

SEVEN WEEKS SUMMER TRAINING REPORT

**on**

**AWS Cloud Computing training**

Submitted by

**Sudhanshu Ranjan Gupta**

**Registration No: 12215256**

**Program Name: Btech. CSE (3rd Year)**

Under the Guidance of:

**Muhammad Samir Akhtar and Mr. Yash Raj**

**Gokboru Tech Pvt. Ltd.,**

**Araria, Bihar (India)**

(June-July,2024)

DECLARATION

I hereby declare that I have completed my seven weeks summer training at Gokboru Pvt. Ltd. from June 02,2024 to July 14, 2024, under the guidance of Muhammad Samir Akhtar and Mr. Yash Raj. I declare that I have worked full dedication during their 7 weeks of training and my learning outcomes fulfill the requirements of training for the award of degree of B.tech CSE, Lovely Professional University, Phagwara.

Date –15 August. 2024 Name of Student –

Sudhanshu Ranjan Gupta

Registration no: 12215256

ACKNOWLEDGEMENT

I would like to express my gratitude towards my university as well as Gokboru Tech Pvt. Ltd. for providing me the golden opportunity to do this wonderful summer training regarding AWS Cloud computing, which also helped me in doing a lot of homework and learning. As a result, I came to know about so many new things. So, I am thankful to them.

Moreover, I would like to thank my friends who helped me a lot whenever I got stuck in some problem related to my course. I am thankful to have such a good support of them as they always have my back whenever I need.

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

Deepest thanks to our Trainer **Muhammad Samir Akhtar (Mentor) and Yash Raj (Trainer for AWS)** for their guidance, monitoring, constant encouragement and correcting various assignments of ours with attention and care. They have taken pain to go through the project and training sessions and make necessary corrections as when needed and we are very grateful for that.

**Summer Training Certificate by Gokboru Tech**

**A close-up of a certificate

Description automatically generated**

**Table of Contents**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No.** | **Title** | | **Page No.** |
| 1 | Introduction | | 06 |
| 2 | Technology Learnt (Weekly Schedule) | |  | | --- | | Week 1 (Page No.: 7-9) | | Week 2 (Page No.: 9-10) | | Week 3 (Page No.: 10-11) | | Week 4 (Page No.: 11) | | Week 5 (Page No.: 12-13) | | Week 6 (Page No.: 14-17) | | 07 - 17 |
| 3 | Reason for choosing AWS Cloud Computing | | 18-20 |
| 4 | Project: | | 21-39 |
| 5 | Real Life Applications of Real-Time Data Processing Pipeline with Kinesis. | | 40-42 |
| 5 | Conclusion | | 43 |
| 6 | Bibliography | | 44 |

INTRODUCTION

Cloud computing is a technology that allows users to access computing resources like servers, storage, databases, networking, software, and analytics over the internet, or “the cloud.” Instead of owning and maintaining physical hardware and software, businesses and individuals can use these resources on-demand from cloud service providers. This approach offers flexibility, scalability, and cost-efficiency. Users can scale up or down depending on their needs, pay only for what they use, and avoid the complexities of managing physical infrastructure.

TECHNOLOGY LEARNT

**It had 16 units which was further divided into chapters and then topics so during my whole 6-week course I learned the following:**

**1st Week:**

1. **Introduction to Cloud Computing:**

Cloud computing is a technology that provides access to computing resources such as servers, storage, and software over the internet. Instead of relying on personal computers or local servers, these resources are utilized remotely. Services are available as needed, with the ability to scale up or down, and payment is made based on usage. This approach offers a flexible and cost-effective solution for managing and processing data.

1. **Definition, Characteristics, Benefits:**

***Definition:*** Cloud computing refers to the delivery of computing services—including servers, storage, databases, networking, software, and analytics—over the internet (“the cloud”). These services enable users to access and manage resources without the need for direct active management by the user.

***Characteristics:***

*On-Demand Self-Service:* Computing resources can be provisioned automatically, as needed, without requiring human intervention.

*Broad Network Access:* Services are accessible over the internet from various devices, such as computers, tablets, and smartphones.

Resource Pooling: Computing resources are pooled to serve multiple users, with resources dynamically assigned and reassigned based on demand.

Rapid Elasticity: Resources can be scaled up or down quickly to meet changing demands, appearing unlimited to users.

Measured Service: Resource usage is monitored and reported, allowing for pay-as-you-go billing based on actual consumption.

***Benefits:***

*Cost Efficiency:* Reduces the need for significant capital investment in hardware and software, as resources are provided on a pay-per-use basis.

*Scalability:* Easily scales resources up or down to handle varying workloads, ensuring that users only pay for what they use.

*Accessibility:* Enables access to applications and data from anywhere with an internet connection, facilitating remote work and collaboration.

*Reliability:* Offers enhanced reliability through data backup, disaster recovery, and redundancy, often exceeding the capabilities of on-premises solutions.

*Flexibility:* Provides the flexibility to choose from a wide range of services and customize them to fit specific business needs.

1. **Cloud Service Models:**

There are 3 types of Cloud Service Models:

***i.*** *Infrastructure as a Service (IaaS):*Provides virtualized computing resources like servers, storage, and networking over the cloud, allowing users to manage operating systems and applications.

***ii*.** *Platform as a Service (PaaS):*Offers a cloud-based platform for developing, testing, and deploying applications, without managing the underlying infrastructure.

***iii.*** *Software as a Service (SaaS):*Delivers software applications over the internet, accessible via a web browser, without requiring installation or management by the user.

1. **Cloud Deployment Models:**
2. There are 4 types of deployment model:

* *Private:* Exclusively used by single organizations, offering greater control and security. Example: VMware vSphere
* *Public:* Resources are owned and managed by a third-party provider, accessible to multiple users. Example: Amazon Web Services (AWS)
* *Hybrid:* Combines public and private clouds, enabling data and application portability between them. Example: Microsoft Azure
* *Community:*Shared infrastructure among organizations with common goals, ensuring collaboration and resource sharing. Example: OpenNebula for Research and Education.

1. **Introduction to Linux:**

* *Linux Operating System Basics and History of Linux.*
* *Basic Linux Commands, Navigation and file management system.*
* *Networking and Internet tools and networking with linux.*
* *Functions and variables, SSH basics.*

**2nd Week:**

1. **Getting Started With AWS:**

* *Creating an AWS account.*
* *Navigating through AWS console.*
* *Overview of AWS services and platforms.*
* *AWS compute services (EC2, Lambda, ECS)*
* *AWS Storage services (S3, EBS)*
* *Understanding security groups and data stream pipelines.*
* *Understanding lambda, triggers, load balancing and auto-scaling.*

**3rd Week:**

1. **AWS Security:**

* *AWS Identity and Access Management (IAM):* It is a framework used to ensure that the right individuals have the appropriate access to technology resources. In cloud computing, IAM is a crucial security feature that helps manage users, permissions and roles within an organization.
* *IAM Users and Group*:

**IAM Users***:*In AWS, an IAM (Identity and Access Management) user is an entity that represents a person or service that interacts with AWS. Each IAM user has unique credentials, such as a username and password, and can have permissions assigned to control what actions they can perform within AWS.

**IAM Groups:**  
An IAM group is a collection of IAM users that share common permissions. By adding users to a group, administrators can manage permissions for multiple users at once, simplifying access management. Groups are typically used to apply consistent policies and access rights to users with similar roles.

* *Multi-Factor Authentication (MFA):* additional layer of security used in AWS to enhance the protection of user accounts and sensitive resources. With MFA enabled, users are required to provide not just their password but also a second form of authentication, such as a one-time code generated by a hardware or software token, when accessing AWS services. This helps ensure that even if a password is compromised, unauthorized access is still prevented.
* *Best Practices for encryption.*
* *AWS Resource Monitoring and Logging*: essential practices for managing and maintaining the performance, availability, and security of your AWS infrastructure.
* ***Monitoring:*** AWS offers services like Amazon CloudWatch to track metrics, set alarms, and visualize logs for various AWS resources, helping detect performance issues and respond to operational changes.
* ***Logging:*** AWS provides logging services such as AWS CloudTrail and Amazon CloudWatch Logs to capture detailed event logs, track user activities, and ensure compliance by keeping an audit trail of all actions taken on your AWS resources.

**4th Week:**

1. **Networking on AWS:** Networking on AWS involves configuring and managing virtual networks that enable the communication between different AWS resources and with external networks.

* *Amazon VPC (Virtual Private Cloud):* Allows you to create isolated networks within AWS, where you can launch resources like EC2 instances. It provides control over IP addresses, subnets, route tables, and gateways.
* *Security Groups and NACLs:* Used to control inbound and outbound traffic to and from resources within a VPC, offering a layer of security.
* *Elastic Load Balancing (ELB):* Distributes incoming traffic across multiple targets, such as EC2 instances, in one or more Availability Zones to ensure high availability.
* *AWS Direct Connect:* Establishes a dedicated network connection from your premises to AWS, providing more consistent network performance.
* **Route 53:** AWS’s scalable DNS and domain name management service, used to route end-user requests to AWS-hosted applications.

