"StreamGuard" - Anti-Piracy Analytics for Live Sports Streaming

Objective

Build a data-driven anti-piracy solution, "StreamGuard," to detect, analyze, and prioritize pirated live sports streams, providing actionable insights for a sports streaming platform to protect its content and revenue during high-stakes events.

Project Overview

StreamGuard will scrape the listed domains for links streaming Specific events (e.g., UFC, NFL), stream the data through Kafka for real-time processing runs in Azure Virtual Machine, save the links in Azure Blob Storage, and collect evidence like screenshots and iframe sources

Technologies Used

- **Apache Kafka**: Real-time streaming of piracy links and evidence (deployed on Azure VM).
- **Azure Virtual Machine (VM):** Ubuntu-based VM hosting Kafka, producer, and consumer scripts.
- **Azure Blob Storage**: Storage for JSON link data, screenshots, and iframe metadata.
- **Python**: Core language with libraries:
  - ***kafka-python***: Kafka producer and consumer logic.
  - ***azure-storage-blob***: Upload to Azure Blob Storage.
  - ***selenium***: Web scraping and evidence collection.
- **CLI Deployment**: Scripts executed via command line on the Azure VM.

Architecture

1. **Kafka Deployment**:
    - Hosted on an Azure VM (Ubuntu 20.04 LTS, B2S instance).
    - Two topics: piracy-links (scraped links) and piracy-evidence (screenshots, iframes).
2. **Producer**:
    - Scrapes target domains for links containing "Denver Onyx vs. New York Exiles" using Selenium in headless mode.
    - Visits each link to capture screenshots and iframe sources as evidence.
    - Streams data to Kafka topics in JSON format.
3. **Consumer**:
    - Two consumers run via CLI:
        - Links consumer: Uploads JSON data to streamguard-links container in Blob Storage.
        - Evidence consumer: Uploads screenshots (PNG) and iframe JSON to streamguard-evidence container.
4. **Storage**:
    - Azure Blob Storage containers: streamguard-links (links) and streamguard-evidence (screenshots, iframes).

Workflow

- **Setup**: Kafka runs on an Azure VM with ports 9092 (Kafka) and 22 (SSH) open. Blob Storage is configured with two containers.
- **Scraping**: The producer script scans domains every 5 minutes (simulating live event monitoring), targeting the event name in URLs or text.
- **Evidence**: For each pirated link, a screenshot of the stream and its iframe source (e.g., CDN URL) are captured.
- **Streaming**: Kafka streams data to topics in real time.

- **Storage**: Consumers upload links and evidence to Blob Storage via CLI execution on the VM.
- **Output**: JSON files (e.g., *links/2025-04-05T14:00:00.json*), screenshots (e.g., *screenshots/screenshot_exiles.png*), and iframe metadata (e.g., *iframes/exiles.json*).

Key Features

- **Event-Specific Detection**: Filters links by "Denver Onyx vs. New York Exiles" for precision.
- **Real-Time Monitoring**: Kafka ensures immediate data flow during live events.
- **Evidence Collection**: Screenshots and iframes provide actionable proof for legal notices.
- **Azure Compatibility**: Leverages Azure VM and Blob Storage, mirroring DAZN's infrastructure.
- **CLI Automation**: Producer and consumer scripts run seamlessly on the VM via command line.

Following is the Step-by-Step Process to implement StreamGuard

Step 1: Create an Azure Virtual Machine (VM)

Action:

Log in to portal.azure.com.

Click "+ Create a resource" > "Virtual Machine" > "Create".

Configure:

- **Subscription**: Free trial or existing.
- **Resource Group**: New (e.g., StreamGuardRG).
- **VM Name**: kafka-vm.
- **Region**: East US (or nearest).
- **Image**: Ubuntu Server 20.04 LTS.
- **Size**: B2S (2 vCPUs, 4 GB RAM) for Kafka + Selenium (use B1S if limited to free tier).
- **Authentication**: SSH public key (generate with ssh-keygen if needed).

- **Inbound Ports**: Allow SSH (22) and Kafka (9092).

Click "Review + Create" > "Create".

After deployment (~2-3 minutes), note the public IP (e.g., 20.123.45.67).

Outcome: Azure VM ready to host Kafka.

Step 2: Connect to Azure VM

- **Action**:

  SSH into VM: ssh -i your-key.pem azureuser@20.123.45.67 (replace with your VM's IP).

- **Outcome**: Command-line access to VM.

