

CLOUD-BASED CAMERA SECURITY SYSTEM

CSC/ECE 547 – Fall 2023

Cloud Computing Technology

Devadharshini Ayyappan (dayyapp), Sumukha Manjunath (smanjun3)

Plagiarism Declaration

We, the team members, understand that copying & pasting material from any source in our project is an allowed practice; we understand that not properly quoting the source constitutes plagiarism.

All team members attest that we have properly quoted the sources in every sentence/paragraph we have copy & pasted in our report. We further attest that we did not change words to make copy & pasted material appear as our work.

1. INTRODUCTION

1.1. Motivation:

The impetus behind the Cloud-Based Camera Security System project stems from a compelling need for cutting-edge security solutions in both residential and commercial contexts. This project is inspired by the desire to not only enhance safety and security but also provide unprecedented convenience, uphold privacy and compliance, and offer scalable and integrated security systems. The project's primary aim is to empower individuals to proactively protect their properties, acknowledging the modern appetite for remote monitoring and management while ensuring data privacy and adherence to regulations.

1.2. Executive Summary:

The Cloud-Based Camera Security System project responds to the surging demand for advanced security solutions in residential and commercial environments. By fusing state-of-the-art camera technology with cloud computing, this project empowers property owners to monitor and safeguard their premises with exceptional efficiency. With an emphasis on safety, convenience, privacy, and scalability, this project aspires to provide user-centric, privacy-aware, and feature-rich security systems. In essence, it's poised to revolutionize security provision for a safer and more interconnected world. This summary is intended for stakeholders and decision-makers, offering a succinct overview of the project's significance and objectives.

* ChatGPT was used in this section to rephrase the sentences and check for grammatical errors.

2. PROJECT REQUIREMENTS

2.1. The Problem:

This project addresses the increasing demand for advanced security solutions in both residential and commercial settings which prompts the need to develop a state-of-the-art Cloud-Based Camera Security System. This system aims to integrate modern camera technology with the power of cloud computing to empower property owners to effectively monitor and secure their premises. To effectively address this challenge, a robust and user-centric solution must be designed, enabling property owners to not only deter intruders and ensure safety but also manage their security systems remotely, safeguard user data, and seamlessly integrate with other smart devices.

The Cloud-Based Camera Security System project offers several valuable facilities to the consumers. Some of them include: Remote Monitoring, Real-Time Alerts, automated analysis for intruder detection, 24/7 security management, integration with local law enforcement services and data backup.

2.2. Business Requirements:

Below are the business requirements for the Cloud-Based Camera Security System.

Consumer's point of view:

BR1 24/7 video monitoring

BR2 Secure remote Access to videos

BR3 Provision to download security videos

BR4 Automatically alerts Law Enforcement and the consumer

BR5 Automated securing in case of a security breach

BR6 Optimization of security protocols and procedures

BR7 Back up for account details and security recordings

BR8 Enough memory allocation for the recordings

BR9 Enable smooth data transfers

BR10 Reduce cost in resource allocations

BR11 Follow rules and guidelines for data usage

2.3. Technical Requirements:

Below are the technical requirements for the Cloud-Based Camera Security System:

TR1.1 The system should be available 99%

TR1.2 Use additional data centers for application hosting as backup

TR2.1 Cloud storage for video archives.

TR2.2 User authentication

TR2.3 Multi-user access with varying permissions.

TR2.4 Identify a tenant and authorize access before performing any operation

TR3.1 Support for video exports in various formats.

TR3.2 Support user-friendly data management interfaces.

TR4.1 Threat detection/identification algorithm

TR4.2 GPS tagging for camera location.

TR4.3 Email alerts for intruder detection

TR5.1 Third-party device integration (e.g., smart locks)

TR6.1 Implement advanced analytics and reporting tools

TR7.1 Account recovery options.

TR7.2 User activity and access logs

TR7.3 Secure video backup

TR8.1 Scalable cloud storage plans.

TR8.2 Use additional data centers for application hosting as backup

TR9.1 Optimize network performance for seamless data transfer.

TR10.1 Restrict resource usage to appropriate limits to prevent cost surges.

TR11.1 Ensure regulatory compliance for data handling.

Mapping of the TRs with the AWS pillars:

- Operational Excellence: TR1.1 TR1.2 TR7.1 TR7.2 TR7.3
- Security: TR2.1 TR2.2 TR2.3 TR2.4 TR4.1 TR4.2 TR4.3 TR11.1
- Reliability: TR1.1 TR1.2 TR3.1 TR3.2 TR8.1 TR8.2
- Performance Efficiency: TR6.1 TR9.1
- Cost Optimization: TR8.1 TR10.1

Relationship between the BRs and the TRs:

BRs	TRs	Justification
BR1 24/7 video monitoring	TR1.1 The system should be available 99% TR1.2 Use additional data centers for application hosting as backup	Making the system available 99% of the time ensures the video monitoring system remains accessible and operational for the business's continuous 24/7 surveillance needs. The backup solution guarantees uninterrupted video monitoring even if one data center encounters issues, meeting the demand for continuous surveillance.
BR2 Secure remote Access to videos	TR2.1 Cloud storage for video archives. TR2.2 User authentication TR2.3 Multi-user access with varying permissions. TR2.4 Identify a tenant and authorize access before performing any operation	By utilizing cloud storage for video archives, the application can securely store video data remotely, while the identification of tenants and authentication ensures that only authorized users can access the stored videos. Multi-user access with varying permissions allows different users to access videos based on their roles, ensuring controlled access.
BR3 Provision to download security videos	TR3.1 Support for video exports in various formats. TR3.2 Support user-friendly data management interfaces.	By incorporating support for video exports in various formats and providing user-friendly data management interfaces, the system facilitates the provision to download security videos, ensuring easy access

		and retrieval of videos in preferred formats for the intended users.
BR4 Automatically alert the Law Enforcement and the consumer	TR4.1 Threat detection/identification algorithm TR4.2 GPS tagging for camera location TR4.3 Email alerts for intruder detection	The threat detection/identification algorithm continuously monitors video feeds for potential security threats, while GPS tagging enables the precise identification of camera locations for accurate alerts. Email alerts promptly notify both law enforcement and the consumer, ensuring timely and automatic communication of any detected intrusions along with the location details.
BR5 Automated securing in case of a security breach	TR5.1 Third-party device integration (e.g., smart locks)	Through third-party device integration, the system can link with smart locks to enable automated responses in case of a security breach. Once a breach is detected, the system triggers the smart locks to secure the premises, enhancing security measures and preventing unauthorized access effectively.
BR7 Back up for account details and security recordings	TR7.1 Account recovery options. TR7.2 User activity and access logs TR7.3 Secure video backup	Account recovery options allow users to regain access to their accounts in case of data loss or account compromise, while user activity and access logs provide a comprehensive record of system interactions to allow identification of potential issues or breaches. Secure video backup mechanisms ensure the periodic replication and storage of security recordings, enabling seamless data recovery in the event of data loss or system failure.
BR8 Enough memory allocation for the recordings	TR8.1 Scalable cloud storage plans. TR8.2 Use additional data centers for application hosting as backup	Scalable cloud storage plans allow for the flexible allocation of storage resources, ensuring that the system can accommodate the increasing volume of recordings without running out of memory.

