Gold Screen cinema

Project Introduction

Gold Screen Cinema is a full-stack cinema website built using the MERN (MongoDB, Express.js, React.js, Node.js) stack. Website is responsive to multiple screen sizes. Deployed on an AWS Virtual Machine, it features a microservices-based architecture where all core services run in separate Docker containers, enhancing maintainability and resource efficiency.

The CI/CD pipeline, powered by GitHub Actions, automates testing, building, and deployment, ensuring consistent quality and rapid feature delivery. Each deployment seamlessly integrates development and production updates for a live system.

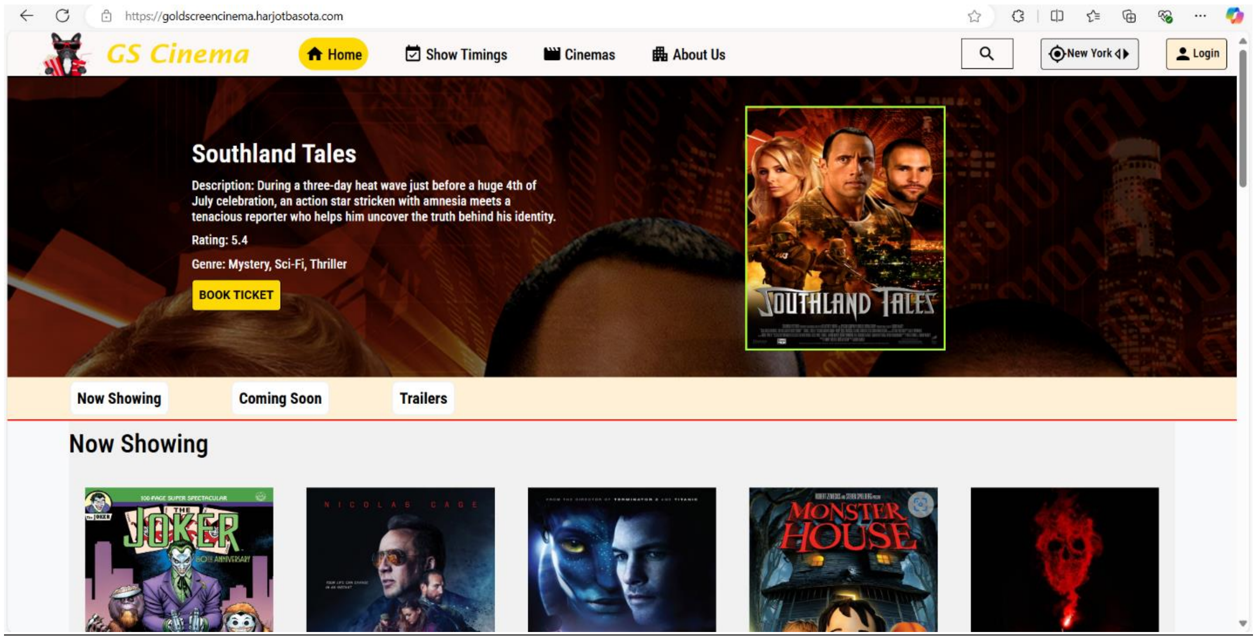
Live: https://goldscreencinema.harjotbasota.com

Repo: https://www.github.com/harjotbasota/goldScreenCinema

Tech Stack

- React.js
- Node.js
- Express.js
- MongoDB
- APIs
- JWT Authentication
- AWS EC2
- AWS Route 53
- AWS Cloudwatch
- Docker
- Certbot
- GitHub actions
- GitHub Secrets
- Shell Scripts
- Shell Scripting

Website



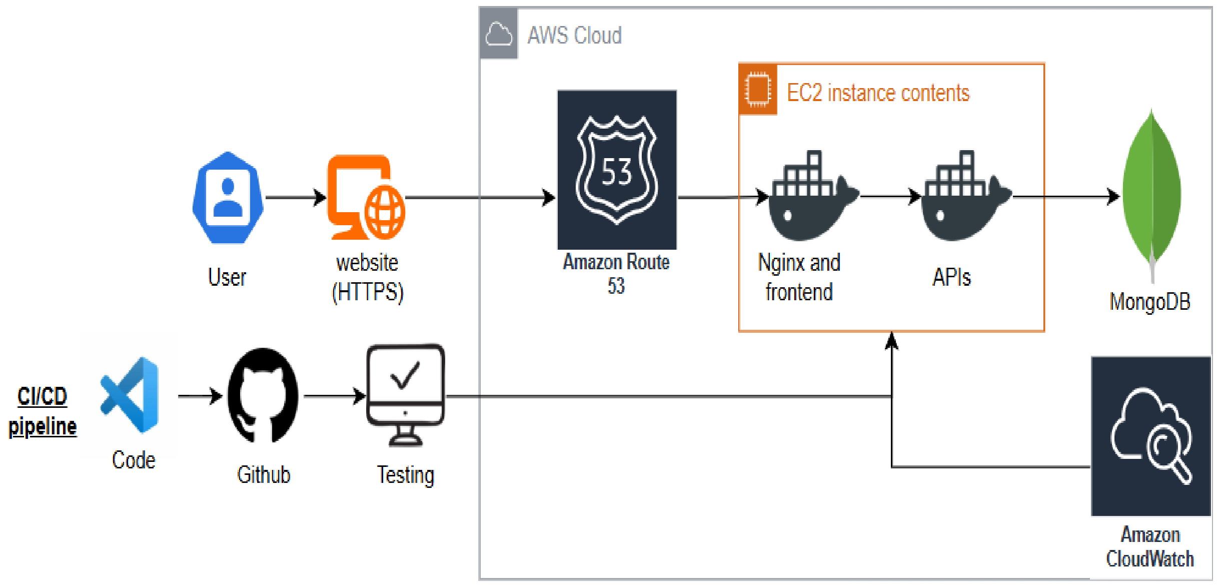
Southland Tales

Description: During a three-day heat wave just before a huge 4th of July celebration, an action star stricken with amnesia meets a tenacious reporter who helps him uncover the truth behind his identity.

