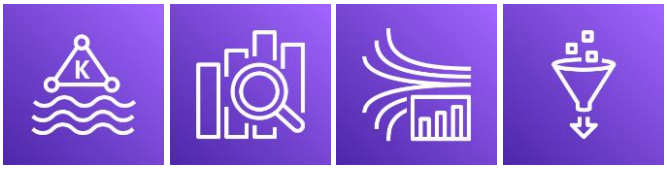

STREAM EVENT PROCESSING AND DATA ANALYTICS WITH AWS MSK-KDA FLINK-GLUE-OSS

ABSTRACT

This solution explores stream processing with AWS MSK as the streaming data persistence layer, Kinesis Data Analytics as the data processing layer, Glue Schema Registry as the data schema central store, serializer and deserializer, and Amazon OpenSearch Service as the data visualization layer. The solution persist mock clickstream data to a topic on MSK after the data schema is validated by Glue. As the data is being persisted a KDA Flink application performs aggregation on certain event types based on tumbling window time of 10s. The aggregated data is then sink to Amazon OSS for visualization. AWS MSK is configured to support both MTLS and IAM authentication and authorization. The IAC delivery tool for this solution is Terraform.

Table of Contents

1.0 Introduction

This guide is a Terraform implementation inspired by the AWS workshop https://catalog.us-east-1.prod.workshops.aws/workshops/976050cc-0606-4b23-b49f-ca7b8ac4b153/en-US/300/350-streaming-with-msk
The solution explores clickstream processing, analysis and visualization for a sample e-commerce application. The clickstream data is mock data generated by this tool - ClickstreamClient-1.0-SNAPSHOT.jar, which is available here https://github.com/aws-samples/clickstream-producer-for-apache-kafka/tree/master , this tool is included as part of the terraform artifact for this solution. Details of how this tool is employed is shown later. As the clickstream data is being generated, it is sent to Glue schema registry for validation of the schema structure and serialization. Once validation succeeds the record is persisted to an Amazon MSK topic clickstream. At this point, the KDA Flink application - ClickstreamProcessor-1.0.jar, which is available here https://github.com/build-on-aws/real-time-streaming-analytics-application-using-apache-kafka, also included as part of the terraform artifact for this solution, starts to consume the clickstream data, deserializing it through Glue schema registry, perform aggregation of the data and then sinks the aggregated data to Amazon OpenSearch Service.

The other aspect of the solution is setting of Mutual TLS between Kafka client and Kafka Cluster. For this we use this java base tool - AuthMSK-1.0-SNAPSHOT.jar, which is available here https://github.com/aws-samples/amazon-msk-client-authentication/tree/master , we have also included this as part of the terraform artifact for this solution.

Thanks to the folks at Amazon who have graciously provided these tools.

2.0 Clone Repository and Initialize Modules and Apply

To begin, please clone the repository to your workspace. Execute the command shown below:

cd /home/dev01 (this workspace may be different in your own case)
git clone https://github.com/sitadconsulting/aws_infra.git
cd aws_infra/projects/proj2 (review the files in this directory and update the "dev.auto.tfvars" file to reflect values from your own environment)

[dev01@houssitdvm02 proj2]$ terraform init
Initializing the backend...
Initializing modules...
- acmpca_ca in ../../modules/acmpca_certificate_authority
- acmpca_ca_certificate in ../../modules/acmpca_certificate_authority_certificate
- acmpca_certificate in ../../modules/acmpca_certificate
- clickstream_registry in ../../modules/glue_registry
- clickstream_schema in ../../modules/glue_schema
- ec2_iam_role in ../../modules/iam_role
- ec2_iam_role_policy_attachment in ../../modules/iam_role_policy_attachment
- eip in ../../modules/eip
- glue_vpc_endpoint in ../../modules/vpc_endpoint
- iam_instance_profile in ../../modules/iam_instance_profile
- internet_gateway in ../../modules/internet_gateway
- internet_gateway_route in ../../modules/route
- kafka_authorization_iam_policy in ../../modules/iam_policy
- kafka_client_instance in ../../modules/instance
- kafka_connect_iam_policy in ../../modules/iam_policy
- kafka_connect_iam_role in ../../modules/iam_role
- kafka_connect_iam_role_policy_attachment in ../../modules/iam_role_policy_attachment
- kafka_connect_target_bucket in ../../modules/s3_bucket
- kafka_connect_vpc_endpoint in ../../modules/vpc_endpoint
- kda_cloudwatch_log_group in ../../modules/cloudwatch_log_group
- kda_cloudwatch_log_stream in ../../modules/cloudwatch_log_stream
- kda_iam_role in ../../modules/iam_role
- kda_iam_role_policy_attachment in ../../modules/iam_role_policy_attachment
- kdaflinkclickstream_application in ../../modules/kinesisanalyticsv2_application
- key_pair in ../../modules/key_pair
- launch_template in ../../modules/launch_template
- msk_broker_log_group in ../../modules/cloudwatch_log_group
- msk_cluster in ../../modules/msk_cluster
- msk_configuraion in ../../modules/msk_configuration
- msk_kms_key in ../../modules/kms_key
- nat_gateway in ../../modules/nat_gateway
- nat_gateway_route in ../../modules/route
- opensearch_service in ../../modules/opensearch_domain
- opensearch_vpc_endpoint in ../../modules/opensearch_vpc_endpoint
- private_route_table in ../../modules/route_table
- private_route_table_association in ../../modules/route_table_association
- private_subnet_1a in ../../modules/subnet
- private_subnet_1b in ../../modules/subnet
- private_subnet_1c in ../../modules/subnet

