CAPSTONE PROJECT

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

GROUP-1 AND TEAM MEMBERS

- EMMANUEL DOKKADI (21P35A0347)
- POTHAMSETTI BHAVYA PRASANNA REDDY (20A91A04G7)
- NALLAMILLI PRIYANKA (20A91A04G0)
- CHALAPATHI CHAITANYA NEELIMA (21A95A0410)
- KAREDLA VIJAYA (21A95A0414)
- ANGADA VEERA VENKATA NITHYA SRI (21A95A0408)
- DAMMU HARI PRIYANKA (21A95A0411)
- NERUSUPALLI MEHER FREINEY (20P31A0439)

INDEX

1. OBJECTIVE OF THE PROJECT

2. PURPOSE OF THE PROJECT

3. SCOPE OF THE PROJECT

4. TECHNOLOGIES USED IN THIS PROJECT

5. TECHNOLOGIES STACK

6. MAJOR TOOLS ANFD SERVICES UTILIZED IN THIS PROJECT

7. IMPLEMENTATION

8. AWS DEPLOYMENT ARCHITECHTURE

9. CONCLUSION

OBJECTIVE OF THE PROJECT

The objective is to deploy a movie listing website, built using ReactJS as frontend, NodeJS as backend, and MongoDB as the database, to the AWS cloud infrastructure with proper scaling. This involves choosing an appropriate AWS service for deployment, configuring the server, uploading the code, configuring the firewall, setting up a load balancer and automatic scaling, and monitoring the website's performance. The end result should be a highly available and scalable website that can handle high traffic volumes.

PURPOSE OF THE PROJECT

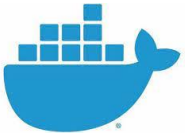
The purpose of the project is to create a movie listing website that provides users with a platform to upload and view movie details. Users will be able to upload movie details, including the title, description, and images, which will be stored in local storage. The website will be deployed to the AWS cloud infrastructure with proper scaling to ensure high availability and scalability to handle high traffic volumes. The project aims to showcase the student's skills in web development, cloud infrastructure deployment, and scaling using AWS services.

By completing this project, we will gain valuable experience in building full-stack web applications using the latest technologies and deploying them to the cloud.

SCOPE OF THE PROJECT

The scope of the project is to create a movie listing website that enables users to upload and view movie details, including the title, description, and images. The website will be built using ReactJS as the frontend technology, NodeJS as the backend technology, and MongoDB as the database technology. The scope of the project includes user authentication and authorization, data validation and error handling, search functionality, and deployment of the website to the AWS cloud infrastructure with proper scaling. The project's focus is on building a functional and secure website that can handle high traffic volumes and is highly available and scalable. The scope does not include advanced features such as movie reviews, ratings, or recommendations.

TECHNOLOGIES USED IN THIS PROJECT



- **AWS:** Amazon Web Services (AWS) is a cloud computing platform offered by Amazon that provides a wide range of cloud-based services, including compute, storage, databases, analytics, machine learning, and more. AWS allows businesses to scale and grow by providing on-demand access to a range of cloud-based computing resources that are flexible, scalable, and cost-effective. AWS offers a variety of services that can be used to build, deploy, and manage web applications, mobile apps, and other services, making it a popular choice for businesses of all sizes.

- **DOCKER:** Docker is an open-source platform that allows developers to build, package, and deploy applications as containers. Containers are lightweight, portable, and self-contained environments that contain an application and its dependencies. Docker provides a way to package and distribute applications as containers, making it easier to deploy and manage applications at scale. With Docker, developers can easily create, test, and deploy applications, which can be scaled up or down quickly to meet changing demands.

- **GIT:** Git is a distributed version control system that allows developers to manage and track changes to source code over time. With Git, developers can work collaboratively on the same codebase without worrying about overwriting each other's work. Git provides a way to create and manage different versions of a codebase, making it easier to track changes, revert to earlier versions, and collaborate on development. Git also provides a way to manage code branches, allowing developers to work on different features or versions of the codebase independently. Git has become the de facto standard for version control in software development and is widely used by developers worldwide.

TECHNOLOGIES STACK

- **Frontend:** ReactJS
- **Backend:** NodeJS
- **Database:** MongoDB

For uploading files, we are using multer in the backend.

MAJOR TOOLS AND SERVICES UTILIZED IN THIS PROJECT

These are the which are being used for completing this project:

- **MongoDB Atlas:** MongoDB Atlas is a fully-managed cloud database service that provides a secure and scalable platform for running MongoDB. MongoDB is a NoSQL database that is designed for handling large and complex data sets, and MongoDB Atlas makes it easy to deploy, manage, and scale MongoDB databases in the cloud.

- **EC2:** Amazon Elastic Compute Cloud (EC2) is a cloud-based computing service that provides resizable compute capacity in the cloud. EC2 allows users to create virtual machines, known as instances, and run applications and workloads on them. EC2 is a core component of Amazon Web Services (AWS) and is widely used for hosting web applications, running big data analytics, and performing other compute-intensive tasks in the cloud.

- **S3:** Amazon Simple Storage Service (S3) is a cloud-based storage service that provides scalable, secure, and highly available object storage in the cloud. S3 is a core component of Amazon Web Services (AWS) and is widely used for storing and retrieving data, hosting static websites, and managing data backups and archives.

- **IAM:** Amazon Identity and Access Management (IAM) is a web service that provides secure access control and user management for AWS resources. IAM allows users to control who can access their AWS resources and what actions they can perform on those resources.

- **ROUTE 53:** Amazon Route 53 is a cloud-based Domain Name System (DNS) web service that provides highly available and scalable domain name registration, routing and management. It enables users to register domain names, and route internet traffic to the AWS resources such as EC2 instances, S3 buckets, and load balancers.

- **Git hub:** GitHub is a web-based hosting service that provides version control using Git. It is primarily used for source code management and software development, but it can also be used for managing and collaborating on any kind of digital content.

- **Visual Studio Code (optional):** Visual Studio Code (VS Code) is a popular, cross-platform code editor developed by Microsoft. It provides a rich set of features for coding, debugging, and building applications, including syntax highlighting, code completion, debugging, and version control integration. VS Code supports a wide range of programming languages and frameworks, and it has a large and active community of developers who contribute to its plugins and extensions.

- **DOCKER HUB:** Docker Hub is a cloud-based registry service for storing and distributing Docker container images. It is the default registry for storing Docker images and is integrated with the Docker engine and Docker CLI, making it easy to store, share, and manage Docker images.

  Docker Hub allows users to upload and download images, create and manage repositories, and set up automated builds using Docker files. It also provides access to public images, making it easy to find and use pre-built images for popular software packages and tools.
- **LOAD BALANCER:** A load balancer typically works by monitoring the incoming traffic and distributing it across multiple servers using a variety of algorithms, such as round-robin, least connections, IP hash, and others. The load balancer can also perform health checks on the servers to ensure that they are available and can handle incoming requests.

  Load balancing provides several benefits, including improved performance, scalability, and availability. By distributing traffic across multiple servers, load balancing can handle more requests and provide faster response times than a single server.

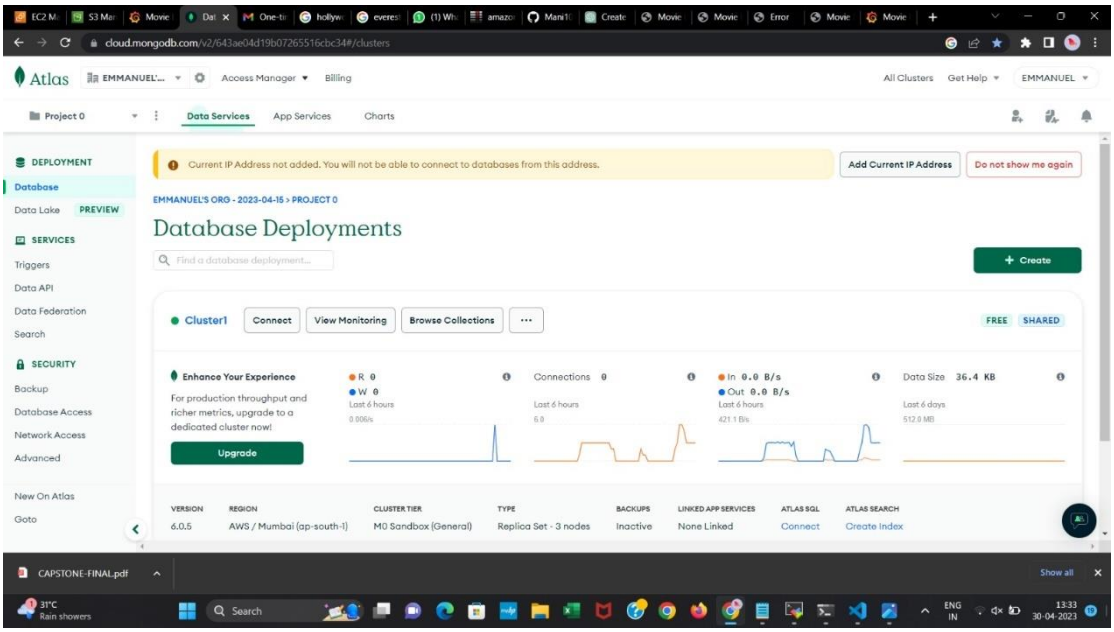
IMPLEMENTATION

STEP 1: Connecting server with MongoDB

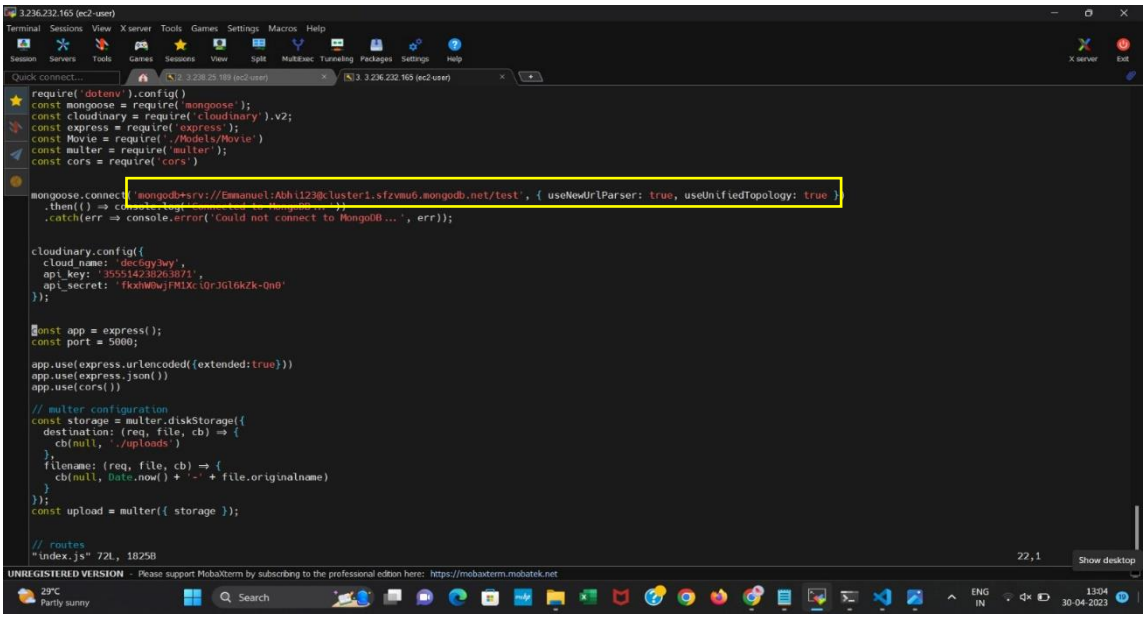
Initially, clone the repository given below into your local desktop:

https://github.com/snehal-herovired/DEVOPS_CAPSTONE

- Create an instance, add the public IP of that instance during the MongoDB creation.



- To connect your server with your MongoDB, replace the local host link with created MongoDB cluster link.

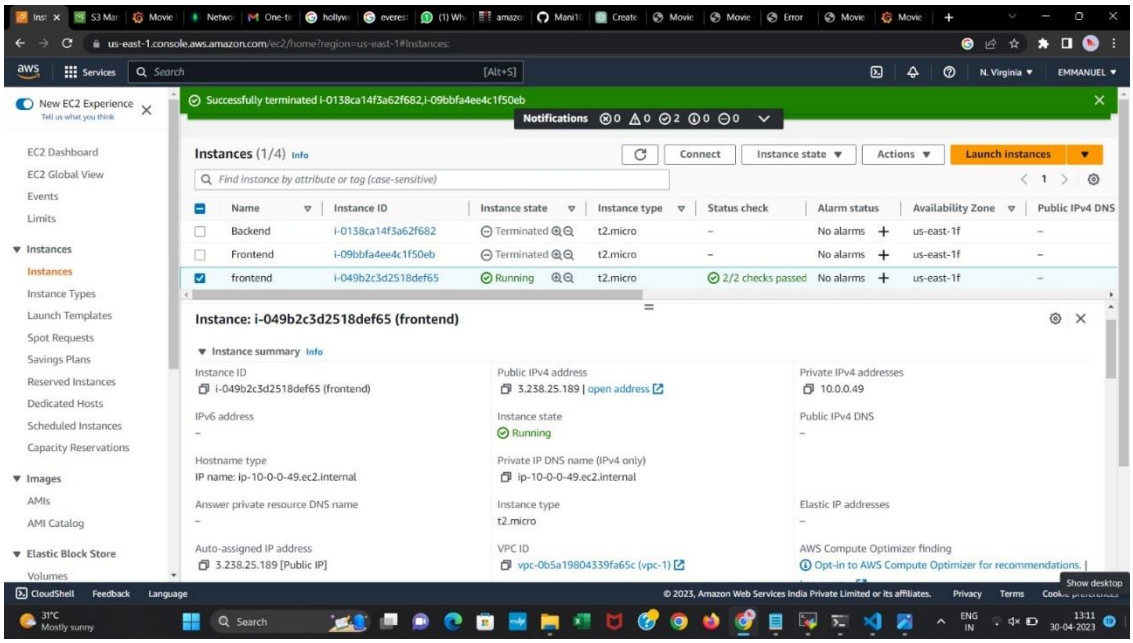


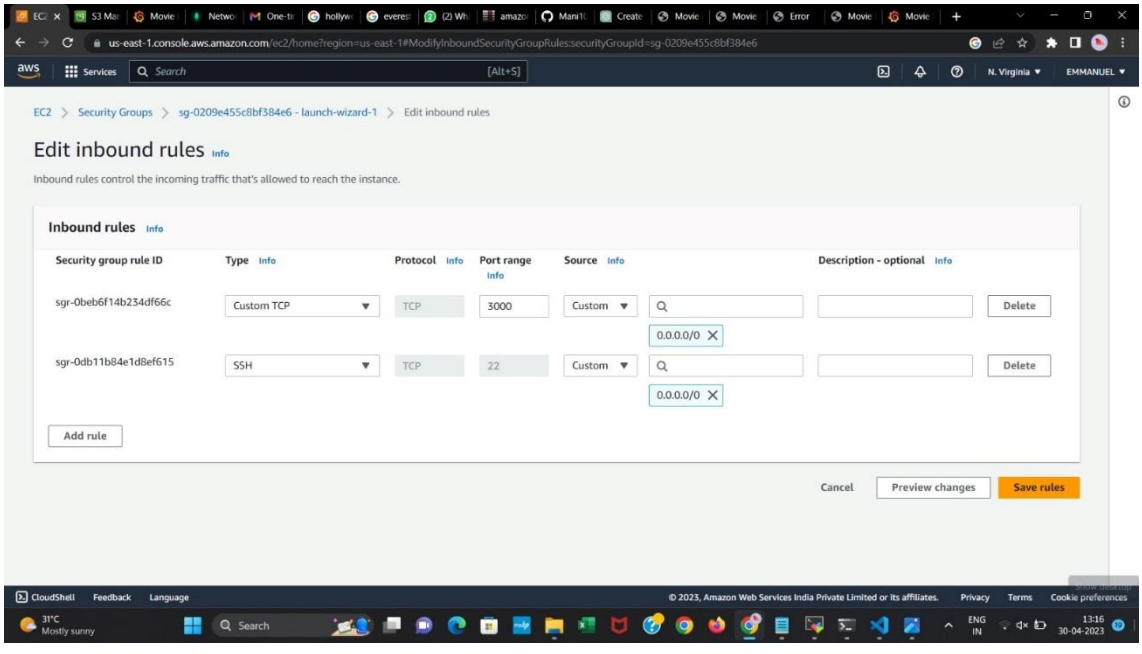
- Commands used to start the integrated terminal:
    1. npm install (for both client and server)
    2. npm start (for both client and server)
    3. node index.js (for both client and server)

STEP 2: Instance creation and security group configuration for front and backend.