Step 3: Install Kafka on Azure VM

- **Action:**
  1. Update VM:

     >: *sudo apt update && sudo apt upgrade -y*

  2. Install Java (Kafka dependency):

     >: *sudo apt install openjdk-11-jdk -y*

  3. Download Kafka (e.g., 3.6.1):

     >: *wget https://downloads.apache.org/kafka/3.6.1/kafka_2.13-3.6.1.tgz*

     >: *tar -xzf kafka_2.13-3.6.1.tgz*

     >: *cd kafka_2.13-3.6.1*

  4. Start Zookeeper:

     >: *bin/zookeeper-server-start.sh -daemon config/zookeeper.properties*

  5. Start Kafka:

     >: *bin/kafka-server-start.sh -daemon config/server.properties*

  6. Create topics:

     >: *bin/kafka-topics.sh --create --topic piracy-links --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1*

```
>: bin/kafka-topics.sh --create --topic piracy-evidence --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1
```

- **Outcome:** Kafka running on VM with topics ready.

Step 4: Install Dependencies on VM

- Action:
  1. Install Python and libraries:

     ```
     >:  sudo apt install python3-pip -y
     >:  pip3 install kafka-python azure-storage-blob selenium
     ```

  2. Install Chrome and ChromeDriver (for Selenium):

     ```
     >:  sudo apt install chromium-browser -y
     >:  wget https://chromedriver.storage.googleapis.com/114.0.5735.90/chromedriver_linux64.zip
     >:  unzip chromedriver_linux64.zip
     >:  sudo mv chromedriver /usr/local/bin/
     ```

- **Outcome:** VM ready to run producer and consumer scripts.

Step 5: Create Azure Blob Storage

- **Action:**
  1. In Azure Portal, click "+ Create a resource" > "Storage account" > "Create".
  2. **Configure:**
     - **Subscription:** Free trial or existing.
     - **Resource Group:** *StreamGuardRG*.
     - **Storage account name:** *streamguardstorage* (unique, lowercase).
     - **Region:** East US (match VM).
     - **Performance:** Standard.
     - **Redundancy:** LRS.
  3. Click "Review + Create" > "Create".

    4. After deployment, go to *streamguardstorage* > "Containers".
    5. **Create containers:**
        - "+ Container" > Name: *streamguard-links* > Private > Create.
        - "+ Container" > Name: *streamguard-evidence* > Private > Create**.**
    6. **Get connection string:**
        - Go to "Access keys" > Show key1 > Copy "Connection string" (e.g., *DefaultEndpointsProtocol=https;AccountName=streamguardstorage;AccountKey=...*).

- **Outcome:** Blob Storage ready with containers.

Step 6: Update Producer Code

- **Action**:
    1. Create KafkaProducer.ipynb locally:

       Refer to The file with same name in folder for python code

Outcome: Producer scrapes links and evidence, sends to Kafka.

Step 7: Update Consumer Code

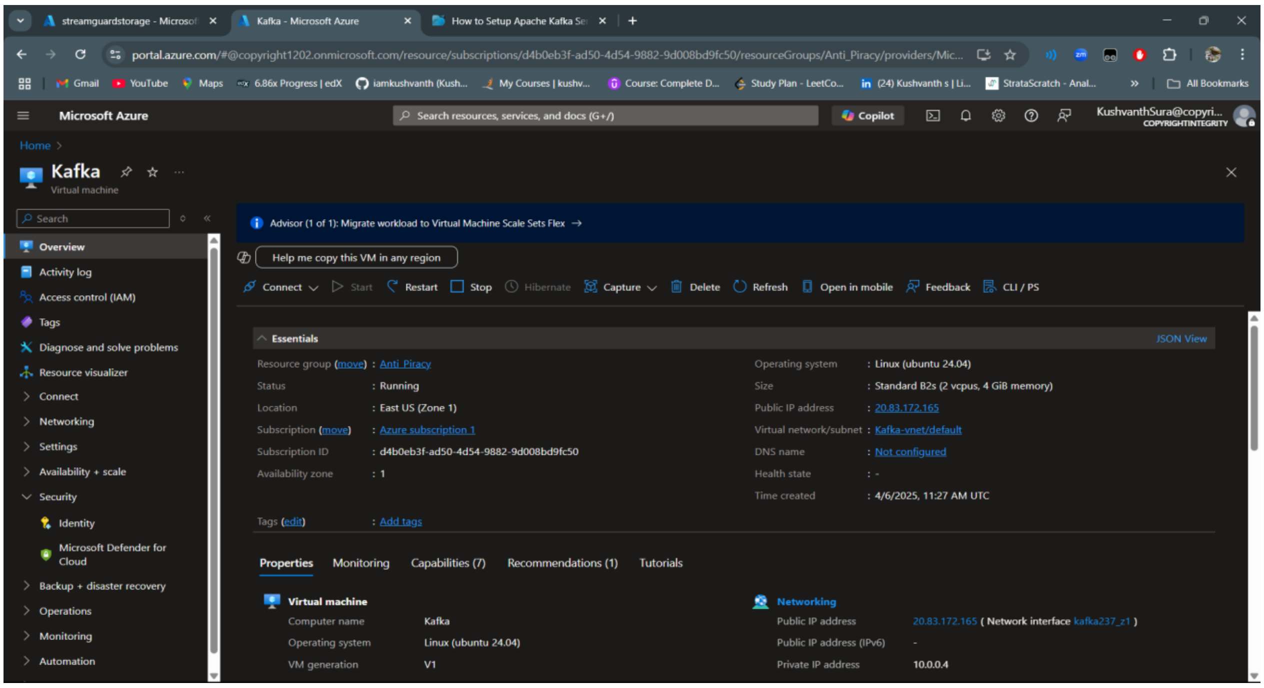
- **Action**:
    1. Create two consumer scripts locally:
        - KafkaConsumer_Links.ipynb

