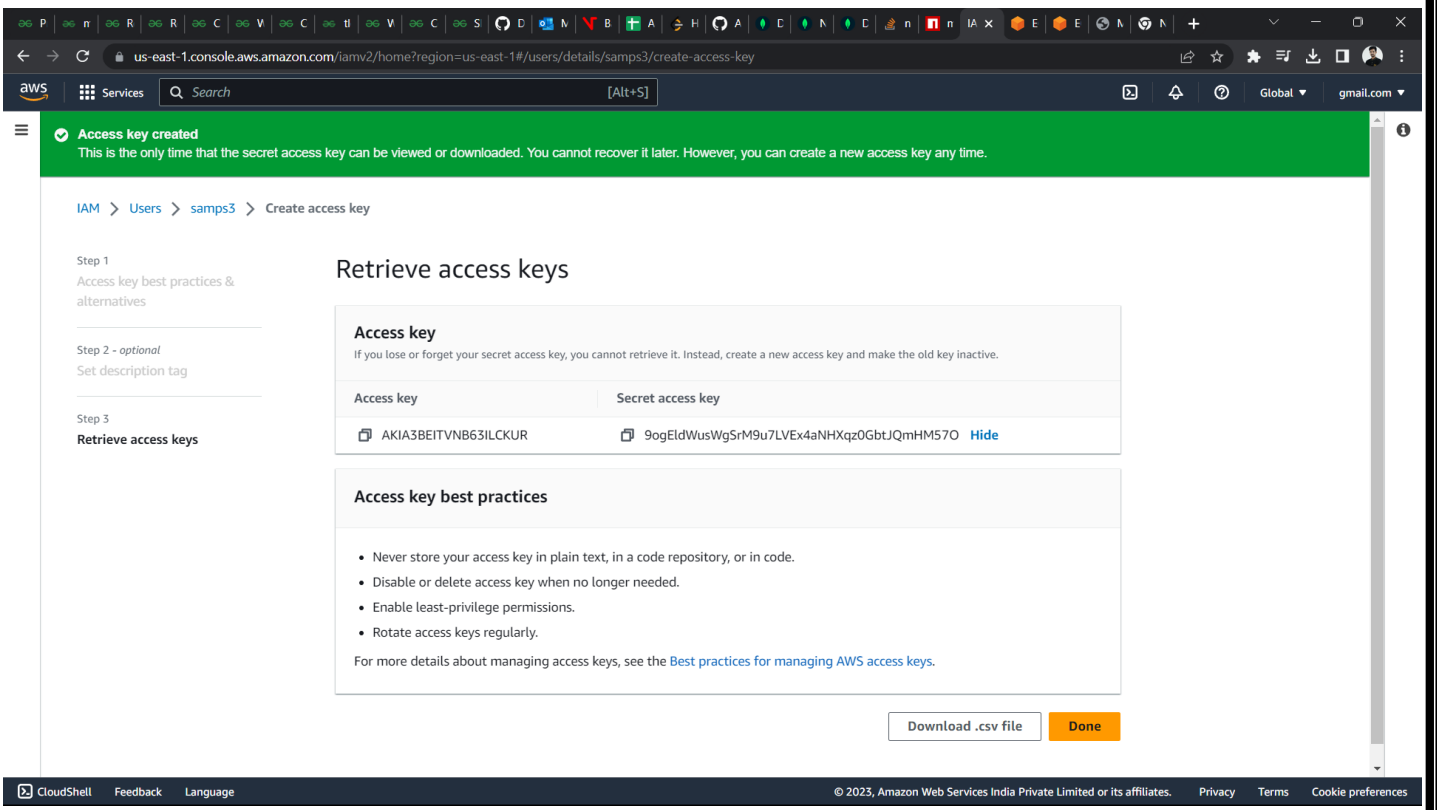

Not secure | 34.239.166.143:3000
Poster:
Choose File bh2.jpg
Upload Movie
Get Movies
Title: iron man 3
Director: Jon Favreau
Release Year: 2013
Title: Bahubali 2
Director: Rajamouli
Release Year: 2017