<u>Frontend</u>:

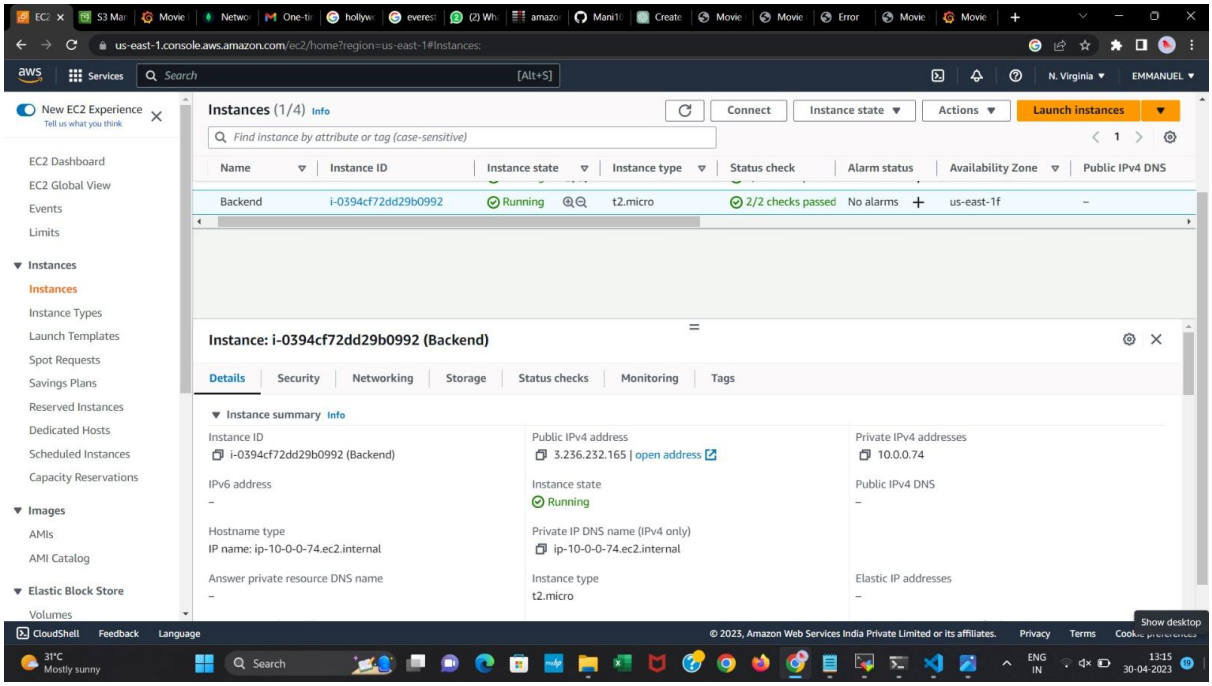
- Security groups port range should be 3000, anywhere (IPv4).

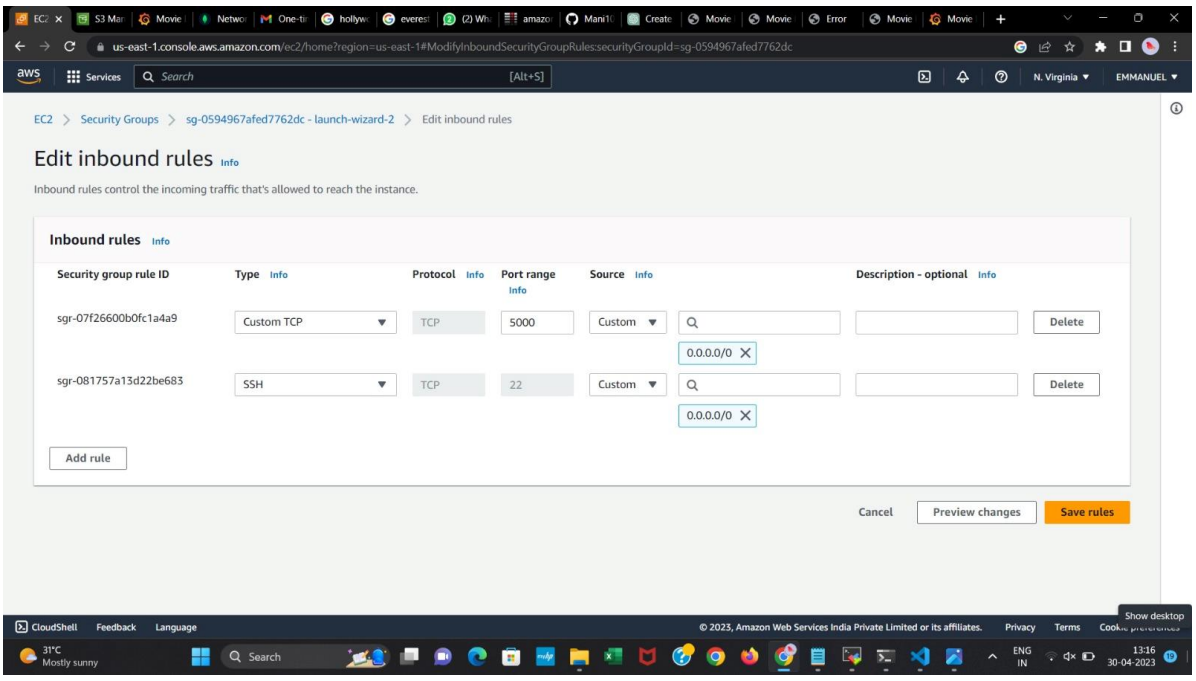




<u>BACKEND</u>:

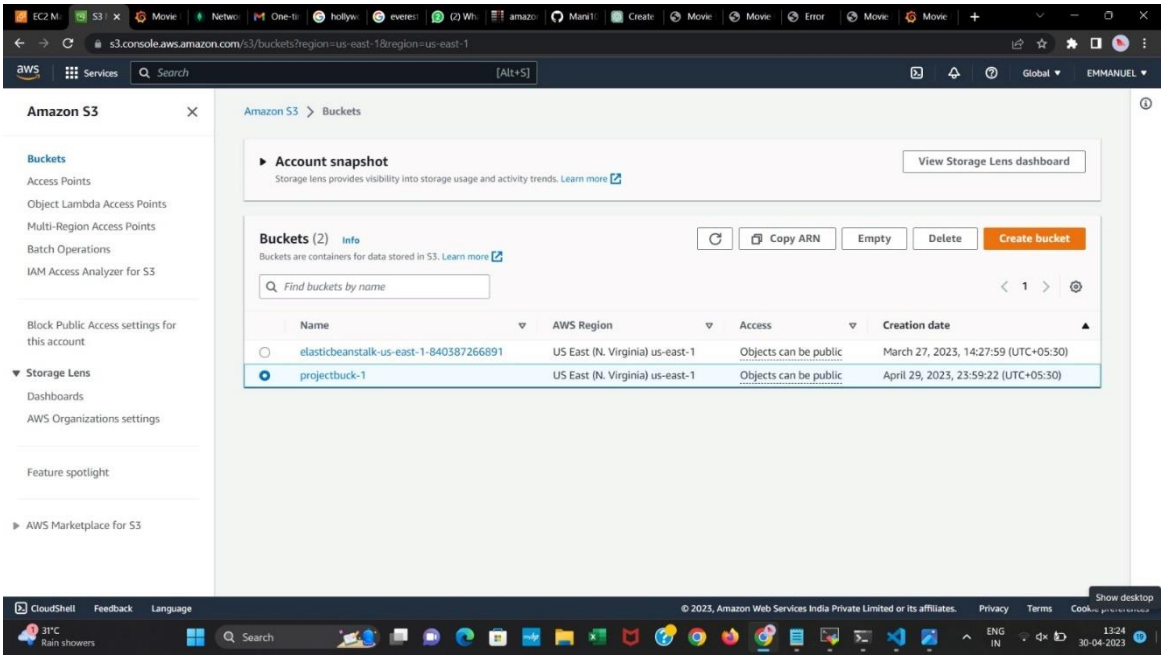
- Security groups port range should be 5000, anywhere (IPv4).



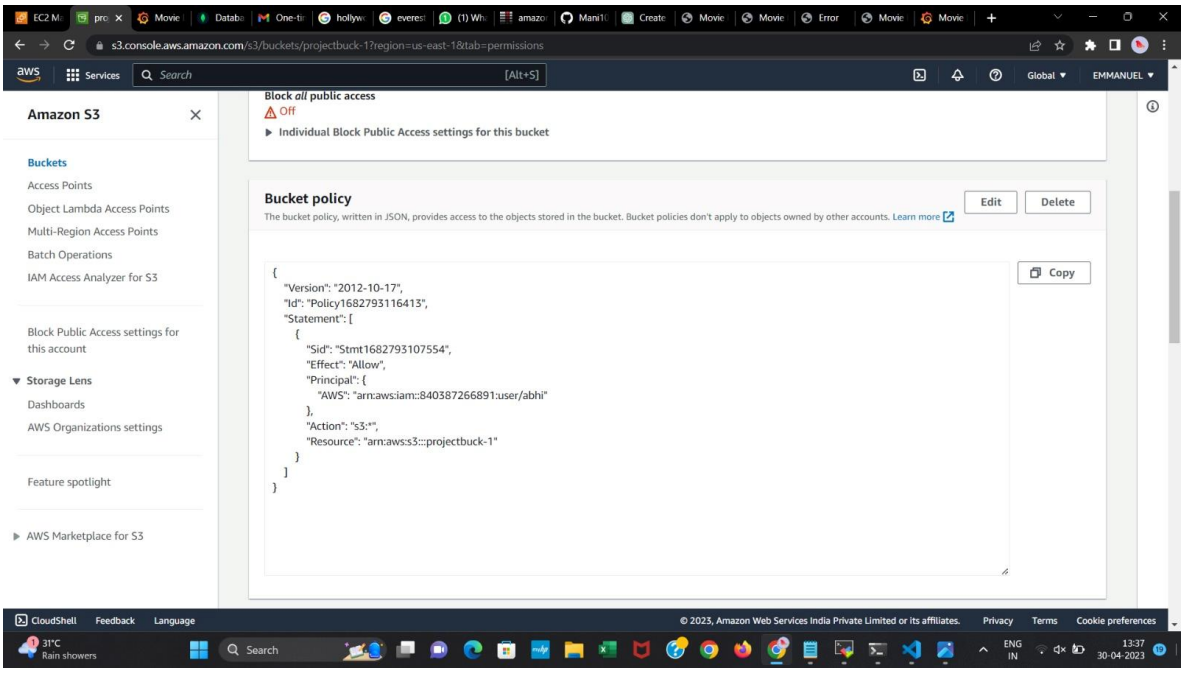


STEP 3: Create S3 bucket for storing the uploaded images

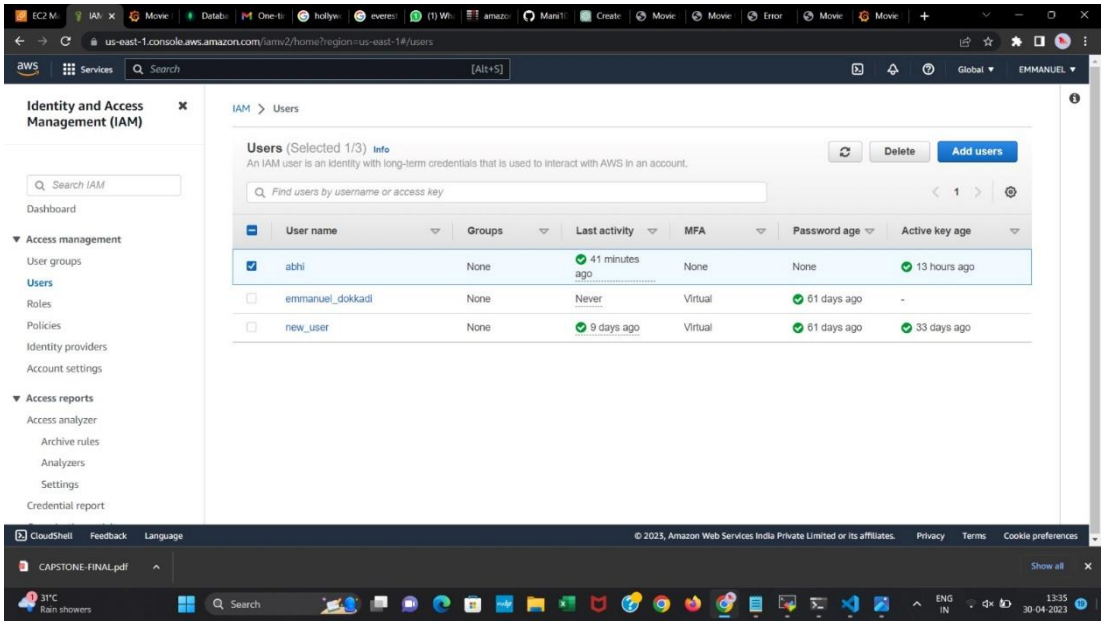
- Create S3 bucket for storing the images which are being uploaded in the website.



- Create a bucket policy.



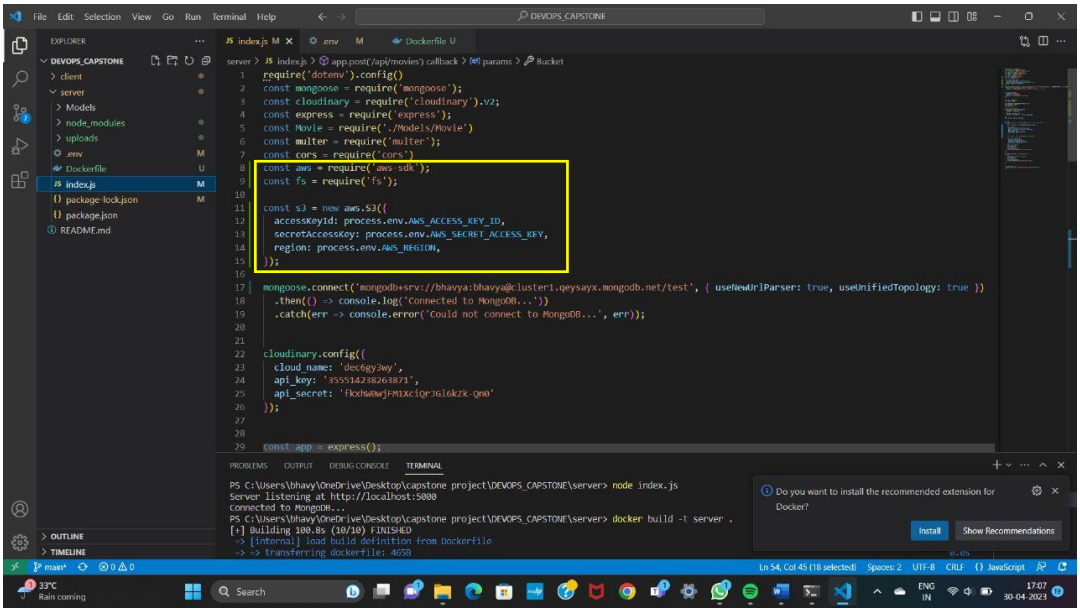
- Create IAM user to give access in order to upload images into bucket.

