# 

# 

# 

Data Ingestion

(Project-1: Using Kafka, Kafka Connect with S3 Sink)

Aditya Dhanraj Shivkar

TABLE OF CONTENTS

[Overview: (In-progress):](#_4a4l7e1u6llb) **5**

[**Data Prep:**](#_1kantns9r50c) **7**

[Setup: (In-progress)](#_29o6gwxk4pl9) **7**

[Start MSK cluster:](#_3ws0iw5bh9rw) 7

[IAM creation for EC2 and MskCluster](#_q0iwc46gg6q2) 12

[Policy Creation (customer inline):](#_213slh2plduo) 12

[Creating 1st Policy:](#_b42dfyti2lgz) 12

[Creating 2nd Policy:](#_fobnroaugz1p) 14

[Creating 3rd Policy:](#_f59f0q7bx3kl) 15

[Creating 4th Policy:](#_1pu2l037jguy) 18

[Role creation:](#_u1pjrliwo5na) 19

[Creating 1st Role:](#_rp4hqymmxf94) 19

[Creating 2nd Role:](#_93mk099tcidf) 20

[Kafka client in EC2:](#_6pfly8b8nc9e) 20

[How to connect to virtual machine(EC2) via MobaXterm:](#_rmgonl3qrivd) 24

[Setting up the requirements in client machine(say MobaXterm):](#_au734ivtoj53) 26

[Create Custom plugin:](#_9sdsj7nuc8r) 28

[Creating S3 Custom Plugin: (Note: Use this when creating connector)](#_2jv8jtx8n9lw) 28

[Creating Glue Custom Plugin:](#_n49c6dh490ng) 29

[\*Setup\* (For Json Schema):](#_m5dtbr5frytc) 29

[Create S3 Bucket:](#_csyku1vxx0nr) 34

[Adding required Jars in S3 bucket:](#_tc9py8sanalk) 34

[Create MSK Connect connector:](#_4uhu62v9yifu) 35

[Creating VPC Endpoints:](#_d08bvzsxdj4j) 35

[Creating S3 endpoint:](#_gcpxt0z7l8pu) 35

[Creating Glue endpoint:](#_uhh9eimq7ogr) 37

[Creating AWS Schema and Registry( For JsonSchema ):](#_5w5fl0rscj35) 40

[End-to-End Run(15 mins):](#_trg5gouepual) **41**

[Json](#_mrreq3mt7lhk) 41

[Json Producer:](#_mqyto9pwjcy7) 41

[Using CLI:](#_11g8jbtqfk97) 41

[Using Java Program:](#_25phplx8fhfk) 41

[Json Consumer:](#_f9ppwus1n4h) 43

[Using CLI:](#_upbt51owfkw8) 43

[Using Java Program:](#_vi1g8kyckzdk) 43

[Using Kafka connect for Json:](#_gkm2sa5asnbe) 43

[JsonShema to Parquet:](#_ar7xleyd2i22) 44

[JSON Schema Producer:](#_sdi8wq2b35ty) 44

[Using CLI:](#_khgprdklep57) 44

[Using Java Program:](#_ejjuhu7ya77w) 44

[Json Schema Consumer:](#_fwr5nlj8lecc) 49

[Using CLI:](#_ahfbogqv018) 49

[Using Java Program:](#_d4ypy5grazi5) 49

[Using kafka connect S3 sink for Json Schema:](#_ff4wzgfm5qxj) 51

[Avro](#_ov3mt6xafr27) 52

[Avro Producer](#_crki5mxsgzih) 52

[Using CLI:](#_b5nlulbnbwpp) 52

[Using Java Program](#_we4ez841qnf6) 52

[Payment.avsc for Avro](#_pcrs0jof2keh) 55

[Avro Consumer](#_2xi5qi4kcp4) 56

[Using CLI:](#_bxj0fkozicx) 56

[Using Java Program](#_zcswp5kweepm) 56

[Using kafka connect S3 sink for Avro:](#_vr3oya4wrtc) 57

[ProtoBuf](#_wm4fhfqdqswp) 59

[ProtoBuf Producer:](#_pi12e9hvhto7) 59

[ProtoBuf Consumer:](#_1dwld3v90pxy) 59

[**MSK Connect Logs:**](#_cogocnm64jrb) **59**

[For Analytics options: (Athena, Pyspark)](#_y249br30m7p6) **59**

[Analytics using AWS Athena](#_a99fhxmlt1qq) 59

[Json](#_4p0cwng7j3e) 59

[JsonSchema](#_rcets83pkg80) 59

[Avro](#_9ki13juf9fls) 60

[ProtoBuf](#_ljzqahom1go1) 61

[AWS Billing:](#_epwey5o99l85) **61**

[Metrics:](#_6r8kuv7jmu91) **62**

[MSK cluster metrics:](#_yc1ggzflzc0x) 62

[MSK Connect connectors metrics:](#_lg94psw6n0ki) 62

[Errors and Resolutions](#_41608z4v6h48) **62**

[● When s3 endpoint is not mentioned](#_ee78nby57z5z) 62

[● When glue endpoint is not mentioned.](#_4ljuxcn7wkyr) 62

[● When the wrong message format is sent.](#_3rkja4ohf8wt) 76

[● When the proper jar is not available in plugin jar/zip.](#_5q703ifri1k3) 76

[● The IAM setup is not properly set up in the Kafka client.](#_2d556jkfqley) 76

[● Proper access to MSKC is not provided in IAM.](#_y4xg83ce37hq) 76

[Clean up:](#_hcc4mneuk0x0) **76**

[**What is IAM?**](#_3mzq4wv) **78**

[**What is Kafka(AWS MSK Kafka cluster)?**](#_2250f4o) **79**

[**What is Kafka Connect?**](#_1gf8i83) **80**

[**What is Glue?**](#_40ew0vw) **80**

[**What is VPC?**](#_2fk6b3p) **81**

[**What is EC2?**](#_upglbi) **82**

[**What is S3?**](#_3ep43zb) **84**

[References](#_r8cr2fen0xux) **84**

## 

## 

## 

## 

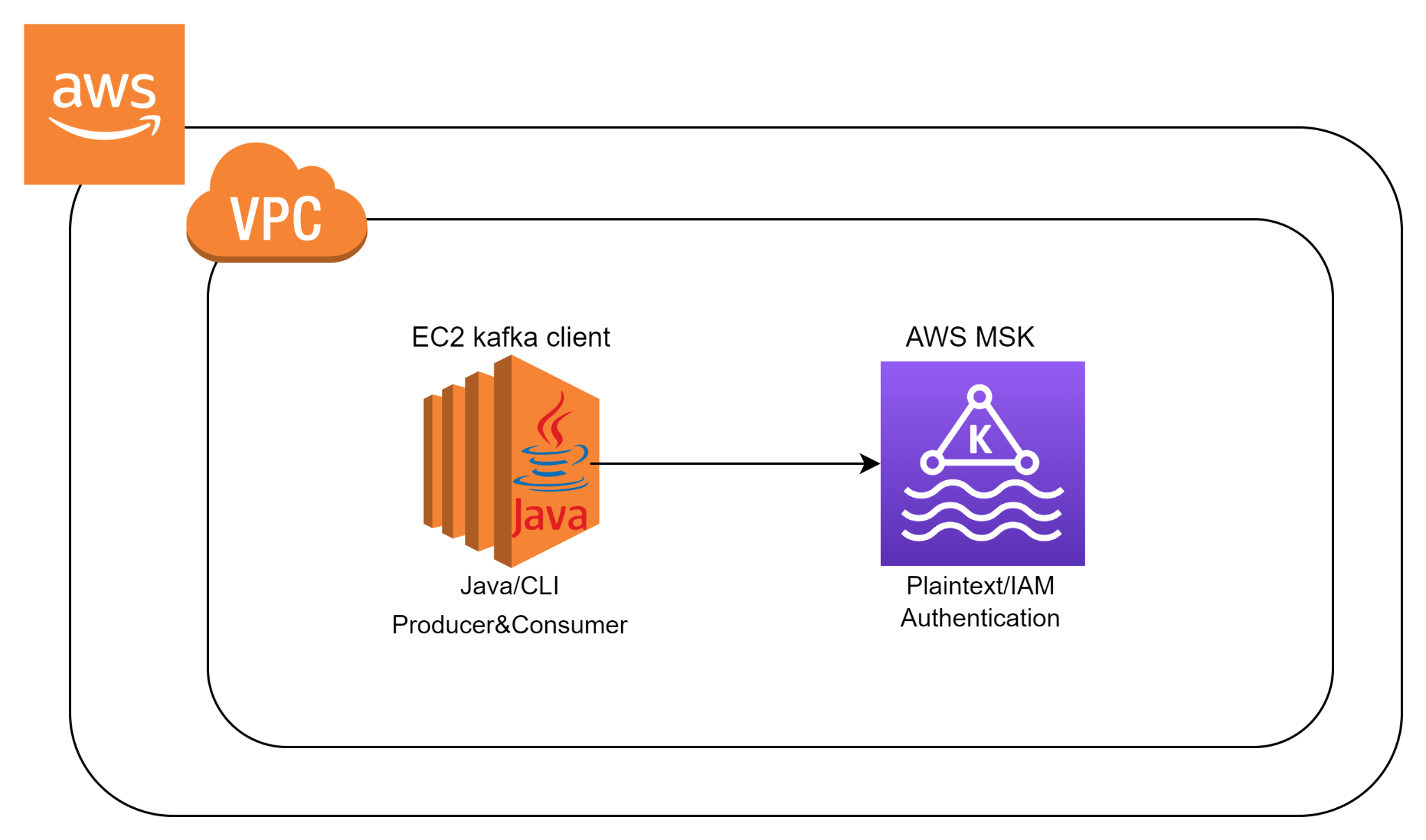
## 

## Overview: (In-progress):

It is divided into two parts:

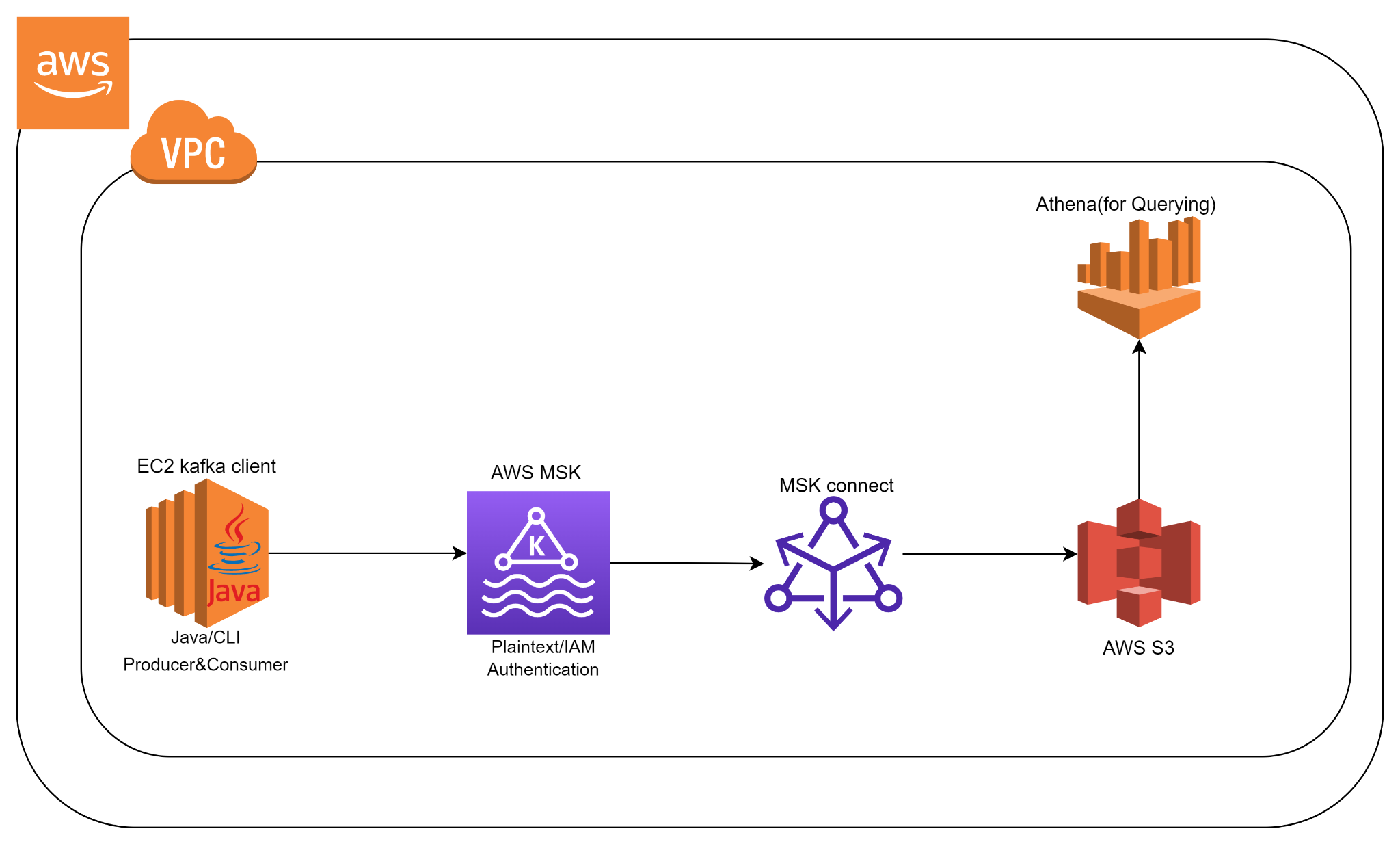
* A Minor project, and
* A Minor project.

Minor Project:



1. Here, using Java Program (or) by using CLI commands, we produce and Consume the data messages in the EC2(CLI) itself.
2. We can perform the above operation by taking Plaintext, as well as using IAM authentication(SASL).

Major Project:



* Here, Data Ingestion is carried in a way that the data ingested into the S3 bucket.
* We produce the data in EC2 using CLI commands (or) Java program, and the MSK connect acts as a consumer which in turn stores the data in S3.
* Reading data from the S3 can be done with the help of Athena, by writing suitable queries.

## Data Prep:

## 

## **Setup**: (In-progress)

Note: You need to have an AWS account to work on this project. If you don’t have one, please open and follow the steps mentioned here. [Creating a Free AWS account](https://youtu.be/P7hVdusJF7I)

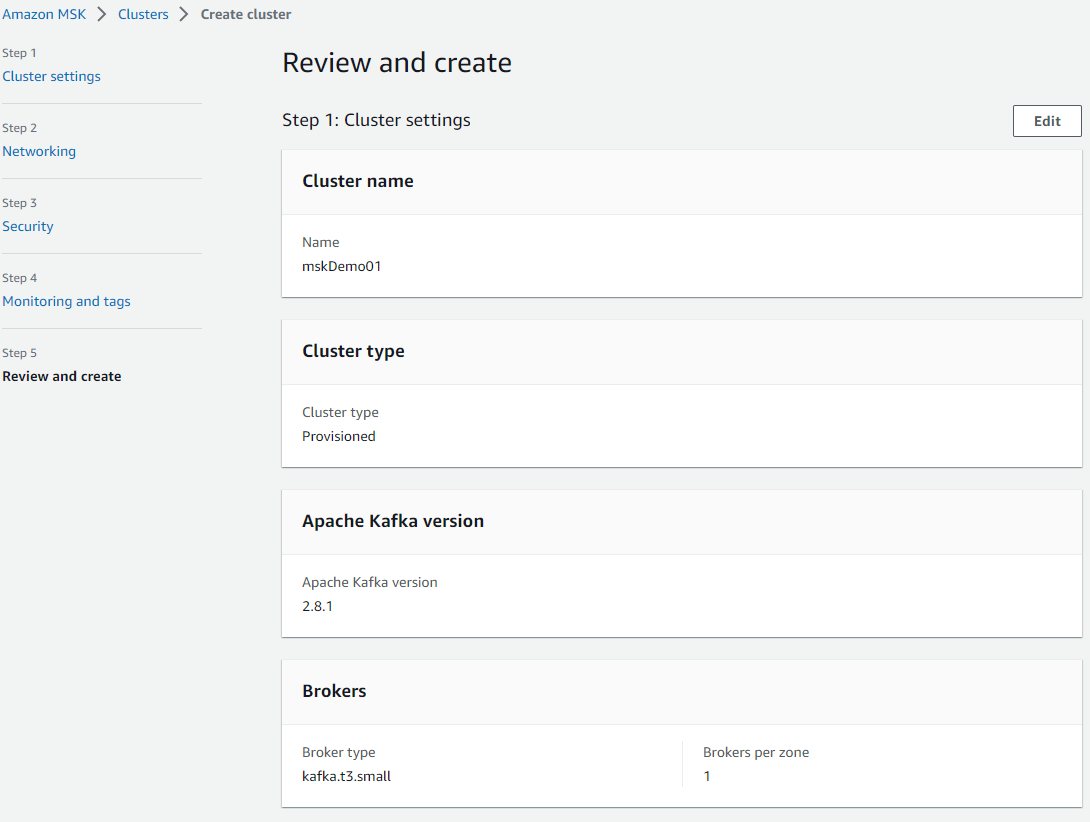
Open the AWS console in your browser.

