International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 02 Issue: 10 October 2024
Page No: 2529 - 2537
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2024.0347

Cinematic Cloud: Implementing a Video Streaming Platform with Containerization, Infrastructure as Code, And CI/CD Data Pipelines

Vinisha Manoju[1], Likhita Konakalla[2], Maneesha Vedantham[3], C Praveen Kumar[4]
[1,2,3]UG -Computer Science and Engineering, Institute of Aeronautical Engineering, Hyderabad, Telangana, India.
[4]Associate Professor, Computer Science and Engineering, Institute of Aeronautical Engineering, Hyderabad, Telangana, India.
Emails: *mvinisha2004@gmail.com[1], konalikhita@gmail.com[2], maneeshavedantham@gmail.com[3], c.praveenkumar@iare.ac.in[4]*

Abstract
In today's digital landscape, video streaming platforms have become the cornerstone of entertainment consumption, reshaping how we engage with media. However, traditional approaches to building these platforms, often relying on monolithic architectures, are struggling to keep pace with the evolving demands of users and the complexities of modern infrastructure. This solution explores a new approach that uses contemporary methods like DevOps and microservices to make video streaming platforms faster, more reliable, and easier to use. The system starts by providing a user interface for streaming content and its entire backend architecture is built using specialized tools and methodologies. The methodologies include containerization using Docker and automation of IT infrastructure using Terraform or AWS CloudFormation which serve as Infrastructure as Code. Lastly, CI/CD pipelines are used to automate all stages of the SDLC life cycle. The application will be deployed on AWS cloud and there will be extensive use of AWS services.

Keywords: *Cloud Computing; AWS; Docker; Adaptive Bitrate Streaming algorithm; containerization; CloudFormation; Infrastructure as Code; CI/CD pipeline.*

1. Introduction
Video streaming is becoming ubiquitous-from the mass- market mainstream platforms like Netflix and Amazon Prime Video to the social media behemoths: Instagram. So, the streaming of videos over the internet without cable is named OTT or over-the-top streaming. However, in the background,building these streaming applications is no cakewalk. Traditionally, the whole pieces of software had to be looked after by developers. This caused the problem with scaling and smooth deployment. This is where containerization comes into the picture. It is the new silver bullet that simplifies the process of deploying software by packaging applications and dependencies into a small container. Such containers carry benefits such as flexibility and consistency across different environments; therefore, tremendous importance should be attached to the streamlining of the processes of deployment. Containers are not sufficient but can be optimized in the right approach if using a DevOps approach. It is the approach which focuses on producing collaboration, automation, and continuous improvement throughout the software development life cycle. In the practice of DevOps, its implementation is going to be used with a CI/CD pipeline. CI/CD refers to continuous integration of code, continuous delivery of products, and continuous deployment of products. CI/CD makes use of tools like Jenkins that provide automation for the testing and delivery of the code also changes hence provide support for rapidly and reliably deployed features. For this reason, infrastructure, servers etc. also get automated through an Infrastructure as Code. [1-5]

2. Literature Review
Video streaming platforms have literally transformed the mode through which content is consuming worldwide with its fast growth. With

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 02 Issue: 10 October 2024
Page No: 2529 - 2537
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2024.0347

this increase in demand for quality and ad- free streaming services, the underlying technology and architecture that power these platforms have evolved by default. Traditionally, monolithic architectures were used to build such a platform, though usually resulting in problems with scalability, flexibility, and deployment. With rising user expectations, comes the scale of video content to handle and deliver. Thus, a need for a more modern and agile approach has come forth. Cloud computing in video streaming came along with tremendous prospects of scalability and flexibility offering immense performance in delivery and consumption of video content. Innovations in cloud-based infrastructure have contributed significantly to live streaming; for example, enhanced cache capacity, adaptive bitrate streaming, and reduced latency, according to Kumar et al. (2024). All of these aspects seem to have exceptional importance in relation to high-quality video delivery to a myriad of devices in different network conditions, making cloud computing an integration necessity in modern streaming services. In other technical details of cloud-based video stream, Li et al. (2021) explained how in both encoding, transcoding, and encryption processes, these developed within the implementation of the cloud-based application to make content delivery easy for users without addressing any momentous research challenges today, such as bandwidth management, data security, and therefore constant innovation required to improve the quality of video streaming. The strategies that such streaming services have to employ about DevOps and containerization regarding scaling and optimization have been discussed by Cukier (2013) and Prabowo et al. (2020). This approach serves to present the evidence of improving resource usage and easy service migration between cloud environments when using orchestrators for containers, like Kubernetes. These developments not only aid in smoothening deployments but also allow streaming services to adapt very fast in changing demands at hand; therefore, overall system reliability and performance increase. [6-10]