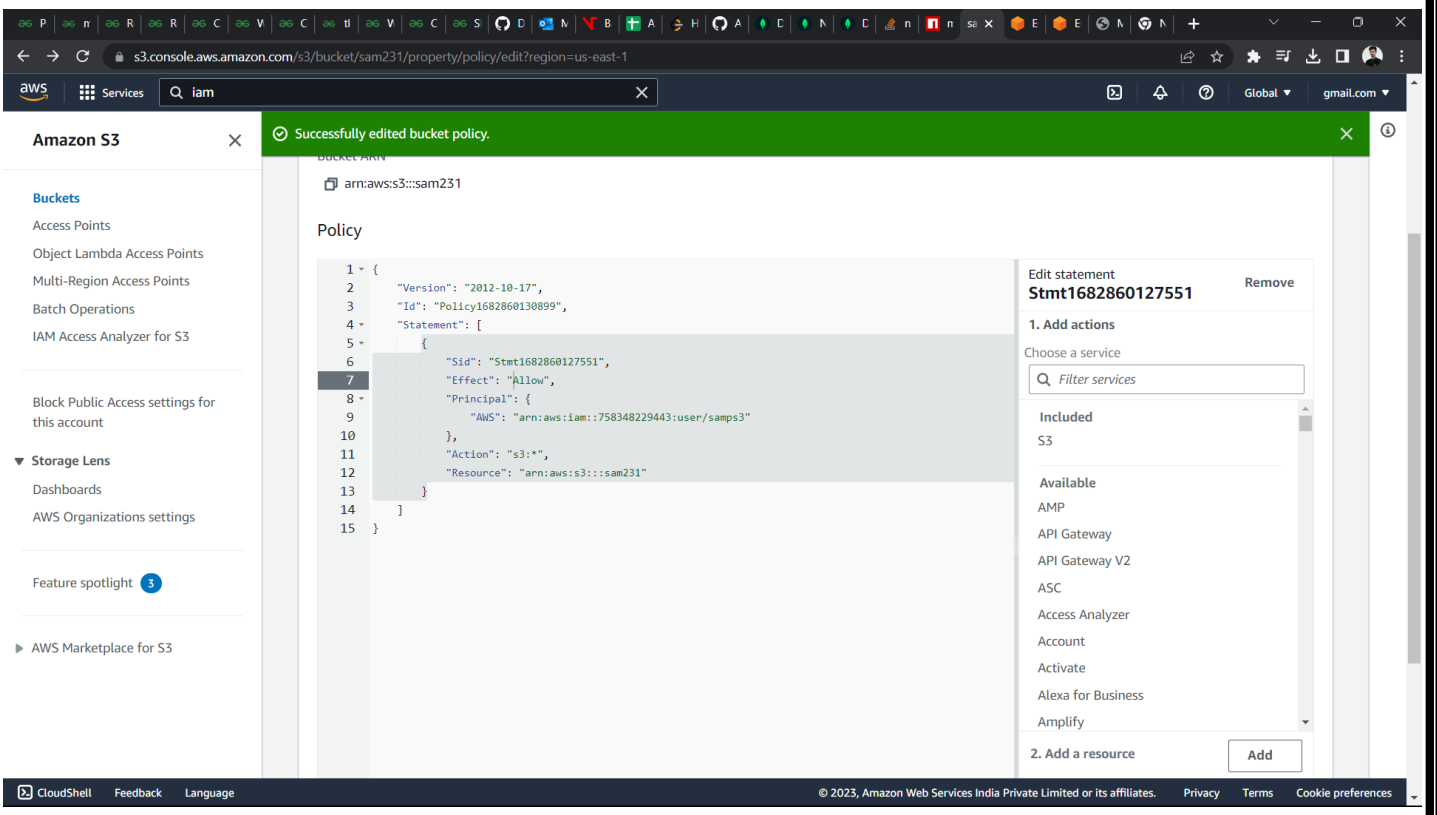


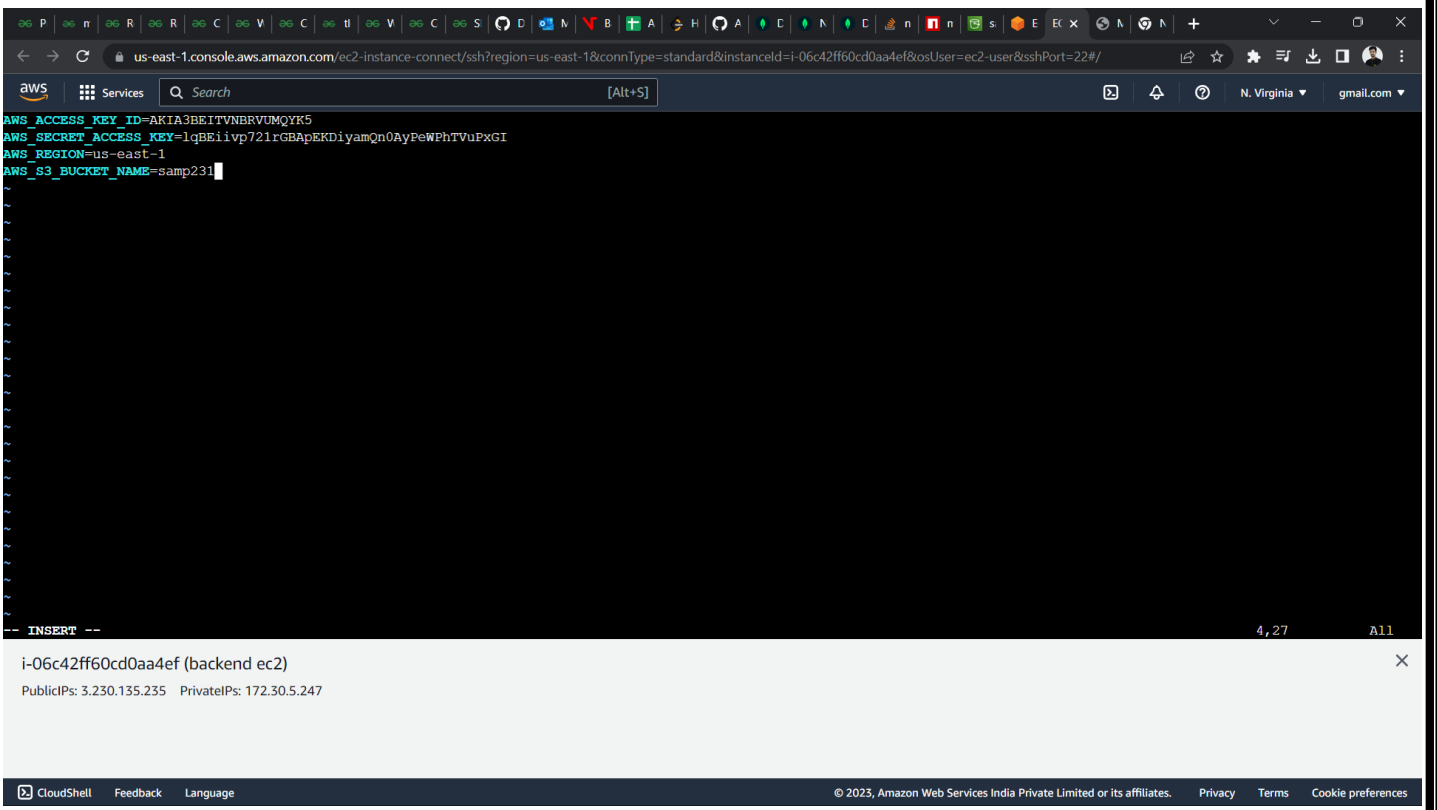
2

# Step 3: Creating S3 Bucket and uploading objects.



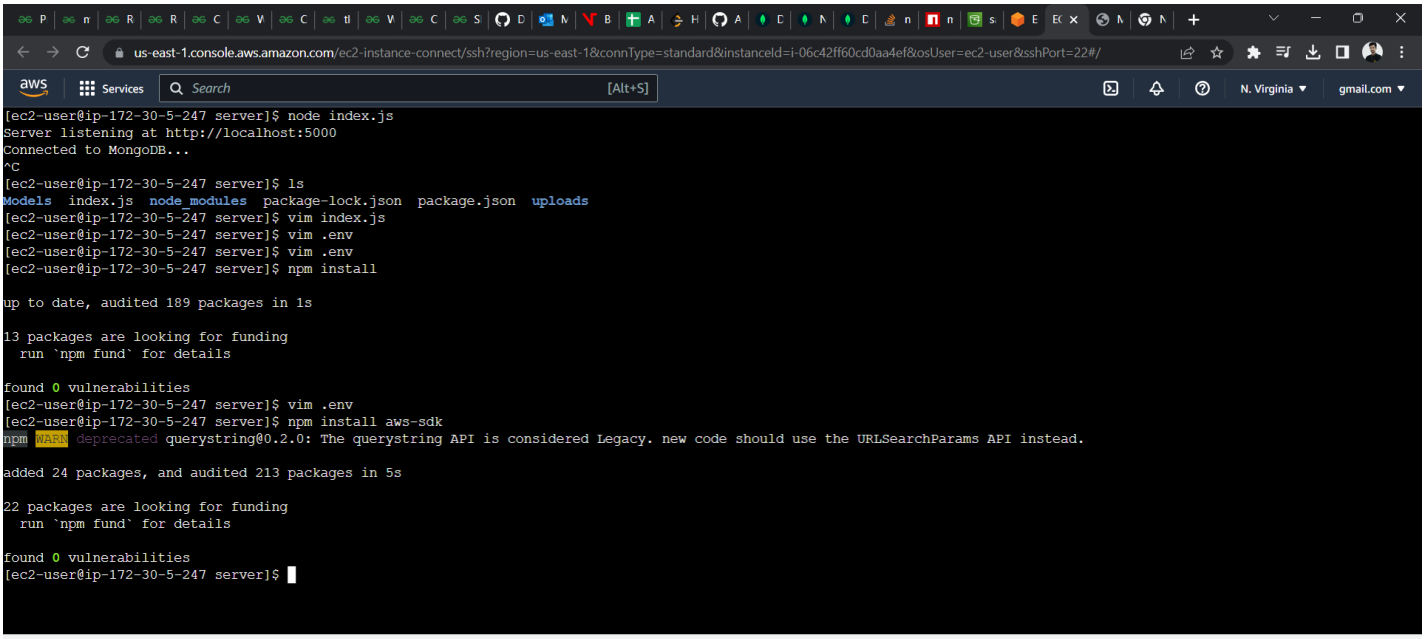



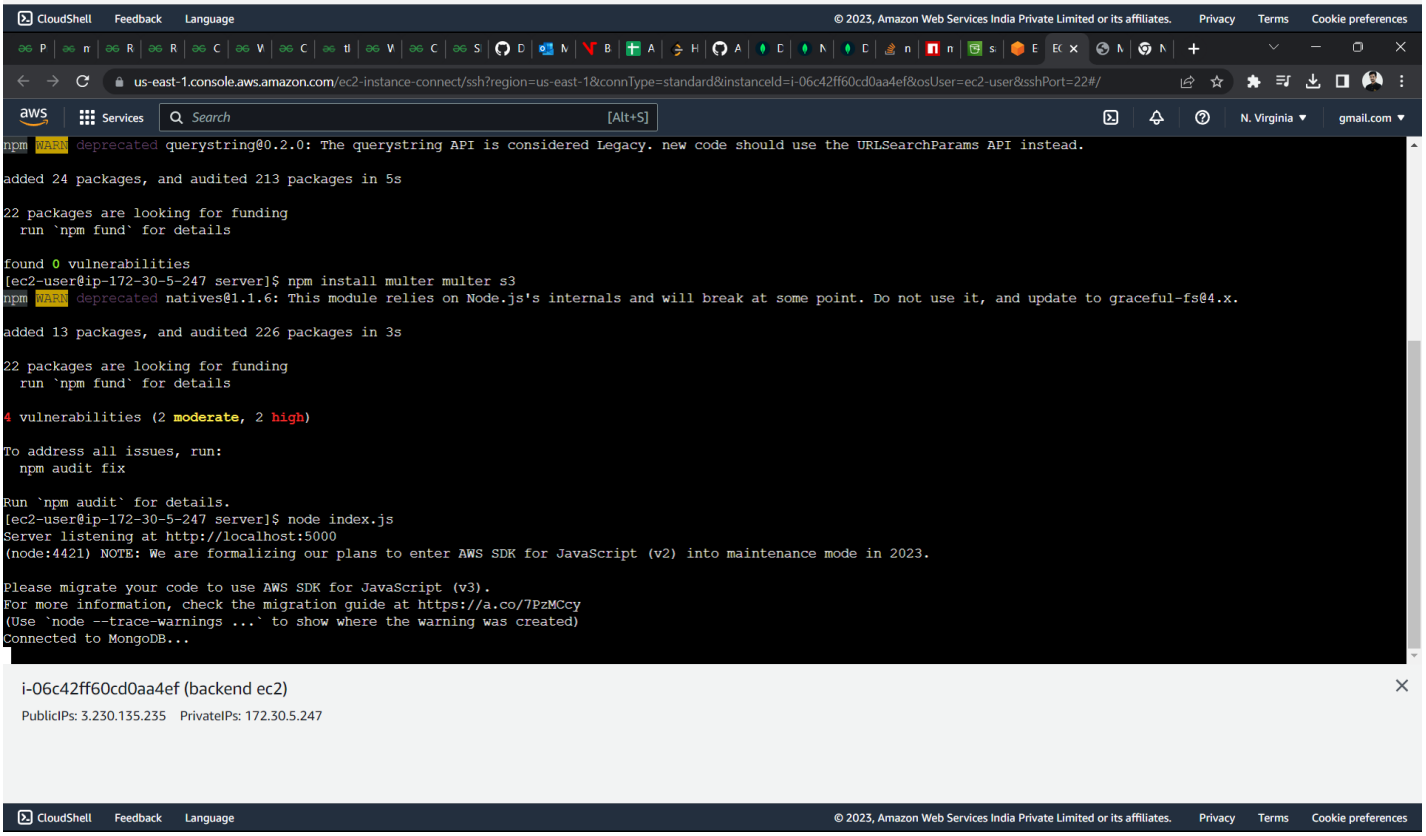

s3.console.aws.amazon.com/s3/bucket/sam231/property/policy/edit?region=us-east-1
Services
iam
Global
gmail.com
Amazon S3
Buckets
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
AWS Organizations settings
Feature spotlight
AWS Marketplace for S3
Successfully edited bucket policy.
arn:aws:s3:::sam231
Policy
{
"Version": "2012-10-17",
"Id": "Policy1682860130899",
"Statement": [
{
"Sid": "Stmt1682860127551",
"Effect": "Allow",
"Principal": {
"AWS": "arn:aws:iam::758348229443:user/samps3"
},
"Action": "s3:*",
"Resource": "arn:aws:s3:::sam231"
}
]
}
Edit statement
Stmt1682860127551
Remove
1. Add actions
Choose a service
Filter services
Included
S3
Available
AMP
API Gateway
API Gateway V2
ASC
Access Analyzer
Account
Activate
Alexa for Business