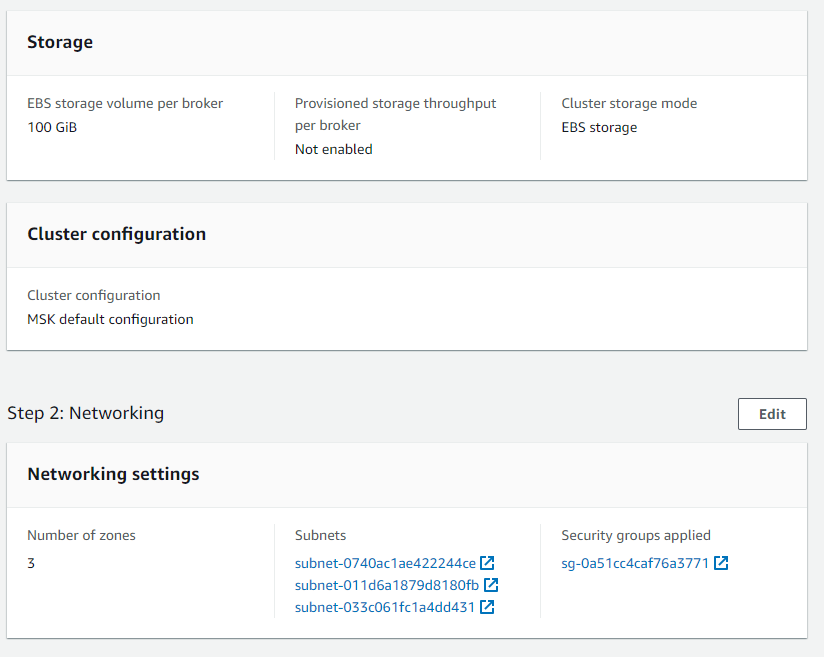
### **Start MSK cluster:**

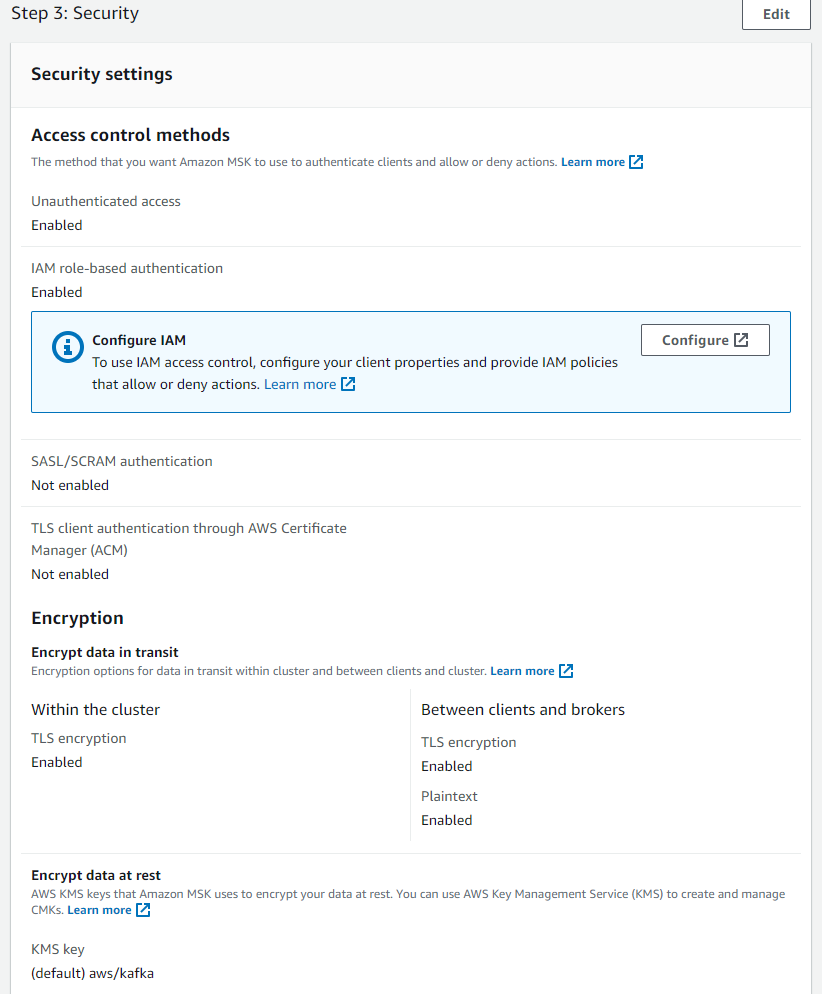
1. Click on Services on the top left corner in your AWS console.
2. Search for MSK, and click on it to open.
3. Now, click on Create cluster.
4. Choose the creation method as Custom create.
5. Give a name to the cluster.(and give it a name lets say; mskDemo01)
6. Select the cluster type as Provisioned.
7. Select the Broker type as Kafka.t3.small.
8. Let the number of zones and EBS be default(recommended).
9. Make Storage as 100GB.(Just to avoid the cost)
10. In the Networking step, select VPCs and Zones.
11. Create a Security Group, (let's say mskDemo01SG) , and select the same security group and proceed follow the below step to create a Security Group..
12. Create the Security group with inline bounds and outline bounds as :

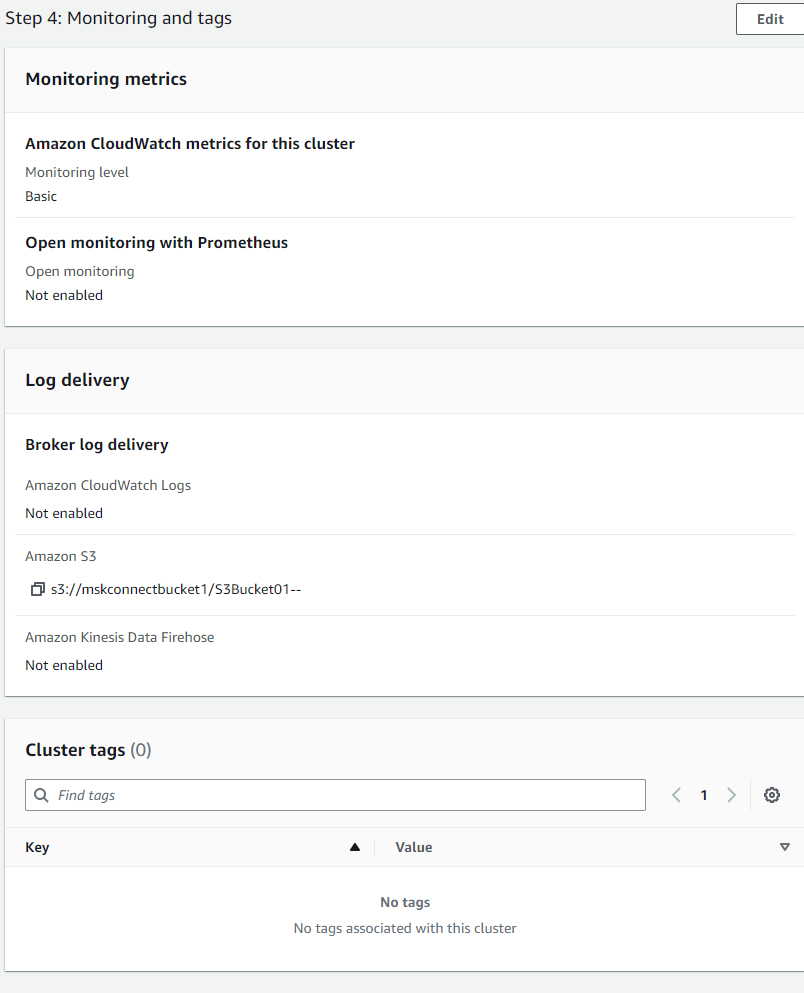
IP version: IP4 and Type: All Traffic

1. In the next page.
2. In the next page choose “Unauthenticated access” and “IAM role-based authentication”.
3. In Encryption select “Plaintext”.
4. For S3 bucket creation refer [Create S3 Bucket](#_csyku1vxx0nr).
5. In broker log delivery select Deliver to Amazon S3, browse and select the S3 Bucket.
6. Click Next.
7. Leave all the steps as default and click on Create cluster.
8. The amount of time required to create a provisioned cluster depends on the size of the cluster. A typical provisioned cluster takes up to 15 min - 30 mins to create.









### **IAM creation for EC2 and MskCluster**

#### Policy Creation (customer inline):

(4 Policies required in total)

##### Creating 1st Policy:

1. From the AWS console, search for IAM, and click on it to open.
2. Now, from the left navigation bar click on Policies .
3. Click on Create policy.
4. Now, select JSON.
5. Paste the following JSON in the box.

| {  "Version": "2012-10-17",  "Statement": [  {  "Sid": "VisualEditor0",  "Effect": "Allow",  "Action": [  "kafka:CreateConfiguration",  "kafka:UpdateClusterKafkaVersion",  "kafka:UpdateBrokerCount",  "kafka:DeleteConfiguration",  "kafka:UpdateClusterConfiguration",  "kafka:ListClusters",  "kafka:CreateClusterV2",  "kafka:UpdateConnectivity",  "kafka:ListScramSecrets",  "kafka:ListKafkaVersions",  "kafka:GetBootstrapBrokers",  "kafka:UpdateBrokerStorage",  "kafka:RebootBroker",  "kafka:DescribeClusterV2",  "kafka:UpdateMonitoring",  "kafka:ListConfigurationRevisions",  "kafka:UpdateStorage",  "kafka:UpdateSecurity",  "kafka:UpdateBrokerType",  "kafka:CreateCluster",  "kafka:GetCompatibleKafkaVersions",  "kafka:DescribeCluster",  "kafka:UpdateConfiguration",  "kafka:ListConfigurations",  "kafka:BatchDisassociateScramSecret",  "kafka:BatchAssociateScramSecret",  "kafka:ListClustersV2",  "kafka:DescribeClusterOperation",  "kafka:ListNodes",  "kafka:DeleteCluster",  "kafka:ListClusterOperations"  ],  "Resource": "\*"  },  {  "Sid": "VisualEditor1",  "Effect": "Allow",  "Action": [  "kafka:\*",  "iam:PassRole"  ],  "Resource": "\*"  },  {  "Sid": "VisualEditor2",  "Effect": "Allow",  "Action": "kafka-cluster:\*",  "Resource": "\*"  }  ]  } |
| --- |

1. Click Next: Tags.
2. Click Next: Review.
3. Give name of the policy ( say mskPolicy).
4. Finally click on Create policy.

##### Creating 2nd Policy:

Follow the same steps as in [Creating 1st Policy](#_b42dfyti2lgz) but replace the JSON in step5 with the JSON given below and name the policy as msk1.

Note: Make sure you mention your cluster name in the JSON below, we are assuming our cluster name as mskDemo1 below.

| {  "Version": "2012-10-17",  "Statement": [  {  "Action": [  "kafka-cluster:DescribeCluster",  "kafka-cluster:AlterCluster",  "kafka-cluster:Connect"  ],  "Resource": "arn:aws:kafka:us-west-2:593035604125:cluster/mskDemo01/\*",  "Effect": "Allow",  "Sid": "kafkacluster"  },  {  "Action": [  "kafka-cluster:\*Topic\*",  "kafka-cluster:ReadData",  "kafka-cluster:WriteData"  ],  "Resource": "arn:aws:kafka:us-west-2:593035604125:topic/mskDemo01/\*",  "Effect": "Allow",  "Sid": "kafkatopic"  },  {  "Action": [  "kafka-cluster:AlterGroup",  "kafka-cluster:DescribeGroup"  ],  "Resource": "arn:aws:kafka:us-west-2:593035604125:group/mskDemo01/\*",  "Effect": "Allow",  "Sid": "kafkagroup"  },  {  "Action": [  "kafka:DescribeCluster",  "kafka:GetBootstrapBrokers",  "kafka:DescribeConfiguration",  "kafka:ListConfigurations",  "kafka:CreateConfiguration"  ],  "Resource": "arn:aws:kafka:us-west-2:593035604125:\*",  "Effect": "Allow",  "Sid": "kafkageneral"  },  {  "Sid": "iampassrole",  "Effect": "Allow",  "Action": [  "kafka:\*",  "iam:PassRole"  ],  "Resource": "\*"  },  {  "Sid": "VisualEditor12",  "Effect": "Allow",  "Action": "kafka-cluster:\*",  "Resource": "\*"  }  ]  } |
| --- |

##### Creating 3rd Policy:

Follow the same steps as in [Creating 1st Policy](#_b42dfyti2lgz) but replace the JSON in step5 with the JSON given below and name the policy as mskcPolicy.

Note: Make sure you mention your cluster name in the JSON below, we are assuming our cluster name as mskDemo1 below.

| {  "Version": "2012-10-17",  "Statement": [  {  "Action": [  "iam:PassRole",  "kafka-cluster:DescribeCluster",  "kafka-cluster:AlterCluster",  "kafka-cluster:Connect"  ],  "Resource": [  "arn:aws:kafka:us-west-2:593035604125:cluster/mskDemo01/\*",  "arn:aws:iam::593035604125:role/mskc"  ],  "Effect": "Allow",  "Sid": "kafkacluster"  },  {  "Action": [  "kafka-cluster:\*Topic\*",  "kafka-cluster:ReadData",  "kafka-cluster:WriteData"  ],  "Resource": "arn:aws:kafka:us-west-2:593035604125:topic/mskDemo01/\*",  "Effect": "Allow",  "Sid": "kafkatopic"  },  {  "Action": [  "kafka-cluster:AlterGroup",  "kafka-cluster:DescribeGroup"  ],  "Resource": "arn:aws:kafka:us-west-2:593035604125:group/mskDemo01/\*",  "Effect": "Allow",  "Sid": "kafkagroup"  },  {  "Action": [  "kafka:DescribeCluster",  "kafka:GetBootstrapBrokers",  "kafka:DescribeConfiguration",  "kafka:ListConfigurations",  "kafka:CreateConfiguration"  ],  "Resource": "arn:aws:kafka:us-west-2:593035604125:\*",  "Effect": "Allow",  "Sid": "kafkageneral"  },  {  "Effect": "Allow",  "Action": [  "kafka-cluster:CreateTopic",  "kafka-cluster:WriteData",  "kafka-cluster:ReadData",  "kafka-cluster:DescribeTopic"  ],  "Resource": [  "arn:aws:kafka:us-west-2:593035604125:topic/mskDemo01/\_amazon\_msk\_connect\*"  ]  },  {  "Effect": "Allow",  "Action": [  "kafka-cluster:AlterGroup",  "kafka-cluster:DescribeGroup"  ],  "Resource": [  "arn:aws:kafka:us-west-2:593035604125:group/mskDemo01/b5d3271b-c395-425a-a868-b53dc8b3c8d7-10/\_amazon\_msk\_connect\*",  "arn:aws:kafka:us-west-2:593035604125:group/mskDemo01/b5d3271b-c395-425a-a868-b53dc8b3c8d7-10/connect-\*"  ]  }  ]  } |
| --- |

##### Creating 4th Policy:

Follow the same steps as in [Creating 1st Policy](#_b42dfyti2lgz) but replace the JSON in step5 with the JSON given below and name the policy as GlueAccess.