          File that collects links from the domain and store them into Container "streamguard-links"
        - KafkaConsumer_Evidence.ipynb

          File that collects evidence for the link that are streaming event including screenshot and iframe src

Outcome: Consumers store data in Blob Storage.

Open Kafka - VM in Azure



Essentials

Resource group (move)	: Anti_Piracy	Operating system	: Linux (ubuntu 24.04)
Status	: Running	Size	: Standard B2s (2 vcpus, 4 GiB memory)
Location	: East US (Zone 1)	Public IP address	: 20.83.172.165
Subscription (move)	: Azure subscription 1	Virtual network/subnet	: Kafka-vnet/default
Subscription ID	: d4b0eb3f-ad50-4d54-9882-9d008bd9fc50	DNS name	: Not configured
Availability zone	: 1	Health state	: -
		Time created	: 4/6/2025, 11:27 AM UTC
Tags (edit)	: Add tags		

Virtual machine

Computer name	Kafka
Operating system	Linux (ubuntu 24.04)
VM generation	V1

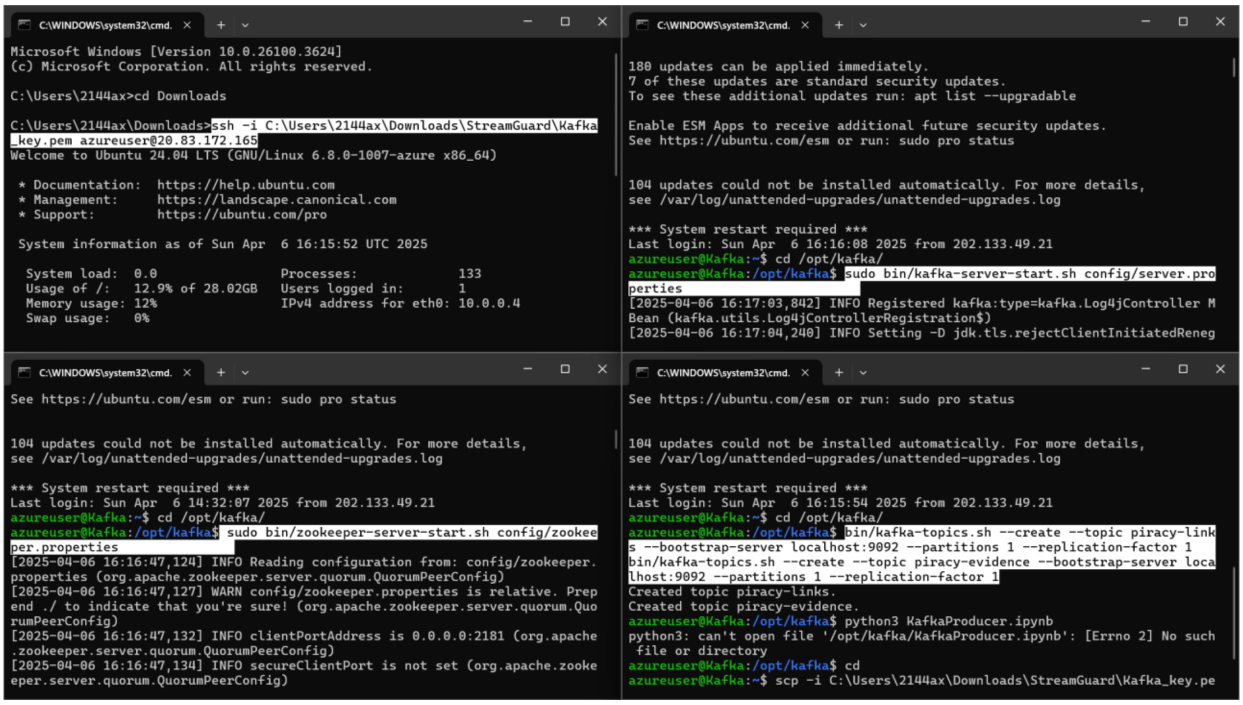
Networking

Public IP address	20.83.172.165 (Network interface kafka237_z1)
Public IP address (IPv6)	-
Private IP address	10.0.0.4