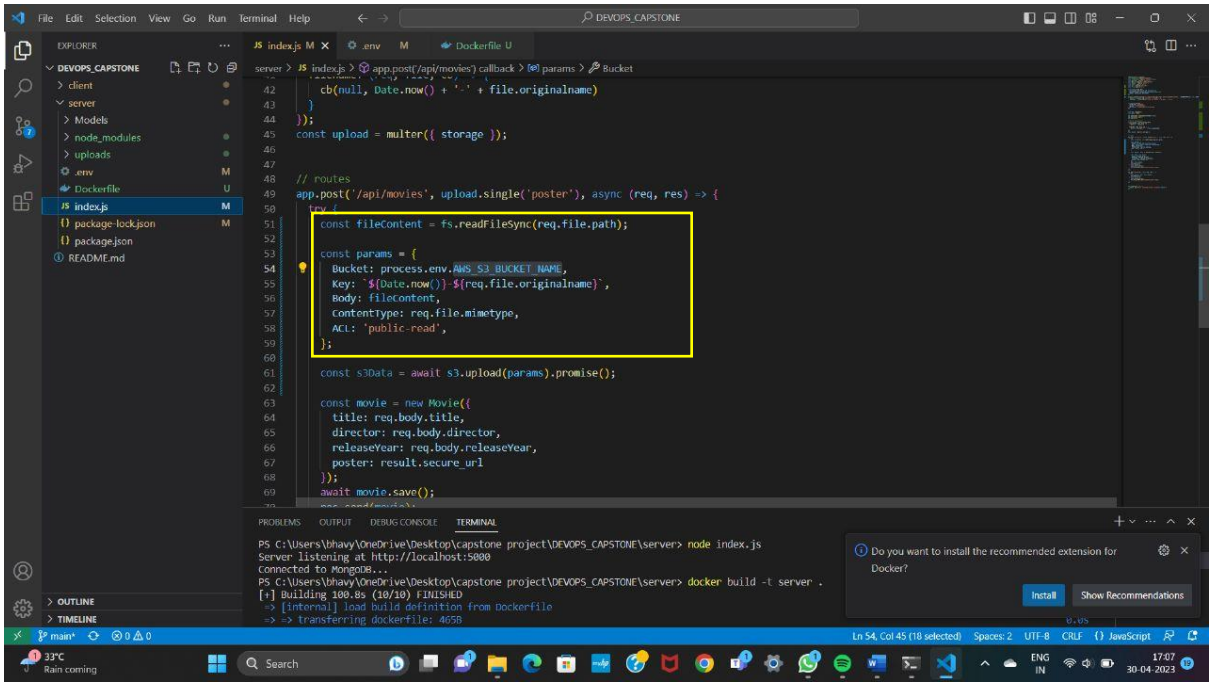


STEP 4: Modifying frontend and backend codes

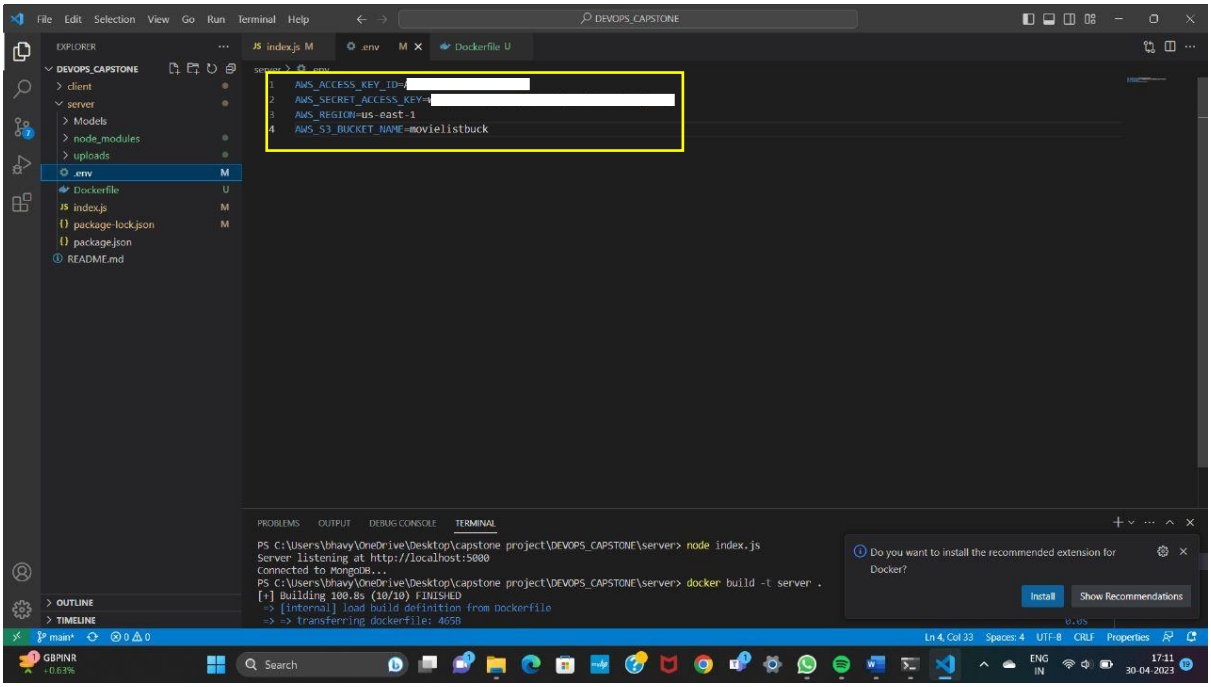
- **Multer S3:** Multer S3 is a middleware for handling file uploads in Node.js applications using Amazon S3 storage. Multer is a popular middleware for handling file uploads in Node.js, and Multer S3 extends its functionality by enabling files to be uploaded directly to an S3 bucket.

- Now make changes in the code for multer configuration.





- Similarly make necessary changes in env. File.
- Add the required details such as Access Key, bucket name, …etc from created IAM User.



STEP 5: Containerization

Docker file for server

Use an official Node.js runtime as a parent image

FROM node:14

Set the working directory to /app

WORKDIR /app

Copy the current directory contents into the container at /app

COPY. /app

Install any needed packages specified in package.json

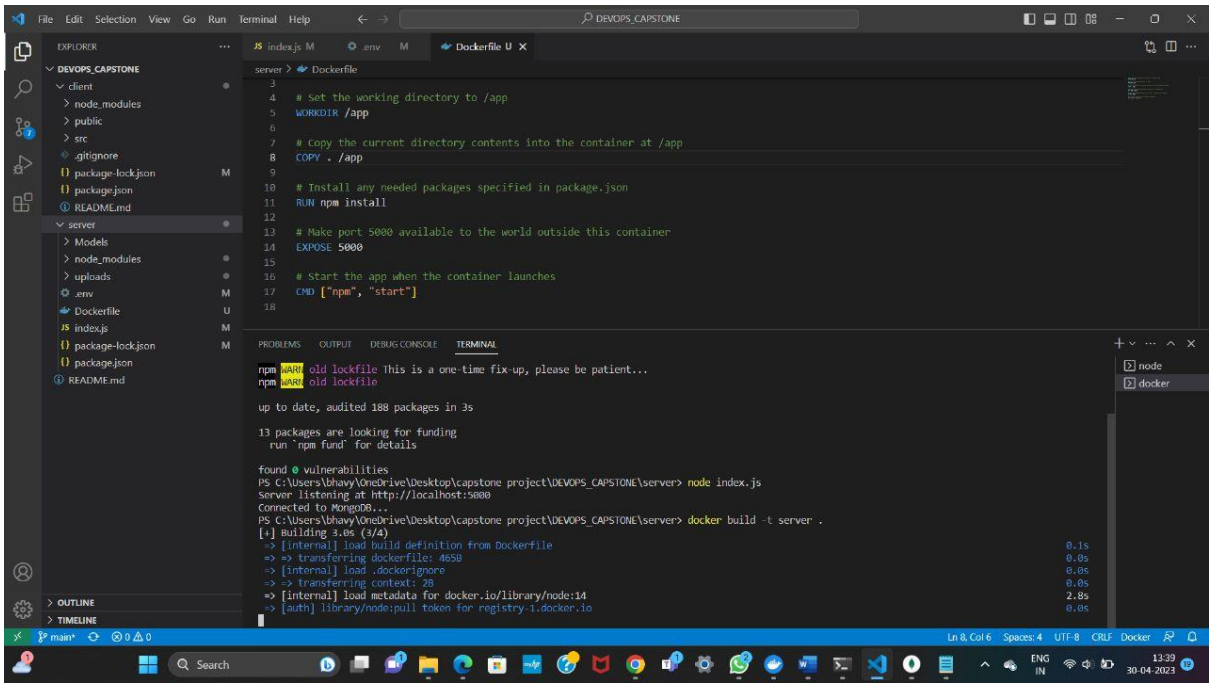
RUN npm install

Make port 5000 available to the world outside this container

EXPOSE 5000

Start the app when the container launches

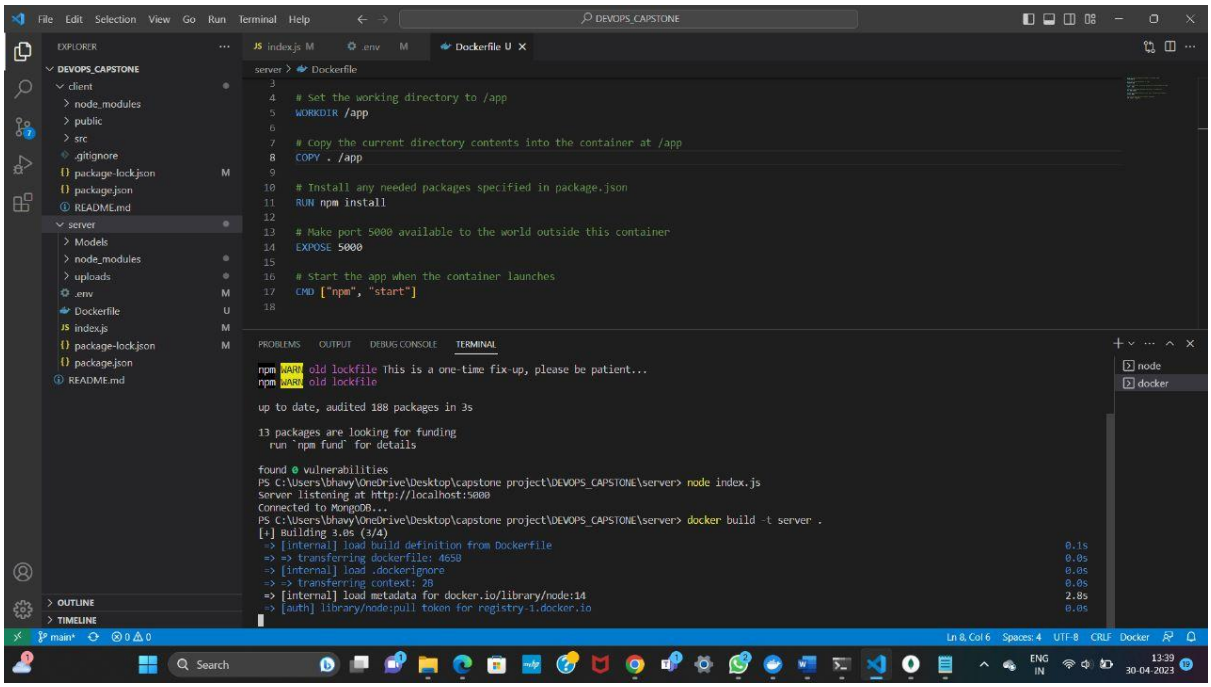
CMD ["npm", "start"]

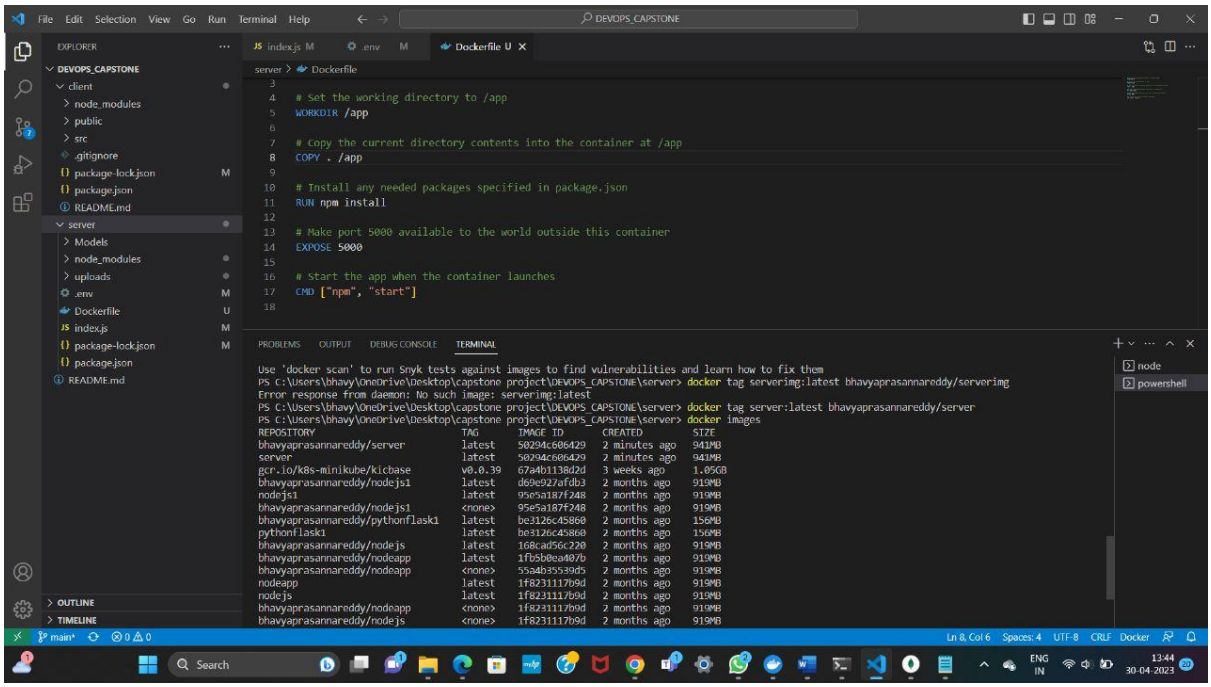


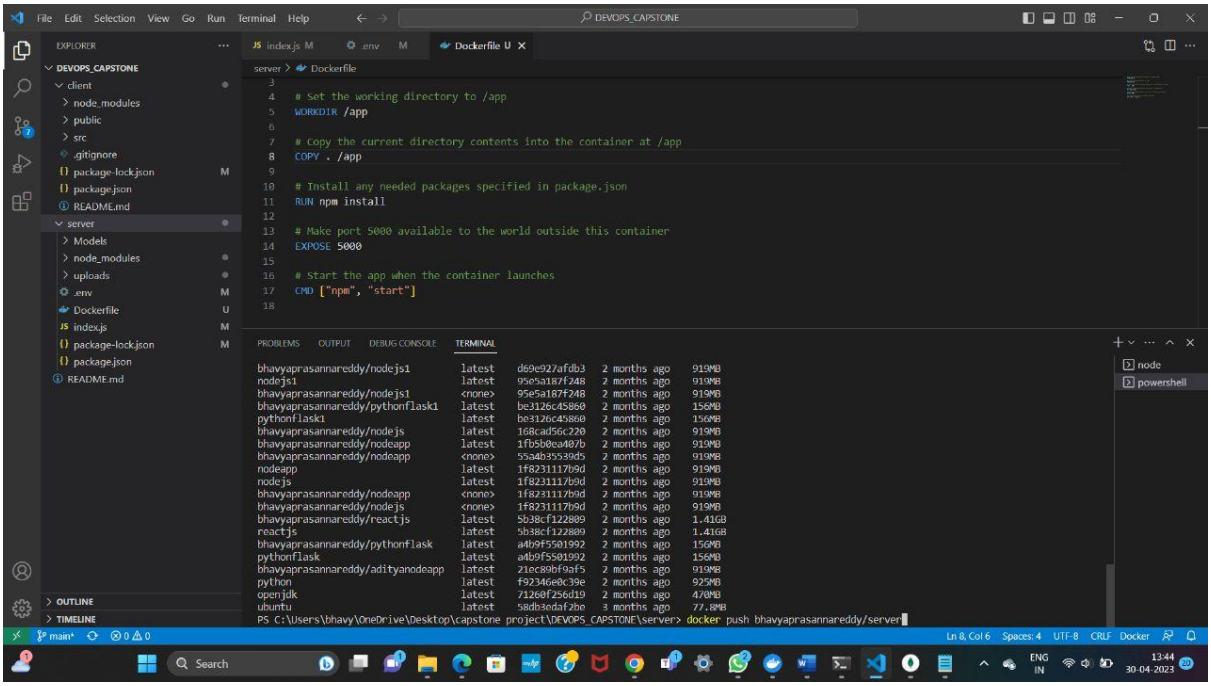
Follow these commands to build the docker image:

- docker build -t server .
- docker tag server:latest bhavyaprasannareddy/server

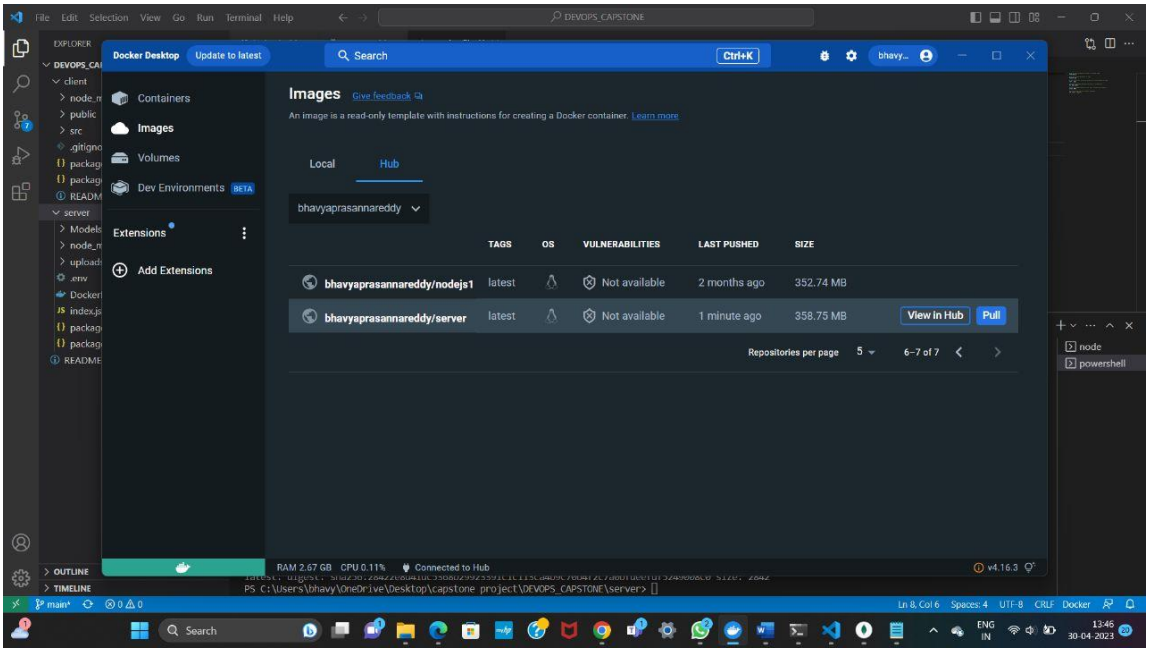
- docker images
- docker push bhavyaprasannareddy/server



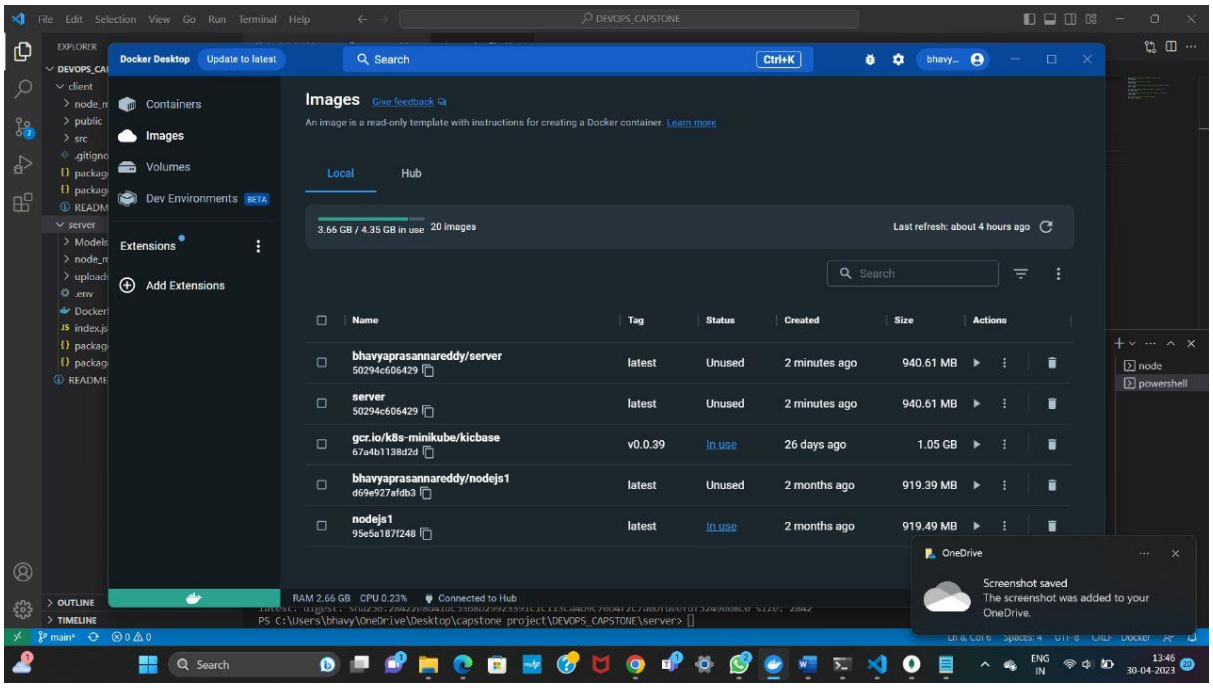


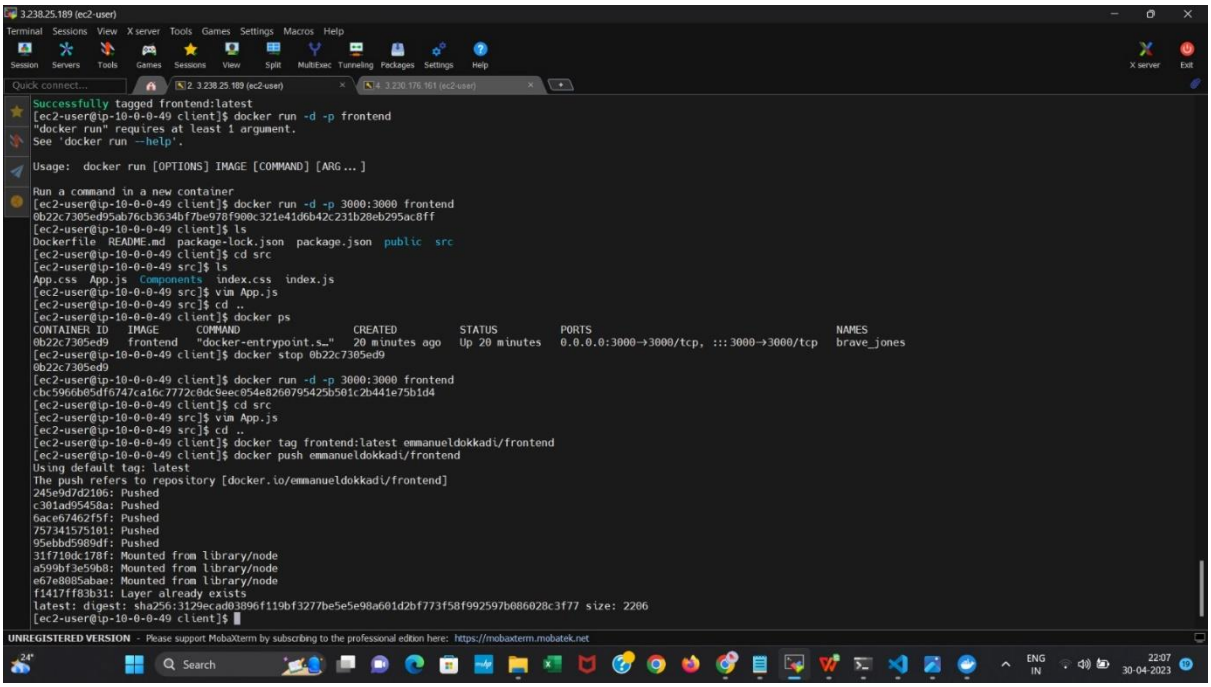


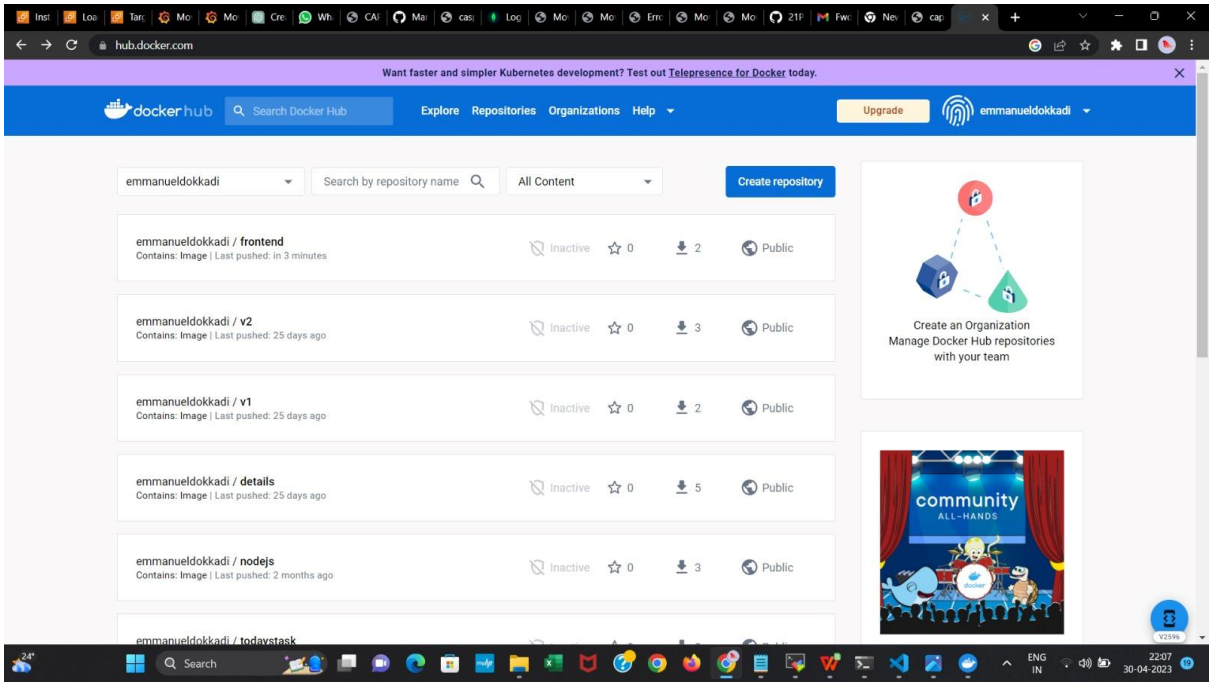
IN DOCKER HUB:



IN LOCAL:


Docker Desktop
Update to latest
Search
Ctrl+K
Containers
Images
Volumes
Dev Environments
BETA
Extensions
Add Extensions
Images
Give feedback
An image is a read-only template with instructions for creating a Docker container. Learn more
Local
Hub
3.66 GB / 4.35 GB in use
20 images
Last refresh: about 4 hours ago
Name
Tag
Status
Created
Size
Actions
bhavyaprasannareddy/server
50294c606429
latest
Unused
2 minutes ago
940.61 MB
server
50294c606429
latest
Unused
2 minutes ago
940.61 MB
gcr.io/k8s-minikube/kicbase
67a4b1138d2d
v0.0.39
In use
26 days ago
1.05 GB
bhavyaprasannareddy/nodejs1
d69e927afdb3
latest
Unused
2 months ago
919.39 MB
nodejs1
95e5a187f248
latest
In use
2 months ago
919.49 MB
RAM 2.66 GB CPU 0.23%
Connected to Hub
PS C:\Users\bhavy\OneDrive\Desktop\capstone project\DEVOPS_CAPSTONE\server>
OneDrive
Screenshot saved
The screenshot was added to your OneDrive.


3.238.25.189 (ec2-user)
Successfully tagged frontend:latest
[ec2-user@ip-10-0-0-49 client]$ docker run -d -p frontend
"docker run" requires at least 1 argument.
See 'docker run --help'.
Usage: docker run [OPTIONS] IMAGE [COMMAND] [ARG ...]
Run a command in a new container
[ec2-user@ip-10-0-0-49 client]$ docker run -d -p 3000:3000 frontend
0b22c7305ed95ab76cb3634bf7be978f900c321e41d6b42c231b28eb295ac8ff
[ec2-user@ip-10-0-0-49 client]$ ls
Dockerfile README.md package-lock.json package.json public src
[ec2-user@ip-10-0-0-49 client]$ cd src
[ec2-user@ip-10-0-0-49 src]$ ls
App.css App.js Components index.css index.js
[ec2-user@ip-10-0-0-49 src]$ vim App.js
[ec2-user@ip-10-0-0-49 src]$ cd ..
[ec2-user@ip-10-0-0-49 client]$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
0b22c7305ed9 frontend "docker-entrypoint.s…" 20 minutes ago Up 20 minutes 0.0.0.0:3000→3000/tcp, :::3000→3000/tcp brave_jones
[ec2-user@ip-10-0-0-49 client]$ docker stop 0b22c7305ed9
0b22c7305ed9
[ec2-user@ip-10-0-0-49 client]$ docker run -d -p 3000:3000 frontend
cbc5966b05df6747ca16c7772c0dc9eec054e8260795425b501c2b441e75b1d4
[ec2-user@ip-10-0-0-49 client]$ cd src
[ec2-user@ip-10-0-0-49 src]$ vim App.js
[ec2-user@ip-10-0-0-49 src]$ cd ..
[ec2-user@ip-10-0-0-49 client]$ docker tag frontend:latest emmanueldokkadi/frontend
[ec2-user@ip-10-0-0-49 client]$ docker push emmanueldokkadi/frontend
Using default tag: latest
The push refers to repository [docker.io/emmanueldokkadi/frontend]
245e9d7d2106: Pushed
c301ad95458a: Pushed
6ace67462f5f: Pushed
757341575101: Pushed
95ebbd5989df: Pushed
31f710dc178f: Mounted from library/node
a599bf3e59b8: Mounted from library/node
e67e8085abae: Mounted from library/node
f1417ff83b31: Layer already exists
latest: digest: sha256:3129ecad03896f119bf3277be5e5e98a601d2bf773f58f992597b086028c3f77 size: 2206
[ec2-user@ip-10-0-0-49 client]$
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net



- Similarly continue the process for client.