1. **AWS Elastic Load Balancing (ELB):** AWS Elastic Load Balancing (ELB) is a service that automatically distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, containers, and IP addresses, across one or more Availability Zones. This helps ensure the high availability and fault tolerance of applications by spreading the load, preventing any single resource from becoming a bottleneck.

* Key features include:
* *Automatic Scaling:* ELB automatically adjusts the load balancing capacity to handle incoming traffic as it changes over time.
* *Health Checks:* ELB performs regular health checks on registered targets and only routes traffic to healthy instances.
* *Security:* ELB supports secure applications by allowing the use of HTTPS and SSL/TLS.
* Types of Load Balancers:
* *Application Load Balancer (ALB):* Best suited for HTTP/HTTPS traffic, operating at the application layer (Layer 7).
* *Network Load Balancer (NLB):* Designed for ultra-high performance and low latency, operating at the transport layer (Layer 4).
* *Gateway Load Balancer:* Distributes traffic across multiple virtual appliances while scaling them automatically.
* Use Case: ELB is essential for applications that require high availability and redundancy, ensuring continuous operation even if individual servers fail.

**5th Week:**

1. **Containerization and Orchestration on AWS:**

**Containerization** is a lightweight form of virtualization that involves packaging an application and its dependencies into a container. This ensures the application runs consistently across different computing environments. Containers are isolated from each other and the underlying infrastructure, making them highly portable and efficient.

**Orchestration** involves managing and automating the deployment, scaling, and operation of containerized applications. On AWS, several services support containerization and orchestration, enabling developers to run and manage containers effectively.

***Key AWS Services for Containerization and Orchestration: -***

*Amazon Elastic Container Service (ECS):*

*Overview:* ECS is a fully managed container orchestration service that allows you to run, stop, and manage Docker containers on a cluster of Amazon EC2 instances.

*Features:*

*Task Definitions:* Define how containers should be run, including the Docker images, CPU, memory requirements, and network settings.

*Cluster Management:* Automatically manages the cluster of EC2 instances, ensuring optimal resource utilization.

*Integration with AWS Services*: Seamlessly integrates with other AWS services like IAM, CloudWatch, and VPC for enhanced security and monitoring.

*Amazon Elastic Kubernetes Service (EKS):*

*Overview:* EKS is a fully managed Kubernetes service that makes it easy to run Kubernetes on AWS without needing to install, operate, and maintain your Kubernetes control plane.

*Features:*

*Kubernetes Compatibility*: EKS runs upstream Kubernetes, ensuring compatibility with existing Kubernetes applications.

*Managed Control Plane:* AWS manages the Kubernetes control plane, ensuring high availability and security.

*Scalability:* Automatically scales the Kubernetes clusters based on demand.

*AWS Fargate:*

*Overview*: AWS Fargate is a serverless compute engine for containers that works with both ECS and EKS. It allows you to run containers without managing the underlying EC2 instances.

*Features:*

*Serverless Operation*: Automatically provisions and scales the infrastructure required to run your containers, eliminating the need to manage servers.

*Pay-as-You-Go:* Only pay for the compute resources used by your containers, with no upfront costs or commitments.

*Security:* Provides enhanced security through isolated runtime environments for each container.

*Amazon ECR (Elastic Container Registry):*

*Overview*: ECR is a fully managed Docker container registry that makes it easy to store, manage, and deploy Docker container images.

*Features:*

*Image Storage:* Securely store container images in a scalable repository.

Integration with CI/CD: Integrates with AWS CodePipeline, Jenkins, and other CI/CD tools for streamlined development and deployment workflows.

*Security:* Offers encryption, access control, and integration with IAM for secure image management.

*Use Cases:*

*Microservices Architecture:* Easily deploy and manage microservices using containers, with orchestration ensuring seamless scaling and availability.

*DevOps Pipelines:* Integrate containerization into CI/CD pipelines for faster development and deployment cycles.

*Hybrid Cloud Environments:* Run containers across on-premises and cloud environments, with Kubernetes providing a consistent orchestration layer.

*Benefits:*

*Portability:* Containers can run consistently across different environments, making it easier to develop and deploy applications.

*Efficiency:* Containers are lightweight and consume fewer resources compared to traditional virtual machines.

*Scalability:* Orchestration tools automatically scale containerized applications based on demand, ensuring optimal performance.

**6th Week:**

1. **Multi-Cloud Models and Providers:**

Multi-Cloud is an IT strategy that involves utilizing multiple cloud computing services from different providers. This approach allows organizations to leverage the unique strengths of each provider, avoid vendor lock-in, and enhance redundancy for critical systems.

**Overview of Multi-Cloud Benefits**

***Vendor Independence:***

* + *Flexibility:* Utilizing multiple cloud providers prevents dependency on a single vendor, offering greater freedom in pricing, service offerings, and technology choices.
  + *Negotiation Power:* Multi-cloud strategies provide leverage during negotiations, as organizations are not committed to a single provider.

***Enhanced Resilience and Redundancy:***

* + *High Availability*: Workloads distributed across multiple cloud providers ensure continuous availability, mitigating the risk of a single provider’s service failure affecting overall application performance.
  + *Disaster Recovery*: Multi-cloud environments offer a robust solution for disaster recovery, ensuring business continuity in the event of a major outage with one provider.

***Best-of-Breed Services:***

* + *Optimized Performance*: Different cloud providers specialize in various areas (e.g., AI/ML, analytics, networking), allowing organizations to select the best services for specific needs.
  + *Innovation*: Leveraging unique offerings from multiple providers helps organizations stay ahead of technological advancements.

***Regulatory Compliance:***

* + *Data Sovereignty:* Multi-cloud strategies enable compliance with regional data residency regulations by selecting providers that meet specific compliance standards.

**13. Multi-Cloud Deployment Models:**

***Poly Cloud:***

* + *Definition:* The use of multiple cloud providers, each chosen for its strengths in specific areas or services.
  + *Example:* Utilizing AWS for infrastructure, Google Cloud for machine learning, and Azure for enterprise integrations.

***Hybrid multi-cloud:***

* + *Definition:* Integration of multiple public cloud providers with on-premises data centers or private clouds.
  + *Example:* Combining AWS for general workloads, Azure for compliance-sensitive applications, and an on-premises data center for legacy systems.

***Distributed Cloud:***

* + *Definition:* Cloud services distributed across different locations (e.g., on-premises, edge, public cloud), managed as a single entity.
  + *Example:* Using Google Cloud Anthos to manage workloads across on-premises environments and multiple clouds from a unified platform.

**14. Key Multi-Cloud Providers:**

***Amazon Web Services (AWS):***

* + *Strengths:* Leader in infrastructure services, global reach, and diverse service offerings.
  + *Use Case:* Ideal for scalable infrastructure, storage, and cloud-native applications.

***Microsoft Azure:***

* + Strengths: Strong enterprise solutions, hybrid cloud capabilities, and integration with Microsoft products.
  + Use Case: Suitable for organizations with existing Microsoft environments or those requiring compliance and security features.

***Google Cloud Platform (GCP):***

* + Strengths: Advanced data analytics, AI/ML services, and networking capabilities.
  + Use Case: Ideal for data-intensive applications, machine learning, and big data analytics.

***IBM Cloud:***

* + Strengths: Enterprise-grade solutions, AI integration (Watson), and hybrid cloud support.
  + Use Case: Best for large enterprises with complex infrastructure needs and AI/ML capabilities.

***Oracle Cloud:***

* + Strengths: Database services, ERP solutions, and integration with Oracle software.
  + Use Case: Preferred by organizations running Oracle databases or applications.

***Vmware Cloud:***

* + Strengths: Consistent infrastructure across clouds and on-premises with Vmware environments.
  + Use Case: Ideal for organizations with significant Vmware investments.

**15. Multi-Cloud Challenges:**

***Operational Complexity:***

* + Managing multiple cloud environments requires advanced skills and tools for consistent security, compliance, and performance.

***Cost Management:***

* + Monitoring and optimizing costs across different cloud providers can be challenging, necessitating sophisticated tools.

***Data Integration:***

* + Ensuring data consistency across multiple cloud environments, particularly for large-scale or real-time data, can be difficult.

***Security and Compliance:***

* + Consistent security policies and compliance across different providers can be complex due to varying security standards.

**16. Tools and Best Practices for Multi-Cloud Management**

***Multi-Cloud Management Platforms:***

* + Tools such as HashiCorp Terraform and Google Anthos provide unified management across multiple cloud environments, simplifying operations and governance.

***Cost Optimization Tools:***

* + Solutions like CloudHealth by Vmware and AWS Cost Explorer assist in monitoring and optimizing cloud spending across providers.

***Security Management:***

* + Implementing unified security frameworks like AWS Security Hub or Azure Security Center ensures consistent security policies across environments.