		The use of additional data centers serve as backup storage allowing uninterrupted storage and access to the user in case of any unexpected loss/ temporary unavailability of primary data centers.
BR9 Enable smooth data transfers	TR9.1 Optimize network performance for seamless data transfer.	<p>Optimizing network performance involves utilizing various techniques such as load balancing, bandwidth management, and minimizing latency to ensure data is transferred efficiently and without interruptions.</p> <p>* ChatGPT was used for this.</p>
BR10 Reduce cost in resource allocations	TR10.1 Restrict resource usage to appropriate limits to prevent cost surges.	By enforcing restrictions on resource usage (compute, storage, network bandwidth, etc) to appropriate limits to prevent cost surges, the system can effectively reduce costs in resource allocations.
BR11 Follow rules and guidelines for data usage	TR11.1 Ensure regulatory compliance for data handling.	<p>Ensuring regulatory compliance for data handling results in adherence to legal standards and guidelines, safeguarding data privacy and security while promoting ethical and responsible data usage practices.</p> <p>* Chatgpt was used for this.</p>
BR6 Optimization of security protocols and procedures	TR6.1 Implement advanced analytics and reporting tools	<p>By implementing advanced analytics and reporting tools, the system can optimize security protocols and procedures by enabling the identification of potential vulnerabilities, providing insights for enhancing security measures, and facilitating informed decision-making to strengthen overall security protocols.</p> <p>* Chatgpt was used for this.</p>

2.4. Trade Offs:

TR9.1 and TR2.2:

The tradeoff between optimizing network performance for seamless data transfer and user authentication lies in the need to balance the efficiency of data transfer with the security of user access. Prioritizing one can often compromise the other, as stringent authentication measures may introduce latency and complexity, while emphasizing network speed might undermine security safeguards.

TR9.1 and TR10.1:

The tradeoff between optimizing network performance for seamless data transfer and restricting resource usage below appropriate limits arises from the need to balance efficient data transmission with cost control. While maximizing network performance ensures swift data transfers, it can lead to increased resource utilization, potentially causing cost overruns, and necessitating a delicate equilibrium between speed and resource efficiency.

TR10.1 and TR1.2:

The tradeoff between restricting resource usage to appropriate limits and using additional data centers for backup lies in the balance between cost control and system redundancy. Restricting resource usage can save costs but may risk downtime while using backup data centers ensures reliability but increases operational expenses.

TR10.1 and TR1.1:

The tradeoff between restricting resource usage to prevent cost surges and ensuring 99% system availability arises from the need to balance cost-efficiency with system reliability. Stricter resource limits can control expenses but might lead to occasional downtime, whereas prioritizing high availability can incur higher costs due to over-provisioning resources.

TR7.3 and TR10.1:

Balancing resource constraints below acceptable thresholds helps control operational costs, while ensuring secure video backup maintains data integrity and compliance, creating a tradeoff between financial efficiency and data protection.

* ChatGPT was used in this section to rephrase the sentences and check for grammatical errors.

3. PROVIDER SELECTION

3.1. Criteria For Choosing a Provider

The main criteria we considered for choosing the cloud provider for our project include

- **Service Offerings:** As the goal is to meet all the business requirements via the technical requirements, we looked for cloud providers whose services can be used to accomplish this.
- **Geographical Reach and Data Backup:** Since the services must be available 99% of the time, the provider's backup and disaster recovery options and the distribution and access to the database centers globally were key factors.
- **Performance:** Based on the reviews and track record of the provider's performance.
- **Developer Proficiency:** The developers of the project should be comfortable working with the cloud provider services.

3.2. Provider Comparison

Provider	Ranking			Reasoning
Criteria	AWS	Azure	GCP	
Service Offerings	1	2	3	AWS offers the widest range of services and has been in the cloud market for a longer time, making it a comprehensive choice. Azure comes next with a broad set of offerings, while GCP has a

				slightly smaller service catalog.
Geographical Reach and Data Backup	1	2	3	<p>AWS has a vast global network of data centers and robust disaster recovery options, making it a top choice for high availability and data backup.</p> <p>Azure also provides strong global coverage, while GCP's data center coverage is slightly more limited.</p>
Performance	1	2	3	<p>AWS is widely regarded for its exceptional performance due to its long-standing presence in the cloud market, massive global infrastructure, and vast user base, making it a top choice for applications that demand high speed and responsiveness.</p>

Developer Proficiency	1	2	3	The team members have prior experience using AWS and thus feel comfortable working with it. Additionally, this would impact the speed of the development of the project.
------------------------------	----------	----------	----------	--

3.3. The Final Selection

We chose AWS for our project because it offers the most services, has been in the cloud business for a long time and has a reputation for great performance. It's like having a big toolbox with all the tools we need. AWS also has many data centers worldwide and strong backup options, ensuring our project stays available. Plus, our team knows how to use it, so we can work faster and get things done efficiently. It's like having experienced builders for our project's construction. This makes AWS the perfect choice for our project's success.

3.3.1. The list of services offered by the winner

- **Amazon Elastic Compute Cloud** (Amazon EC2) and **Amazon Elastic Load Balancing** (ELB) can be used to ensure high availability by distributing traffic across multiple instances and data centers. **(TR1.1)**
- **Amazon Simple Storage Service** (Amazon S3) is an excellent choice for cloud storage. You can store video archives securely and cost-effectively. **(TR2.1)**
- **AWS Identity and Access Management** (IAM) provides user authentication and authorization capabilities, allowing you to control access to AWS resources securely. **(TR2.2)**
- **Amazon Elastic Transcoder** is a service that can be used to convert video files into different formats suitable for various devices and platforms. **(TR3.1)**

* ChatGPT was used in this section to rephrase the sentences and check for grammatical errors.

4. THE FIRST DESIGN DRAFT:

From the listed TRs in Section 2.4, we have selected the below:

Selected TRs	Mapped Services
TR2.4 Identify a tenant and authorize access before performing any operation (Tenant Identification)	AWS Identity and Access Management (IAM)
TR7.2 User activity and access logs (Monitoring)	Amazon CloudWatch Logs
TR4.3 Email Alerts for Intruder Detection (Monitoring)	Amazon Rekognition Amazon SES (Simple Email Service)
TR3.1 Support for Video Exports in Various Formats (Elasticity)	Amazon Elastic Transcoder
TR10.1 Restrict resource usage to below appropriate limits to prevent cost surges (Elasticity)	AWS Budgets and Cost Explorer
TR7.1 Account Recovery Options (Reliability TR from WAF)	Amazon Cognito
TR1.1 The system should be available 99% (Reliability TR from WAF and conflicting TR10.1)	Amazon EC2 Auto Scaling and Amazon Elastic Load Balancing (ELB)
TR2.1 Cloud storage for video archives.	Amazon S3