  Docker file for client

  # base image

  FROM node:14-alpine

  # set working directory

  WORKDIR /app

  # add `/app/node_modules/.bin` to $PATH

  ENV PATH /app/node_modules/.bin:$PATH

  # install and cache app dependencies

  COPY client/package.json /app/package.json

  RUN npm install --silent

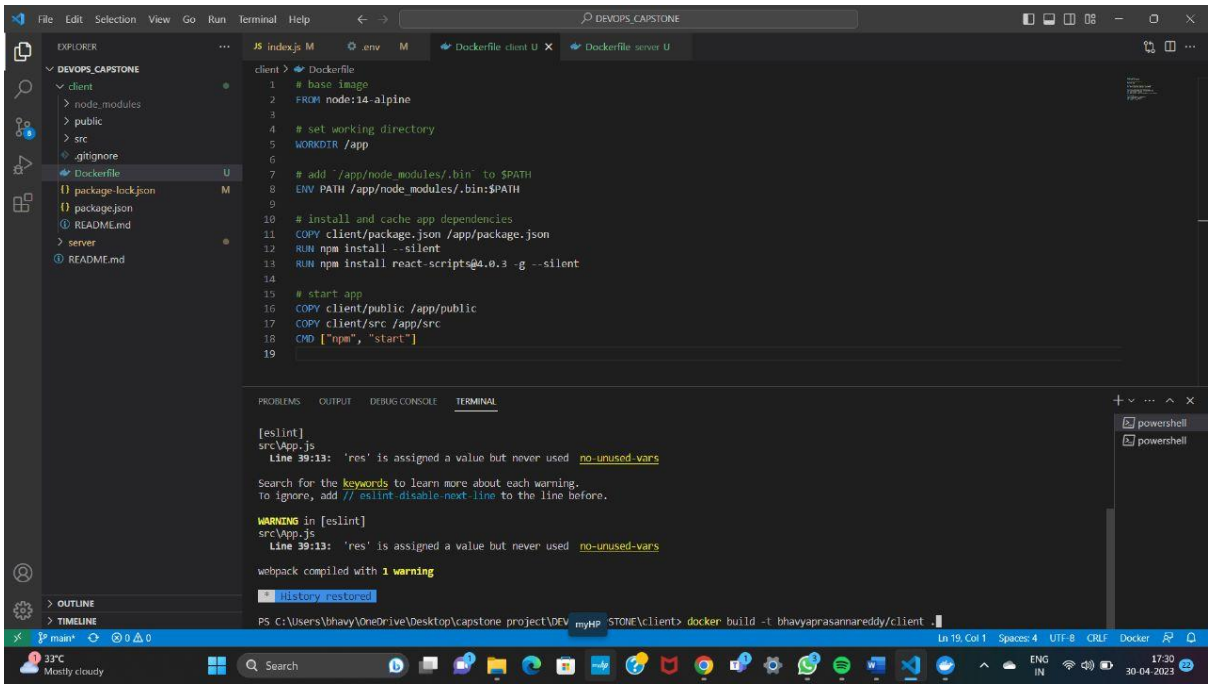
  RUN npm install react-scripts@4.0.3 -g --silent

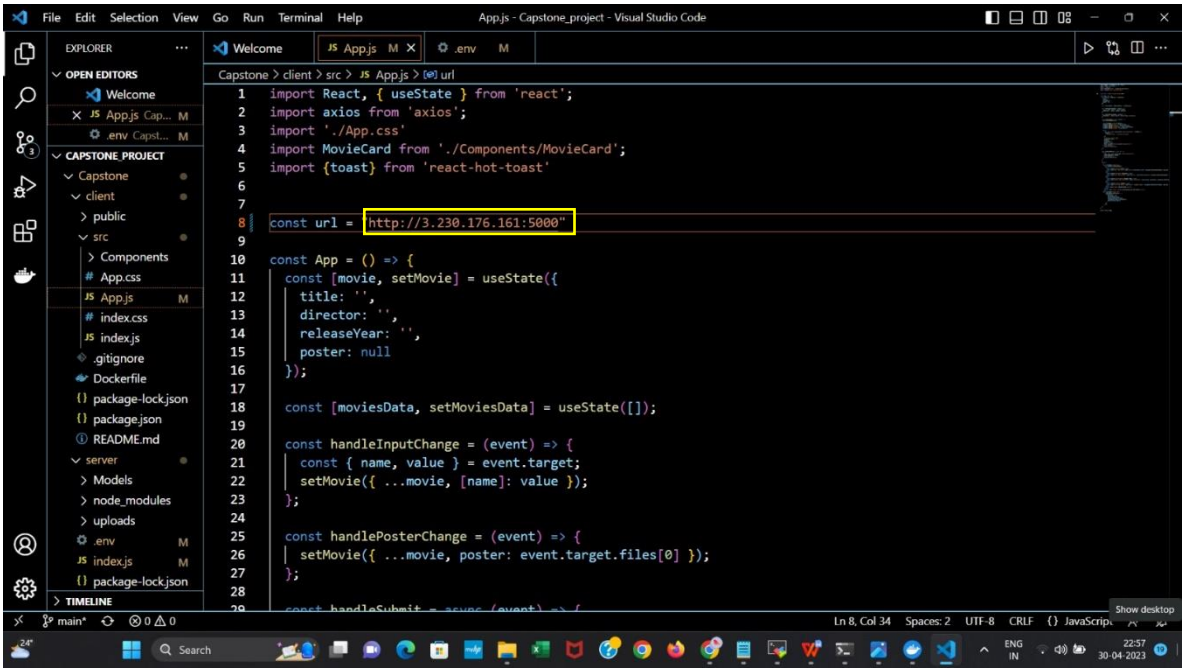
  # start app

  COPY client/public /app/public

  COPY client/src /app/src

  CMD ["npm", "start"]

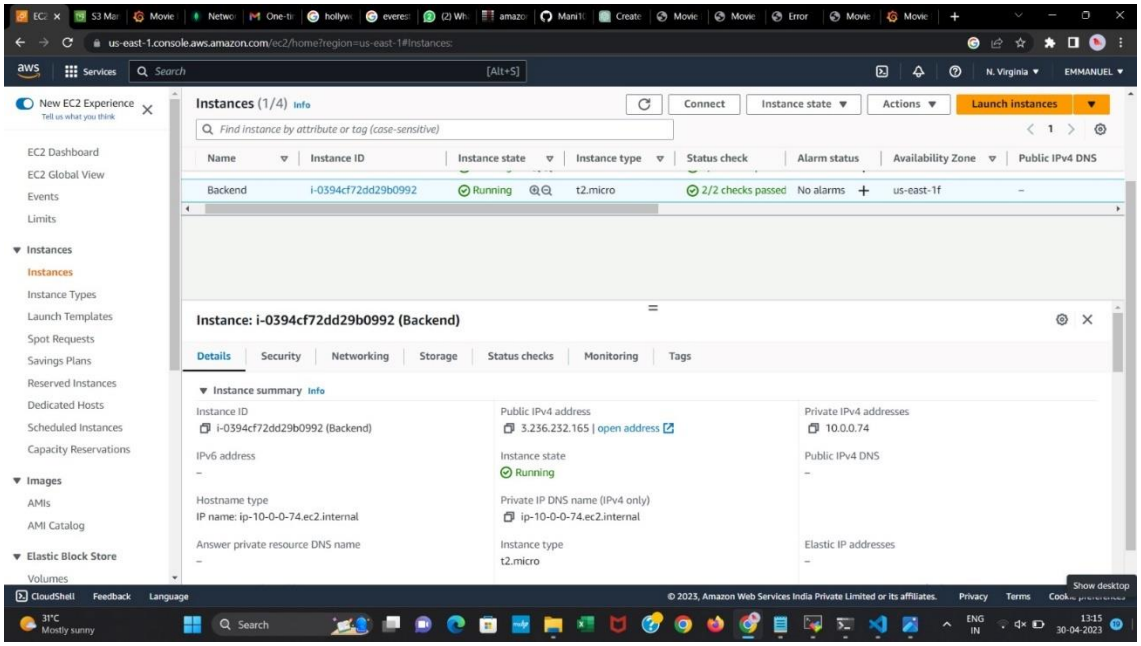


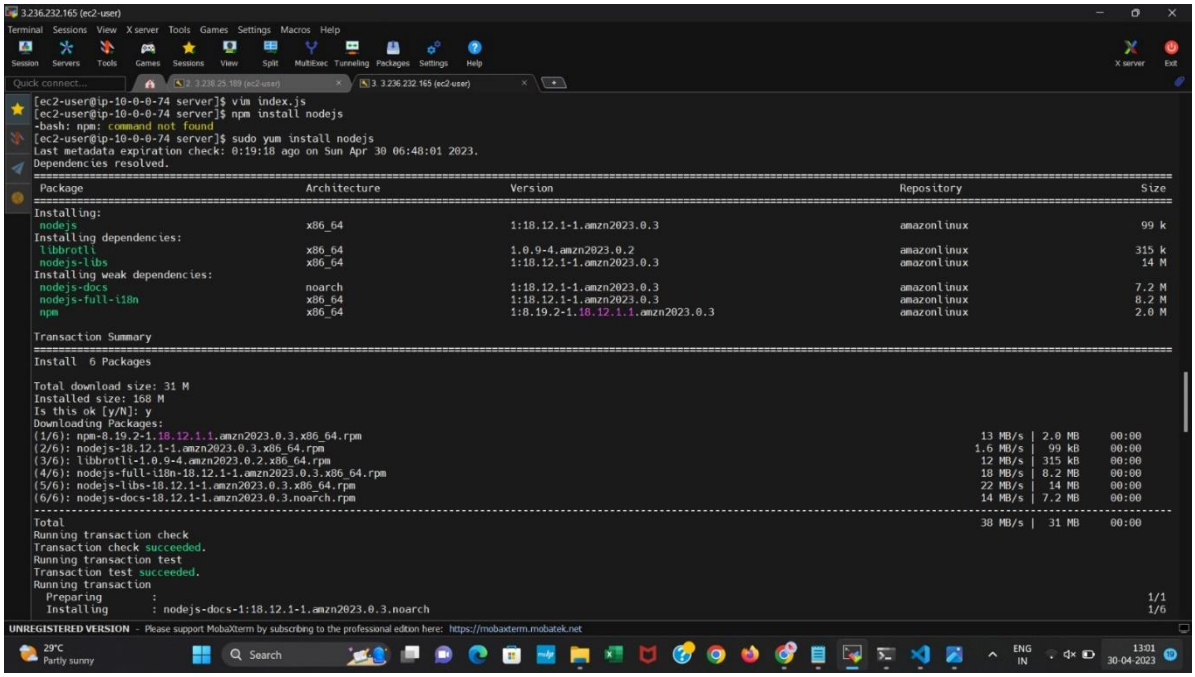


- Replace the URL with instance IP address and port in the file App.js.