3. Video Streaming

Video streaming is the continuous transmission of video and audio data from a server to a client. It allows users to watch videos online without downloading them and sometimes embed them in high resolutions up to 4K. Video streaming can provide a variety of content on the Internet, such as interesting videos, video conferences, movies, live concerts, etc. Whenever a video is played, it begins streaming from the server to the user's device in real time. The user's device can be anything from a computer or laptop to a phone which has internet access. The video and audio files are broken down into small pieces called packets, which are then transmitted over the internet to the user's device. These packets are reassembled by the client device into video and audio files. There are two types of video streaming:

- On-Demand Streaming: This involves using pre-recorded videos to deliver content. Examples include video lectures, shows, and movies on platforms like Netflix.
- Live Streaming: These broadcasts live video and audio content in real time. Examples include sports events and YouTube live streams.

For on-demand streaming, we use HTTP streaming protocols such as HLS (HTTP Live Streaming) and DASH (Dynamic Adaptive Streaming over HTTP). For live streaming, protocols like RTMP (Real-Time Messaging Protocol) and DASH are commonly used. When delivering video content, a higher bitrate transmits more information, resulting in higher quality. However, a higher bitrate also requires more storage space and bandwidth. To manage this, we use ABS (Adaptive Bitrate Streaming). ABS works by encoding the video at multiple bitrates and allowing the viewer's device to automatically switch between different bitrates based on the internet speed. This ensures the highest possible quality without buffering or interruptions.

4. Abbreviations and Acronyms

- ABS - Adaptive Bitrate Streaming
- VPC – Virtual Private Cloud
- CDN – Content Delivery Network
- IaC – Infrastructure as Code
- IaaS – Infrastructure as a Service

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 02 Issue: 10 October 2024
Page No: 2529 - 2537
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2024.0347

- PaaS – Platform as a Service
- AWS – Amazon Web Services
- SDLC – Software Development Lifecycle
- API – Application Program Interface
- OTT – Over The Top
- EBS – Elastic Block Storage
- EFS – Elastic File Storage
- EC2 – Elastic Compute Cloud
- DASH – Dynamic Adaptive Streaming over HTTP
- CI/CD-Continuous Integration and Continuous Deployment.

5. Existing System

The current video streaming platform is built using a monolithic architecture, where all components, such as the user interface, business logic, and data access layer, are tightly integrated and operate as a single service. The entire application is deployed as a single unit, meaning any update or change requires redeployment of the entire system. Scaling is typically achieved by duplicating the entire application across multiple servers. Testing is conducted only after the implementation phase is complete.

5.1. Drawbacks of the Existing System

- **Difficulty in Introducing New Services:** The tightly coupled nature of the monolithic architecture makes adding new services cumbersome and time-consuming. Every change affects the entire system, leading to prolonged development cycles.
- **Intensive Testing Requirements:** The close integration of components necessitates frequent and extensive unit and system testing. Regression testing must be performed at an extensive level to ensure that if changes are made to one part of the application, the rest of the code doesn't break due to dependencies.