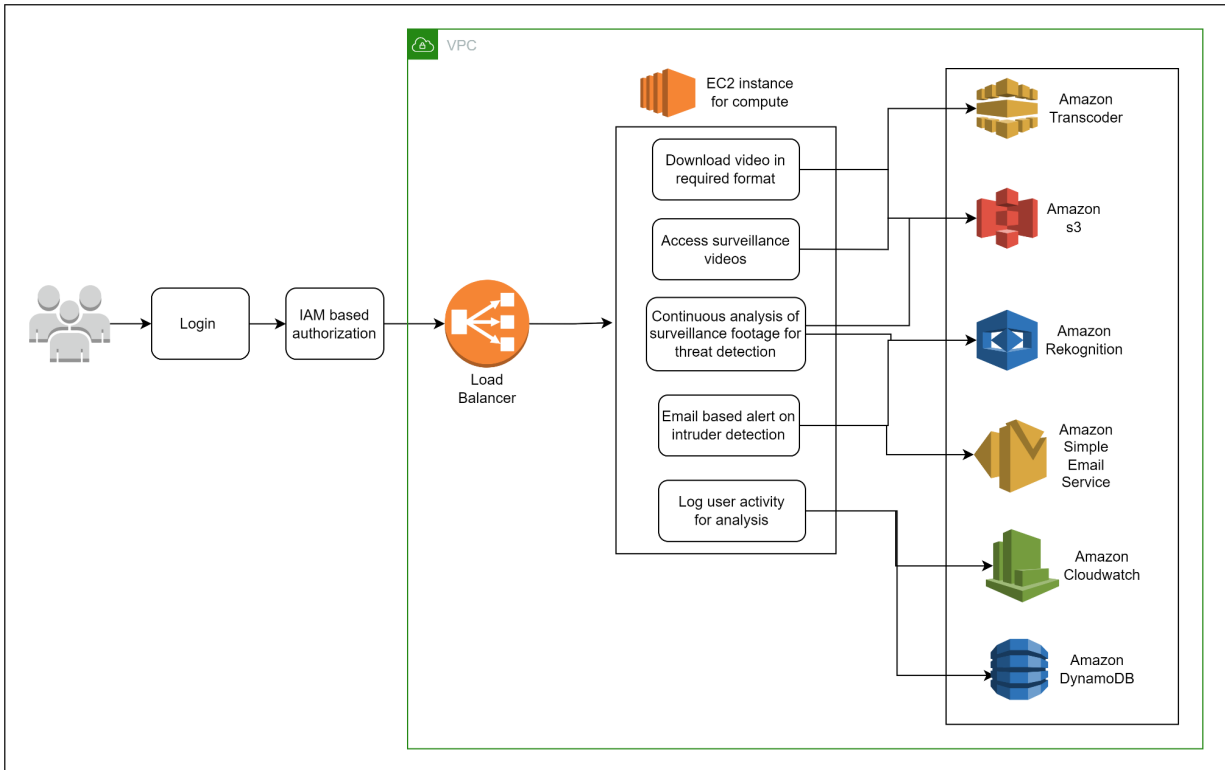


Figure 1. First design of Cloud Surveillance application

4.1. The basic building blocks of the design:

- **Storage:**
 - DynamoDB: Storage for tenant data.
 - Amazon S3: A secure and durable cloud storage solution for video archives.
- **Security:**
 - Cognito: For user identification and account management.
 - IAM Authentication: Enhances security by managing permissions and access controls for system users.
- **Compute:**
 - Rekognition: For intruder detection through video analysis.
 - EC2: Each user instance uses EC2 instances for performing the required operations.
- **Networking:**
 - EC2 Autoscaling: For system availability and uptime maintenance.
 - Load Balancing: For distributing the incoming traffic evenly.
- **Others:**
 - Amazon CloudWatch Logs: Used for real-time monitoring and log storage.
 - Amazon SES (Simple Email Service): For sending email alerts to the user.

- Amazon Elastic Transcoder: For converting the surveillance footage into different formats for export.

4.2. Top-level Informal Validation:

Storage:

- **Tenant Data:** DynamoDB is a highly available and scalable NoSQL database that meets the "Identify a tenant and authorize access before performing any operation" TR. It provides secure storage for tenant data, ensuring that authorized tenants can access the system.
- **Storing video recording:** Amazon S3 offers durable, scalable, and secure storage for video archives, addressing the "Cloud storage for video archives" TR. It is designed to handle large volumes of data efficiently and provides data durability.

Performance:

- **Amazon Rekognition:** Amazon Rekognition enhances performance by providing fast and accurate intruder detection through image and video analysis. This aligns with the "Email Alerts for Intruder Detection" TR, ensuring that intruders are promptly detected and appropriate alerts are sent.
- **Amazon Elastic Transcoder:** Amazon Elastic Transcoder contributes to performance by supporting the "Support for Video Exports in Various Formats" TR. It allows users to convert surveillance footage into different formats, making it versatile and accessible.

Security:

- **Cognito:** Amazon Cognito ensures security by managing user identities and providing secure authentication and authorization. It fulfills the "Identify a tenant and authorize access before performing any operation" TR, ensuring that only authorized tenants can access the system.
- **IAM Authentication:** IAM Authentication enhances security by managing permissions and access controls, aligning with the "Identify a tenant and authorize access before performing any operation" TR. It ensures that users are granted the appropriate level of access.

Networking:

- **EC2 Auto Scaling and Load Balancing:** The combination of Amazon EC2 Auto Scaling and Elastic Load Balancing (ELB) ensures high availability and fault tolerance,

fulfilling the "The system should be available 99%" TR. It distributes traffic evenly and automatically scales the application based on demand, maintaining uptime.

In summary, the proposed solution effectively aligns with the Technical Requirements by providing secure and scalable storage, enhancing performance through intruder detection and video export capabilities, and ensuring robust security and high availability through authentication and load balancing. These elements collectively support the successful implementation of the camera security monitoring application.

4.3. Action items and rough timeline

skipped

* ChatGPT was used in this section to rephrase the sentences and check for grammatical errors.

5. THE SECOND DESIGN

5.1. Use of the Well-Architected Framework:

The AWS Well-Architected Framework consists of six distinct pillars, which provide a structured approach to assess and improve your architecture's performance and efficiency. The six pillars are as follows:

- **Operational Excellence:** This pillar focuses on the ability to support the development and operation of workloads effectively. It emphasizes gaining insight into operations and continuously improving supporting processes and procedures to deliver business value.
- **Security:** The Security pillar is all about using cloud technologies to protect data, systems, and assets effectively. It helps improve your security posture and is crucial for ensuring that your architecture is secure.
- **Reliability:** This pillar encompasses the ability of a workload to perform its intended function correctly and consistently when expected. It also includes the ability to operate and test the workload throughout its lifecycle, ensuring it remains reliable.
- **Performance Efficiency:** Performance Efficiency is about using computing resources efficiently to meet system requirements. It focuses on maintaining efficiency as demand changes and as technologies evolve.
- **Cost Optimization:** The Cost Optimization pillar is dedicated to running systems that deliver business value at the lowest possible cost. It's essential for optimizing your infrastructure spending and resource utilization.
- **Sustainability:** Sustainability highlights the importance of continually improving sustainability impacts. It involves reducing energy consumption and increasing efficiency across all components of a workload by maximizing the benefits from provisioned resources and minimizing the total resources required.