Rating: 5.4

Genre: Mystery, Sci-Fi, Thriller

Architecture



Application Description - Frontend

The frontend of Gold Screen Cinema is a dynamic and user-focused platform built using **React**. It leverages modern tools like **Material-UI**, **Google Icons**, and **Context API** to provide a seamless and visually appealing experience. Styled with **CSS**, the website is fully responsive, ensuring accessibility and functionality across various devices.

Key Features

1. **JWT Authentication**:
   - User sessions are managed using JWTs, with access tokens stored securely in memory and refresh tokens in cookies.
   - Logged-in users gain access to exclusive features such as ticket booking and a personalized profile page.
2. **Movie Browsing and Booking**:
   - The **Home Page** showcases all available movies with a clean and intuitive design.
   - The **Shows Page** provides detailed information, including movies, cinema locations, and showtimes.
   - Users can search for movies effortlessly and book tickets through an interactive visual seat selection interface.
   - Booked tickets are stored and displayed in the **Profile Page** for easy access.
3. **Dynamic and Interactive UI**:
   - The front page features a rotating **movie banner**, highlighting featured films every few seconds.
   - Custom error messages enhance user experience by providing clear feedback during issues like login failures or ticket availability errors.
4. **REST API Integration**:
   - The frontend communicates with the backend via REST APIs for all core functionalities, ensuring smooth data flow and reliable performance.

User Experience

The frontend is designed with a user-centric approach, incorporating responsive layouts and engaging elements. The combination of Material-UI components and CSS styling ensures a modern and visually consistent interface.

Application Description - APIs

The backend of Gold Screen Cinema is built using **Node.js** with the **Express** framework, providing robust functionality for authentication and ticket management. This service is designed to handle user registration, login, ticket booking, and viewing of booked tickets securely and efficiently.

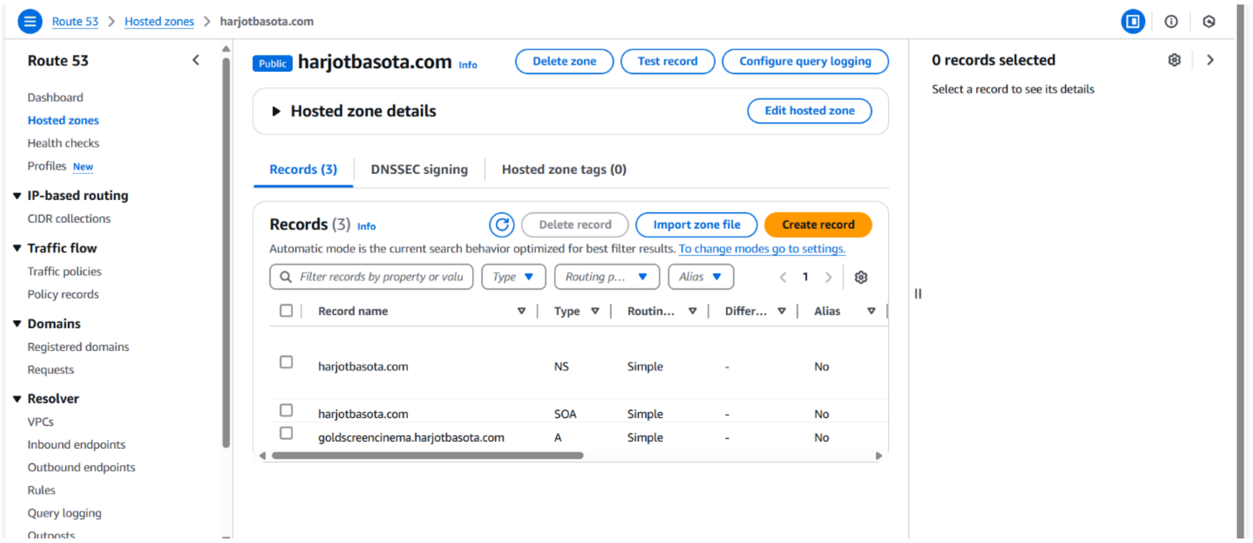
Key Features

1. **Authentication**:
    - **JWT Authentication** is used for managing user sessions. On login, a token is generated and stored, while each request to protected routes requires token validation using **bcrypt**.
    - The backend ensures that only authenticated users can access private routes such as profile information and ticket booking.
    - All API requests are made over **HTTPS** to ensure secure data transmission.
2. **MongoDB Integration**:
    - The backend utilizes **MongoDB** for storing user data and ticket management.
    - **Two collections** are used: one for storing user authentication information (credentials, token) and another for managing ticket bookings.
3. **API Endpoints**:
    - **/auth**: Handles **signup**, **login**, and **logout** operations.
    - **/user**: Provides access to user profile data.
    - **/show**: Allows users to view tickets they have already booked.
4. **Testing**:
    - Automated tests are written using **Jest** to ensure the integrity and functionality of the backend, covering endpoints and business logic.

This backend setup supports a clean and organized structure for handling core functionalities, providing a solid foundation for the Gold Screen Cinema platform.