Rigid Technology Stack: In monolithic architecture, a single technology stack is used to build the entire application. When new technologies arise that are more efficient, it is difficult to migrate current services to the new technologies without rebuilding the entire application. Hence, the application becomes outdated and inefficient compared to competitive applications. (Refer Figure 1) [11-16]

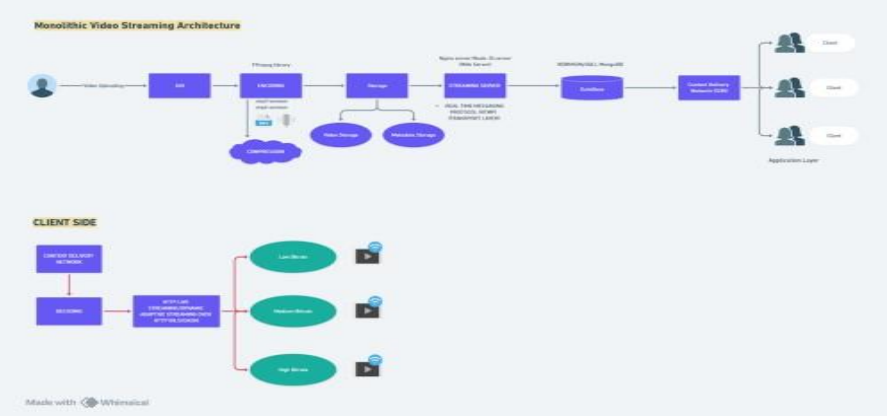

Figure 1 Monolithic Architecture of Video Streaming Platform

6. Proposed System

To overcome the limitations of monolithic architectures in video streaming platforms, the application uses micro services architecture to build the various components. This approach involves building independent micro services, each one responsible for a specific functionality in the video streaming platform.

6.1. Technologies and Tools

- **Terraform**: This tool automates the process of creation of infrastructure on AWS using Hashicorp language (HCL). This is an easier way of provisioning infrastructure in comparison to AWS management console.
- **AWS Account**: An AWS account is required to create infrastructure required for the video

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 02 Issue: 10 October 2024
Page No: 2529 - 2537
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2024.0347

streaming platform.

- **Docker Containers**: Docker is used for containerization of the components. Every container is composed of all the packages and dependencies required for the development and deployment of that specific component.
- **Container Deployment**: To deploy the containers, the containers are first stored in ECR (Elastic Container Registry) and managed using ECS (Elastic Container Service).
- **CI/CD Pipeline**: The CI/CD Pipeline is used to automate the deployment process, whenever new changes are made to the application. Multiple environments can be configured here like dev, stag, test and prod. This can be undergone using AWS developer tools or github actions.
- **Message Queuing:** AWS SQS is used for communication between micro services by passing and storing the messages in a queue. The component sending the message passes the message into a queue, and the component receiving the message, fetches it from the queue.
- **Content Delivery Network**: To ensure that the content is distributed in a fast and efficient way, AWS Cloud Front is utilized. AWS Cloud Front uses edge locations for caching frequently accessed data hence, providing faster delivery of videos.
- **Frontend Application** A client side interface is built for the users to interact with the application seamlessly. The interface provides various functionalities and features that improve the experience for users.
- **Adaptive Bitrate Streaming Algorithm**: This algorithm is essential for the video streaming platform. This algorithm is responsible for automatically adjusting the quality of video viewed by user based on the user's internet speed.
- **Streaming Protocols**: Two streaming protocols are implemented for adaptive bitrate streaming. HLS (HTTP Live Streaming) and DASH (Dynamic Adaptive Streaming over HTTP). Implements HLS (HTTP Live Streaming) and DASH (Dynamic Adaptive Streaming over HTTP) for adaptive bitrate streaming.

7. Methodology And Implementation

Terraform: Before the development and deployment of microservices, the necessary cloud infrastructure must be provisioned in the respective AWS account. This infrastructure includes Virtual Private Clouds (VPCs), public and private Subnets, Elastic Container Service (ECS), Elastic Container Registry (ECR), EC2 instances, security groups, etc. Terraform is used to easily provision the infrastructure by creating configuration files for the infrastructure. This way, the same configuration file can be used to provision huge number of infrastructure components without concern about manual errors. These configuration files also enable creation of same infrastructure in multiple environments using workspaces. Hence, the architecture is provisioned effectively. (Refer Figure 2)