- public_route_table in ../../modules/route_table
- public_route_table_association in ../../modules/route_table_association
- public_subnet_1a in ../../modules/subnet
- public_subnet_1b in ../../modules/subnet
- public_subnet_1c in ../../modules/subnet
- s3_object in ../../modules/s3_object
- s3_object_file_upload in ../../modules/s3_object
- s3_object_jar_file_upload in ../../modules/s3_object
- sales_db_parameter_group in ../../modules/db_parameter_group
- sales_db_rds_cluster in ../../modules/rds_cluster
- sales_db_rds_cluster_instance in ../../modules/rds_cluster_instance
- sales_db_rds_cluster_parameter_group in ../../modules/rds_cluster_parameter_group
- sales_db_subnet_group in ../../modules/db_subnet_group
- security_group in ../../modules/security_group
- streaming_artifacts_bucket in ../../modules/s3_bucket
- vpc in ../../modules/vpc
- vpc_dhcp_options in ../../modules/vpc_dhcp_options
- vpc_dhcp_options_assoc in ../../modules/vpc_dhcp_options_assoc
- vpc_security_group_rule in ../../modules/vpc_security_group_rule

Initializing provider plugins...
- Finding latest version of hashicorp/template...
- Finding hashicorp/local versions matching "~> 2.3.0"...
- Finding hashicorp/aws versions matching "5.16.0"...
- Installing hashicorp/template v2.2.0...
- Installed hashicorp/template v2.2.0 (signed by HashiCorp)
- Installing hashicorp/local v2.3.0...
- Installed hashicorp/local v2.3.0 (signed by HashiCorp)
- Installing hashicorp/aws v5.16.0...
- Installed hashicorp/aws v5.16.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider selections it made above. Include this file in your version control repository so that Terraform can guarantee to make the same selections by default when you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see any changes that are required for your infrastructure. All Terraform commands should now work.

If you ever set or change modules or backend configuration for Terraform, rerun this command to reinitialize your working directory. If you forget, other commands will detect it and remind you to do so if necessary.

proj2]$ ***terraform apply***

Once terraform apply execution completes, terraform outputs some values, we will use the output shown below:

acmpca_ca_arn – required to set up MTLS between Kafka clients and the Cluster

3.0 Setup Mutual TLS for Kafka client and MSK Cluster.

Get logged on the kafka client instance

[dev01@houssitdvm02 proj2]$ ssh -i "kafka_client_instance-key.pem" ec2-user@ec2-52-87-157-83.compute-1.amazonaws.com
Warning: Identity file kafka_client_instance-key.pem not accessible: No such file or directory.

```
  ,     #_
  ~\_  ####_        Amazon Linux 2023
 ~~  \_#####\
 ~~     \###|
 ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
  ~~       V~' '->
   ~~~         /
     ~~._.   _/
        _/ _/
      _/m/'
```

Last login: Fri Sep 29 13:00:42 2023
[ec2-user@ip-10-0-1-181 ~]$

Switch to kafka user

[ec2-user@ip-10-0-1-181 ~]$ sudo su - kafka
Last login: Fri Sep 29 13:01:06 UTC 2023 on pts/0
[kafka@ip-10-0-1-181 ~]$

[kafka@ip-10-0-1-181 ~]$ java -jar utilities/AuthMSK-1.0-SNAPSHOT.jar -caa <terraform output acmpca_ca_arn> -ksp cl3z0n3p@s -ksa msk -ksl /etc/kafka/security/ssl/tls/keystore/kafka.client.keystore.jks -cfv 5 -pkf /etc/kafka/security/ssl/private_key -ccf /etc/kafka/security/ssl/certificate >mskauth.out 2>mskauth.err

Create truststore

[kafka@ip-10-0-1-181 ~]$ find /usr/lib/jvm/ -name "cacerts" -exec cp {} /etc/kafka/security/ssl/tls/truststore/kafka.client.truststore.jks \;

4.0 Create Kafka topics for persisting data and generate mock clickstream data

Create clickstream topic