Note: Make sure you mention your cluster name in the JSON below, we are assuming our schema name as testJsonSchema and Registry name as testRegistry below.

| {  "Version": "2012-10-17",  "Statement": [  {  "Sid": "VisualEditor0",  "Effect": "Allow",  "Action": "glue:\*",  "Resource": [  "arn:aws:glue:us-west-2:593035604125:registry/\*",  "arn:aws:glue:us-west-2:593035604125:schema/testRegistry/testJsonSchema",  "arn:aws:glue:us-west-2:593035604125:schema/testRegistry/\*"  ]  },  {  "Sid": "GetSchemaVersion",  "Effect": "Allow",  "Action": [  "glue:GetSchemaVersion"  ],  "Resource": [  "\*"  ]  }  ]  } |
| --- |

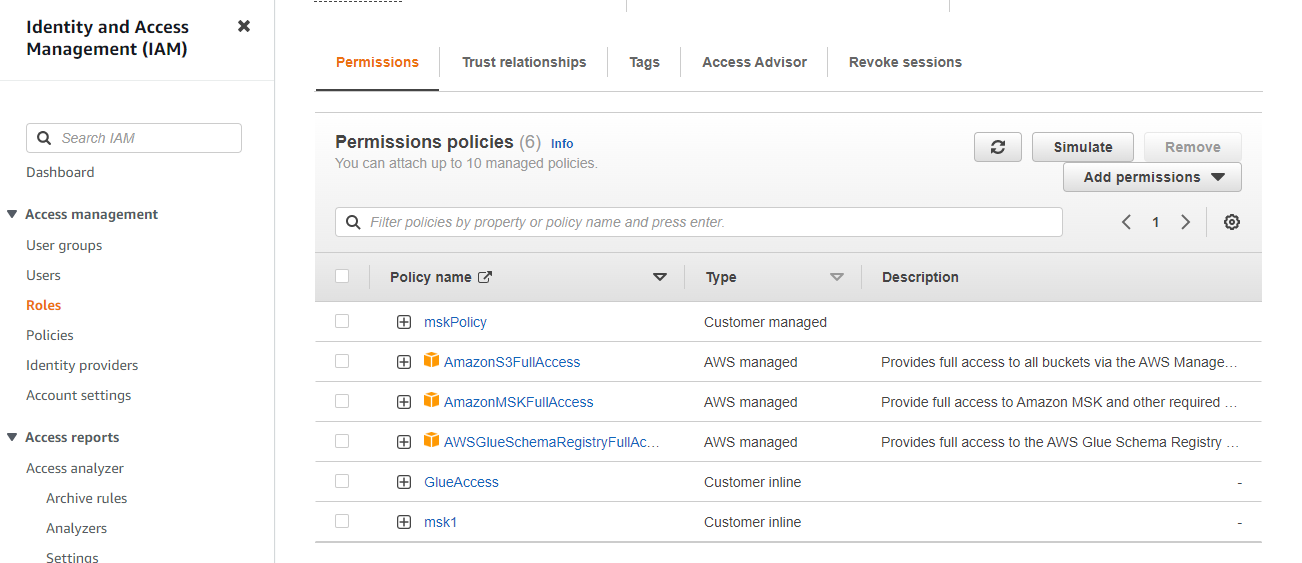
#### 

#### Role creation:

( Two roles required)

##### Creating 1st Role:

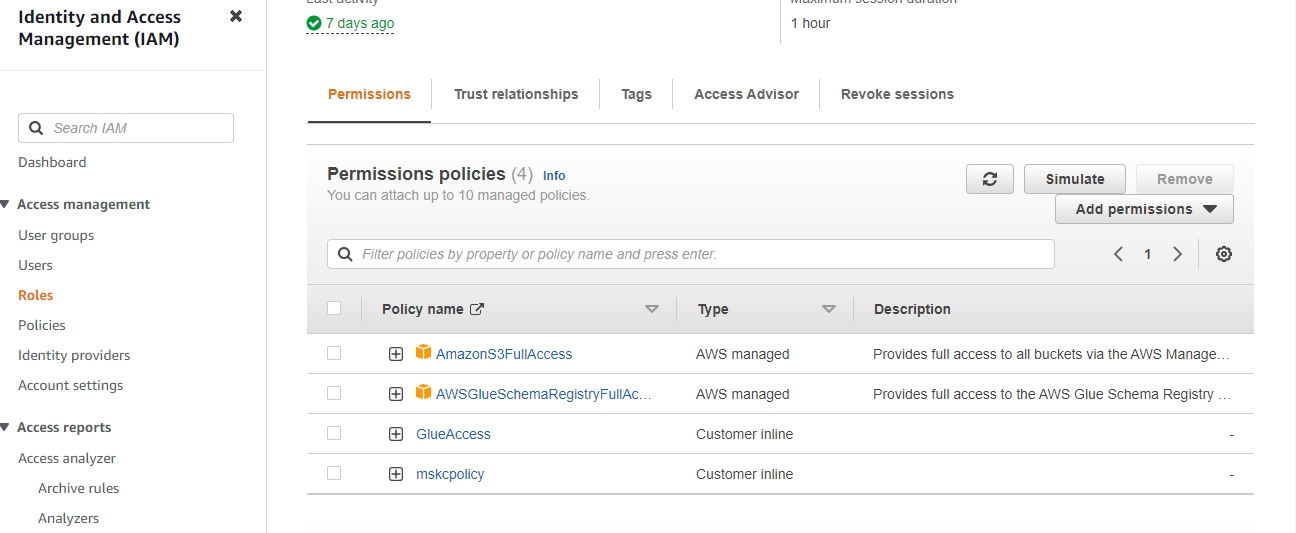
1. From the AWS console, search for IAM, and click on it to open.
2. Now, from the left navigation bar click on Policies .
3. Click on Create role.
4. Select Trusted entity type as AWS service.
5. Select Use case as EC2.
6. Click Next.
7. In Add permissions, search and select the following Policies as shown below.



1. Click Next.
2. Give the Role name as mskRole.
3. Finally, click on Create role.

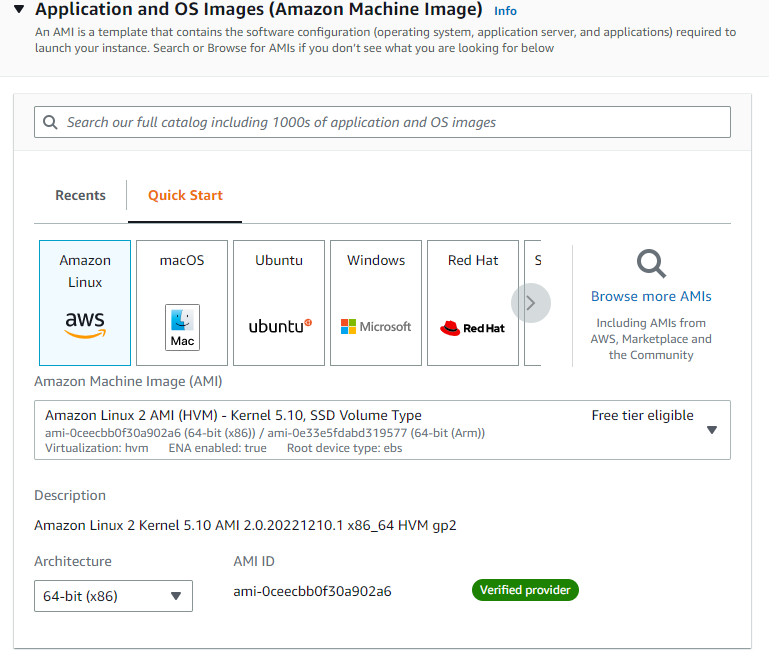
##### Creating 2nd Role:

Follow the same steps as in [Creating 1st Role](#_rp4hqymmxf94) but replace the step7( Add permissions) as shown below and replace the Role name as mskc.

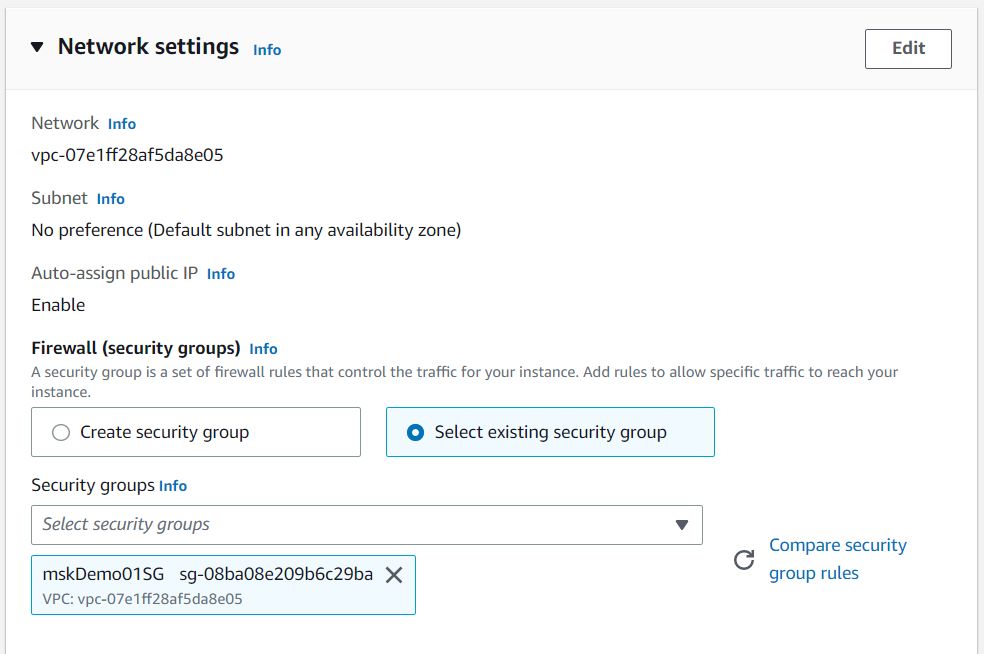


### **Kafka client in EC2:**

1. From the AWS console, click services on top left corner of page and search for EC2 service
2. Open EC2 and click on “Launch instance”.
3. Give a name to the instance (let's say MyInstance).
4. Configure the following Amazon Machine Image.



1. Select Instance type as t2.micro(free tier eligible).
2. Create and select a Key pair, and download it in your system.
3. While configuring, select the same Security Group(say mskDemo 01SG),which we created in the above step while creating a MSK cluster

this can be done by clicking use existing security group while creating EC2 (Refer 11 point in [Start MSK cluster](#_3ws0iw5bh9rw)).

1. Click on “Launch instance”.
2. Click on the instance, which we created and click on Actions on the top left corner.
3. Then select Security and click on Modify IAM role and select an IAM role(mskRole).

Assign below IAM role to the instance(mskRole).

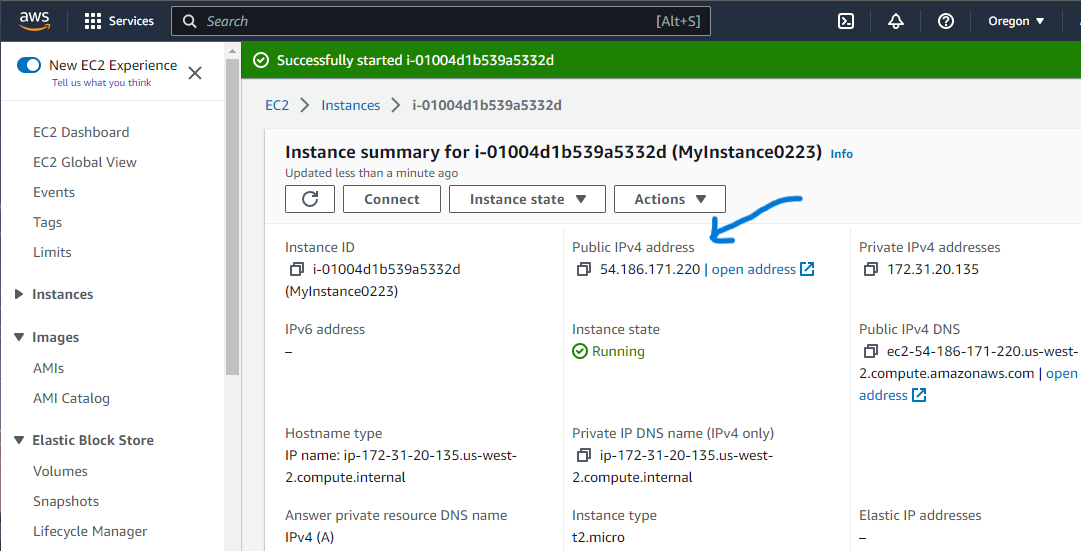
| {  "Version": "2012-10-17",  "Statement": [  {  "Sid": "VisualEditor0",  "Effect": "Allow",  "Action": [  "kafka:CreateConfiguration",  "kafka:UpdateClusterKafkaVersion",  "kafka:UpdateBrokerCount",  "kafka:DeleteConfiguration",  "kafka:UpdateClusterConfiguration",  "kafka:ListClusters",  "kafka:CreateClusterV2",  "kafka:UpdateConnectivity",  "kafka:ListScramSecrets",  "kafka:ListKafkaVersions",  "kafka:GetBootstrapBrokers",  "kafka:UpdateBrokerStorage",  "kafka:RebootBroker",  "kafka:DescribeClusterV2",  "kafka:UpdateMonitoring",  "kafka:ListConfigurationRevisions",  "kafka:UpdateStorage",  "kafka:UpdateSecurity",  "kafka:UpdateBrokerType",  "kafka:CreateCluster",  "kafka:GetCompatibleKafkaVersions",  "kafka:DescribeCluster",  "kafka:UpdateConfiguration",  "kafka:ListConfigurations",  "kafka:BatchDisassociateScramSecret",  "kafka:BatchAssociateScramSecret",  "kafka:ListClustersV2",  "kafka:DescribeClusterOperation",  "kafka:ListNodes",  "kafka:DeleteCluster",  "kafka:ListClusterOperations"  ],  "Resource": "\*"  },  {  "Sid": "VisualEditor1",  "Effect": "Allow",  "Action": [  "kafka:\*",  "iam:PassRole"  ],  "Resource": "\*"  },  {  "Sid": "VisualEditor2",  "Effect": "Allow",  "Action": "kafka-cluster:\*",  "Resource": "\*"  }  ]  } |
| --- |

#### 

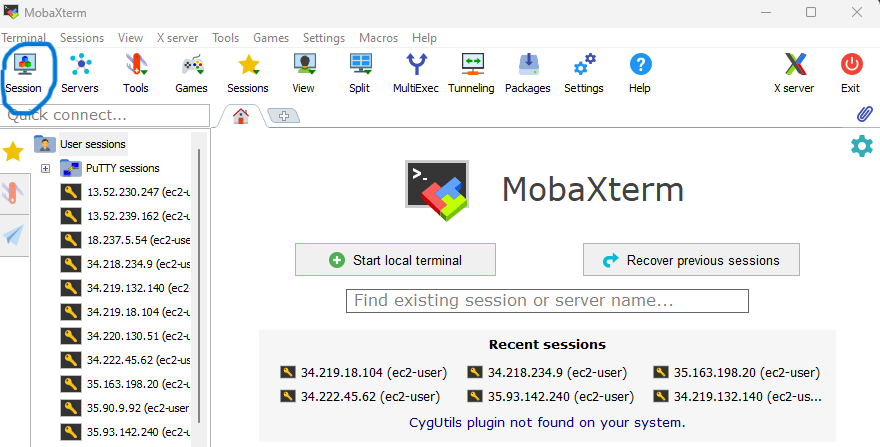
### **How to connect to virtual machine(EC2) via MobaXterm:**

(Note: MobaXterm must be downloaded in your system)

1. Open AWS console.
2. Click on Services on the top left corner.
3. Search for EC2 and open it.
4. Click on the instance(say MyInstance), which you have created in the above step(Refer name of the instance created in [Kafka client in EC2](#_tytlgv7cxrt8)).
5. Copy IPv4 public IP address.



1. Now, open MobaXterm and click on Session on the top left corner.



1. Click on SSH.
2. Give the Remote host as the IPV4 address which you have copied in the 5th step.
3. Specify username as ec2-user.
4. In Advanced SSH settings, select use private key and add the key pair downloaded (refer [Kafka client in EC2](#_tytlgv7cxrt8), step 6).
5. Finally, click on OK.
6. Now, you are connected to the EC2 client.

### **Setting up the requirements in client machine(say MobaXterm):**

1. Download Java in your virtual machine(EC2)

| sudo yum install java-1.8.0-openjdk-devel |
| --- |

1. Next, download kafka using following command

| wget <https://archive.apache.org/dist/kafka/3.0.0/kafka_2.13-3.0.0.tgz> |
| --- |

1. To unzip the above .tgz file, use the following command

| tar -xzf kafka\_2.13-3.0.0.tgz |
| --- |

1. Go into the kafka directory

| cd kafka\_2.13-3.0.0/ |
| --- |

1. Go to Amazon MSK and click on the Cluster created (mskDemo01) .
2. Now, click on view client information in the top right corner and copy the IAM BootstrapServer, and copy it to a variable using the following command.

| export bs=”<IAM BootstrapServer authentication from MSK cluster>” |
| --- |

1. Go to the “bin” folder and create a file “client.properties”.

| cd bin  vi client.properties |
| --- |

1. Add the following authentication code in it,

| security.protocol=SASL\_SSL  sasl.mechanism=AWS\_MSK\_IAM  sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;  sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler |
| --- |

1. Come back to the previous folder and enter the folder “libs”.

| cd .. //comes to previous folder  cd libs //enters into libs folder |
| --- |

1. And execute the following command to add a jar file.

| wget <https://github.com/aws/aws-msk-iam-auth/releases/download/v1.1.5/aws-msk-iam-auth-1.1.5-all.jar> |
| --- |

1. Ensure you come back to the previous folder before proceeding.

| cd .. |
| --- |

1. Create a kafka topic using the following command(Name of the Topic must be same as mentioned in the MSk Connector configuration)

| bin/kafka-topics.sh --create --bootstrap-server $bs --replication-factor 2 --partitions 1 --topic TopicName --config retention.ms=1814400000 --config segment.bytes=134217728 --command-config bin/client.properties |
| --- |

—---------------------------------------------------------------------------------------------------------

#### 

#### 

### **Create Custom plugin:**

##### Creating S3 Custom Plugin: (Note: Use this when creating connector)

* Open Amazon MSK from the console.
* From the left navigation table, click on Custom plugins( refer [Create Custom plugin](#_v54m1ppp75zn)).
* Then, click on create custom plugin.
* In the S3 bucket download the required jar file from below link and upload it to the s3 bucket(say mskconnectbucket1) . <https://www.confluent.io/hub/confluentinc/kafka-connect-s3> (This jar file is for Json)
* Browse the S3 bucket and select the jar file from it.