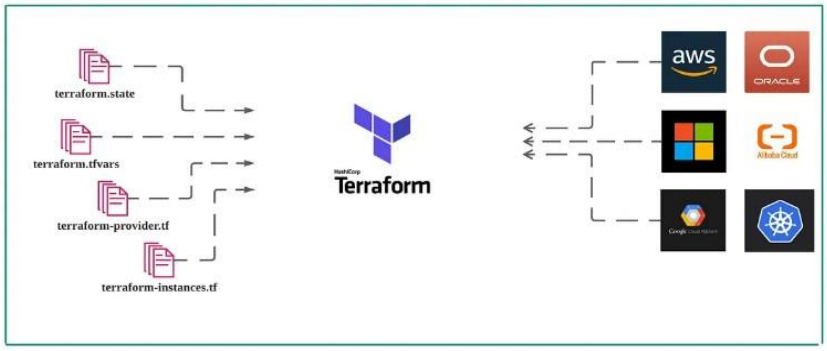

Figure 2 Terraform Templates to Build Infrastructure

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 02 Issue: 10 October 2024
Page No: 2529 - 2537
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2024.0347

Microservice: Implementing a microservices architecture allows each service to act as an isolated unit. Every service can be built with a different tech stack, and can be scaled individually based on the demand. The entire microservice is packaged as a docker container along with all the required dependencies for the service. The entire system is built as a decoupled architecture. The containers are developed in a Docker environment, and stored in a container registry (ECR). These containers are later managed and deployed using Elastic Containerization Service (ECS). This approach is flexible and suitable for high scalability. (Refer Figure 3)

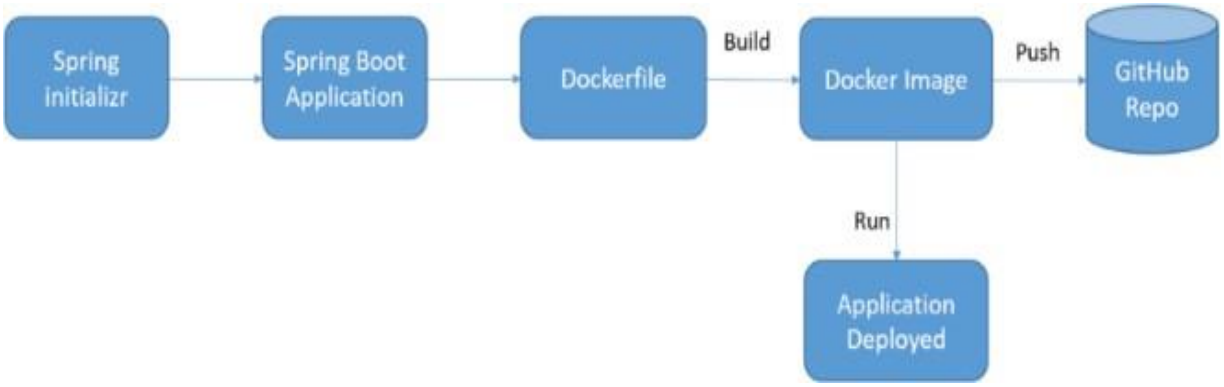

Figure 3 Implementing Micro Service Architecture Using Docker

Continuous Integration/Continuous Deployment (CI/CD): For every change made to the application, and with new versions being released, production must be an ongoing process. Therefore, the development and deployment processes will be streamlined. The platform utilizes AWS Developer Tools, such as AWS Code Pipeline, along with Github actions to implement the CI/CD pipeline. This ensures that updates to microservices are automatically tested in testing environment and deployed to go live. (Refer Figure 4)