***Automation and Orchestration:***

* + Tools such as Ansible, Puppet, and Kubernetes support efficient deployment and management of applications across multiple clouds.

**Reasons for Choosing AWS Cloud Computing**

Amazon Web Services (AWS) is a leading cloud computing platform that has become a preferred choice for businesses of all sizes. Here are several detailed reasons why AWS stands out as a top choice for cloud computing:

**2. Broad Range of Services**

AWS provides an extensive array of services that cover virtually every aspect of cloud computing. These include:

* **Compute Services**: EC2 instances, Lambda (serverless computing), and ECS (container services).
* **Storage Solutions**: S3 for object storage, EBS for block storage, and Glacier for archival storage.
* **Databases**: Managed relational databases with RDS, NoSQL databases like DynamoDB, and fully managed data warehouse services with Redshift.
* **Networking**: VPC (Virtual Private Cloud), Route 53 for DNS services, and Direct Connect for dedicated network connections.

The wide range of services allows businesses to choose exactly what they need, and the modularity of these services ensures that organizations only pay for what they use.

**3. Global Presence and Availability**

AWS operates in multiple geographic regions around the world, each with multiple availability zones. This global presence ensures that applications can be deployed in multiple regions to achieve low latency, comply with local data regulations, and enhance disaster recovery capabilities. AWS’s network infrastructure is designed for reliability, with built-in redundancy across regions and availability zones, making it a reliable choice for mission-critical applications.

**4. Security and Compliance**

Security is a top priority for AWS. The platform offers robust security features such as:

* **Encryption**: AWS provides encryption in transit and at rest across its services.
* **Identity and Access Management (IAM)**: Granular control over who can access what resources.
* **Monitoring and Logging**: Services like CloudWatch and CloudTrail help monitor and audit actions and events within AWS.

Additionally, AWS complies with a wide array of global security standards and certifications, such as ISO 27001, SOC 1/2/3, GDPR, and HIPAA. This makes AWS a strong choice for industries with stringent regulatory requirements.

**5. Cost Efficiency**

AWS operates on a pay-as-you-go pricing model, which means organizations only pay for the resources they actually use. This pricing structure is ideal for businesses of all sizes, as it allows them to scale their expenses with their growth. AWS also offers various pricing models such as Reserved Instances and Spot Instances, which can further reduce costs for predictable or flexible workloads. Additionally, AWS provides cost management tools like the AWS Pricing Calculator, AWS Budgets, and Cost Explorer to help organizations monitor and optimize their spending.

**6. Innovation and Ecosystem**

AWS is known for its rapid pace of innovation, frequently releasing new services and features that enable businesses to stay ahead of technological trends. For example:

* **AI and Machine Learning**: Services like Amazon SageMaker for building and training machine learning models.
* **IoT**: AWS IoT Core for managing connected devices.
* **Blockchain**: Amazon Managed Blockchain for creating and managing blockchain networks.

AWS also has a vast ecosystem of partners and third-party integrations available through the AWS Marketplace, providing additional tools and services to enhance the AWS environment.

**7. Developer and Enterprise Support**

AWS offers a wide range of support options to meet the needs of developers and enterprises alike.

* **Developer Tools**: Services like AWS CodePipeline, CodeBuild, and CodeDeploy help automate and streamline the software development lifecycle.
* **Enterprise Support**: AWS provides 24/7 support through various support plans, including technical account managers, architecture support, and business-critical system monitoring.
* **Training and Certification**: AWS offers extensive training resources, certification programs, and a well-established community that helps users build their cloud skills and stay updated with best practices.

**8. Interoperability and Integration**

AWS supports a wide range of programming languages, operating systems, databases, and other third-party applications, making it easy to integrate with existing IT environments. This interoperability allows businesses to migrate to the cloud without having to re-engineer their applications entirely, which is particularly advantageous for organizations with complex IT infrastructures.

**9. Customer Success Stories**

Many of the world’s largest companies, including Netflix, Airbnb, and Coca-Cola, have successfully adopted AWS for their cloud computing needs. These customer success stories provide tangible evidence of AWS’s capability to support large-scale, highly demanding applications.

**10. Continuous Improvement and Updates**

AWS continuously invests in its platform, regularly adding new features, updating existing services, and improving overall performance. This commitment to continuous improvement ensures that AWS customers always have access to the latest technologies and can stay competitive in their industries.

**Project:** **Implement a Real-Time Data Processing Pipeline with Kinesis.**

**Introduction: Real-Time Data Processing Pipeline**

In today’s data-driven world, real-time data processing is crucial for organizations that need to act quickly on streaming data. Whether it’s for monitoring systems, analyzing user behavior, processing transactions, or responding to events in real-time, a robust and scalable architecture is required to handle large volumes of data with low latency. This project demonstrates how to build such a pipeline using AWS services, specifically leveraging AWS S3, AWS Lambda, and Amazon Kinesis Streams to create a seamless flow of data from ingestion to processing.

**1. AWS S3 (Simple Storage Service): The Ingestion Point**

**Purpose:** AWS S3 is a highly scalable, durable, and secure object storage service. It is used as the entry point for data in this pipeline, where raw data is ingested before it is processed and analysed. S3 is chosen because it can handle virtually unlimited data with high availability, making it ideal for storing diverse data types such as logs, files, images, and streaming data.

**Functionality:**

* **Storage:** Data from various sources, such as user uploads, system logs, IoT devices, or any data-generating applications, is stored in S3 buckets. Each bucket is designed to store data objects, which can range from a few bytes to several terabytes.
* **Trigger:** S3 can trigger AWS Lambda functions when specific events occur, such as when a new object is uploaded. This event-driven architecture allows the system to automatically process incoming data as soon as it arrives, without the need for manual intervention.

**Flow in the Image:**

* The image begins with an icon representing a user or system that is responsible for uploading data. This data is then stored in an S3 bucket, represented by the red bucket icon. The S3 bucket acts as a storage layer and an event source for the subsequent AWS Lambda function.

**2. AWS Lambda (Producer): Serverless Data Processing**

**Purpose:** AWS Lambda is a serverless compute service that automatically runs your code in response to triggers, such as an S3 event, HTTP requests, or messages from other AWS services. In this architecture, the Lambda function serves as the producer that processes data upon ingestion and prepares it for streaming into Amazon Kinesis.

**Functionality:**

* **Event-Driven Execution:** When a new file or data object is uploaded to the S3 bucket, the Lambda function is automatically invoked. This function can be configured to handle various data formats, such as JSON, CSV, or binary data, depending on the nature of the data being processed.
* **Data Transformation:** The Lambda function processes the raw data by performing tasks such as filtering, parsing, aggregating, or transforming the data into a format suitable for streaming. For example, if the data consists of JSON logs, the Lambda function might extract specific fields, enrich the data with additional information, or convert it to a different format.
* **Pushing to Kinesis:** After processing, the Lambda function acts as a producer by sending the processed data to the Amazon Kinesis Stream. This step is crucial because it ensures that data is immediately available for real-time processing by downstream consumers.

**Flow in the Image:**

* The Lambda function, represented by the orange hexagon with the Lambda symbol, receives data from the S3 bucket, processes it, and forwards it to the Kinesis Stream. This function is responsible for ensuring that the data is correctly formatted and enriched before it enters the streaming pipeline.

**3. Amazon Kinesis Stream: Real-Time Data Streaming**

**Purpose:** Amazon Kinesis is a platform that allows you to collect, process, and analyze real-time streaming data. In this architecture, Kinesis Streams is used to manage the continuous flow of data processed by the Lambda function. Kinesis ensures that data is available in real-time, enabling multiple consumers to access and process the data simultaneously.

**Functionality:**

* **Scalable Stream Processing:** Kinesis Streams can handle large volumes of data with low latency. It partitions the incoming data into shards, each of which can process a specified rate of data. This allows the system to scale horizontally by adding more shards to accommodate higher data throughput.
* **Data Retention:** Kinesis Streams retains the data for a configurable period (from 24 hours up to 7 days), allowing consumers to access the data at their own pace. This is particularly useful when dealing with bursty or unevenly distributed data loads, as it ensures that no data is lost and consumers can process data even after it has been streamed.
* **Multiple Consumers:** Kinesis supports multiple consumers, meaning that different applications or services can process the same data stream concurrently. This is ideal for scenarios where data needs to be analyzed in different ways or sent to multiple destinations.

**Flow in the Image:**

* The Kinesis Stream, represented by the purple box, receives data from the producer Lambda function. The stream acts as a conduit, making the data available in real-time to the downstream Lambda consumers. Kinesis manages the data flow, ensuring that it is distributed evenly across shards and available for immediate processing.

**4. AWS Lambda (Consumers #1 and #2): Real-Time Data Consumers**

**Purpose:** In this architecture, the AWS Lambda functions act as consumers of the Kinesis Stream. Each consumer is responsible for processing the data in real-time and performing specific tasks based on the application’s requirements. By using multiple Lambda functions as consumers, the architecture can support various real-time applications, such as data analytics, alerting systems, or ETL (Extract, Transform, Load) processes.