##### Creating Glue Custom Plugin:

### 

### 

### **\*Setup\* (For Json Schema):**

1. Come to home directory.

| cd ~ |
| --- |

1. Create a folder.

| mkdir myPrograms |
| --- |

1. Copy all the jar files present in kafka/libs into the present directory.

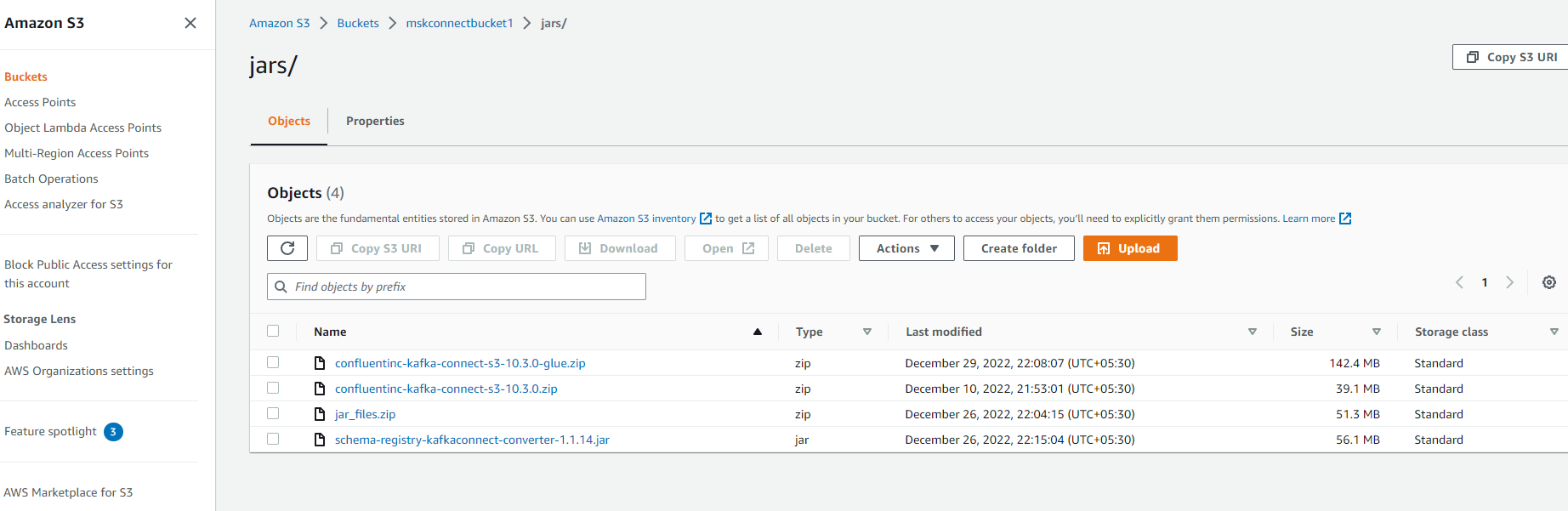
| cp ../kafka\_2.13-3.0.0/libs/\*.jar . |
| --- |

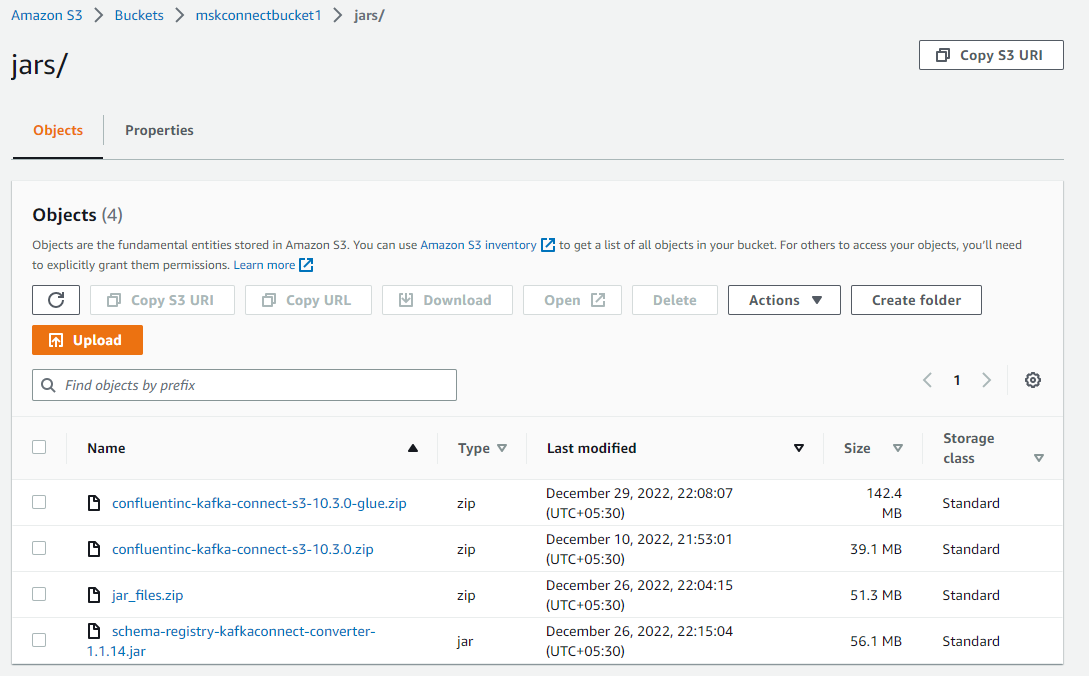
1. In the current folder execute the following commands which helps to run a Java code(producer).

| wget <https://repo1.maven.org/maven2/software/amazon/glue/jsonschema-kafkaconnect-converter/1.1.14/jsonschema-kafkaconnect-converter-1.1.14.jar>  wget <https://repo1.maven.org/maven2/software/amazon/glue/schema-registry-common/1.1.14/schema-registry-common-1.1.14.jar>  wget <https://repo1.maven.org/maven2/software/amazon/glue/schema-registry-serde/1.1.14/schema-registry-serde-1.1.14.jar>  wget <https://repo1.maven.org/maven2/software/amazon/glue/schema-registry-kafkaconnect-converter/1.1.14/schema-registry-kafkaconnect-converter-1.1.14.jar> |
| --- |

1. Create a folder in myPrograms and add the jar files present in s3 into the latest directory.

| mkdir mskc\_setup |
| --- |





1. Copy the zip files from s3 bucket to the current file.

| cd mskc\_setup |
| --- |

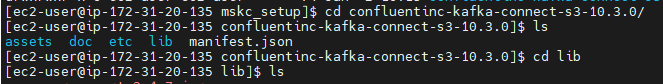
| aws s3 cp s3://mskconnectbucket1/jars/confluentinc-kafka-connect-s3-10.3.0-glue.zip . |
| --- |

1. Unzip the above file.

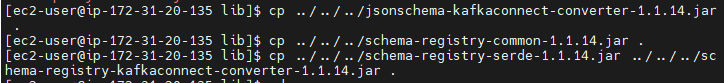
| unzip confluentinc-kafka-connect-s3-10.3.0-glue.zip |
| --- |

1. Enter the file.

| cd confluentinc-kafka-connect-s3-10.3.0-glue.zip |
| --- |



1. Copy the Four jar files which we downloaded in myPrograms folder into the current folder.



| cp ../../../jsonschema-kafkaconnect-converter-1.1.14.jar .  cp ../../../schema-registry-common-1.1.14.jar .  cp ../../../schema-registry-serde-1.1.14.jar .    cp ../../../schema-registry-kafkaconnect-converter-1.1.14.jar . |
| --- |

(or) All the files can be copied at once in the current directly using single command;

| cp ../../../jsonschema-kafkaconnect-converter-1.1.14.jar ../../../schema-registry-common-1.1.14.jar ../../../schema-registry-serde-1.1.14.jar ../../../schema-registry-kafkaconnect-converter-1.1.14.jar . |
| --- |

1. Move back one directory (come back to mskc\_setup)

| cd .. |
| --- |

1. Zip the previous file.

| zip -r confluentinc-kafka-connect-s3-10.3.0\_glue.zip confluentinc-kafka-connect-s3-10.3.0/ |
| --- |

1. Come back to myPrograms folder.

| cd .. |
| --- |

1. Now, Finally run the Java program for JsonSchema.(refer End-to-End: JsonSchema to parquet )

### **Create S3 Bucket:**

1. Open the AWS console, and click Services on the top left corner.
2. Now from the left navigation panel, select buckets, and click on Create bucket.
3. Give Bucket name(let's say mskconnectbucket1).
4. Leave other configurations as default and finally click on Create bucket.

### **Adding required Jars in S3 bucket:**

1. Google "confluent s3 sink connector zip".
2. Download zip from the website -<https://www.confluent.io/hub/confluentinc/kafka-connect-s3>
3. Now, create a folder called jars in our S3 bucket “MskConnect01”.
4. Open the bucket in AWS S3(which we created in the [Create S3 Bucket](#_csyku1vxx0nr) step) and upload the zip file into the "jars" folder.
5. Now, in MSK create a plugin pointing to the S3 zip file created in step3.
6. For Glue?

### **Create MSK Connect connector:**

1. Click on Connectors from the left panel in MSK.
2. Select Use Existing custom plugin (created in the step [Creating Custom Plugin](#_2jv8jtx8n9lw)).
3. Select the plugin in Custom plugins(which we created in S3).
4. Enter the Connector name (let's say MskConnect01).
5. Select the Cluster which we created in the step [Start MSK cluster](#_3ws0iw5bh9rw) and in the Authentication dropdown, select IAM (say mskc) (refer [Creating MSK role for Msk connector](#_f46tdpwd8s55)).

| Note: Connector configuration can be set, depending upon the type of message.  (Refer; End-to-End Run: Using kafka connect\*\*\*). |
| --- |

1. Select Connector capacity as Autoscaled.
2. In Access permissions, Select the IAM service role mskc.

Note: If not created IAM earlier, Click on Create (refer [Creating IAM creation](#_q0iwc46gg6q2) ).

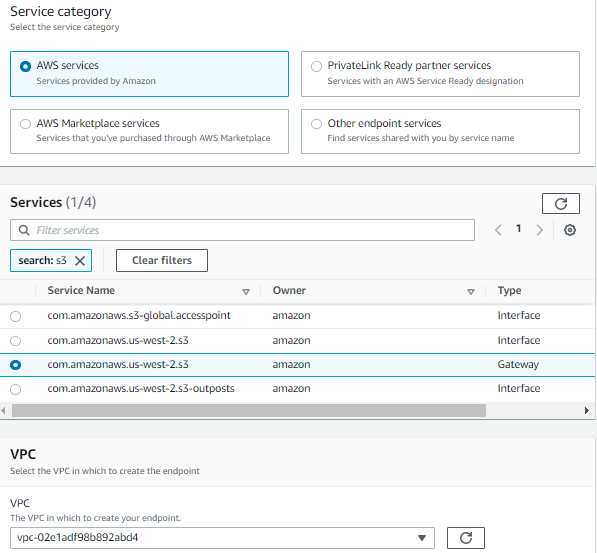
1. Select AWS service, and Use case as EC2 and MSK connect (search and select).
2. Give the following permissions to the role.
3. Click Next.
4. In Logs, select Deliver to Amazon s3 and select the Bucket accordingly.
5. Click Next and Create connector.

#### 

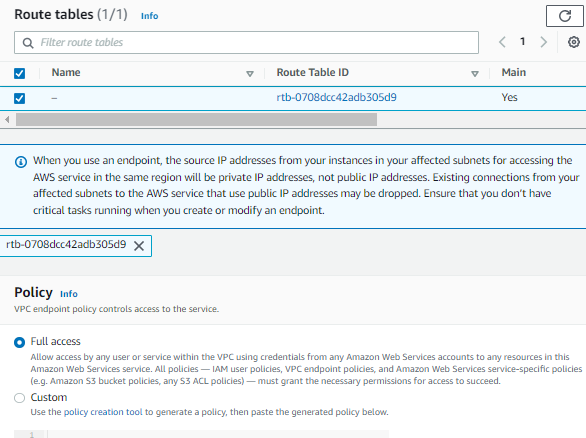
### **Creating VPC Endpoints:**

#### Creating S3 endpoint:

1. From the AWS console, search VPC and click on it.
2. Now,click on create endpoint.
3. Give a name to the endpoint(let's say s3endpoint).
4. In the Service category select AWS services.
5. Search s3 in Services and select the given service with type Gateway.



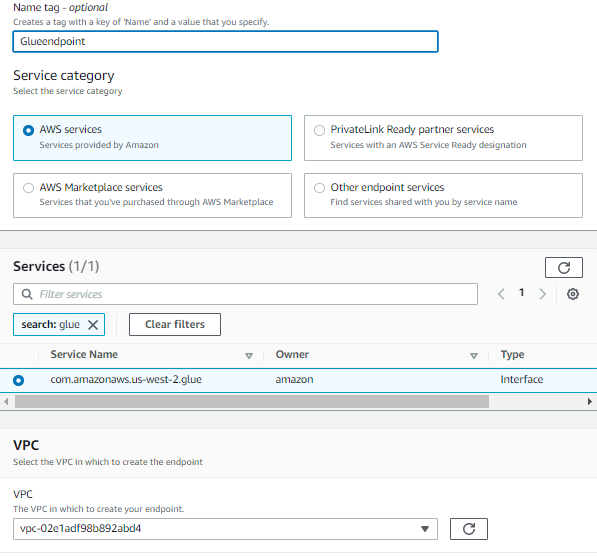
1. Select the Route table (default).
2. In Policy, select Full access.



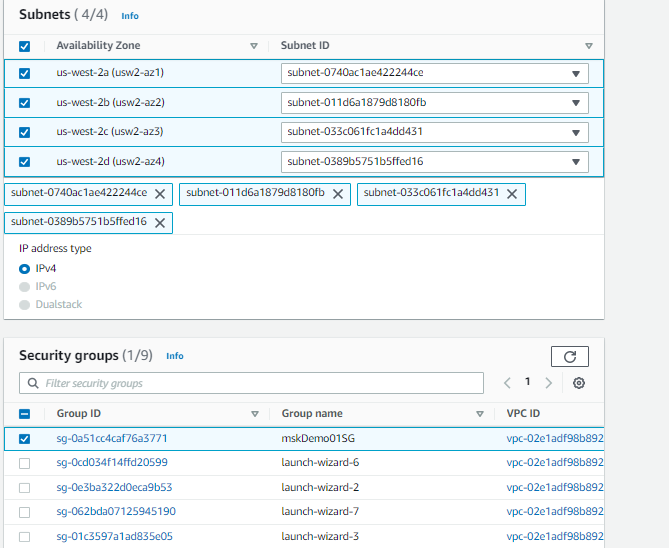
1. Finally, click on create endpoint.

#### Creating Glue endpoint:

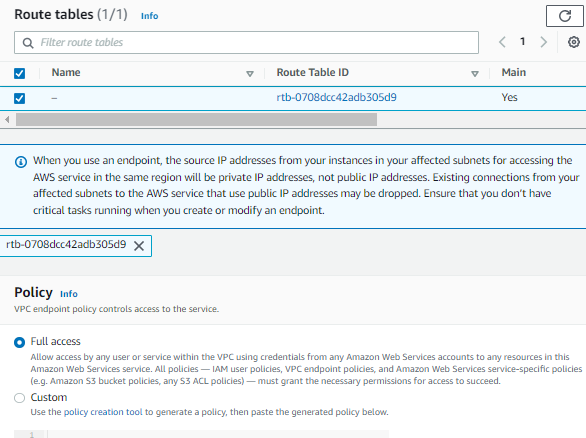
1. From the AWS console, search VPC and click on it to open.
2. Now,click on create endpoint.
3. Give a name to the endpoint(let's say Glueendpoint).
4. In the Service category select AWS services.
5. Search glue in Services and select the given service with type Gateway.



1. Select the Subnets and select IP address type as IPv4.



1. In Policy, select Full access.



1. Finally, click on create endpoint.

### **Creating AWS Schema and Registry( For Avro ):**

1. From the AWS console, search Glue and click on it to open.
2. Now, from the left bar, click on Schemas.
3. Click on Add schema.
4. Give Schema name(let's say Test\_schema).
5. Now, click on Add new registry , and give Registry name(let's say Test\_registry).
6. Click on Add registry at the bottom of the page and come back to the Schemas page.
7. Select the Registry created in the previous step ,and Data format as Apache Avro.
8. Add Payment.avsc in First schema version.( refer [Payment.avsc for Avro](#_pcrs0jof2keh))
9. Finally, click Create schema and version.

Note: The Registry name and Schema name must be mentioned in the kafka connect configuration for JsonSchema to parquet and Avro message formats.