STEP 6: Deploying server on EC2 using docker

- Creating EC2 instance to deploy the backend with the help of docker

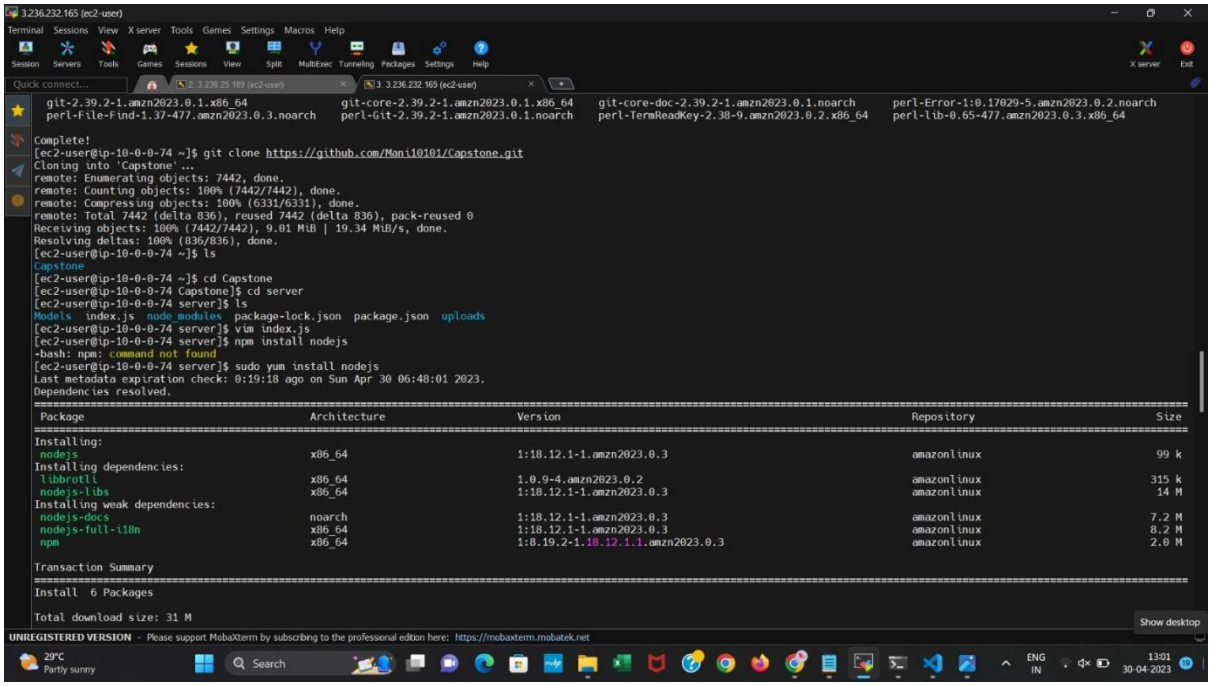


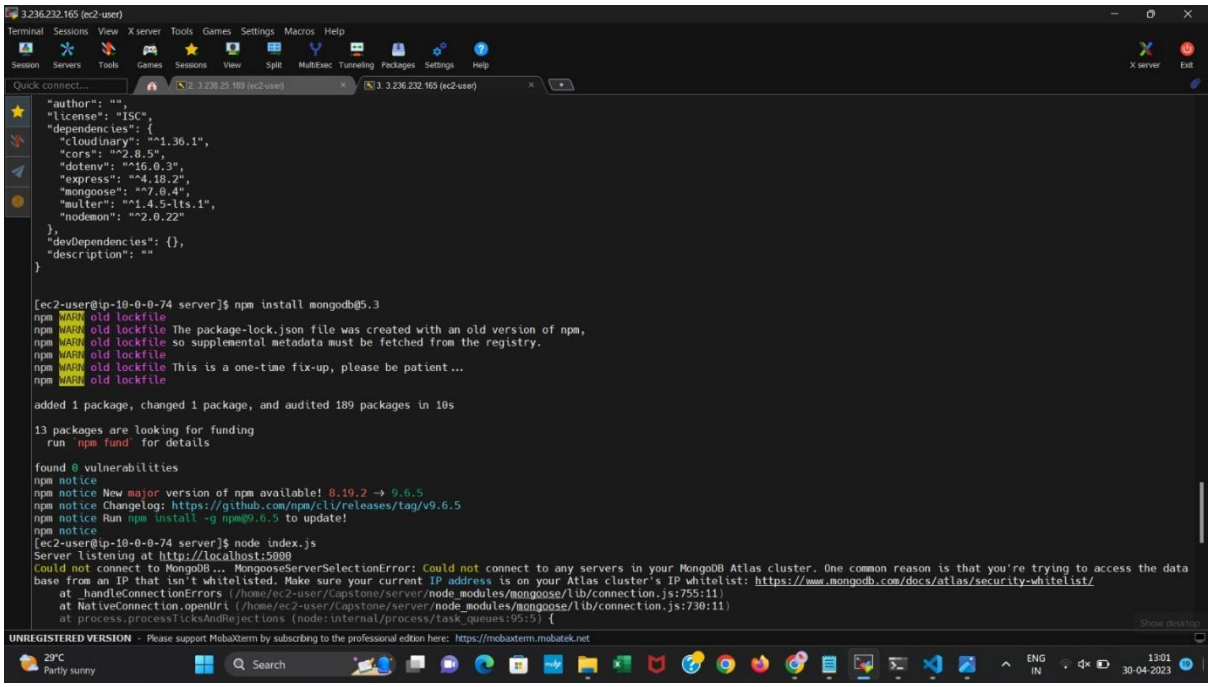


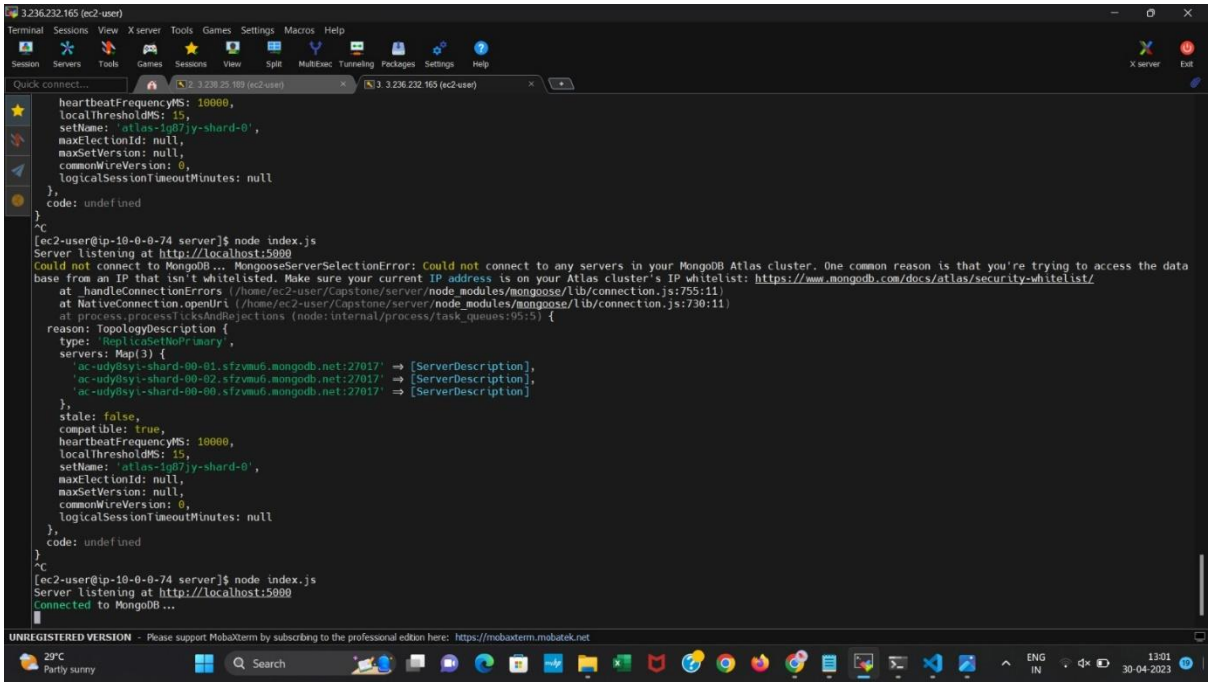
- Connect EC2 instance through local machine using MobaXterm and follow the below commands:

1. Sudo yum update -y
2. Sudo yum Install docker -y
3. Sudo service docker start

4. Sudo usermod -a -G docker ec2-user
5. Sudo chmod 777 /var/run/docker.sock
6. Sudo yum install nodejs
7. Npm init -y
8. Npm Install mongodb@5.3
9. Node index.js

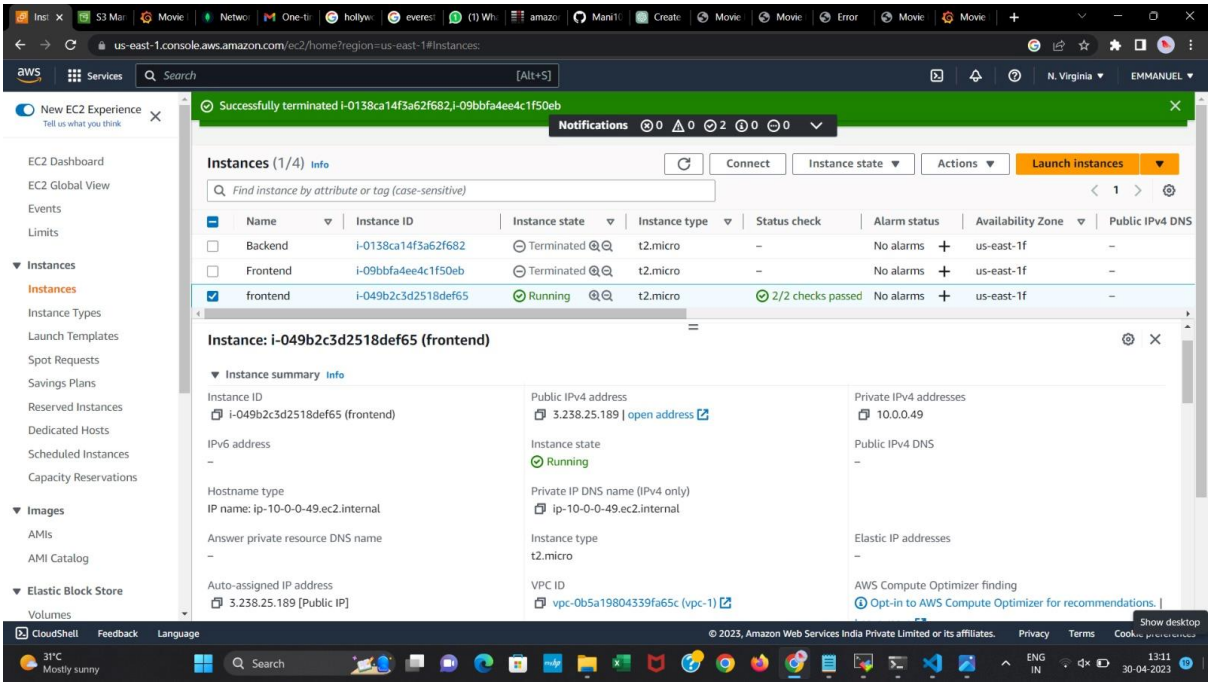






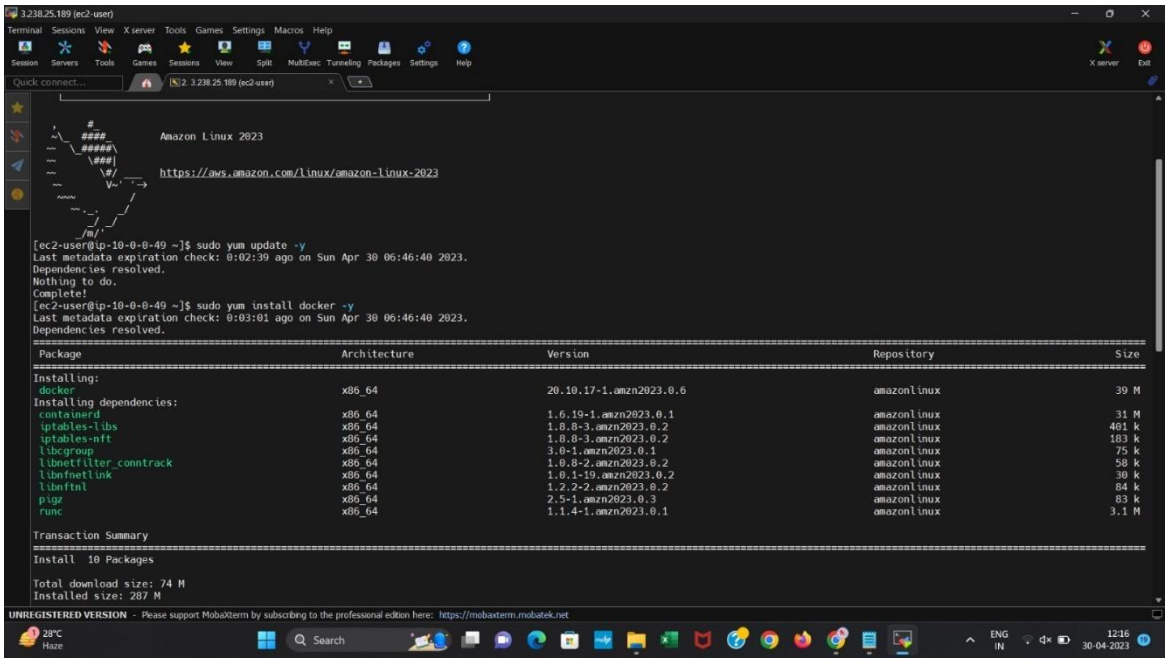
STEP 7: Deploying client on EC2 using docker

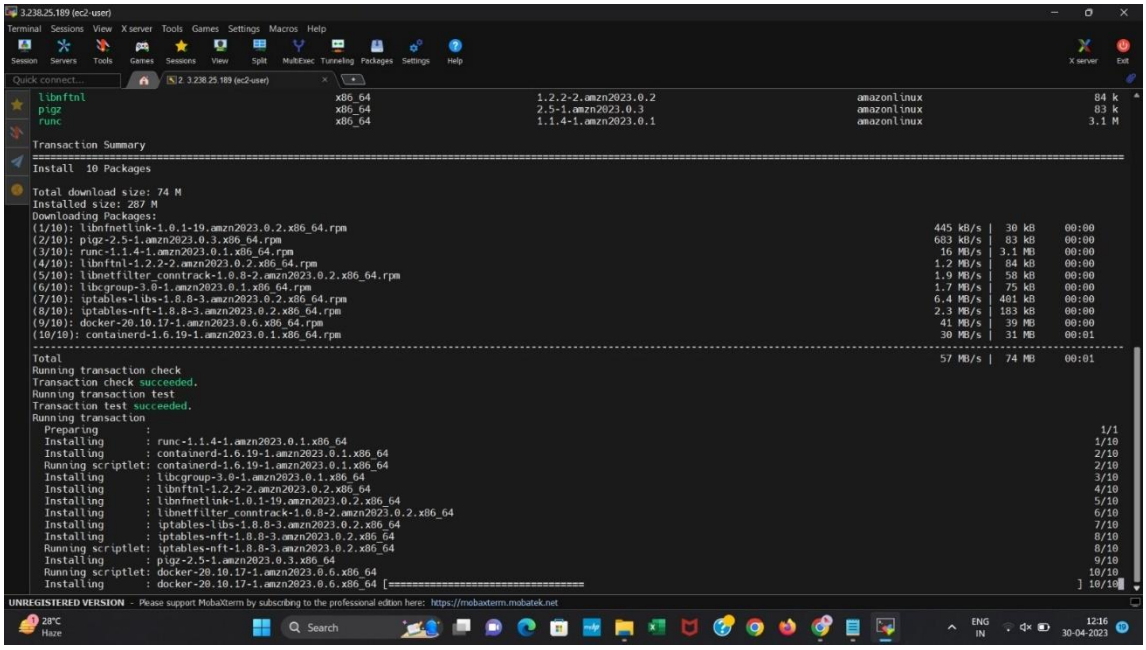
- Creating EC2 instance to deploy the frontend with the help of docker.

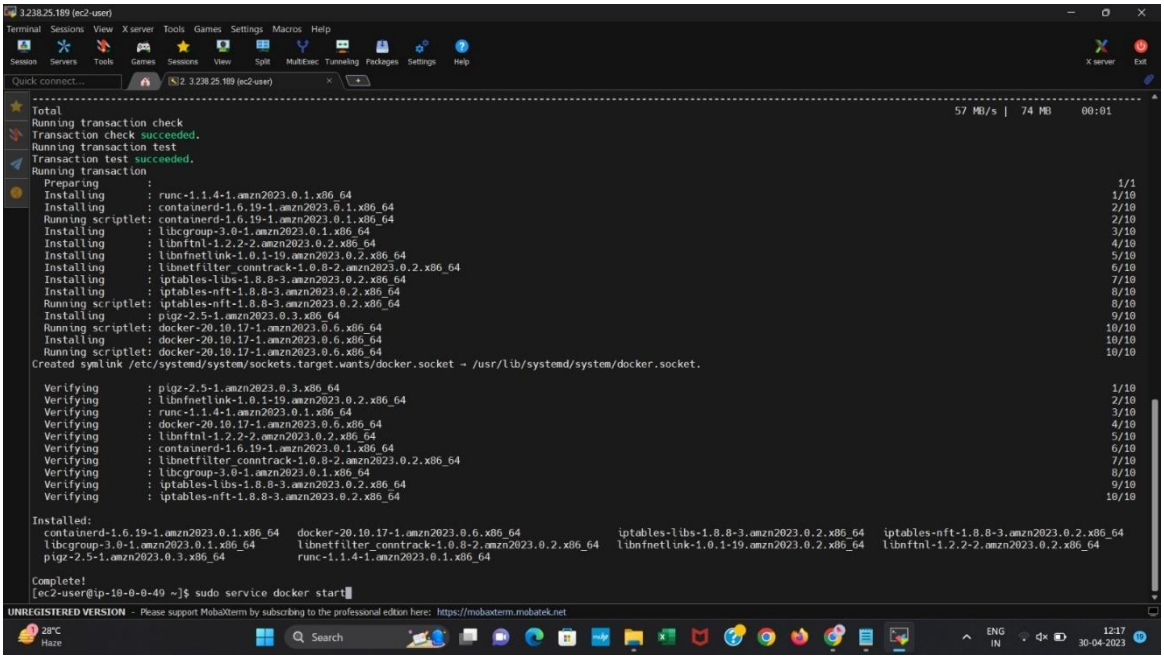


- Follow the commands as shown:
    1. Sudo yum update -y
    2. Sudo yum Install docker -y
    3. Sudo service docker start

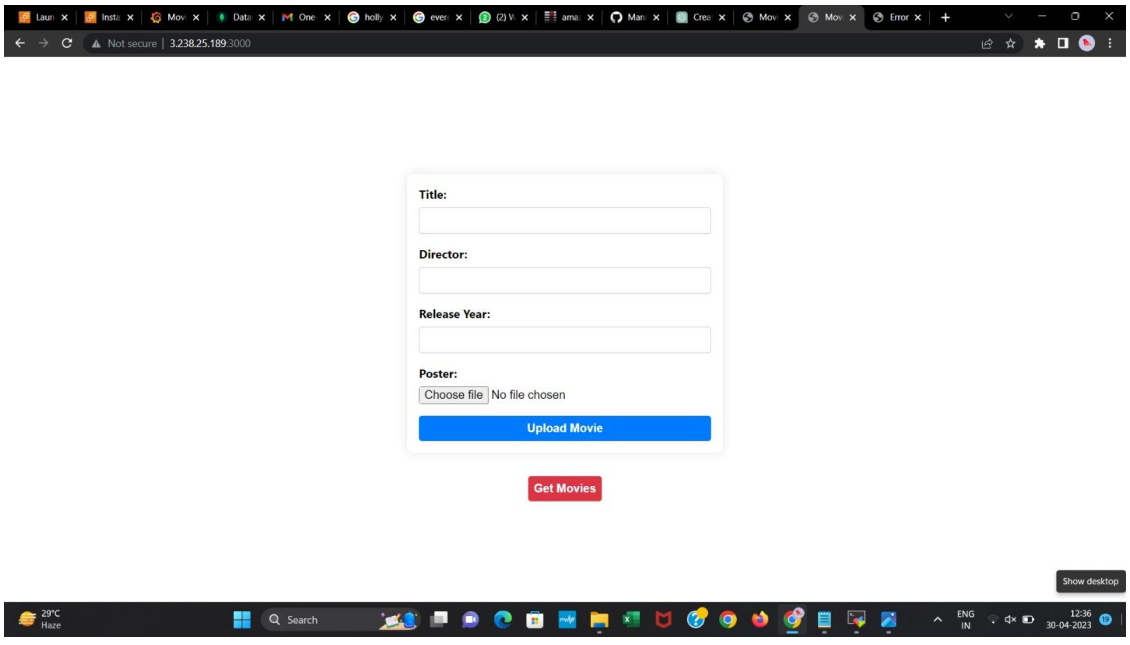
4. Sudo usermod -a -G docker ec2-user
5. Sudo chmod 777 /var/run/docker.sock
6. Sudo yum install git -y
7. Git clone url
8. Docker build -t frontend .
9. Docker run -d -p 3000:3000 frontend