Amplify
2. Add a resource
Add
CloudShell
Feedback
Language
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences


us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?region=us-east-1&connType=standard&instanceId=i-06c42ff60cd0aa4ef&osUser=ec2-user&sshPort=22#/
Services
Search
[Alt+S]
N. Virginia
gmail.com
AWS_ACCESS_KEY_ID=AKIA3BEITVNBRVUMQYK5
AWS_SECRET_ACCESS_KEY=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
AWS_REGION=us-east-1
AWS_S3_BUCKET_NAME=samp231
-- INSERT --
4,27
All
i-06c42ff60cd0aa4ef (backend ec2)
PublicIPs: 3.230.135.235 PrivateIPs: 172.30.5.247
CloudShell
Feedback
Language
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences



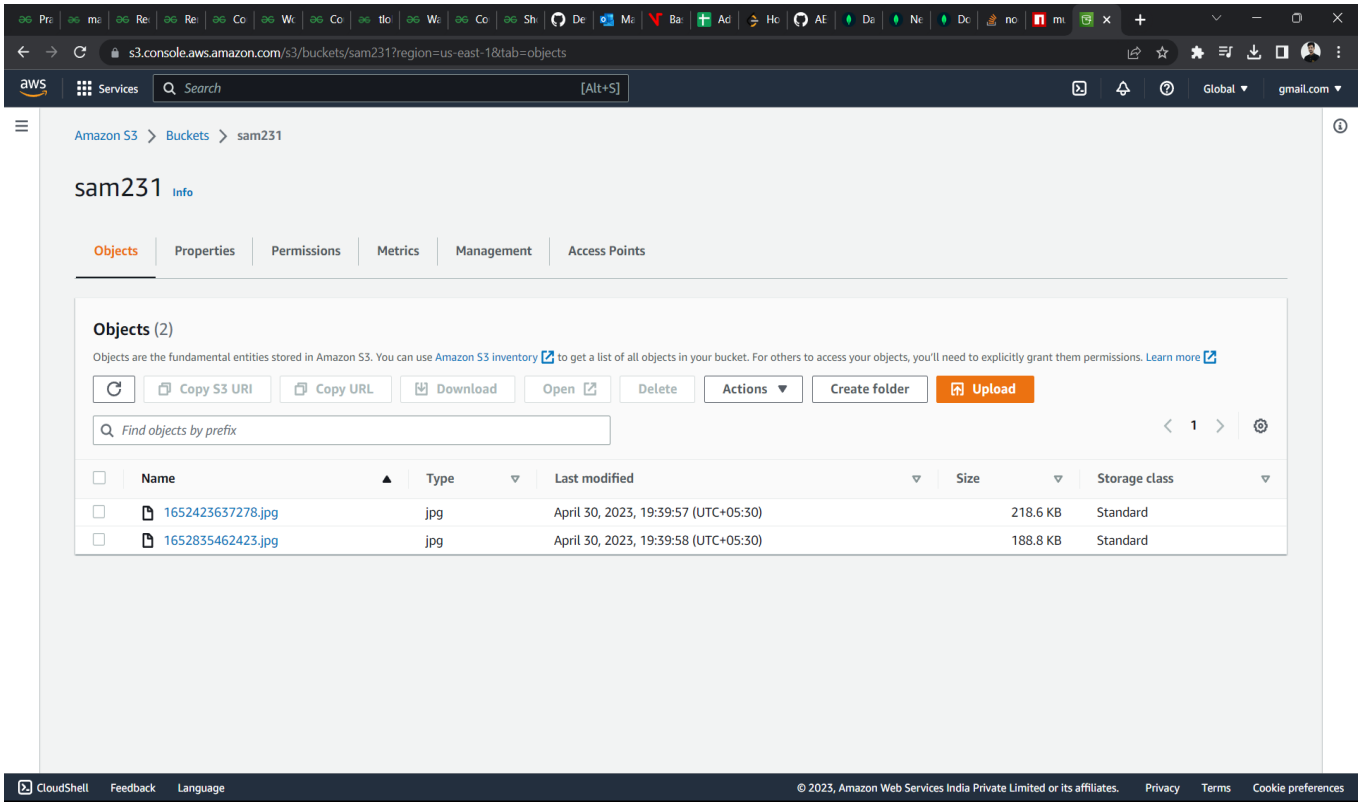
2


us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?region=us-east-1&connType=standard&instanceId=i-06c42ff60cd0aa4ef&osUser=ec2-user&sshPort=22#/
Services
Search
[Alt+S]
N. Virginia
gmail.com
[ec2-user@ip-172-30-5-247 server]$ node index.js
Server listening at http://localhost:5000
Connected to MongoDB...
^C
[ec2-user@ip-172-30-5-247 server]$ ls
Models index.js node_modules package-lock.json package.json uploads
[ec2-user@ip-172-30-5-247 server]$ vim index.js
[ec2-user@ip-172-30-5-247 server]$ vim .env
[ec2-user@ip-172-30-5-247 server]$ vim .env
[ec2-user@ip-172-30-5-247 server]$ npm install
up to date, audited 189 packages in 1s
13 packages are looking for funding
run `npm fund` for details
found 0 vulnerabilities
[ec2-user@ip-172-30-5-247 server]$ vim .env
[ec2-user@ip-172-30-5-247 server]$ npm install aws-sdk
npm WARN deprecated querystring@0.2.0: The querystring API is considered Legacy. new code should use the URLSearchParams API instead.
added 24 packages, and audited 213 packages in 5s
22 packages are looking for funding
run `npm fund` for details
found 0 vulnerabilities
[ec2-user@ip-172-30-5-247 server]$
i-06c42ff60cd0aa4ef (backend ec2)
PublicIPs: 3.230.135.235 PrivateIPs: 172.30.5.247