Open Treminal and Connect VM locally using SSH

Open Terminal 2 and run Zookeeper server

In Terminal 3 start Kafka-Server and last and Terminal 4 Create The topics where Producer and Consumer would run



```
Microsoft Windows [Version 10.0.26100.3624]
(c) Microsoft Corporation. All rights reserved.

C:\Users\2144ax>cd Downloads

C:\Users\2144ax\Downloads>ssh -i C:\Users\2144ax\Downloads\StreamGuard\Kafka
_key.pem azureuser@20.83.172.165
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1007-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sun Apr  6 16:15:52 UTC 2025

  System load:  0.0               Processes:             133
  Usage of /:   12.9% of 28.02GB   Users logged in:       1
  Memory usage: 12%               IPv4 address for eth0: 10.0.0.4
  Swap usage:   0%
```

```
180 updates can be applied immediately.
7 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


104 updates could not be installed automatically. For more details,
see /var/log/unattended-upgrades/unattended-upgrades.log

*** System restart required ***
Last login: Sun Apr  6 16:16:08 2025 from 202.133.49.21
azureuser@Kafka:~$ cd /opt/kafka/
azureuser@Kafka:/opt/kafka$ sudo bin/kafka-server-start.sh config/server.pro
perties
[2025-04-06 16:17:03,842] INFO Registered kafka:type=kafka.Log4jController M
Bean (kafka.utils.Log4jControllerRegistration$)
[2025-04-06 16:17:04,240] INFO Setting -D jdk.tls.rejectClientInitiatedReneg
```

```
See https://ubuntu.com/esm or run: sudo pro status


104 updates could not be installed automatically. For more details,
see /var/log/unattended-upgrades/unattended-upgrades.log

*** System restart required ***
Last login: Sun Apr  6 14:32:07 2025 from 202.133.49.21
azureuser@Kafka:~$ cd /opt/kafka/
azureuser@Kafka:/opt/kafka$ sudo bin/zookeeper-server-start.sh config/zookee
per.properties
[2025-04-06 16:16:47,124] INFO Reading configuration from: config/zookeeper.
properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2025-04-06 16:16:47,127] WARN config/zookeeper.properties is relative. Prep
end ./ to indicate that you're sure! (org.apache.zookeeper.server.quorum.Quo
rumPeerConfig)
[2025-04-06 16:16:47,132] INFO clientPortAddress is 0.0.0.0:2181 (org.apache
.zookeeper.server.quorum.QuorumPeerConfig)
[2025-04-06 16:16:47,134] INFO secureClientPort is not set (org.apache.zooke
eper.server.quorum.QuorumPeerConfig)
```