These six pillars provide a comprehensive framework for assessing and improving the architecture of your applications and workloads on AWS. By focusing on these areas, you can ensure that your systems are well-architected, which leads to better performance, security, and cost-effectiveness.

5.2. Discussion of pillars:

Let's delve into two of the pillars of the AWS Well-Architected Framework in more detail:

Security: The Security pillar is of utmost importance in any cloud architecture. It focuses on safeguarding data, systems, and assets to ensure the confidentiality, integrity, and availability of your resources. Here are some key aspects of the Security pillar:

- **Identity and Access Management (IAM):** IAM is central to controlling access to AWS resources. You must implement proper identity and access management policies to ensure that only authorized individuals or systems can interact with your resources. This includes user roles, permissions, and the principle of least privilege.
- **Data Protection:** Data security is crucial. You need to encrypt sensitive data at rest and in transit. AWS provides services like AWS Key Management Service (KMS) for encryption and Amazon Macie for data discovery and classification.
- **Network Security:** This pillar emphasizes the secure design of your network architecture. Utilize Virtual Private Clouds (VPCs), security groups, and Network Access Control Lists (NACLs) to control network access. AWS WAF (Web Application Firewall) helps protect web applications from common web exploits.
- **Incident Response:** Being prepared for security incidents is vital. Establish an incident response plan, and use AWS services like Amazon GuardDuty for threat detection and AWS Config for continuous monitoring.
- **Compliance and Governance:** Adhere to regulatory requirements and implement governance practices to ensure that your architecture complies with industry standards and internal policies.

Cost Optimization: Cost Optimization is about getting the most value from your cloud resources while minimizing unnecessary spending. Here's a deeper look at this pillar:

- **Resource Right-Sizing:** Select the right type and size of resources based on your actual workload requirements. AWS provides tools like AWS Trusted Advisor and AWS Cost Explorer to help you analyze and optimize your resource usage.
- **Elasticity:** Take advantage of auto-scaling capabilities to match your resources with the current demand. This ensures you're not over-provisioning resources, especially during peak loads.
- **Reserved Instances (RIs):** RIs allow you to reserve instances in exchange for a lower hourly rate, providing significant cost savings for steady-state workloads.
- **Spot Instances:** For non-critical and fault-tolerant workloads, you can use Spot Instances to access spare AWS capacity at a much lower cost. These are ideal for batch processing and data analysis tasks.
- **Cost Allocation:** Implement effective cost allocation and tagging strategies to monitor and allocate costs to different teams or projects. This helps in understanding where your expenses are concentrated.
- **Analyze and Monitor:** Continuously monitor your cloud resources to identify opportunities for cost reduction. AWS provides various tools for this, such as AWS Cost Explorer, AWS Budgets, and AWS Cost and Usage Reports.

These pillars are not only about initial design but require regular review and adaptation as the architecture evolves. Paying attention to Security and Cost Optimization, ensures AWS workloads are both secure and cost-efficient.

5.3. Use of Cloudformation diagrams:

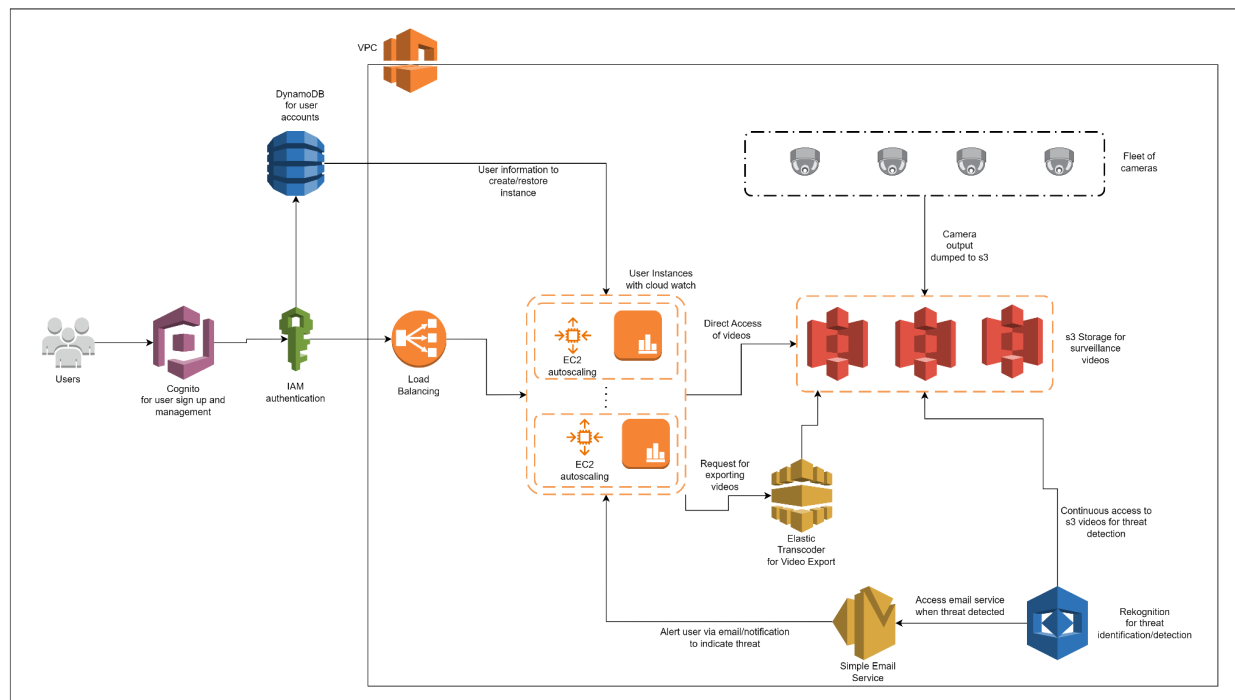


Figure 2. Final Architectural diagram of Cloud Surveillance application

5.4. Validation of the design:

1. TR2.4 Identify a tenant and authorize access before performing any operation (Tenant Identification)

Identity and Access Management's (IAM) granular access control allows the management of user access to AWS resources, ensuring specific permissions are assigned to identify and control tenant access. By employing role-based access policies, IAM ensures that only authorized individuals, based on their roles within the cloud based surveillance application, can perform operations. Upon successful authorization using the IAM credentials, user settings and logs can be accessed from DynamoDB to setup EC2 based user instances to allow the user to perform operations such as accessing and

downloading video footage. Additionally, IAM's support for multi-factor authentication (MFA) adds an extra layer of security, verifying the identity of tenants before granting access to AWS resources.

2. TR7.2 User activity and access logs (Monitoring)

With CloudWatch Logs, the system can continuously monitor and track user interactions and activities within the security system, providing a comprehensive record of events for auditing and analysis purposes. These logs can include information such as user login attempts, system configuration changes, and resource access requests. This capability enables the project to maintain a detailed history of user actions, helping to identify any unauthorized access attempts or suspicious activities. By leveraging the features of Amazon CloudWatch Logs, the Cloud-Based Camera Security System can enhance its overall security posture, promote accountability, and enable proactive monitoring and management of user activities and system access.