Application Deployment

1. Setup Route 53 to point domain to VM IP



Route 53 > Hosted zones > harjotbasota.com

Public harjotbasota.com

Records (3)

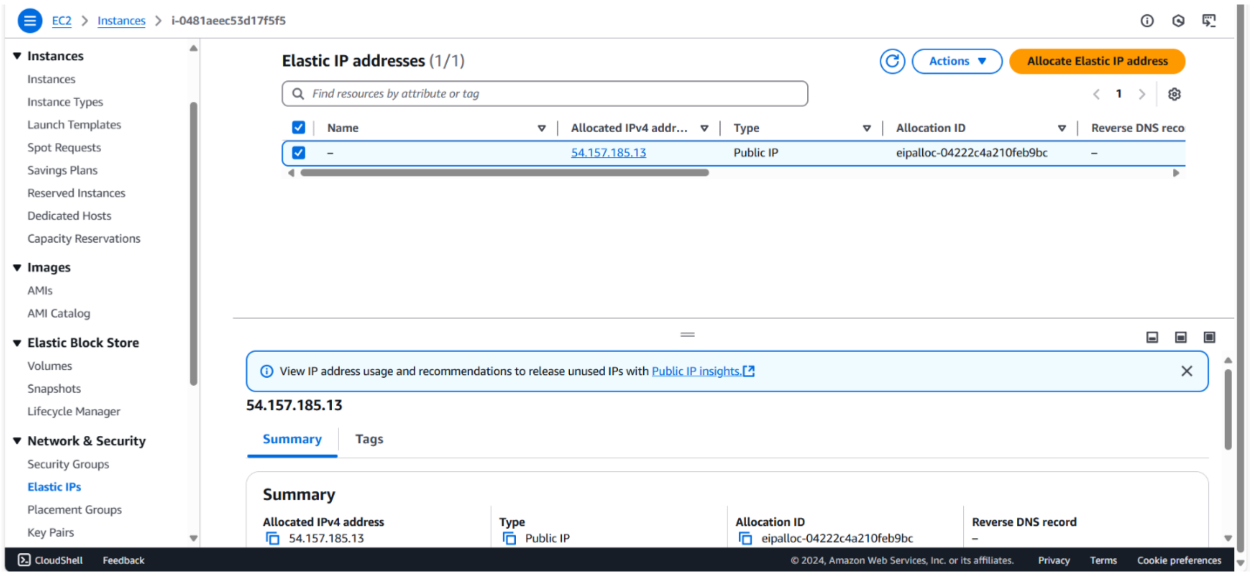
Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings.

Record name	Type	Routin...	Differ...	Alias
harjotbasota.com	NS	Simple	-	No
harjotbasota.com	SOA	Simple	-	No
goldscreencinema.harjotbasota.com	A	Simple	-	No

0 records selected

Select a record to see its details

2. Create a VM and attach the IP for your domain to VM (port 22, 443, 80 should be open)



EC2 > Instances > i-0481aeec53d17f5f5

Elastic IP addresses (1/1)

Name	Allocated IPv4 addr...	Type	Allocation ID	Reverse DNS reco
–	54.157.185.13	Public IP	eipalloc-04222c4a210feb9bc	–

View IP address usage and recommendations to release unused IPs with Public IP insights.

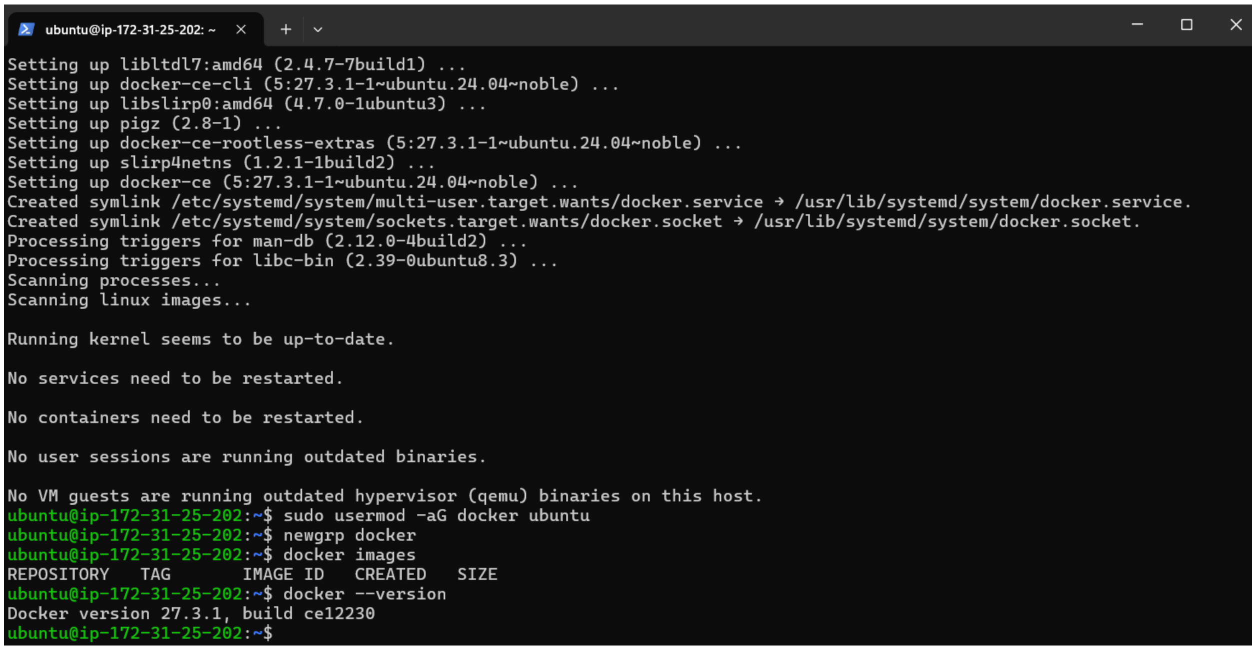
54.157.185.13

Summary

Allocated IPv4 address	Type	Allocation ID	Reverse DNS record
54.157.185.13	Public IP	eipalloc-04222c4a210feb9bc	–

© 2024, Amazon Web Services, Inc. or its affiliates.