**Functionality:**

* **Real-Time Processing:** The consumer Lambda functions are triggered as soon as new data is available in the Kinesis Stream. These functions can perform a wide range of tasks, from simple data transformations to complex analytics. For example, one consumer might filter the data and store it in a database, while another might perform real-time analysis and trigger alerts based on predefined conditions.
* **Scalability:** Each Lambda function scales automatically in response to the volume of incoming data. If the data stream contains a large volume of data, AWS Lambda will automatically allocate more compute resources to handle the load, ensuring that data is processed in real-time without delays.
* **Integration with Other AWS Services:** The consumer Lambda functions can be integrated with other AWS services to extend the pipeline’s functionality. For example, a Lambda function might send processed data to Amazon DynamoDB for storage, trigger notifications via Amazon SNS (Simple Notification Service), or load the data into Amazon Redshift for further analysis.

**Flow in the Image:**

* On the right side of the image, two Lambda icons represent the consumers that process the data from the Kinesis Stream. Each consumer function can perform different tasks, depending on the specific requirements of the application. The arrows indicate that these Lambda functions are downstream consumers, receiving and processing data from the Kinesis Stream.

**5. Use Cases and Applications**

The architecture depicted in the image is versatile and can be applied to a wide range of real-time data processing scenarios:

* **Real-Time Analytics:** Businesses can use this pipeline to analyze customer behavior, monitor application performance, or process financial transactions in real-time. For instance, an e-commerce platform might analyze user clicks and purchases in real-time to offer personalized recommendations.
* **Monitoring and Alerts:** IT organizations can use this architecture to monitor system logs and metrics, triggering alerts when certain conditions are met. For example, a monitoring system could analyze server logs in real-time and alert the DevOps team if an error rate exceeds a certain threshold.
* **IoT Data Processing:** This pipeline can be used to process data from IoT devices, such as sensors or smart appliances, in real-time. For example, a smart home system could process temperature and humidity data from sensors to optimize climate control.
* **Data Lake Ingestion:** The architecture can be used to ingest and process data into a data lake, where it can be stored and analyzed later. For example, a financial institution might use this pipeline to ingest transactional data into a data lake for compliance reporting.

**Conclusion: Building a Scalable, Real-Time Data Pipeline**

This real-time data processing pipeline built on AWS services provides a scalable, cost-effective, and efficient way to process streaming data. By leveraging the serverless architecture of AWS Lambda and the real-time capabilities of Amazon Kinesis, the pipeline can handle large volumes of data with low latency, making it suitable for a wide range of applications.

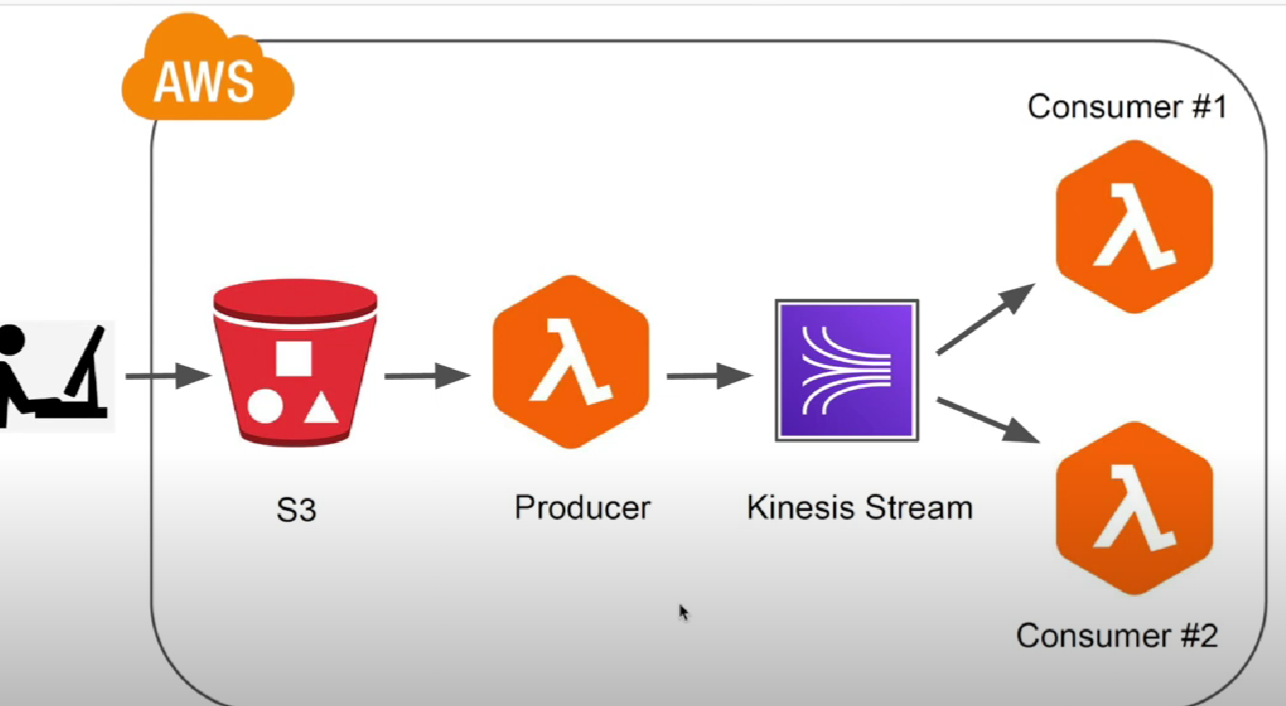
The image illustrates how data flows through the pipeline, from ingestion in S3, through processing in Lambda, to streaming via Kinesis, and finally, to real-time processing by consumer Lambda functions. This architecture is not only powerful but also flexible, allowing you to tailor each component to meet specific application needs.

By implementing this pipeline, organizations can gain immediate insights from their data, respond quickly to events, and build robust applications that leverage the power of real-time processing.

**Steps For creating My project:**

*Step 1:* First an AWS an account is created and then logged in through using root user.

*Step 2:* Then A flow Chart is created for creation and understanding for my project.

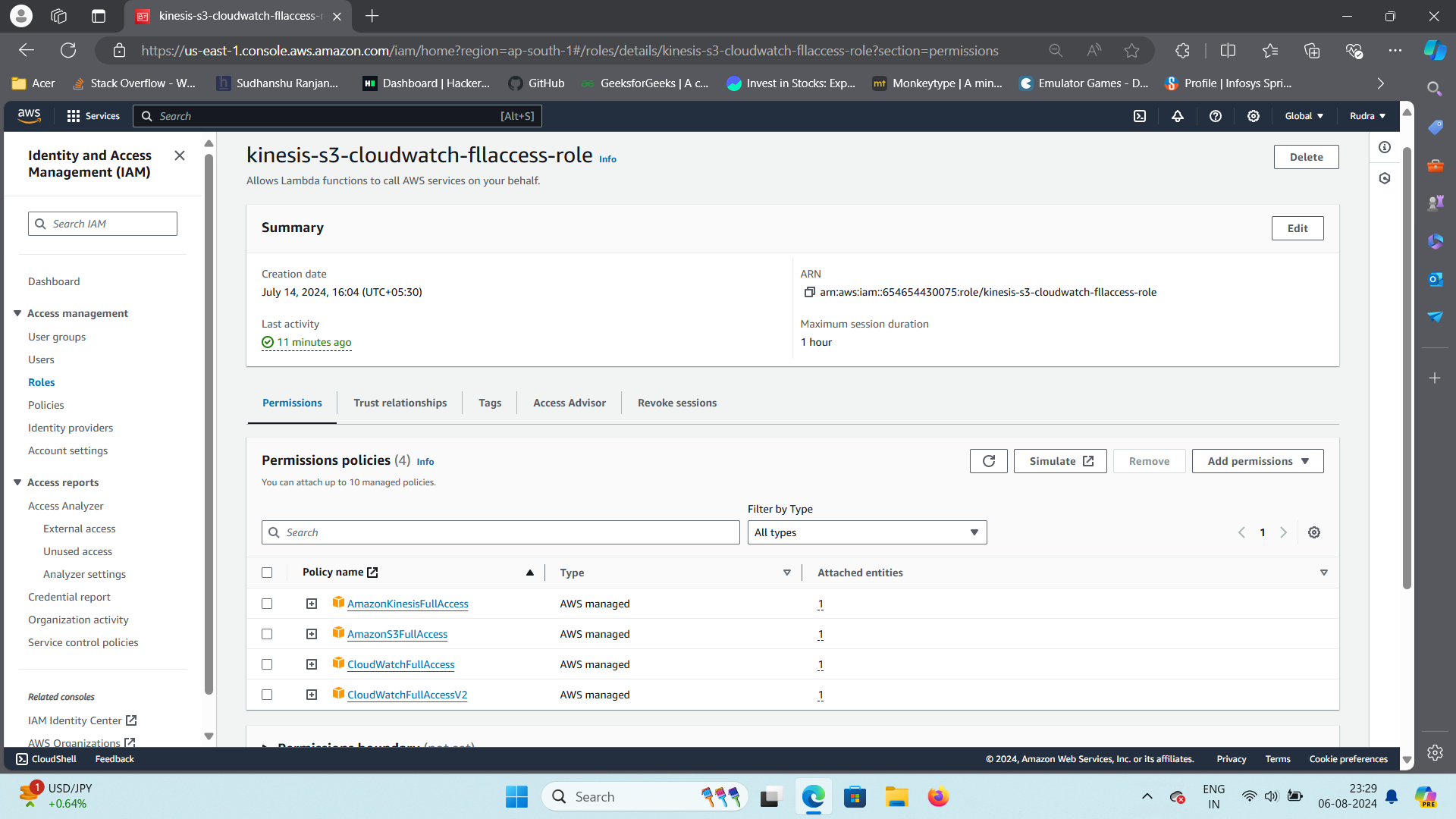


*Step 3:* So, for the above project an role is create by the name of **Kinesis-s3-cloudwatch-fllaccess-role** which includes the following permissions policies:

* **AmazonKinesisFullAccess**
* **AmazonS3FullAccess**
* **CloudWatchFullAccess**
* **CloudWatchFullAccessV2**

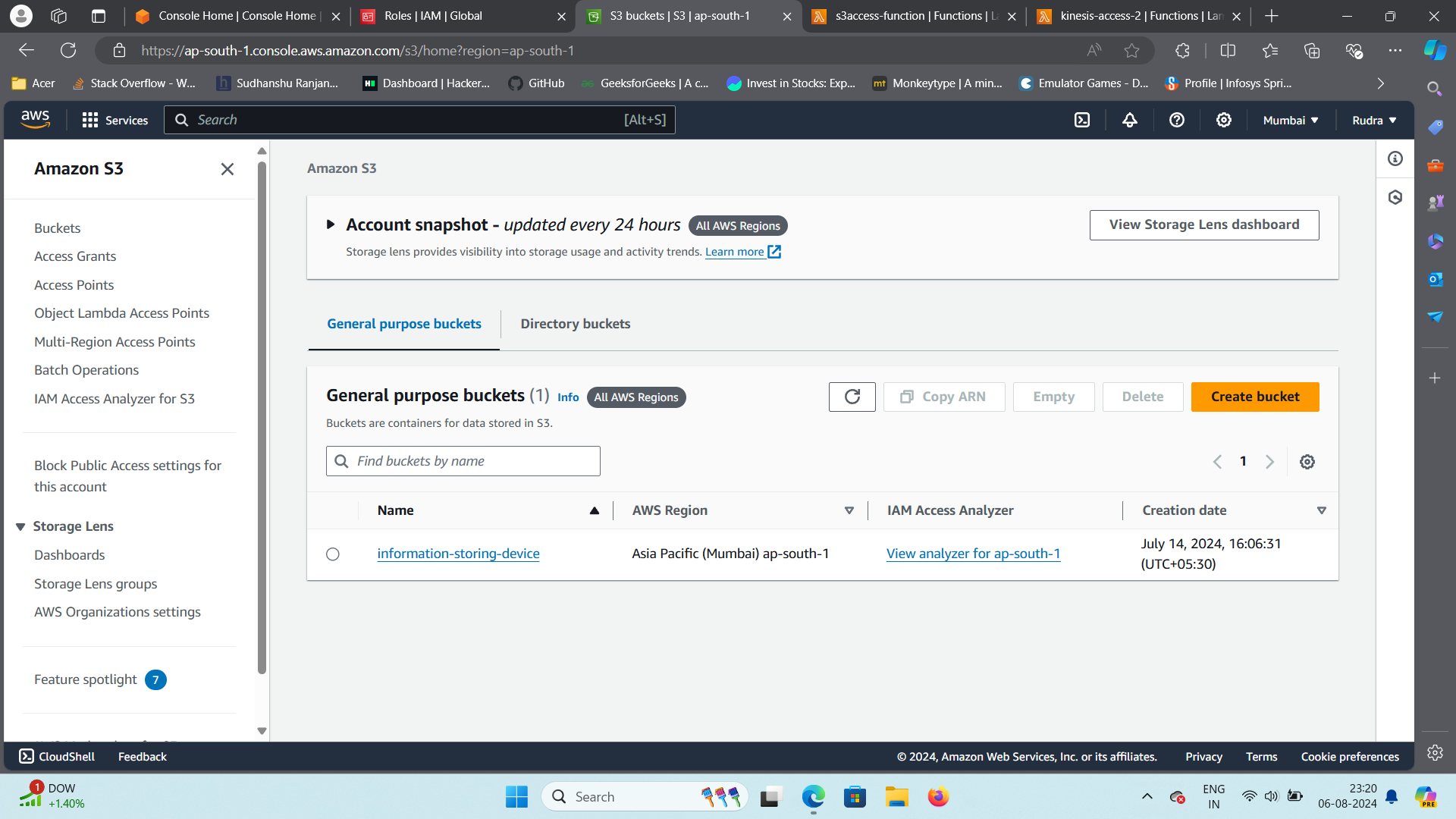
IAM Roles


This is IAM roles image where **Kinesis-s3-cloudwatch-fllaccess-role** is made.



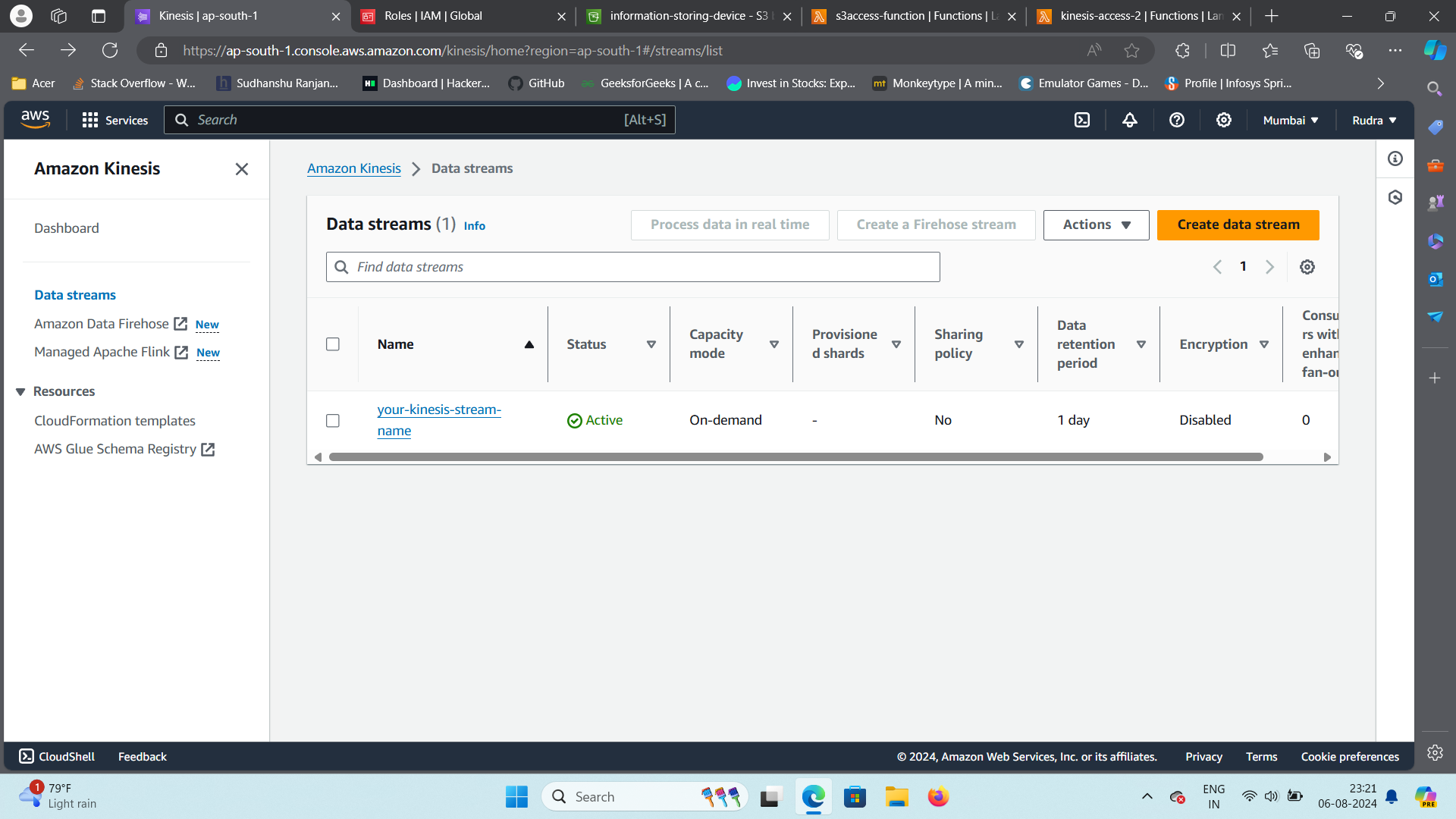
Here are the permission Policies that I used for my **Kinesis-s3-cloudwatch-fllaccess-role.**