## End-to-End Run(15 mins):

### **Json**

#### Json Producer:

##### Using CLI:

| bin/kafka-console-producer.sh --bootstrap-server $bs --producer.config bin/client.properties --topic TestTopic2 |
| --- |

##### Using Java Program:

Use the following Java Program to create a producer;

| import java.io.IOException;  import java.util.\*;  import org.apache.kafka.clients.producer.\*;  import java.io.File;  import org.apache.kafka.clients.producer.ProducerConfig;  public class Producer\_IAM {  public static void main(String[] args) {  String topicName = "TestTopic2";  String key = "test1";  String value = "{\"firstName\":\"Aditya\",\"lastName\":\"Dhanraj\",\"age\":\"20\",\"city\":\"hyderabad\",\"state\":\"telangana\",\"country\":\"India\"}";  Properties properties = new Properties();  properties.put("bootstrap.servers","b-1.mskdemo01.ort0t6.c13.kafka.us-west-2.amazonaws.com:9098,b-2.mskdemo01.ort0t6.c13.kafka.us-west-2.amazonaws.com:9098,b-3.mskdemo01.ort0t6.c13.kafka.us-west-2.amazonaws.com:9098");  properties.put("security.protocol","SASL\_SSL");  properties.put("sasl.mechanism","AWS\_MSK\_IAM");  properties.put("sasl.jaas.config","software.amazon.msk.auth.iam.IAMLoginModule required;");  properties.put("sasl.client.callback.handler.class","software.amazon.msk.auth.iam.IAMClientCallbackHandler");  properties.put("key.serializer","org.apache.kafka.common.serialization.StringSerializer");  properties.put("value.serializer","org.apache.kafka.common.serialization.StringSerializer");  properties.put("group.id", "cdi2");    Producer<String, String> producer = new KafkaProducer <>(properties);  ProducerRecord<String, String> record = new ProducerRecord<>(topicName,key,value);  producer.send(record);  producer.close();  System.out.println("SimpleProducer Completed.");  }  } |
| --- |

#### Json Consumer:

##### Using CLI:

| bin/kafka-console-consumer.sh --bootstrap-server $bs --consumer.config bin/client.properties --topic TestTopic2 |
| --- |

##### Using Java Program:

#### Using Kafka connect for Json:

| connector.class=io.confluent.connect.s3.S3SinkConnector  s3.region=us-west-2  format.class=io.confluent.connect.s3.format.json.JsonFormat  flush.size=1  schema.compatibility=NONE  topics=TestTopic2  tasks.max=1  partitioner.class=io.confluent.connect.storage.partitioner.DefaultPartitioner  storage.class=io.confluent.connect.s3.storage.S3Storage  s3.bucket.name=mskconnectbucket1  schema.compatibility=NONE  key.converter.schemas.enable=false  key.converter=org.apache.kafka.connect.storage.StringConverter  value.converter.schemas.enable=false  value.converter=org.apache.kafka.connect.json.JsonConverter |
| --- |

### **JsonShema to Parquet:**

#### JSON Schema Producer:

##### Using CLI:

This is not available using open kafka. This is available only in Confluent Kafka.

Confluent reference link for JSON Schema: <https://docs.confluent.io/platform/current/schema-registry/serdes-develop/serdes-json.html#schema-references-in-json-schemas>

##### Using Java Program:

Note: We use Glue for Schema registration (Refer End-to-End setup: Creating AWS Schema and Registry.

| import com.amazonaws.services.schemaregistry.serializers.GlueSchemaRegistryKafkaSerializer;  import com.amazonaws.services.schemaregistry.serializers.json.JsonDataWithSchema;  import com.amazonaws.services.schemaregistry.utils.AWSSchemaRegistryConstants;  import org.apache.kafka.clients.producer.KafkaProducer;  import org.apache.kafka.clients.producer.ProducerConfig;  import org.apache.kafka.clients.producer.ProducerRecord;  import org.apache.kafka.common.errors.SerializationException;  import org.apache.kafka.common.serialization.StringSerializer;  import software.amazon.awssdk.services.glue.model.DataFormat;  import java.util.ArrayList;  import java.util.List;  import java.util.Properties;  public class JsonSchemaProducer {  public static void main(String[] args) {  String topic = "TestJsonSchema";  Properties properties = new Properties();  //producer object uses this list to connect to kafka cluster  String bs="b-3.mskcluster2023.dusup6.c8.kafka.us-west-2.amazonaws.com:9098,b-2.mskcluster2023.dusup6.c8.kafka.us-west-2.amazonaws.com:9098,b-1.mskcluster2023.dusup6.c8.kafka.us-west-2.amazonaws.com:9098";  properties.put("bootstrap.servers", bs);  properties.put("security.protocol", "SASL\_SSL");  properties.put("sasl.mechanism", "AWS\_MSK\_IAM");  properties.put("sasl.jaas.config", "software.amazon.msk.auth.iam.IAMLoginModule required;");  properties.put("sasl.client.callback.handler.class", "software.amazon.msk.auth.iam.IAMClientCallbackHandler");  //props.put("key.serializer","org.apache.kafka.common.serialization.StringSerializer");  //props.put("value.serializer","org.apache.kafka.common.serialization.StringSerializer");  properties.put("group.id", "cdi2");  properties.put(ProducerConfig.KEY\_SERIALIZER\_CLASS\_CONFIG, StringSerializer.class.getName());  properties.put(ProducerConfig.VALUE\_SERIALIZER\_CLASS\_CONFIG, GlueSchemaRegistryKafkaSerializer.class.getName());  properties.put(AWSSchemaRegistryConstants.DATA\_FORMAT, DataFormat.JSON.name());  properties.put(AWSSchemaRegistryConstants.AWS\_REGION, "us-west-2");  properties.put(AWSSchemaRegistryConstants.REGISTRY\_NAME, "cdi2\_test");  properties.put(AWSSchemaRegistryConstants.SCHEMA\_NAME, "cdi2\_json\_schema1");  properties.put(AWSSchemaRegistryConstants.SCHEMA\_AUTO\_REGISTRATION\_SETTING, true);  String jsonSchema = "{\n" +  "\"$id\": \"https://example.com/segment.schema.json\",\n" +  "\"$schema\": \"http://json-schema.org/draft-07/schema#\",\n" +  "\"title\": \"Segment\",\n" +  "\"type\": \"object\",\n" +  "\"properties\": {\n" +  "\"Origin\": {\n" +  "\"description\": \"The App Origin\",\n" +  "\"type\": \"string\"\n" +  "},\n" +  "\"Channel\": {\n" +  "\"description\": \"The App Channel\",\n" +  "\"type\": \"string\"\n" +  "},\n" +  "\"Platform\": {\n" +  "\"description\": \"The Platform of BU\",\n" +  "\"type\": \"string\"\n" +  "},\n" +  "\"SentAt\": {\n" +  "\"description\": \"message SentAt\",\n" +  "\"type\": \"string\"\n" +  "},\n" +  "\"EventID\": {\n" +  "\"description\": \"An EventId\",\n" +  "\"type\": \"string\"\n" +  "},\n" +  "\"AnonymousId\": {\n" +  "\"description\": \"An AnonymousId\",\n" +  "\"type\": \"string\"\n" +  "},\n" +  "\"IPAddress\": {\n" +  "\"description\": \"IP Address\",\n" +  "\"type\": \"string\"\n" +  "},\n" +  "\"City\": {\n" +  "\"description\": \"City\",\n" +  "\"type\": \"string\"\n" +  "},\n" +  "\"Topic\": {\n" +  "\"description\": \"Topic Name\",\n" +  "\"type\": \"string\"\n" +  "},\n" +  "\"EventType\": {\n" +  "\"description\": \"Type of Event\",\n" +  "\"type\": \"string\"\n" +  "},\n" +  "\"Data\": {\n" +  "\"description\": \"Actual Data\",\n" +  "\"type\": \"string\"\n" +  "}\n" +  "}\n" +  "}";  String jsonPayload="{\"Origin\":\"cdi2\",\"Channel\":\"\",\"Platform\":\"platform\",\"SentAt\":\"2022-12-15T23:18:28.678506601Z\",\"EventID\":\"294ebad0-08e0-465e-8edc-28599312af1a\",\"AnonymousId\":\"c43e21ab-3189-48eb-b34c-a42387ecbbd5\",\"IPAddress\":\"172.17.0.1:60182\",\"City\":\"NA\",\"Topic\":\"CDI2\_TOPIC\",\"EventType\":\"default\",\"Data\":\"{\\\"firstName\\\":\\\"pavan\\\",\\\"LastName\\\":\\\"Sabinikari\\\"}\"}";  JsonDataWithSchema jsonSchemaWithData = JsonDataWithSchema.builder(jsonSchema, jsonPayload).build();  List<JsonDataWithSchema> genericJsonRecords = new ArrayList<>();  genericJsonRecords.add(jsonSchemaWithData);  genericJsonRecords.add(jsonSchemaWithData);  genericJsonRecords.add(jsonSchemaWithData);  genericJsonRecords.add(jsonSchemaWithData);  genericJsonRecords.add(jsonSchemaWithData);  genericJsonRecords.add(jsonSchemaWithData);  genericJsonRecords.add(jsonSchemaWithData);  genericJsonRecords.add(jsonSchemaWithData);  genericJsonRecords.add(jsonSchemaWithData);  genericJsonRecords.add(jsonSchemaWithData);  try (KafkaProducer<String, JsonDataWithSchema> producer = new KafkaProducer<String, JsonDataWithSchema>(properties)) {  for (int i = 0; i < genericJsonRecords.size(); i++) {  JsonDataWithSchema r = genericJsonRecords.get(i);  System.out.println("recored is "+r.toString());  final ProducerRecord<String, JsonDataWithSchema> record;  record = new ProducerRecord<String, JsonDataWithSchema>(topic, "message-" + i, r);  //record.headers().add("Origin", "cdi2".getBytes());  //record.headers().add("Platform", "fox\_ios".getBytes());  producer.send(record);  System.out.println("Sent message " + i);  //Thread.sleep(1000L);  }  producer.flush();  System.out.println("Successfully produced 10 messages to a topic called " + topic);  } catch ( Exception e) {  e.printStackTrace();  }  }  } |
| --- |

1. Now, start the Producer using the following command and give the Topic name(let's say TestJsonSchema).
2. Go to S3 and check if the topic is created.

### 

#### Json Schema Consumer:

##### Using CLI:

Note: we are using open source kafka, which does not have capability for Json Schema CLI producer. If you install Confluent Kafka you will get CLI JSON schema CLI producer.

##### Using Java Program:

| import com.amazonaws.services.schemaregistry.deserializers.GlueSchemaRegistryKafkaDeserializer;  import com.amazonaws.services.schemaregistry.serializers.json.JsonDataWithSchema;  import com.amazonaws.services.schemaregistry.utils.AWSSchemaRegistryConstants;  import org.apache.kafka.clients.consumer.Consumer;  import org.apache.kafka.clients.consumer.ConsumerRecord;  import org.apache.kafka.clients.consumer.ConsumerRecords;  import org.apache.kafka.clients.consumer.KafkaConsumer;  import org.apache.kafka.clients.producer.ProducerConfig;  import org.apache.kafka.clients.consumer.ConsumerConfig;  import org.apache.kafka.common.serialization.StringDeserializer;  import org.apache.kafka.common.serialization.StringSerializer;  import software.amazon.awssdk.services.glue.model.DataFormat;  import java.util.Arrays;  import java.util.Properties;  public class JsonSchemaConsumer {  public static void main(String[] args) {  String topic = "TestJsonSchema";  Properties properties = new Properties();  //producer object uses this list to connect to kafka cluster  String bs = "b-1.testcluster001.fdi872.c8.kafka.us-west-2.amazonaws.com:9098,b-3.testcluster001.fdi872.c8.kafka.us-west-2.amazonaws.com:9098,b-2.testcluster001.fdi872.c8.kafka.us-west-2.amazonaws.com:9098";  properties.put("bootstrap.servers", bs);  properties.put("security.protocol", "SASL\_SSL");  properties.put("sasl.mechanism", "AWS\_MSK\_IAM");  properties.put("sasl.jaas.config", "software.amazon.msk.auth.iam.IAMLoginModule required;");  properties.put("sasl.client.callback.handler.class", "software.amazon.msk.auth.iam.IAMClientCallbackHandler");  properties.put("group.id", "cdi2");  properties.put(ConsumerConfig.KEY\_DESERIALIZER\_CLASS\_CONFIG, StringDeserializer.class.getName());  properties.put(ConsumerConfig.VALUE\_DESERIALIZER\_CLASS\_CONFIG, GlueSchemaRegistryKafkaDeserializer.class.getName());  properties.put(AWSSchemaRegistryConstants.DATA\_FORMAT, DataFormat.JSON.name());  properties.put(AWSSchemaRegistryConstants.AWS\_REGION, "us-west-2");  properties.put(AWSSchemaRegistryConstants.REGISTRY\_NAME, "cdi2\_test");  properties.put(AWSSchemaRegistryConstants.SCHEMA\_NAME, "cdi2\_json\_schema1");  //properties.put(AWSSchemaRegistryConstants.SCHEMA\_AUTO\_REGISTRATION\_SETTING, true);  Consumer<String, JsonDataWithSchema> consumer = new KafkaConsumer<String, JsonDataWithSchema>(properties);  consumer.subscribe(Arrays.asList(topic));  try {  while (true) {  @SuppressWarnings("deprecation")  ConsumerRecords<String, JsonDataWithSchema> records = consumer.poll(100);  for (ConsumerRecord<String, JsonDataWithSchema> record : records) {  String key = record.key();  JsonDataWithSchema value = record.value();  System.out.println("Received message: key = " + key + ", value = " + value);  }  }  } finally {  consumer.close();  }  }  } |
| --- |

#### 

#### Using kafka connect S3 sink for Json Schema:

| connector.class=io.confluent.connect.s3.S3SinkConnector  value.converter.schemaAutoRegistrationEnabled=true  s3.region=us-west-2  key.converter.endpoint=https://glue.us-west-2.amazonaws.com  flush.size=1  tasks.max=1  key.converter.region=us-west-2  value.converter.endpoint=https://glue.us-west-2.amazonaws.com  key.converter.schemaName=Test\_schema  internal.key.converter.schemas.enable=false  value.converter.avroRecordType=GENERIC\_RECORD  format.class=io.confluent.connect.s3.format.parquet.ParquetFormat  value.converter.schemaName=Test\_schema  key.converter.dataFormat=JSON  key.converter.avroRecordType=GENERIC\_RECORD  value.converter=com.amazonaws.services.schemaregistry.kafkaconnect.jsonschema.JsonSchemaConverter  s3.bucket.name=mskconnectbucket1  key.converter=org.apache.kafka.connect.storage.StringConverter  schema.compatibility=FULL  topics=TestJsonSchema  value.converter.dataFormat=JSON  value.converter.registry.name=Test\_registry  key.converter.registry.name=Test\_registry  value.converter.region=us-west-2  internal.key.converter=org.apache.kafka.connect.storage.StringConverter  key.converter.schemas.enable=false  key.converter.schemaAutoRegistrationEnabled=true  partitioner.class=io.confluent.connect.storage.partitioner.DefaultPartitioner  internal.value.converter.schemas.enable=false  internal.value.converter=org.apache.kafka.connect.json.JsonConverter  value.converter.schemas.enable=true  storage.class=io.confluent.connect.s3.storage.S3Storage |
| --- |

### **Avro**

To know more about Avro refer this link

<https://avro.apache.org/docs/1.11.1/>

#### Avro Producer

##### Using CLI:

(No CLI)

<https://docs.confluent.io/platform/current/schema-registry/serdes-develop/serdes-avro.html#avro-schema-serializer-and-deserializer>

##### Using Java Program

| import java.io.IOException;  import java.util.\*;  import org.apache.avro.Schema;  import org.apache.avro.generic.GenericData;  import org.apache.avro.generic.GenericRecord;  import org.apache.kafka.clients.producer.\*;  import com.amazonaws.services.schemaregistry.utils.AWSSchemaRegistryConstants;  import com.amazonaws.services.schemaregistry.serializers.GlueSchemaRegistryKafkaSerializer;  import java.io.File;  import org.apache.kafka.clients.producer.ProducerConfig;  import org.apache.kafka.common.errors.SerializationException;  import org.apache.kafka.common.serialization.StringSerializer;  import software.amazon.awssdk.services.glue.model.DataFormat;  import org.apache.avro.Schema.Parser;  public class Producer\_Glue {  public static void main(String[] args) {  String topic = "TestAvroTopic";  Properties properties = new Properties();  //producer object uses this list to connect to kafka cluster  properties.put("bootstrap.servers", "b-3.mskdemo01.bk3adt.c8.kafka.us-west-2.amazonaws.com:9098,b-1.mskdemo01.bk3adt.c8.kafka.us-west-2.amazonaws.com:9098,b-2.mskdemo01.bk3adt.c8.kafka.us-west-2.amazonaws.com:9098");  properties.put("security.protocol","SASL\_SSL");  properties.put("sasl.mechanism","AWS\_MSK\_IAM");  properties.put("sasl.jaas.config","software.amazon.msk.auth.iam.IAMLoginModule required;");  properties.put("sasl.client.callback.handler.class","software.amazon.msk.auth.iam.IAMClientCallbackHandler");  properties.put("group.id", "cdi2");  properties.put(ProducerConfig.KEY\_SERIALIZER\_CLASS\_CONFIG, StringSerializer.class.getName());  properties.put(ProducerConfig.VALUE\_SERIALIZER\_CLASS\_CONFIG, GlueSchemaRegistryKafkaSerializer.class.getName());  properties.put(AWSSchemaRegistryConstants.DATA\_FORMAT, DataFormat.AVRO.name());  properties.put(AWSSchemaRegistryConstants.AWS\_REGION, "us-west-2");  properties.put(AWSSchemaRegistryConstants.REGISTRY\_NAME, "cdi2\_test");  properties.put(AWSSchemaRegistryConstants.SCHEMA\_NAME, "Test\_schema");  Schema schema\_payment = null;  try {  schema\_payment = new Parser().parse(new File("/home/ec2-user/myPrograms  /Payment.avsc"));  } catch (IOException e) {  e.printStackTrace();  }  GenericRecord musical = new GenericData.Record(schema\_payment);  musical.put("id", "entertainment\_2");  musical.put("amount", 105.0);    GenericRecord mus = new GenericData.Record(schema\_payment);  mus.put("id", "entertainment\_2");  mus.put("amount", 120.0);  GenericRecord musical1 = new GenericData.Record(schema\_payment);  musical1.put("id", "entertainment\_1");  musical1.put("amount", 115.0);  GenericRecord mus1 = new GenericData.Record(schema\_payment);  mus1.put("id", "entertainment\_1");  mus1.put("amount", 125.0);    List<GenericRecord> misc = new ArrayList<>();  misc.add(musical);  misc.add(musical1);  misc.add(mus);  misc.add(mus1);  try (KafkaProducer<String, GenericRecord> producer = new KafkaProducer<String, GenericRecord>(properties)) {  for (int i = 0; i < 4; i++) {  GenericRecord r = misc.get(i);  final ProducerRecord<String, GenericRecord> record;  record = new ProducerRecord<String, GenericRecord>(topic, r.get("id").toString(), r);  producer.send(record);  System.out.println("Sent message " + i);  Thread.sleep(1000L);  }  producer.flush();  System.out.println("Successfully produced 4 messages to a topic called " + topic);  } catch (final InterruptedException | SerializationException e) {  e.printStackTrace();  }  }  } |
| --- |