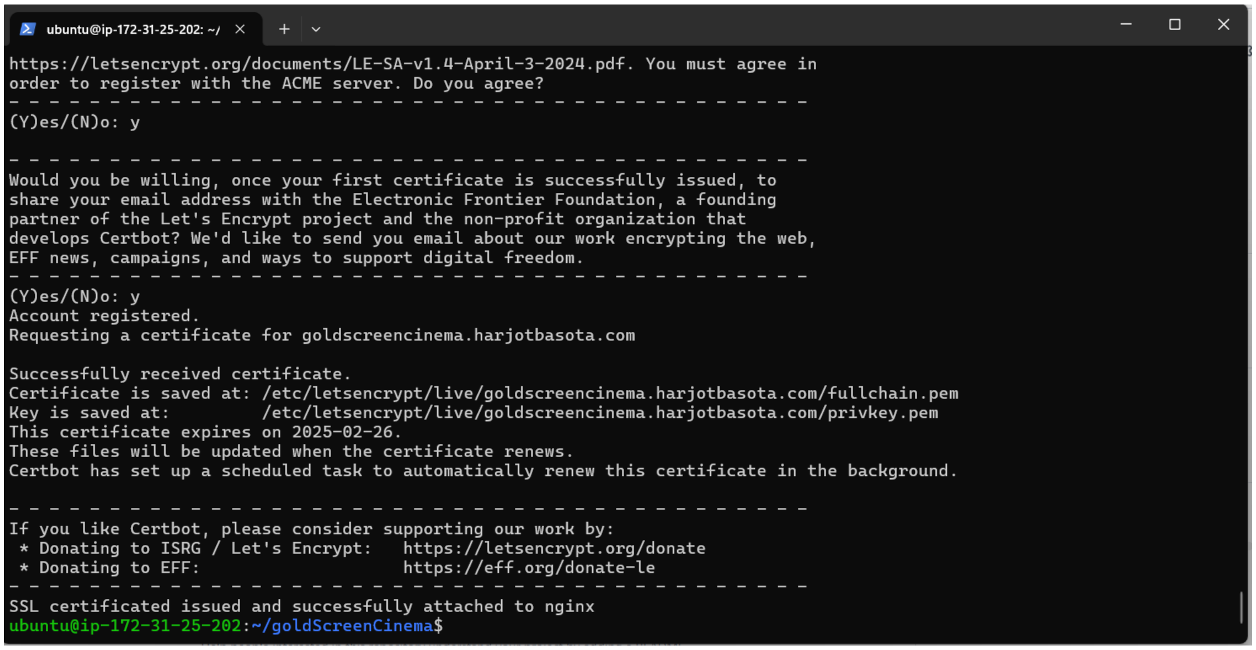
3. Install docker and add ubuntu user to the docker group



ubuntu@ip-172-31-25-202: ~

```
Setting up libltdl7:amd64 (2.4.7-7build1) ...
Setting up docker-ce-cli (5:27.3.1-1~ubuntu.24.04~noble) ...
Setting up libslirp0:amd64 (4.7.0-1ubuntu3) ...
Setting up pigz (2.8-1) ...
Setting up docker-ce-rootless-extras (5:27.3.1-1~ubuntu.24.04~noble) ...
Setting up slirp4netns (1.2.1-1build2) ...
Setting up docker-ce (5:27.3.1-1~ubuntu.24.04~noble) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.3) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-25-202:~$ sudo usermod -aG docker ubuntu
ubuntu@ip-172-31-25-202:~$ newgrp docker
ubuntu@ip-172-31-25-202:~$ docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
ubuntu@ip-172-31-25-202:~$ docker --version
Docker version 27.3.1, build ce12230
ubuntu@ip-172-31-25-202:~$
```

4. Clone GitHub repo and execute generateSSL script as sudo to get the ssl certificate and attach it to nginx and api. **Make privkey read for all.** (NOTE: set the domain name in this script before running and update the nginx configuration file for your domain. After changing, sync with git hub)



ubuntu@ip-172-31-25-202: ~/

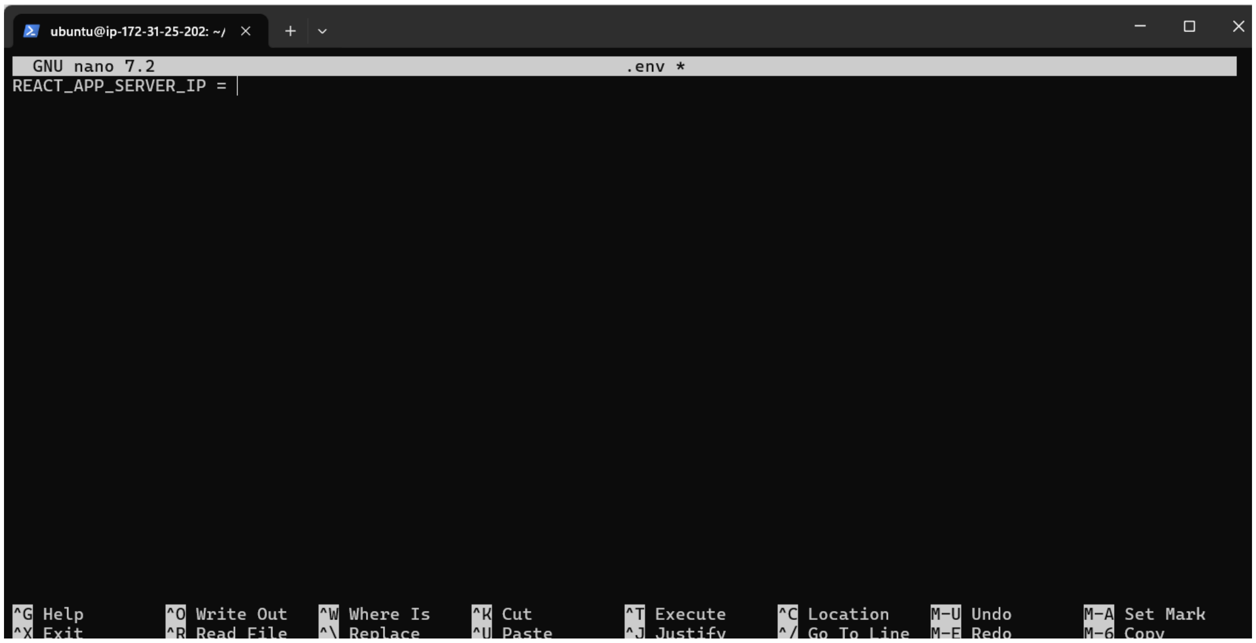
```
https://letsencrypt.org/documents/LE-SA-v1.4-April-3-2024.pdf. You must agree in
order to register with the ACME server. Do you agree?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: y

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Would you be willing, once your first certificate is successfully issued, to
share your email address with the Electronic Frontier Foundation, a founding
partner of the Let's Encrypt project and the non-profit organization that
develops Certbot? We'd like to send you email about our work encrypting the web,
EFF news, campaigns, and ways to support digital freedom.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: y
Account registered.
Requesting a certificate for goldscreencinema.harjotbasota.com

Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/goldscreencinema.harjotbasota.com/fullchain.pem
Key is saved at:         /etc/letsencrypt/live/goldscreencinema.harjotbasota.com/privkey.pem
This certificate expires on 2025-02-26.
These files will be updated when the certificate renews.
Certbot has set up a scheduled task to automatically renew this certificate in the background.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
If you like Certbot, please consider supporting our work by:
 * Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
 * Donating to EFF:                    https://eff.org/donate-le
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
SSL certificated issued and successfully attached to nginx
ubuntu@ip-172-31-25-202:~/goldScreenCinema$
```