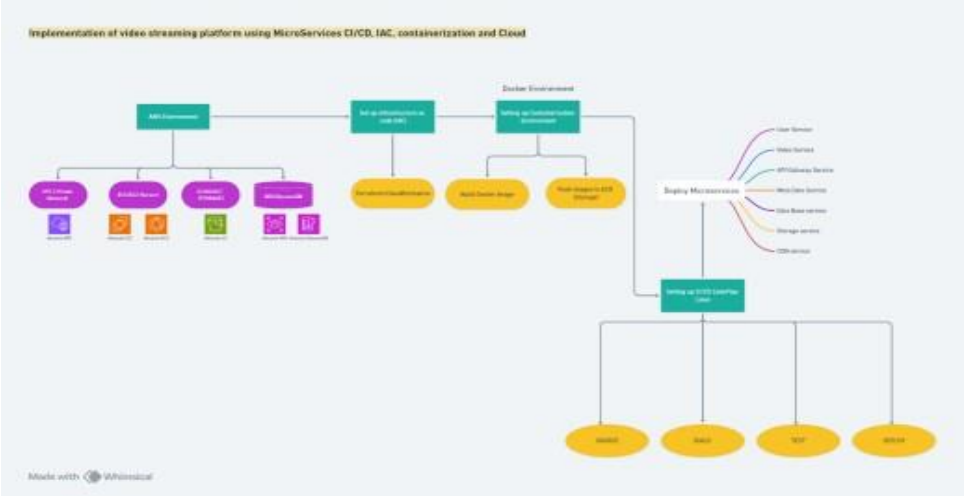

Figure 4 System Architecture

7.1. Microservices Overview

1. **Video Uploading Service:** This service enables users to upload videos using the client side interface.
2. **Video Processing Service:** This service manages encoding, transcoding, and other

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 02 Issue: 10 October 2024
Page No: 2529 - 2537
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2024.0347

processing tasks to ensure videos are available in varied formats and resolutions for the user.

3. **Storage Service:** To store raw and processed video files, AWS Simple Storage Service (S3) is used as a storage service.
4. **Database Service:** This service is used to host the databases that manage the application's data, along with user information, metadata, and service configurations.
5. **Metadata Service:** This service is responsible for handling the creation and management of metadata associated with the videos.
6. **CDN Service:** Integrates with AWS CloudFront for distributing content efficiently to users across the globe.
7. **User/Frontend Service:** Provides the frontend interface for users, including video playback and user account management. (Refer Figure 5)

Figure 5 Microservice Architecture of Video Streaming Platform

Inter-Service Communication: To facilitate communication between microservices, a queueing service is employed. Both AWS SQS and RabbitMQ are viable options for message queuing; however, the application uses AWS SQS. This queueing service facilitates asynchronous communication between components of a distributed system, allowing messages—representing information or tasks—to be transferred and processed by microservices as needed, allowing processes to remain decoupled and operate independently without relying on direct dependencies. (Refer Figure 6)

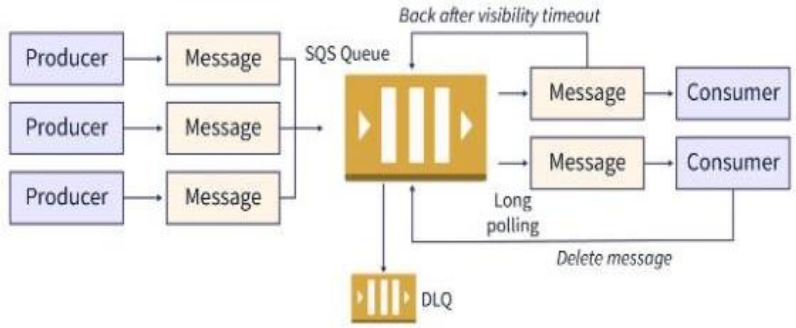

Figure 6 Inter-Service Communication Using Message Queue

Content Delivery: Once processed, the video content is distributed to users through a Content Delivery Network (CDN) to cache content and ensure low latency and high transfer speeds globally. While options like Cloudflare and CloudFront are available, the application

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 02 Issue: 10 October 2024
Page No: 2529 - 2537
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2024.0347

specifically utilizing AWS CloudFront for this purpose.

Frontend Development/User Service: Develop the frontend user interface using React, Angular, or Vue.js frameworks. Implement features such as video playback, user authentication, search functionality, etc.

Frontend Technologies:

- React/Angular

Middleware Technologies:

- NodeJS, ExpressJS

Video Streaming:

- FFmpeg Library

Algorithms:

- Adaptive Bitrate Streaming Algorithm
- Content Delivery Network Algorithms
- Load Balancing Algorithms

Functional Requirements:

- Content Upload: Content creators should be able to upload videos, including metadata (title, description, tags).
- Adaptive Streaming: Automatically adjust streaming quality based on user's bandwidth.
- Streaming Quality: Support for multiple streaming qualities (e.g., 240p, 360p, 480p, 720p, 1080p, 4K).