CloudShell
Feedback
Language
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences
us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?region=us-east-1&connType=standard&instanceId=i-06c42ff60cd0aa4ef&osUser=ec2-user&sshPort=22#/
Services
Search
[Alt+S]
N. Virginia
gmail.com
npm WARN deprecated querystring@0.2.0: The querystring API is considered Legacy. new code should use the URLSearchParams API instead.
added 24 packages, and audited 213 packages in 5s
22 packages are looking for funding
run `npm fund` for details
found 0 vulnerabilities
[ec2-user@ip-172-30-5-247 server]$ npm install multer multer s3
npm WARN deprecated natives@1.1.6: This module relies on Node.js's internals and will break at some point. Do not use it, and update to graceful-fs@4.x.
added 13 packages, and audited 226 packages in 3s
22 packages are looking for funding
run `npm fund` for details
4 vulnerabilities (2 moderate, 2 high)
To address all issues, run:
npm audit fix
Run `npm audit` for details.
[ec2-user@ip-172-30-5-247 server]$ node index.js
Server listening at http://localhost:5000
(node:4421) NOTE: We are formalizing our plans to enter AWS SDK for JavaScript (v2) into maintenance mode in 2023.
Please migrate your code to use AWS SDK for JavaScript (v3).
For more information, check the migration guide at https://a.co/7PzMCcy
(Use `node --trace-warnings ...` to show where the warning was created)
Connected to MongoDB...
i-06c42ff60cd0aa4ef (backend ec2)
PublicIPs: 3.230.135.235 PrivateIPs: 172.30.5.247
CloudShell
Feedback
Language
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences



2

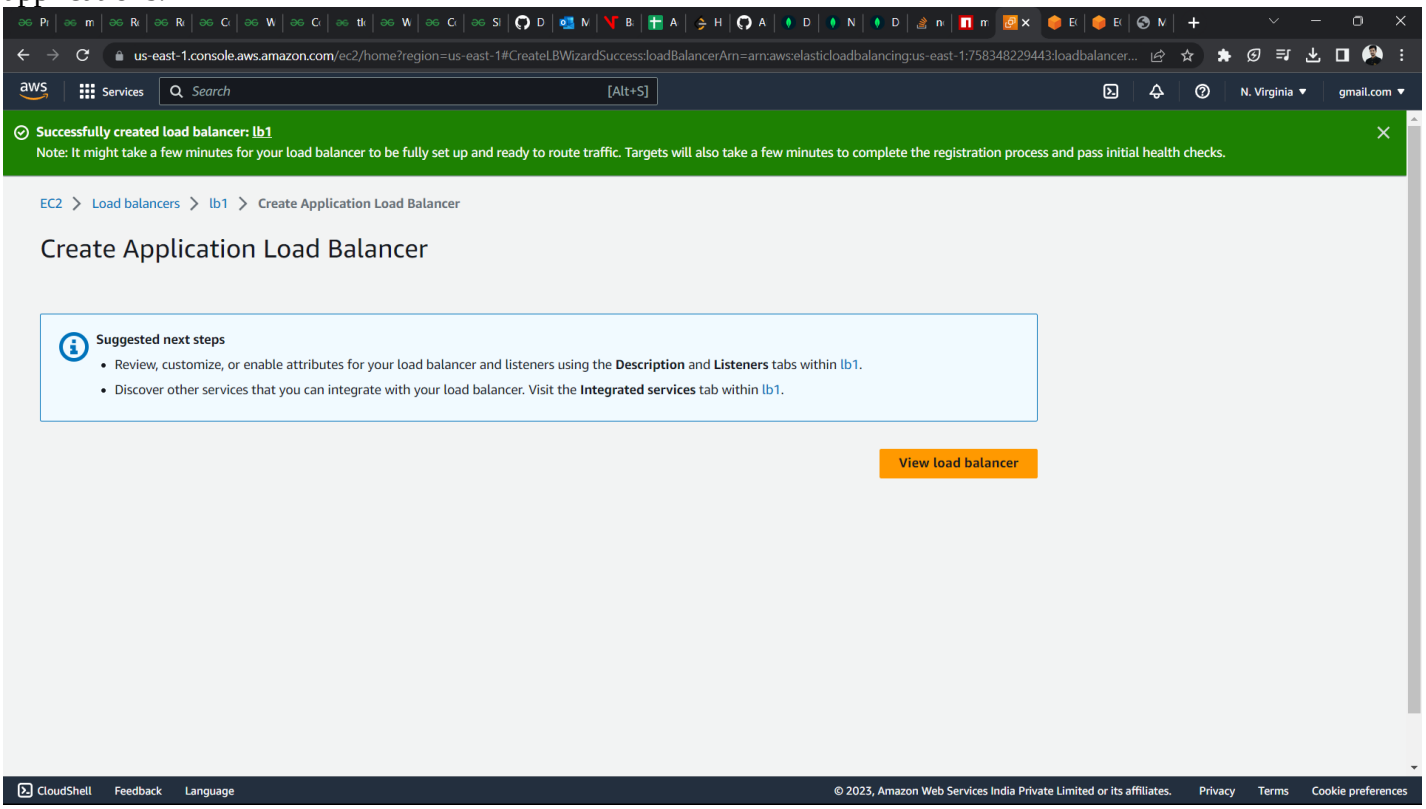

s3.console.aws.amazon.com/s3/buckets/sam231?region=us-east-1&tab=objects
aws
Services
Search
[Alt+S]
Global
gmail.com
Amazon S3 > Buckets > sam231
sam231 Info
Objects
Properties
Permissions
Metrics
Management
Access Points
Objects (2)
Objects are the fundamental entities stored in Amazon S3. You can 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
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Find objects by prefix
1
Name
Type
Last modified
Size
Storage class
1652423637278.jpg
jpg
April 30, 2023, 19:39:57 (UTC+05:30)
218.6 KB
Standard
1652835462423.jpg
jpg
April 30, 2023, 19:39:58 (UTC+05:30)
188.8 KB
Standard
CloudShell
Feedback
Language
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences

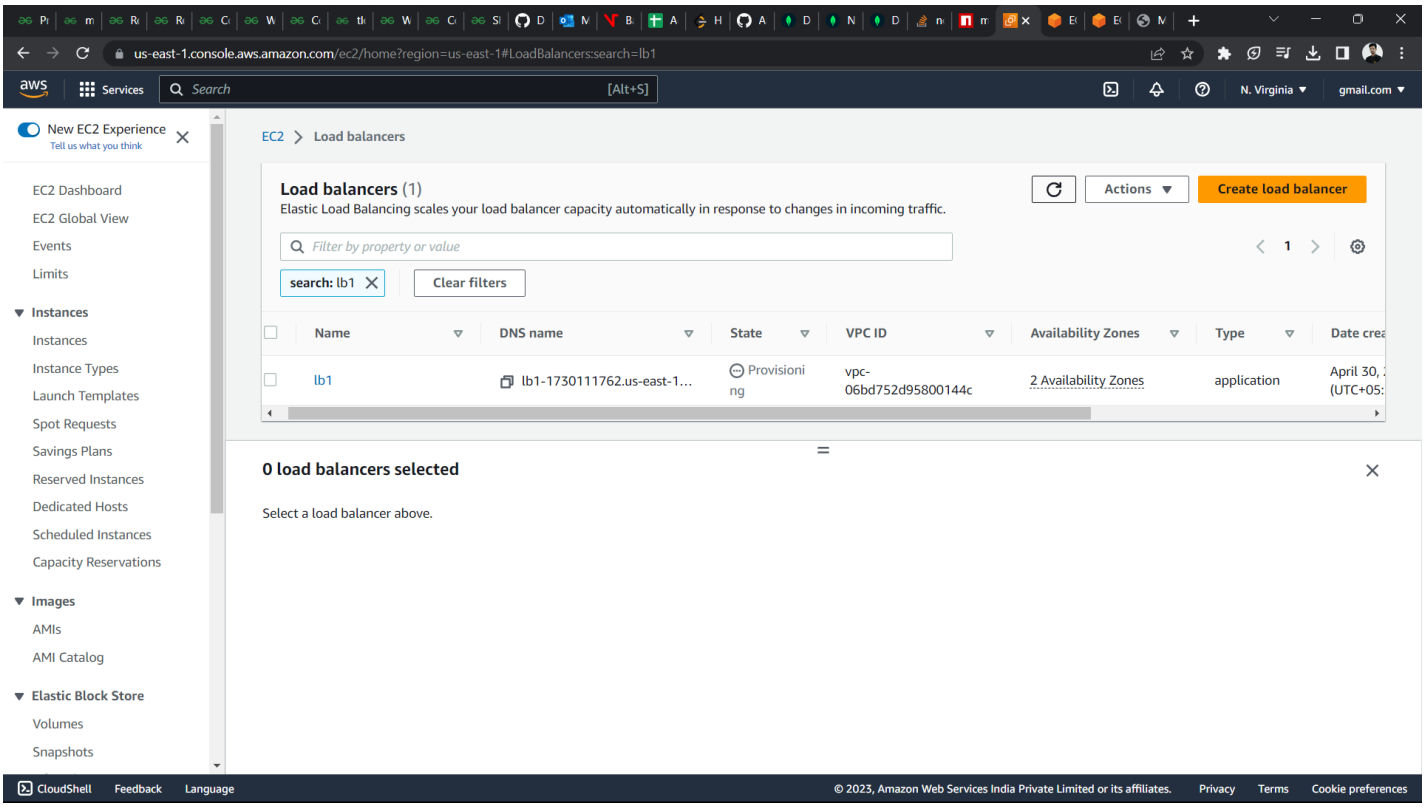


2

## Step 4: Attaching load balancer

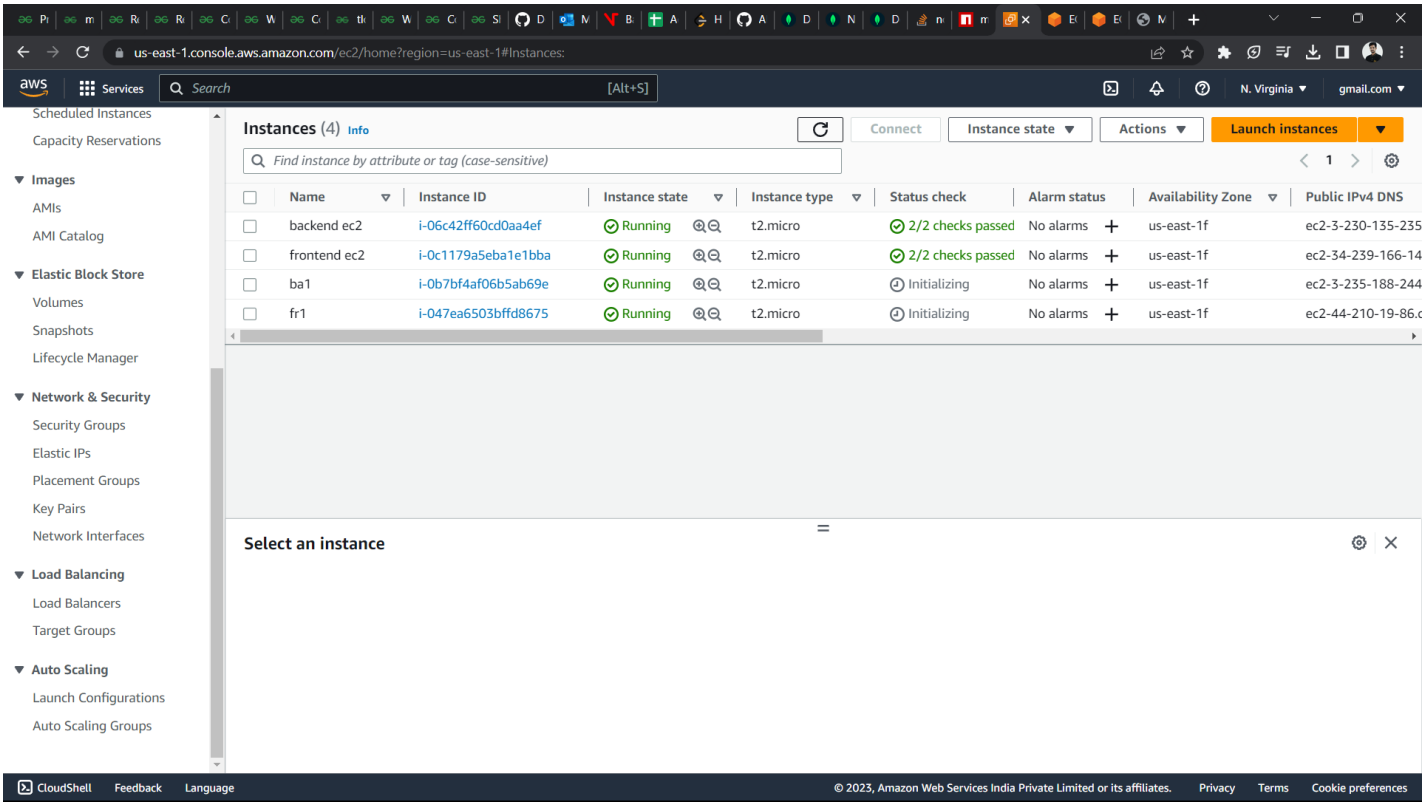
Create the load balancer which is used to increase capacity (concurrent users) and reliability of applications.