5. Setup the env files in both api and frontend folder.

Frontend: REACT_APP_SERVER_IP (your domain name here)

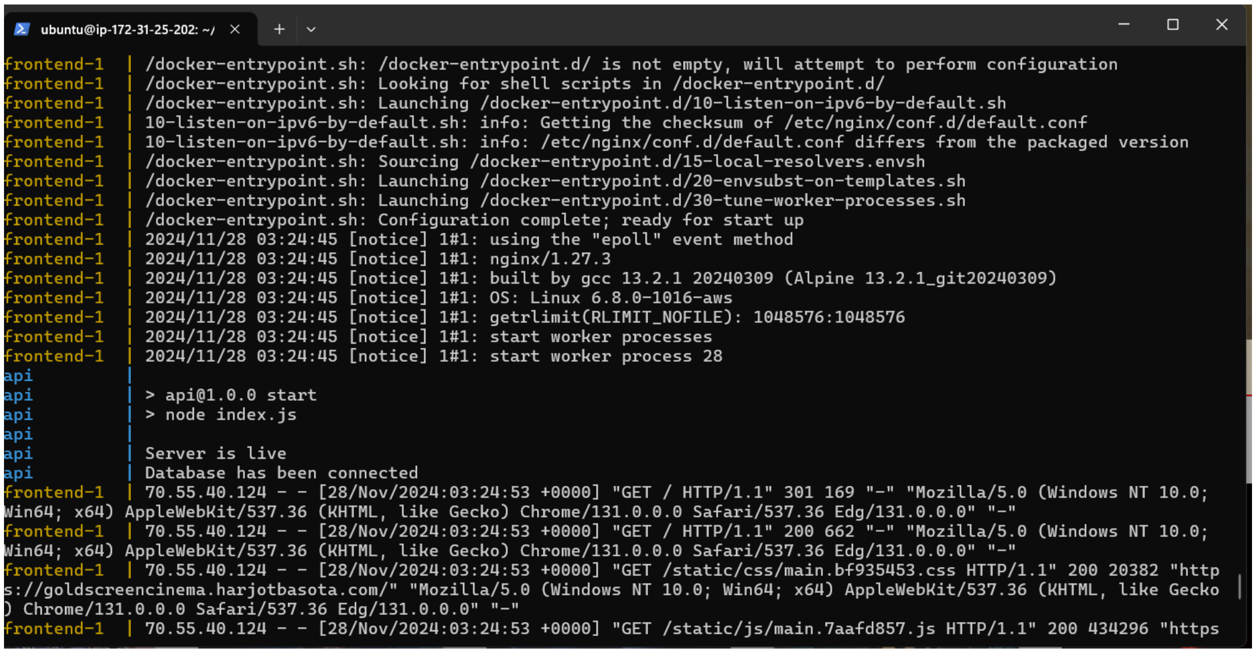
Api: PORT ,SERVER_IP(domain name here), JWT_SECRET_ACCESS_KEY