Store playback data: Save the playback position data on the server associated with the user's account

8. Results And Discussions

The resulting application is a robust video streaming platform architected using a microservices approach, ensuring scalability, maintainability, and fault tolerance. The infrastructure is provisioned using Terraform configuration files utilizing Infrastructure as Code (IAC) practices. This ensures consistency across all environments. Docker containers are used to package the various microservices into isolated units. They are deployed and orchestrated using AWS ECS. The platform utilizes CI/CD pipeline, by usage of AWS Code Pipeline. This pipeline automates the build, test, and deployment processes, reducing manual intervention and therefore reducing repetitive tasks. AWS SQS serves as the backbone for communication by ensuring that messages are passed effectively between microservices. It decouples the components and handles message queues efficiently. Additionally, the platform integrates AWS CloudFront as a Content Delivery Network (CDN) tool, which is responsible for caching and delivering video content to users in distant locations with low latency. The application also utilizes adaptive bitrate streaming algorithms to ensure that video quality is dynamically adjusted based on the user's network conditions and connectivity issues. This provides a smooth viewing experience for users. The primary objective of the platform is to provide top-quality video streaming that's quick, reliable, and able to handle a growing number of users, all while making sure the experience stays smooth and enjoyable for everyone. (Results are shown in Figure 7-10)

Conclusions and Future Scope

A. Conclusion

In conclusion, the growth of video streaming platforms has been greatly influenced by cloud computing, DevOps practices, and containerization technologies. These innovations have solved many problems that came with older, monolithic systems by providing scalable, flexible, and efficient content delivery solutions. DevOps, along with CI/CD pipelines, has made the development and deployment process faster and more reliable, allowing new features and updates to be released more smoothly. Containerization, especially using tools like Docker, has improved how streaming services are deployed, making it easier to scale quickly. As the video streaming industry continues to advance, using these technologies will be key for staying competitive and ensuring platforms can deliver smooth experiences in a more demanding digital world.

B. Future Scope

The current application uses a microservice design for a video streaming platform with an on-demand model, meaning videos are streamed as needed and safely stored. In the future, the goal is to expand this setup to include live streaming, allowing real-time video playback. This change will add interactive features like live chat, reactions, and comments, making the platform more engaging and offering a lively experience for both on-demand and live content.

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 02 Issue: 10 October 2024
Page No: 2529 - 2537
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2024.0347