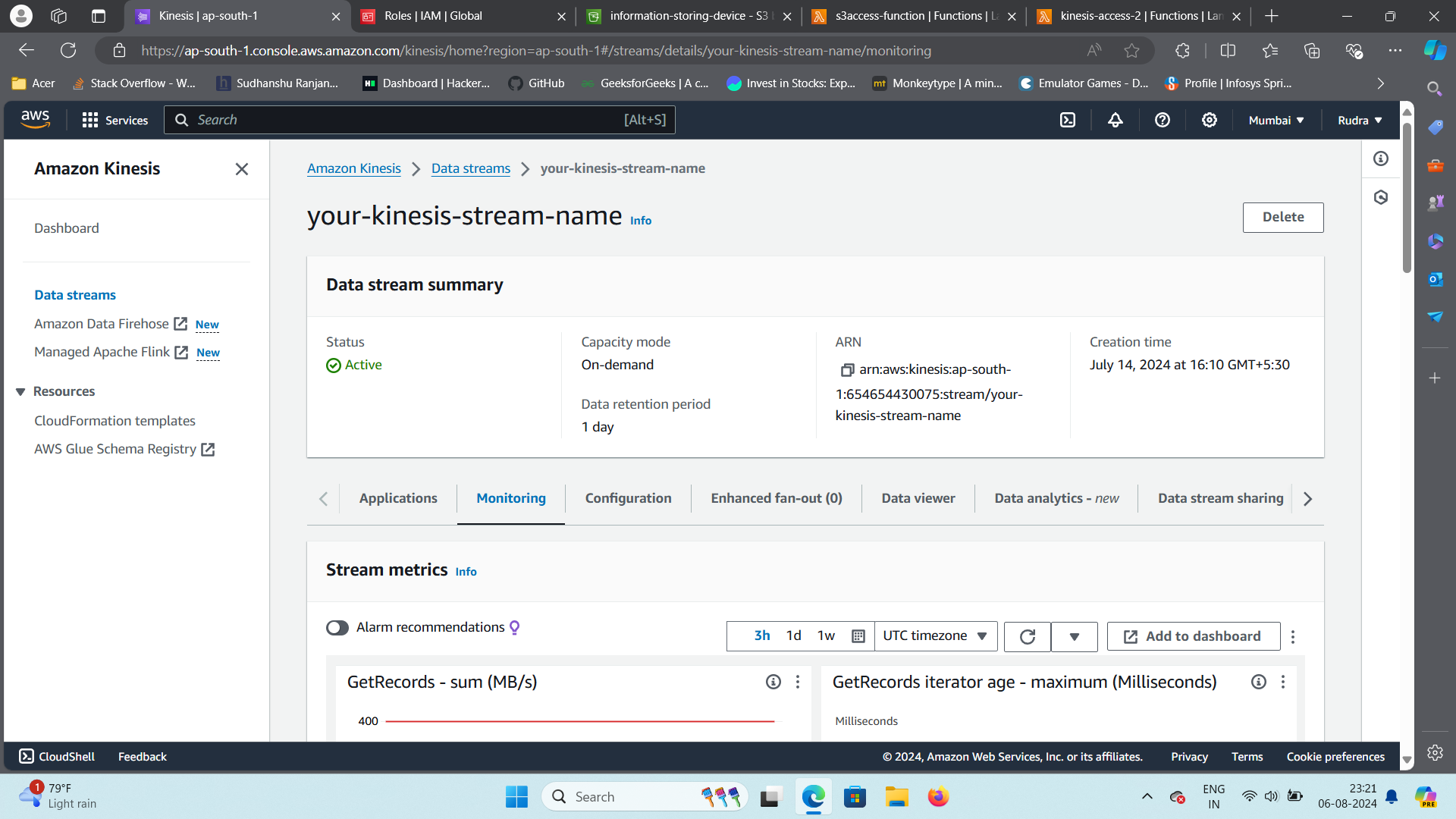
*Step 4:* an S3 bucket is created with name **Information-storing-device** where all type of objects i.e. images, videos, text files and other type of files can be uploaded.



Here is my S3 bucket with name **information-storing-device.**

*Step 4:* A kinesis stream is created with the name **your-kinesis-stream-name** and its status is checked if it is active or not.

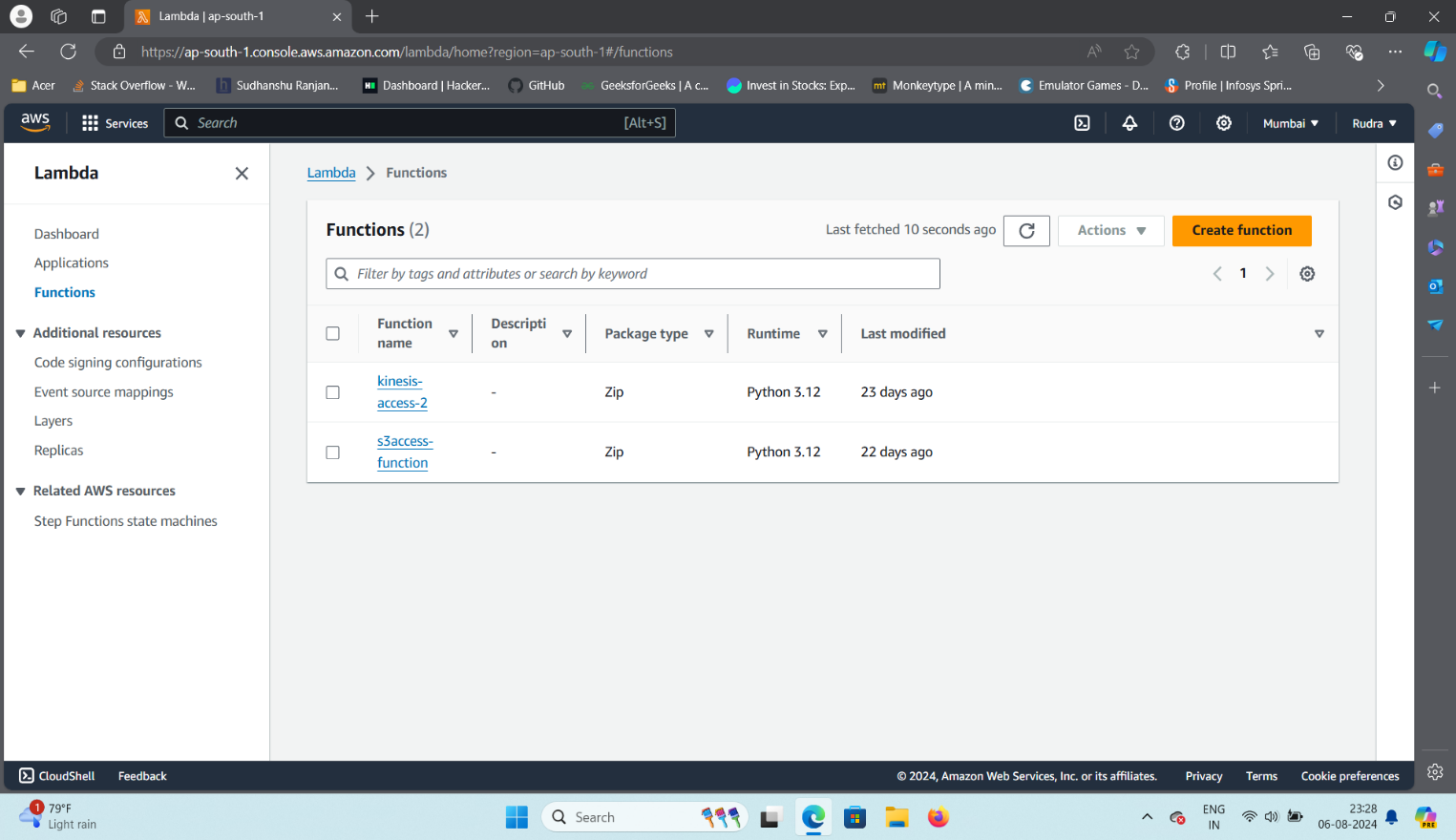
****

****

This kinesis stream had the role **Kinesis-s3-cloudwatch-fllaccess-role.**

*Step 5:* Two functions were created from scratch named:

* ***Kinesis-access2:*** It has the role **Kinesis-s3-cloudwatch-fllaccess-role** and will access the kinesis stream from the user side so that users can access the object of S3 bucket even at same time.
* ***S3accessfunction:*** It has the role **Kinesis-s3-cloudwatch-fllaccess-role** and will access the S3 bucket and provide it to the kinesis stream so that users can access the object of S3 bucket even at same time.
* The above functions were made using python 3.12 and the code was written in them.
* **S3accessfunction** is the producer function i.e. it will provide data to kinesis stream which will be accessed by consumer function.
* **Kinesis-access2** is the consumer function i.e. it will access the kinesis stream which has the data provided from producer function. Multiple consumer function can access the kinesis stream at same time.