3. TR4.3 Email Alerts for Intruder Detection (Monitoring)

Amazon Rekognition's advanced image and video analysis capabilities enable the system to identify and analyze any suspicious activities or unauthorized access attempts in real-time, triggering immediate alerts for potential intruder detection. These alerts prompt the system to generate and send email notifications to designated recipients through Amazon SES, ensuring that relevant stakeholders, including property owners and authorities, are promptly informed of any security incidents. By combining the powerful features of Amazon Rekognition for automated intrusion detection and Amazon SES for email alert notifications, the Cloud-Based Camera Security System can effectively enhance its security monitoring capabilities, enabling timely responses and proactive measures to mitigate potential security threats.

4. TR3.1 Support for Video Exports in Various Formats (Elasticity)

Amazon Elastic Transcoder service efficiently converts video files into multiple formats, resolutions, and bitrates, allowing users to download and access security footage in their preferred file formats. By leveraging the comprehensive transcoding features of Amazon Elastic Transcoder, the Cloud-Based Camera Security System ensures broad accessibility and compatibility of exported video content, enhancing user experience and convenience, particularly for offline viewing and storage of security footage across a wide range of playback devices and applications.

5. TR10.1 Restrict resource usage to below appropriate limits to prevent cost surges (Elasticity)

AWS Budgets and Cost explorer enable the system to set custom budgets and allocate predefined spending thresholds for different resource usage categories, ensuring that resource consumption remains within predefined limits to avoid unexpected cost overruns. Additionally, Cost Explorer provides comprehensive visibility into the system's resource usage patterns and cost trends, allowing the project to monitor and analyze resource consumption in real time. By leveraging these services, the Cloud-Based Camera Security System effectively manages and controls resource utilization, preventing excessive spending and enabling proactive cost management strategies to optimize operational efficiency and maintain cost-effectiveness.

6. TR7.1 Account Recovery Options (Reliability TR from WAF)

Amazon Cognito allows the system to enable various user verification methods, such as email and phone number verification, ensuring that users can regain access to their accounts securely in the event of password loss or account lockouts. Additionally, Amazon Cognito's integration with multi-factor authentication (MFA) enhances the security of the account recovery process, reducing the risk of unauthorized access.

7. TR1.1 The system should be available 99% (Reliability TR from WAF and conflicting TR10.1)

Amazon EC2 Auto Scaling dynamically adjusts the system's capacity in response to fluctuations in traffic and demand, automatically adding or removing instances to maintain consistent performance levels even during peak usage periods. This feature prevents resource bottlenecks and overload situations, ensuring that the system can efficiently handle varying workloads and traffic spikes without compromising its availability. Additionally, Amazon Elastic Load Balancing (ELB) evenly distributes incoming application traffic across multiple instances, preventing any single server from becoming overwhelmed and minimizing the risk of downtime due to server failures or traffic surges. Although these services contribute significantly to minimizing downtime and ensuring continuous system operation, a 99% availability target is set to account for these inevitable maintenance activities and potential unforeseen events, allowing for necessary system upkeep and ensuring a reliable and consistent user experience while mitigating the impact of any unforeseen downtime on system operations.

8. TR2.1 Cloud storage for video archives.

Amazon S3 offers a highly scalable and secure storage solution that enables the system to store and retrieve large volumes of video data reliably and efficiently. By leveraging

Amazon S3's durable and cost-effective storage capabilities, the system ensures the safe and long-term retention of video archives, providing users with seamless access to historical footage for review, analysis, and retrieval as needed.

5.5. Design principles and best practices used:

For this architectural design, several design principles and best practices have been applied to ensure a robust and effective system. These principles are derived from the AWS Well-Architected Framework and the cloud-native architect best practices:

- **Design for Automation:** In line with the "Perform operations as code" principle, automation is a core aspect of the design. Operations such as load balancing, auto-scaling, and intruder detection are automated to ensure high consistency, faster reaction times, and low error rates.
- **Scalability:** The design emphasizes scalability, as indicated by the "Scale horizontally to increase aggregate workload availability" principle. It uses multiple smaller resources and load distribution to handle increasing workloads, enhancing availability.
- **Optimize for Cost:** While not explicitly mentioned, the design is cost-efficient by leveraging AWS services and scaling resources as needed. Further optimization opportunities may include reserved instances or spot instances.
- **Security:** The design follows the "Protect data in transit and at rest" principle by utilizing AWS services that provide built-in encryption for data and network traffic. This ensures data security, a crucial aspect of camera security monitoring.
- **Performance:** The design aligns with the "Stop guessing capacity" principle by using metrics to accurately judge resource requirements, providing high-quality customer experiences. This is particularly important for intruder detection and video analysis.
- **Design for Automation:** Automation is at the core of the design, ensuring efficient operations and responsiveness.
- **Favor Managed Services:** The design heavily relies on AWS managed services, such as DynamoDB, S3, Rekognition, and Cognito, demonstrating a preference for managed solutions.
- **Practice Defense in Depth:** The combination of authentication, authorization, intruder detection, and data encryption reinforces the security posture of the system, following the principle of defense in depth.
- **Always Be Architecting:** The design reflects a continuous improvement approach, with potential areas for further optimization, such as cost-efficiency and caching.

In summary, the design for the camera security monitoring application is based on a solid foundation of cloud architecture principles and best practices. These principles ensure that the system is automated, scalable, cost-effective, secure, and high-performing. However, it's

essential to consider the specific trade-offs and unique project requirements when implementing these principles in practice.

5.6. Tradeoffs revisited:

Trade-off Between Resource Restriction and System Availability:

Conflicting Technical Requirements:

- TR10.1: Restrict resource usage to below appropriate limits to prevent cost surges (Elasticity)
- TR1.1: The system should be available 99% (Reliability)

Decision:

The chosen optimization is in favor of ensuring 99% system availability (TR1.1) over strictly restricting resource usage (TR10.1).

Reasoning:

- Prioritizing system availability is critical for a security monitoring application, where continuous uptime is essential for effective surveillance and intruder detection.
- Over-provisioning resources, despite potential cost implications, ensures that the system can handle peak loads and maintains reliability.
- Amazon EC2 Auto Scaling and Elastic Load Balancing are employed to dynamically scale resources and distribute traffic, optimizing for high availability.

Explanation:

Key Considerations:

- The primary focus is on meeting the reliability requirement of 99% system availability, aligning with the critical nature of security monitoring.
- Acknowledging that strict resource restrictions might lead to occasional downtime, the decision is made to prioritize user experience and reliability over potential cost savings.
- The trade-off leans towards the side of over-provisioning resources to handle peak demand efficiently and prevent disruptions in surveillance operations.