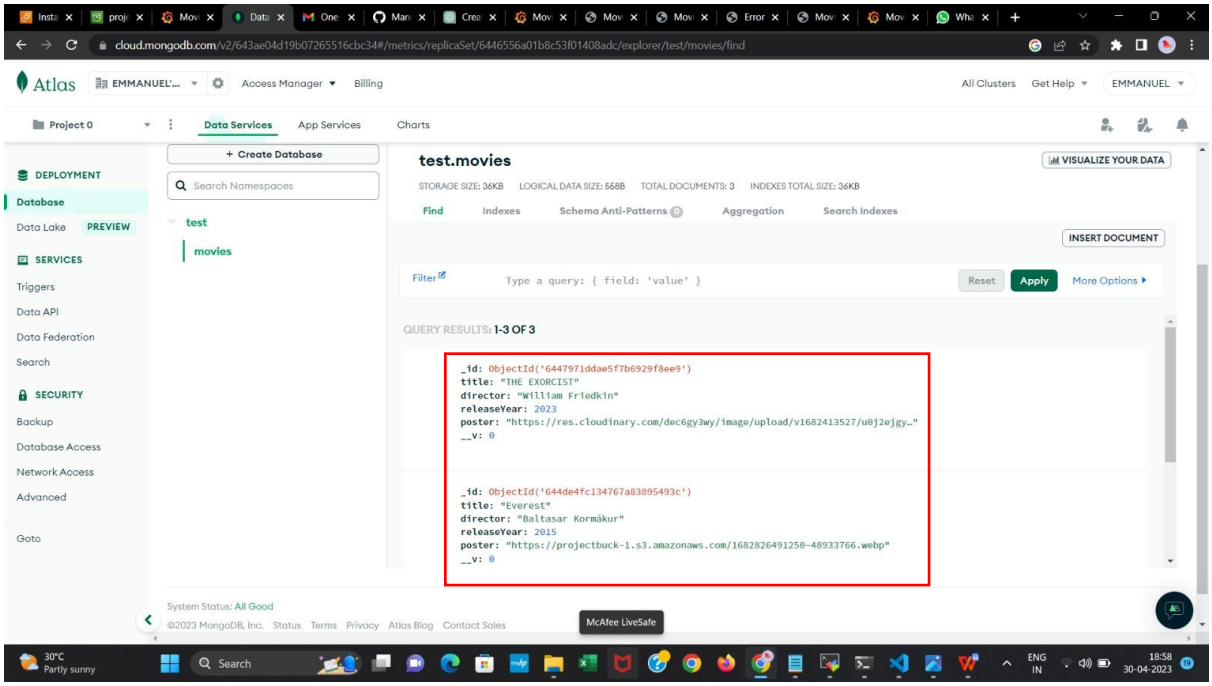


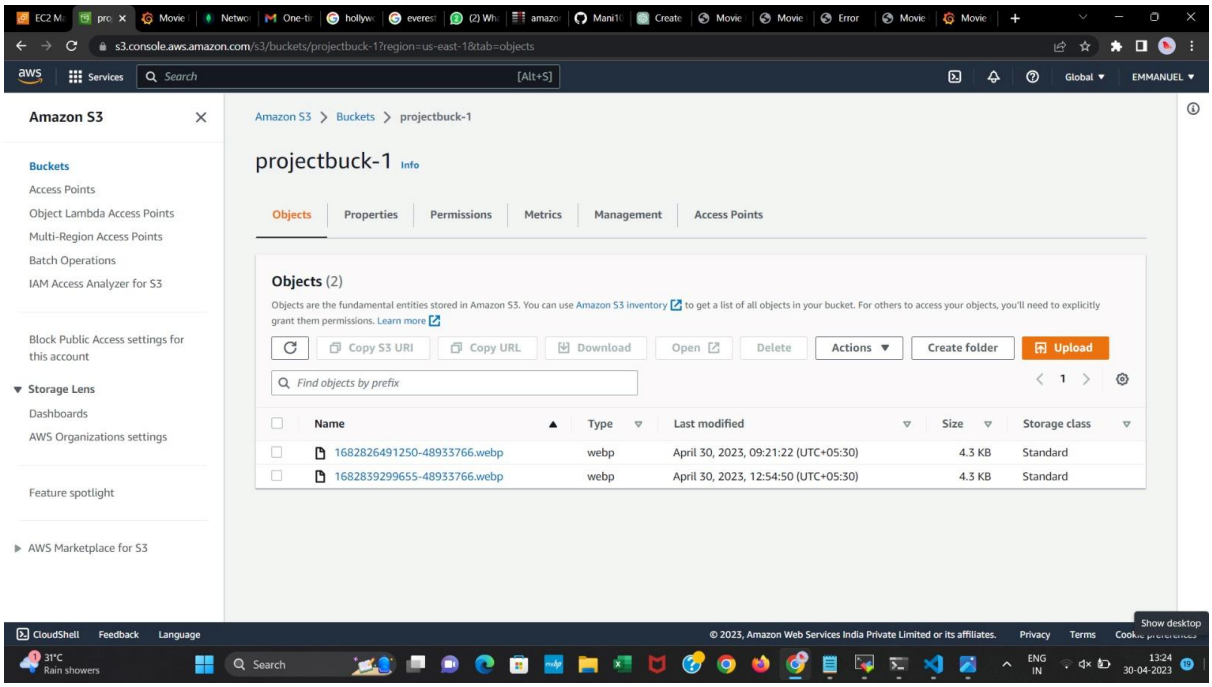


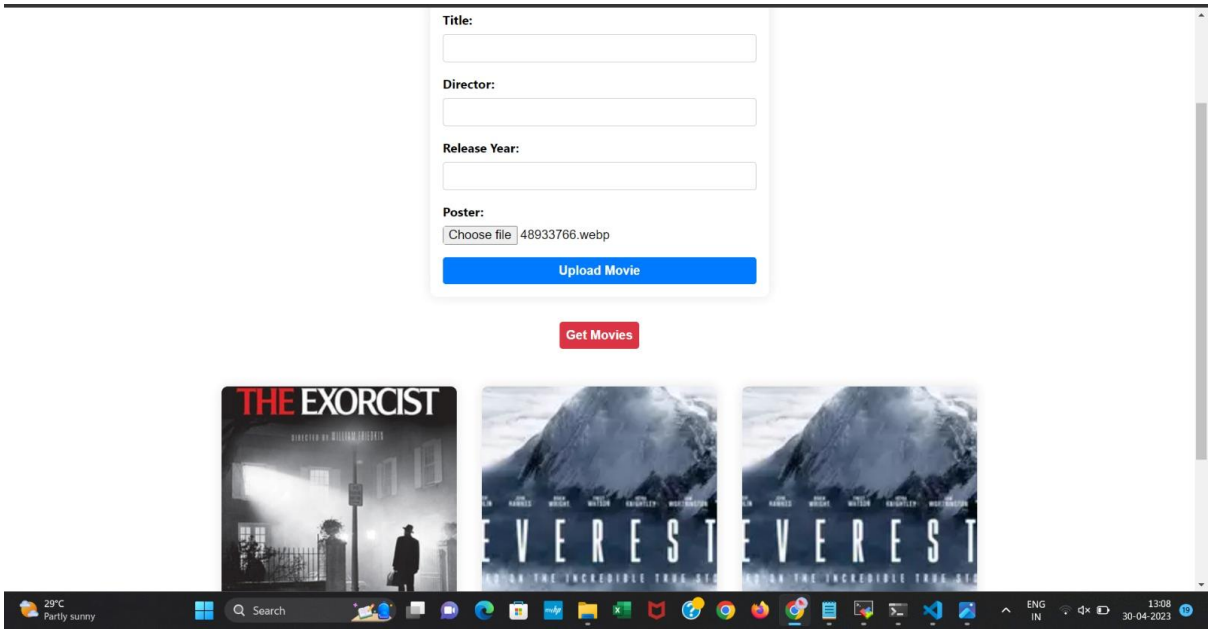
- After deploying to check whether the website we the public IP address of instance and port number.



- The web page has been deployed, then enter the details of the given parameters and then your data is been stored in database and your images are stored in S3 bucket.


cloud.mongodb.com/v2/643ae04d19b07265516cbc34#/metrics/replicaSet/6446556a01b8c53f01408adc/explorer/test/movies/find
Atlas
Access Manager
Billing
All Clusters
Get Help
EMMANUEL
Project 0
Data Services
App Services
Charts
+ Create Database
Search Namespaces
test
movies
DEPLOYMENT
Database
Data Lake
PREVIEW
SERVICES
Triggers
Data API
Data Federation
Search
SECURITY
Backup
Database Access
Network Access
Advanced
Goto
test.movies
VISUALIZE YOUR DATA
STORAGE SIZE: 36KB
LOGICAL DATA SIZE: 568B
TOTAL DOCUMENTS: 3
INDEXES TOTAL SIZE: 36KB
Find
Indexes
Schema Anti-Patterns
Aggregation
Search Indexes
INSERT DOCUMENT
Filter
Type a query: { field: 'value' }
Reset
Apply
More Options
QUERY RESULTS: 1-3 OF 3
_id: ObjectId('6447971ddae5f7b6929f8ee9')
title: "THE EXORCIST"
director: "William Friedkin"
releaseYear: 2023
poster: "https://res.cloudinary.com/dec6gy3wy/image/upload/v1682413527/u8j2ejgy…"
__v: 0
_id: ObjectId('644de4fc134767a83895493c')
title: "Everest"
director: "Baltasar Kormákur"
releaseYear: 2015
poster: "https://projectbuck-1.s3.amazonaws.com/1682826491250-48933766.webp"
__v: 0
System Status: All Good
©2023 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales
McAfee LiveSafe
30°C
Partly sunny
Search
ENG
IN
18:58
30-04-2023


s3.console.aws.amazon.com/s3/buckets/projectbuck-1?region=us-east-1&tab=objects
aws
Services
Search
[Alt+S]
Global
EMMANUEL
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
Amazon S3 > Buckets > projectbuck-1
projectbuck-1 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
Name
Type
Last modified
Size
Storage class
1682826491250-48933766.webp
webp
April 30, 2023, 09:21:22 (UTC+05:30)
4.3 KB
Standard
1682839299655-48933766.webp
webp
April 30, 2023, 12:54:50 (UTC+05:30)
4.3 KB
Standard
CloudShell
Feedback
Language
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Show desktop
31°C
Rain showers
Search
ENG
IN
13:24
30-04-2023

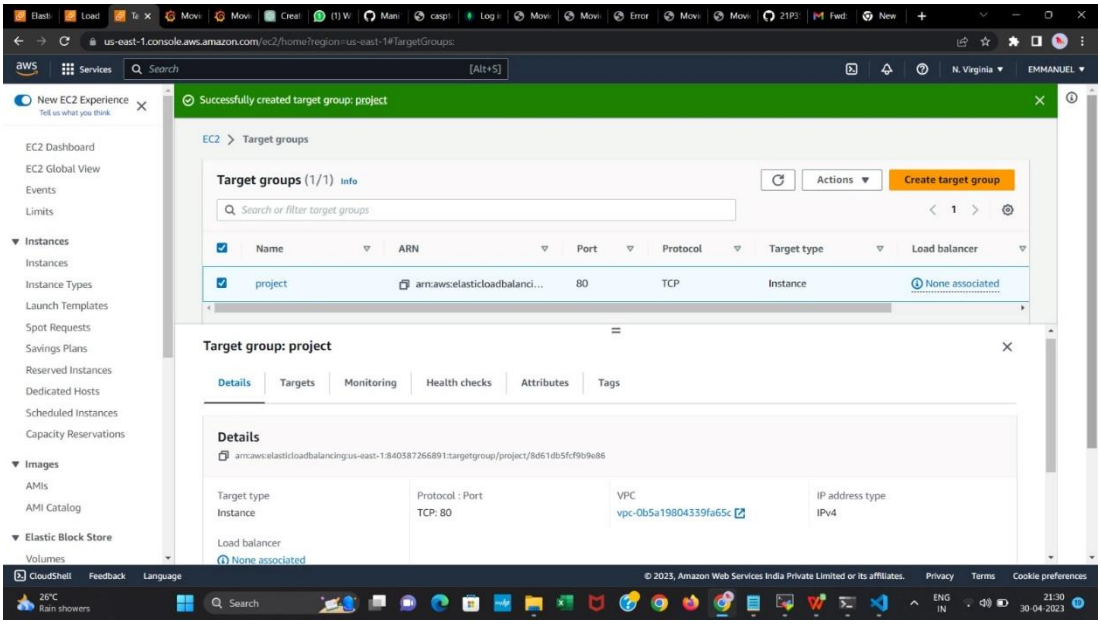


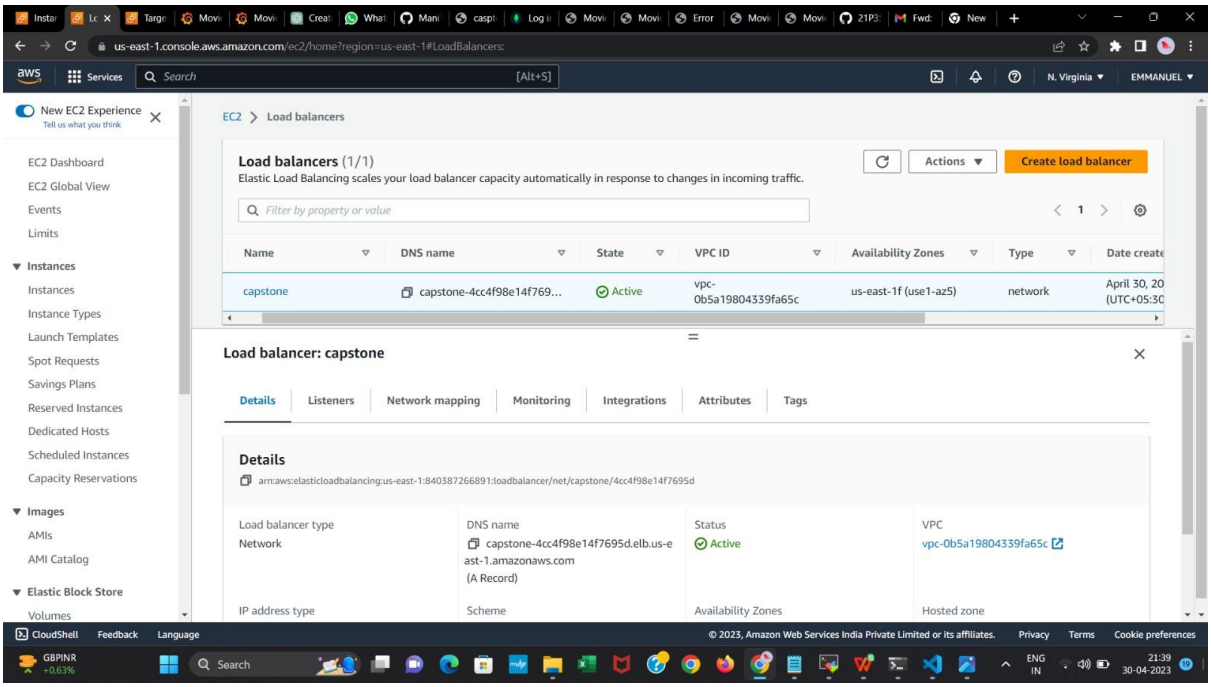
STEP8: Creation of target groups and load balancer