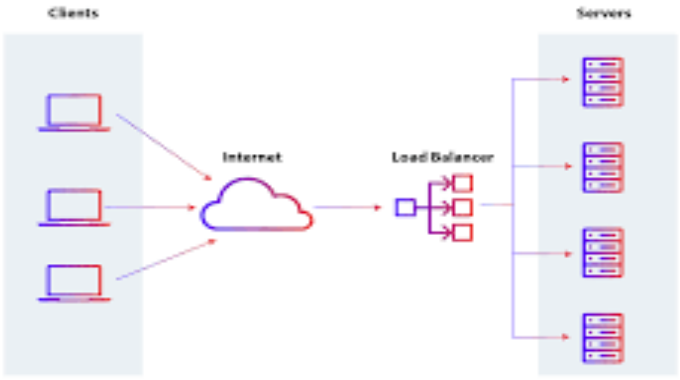





2

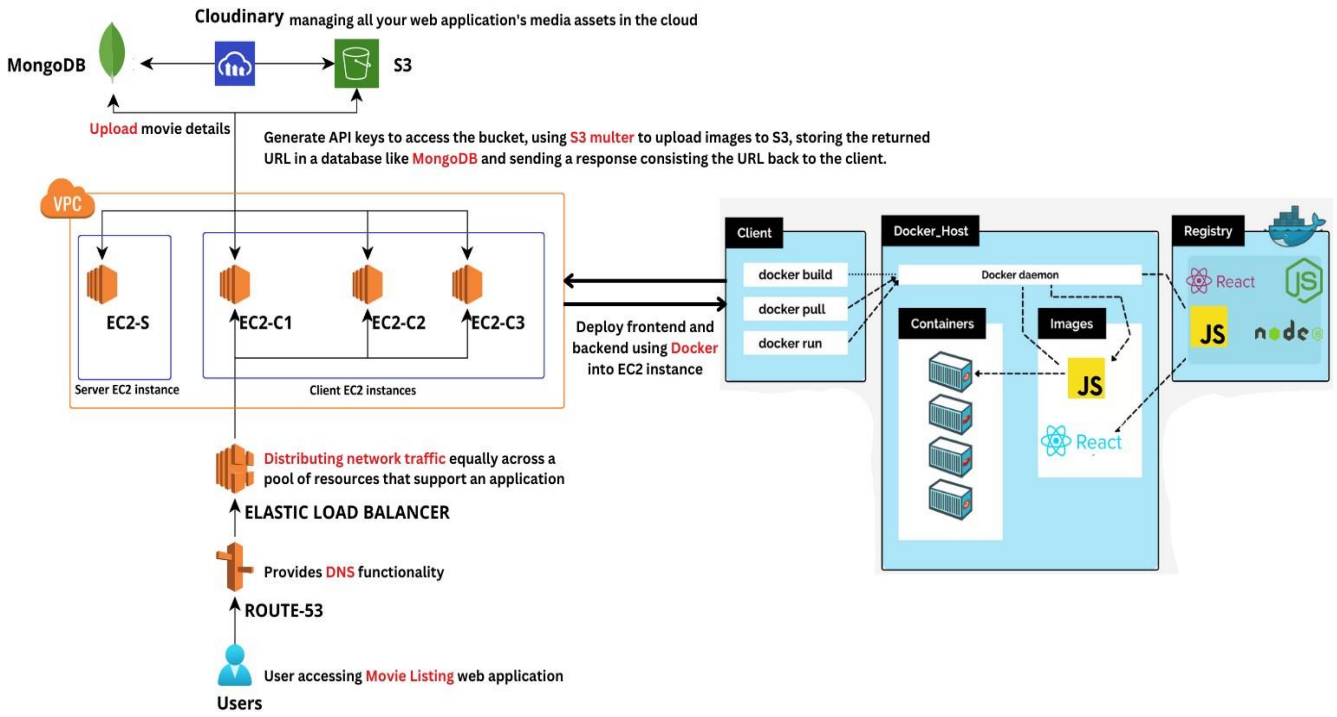

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:
Services
Search
[Alt+S]
N. Virginia
gmail.com
Scheduled Instances
Capacity Reservations
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Load Balancers
Target Groups
Auto Scaling
Launch Configurations
Auto Scaling Groups
Instances (4) Info
Connect
Instance state
Actions
Launch instances
Find instance by attribute or tag (case-sensitive)
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv4 DNS
backend ec2
i-06c42ff60cd0aa4ef
Running
t2.micro
2/2 checks passed
No alarms
us-east-1f
ec2-3-230-135-235
frontend ec2
i-0c1179a5eba1e1bba
Running
t2.micro
2/2 checks passed
No alarms
us-east-1f
ec2-34-239-166-14
ba1
i-0b7bf4af06b5ab69e
Running
t2.micro
Initializing
No alarms
us-east-1f
ec2-3-235-188-244
fr1
i-047ea6503bffd8675
Running
t2.micro
Initializing
No alarms
us-east-1f
ec2-44-210-19-86.
Select an instance
CloudShell
Feedback
Language
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences


Clients
Internet
Load Balancer
Servers



2

## AWS DEPLOYMENT DIAGRAM:



This project includes the utilization of various tools. Multiple observations are noted based on the tools. Some key observations are:

**Visual Studio Code**: Visual Studio Code is a source-code editor that can be used with a variety of programming languages. The code has been according to the requirements using VS Code.