Mitigations and Monitoring:

- Robust monitoring and alerting systems are implemented to identify resource spikes and potential cost overruns promptly.
- Auto Scaling policies are configured dynamically to adapt to varying workloads, ensuring efficient resource utilization while maintaining high availability.
- Continuous cost-benefit analysis is conducted to optimize resource limits and scaling policies based on real-time demands and budget considerations.

Overall Impact:

- While the trade-off may lead to increased infrastructure costs, it ensures a reliable and resilient system that meets the high availability requirements of a camera security monitoring application.
- This trade-off decision reflects a strategic choice to prioritize system reliability and user experience over potential cost savings, emphasizing the critical importance of continuous surveillance and intrusion detection in the context of security monitoring.

5.7. Discussion of an alternate design:

Skipped

* ChatGPT was used in this section to rephrase the sentences and check for grammatical errors.

6. KUBERNETES EXPERIMENTATION

6.1 Experiment Design

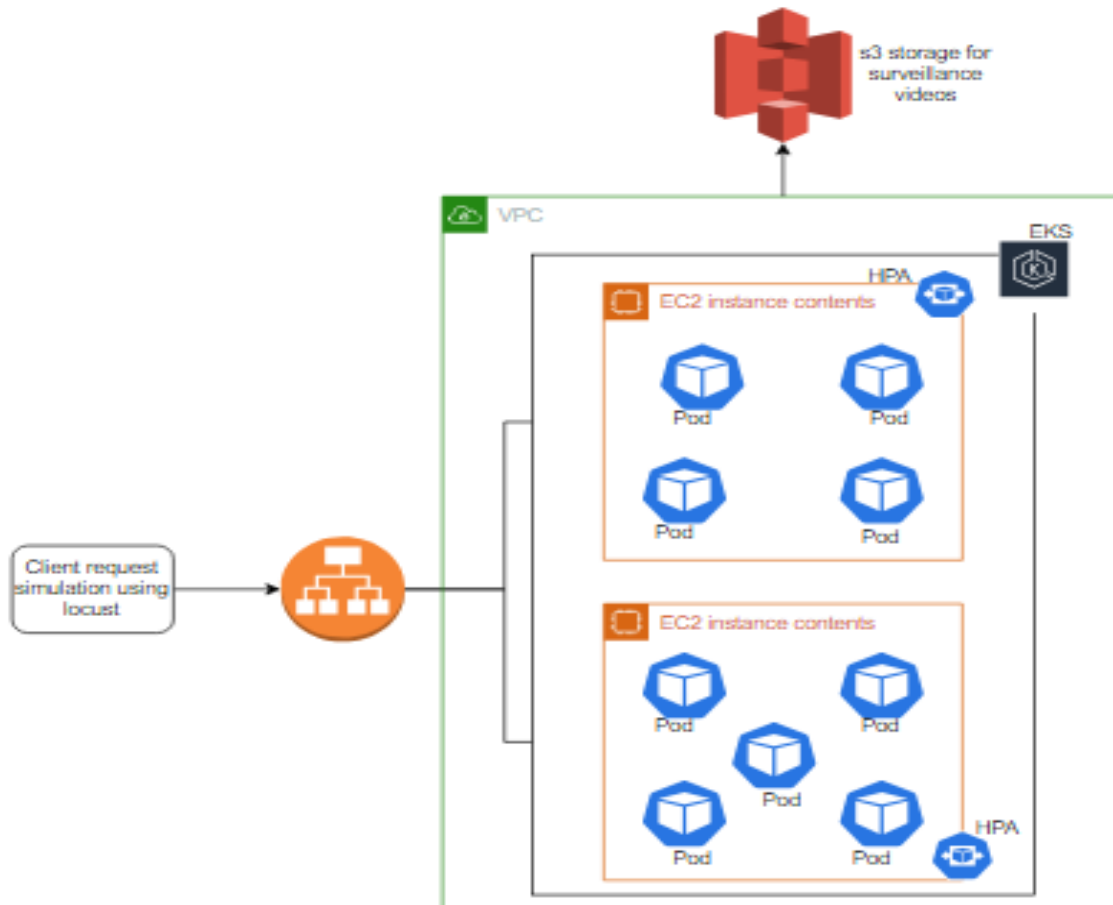


Figure 3. Experiment pipeline design

1. Target Technical Requirement:

TR1.1 The system should be available 99%. In this section, we are simulating one of the microservices of our application called Surveillance Log. This microservice allows the user to retrieve the 20 most recent surveillance videos from s3 storage. We demonstrate how the system availability to the users can be maintained at at least 99% using services from the cloud provider Amazon.

2. Experiment pipeline:

- We simulate user requests using the locust tool. When the user requests for the previous surveillance logs, the request is sent through Amazon Application Load Balancer (ALB).

- Based on the availability of the compute nodes in the EKS cluster, the request is directed to one of the pods in one of the EC2 instances. This depends on the criteria defined for the load balancer.
- The pods to which the request is directed has the docker image of the application to retrieve the surveillance logs running.
- The retrieved logs is sent back to the user completing the processing of the request.

3. Experiment setup:

The list of services we used in this project:

Amazon VPCs (Virtual Private Clouds):

- a. Created VPCs to isolate and organize the network resources.

Amazon EKS Cluster:

- b. Established an Amazon Elastic Kubernetes Service (EKS) cluster to orchestrate and manage containerized applications.

Node Group:

- c. Created a node group within the EKS cluster with nodes inside it.
- d. These nodes serve as the underlying compute resources for running containers.

Pods and Deployments:

- e. Deployed the Surveillance Log microservice as containers within pods.
- f. Used a Kubernetes Deployment YAML file to define and manage the application's deployment, specifying the desired state.

Amazon Application Load Balancer (ALB):

- g. Set up an Application Load Balancer to distribute incoming user requests among the pods in the EKS cluster.
- h. Configured the ALB to route traffic based on specified criteria, ensuring even distribution.

Horizontal Pod Autoscaler (HPA):

- i. Configured Horizontal Pod Autoscaling to dynamically adjust the number of pods in response to changes in demand.
- j. Set up criteria, such as CPU utilization or other metrics, to trigger autoscaling.

Overview of the sequence of experimental execution is as follows:

Locust Tool:

- Utilized the Locust tool to simulate user requests.
- Simulated users accessing the Surveillance Log microservice by sending requests through the Amazon ALB.

Load Balancer Routing:

- Observed how the ALB directs user requests based on the defined criteria to one of the pods within the EKS cluster.

Autoscaling with HPA:

- Sent varying levels of load through Locust to the ALB to simulate different levels of user traffic.
- Monitored the behavior of the HPA, observing how it dynamically adjusts the number of pods based on the configured metrics.


4. Expectations:

- When the number of user requests are high, a single pod would not be sufficient to keep the response time low.
- When the user request comes in and the capacity of the existing pods is beyond a particular threshold, we expect the autoscaler to increase the number of pods in the node.
- We also expect the load balancer to assign the incoming request to the right pod. The right pod here would refer to the one which is relatively free compared to the other pods.
- When the user request rate is low and the usage of the pods is below a particular threshold, we expect the autoscaler to reduce the number of pods in the node.
- Based on these features, we expect the system to be highly available to process and respond to the user requests.