##### Payment.avsc for Avro

| {  "namespace": "com.cdi2.services.schemaregistry.integrationtests",  "type": "record",  "name": "Payment",  "fields": [  {"name": "id", "type": "string"},  {"name": "amount", "type": "double"}  ]  } |
| --- |

#### Avro Consumer

##### Using CLI:

No CLI

##### Using Java Program

| import java.util.\*;  import com.amazonaws.services.schemaregistry.deserializers.GlueSchemaRegistryKafkaDeserializer;  import com.amazonaws.services.schemaregistry.utils.AvroRecordType;  import org.apache.avro.generic.GenericRecord;  import org.apache.kafka.clients.consumer.ConsumerConfig;  import org.apache.kafka.clients.consumer.ConsumerRecord;  import org.apache.kafka.clients.consumer.ConsumerRecords;  import org.apache.kafka.clients.consumer.KafkaConsumer;  import com.amazonaws.services.schemaregistry.utils.AWSSchemaRegistryConstants;  import org.apache.kafka.common.serialization.StringDeserializer;  public class Consumer\_Glue {  public static void main(String[] args) {  String topic = "TestAvroTopic";  Properties properties = new Properties();  properties.put("bootstrap.servers", "b-3.mskdemo01.bk3adt.c8.kafka.us-west-2.amazonaws.com:9098,b-2.mskdemo01.bk3adt.c8.kafka.us-west-2.amazonaws.com:9098,b-1.mskdemo01.bk3adt.c8.kafka.us-west-2.amazonaws.com:9098");  properties.put("security.protocol","SASL\_SSL");  properties.put("sasl.mechanism","AWS\_MSK\_IAM");  properties.put("sasl.jaas.config","software.amazon.msk.auth.iam.IAMLoginModule required;");  properties.put("sasl.client.callback.handler.class","software.amazon.msk.auth.iam.IAMClientCallbackHandler");  properties.put("group.id", "cdi2");  properties.put(ConsumerConfig.KEY\_DESERIALIZER\_CLASS\_CONFIG, StringDeserializer.class.getName());  properties.put(ConsumerConfig.AUTO\_OFFSET\_RESET\_CONFIG, "earliest");  properties.put(ConsumerConfig.VALUE\_DESERIALIZER\_CLASS\_CONFIG, GlueSchemaRegistryKafkaDeserializer.class.getName());  properties.put(AWSSchemaRegistryConstants.AWS\_REGION, "us-west-2");  properties.put(AWSSchemaRegistryConstants.AVRO\_RECORD\_TYPE, AvroRecordType.GENERIC\_RECORD.getName());  try (final KafkaConsumer<String, GenericRecord> consumer = new KafkaConsumer<String, GenericRecord>(properties)) {  consumer.subscribe(Collections.singletonList(topic));  while (true) {  final ConsumerRecords<String, GenericRecord> records = consumer.poll(100);  for (final ConsumerRecord<String, GenericRecord> record : records) {  final String key = record.key();  final GenericRecord value = record.value();  System.out.println("Received message: key = " + key + ", value = " + value);  }  }  }  }  } |
| --- |

##### Using kafka connect S3 sink for Avro:

| connector.class=io.confluent.connect.s3.S3SinkConnector  value.converter.schemaAutoRegistrationEnabled=true  s3.region=us-west-2  key.converter.endpoint=https://glue.us-west-2.amazonaws.com  flush.size=1  tasks.max=2  key.converter.region=us-west-2  value.converter.endpoint=https://glue.us-west-2.amazonaws.com  key.converter.schemaName=Test\_schema  internal.key.converter.schemas.enable=false  value.converter.avroRecordType=GENERIC\_RECORD  format.class=io.confluent.connect.s3.format.avro.AvroFormat  value.converter.schemaName=Test\_schema  key.converter.avroRecordType=GENERIC\_RECORD  value.converter=com.amazonaws.services.schemaregistry.kafkaconnect.AWSKafkaAvroConverter  s3.bucket.name=mskconnectbucket1  key.converter=org.apache.kafka.connect.storage.StringConverter  schema.compatibility=FULL  topics=TestAvroTopic  value.converter.registry.name=Test\_registry  key.converter.registry.name=Test\_registry  value.converter.region=us-west-2  internal.key.converter=com.amazonaws.services.schemaregistry.kafkaconnect.AWSKafkaAvroConverter  key.converter.schemas.enable=false  key.converter.schemaAutoRegistrationEnabled=true  partitioner.class=io.confluent.connect.storage.partitioner.DefaultPartitioner  internal.value.converter.schemas.enable=false  value.converter.schemas.enable=true  storage.class=io.confluent.connect.s3.storage.S3Storage |
| --- |

### **ProtoBuf**

Step1: Create schema as below.

Reference: https://developers.google.com/protocol-buffers/docs/javatutorial

| syntax = "proto3";  import "google/protobuf/timestamp.proto";  option java\_multiple\_files = true;  option java\_outer\_classname = "CollectorProtos";  option go\_package = "./";  message CollectorEvent{  optional string Origin = 1;  optional string Channel = 2;  optional string Platform = 3;  optional google.protobuf.Timestamp SentAt = 4;  optional string EventID = 5;  optional string AnonymousId = 6;  optional string IPAddress = 7;  optional string City = 8;  optional string Country = 9;  optional string Topic = 10;  optional string EventType = 11;  optional bytes Data = 12;  } |
| --- |

Step 2: Generate Java code using above schema:

* Download 3.19.6 jar for a suitable OS from <https://github.com/protocolbuffers/protobuf/releases>

This jar will be useful for generating protobuf java code. Set up the environment variable to the bin folder after unzipping the downloaded zip file.

* Create shell script(.sh file) with content as below: (say my\_file.sh)

| cd $(dirname $0)  SRC\_DIR=.  DST\_DIR=/home/ec2-user/test\_programs/protobuf\_examples  protoc -I=$SRC\_DIR --java\_out=$DST\_DIR $SRC\_DIR/collector.proto |
| --- |

* Execute shell script using: sh my\_file.sh. You will see all .java protbuf files generated.
* Compile the java files using: java -cp “.:\*” \*.java
* Copy all the class files to the folder where kafka jar files exist.
* Create a schema created in step 1 in Glue under registry before running the producer.
* Create below Producer.

| import com.amazonaws.services.schemaregistry.serializers.GlueSchemaRegistryKafkaSerializer;  import com.amazonaws.services.schemaregistry.utils.AWSSchemaRegistryConstants;  import com.amazonaws.services.schemaregistry.utils.ProtobufMessageType;  import org.apache.kafka.clients.producer.\*;  import org.apache.kafka.common.serialization.StringSerializer;  import software.amazon.awssdk.services.glue.model.DataFormat;  import com.google.protobuf.ByteString;  import java.util.Properties;  import java.time.Instant;  import java.time.OffsetDateTime;  public class ProtoBufProducer {  public static void main(String[] args) throws InterruptedException{  String topic = "protobuf-cdi2-topic";  String bootstrapServers = "b-3.cdi2pocprivatecdk.3pcb6k.c10.kafka.us-west-2.amazonaws.com:9098,b-2.cdi2pocprivatecdk.3pcb6k.c10.kafka.us-west-2.amazonaws.com:9098,b-1.cdi2pocprivatecdk.3pcb6k.c10.kafka.us-west-2.amazonaws.com:9098";  Properties props = new Properties();  props.put("security.protocol", "SASL\_SSL");  props.put("sasl.mechanism", "AWS\_MSK\_IAM");  props.put("sasl.jaas.config", "software.amazon.msk.auth.iam.IAMLoginModule required;");  props.put("sasl.client.callback.handler.class", "software.amazon.msk.auth.iam.IAMClientCallbackHandler");  props.put(ProducerConfig.BOOTSTRAP\_SERVERS\_CONFIG, bootstrapServers);  props.put(ProducerConfig.KEY\_SERIALIZER\_CLASS\_CONFIG, StringSerializer.class.getName());  props.put(ProducerConfig.VALUE\_SERIALIZER\_CLASS\_CONFIG, GlueSchemaRegistryKafkaSerializer.class.getName());  props.put(AWSSchemaRegistryConstants.DATA\_FORMAT, DataFormat.PROTOBUF.name());  props.put(AWSSchemaRegistryConstants.AWS\_REGION,"us-west-2");  props.put(AWSSchemaRegistryConstants.REGISTRY\_NAME, "cdi2\_test");  //props.put(AWSSchemaRegistryConstants.SCHEMA\_NAME, "test.proto");  props.put(AWSSchemaRegistryConstants.SCHEMA\_NAME, "Collector.proto");  props.put(AWSSchemaRegistryConstants.PROTOBUF\_MESSAGE\_TYPE, ProtobufMessageType.POJO.getName());  String exampleInput = "2020-08-27T20:13:10+02:00";  Instant javaTimeInstant = OffsetDateTime.parse(exampleInput).toInstant();  com.google.protobuf.Timestamp ts = com.google.protobuf.Timestamp.newBuilder()  .setSeconds(javaTimeInstant.getEpochSecond())  .setNanos(javaTimeInstant.getNano())  .build();  //SegmentData.Builder segBuilder = SegmentData.newBuilder();  //String email = "{\"\_metadata\":{\"bundled\":[\"Adobe Analytics\",\"Chartbeat\",\"comScore\",\"DoubleClick Floodlight\",\"Google AdWords New\",\"Google Tag Manager\",\"Nielsen DTVR\",\"Optimizely\",\"Segment.io\"],\"bundledIds\":[\"5ee7fc26da14581236dccb61\",\"5fad927ca696c58c2ec29a3c\",\"609c3e0e7e45125be62b262a\",\"5ed18a0e9be5bbc845b89a7d\",\"5ed18a5bcc36af81cbae8c8f\",\"5ed18a7c74db4b599c830df4\",\"5ed18ab4730267d93eff97a3\",\"5ed18b11ce1fe182b3c32528\"],\"unbundled\":[]},\"anonymousId\":\"6b90a26f-18f9-4f3a-a841-e898cd40f111\",\"channel\":\"client\",\"context\":{\"ip\":\"71.78.47.82\",\"library\":{\"name\":\"analytics.js\",\"version\":\"4.1.8\"},\"locale\":\"en-US\",\"page\":{\"path\":\"/live/fs1\",\"referrer\":\"https://www.foxsports.com/live\",\"search\":\"\",\"title\":\"Watch FS1 Live | Stream Games & Shows on FS1 | FOX Sports\",\"url\":\"https://www.foxsports.com/live/fs1\"},\"protocols\":{\"sourceId\":\"4VurYnkj2mjWbHyHJpKeSB\"},\"traits\":{\"appsflyer\_id\":\"\",\"campaign\_tracking\_code\":null,\"dcg\_profile\_id\":\"\",\"lastAnonymousProfileId\":\"d2ViYmE4MWI4NTYtYjQzYS01OWE4LTMwZjAtNjhkNThmOTc2MWFj\",\"lastKnownProfileId\":\"\",\"mvpd\":\"Spectrum\",\"network\_entitlement\_list\":\"fbc-fox,foxdep,fs1,fs2\"},\"userAgent\":\"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.82 Safari/537.36\"},\"event\":\"Nielsen ID3 Tag Detected\",\"integrations\":{},\"messageId\":\"ajs-8c6e2f65a69ff0a33f52903a87d011e8\",\"originalTimestamp\":\"2021-09-17T09:42:22.479Z\",\"projectId\":\"4VurYnkj2mjWbHyHJpKeSB\",\"properties\":{\"app\_build\":\"384\",\"app\_name\":\"foxsports.com\",\"app\_platform\":\"web\",\"app\_version\":\"384\",\"favorites\_athletes\_length\":0,\"favorites\_content\_length\":0,\"favorites\_sports\_length\":0,\"favorites\_teams\_length\":0,\"load\_type\":\"dynamic\",\"nielsen\_id3\_tag\":\"www.nielsen.com/X100zdCIGeIlgZnkYj6UvQ==/pGhOkAY66nw7wvMjOlZk5Q==/fScC8QO8AibkJldngSRRIHboD1ZvnV\_mB-agezEbzyHb3UMJv998m8u8ehAOZ8QeNg6XYV-iLUKPFxyWMrK2nwi1yXFQX0nVw\_wXQIYxLZD2ch1-YehzbKcNyq3XWqoqDpF\_X100zdCIGeIlgZnkYj6UvWFEYs5hRGLYN7E=/00000/09699/00\",\"page\_login\_state\":\"logged out\",\"page\_name\":\"fscom:live-tv:fs1\",\"page\_type\":\"live-tv\",\"position\":0,\"primary\_business\_unit\":\"fng\",\"secondary\_business\_unit\":\"fox sports\",\"session\_id\":\"EP024970980730H\",\"us\_privacy\":\"1---\",\"video\_ad\_model\":\"linear\",\"video\_ad\_supported\":true,\"video\_airplay\":false,\"video\_asset\_category\":\"Live\",\"video\_asset\_title\":\"SKIP and SHANNON: UNDISPUTED\",\"video\_authorizing\_network\":\"fs1\",\"video\_content\_cdn\":\"limelight\",\"video\_content\_channel\":\"fs1\",\"video\_content\_length\":9000,\"video\_content\_length\_format\":\"long-form\",\"video\_content\_platform\":\"vdms\",\"video\_content\_subscription\_type\":\"locked\",\"video\_content\_type\":\"Live\",\"video\_cross\_device\_play\":false,\"video\_episode\_number\":\"189\",\"video\_feed\":\"FS1 Live\",\"video\_first\_air\_date\":\"2021-09-16T12:00:00.000Z\",\"video\_first\_digital\_date\":\"2021-09-16T13:30:00.000Z\",\"video\_fox\_profile\":false,\"video\_freewheel\_id\":\"120876101910\",\"video\_genre\":\"Sports\",\"video\_is\_autoplay\":true,\"video\_is\_fullscreen\":false,\"video\_is\_livestream\":true,\"video\_is\_restart\":false,\"video\_is\_resume\":false,\"video\_max\_bitrate\":0,\"video\_media\_type\":\"video\",\"video\_network\":\"fs1\",\"video\_nielsen\_clientid\":\"us-800251\",\"video\_nielsen\_subbrand\":\"c08\",\"video\_originator\":\"fs1\",\"video\_playback\_speed\":\"1\",\"video\_player\_content\_type\":\"main\",\"video\_player\_state\":\"ready\",\"video\_player\_type\":\"standard player\",\"video\_primary\_business\_unit\":\"fng\",\"video\_screen\_layout\":\"no multiview\",\"video\_season\_number\":\"2021\",\"video\_secondary\_business\_unit\":\"fox sports\",\"video\_series\_name\":\"Skip and Shannon: Undisputed\",\"video\_station\_id\":\"FS1\",\"video\_tms\_id\":\"EP024970980730\",\"video\_uid\":\"skip-and-shannon-undisputed\_ep024970980730\",\"video\_volume\":100},\"receivedAt\":\"2021-09-17T09:42:20.611Z\",\"sentAt\":\"2021-09-17T09:42:22.488Z\",\"timestamp\":\"2021-09-17T09:42:20.602Z\",\"type\":\"track\",\"userId\":null,\"version\":2}";  //SegmentData seg = segBuilder.setEmail(email).setId(2).setName("test\_name").build();  CollectorEvent.Builder collBuilder = CollectorEvent.newBuilder();  String data = "{\"\_metadata\":{\"bundled\":[\"Adobe Analytics\",\"Chartbeat\",\"comScore\",\"DoubleClick Floodlight\",\"Google AdWords New\",\"Google Tag Manager\",\"Nielsen DTVR\",\"Optimizely\",\"Segment.io\"],\"bundledIds\":[\"5ee7fc26da14581236dccb61\",\"5fad927ca696c58c2ec29a3c\",\"609c3e0e7e45125be62b262a\",\"5ed18a0e9be5bbc845b89a7d\",\"5ed18a5bcc36af81cbae8c8f\",\"5ed18a7c74db4b599c830df4\",\"5ed18ab4730267d93eff97a3\",\"5ed18b11ce1fe182b3c32528\"],\"unbundled\":[]},\"anonymousId\":\"6b90a26f-18f9-4f3a-a841-e898cd40f111\",\"channel\":\"client\",\"context\":{\"ip\":\"71.78.47.82\",\"library\":{\"name\":\"analytics.js\",\"version\":\"4.1.8\"},\"locale\":\"en-US\",\"page\":{\"path\":\"/live/fs1\",\"referrer\":\"https://www.foxsports.com/live\",\"search\":\"\",\"title\":\"Watch FS1 Live | Stream Games & Shows on FS1 | FOX Sports\",\"url\":\"https://www.foxsports.com/live/fs1\"},\"protocols\":{\"sourceId\":\"4VurYnkj2mjWbHyHJpKeSB\"},\"traits\":{\"appsflyer\_id\":\"\",\"campaign\_tracking\_code\":null,\"dcg\_profile\_id\":\"\",\"lastAnonymousProfileId\":\"d2ViYmE4MWI4NTYtYjQzYS01OWE4LTMwZjAtNjhkNThmOTc2MWFj\",\"lastKnownProfileId\":\"\",\"mvpd\":\"Spectrum\",\"network\_entitlement\_list\":\"fbc-fox,foxdep,fs1,fs2\"},\"userAgent\":\"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.82 Safari/537.36\"},\"event\":\"Nielsen ID3 Tag Detected\",\"integrations\":{},\"messageId\":\"ajs-8c6e2f65a69ff0a33f52903a87d011e8\",\"originalTimestamp\":\"2021-09-17T09:42:22.479Z\",\"projectId\":\"4VurYnkj2mjWbHyHJpKeSB\",\"properties\":{\"app\_build\":\"384\",\"app\_name\":\"foxsports.com\",\"app\_platform\":\"web\",\"app\_version\":\"384\",\"favorites\_athletes\_length\":0,\"favorites\_content\_length\":0,\"favorites\_sports\_length\":0,\"favorites\_teams\_length\":0,\"load\_type\":\"dynamic\",\"nielsen\_id3\_tag\":\"www.nielsen.com/X100zdCIGeIlgZnkYj6UvQ==/pGhOkAY66nw7wvMjOlZk5Q==/fScC8QO8AibkJldngSRRIHboD1ZvnV\_mB-agezEbzyHb3UMJv998m8u8ehAOZ8QeNg6XYV-iLUKPFxyWMrK2nwi1yXFQX0nVw\_wXQIYxLZD2ch1-YehzbKcNyq3XWqoqDpF\_X100zdCIGeIlgZnkYj6UvWFEYs5hRGLYN7E=/00000/09699/00\",\"page\_login\_state\":\"logged out\",\"page\_name\":\"fscom:live-tv:fs1\",\"page\_type\":\"live-tv\",\"position\":0,\"primary\_business\_unit\":\"fng\",\"secondary\_business\_unit\":\"fox sports\",\"session\_id\":\"EP024970980730H\",\"us\_privacy\":\"1---\",\"video\_ad\_model\":\"linear\",\"video\_ad\_supported\":true,\"video\_airplay\":false,\"video\_asset\_category\":\"Live\",\"video\_asset\_title\":\"SKIP and SHANNON: UNDISPUTED\",\"video\_authorizing\_network\":\"fs1\",\"video\_content\_cdn\":\"limelight\",\"video\_content\_channel\":\"fs1\",\"video\_content\_length\":9000,\"video\_content\_length\_format\":\"long-form\",\"video\_content\_platform\":\"vdms\",\"video\_content\_subscription\_type\":\"locked\",\"video\_content\_type\":\"Live\",\"video\_cross\_device\_play\":false,\"video\_episode\_number\":\"189\",\"video\_feed\":\"FS1 Live\",\"video\_first\_air\_date\":\"2021-09-16T12:00:00.000Z\",\"video\_first\_digital\_date\":\"2021-09-16T13:30:00.000Z\",\"video\_fox\_profile\":false,\"video\_freewheel\_id\":\"120876101910\",\"video\_genre\":\"Sports\",\"video\_is\_autoplay\":true,\"video\_is\_fullscreen\":false,\"video\_is\_livestream\":true,\"video\_is\_restart\":false,\"video\_is\_resume\":false,\"video\_max\_bitrate\":0,\"video\_media\_type\":\"video\",\"video\_network\":\"fs1\",\"video\_nielsen\_clientid\":\"us-800251\",\"video\_nielsen\_subbrand\":\"c08\",\"video\_originator\":\"fs1\",\"video\_playback\_speed\":\"1\",\"video\_player\_content\_type\":\"main\",\"video\_player\_state\":\"ready\",\"video\_player\_type\":\"standard player\",\"video\_primary\_business\_unit\":\"fng\",\"video\_screen\_layout\":\"no multiview\",\"video\_season\_number\":\"2021\",\"video\_secondary\_business\_unit\":\"fox sports\",\"video\_series\_name\":\"Skip and Shannon: Undisputed\",\"video\_station\_id\":\"FS1\",\"video\_tms\_id\":\"EP024970980730\",\"video\_uid\":\"skip-and-shannon-undisputed\_ep024970980730\",\"video\_volume\":100},\"receivedAt\":\"2021-09-17T09:42:20.611Z\",\"sentAt\":\"2021-09-17T09:42:22.488Z\",\"timestamp\":\"2021-09-17T09:42:20.602Z\",\"type\":\"track\",\"userId\":null,\"version\":2}";  CollectorEvent payload = collBuilder.setOrigin("cdi2")  .setChannel("")  .setPlatform("platform")  .setSentAt(ts)  .setEventID("294ebad0-08e0-465e-8edc-28599312af1a")  .setAnonymousId("c43e21ab-3189-48eb-b34c-a42387ecbbd5")  .setIPAddress("172.17.0.1:60182")  .setCity("NA")  .setCountry("NA")  .setTopic("CDI2\_TOPIC")  .setEventType("default")  .setData(ByteString.copyFromUtf8(data))  .build();  //CollBuilder.s  KafkaProducer<String, CollectorEvent> producer = new KafkaProducer<String, CollectorEvent>(props);  int i=0;  while(i<10){  String key = "key-" + i;  ProducerRecord<String, CollectorEvent> record = new ProducerRecord<String, CollectorEvent>(topic, key, payload);  producer.send(record, new ProducerCallback());  i++;  }  producer.flush();  }  } |
| --- |