[kafka@ip-10-0-1-181 ~]$ bin/kafka-topics.sh --create --bootstrap-server $MSK_BOOTSTRAP_IAM_ADDRESS --replication-factor 3 --partitions 1 --topic clickstream --command-config /etc/kafka/security/iam/client.properties
Created topic clickstream.

Create topics where KDA Flink application would send processed data

[kafka@ip-10-0-1-181 ~]$ bin/kafka-topics.sh --create --bootstrap-server $MSK_BOOTSTRAP_IAM_ADDRESS --replication-factor 3 --partitions 1 --topic Departments_Agg --command-config /etc/kafka/security/iam/client.properties
Created topic Departments_Agg.

[kafka@ip-10-0-1-181 ~]$ bin/kafka-topics.sh --create --bootstrap-server $MSK_BOOTSTRAP_IAM_ADDRESS --replication-factor 3 --partitions 1 --topic ClickEvents_UserId_Agg_Result --command-config /etc/kafka/security/iam/client.properties
Created topic ClickEvents_UserId_Agg_Result.

[kafka@ip-10-0-1-181 ~]$ bin/kafka-topics.sh --create --bootstrap-server $MSK_BOOTSTRAP_IAM_ADDRESS --replication-factor 3 --partitions 1 --topic User_Sessions_Aggregates_With_Order_Checkout --command-config /etc/kafka/security/iam/client.properties
Created topic User_Sessions_Aggregates_With_Order_Checkout.

Start KDA Flink application

Get log on to the KDA console and start the KDAFlinkClickstream application.

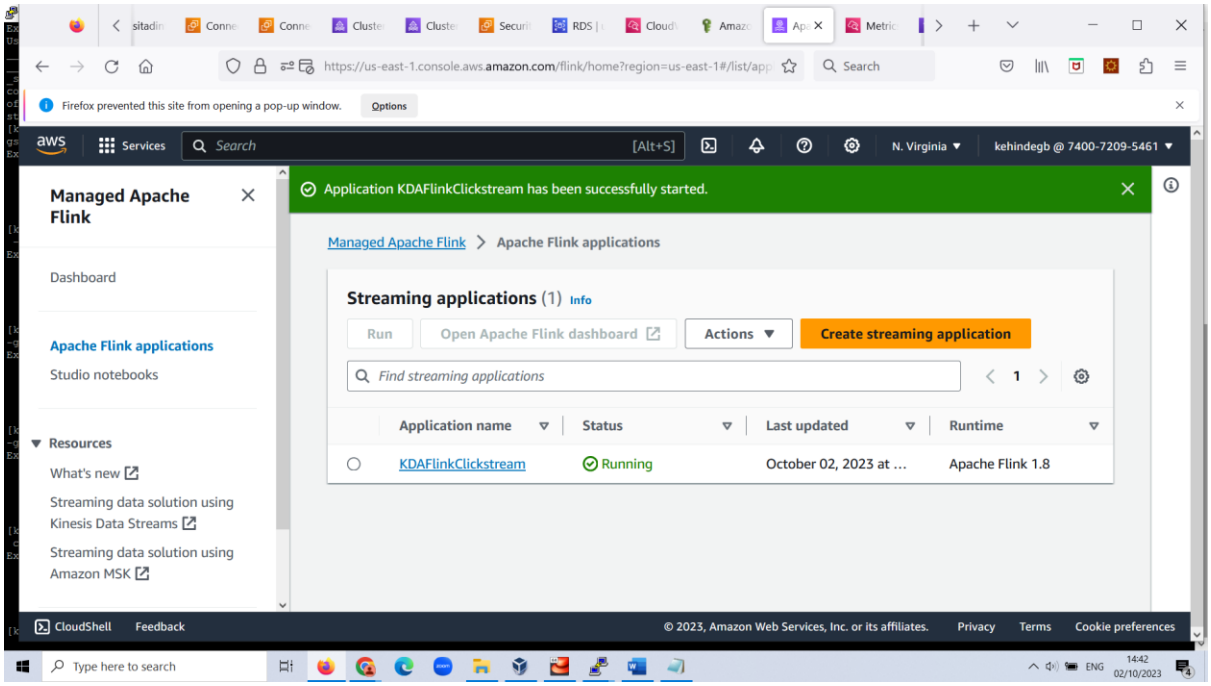

Fig 1: Running instance of the KDA Flink Application

Once the application is running, proceed to generating the mock clickstream data.

Generate mock clickstream data