6.2 Workload generation with Locust

Locust is an open-source Python-based load testing tool designed for simulating large numbers of users accessing a web application concurrently. It enables developers and testers to define user behavior through Python code, allowing the simulation of diverse interactions with web servers, APIs, or other systems. Users create scenarios by specifying tasks, such as making HTTP requests, and then employ Locust to generate heavy loads on the target, assessing how well it performs under stress. The tool is valuable for identifying performance bottlenecks, analyzing response times, and ensuring that applications can handle high levels of concurrent users, ultimately aiding in the optimization and robustness of web-based services.

Locust configuration for simulating user requests:


LOCUST

HOST
STATUS
READY
0 users

Start new load test

Number of users (peak concurrency)

Spawn rate (users started/second)

Host (e.g. http://www.example.com)

Advanced options

The host address to which the user requests were to be simulated was the public DNS of the load balancer of our application:

We set the maximum number of users, that is, the maximum number of requests that the application can face concurrently to 50000 and we simulate the number of user requests started per second (spawn rate) to be 250. This simulation of user requests will help test our application for its reliability.

6.3 Analysis of the results

```

C:\Users\devad>kubectll get nodes
NAME                                STATUS  ROLES  AGE    VERSION
ip-172-31-40-124.ec2.internal      Ready  <none> 3h12m  v1.28.3-eks-e71965b

C:\Users\devad>kubectll get pods
No resources found in default namespace.

C:\Users\devad>kubectll get services
NAME      TYPE        CLUSTER-IP  EXTERNAL-IP  PORT(S)  AGE
kubernetes ClusterIP  10.100.0.1  <none>       443/TCP    3h57m

C:\Users\devad>kubectll get hpa
No resources found in default namespace.

C:\Users\devad>kubectll get deployments
No resources found in default namespace.

```

As described in Section 6.1, we create a node group within the EKS cluster. As it can be seen from this figure, there is one EC2 instance in the cluster and there are no pods or deployments initially.

```
C:\Users\devad>kubectl apply -f D:\sem3\ece547\project\yaml_files\simple_deployment\deployment_creation.yaml
deployment.apps/surveillance-log-deployment created

C:\Users\devad>kubectl get deployments
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
surveillance-log-deployment  1/1     1             1           8s

C:\Users\devad>kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
surveillance-log-deployment-d85c77b5d-vtgjt  1/1     Running   0           24s

C:\Users\devad>kubectl get services
NAME                TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes           ClusterIP   10.100.0.1    <none>         443/TCP    4h

C:\Users\devad>kubectl apply -f D:\sem3\ece547\project\yaml_files\simple_deployment\loadbalancer_creation.yaml
service/surveillance-log-loadbalancer created

C:\Users\devad>kubectl get services
NAME                TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes           ClusterIP   10.100.0.1    <none>         443/TCP    4h1m
surveillance-log-loadbalancer  LoadBalancer  10.100.2.102  af4746381443b4e05bdecdf75f008a95a-1201598351.us-east-1.elb.amazonaws.com  80:30928/TCP  13s

C:\Users\devad>kubectl get hpa
No resources found in default namespace.

C:\Users\devad>kubectl autoscale deployment surveillance-log-deployment --cpu-percent=50 --min=1 --max=6
horizontalpodautoscaler.autoscaling/surveillance-log-deployment autoscaled
```

We then deploy our application on one pod. We also launch a load balancer to be associated with the pods in the node. We also create an autoscaler which is set to scale up or scale down the number of pods between 1 to 6 based on the load on the current pods. If the load on the existing pods goes beyond 10 percent, additional pods are added, if the load on the existing pods is below 10 percent, the number of pods are reduced.

```
C:\Users\devad>kubectl get hpa
NAME                REFERENCE                TARGETS  MINPODS  MAXPODS  REPLICAS  AGE
surveillance-log-deployment  Deployment/surveillance-log-deployment  0%/10%   1         6         1          2m39s

C:\Users\devad>kubectl get hpa surveillance-log-deployment --watch
NAME                REFERENCE                TARGETS  MINPODS  MAXPODS  REPLICAS  AGE
surveillance-log-deployment  Deployment/surveillance-log-deployment  0%/10%   1         6         1          2m46s
surveillance-log-deployment  Deployment/surveillance-log-deployment  16%/10%   1         6         1           4m
surveillance-log-deployment  Deployment/surveillance-log-deployment  24%/10%   1         6         2          4m16s
surveillance-log-deployment  Deployment/surveillance-log-deployment  15%/10%   1         6         3          4m31s
surveillance-log-deployment  Deployment/surveillance-log-deployment  9%/10%    1         6         3          4m46s
surveillance-log-deployment  Deployment/surveillance-log-deployment  8%/10%    1         6         3           5m1s
surveillance-log-deployment  Deployment/surveillance-log-deployment  8%/10%    1         6         3          5m16s
surveillance-log-deployment  Deployment/surveillance-log-deployment  8%/10%    1         6         3          5m31s
```

As mentioned above, in our configuration we start with one pod. When the load on the pod goes beyond 10 percent of its total capacity, an additional pod (with the same application running) is created to handle the additional load. When the load on these two pods goes beyond the 10 percent threshold, another pod is added. Once the number of pods reaches 3, we can see that the load on the pods is stabilized at 8 percent.

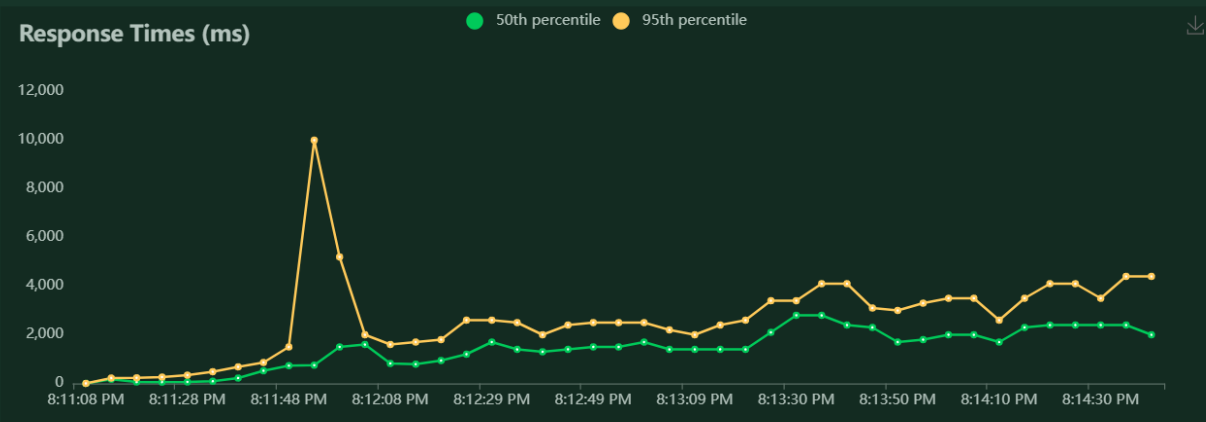
```
C:\Users\devad>kubectl get hpa surveillance-log-deployment --watch
NAME                REFERENCE                TARGETS  MINPODS  MAXPODS  REPLICAS  AGE
surveillance-log-deployment  Deployment/surveillance-log-deployment  8%/10%   1         6         4          26m
surveillance-log-deployment  Deployment/surveillance-log-deployment  9%/10%   1         6         4          26m
surveillance-log-deployment  Deployment/surveillance-log-deployment  6%/10%   1         6         4          27m
surveillance-log-deployment  Deployment/surveillance-log-deployment  1%/10%   1         6         4          27m
surveillance-log-deployment  Deployment/surveillance-log-deployment  0%/10%   1         6         4          27m
surveillance-log-deployment  Deployment/surveillance-log-deployment  0%/10%   1         6         4          27m
surveillance-log-deployment  Deployment/surveillance-log-deployment  0%/10%   1         6         4          28m
surveillance-log-deployment  Deployment/surveillance-log-deployment  0%/10%   1         6         4          28m
surveillance-log-deployment  Deployment/surveillance-log-deployment  0%/10%   1         6         4          29m
surveillance-log-deployment  Deployment/surveillance-log-deployment  0%/10%   1         6         4          31m
surveillance-log-deployment  Deployment/surveillance-log-deployment  0%/10%   1         6         3          32m
surveillance-log-deployment  Deployment/surveillance-log-deployment  0%/10%   1         6         1          32m
surveillance-log-deployment  Deployment/surveillance-log-deployment  0%/10%   1         6         1          32m
```