* Kafka S3 Sink:

| connector.class=io.confluent.connect.s3.S3SinkConnector  value.converter.schemaAutoRegistrationEnabled=true  s3.region=us-west-2  key.converter.endpoint=https://glue.us-west-2.amazonaws.com  flush.size=10  tasks.max=1  key.converter.region=us-west-2  value.converter.endpoint=https://glue.us-west-2.amazonaws.com  key.converter.schemaName=Collector.proto  internal.key.converter.schemas.enable=false  format.class=io.confluent.connect.s3.format.parquet.ParquetFormat  value.converter.schemaName=Collector.proto  key.converter.dataFormat=PROTBUF  value.converter=com.amazonaws.services.schemaregistry.kafkaconnect.protobuf.ProtobufSchemaConverter  s3.bucket.name=segment-testing-bucket-1  key.converter=org.apache.kafka.connect.storage.StringConverter  schema.compatibility=FULL  topics=protobuf-cdi2-topic  value.converter.dataFormat=PROTBUF  value.converter.registry.name=cdi2\_test  key.converter.registry.name=cdi2\_test  value.converter.region=us-west-2  internal.key.converter=org.apache.kafka.connect.storage.StringConverter  key.converter.schemas.enable=false  key.converter.schemaAutoRegistrationEnabled=true  partitioner.class=io.confluent.connect.storage.partitioner.DefaultPartitioner  internal.value.converter.schemas.enable=false  internal.value.converter=org.apache.kafka.connect.json.JsonConverter  value.converter.schemas.enable=true  storage.class=io.confluent.connect.s3.storage.S3Storage |
| --- |

#### ProtoBuf Producer:

#### ProtoBuf Consumer:

## MSK Connect Logs:

## **For Analytics options: (Athena, Pyspark)**

### **Analytics using AWS Athena**

1. From AWS console, open Athena
2. Click on Launch query editor.
3. Example Query that can be performed in Athena:

#### Json

| CREATE EXTERNAL TABLE IF NOT EXISTS my\_testtable1(  firstName string,  lastName string,  country string,  city string,  state string)  ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'  WITH SERDEPROPERTIES ('serialization.format' = '0')  LOCATION 's3://mskconnectbucket1/topics/TestTopic3/partition=0/'; |
| --- |

#### JsonSchema

| CREATE EXTERNAL TABLE IF NOT EXISTS JsonSchemaParquet\_parquettable(  Origin string,  Channel string,  Platform string,  SentAt string,  EventID string,  AnonymousId string,  IPAddress string,  City string,  Topic string,  EventType string,  Data string  )  STORED AS PARQUET  LOCATION 's3://mskconnectbucket1/topics/TestJsonSchema/partition=0/'  tblproperties ("parquet.compression"="SNAPPY"); |
| --- |

#### Avro

| CREATE EXTERNAL TABLE test\_table (  id STRING,  amount double  )  ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'  WITH SERDEPROPERTIES ('avro.schema.literal'='  {  "type": "record",  "name": "Payment",  "fields": [  {"name": "id", "type": "string"},  {"name": "amount", "type": "double"}  ]  }  ')  STORED AS AVRO  LOCATION 's3://segment-testing-bucket-1/topics/cdi2glueavro2/partition=0/';  --TBLPROPERTIES ('compressionType'='snappy'); -- if compression enabled |
| --- |

1. To view results, use the query;

* Now your producer has started, enter any message and press enter. Repeat the same step and enter messages 2-3 times. Every time you press “enter” for the next message that line is sent as a separate message.

| select \* from my\_testtable1; |
| --- |

1. Finally, drop the table using the query;

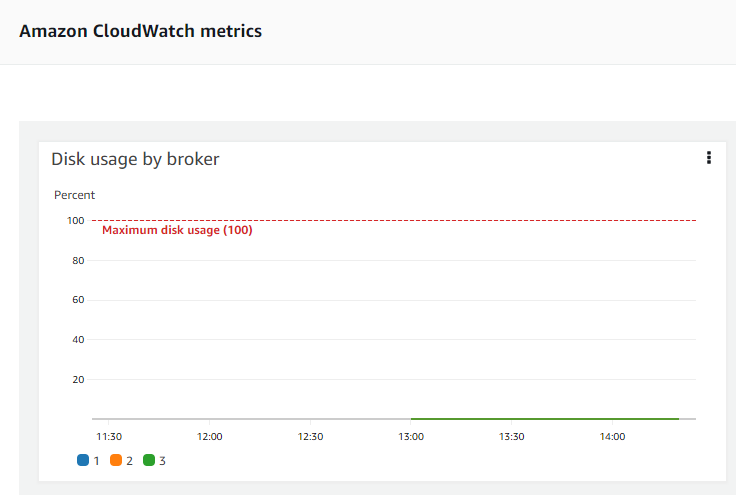
| drop table if exists my\_testtable1; |
| --- |

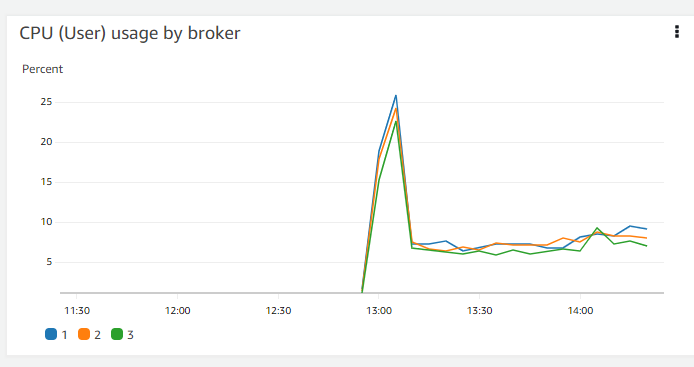
#### ProtoBuf

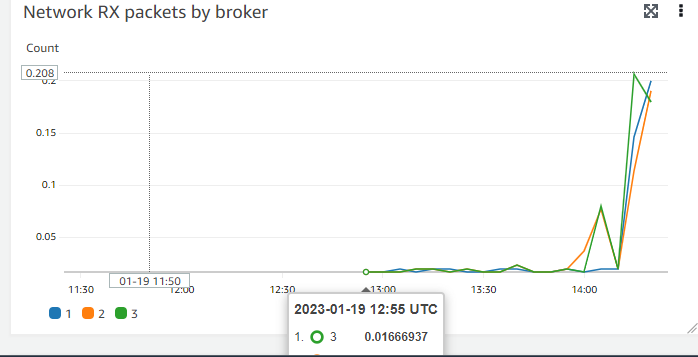
## AWS Billing:

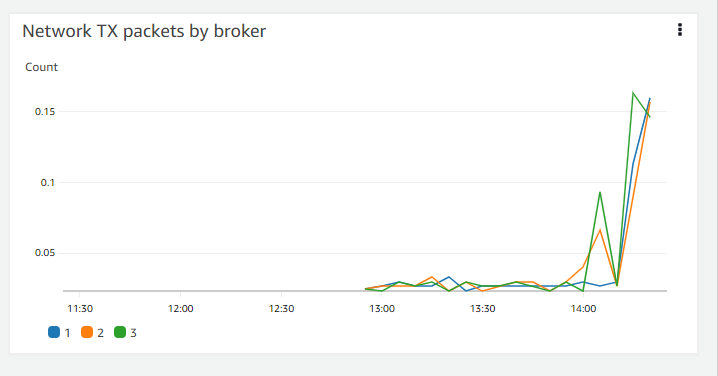
## Metrics:

### MSK cluster metrics:









### MSK Connect connectors metrics:

## Errors and Resolutions

### When s3 endpoint is not mentioned

* When BootstrapServer is not assigned.

| log4j:WARN No appenders could be found for logger (org.apache.kafka.clients.producer.ProducerConfig).  log4j:WARN Please initialize the log4j system properly.  log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.  Exception in thread "main" org.apache.kafka.common.KafkaException: Failed to construct kafka producer  at org.apache.kafka.clients.producer.KafkaProducer.<init>(KafkaProducer.java:439)  at org.apache.kafka.clients.producer.KafkaProducer.<init>(KafkaProducer.java:290)  at org.apache.kafka.clients.producer.KafkaProducer.<init>(KafkaProducer.java:317)  at org.apache.kafka.clients.producer.KafkaProducer.<init>(KafkaProducer.java:302)  at Producer\_IAM.main(Producer\_IAM.java:27)  Caused by: org.apache.kafka.common.config.ConfigException: No resolvable bootstrap urls given in bootstrap.servers  at org.apache.kafka.clients.ClientUtils.parseAndValidateAddresses(ClientUtils.java:88)  at org.apache.kafka.clients.ClientUtils.parseAndValidateAddresses(ClientUtils.java:47)  at org.apache.kafka.clients.producer.KafkaProducer.<init>(KafkaProducer.java:413)  ... 4 more |
| --- |