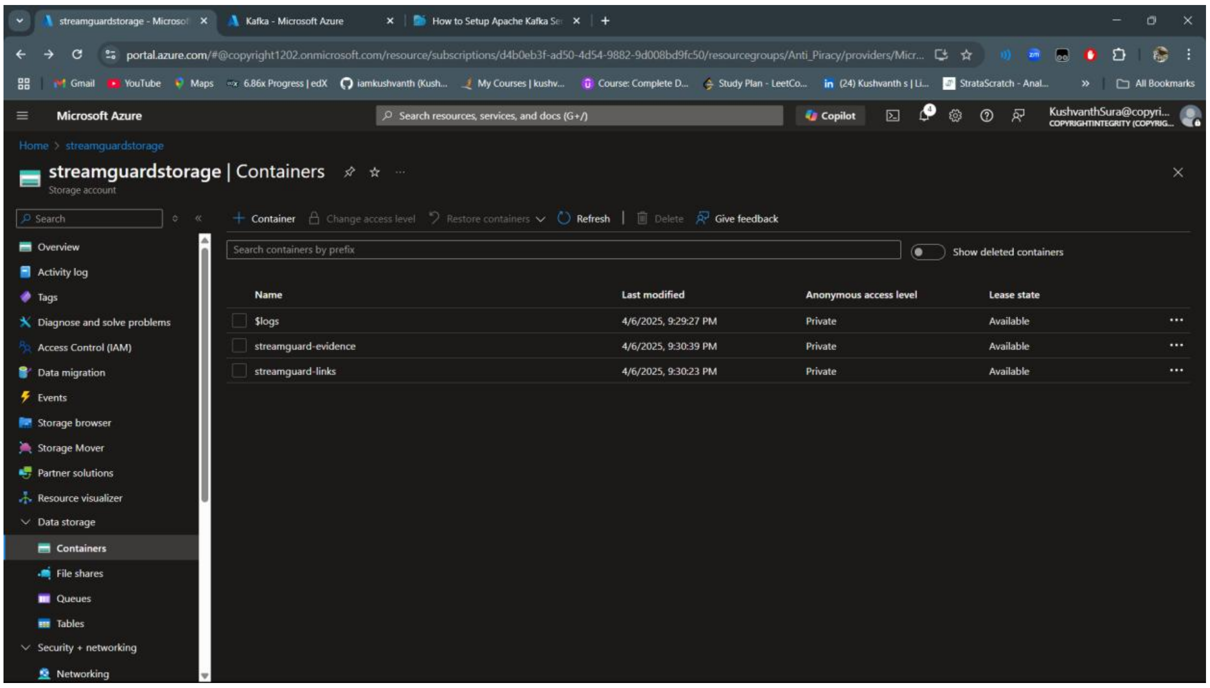
```
See https://ubuntu.com/esm or run: sudo pro status


104 updates could not be installed automatically. For more details,
see /var/log/unattended-upgrades/unattended-upgrades.log

*** System restart required ***
Last login: Sun Apr  6 16:15:54 2025 from 202.133.49.21
azureuser@Kafka:~$ cd /opt/kafka/
azureuser@Kafka:/opt/kafka$ bin/kafka-topics.sh --create --topic piracy-link
s --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1
bin/kafka-topics.sh --create --topic piracy-evidence --bootstrap-server loca
lhost:9092 --partitions 1 --replication-factor 1
Created topic piracy-links.
Created topic piracy-evidence.
azureuser@Kafka:/opt/kafka$ python3 KafkaProducer.ipynb
python3: can't open file '/opt/kafka/KafkaProducer.ipynb': [Errno 2] No such
 file or directory
azureuser@Kafka:/opt/kafka$ cd
azureuser@Kafka:~$ scp -i C:\Users\2144ax\Downloads\StreamGuard\Kafka_key.pe
```

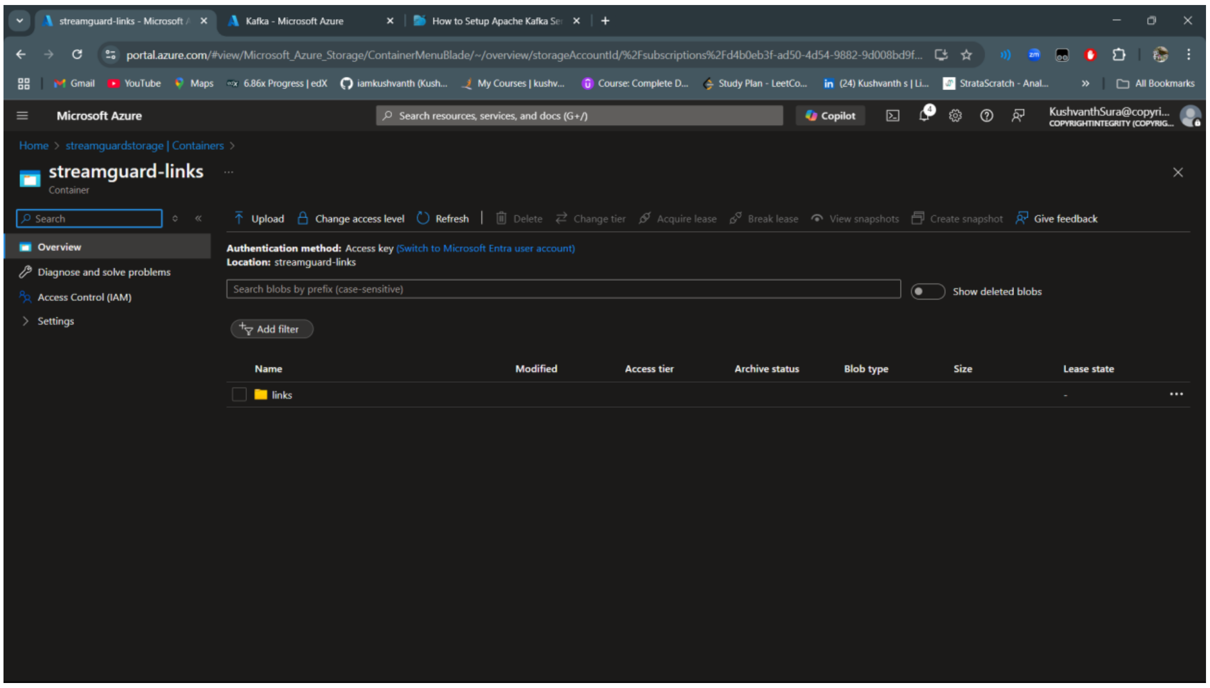
After Successfully setting up Kafka server run the Producer and Consumer code in the respective folder which once uploads the links and evidence in respective containers