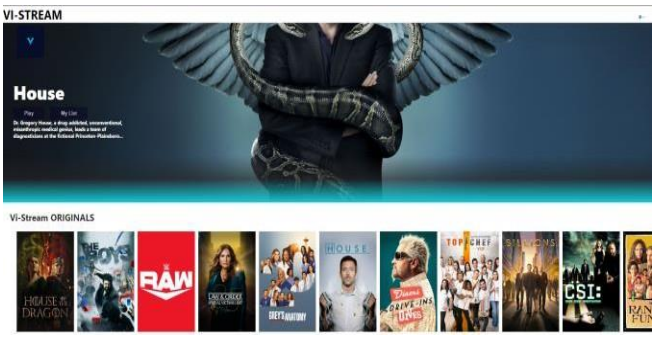

Figure 7 The Home Page

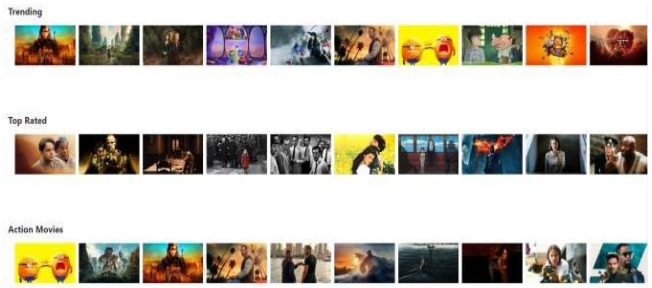

Figure 8 The Filter Section Displaying the Trending, Top Rated and Action Movies

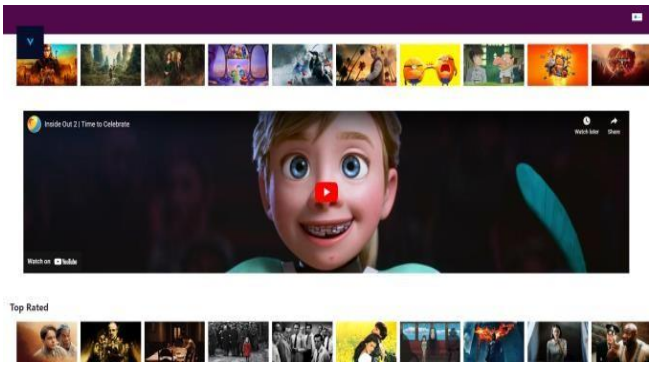

Figure 9 The Video Streaming Section

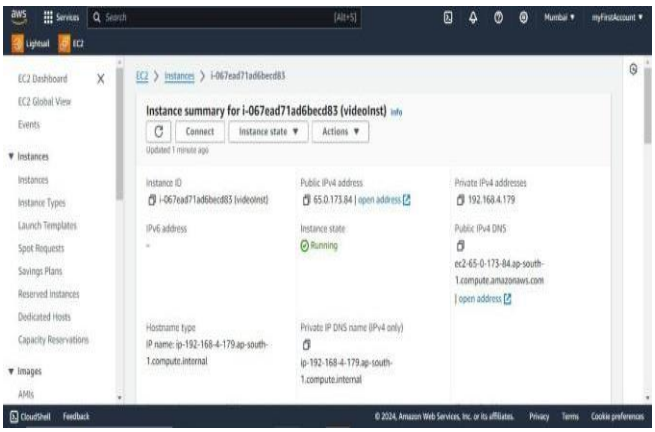

Figure 10 The Running EC2 Instance

References

[1]. Kumar, T., Sharma, P., Tanwar, J., Alsghier, H., Bhushan, S.,Alhumyani, H., Sharma, V., &Alutaibi, A. I. (2024). Cloud-based video streaming services: Trends, challenges, and opportunities. CAAI Transactions on Intelligence Technology, 9(2), 265–285.

[2]. Li, X., Darwich, M., Salehi, M. A., & Bayoumi, M. (2021). A survey on cloud-based video streaming services. In Advances in computers (pp. 193–244).

[3]. Sidik Prabowo,Rizal Dwi Prayogo,SA Karima(2020) Performance analysis of video streaming using container orchestration. IOP Conference Series Materials Science And Engineering 830(2):022100

[4]. Midoglu, C., Zabrovskiy, A., Alay, Ö., Hoelbling-Inzko, D., Griwodz, C., & Timmerer, C. (2019).Docker-Based Evaluation Framework for Video Streaming QOE in broadband networks.International Journal of Network Management.

[5]. Cukier, D. (2013). DevOps Patterns to scale web applications using cloud services.Proceedings of the 2013 companion publication for conference on Systems, programming, & applications: software for humanity

[6]. Patel, N., Shah, M., & Desai, K. Privacy-Preserving Data Sharing in IoT-Enabled Healthcare Systems(2020). 50, 771–998.

[7].

[8]. Bodi, A. (2024, July 25). Building live streaming and VOD workflows on AWS. TrackIt - Cloud Consulting & S/W Development. https://trackit.io/build-live-streaming-vod-workflows-on-aws/

[9]. Xiangbo Li*f, Mohsen Amini Salehif, Magdy Bayoumi*,"VLSC: Video Live Streaming Using Cloud Services",2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud).

[10]. Abubakr O. Al-Abbasi,"Multi-Tier Caching Analysis in CDNBased

[11]. Over-the-Top Video Streaming Systems",published in 2019.

[12]. Private Cloud Computing

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 02 Issue: 10 October 2024
Page No: 2529 - 2537
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2024.0347

Infrastructure",Vol. 7, No. 6, December 2017, pp. 3529~3535.

[13]. Xin Liu1, Dehai Zhao2, Liang Xu2, Weishan Zhang2, Jijun Yin1, And Xiufeng Chen1,"A Distributed Video Management

[14]. Cloud Platform Using Hadoop",date of current version December 22, 2015.

[15]. Akshay Kashyap,"Efficient HD Video Streaming Over the Internet".

[16]. Aws:"Working with Amazon s3 buckets".