**MongoDB Atlas**: MongoDB Atlas is a fully-managed cloud database that handles all the complexity of deploying, managing, and healing your deployments on the cloud service provider of your choice (AWS, Azure, and GCP). The local database has been replaced by this database and the data uploaded is stored here.

**EC2**: EC2 (Elastic Compute Cloud) is a web service provided by Amazon Web Services (AWS) that allowsusers to rent virtual computers, known as instances, on which they can run their own applications. The containerized frontend and backend have been deployed using EC2.The client instances are also attached to the load balancer to manage the traffic.

**S3:** Amazon S3 (Simple Storage Service) is a cloud-based storage service that enables users to store and retrieve data from anywhere on the internet. The images we have uploaded in the frontend have been stored in S3 buckets which was created.

**IAM:** AWS Identity and Access Management (IAM) is a web service provided by Amazon Web Services (AWS) that enables users to securely manage access to AWS resources. A user have been created for the access controls and full administrator access was given.

**Route 53:** Amazon Route 53 is a highly available and scalable cloud-based Domain Name System (DNS) service provided by Amazon Web Services (AWS). It helps the developers and business owners by translating the domain name into IP address. DNS has been assigned which can be accessed by anyone to view the website.

**GitHub:** GitHub is a web-based platform that provides developers with a variety of tools for software development and collaboration. The project has been uploaded in GitHub which would be accessible for future changes.

A project requires clear objectives and ground rules to work on to achieve the goal of any aspect.

Jira has been used to plan the sprints, track the changes and complete the project efficiently.

Slack a great communication tool where we can interact efficiently and to collaborate.
Zoom is a collaboration app that enabled conversations.

## TASKS ASSIGNED FOR TEAM MEMBERS:

During the project, we distributed the workload among team members based on their individual strengths and interests.

We communicated regularly and updates on our project management tool through WhatsApp group.

In the "Tasks Assigned" column, we listed each task thatneeded to be completed for the project, along with a brief description of the task. In the "Team Members"column, we listed the name of the team member who was responsible for completing that task.

By using this spreadsheet, we were able to keep track of who was responsible for each task and ensure that everyone was contributing equally to the project**.** We updated the spreadsheet regularly to reflect any changes to the task assignments, and we used it as a reference during our team meetingsto discuss progress and identify any issues.

| S No. | TASKS ASSIGNED | TEAM MEMBERS |
|---|---|---|
| 1 | Connecting the Server with Atlas MongoDB. | Siva durga, Siva |
| 2 | Creating IAM user and Configuring S3-bucket. | Vivek, Chitti, Suvarna |
| 3 | Configuring the Application Code with S3-multer. | BangarRaju, Siva durga |
| 4 | Containerization of the code using Dockerfile. | Siva, Ankitha |
| 5 | Deploying server on EC2 using Docker. | Suvarna, Siva durga |
| 6 | Deploying client on EC2 using Docker. | Ankitha, Deepika |
| 7 | Creation a target group and Load Balancer. | Siva, Vivek, Chitti |
| 8 | Creating AWS Deployment Diagram. | Deepika, Bangaraju |
| 9 | Preparing Project Documentation. | Chitti , Vivek |

Overall, we believe that our team was successful in distributing the workload effectively and working together to achieve our goals. By outlining our roles and responsibilities, maintaining open communication, and evaluating our performance, we were able to complete the project on time andto a high standard.

## CHALLENGES FACED

- Due to lack of node of react knowledge, it took lot of time to analyses the code.
- We face most of the issues while modifying the server code for configuring with S3-multer.
- Due to lack of good internet, Docker image creation took a lot of time.
- When something went wrong, we had to reframe the frontend (connection) and dockarize the frontend code again.
- Due to our lack of knowledge on how to create an AWS architecture diagram, we had to learn how to work with different editing tools like Canva and figma...etc.

## RESOURCES

**MongoDB :** https://www.mongodb.com/docs/

**AWS :** https://docs.aws.amazon.com/

**S3-multer :** https://www.npmjs.com/package/multer-s3

**Docker :** https://docs.docker.com/

**Git :** https://git-scm.com/docs

**AWS Architecture Diagram Developing tools:** https://aws.amazon.com/architecture/icons/

## CONCLUSION

At the end of this project, we successfully deployed frontend using docker into EC2 instance. The given capstone project involves using ReactJS as Frontend, NodeJS as Backend, MongoDB as Database, multer in Backend for file upload and cloud infrastructure (AWS) to deploy. The deployed frontend application gives us the required "Movie Listing" web site in which users can upload movie details.

For achieving this, we used multer-s3 library for storing images, replaced local database with Atlas MongoDB cloud infrastructure to take the database into the cloud, deployed backend and frontend in EC2 instances using Docker and attached Elastic IP to this instance and modified the Frontend Code to be able to fetch data from Backend.

The frontend user interface of the application having 4 fields like **Title, Director, Release year** and **Poster**. And having 2 functions **upload Movie** and **Get Movies**. When the user enters the details of the movie the data will be stored in mongoDB database. The poster field having images these images are stored in S3 bucket using S3-multer and by using cloudinary which provides a URL of thatimage which is stored in mongoDB database. When the user clicks on Get Movies button the user willget details of all the movies which are stored in mongoDB database and images are fetched from S3 bucket.