Blob Storage : streamgueardstorage, with **Containers**: streamguard-evidence , streamguard-links



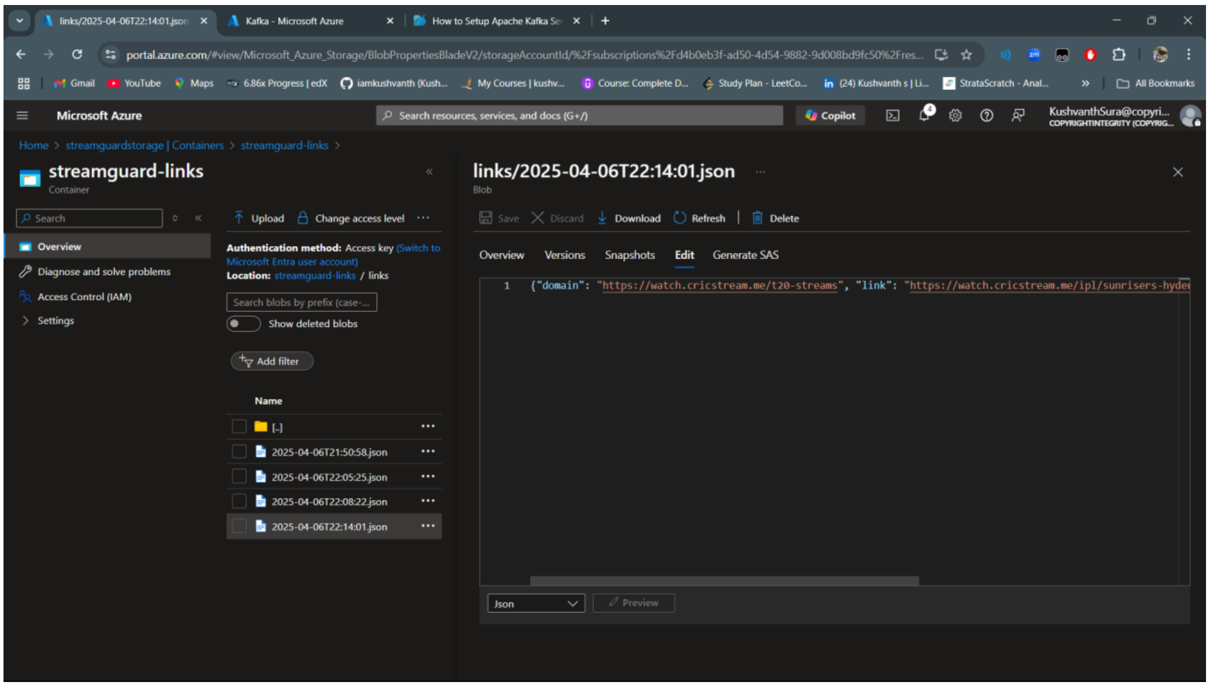
Name	Last modified	Anonymous access level	Lease state
$logs	4/6/2025, 9:29:27 PM	Private	Available
streamguard-evidence	4/6/2025, 9:30:39 PM	Private	Available
streamguard-links	4/6/2025, 9:30:23 PM	Private	Available

After Successful execution of code the containers would look something like this



Authentication method: Access key (Switch to Microsoft Entra user account)
Location: streamguard-links