****

These two functions have the python 3.12 code as the following:

***Producer Function:***

**import json**

**import boto3**

**from botocore.exceptions import ClientError**

**kinesis = boto3.client(‘kinesis’)**

**s3 = boto3.client(‘s3’)**

**def lambda\_handler(event, context):**

**try:**

**# Extract bucket name and file key from the event**

**bucket = event[‘Records’][0][‘s3’][‘bucket’][‘name’]**

**key = event[‘Records’][0][‘s3’][‘object’][‘key’]**

**# Get the file type (assumption: file type is the extension)**

**filetype = key.split(‘.’)[-1]**

**# Prepare data for Kinesis**

**data = {**

**‘filename’: key,**

**‘filetype’: filetype**

**}**

**# Send data to Kinesis Stream**

**kinesis.put\_record(**

**StreamName=’your-kinesis-stream-name’,**

**Data=json.dumps(data),**

**PartitionKey=key**

**)**

**return {**

**‘statusCode’: 200,**

**‘body’: json.dumps(‘File processed successfully’)**

**}**

**except ClientError as e:**

**return {**

**‘statusCode’: 500,**

**‘body’: json.dumps(f”ClientError: {e}”)**

**}**

**except Exception as e:**

**return {**

**‘statusCode’: 500,**

**‘body’: json.dumps(f”Exception: {e}”)**

**}**

***Consumer Function:***

**import json**

**import base64**

**def lambda\_handler(event, context):**

**try:**

**for record in event[‘Records’]:**

**# Decode the Kinesis data**

**data = base64.b64decode(record[‘kinesis’][‘data’]).decode(‘utf-8’)**

**# Verify that the data is not empty**

**if not data:**

**print(“Empty data received.”)**

**continue**

**payload = json.loads(data)**

**filename = payload.get(‘filename’, ‘Unknown’)**

**filetype = payload.get(‘filetype’, ‘Unknown’)**

**print(f”The provided file is {filename}/{filetype}”)**

**return {**

**‘statusCode’: 200,**

**‘body’: json.dumps(‘File type logged successfully’)**

**}**

**except json.JSONDecodeError as e:**

**print(f”JSONDecodeError: {e}”)**

**return {**

**‘statusCode’: 400,**

**‘body’: json.dumps(f”JSONDecodeError: {e}”)**

**}**

**except KeyError as e:**

**print(f”KeyError: {e}”)**

**return {**

**‘statusCode’: 400,**

**‘body’: json.dumps(f”KeyError: {e}”)**

**}**

**except Exception as e:**

**print(f”Exception: {e}”)**

**return {**

**‘statusCode’: 500,**

**‘body’: json.dumps(f”Exception: {e}”)**

**}**

* This two-function code will help the functions to access their respective tasks S3bucket and Kinesis access stream.

*Step 6:* The **s3access-function** was given a s3 trigger which will access the s3 bucket.

**A screenshot of a computer

Description automatically generated**  This function’s active status has been enabled and now it is ready to run.

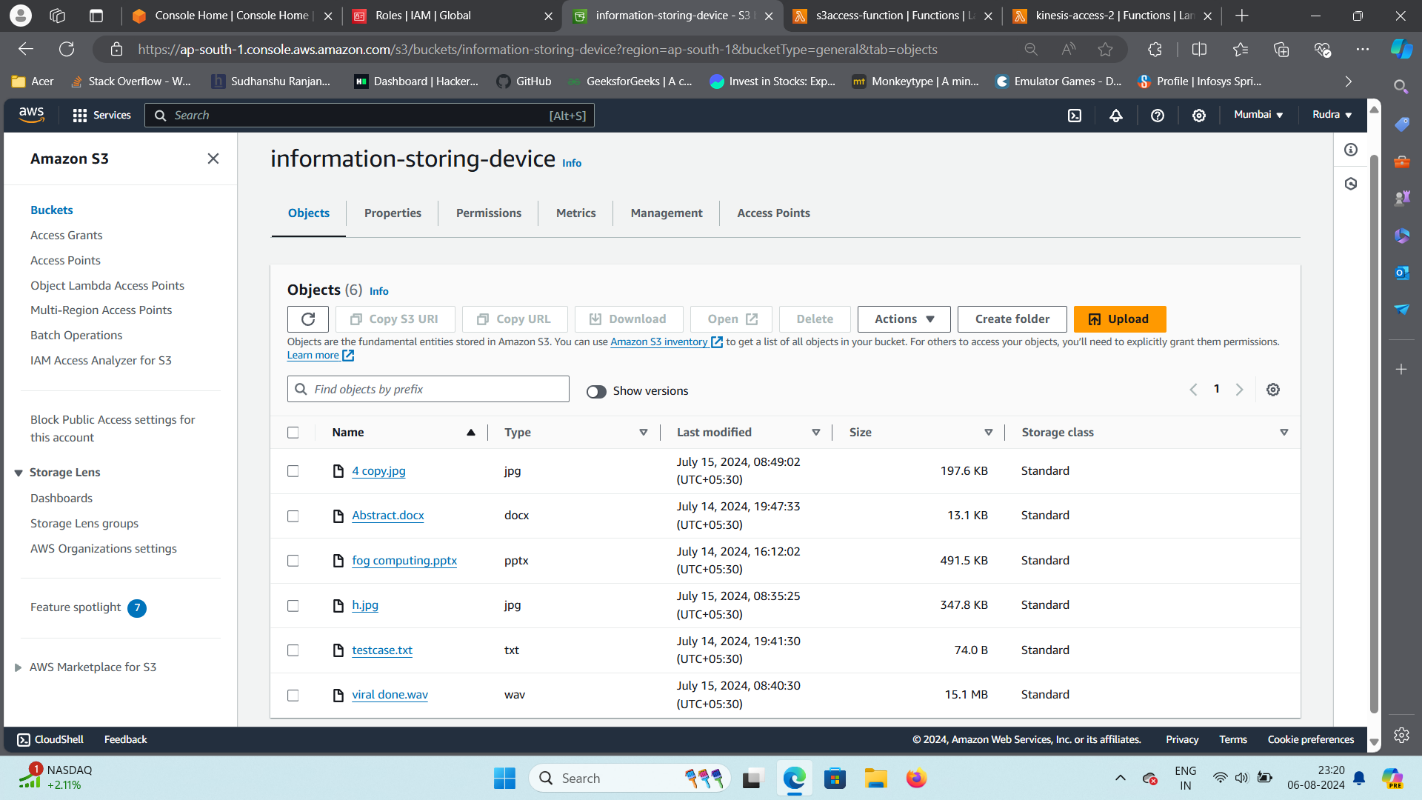
*Step 7:* **Kinesis-access2 function** was given a kinesis trigger which will access the kinesis data stream.