[kafka@ip-10-0-1-181 ~]$ java -jar utilities/ClickstreamClient-1.0-SNAPSHOT.jar -t clickstream -pfp /etc/kafka/security/iam/producer.properties -nt 8 -rf 900 -iam -gsr -gsrr "us-east-1" -grn "clickstream" -gsn "clickstream" -gsd "Clickstream schema" -gar -gcs "FULL"

Whiles the clickstream data is being generated access Amazon OpenSearch Console to gain real-time visualization of the aggregated clickstream events processed by the KDA Flink application.

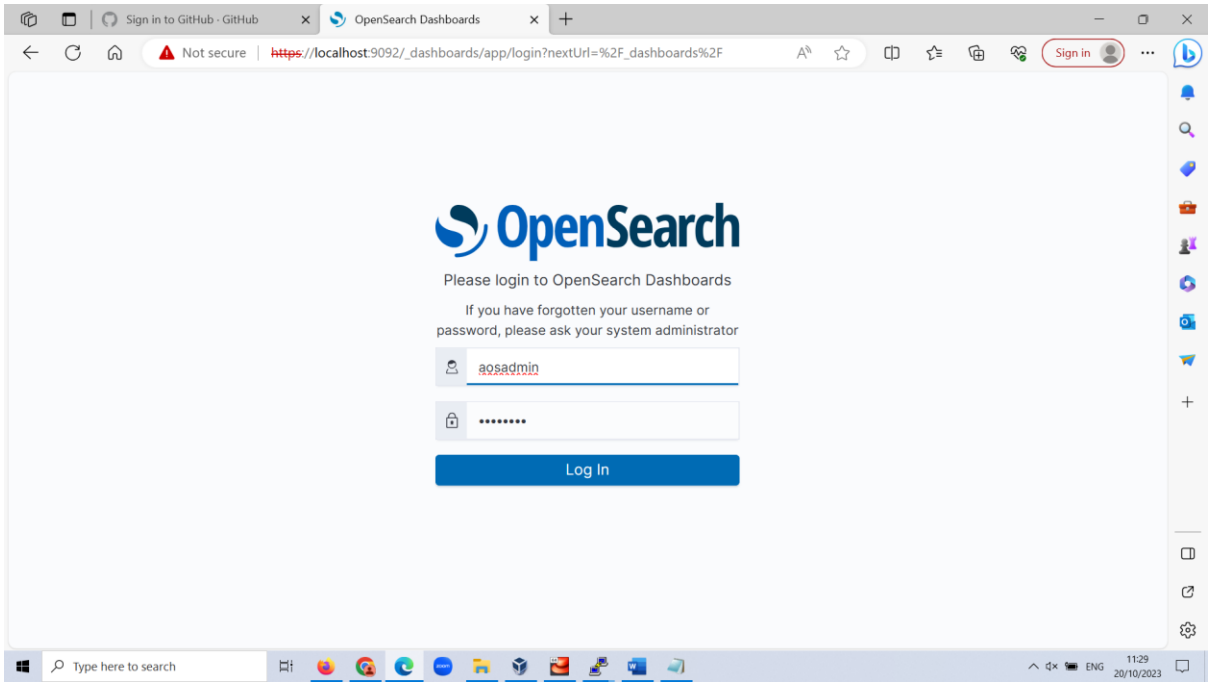

Fig 2: OpenSearch log on screen (accessed by ssh tunnelling through kafka client instance)

Below are some of the outputs:

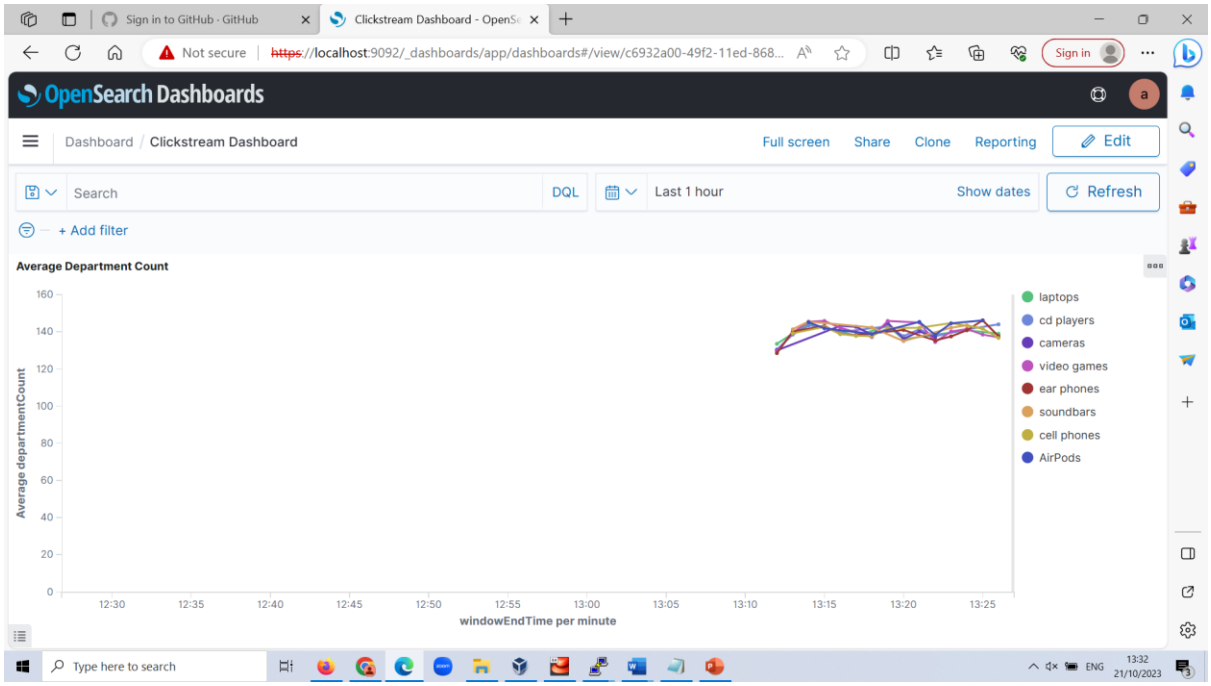

Fig 3. Average Department Count

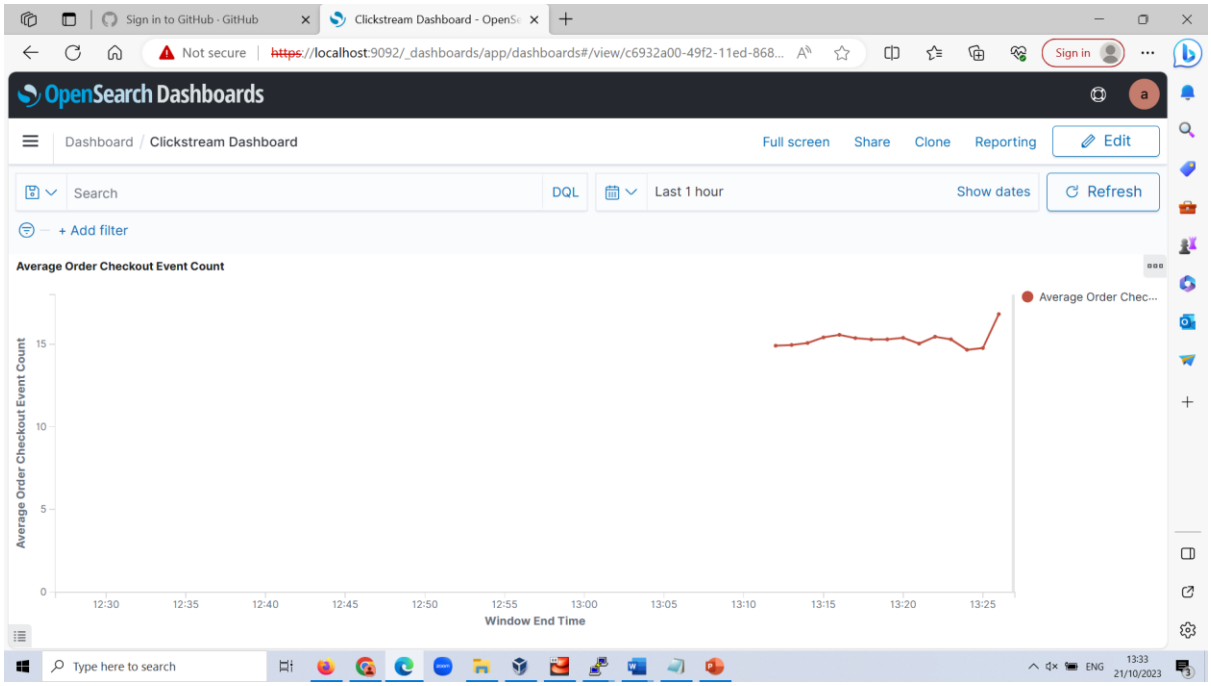

Fig 4: Average Order Checkout Event Count

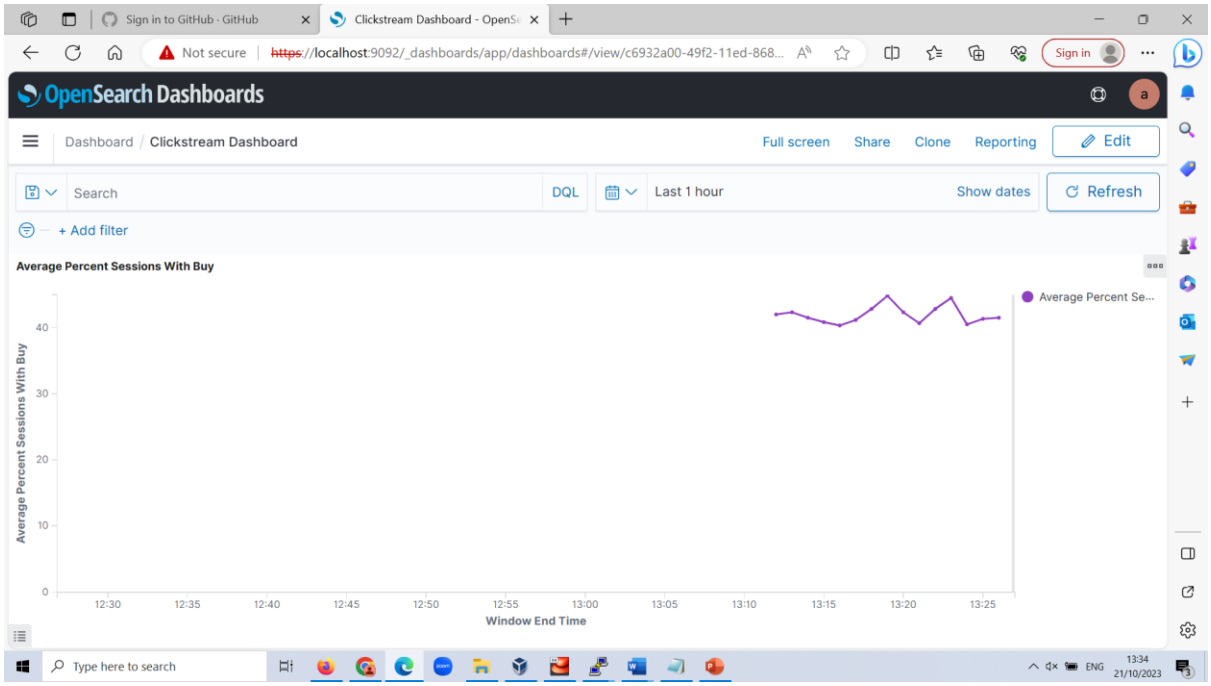