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
links						-



streamguard-links
Container

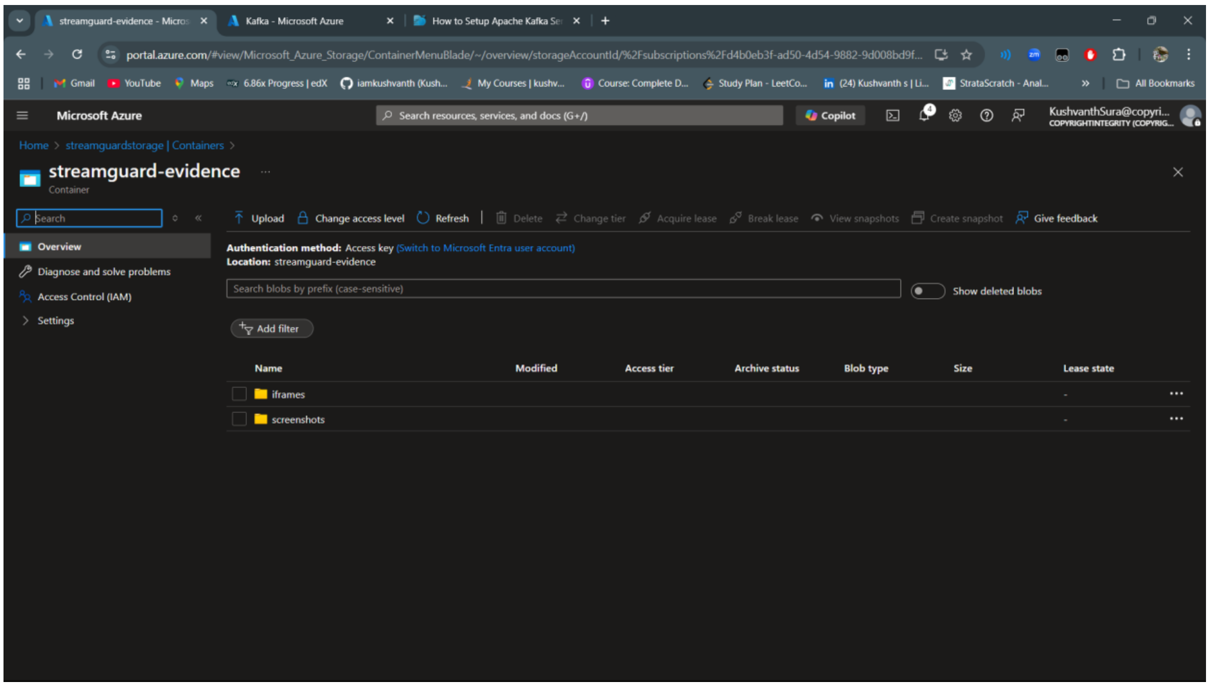
Authentication method: Access key (Switch to Microsoft Entra user account)
Location: streamguard-links / links

Show deleted blobs

Name
[..]
2025-04-06T21:50:58.json
2025-04-06T22:05:25.json
2025-04-06T22:08:22.json
2025-04-06T22:14:01.json

links/2025-04-06T22:14:01.json
Blob

```
{"domain": "https://watch.cricstream.me/t20-streams", "link": "https://watch.cricstream.me/ipl/sunrisers-hyde
```

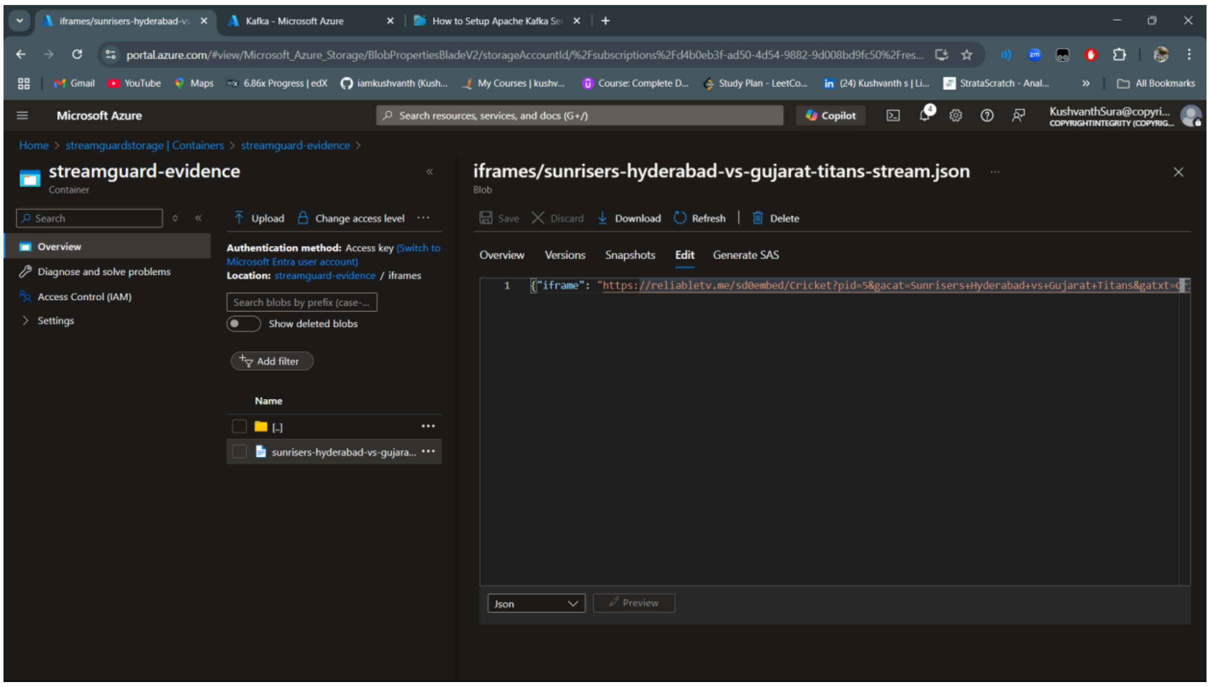


streamguard-evidence
Container

Authentication method: Access key (Switch to Microsoft Entra user account)
Location: streamguard-evidence