A screenshot of a computer

Description automatically generated

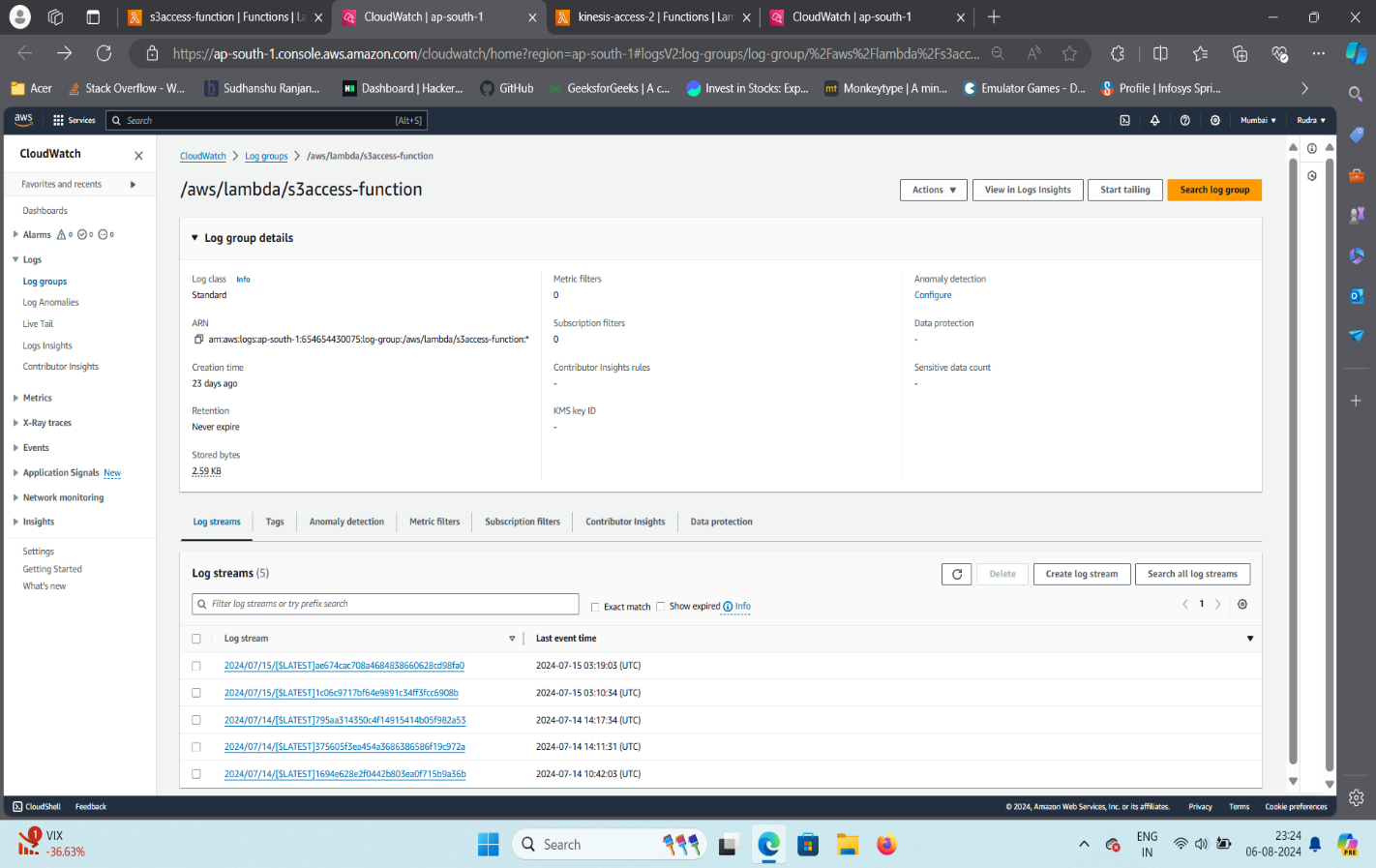
The function was enabled and now it is ready to access the kinesis data stream.

*Step 8:* Some files were added in S3 bucket.



These files were the mix of .txt, .mp4, .jpg etc.

*Step 9:* AWS whole system was refreshed and then S3access function was opened and directly went to its cloudwatch log to check whether the function is accessing the S3 bucket.



With this it is verified that s3access function is accessing the S3 bucket and its uploading it to the kinesis data stream.

*Step 10:* Now on the consumer side the kinesis access function is checked whether its accessing the kinesis data stream or not.

A screenshot of a computer

Description automatically generated

With this it is verified that the consumer side function is able to access the kinesis data stream and it provides which type of data it is.

**Real Life Applications of Real-Time Data Processing Pipeline with Kinesis.**

Here are 10 real-world applications of Kinesis:

1. **IoT Data Processing:** Analyse data from IoT devices to monitor equipment health, detect anomalies, and optimize operations.
2. **Real-Time Analytics:** Analyse streaming data from web and mobile apps to understand user behaviour, personalize recommendations, and detect fraud.
3. **Financial Data Processing:** Process high-volume financial data for risk assessment, fraud detection, and algorithmic trading.
4. **Social Media Analytics:** Analyse social media data to track sentiment, identify trends, and engage with customers.
5. **Gaming Analytics:** Analyse gaming data to understand player behaviour, optimize game design, and detect cheating.
6. **Machine Learning:** Train machine learning models on streaming data for real-time predictions and recommendations.
7. **Customer Support:** Analyse customer support interactions to improve response times, identify common issues, and enhance customer satisfaction.
8. **Log Analysis:** Process and analyse logs from applications and infrastructure to monitor performance, detect anomalies, and troubleshoot issues.
9. **Clickstream Analysis:** Analyse user clickstream data to understand website navigation, identify conversion bottlenecks, and optimize user experience.
10. **Fraud Detection:** Analyse transactional data to detect fraudulent activity in real-time.

Here is name of the companies that use real-time data processing pipeline with Kinesis:

**IoT Data Processing**

* **Smart Agriculture:** **Cropify:** Uses Kinesis to analyse data from sensors in farms for real-time monitoring and optimization.
* **Smart Cities:** **Tata Power:** Implements Kinesis in smart city initiatives for data analysis and infrastructure management.

**Real-Time Analytics**

* **E-commerce:** **Myntra:** Utilizes Kinesis to analyze customer behavior, personalize recommendations, and optimize the shopping experience.
* **Financial Services:** **HDFC Bank:** Leverages Kinesis for real-time fraud detection, risk assessment, and personalized financial advice.

**Financial Data Processing**

* **Stock Market:** **National Stock Exchange of India (NSE):** Processes high-volume financial data using Kinesis for real-time market analysis and trading.
* **Payment Processing:** **Paytm:** Employs Kinesis to process payment data, detect fraud, and ensure secure transactions.

**Social Media Analytics**

* **Political Analysis:** **India Today:** Uses Kinesis to analyse social media data for political sentiment analysis and tracking trends.
* **Brand Monitoring:** **Brandwatch:** Leverages Kinesis to monitor social media mentions of brands, track sentiment, and engage with customers.

**Gaming Analytics**

* **Online Gaming:** **Dream11:** A popular fantasy sports platform, utilizes Kinesis for real-time game analysis and player behaviour tracking.
* **E-sports:** **Nodwin Gaming:** Processes data from e-sports tournaments using Kinesis to analyse player performance and enhance viewer experience.

**Machine Learning**

* **Predictive Maintenance:** **Tata Consultancy Services (TCS):** Applies Kinesis for predictive maintenance in industrial applications to optimize operations and reduce downtime.
* **Fraud Detection:** **Axis Bank:** Leverages Kinesis to train machine learning models for fraud detection in real-time.

**Customer Support**

* **Call Centres:** **Tata Communications:** Uses Kinesis to analyze customer support interactions, improve response times, and enhance customer satisfaction.
* **Chatbots:** **Infosys:** Implements Kinesis to power AI-driven chatbots for customer support and engagement.

**Log Analysis**

* **IT Infrastructure:** **Wipro:** Leverages Kinesis for log analysis to monitor IT infrastructure, detect anomalies, and troubleshoot issues.
* **Cloud Computing:** **Amazon Web Services (AWS):** Uses Kinesis for log analysis and monitoring of cloud infrastructure.

**Clickstream Analysis**

* **E-commerce:** **Flipkart:** Analyses user clickstream data using Kinesis to optimize website navigation and improve user experience.
* **Digital Marketing:** **Google Marketing Platform:** Leverages Kinesis for real-time data analysis and optimization of digital marketing campaigns.

**Fraud Detection**

* **Online Payments:** **PayU:** Employs Kinesis to detect fraudulent transactions in online payments and protect customers.
* **Insurance Claims:** **ICICI Lombard:** Utilizes Kinesis for real-time analysis of insurance claim data to identify fraudulent activities.

**Conclusion**

Through this project, we gained an in-depth understanding of how to construct a real-time data processing pipeline using various AWS services, including S3, Lambda, and Kinesis Streams. The project allowed us to delve into the intricacies of integrating these components into a cohesive, event-driven architecture. This integration facilitated automated data ingestion, processing, and streaming, demonstrating the power and efficiency of AWS's cloud-native solutions. We explored how AWS Lambda, as a serverless compute service, can be effectively leveraged to act as both a producer and a consumer within this pipeline. This not only underscored the flexibility and scalability of serverless computing but also highlighted the seamless scalability of Kinesis Streams in handling high volumes of real-time data. Our exploration extended into practical applications such as real-time analytics, system monitoring, and IoT data management, where these AWS services can be applied to solve complex, real-world problems.

Overall, this project significantly enhanced our problem-solving abilities by enabling us to apply theoretical knowledge to practical scenarios. We gained hands-on experience in designing and implementing a scalable, efficient, and automated data processing pipeline, which has fortified our understanding of cloud architecture. The knowledge acquired through this process equips us with the confidence and skills necessary to tackle similar challenges in the future. Moreover, our comprehension of how to utilize AWS services for real-time data processing has broadened, laying a strong foundation for future projects that require robust, scalable, and real-time data processing capabilities.

**Bibliography**

**1.** Gokboru Tech for AWS learning basics and getting experience.

**2**. AWS documentation and manuals for learning how to use their tools.

**3**. Project images are the screenshot of the project.

**4.** Code was written and helped by Trainer Mr. Yash Raj.

**5.** Real Life applications of Real time data processing pipeline with kinesis: [What is AWS Kinesis? Design Pattern, Use Cases & Comparison | upGrad blog](https://www.upgrad.com/blog/what-is-aws-kinesis/)