, JWT_REFRESH_TOKEN_SECRET, MONGO_AUTH



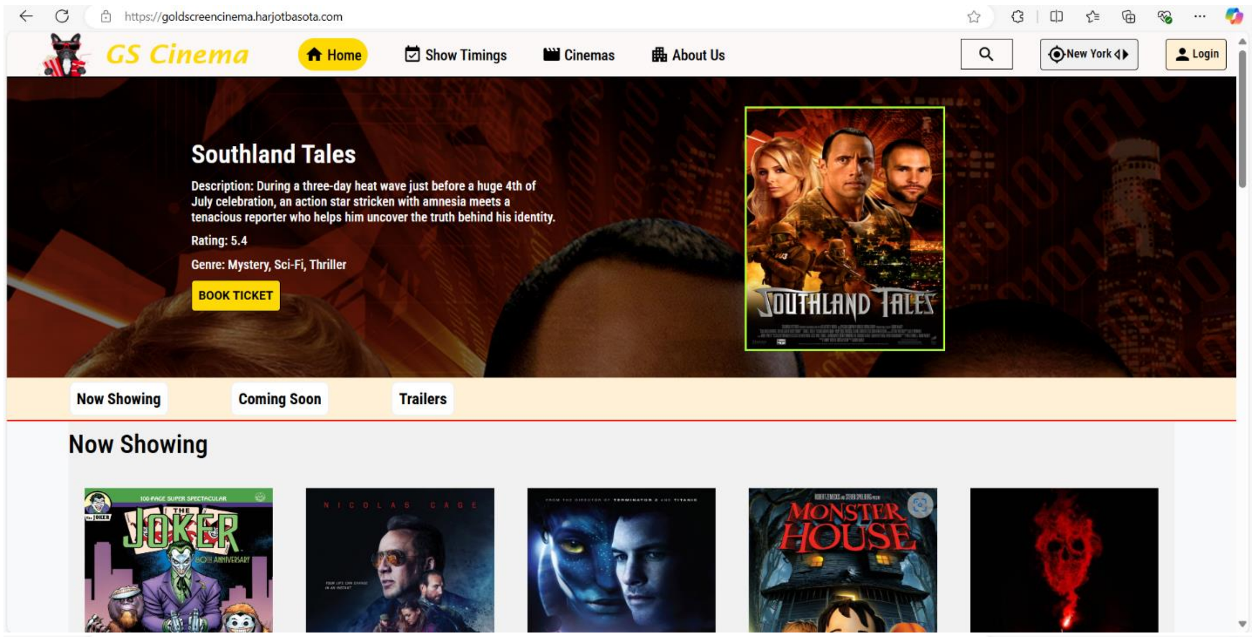
```
ubuntu@ip-172-31-25-202: ~/
  GNU nano 7.2                                           .env *
REACT_APP_SERVER_IP =

^G Help          ^O Write Out     ^W Where Is      ^K Cut           ^T Execute       ^C Location      M-U Undo         M-A Set Mark
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify       ^/ Go To Line    M-E Redo         M-6 Copy
```

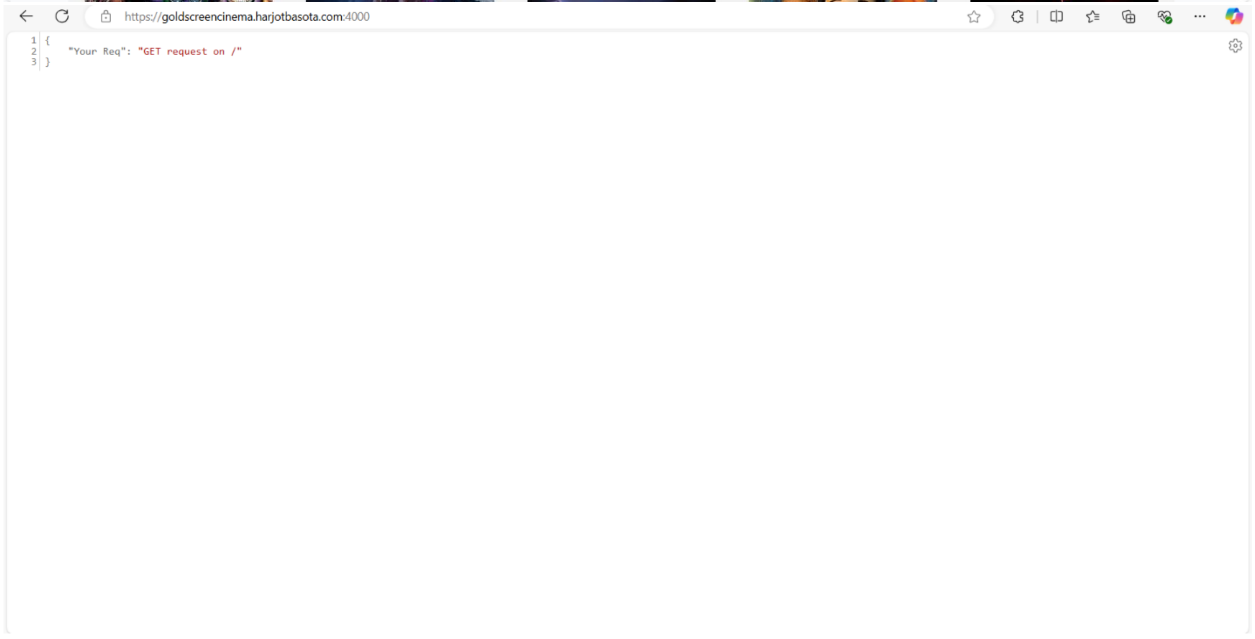
6. Execute docker compose up



```
ubuntu@ip-172-31-25-202: ~/
frontend-1  | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
frontend-1  | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
frontend-1  | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
frontend-1  | 10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
frontend-1  | 10-listen-on-ipv6-by-default.sh: info: /etc/nginx/conf.d/default.conf differs from the packaged version
frontend-1  | /docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
frontend-1  | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
frontend-1  | /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
frontend-1  | /docker-entrypoint.sh: Configuration complete; ready for start up
frontend-1  | 2024/11/28 03:24:45 [notice] 1#1: using the "epoll" event method
frontend-1  | 2024/11/28 03:24:45 [notice] 1#1: nginx/1.27.3
frontend-1  | 2024/11/28 03:24:45 [notice] 1#1: built by gcc 13.2.1 20240309 (Alpine 13.2.1_git20240309)
frontend-1  | 2024/11/28 03:24:45 [notice] 1#1: OS: Linux 6.8.0-1016-aws
frontend-1  | 2024/11/28 03:24:45 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
frontend-1  | 2024/11/28 03:24:45 [notice] 1#1: start worker processes
frontend-1  | 2024/11/28 03:24:45 [notice] 1#1: start worker process 28
api         |
api         | > api@1.0.0 start
api         | > node index.js
api         |
api         | Server is live
api         | Database has been connected
frontend-1  | 70.55.40.124 - - [28/Nov/2024:03:24:53 +0000] "GET / HTTP/1.1" 301 169 "-" "Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36 Edg/131.0.0.0" "-"
frontend-1  | 70.55.40.124 - - [28/Nov/2024:03:24:53 +0000] "GET / HTTP/1.1" 200 662 "-" "Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36 Edg/131.0.0.0" "-"
frontend-1  | 70.55.40.124 - - [28/Nov/2024:03:24:53 +0000] "GET /static/css/main.bf935453.css HTTP/1.1" 200 20382 "http
s://goldscreencinema.harjotbasota.com/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko
) Chrome/131.0.0.0 Safari/537.36 Edg/131.0.0.0" "-"
frontend-1  | 70.55.40.124 - - [28/Nov/2024:03:24:53 +0000] "GET /static/js/main.7aafd857.js HTTP/1.1" 200 434296 "https
```