Show deleted blobs

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
iframes						-
screenshots						-



streamguard-evidence

Container

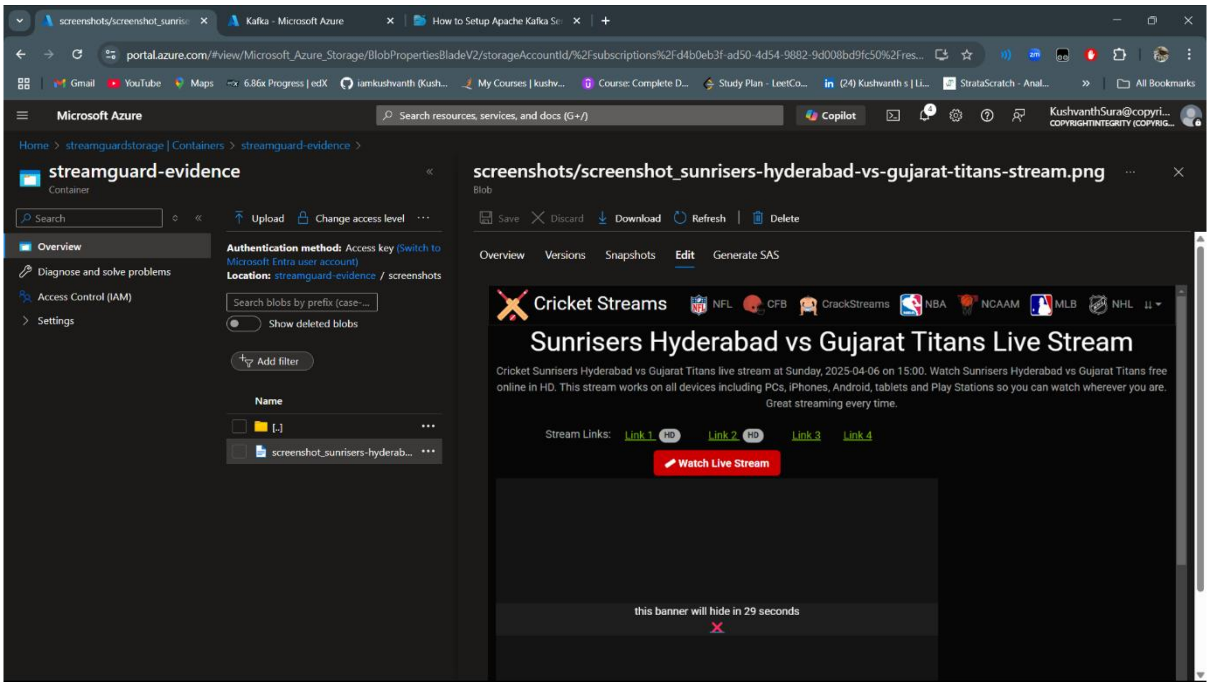
Authentication method: Access key (Switch to Microsoft Entra user account)

Location: streamguard-evidence / iframes

iframes/sunrisers-hyderabad-vs-gujarat-titans-stream.json

Blob

```
{"iframe": "https://reliabletv.me/sd0embed/Cricket?pid=5&gacat=Sunrisers+Hyderabad+vs+Gujarat+Titans&gatxt=C
```



streamguard-evidence

Container

Authentication method: Access key (Switch to Microsoft Entra user account)

Location: streamguard-evidence / screenshots

screenshots/screenshot_sunrisers-hyderabad-vs-gujarat-titans-stream.png

Blob

Cricket Streams

Sunrisers Hyderabad vs Gujarat Titans Live Stream

Cricket Sunrisers Hyderabad vs Gujarat Titans live stream at Sunday, 2025-04-06 on 15:00. Watch Sunrisers Hyderabad vs Gujarat Titans free online in HD. This stream works on all devices including PCs, iPhones, Android, tablets and Play Stations so you can watch wherever you are. Great streaming every time.

Stream Links: Link 1 HD Link 2 HD Link 3 Link 4

this banner will hide in 29 seconds

Way Forward

Extend the "StreamGuard" project to derive actionable insights from collected piracy links by:

1. Applying machine learning (ML) with the Isolation Forest technique to detect anomalies in piracy patterns (e.g., unusual spikes in activity).
2. Creating dashboards to visualize piracy trends by loading data directly from Azure Blob Storage.
3. Analyzing the data in Alteryx to uncover deeper insights (e.g., temporal or geographic trends).