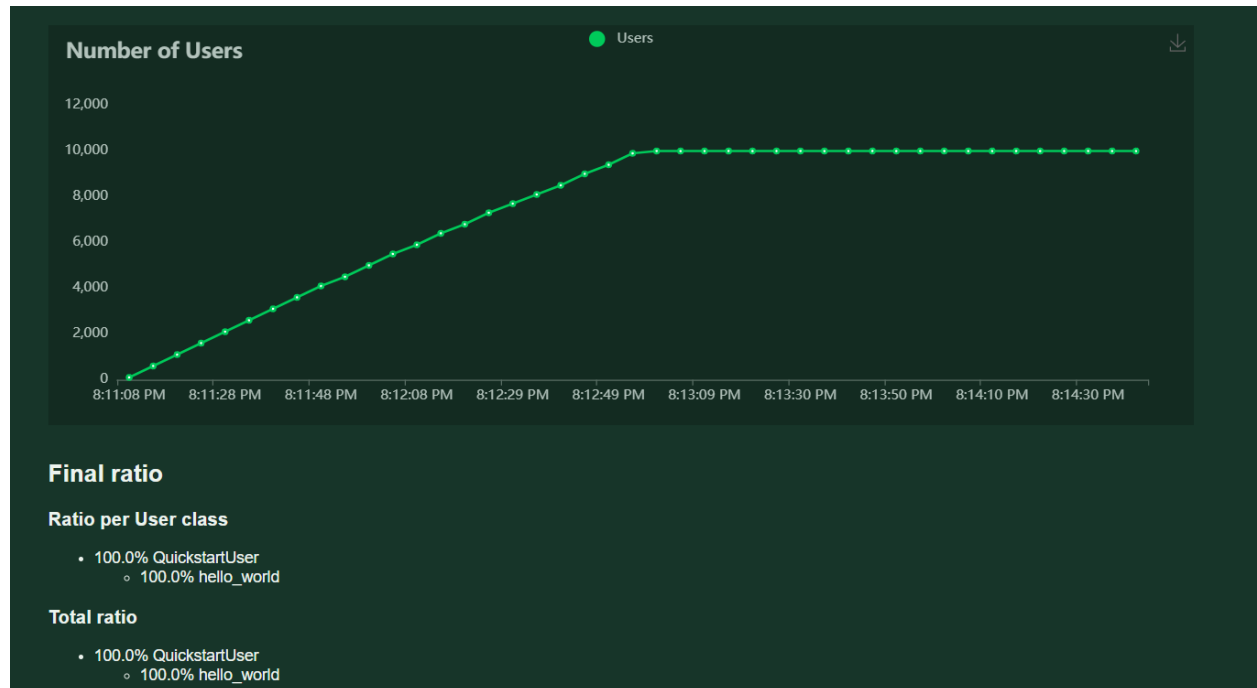
When the load on the pods reduces, we can see that the number of pods is reduced and maintained at 1 when there is no load since 1 is the minimum number of pods in our configuration. We can see that the use of the load balancer and autoscaler, automates the successful handling of user requests.



This figure indicates the total number of requests processed by the application. It also indicates the number of failures. Overall the number of failures per second is at 0.8/s considering that the application has a response per second rate of 711.8 per second. This indicates that the combination of load balancer and autoscaler is able to successfully add additional pods and route the requests to the optimal pod to ensure high availability of the application.

Charts





We can see from the above two figures that as the number of requests per second is scaled up, the response time is mostly maintained at a constant rate. This is possible due to the functionalities of the load balancer and the autoscaler.

* ChatGPT was used in this section to rephrase the sentences and check for grammatical errors.

7. ANSIBLE PLAYBOOKS

Skipped

8. DEMONSTRATION

Skipped

9. COMPARISONS

Skipped

10. CONCLUSION:

In conclusion, the experimental setup successfully demonstrated the robustness of the Surveillance Log microservice within the Amazon cloud environment. By employing Amazon EKS for container orchestration, an Application Load Balancer for efficient traffic distribution, and Horizontal Pod Autoscaling for dynamic resource adjustment, the system showcased its ability to maintain a high availability rate of at least 99%. The Locust tool effectively simulated user requests, allowing for the observation of the system's behavior under different loads. The documented configurations, deployment files, and observed performance metrics provide valuable insights into the system's resilience and scalability, affirming the reliability of the chosen cloud services for ensuring optimal performance and meeting technical requirements.

* ChatGPT was used in this section to rephrase the sentences and check for grammatical errors.

11. REFERENCES:

- [1] Amazon Well-Architected Framework Pillars, [Amazon Well-Architected Framework](#)
- [2] AWS Identity and Access Management (IAM), [AWS IAM Documentation](#)
- [3] Amazon CloudWatch Logs, [Amazon CloudWatch Logs Documentation](#)
- [4] Amazon Rekognition, [Amazon Rekognition Documentation](#)
- [5] Amazon SES (Simple Email Service), [Amazon SES Documentation](#)
- [6] Amazon Elastic Transcoder, [Amazon Elastic Transcoder Documentation](#)
- [7] AWS Budgets and Cost Explorer, [AWS Budgets Documentation](#), [AWS Cost Explorer Documentation](#)
- [8] Amazon Cognito, [Amazon Cognito Documentation](#)
- [9] Amazon EC2 Auto Scaling and Amazon Elastic Load Balancing (ELB), [Amazon EC2 Auto Scaling Documentation](#), [Amazon Elastic Load Balancing Documentation](#)
- [10] Amazon S3, [Amazon S3 Documentation](#)