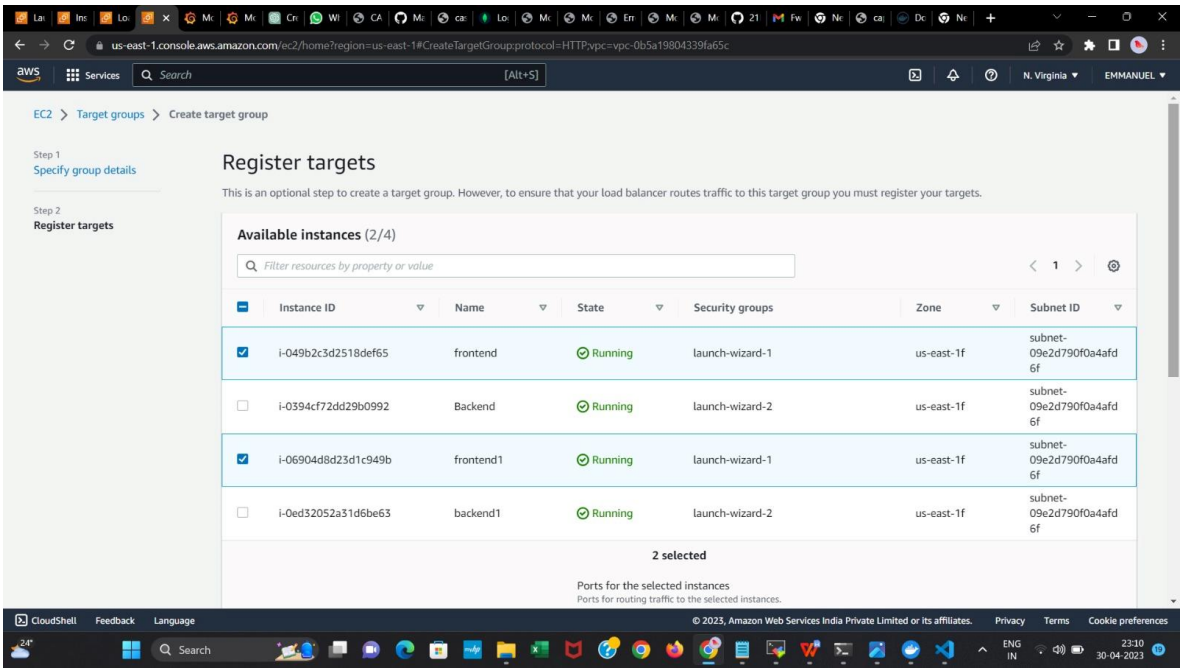
Load balancer: Load balancing is the process of distributing network traffic across multiple servers to ensure that no single server is overloaded with too much traffic. This helps to improve the overall performance, availability, and scalability of the application or service.

Target groups: Target groups are a feature of load balancers in Amazon Web Services (AWS) that enable traffic to be directed to a set of instances that are registered with the target group. A target group can contain one or more instances, and multiple target groups can be associated with a single load balancer.

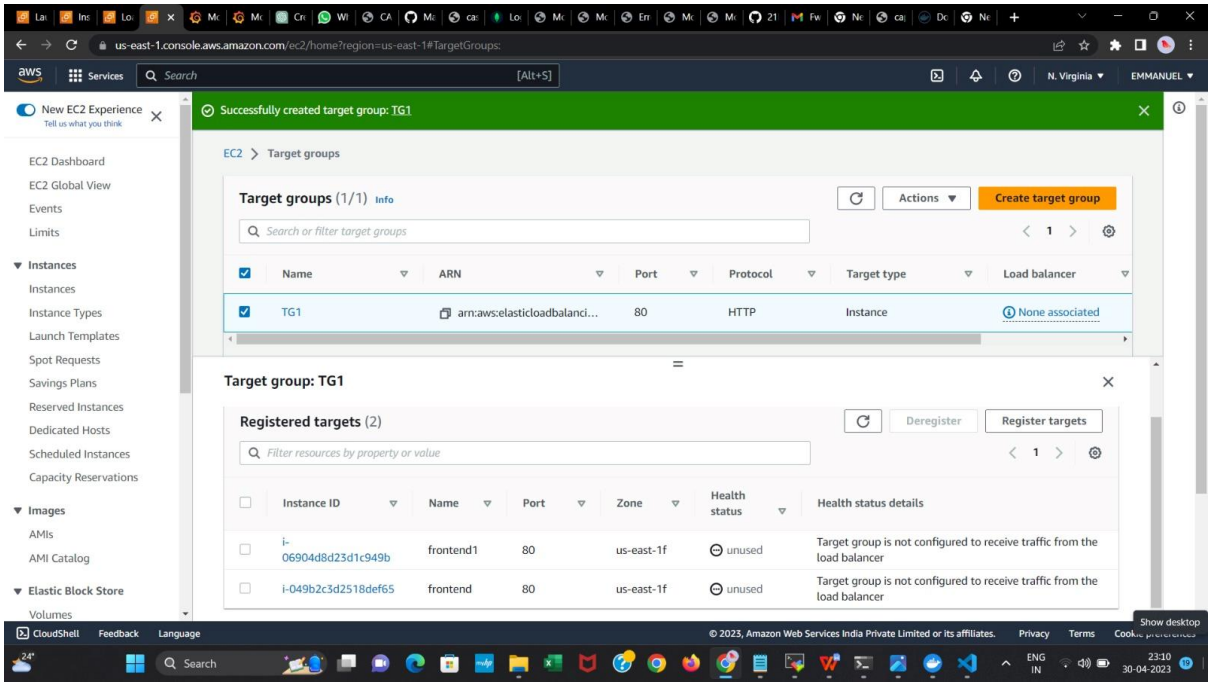
FRONTEND:

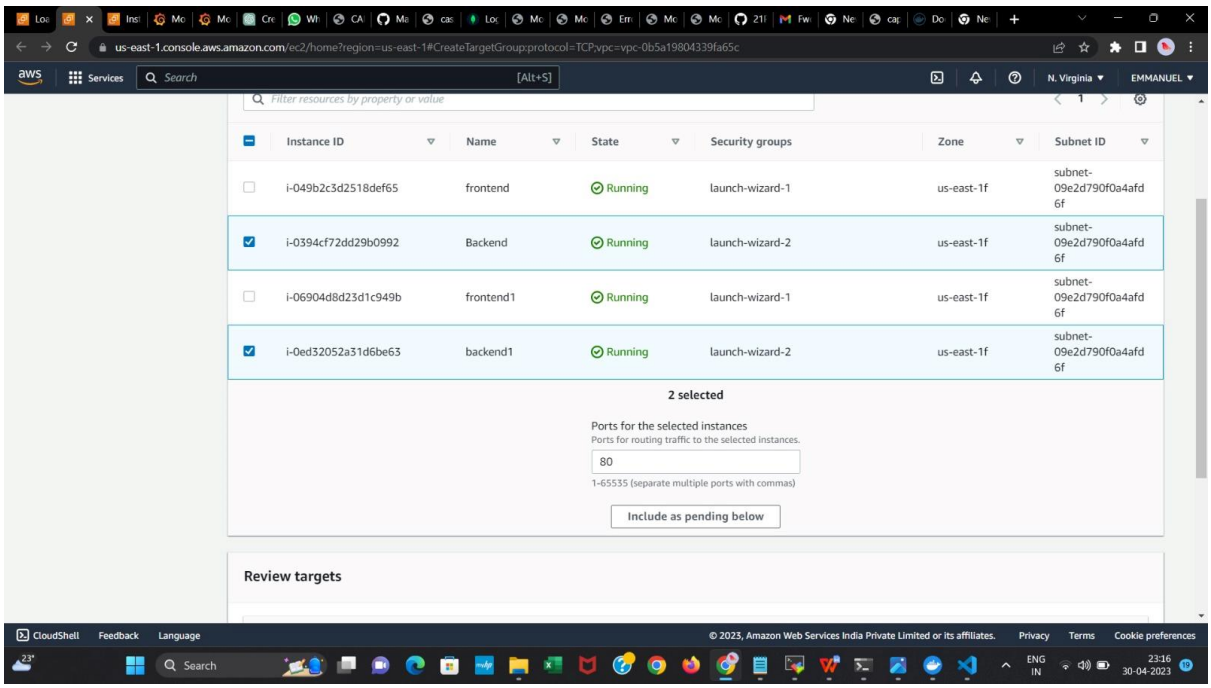



us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LoadBalancers:
EC2 > Load balancers
Load balancers (1/1)
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.
Actions
Create load balancer
Name
DNS name
State
VPC ID
Availability Zones
Type
capstone
capstone-4cc4f98e14f769...
Active
vpc-0b5a19804339fa65c
us-east-1f (use1-az5)
network
Load balancer: capstone
Details
Listeners
Network mapping
Monitoring
Integrations
Attributes
Tags
Load balancer type
Network
DNS name
capstone-4cc4f98e14f7695d.elb.us-east-1.amazonaws.com
(A Record)
Status
Active
VPC
vpc-0b5a19804339fa65c
IP address type
Scheme
Availability Zones
Hosted zone


us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#CreateTargetGroup:protocol=HTTP;vpc=vpc-0b5a19804339fa65c
EC2 > Target groups > Create target group
Step 1
Specify group details
Step 2
Register targets
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 (2/4)
Instance ID
Name
State
Security groups
Zone
Subnet ID
i-049b2c3d2518def65
frontend
Running
launch-wizard-1
us-east-1f
i-0394cf72dd29b0992
Backend
Running
launch-wizard-2
us-east-1f
i-06904d8d23d1c949b
frontend1
Running
launch-wizard-1
us-east-1f
i-0ed32052a31d6be63
backend1
Running
launch-wizard-2
us-east-1f
2 selected
Ports for the selected instances
Ports for routing traffic to the selected instances.

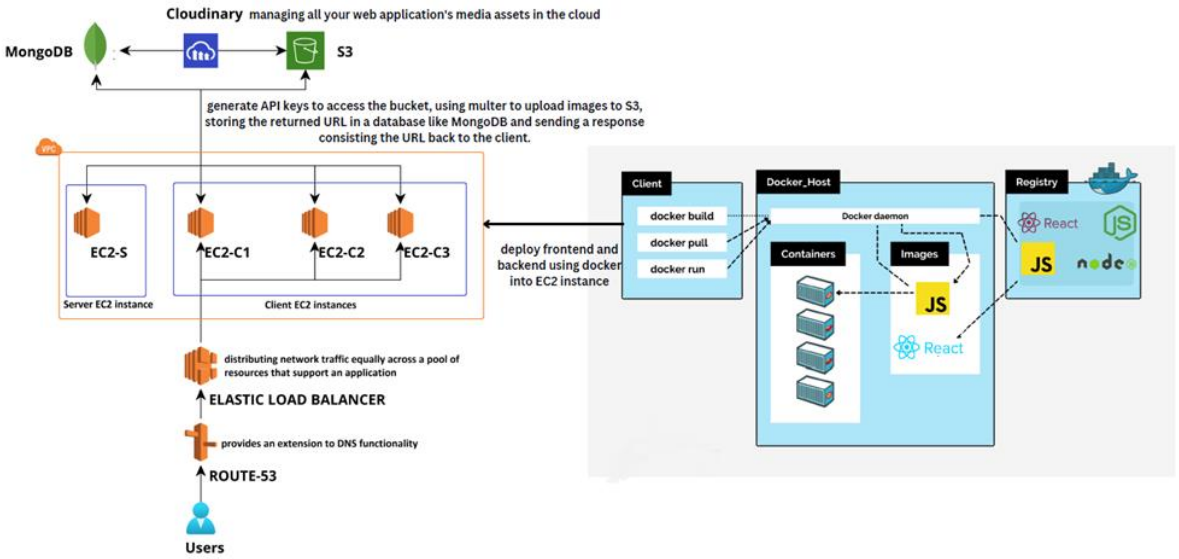
BACKEND:





- **DNS (Domain Name System):** is a system used to translate human-readable domain names into IP addresses that computers can understand.
  LINK: http://Casptone-ac1c7a45280f6a36.elb.us-east1.amazonaws.com
  GIT LINK: https://github.com/21P35A0347/Capstone_project.git

AWS DEPLOYMENT ARCHETECHTURE



METHODS TO IMPROVE AWS DEPLOYMENT

- Use Infrastructure as Code (IaC): Use IaC tools like AWS CloudFormation or AWS Terraform to define and deploy infrastructure. This allows you to define your infrastructure in code and manage it like any other software application.
- Automate deployment: Use continuous integration and continuous deployment (CI/CD) pipelines to automate your deployment process. This ensures that code changes are automatically deployed to production after passing a series of automated tests.
- Implement scalability: Design your infrastructure to scale horizontally or vertically based on usage patterns. This can be achieved by using AWS Auto Scaling groups, which automatically adjust the number of instances running based on demand.
- Use AWS best practices: Follow AWS best practices to ensure your infrastructure is secure, highly available, and cost-effective. This includes using AWS services like AWS Identity and Access Management (IAM) for access control, Amazon Elastic Compute Cloud (EC2) for compute resources, and Amazon Relational Database Service (RDS) for databases.
- Monitor performance: Use AWS CloudWatch to monitor the performance of your infrastructure and applications. This allows you to

identify and troubleshoot performance issues before they impact your users.

- Use AWS Managed Services: Consider using AWS Managed Services like Amazon Elastic Kubernetes Service (EKS), AWS Lambda, or Amazon Aurora to reduce the operational overhead of managing infrastructure.
- Ensure Disaster Recovery: Implement a disaster recovery plan that includes regular backups and replication of your infrastructure and data to another AWS region or third-party provider.

By implementing these methods, you can improve the reliability, scalability, security, and performance of your AWS deployment.

Connecting the server with atlas Mongo DB	EMMANUEL, NEELIMA
Creating IAM user and S3	BHAVYA, PRIYANKA
Configuration the Application Code with S3-multer	VIJAYA, HARI PRIYANKA
Configuration of the code using Docker file	NITHYA SRI, FREINEY
Deploying server on EC2 using Docker	BHAVYA, PRIYANKA
Creation a target group and Load Balancer	EMMANUEL, NEELIMA
Creating AWS Deployment Diagram	VIJAYA, HARI PRIYANKA, NITHYA SRI, FREINEY
Preparing Project Documentation	VIJAYA, HARI PRIYANKA, NITHYA SRI, FREINEY, EMMANUEL, NEELIMA, BHAVYA, PRIYANKA.

CONCLUSION

In conclusion, deploying a movie listing website to the cloud infrastructure using AWS can provide numerous benefits such as scalability, reliability, and availability. The project involved using ReactJS as the frontend, NodeJS as the backend, and MongoDB as the database, with Multer being used for file uploads.

The project scope included deploying the application to the cloud using AWS services like EC2, S3, IAM, and Route53. Additionally, load balancing was implemented using target groups to distribute traffic across instances.

Overall, this project demonstrates how deploying a web application to the cloud infrastructure can provide numerous benefits and is an essential skill for modern software development.