Fig 5: Average Percentage Sessions with Buy

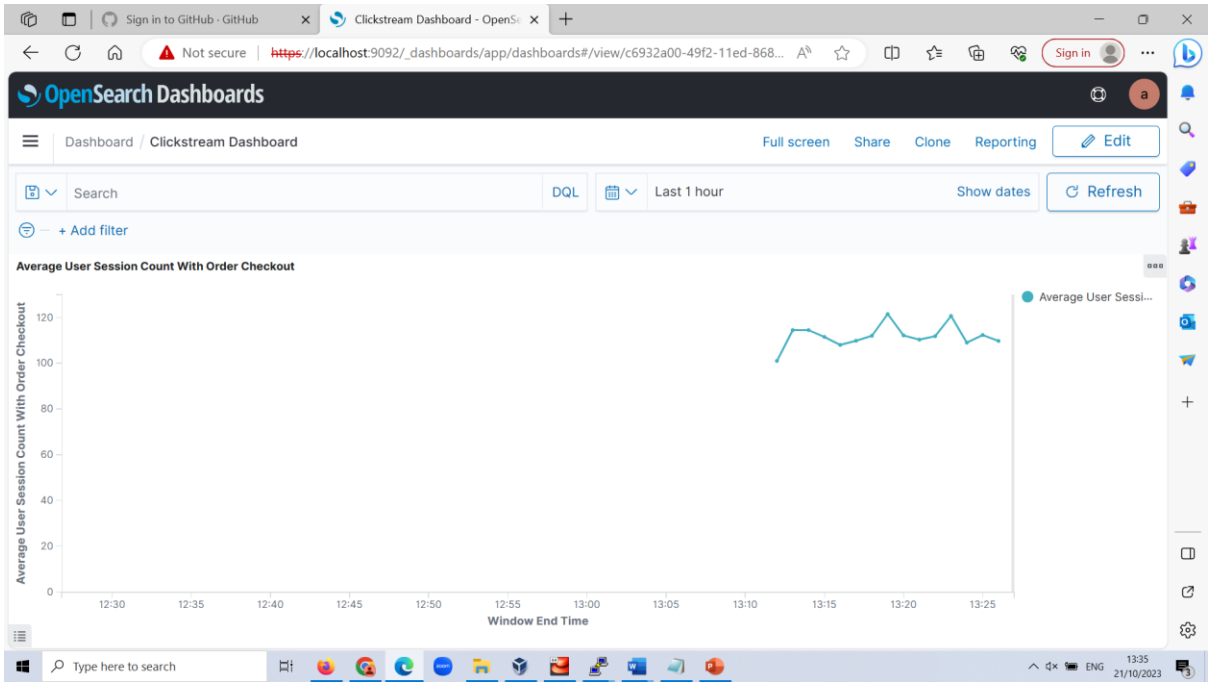

Fig 6: Average User Session Count With Order Checkout

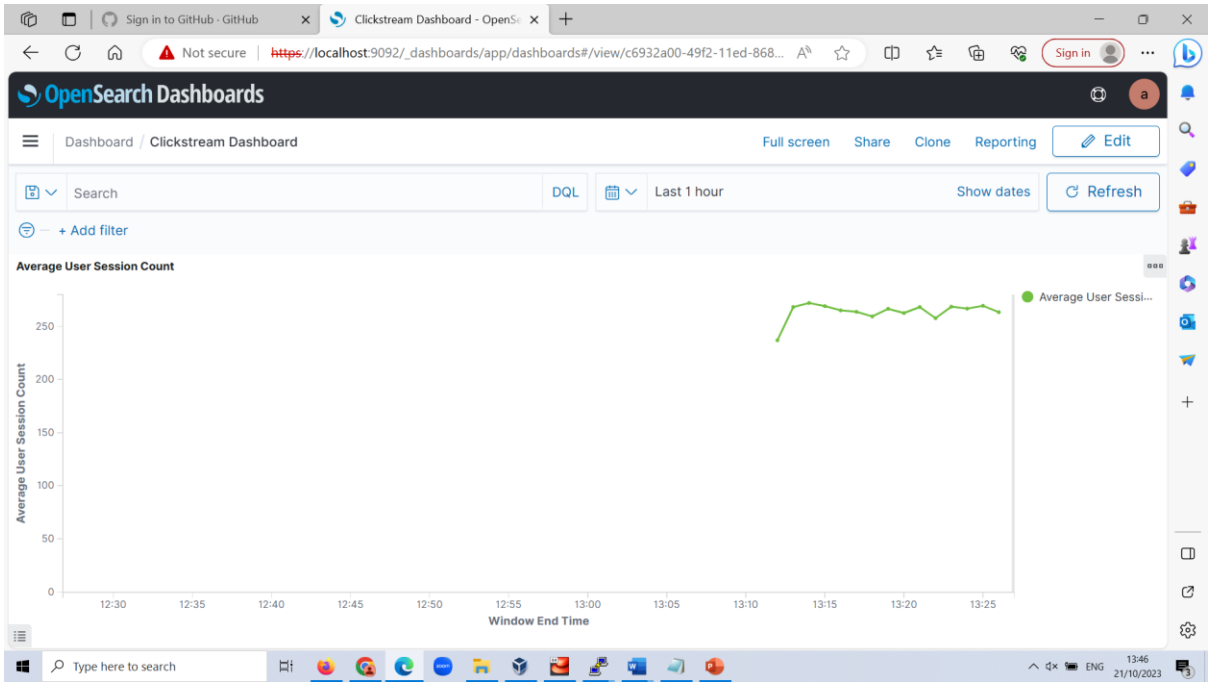

Fig 7: Average User Session Count

Appendix

For completeness we provide some screen shots of other aspect of the solution below:

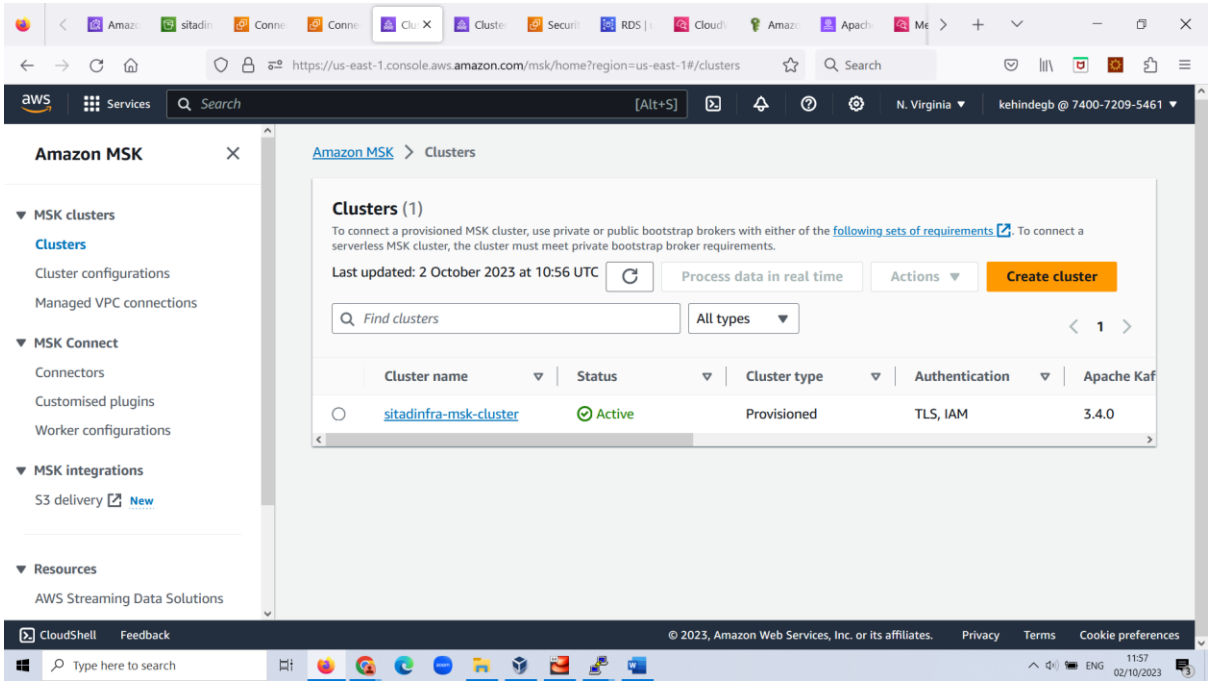

Fig 8: MSK Cluster with TLS and IAM Authentication

Fig 9: Flink Streaming Job Run Graph

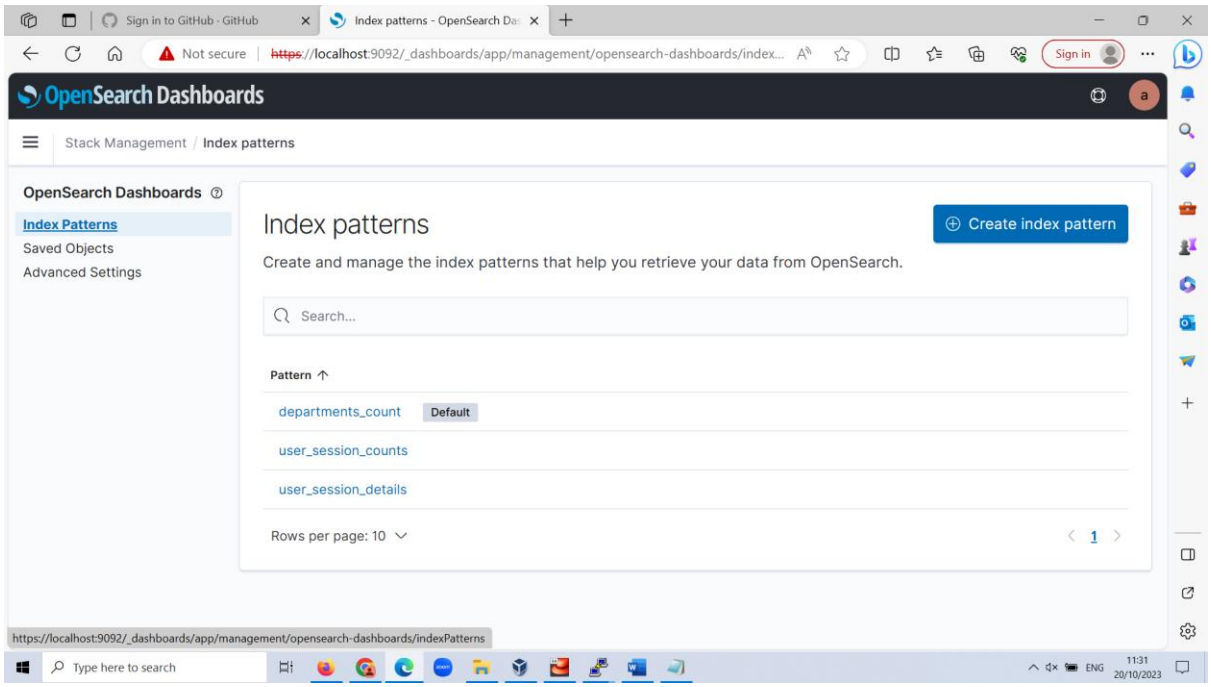

Fig 10: Index patterns – showing the aggregated events populated by KDA Flink Application

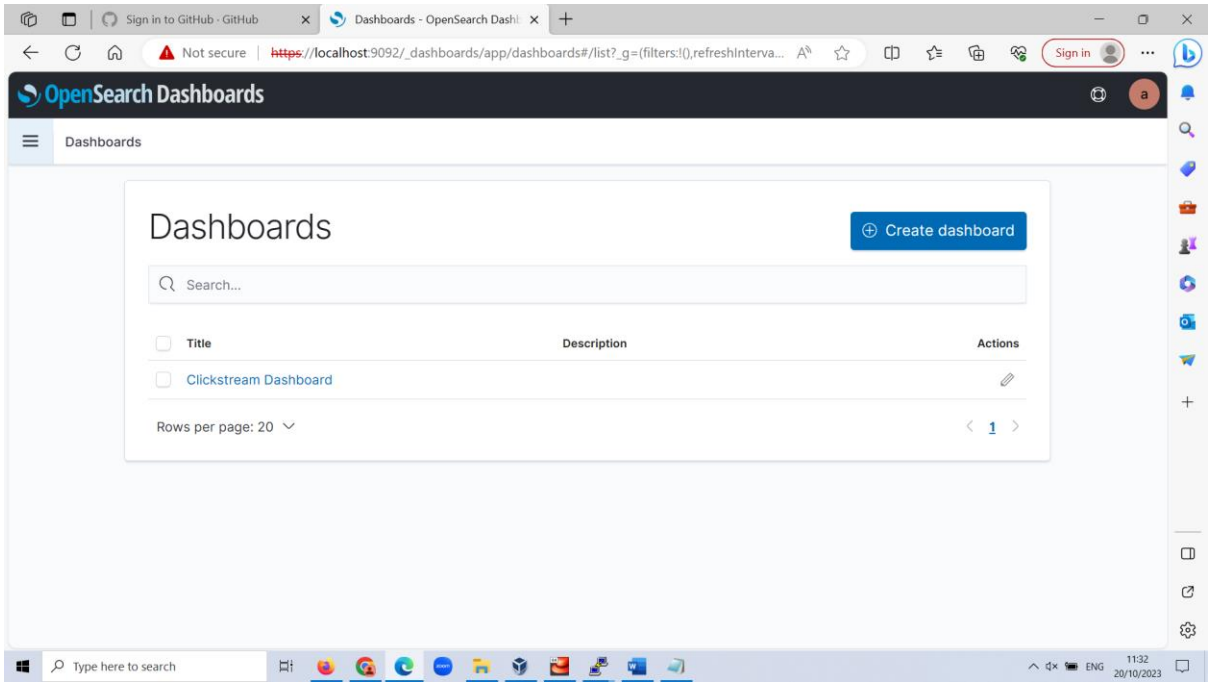

Fig 11: The Clickstream Dashboard

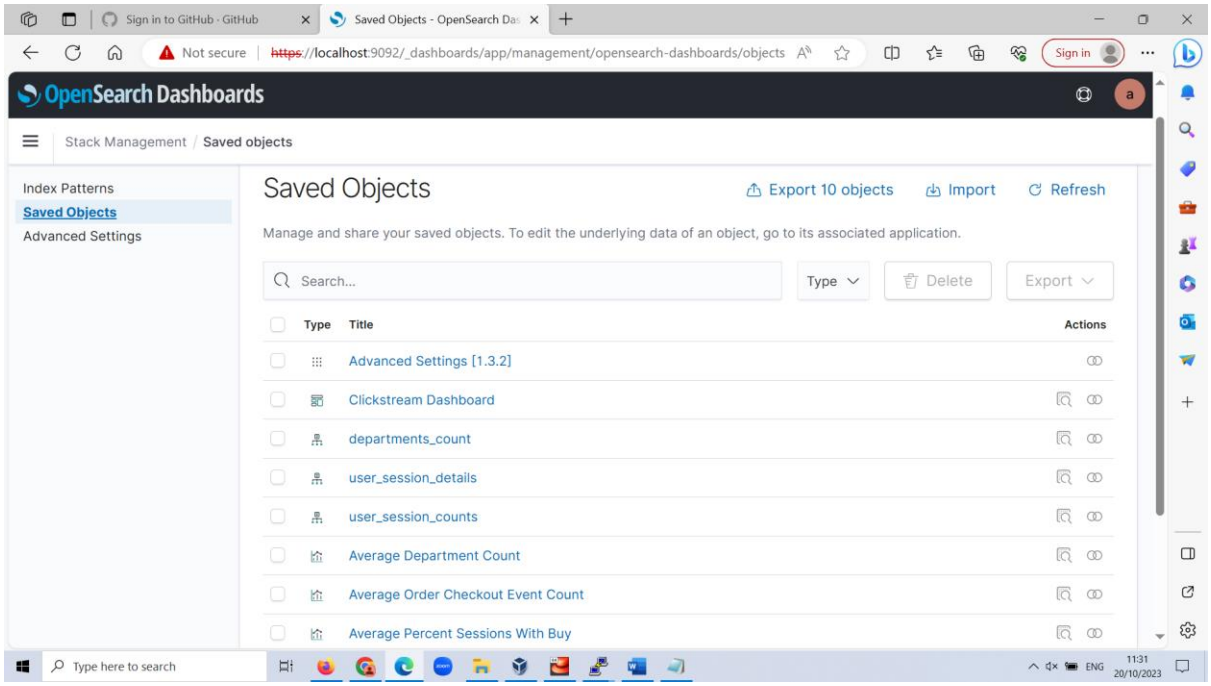

Fig 12: Saved Objects