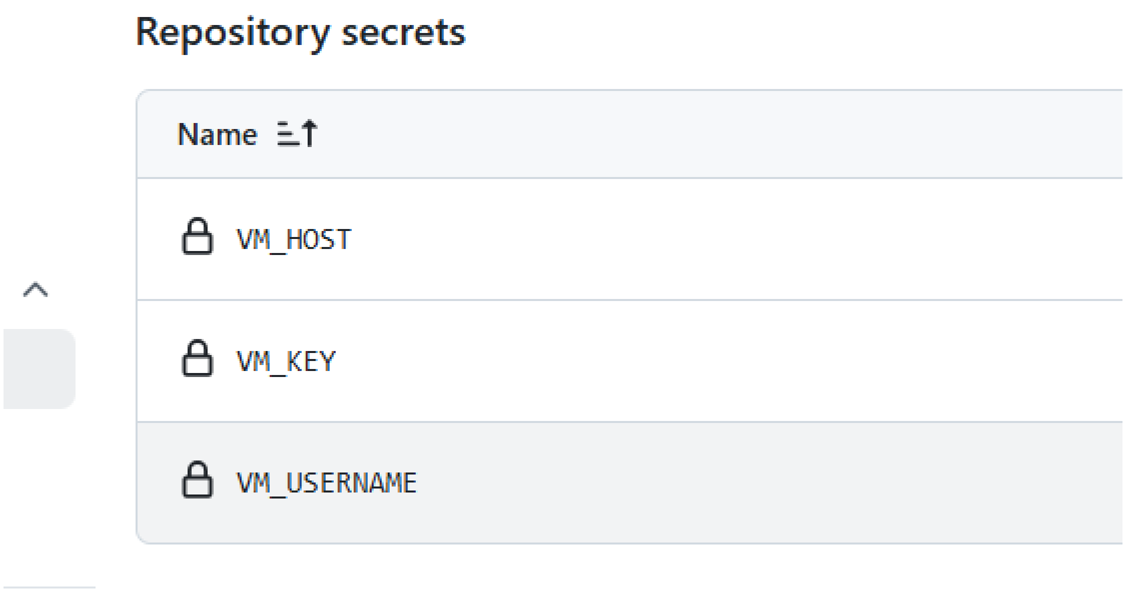
7. Check live website and api server at port 4000


https://goldscreencinema.harjotbasota.com
GS Cinema
Home
Show Timings
Cinemas
About Us
New York
Login
Southland Tales
Description: During a three-day heat wave just before a huge 4th of July celebration, an action star stricken with amnesia meets a tenacious reporter who helps him uncover the truth behind his identity.
Rating: 5.4
Genre: Mystery, Sci-Fi, Thriller
BOOK TICKET
Now Showing
Coming Soon
Trailers
Now Showing



```
{
  "Your Req": "GET request on /"
}
```

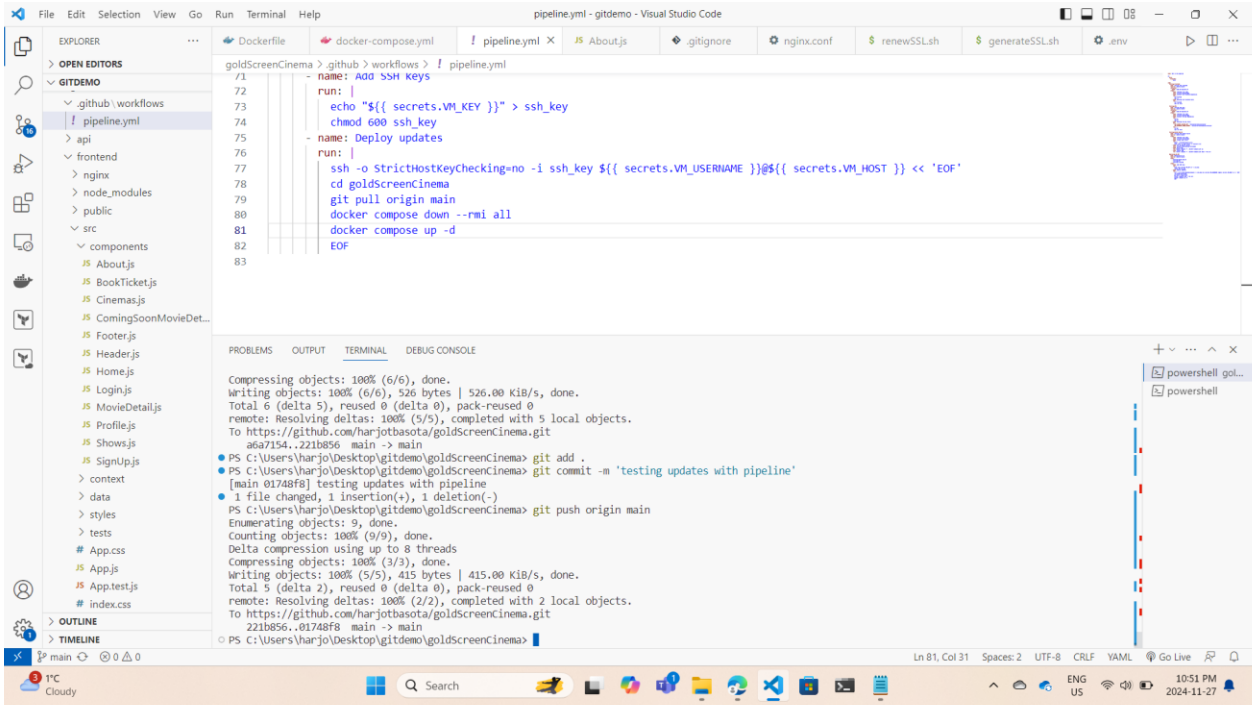
8. Add the following secrets repository at GitHub



Repository secrets

Name
VM_HOST
VM_KEY
VM_USERNAME

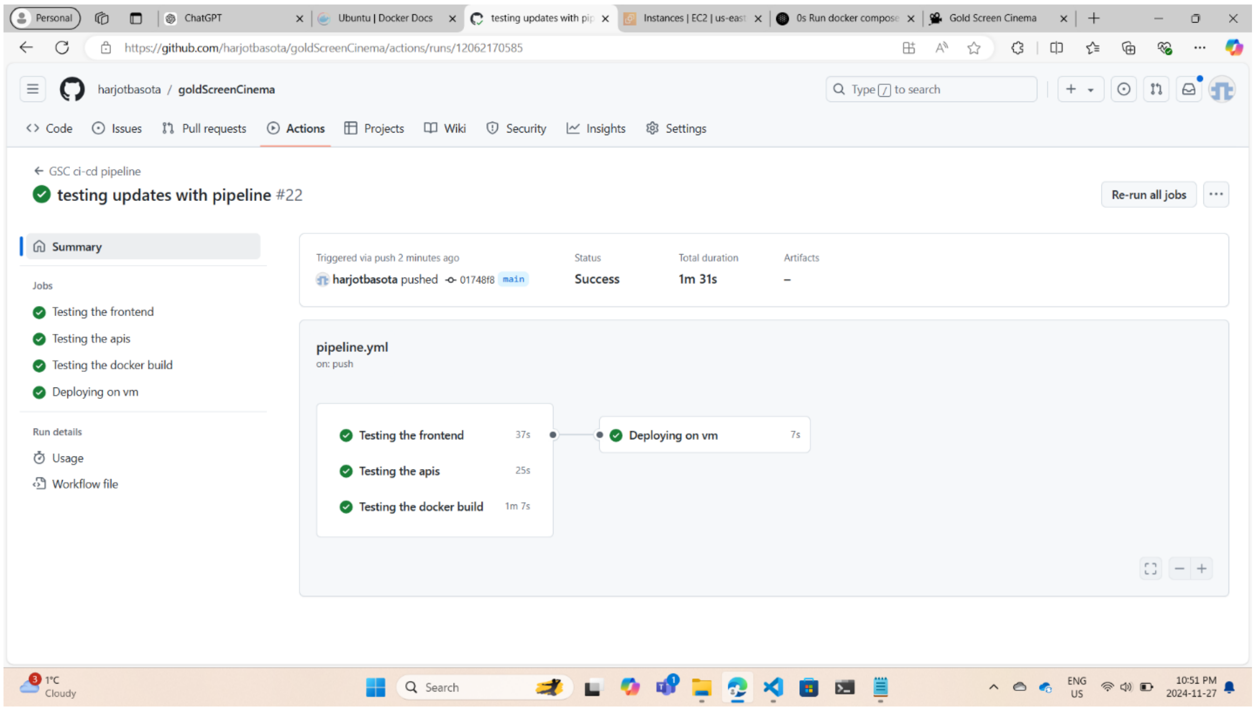
9. Test the pipeline by making changes and pushing it to github



pipeline.yml - gitdemo - Visual Studio Code
goldScreenCinema > .github > workflows > pipeline.yml

```yaml
      - name: Add SSH keys
        run: |
          echo "${{ secrets.VM_KEY }}" > ssh_key
          chmod 600 ssh_key
      - name: Deploy updates
        run: |
          ssh -o StrictHostKeyChecking=no -i ssh_key ${{ secrets.VM_USERNAME }}@${{ secrets.VM_HOST }} << 'EOF'
          cd goldScreenCinema
          git pull origin main
          docker compose down --rmi all
          docker compose up -d
          EOF
```

```
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 526 bytes | 526.00 KiB/s, done.
Total 6 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), completed with 5 local objects.
To https://github.com/harjotbasota/goldScreenCinema.git
   a6a7154..221b856  main -> main
PS C:\Users\harjo\Desktop\gitdemo\goldScreenCinema> git add .
PS C:\Users\harjo\Desktop\gitdemo\goldScreenCinema> git commit -m 'testing updates with pipeline'
[main 01748f8] testing updates with pipeline
 1 file changed, 1 insertion(+), 1 deletion(-)
PS C:\Users\harjo\Desktop\gitdemo\goldScreenCinema> git push origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 415 bytes | 415.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/harjotbasota/goldScreenCinema.git
   221b856..01748f8  main -> main
PS C:\Users\harjo\Desktop\gitdemo\goldScreenCinema>
```

10. 

GSC ci-cd pipeline

testing updates with pipeline #22

Triggered via push 2 minutes ago — harjotbasota pushed 01748f8 main

Status: Success

Total duration: 1m 31s

Artifacts: –

pipeline.yml
on: push

Testing the frontend 37s
Testing the apis 25s
Testing the docker build 1m 7s
Deploying on vm 7s