### When glue endpoint is not mentioned.

Error:

| [2023-01-04 15:50:32,068] ERROR [MskConnect01|task-0] WorkerSinkTask{id=MskConnect01-0} Error converting message value in topic 'TestAvroTopic' partition 0 at offset 0 and timestamp 1672845972283: Converting byte[] to Kafka Connect data failed due to serialization error: (org.apache.kafka.connect.runtime.WorkerSinkTask:547)  [Worker-0c75d189e32130784] org.apache.kafka.connect.errors.DataException: Converting byte[] to Kafka Connect data failed due to serialization error:  [Worker-0c75d189e32130784] at com.amazonaws.services.schemaregistry.kafkaconnect.AWSKafkaAvroConverter.toConnectData(AWSKafkaAvroConverter.java:118)  [Worker-0c75d189e32130784] at org.apache.kafka.connect.storage.Converter.toConnectData(Converter.java:87)  [Worker-0c75d189e32130784] at org.apache.kafka.connect.runtime.WorkerSinkTask.convertValue(WorkerSinkTask.java:545)  [Worker-0c75d189e32130784] at org.apache.kafka.connect.runtime.WorkerSinkTask.lambda$convertAndTransformRecord$1(WorkerSinkTask.java:501)  [Worker-0c75d189e32130784] at org.apache.kafka.connect.runtime.errors.RetryWithToleranceOperator.execAndRetry(RetryWithToleranceOperator.java:156)  [Worker-0c75d189e32130784] at org.apache.kafka.connect.runtime.errors.RetryWithToleranceOperator.execAndHandleError(RetryWithToleranceOperator.java:190)  [Worker-0c75d189e32130784] at org.apache.kafka.connect.runtime.errors.RetryWithToleranceOperator.execute(RetryWithToleranceOperator.java:132)  [Worker-0c75d189e32130784] at org.apache.kafka.connect.runtime.WorkerSinkTask.convertAndTransformRecord(WorkerSinkTask.java:501)  [Worker-0c75d189e32130784] at org.apache.kafka.connect.runtime.WorkerSinkTask.convertMessages(WorkerSinkTask.java:478)  [Worker-0c75d189e32130784] at org.apache.kafka.connect.runtime.WorkerSinkTask.poll(WorkerSinkTask.java:328)  [Worker-0c75d189e32130784] at org.apache.kafka.connect.runtime.WorkerSinkTask.iteration(WorkerSinkTask.java:232)  [Worker-0c75d189e32130784] at org.apache.kafka.connect.runtime.WorkerSinkTask.execute(WorkerSinkTask.java:201)  [Worker-0c75d189e32130784] at org.apache.kafka.connect.runtime.WorkerTask.doRun(WorkerTask.java:189)  [Worker-0c75d189e32130784] at org.apache.kafka.connect.runtime.WorkerTask.run(WorkerTask.java:238)  [Worker-0c75d189e32130784] at java.base/java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:515)  [Worker-0c75d189e32130784] at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:264)  [Worker-0c75d189e32130784] at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1128)  [Worker-0c75d189e32130784] at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:628)  [Worker-0c75d189e32130784] at java.base/java.lang.Thread.run(Thread.java:829)  [Worker-0c75d189e32130784] Caused by: com.amazonaws.services.schemaregistry.exception.AWSSchemaRegistryException: com.amazonaws.services.schemaregistry.exception.AWSSchemaRegistryException: Failed to get schema version Id = 6498496c-16e6-4ee8-a0bb-40ec9ac6b57d  [Worker-0c75d189e32130784] at com.amazonaws.services.schemaregistry.deserializers.GlueSchemaRegistryDeserializationFacade.retrieveSchemaRegistrySchema(GlueSchemaRegistryDeserializationFacade.java:219)  [Worker-0c75d189e32130784] at com.amazonaws.services.schemaregistry.deserializers.GlueSchemaRegistryDeserializationFacade.getAwsDeserializerSchema(GlueSchemaRegistryDeserializationFacade.java:201)  [Worker-0c75d189e32130784] at com.amazonaws.services.schemaregistry.deserializers.GlueSchemaRegistryDeserializationFacade.deserialize(GlueSchemaRegistryDeserializationFacade.java:167)  [Worker-0c75d189e32130784] at com.amazonaws.services.schemaregistry.deserializers.avro.AWSKafkaAvroDeserializer.deserializeByHeaderVersionByte(AWSKafkaAvroDeserializer.java:149)  [Worker-0c75d189e32130784] at com.amazonaws.services.schemaregistry.deserializers.avro.AWSKafkaAvroDeserializer.deserialize(AWSKafkaAvroDeserializer.java:114)  [Worker-0c75d189e32130784] at com.amazonaws.services.schemaregistry.kafkaconnect.AWSKafkaAvroConverter.toConnectData(AWSKafkaAvroConverter.java:116)  [Worker-0c75d189e32130784] ... 18 more  [Worker-0c75d189e32130784] Caused by: com.amazonaws.services.schemaregistry.exception.AWSSchemaRegistryException: Failed to get schema version Id = 6498496c-16e6-4ee8-a0bb-40ec9ac6b57d  [Worker-0c75d189e32130784] at com.amazonaws.services.schemaregistry.common.AWSSchemaRegistryClient.getSchemaVersionResponse(AWSSchemaRegistryClient.java:170)  [Worker-0c75d189e32130784] at com.amazonaws.services.schemaregistry.deserializers.GlueSchemaRegistryDeserializationFacade$GlueSchemaRegistryDeserializationCacheLoader.load(GlueSchemaRegistryDeserializationFacade.java:257)  [Worker-0c75d189e32130784] at com.amazonaws.services.schemaregistry.deserializers.GlueSchemaRegistryDeserializationFacade$GlueSchemaRegistryDeserializationCacheLoader.load(GlueSchemaRegistryDeserializationFacade.java:253)  [Worker-0c75d189e32130784] at com.google.common.cache.LocalCache$LoadingValueReference.loadFuture(LocalCache.java:3529)  [Worker-0c75d189e32130784] at com.google.common.cache.LocalCache$Segment.loadSync(LocalCache.java:2278)  [Worker-0c75d189e32130784] at com.google.common.cache.LocalCache$Segment.lockedGetOrLoad(LocalCache.java:2155)  [Worker-0c75d189e32130784] at com.google.common.cache.LocalCache$Segment.get(LocalCache.java:2045)  [Worker-0c75d189e32130784] at com.google.common.cache.LocalCache.get(LocalCache.java:3951)  [Worker-0c75d189e32130784] at com.google.common.cache.LocalCache.getOrLoad(LocalCache.java:3974)  [Worker-0c75d189e32130784] at com.google.common.cache.LocalCache$LocalLoadingCache.get(LocalCache.java:4935)  [Worker-0c75d189e32130784] at com.amazonaws.services.schemaregistry.deserializers.GlueSchemaRegistryDeserializationFacade.retrieveSchemaRegistrySchema(GlueSchemaRegistryDeserializationFacade.java:217)  [Worker-0c75d189e32130784] ... 23 more  [Worker-0c75d189e32130784] Caused by: software.amazon.awssdk.core.exception.SdkClientException: Unable to execute HTTP request: connect timed out  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.exception.SdkClientException$BuilderImpl.build(SdkClientException.java:98)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.exception.SdkClientException.create(SdkClientException.java:43)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.utils.RetryableStageHelper.setLastException(RetryableStageHelper.java:204)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.RetryableStage.execute(RetryableStage.java:83)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.RetryableStage.execute(RetryableStage.java:36)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.RequestPipelineBuilder$ComposingRequestPipelineStage.execute(RequestPipelineBuilder.java:206)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.StreamManagingStage.execute(StreamManagingStage.java:56)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.StreamManagingStage.execute(StreamManagingStage.java:36)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.ApiCallTimeoutTrackingStage.executeWithTimer(ApiCallTimeoutTrackingStage.java:80)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.ApiCallTimeoutTrackingStage.execute(ApiCallTimeoutTrackingStage.java:60)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.ApiCallTimeoutTrackingStage.execute(ApiCallTimeoutTrackingStage.java:42)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.ApiCallMetricCollectionStage.execute(ApiCallMetricCollectionStage.java:48)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.ApiCallMetricCollectionStage.execute(ApiCallMetricCollectionStage.java:31)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.RequestPipelineBuilder$ComposingRequestPipelineStage.execute(RequestPipelineBuilder.java:206)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.RequestPipelineBuilder$ComposingRequestPipelineStage.execute(RequestPipelineBuilder.java:206)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.ExecutionFailureExceptionReportingStage.execute(ExecutionFailureExceptionReportingStage.java:37)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.ExecutionFailureExceptionReportingStage.execute(ExecutionFailureExceptionReportingStage.java:26)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.AmazonSyncHttpClient$RequestExecutionBuilderImpl.execute(AmazonSyncHttpClient.java:193)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.handler.BaseSyncClientHandler.invoke(BaseSyncClientHandler.java:103)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.handler.BaseSyncClientHandler.doExecute(BaseSyncClientHandler.java:167)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.handler.BaseSyncClientHandler.lambda$execute$1(BaseSyncClientHandler.java:82)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.handler.BaseSyncClientHandler.measureApiCallSuccess(BaseSyncClientHandler.java:175)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.handler.BaseSyncClientHandler.execute(BaseSyncClientHandler.java:76)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.client.handler.SdkSyncClientHandler.execute(SdkSyncClientHandler.java:45)  [Worker-0c75d189e32130784] at software.amazon.awssdk.awscore.client.handler.AwsSyncClientHandler.execute(AwsSyncClientHandler.java:56)  [Worker-0c75d189e32130784] at software.amazon.awssdk.services.glue.DefaultGlueClient.getSchemaVersion(DefaultGlueClient.java:7387)  [Worker-0c75d189e32130784] at com.amazonaws.services.schemaregistry.common.AWSSchemaRegistryClient.getSchemaVersionResponse(AWSSchemaRegistryClient.java:166)  [Worker-0c75d189e32130784] ... 33 more  [Worker-0c75d189e32130784] Caused by: java.net.SocketTimeoutException: connect timed out  [Worker-0c75d189e32130784] at java.base/java.net.PlainSocketImpl.socketConnect(Native Method)  [Worker-0c75d189e32130784] at java.base/java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl.java:412)  [Worker-0c75d189e32130784] at java.base/java.net.AbstractPlainSocketImpl.connectToAddress(AbstractPlainSocketImpl.java:255)  [Worker-0c75d189e32130784] at java.base/java.net.AbstractPlainSocketImpl.connect(AbstractPlainSocketImpl.java:237)  [Worker-0c75d189e32130784] at java.base/java.net.SocksSocketImpl.connect(SocksSocketImpl.java:392)  [Worker-0c75d189e32130784] at java.base/java.net.Socket.connect(Socket.java:609)  [Worker-0c75d189e32130784] at java.base/sun.security.ssl.SSLSocketImpl.connect(SSLSocketImpl.java:305)  [Worker-0c75d189e32130784] at java.base/sun.net.NetworkClient.doConnect(NetworkClient.java:177)  [Worker-0c75d189e32130784] at java.base/sun.net.www.http.HttpClient.openServer(HttpClient.java:507)  [Worker-0c75d189e32130784] at java.base/sun.net.www.http.HttpClient.openServer(HttpClient.java:602)  [Worker-0c75d189e32130784] at java.base/sun.net.www.protocol.https.HttpsClient.<init>(HttpsClient.java:266)  [Worker-0c75d189e32130784] at java.base/sun.net.www.protocol.https.HttpsClient.New(HttpsClient.java:373)  [Worker-0c75d189e32130784] at java.base/sun.net.www.protocol.https.AbstractDelegateHttpsURLConnection.getNewHttpClient(AbstractDelegateHttpsURLConnection.java:207)  [Worker-0c75d189e32130784] at java.base/sun.net.www.protocol.http.HttpURLConnection.plainConnect0(HttpURLConnection.java:1187)  [Worker-0c75d189e32130784] at java.base/sun.net.www.protocol.http.HttpURLConnection.plainConnect(HttpURLConnection.java:1081)  [Worker-0c75d189e32130784] at java.base/sun.net.www.protocol.https.AbstractDelegateHttpsURLConnection.connect(AbstractDelegateHttpsURLConnection.java:193)  [Worker-0c75d189e32130784] at java.base/sun.net.www.protocol.https.HttpsURLConnectionImpl.connect(HttpsURLConnectionImpl.java:168)  [Worker-0c75d189e32130784] at software.amazon.awssdk.http.urlconnection.UrlConnectionHttpClient$RequestCallable.call(UrlConnectionHttpClient.java:209)  [Worker-0c75d189e32130784] at software.amazon.awssdk.http.urlconnection.UrlConnectionHttpClient$RequestCallable.call(UrlConnectionHttpClient.java:197)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.util.MetricUtils.measureDurationUnsafe(MetricUtils.java:64)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.MakeHttpRequestStage.executeHttpRequest(MakeHttpRequestStage.java:77)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.MakeHttpRequestStage.execute(MakeHttpRequestStage.java:56)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.MakeHttpRequestStage.execute(MakeHttpRequestStage.java:39)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.RequestPipelineBuilder$ComposingRequestPipelineStage.execute(RequestPipelineBuilder.java:206)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.RequestPipelineBuilder$ComposingRequestPipelineStage.execute(RequestPipelineBuilder.java:206)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.RequestPipelineBuilder$ComposingRequestPipelineStage.execute(RequestPipelineBuilder.java:206)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.RequestPipelineBuilder$ComposingRequestPipelineStage.execute(RequestPipelineBuilder.java:206)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.ApiCallAttemptTimeoutTrackingStage.execute(ApiCallAttemptTimeoutTrackingStage.java:73)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.ApiCallAttemptTimeoutTrackingStage.execute(ApiCallAttemptTimeoutTrackingStage.java:42)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.TimeoutExceptionHandlingStage.execute(TimeoutExceptionHandlingStage.java:78)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.TimeoutExceptionHandlingStage.execute(TimeoutExceptionHandlingStage.java:40)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.ApiCallAttemptMetricCollectionStage.execute(ApiCallAttemptMetricCollectionStage.java:50)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.ApiCallAttemptMetricCollectionStage.execute(ApiCallAttemptMetricCollectionStage.java:36)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.RetryableStage.execute(RetryableStage.java:81)  [Worker-0c75d189e32130784] ... 56 more  [Worker-0c75d189e32130784] [2023-01-04 15:50:32,070] ERROR [MskConnect01|task-0] WorkerSinkTask{id=MskConnect01-0} Task threw an uncaught and unrecoverable exception. Task is being killed and will not recover until manually restarted (org.apache.kafka.connect.runtime.WorkerTask:191)  [Worker-0c75d189e32130784] org.apache.kafka.connect.errors.ConnectException: Tolerance exceeded in error handler  [Worker-0c75d189e32130784] at org.apache.kafka.connect.runtime.errors.RetryWithToleranceOperator.execAndHandleError(RetryWithToleranceOperator.java:206)  [Worker-0c75d189e32130784] at org.apache.kafka.connect.runtime.errors.RetryWithToleranceOperator.execute(RetryWithToleranceOperator.java:132)  [Worker-0c75d189e32130784] at org.apache.kafka.connect.runtime.WorkerSinkTask.convertAndTransformRecord(WorkerSinkTask.java:501)  [Worker-0c75d189e32130784] at org.apache.kafka.connect.runtime.WorkerSinkTask.convertMessages(WorkerSinkTask.java:478)  [Worker-0c75d189e32130784] at org.apache.kafka.connect.runtime.WorkerSinkTask.poll(WorkerSinkTask.java:328)  [Worker-0c75d189e32130784] at org.apache.kafka.connect.runtime.WorkerSinkTask.iteration(WorkerSinkTask.java:232)  [Worker-0c75d189e32130784] at org.apache.kafka.connect.runtime.WorkerSinkTask.execute(WorkerSinkTask.java:201)  [Worker-0c75d189e32130784] at org.apache.kafka.connect.runtime.WorkerTask.doRun(WorkerTask.java:189)  [Worker-0c75d189e32130784] at org.apache.kafka.connect.runtime.WorkerTask.run(WorkerTask.java:238)  [Worker-0c75d189e32130784] at java.base/java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:515)  [Worker-0c75d189e32130784] at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:264)  [Worker-0c75d189e32130784] at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1128)  [Worker-0c75d189e32130784] at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:628)  [Worker-0c75d189e32130784] at java.base/java.lang.Thread.run(Thread.java:829)  [Worker-0c75d189e32130784] Caused by: org.apache.kafka.connect.errors.DataException: Converting byte[] to Kafka Connect data failed due to serialization error:  [Worker-0c75d189e32130784] at com.amazonaws.services.schemaregistry.kafkaconnect.AWSKafkaAvroConverter.toConnectData(AWSKafkaAvroConverter.java:118)  [Worker-0c75d189e32130784] at org.apache.kafka.connect.storage.Converter.toConnectData(Converter.java:87)  [Worker-0c75d189e32130784] at org.apache.kafka.connect.runtime.WorkerSinkTask.convertValue(WorkerSinkTask.java:545)  [Worker-0c75d189e32130784] at org.apache.kafka.connect.runtime.WorkerSinkTask.lambda$convertAndTransformRecord$1(WorkerSinkTask.java:501)  [Worker-0c75d189e32130784] at org.apache.kafka.connect.runtime.errors.RetryWithToleranceOperator.execAndRetry(RetryWithToleranceOperator.java:156)  [Worker-0c75d189e32130784] at org.apache.kafka.connect.runtime.errors.RetryWithToleranceOperator.execAndHandleError(RetryWithToleranceOperator.java:190)  [Worker-0c75d189e32130784] ... 13 more  [Worker-0c75d189e32130784] Caused by: com.amazonaws.services.schemaregistry.exception.AWSSchemaRegistryException: com.amazonaws.services.schemaregistry.exception.AWSSchemaRegistryException: Failed to get schema version Id = 6498496c-16e6-4ee8-a0bb-40ec9ac6b57d  [Worker-0c75d189e32130784] at com.amazonaws.services.schemaregistry.deserializers.GlueSchemaRegistryDeserializationFacade.retrieveSchemaRegistrySchema(GlueSchemaRegistryDeserializationFacade.java:219)  [Worker-0c75d189e32130784] at com.amazonaws.services.schemaregistry.deserializers.GlueSchemaRegistryDeserializationFacade.getAwsDeserializerSchema(GlueSchemaRegistryDeserializationFacade.java:201)  [Worker-0c75d189e32130784] at com.amazonaws.services.schemaregistry.deserializers.GlueSchemaRegistryDeserializationFacade.deserialize(GlueSchemaRegistryDeserializationFacade.java:167)  [Worker-0c75d189e32130784] at com.amazonaws.services.schemaregistry.deserializers.avro.AWSKafkaAvroDeserializer.deserializeByHeaderVersionByte(AWSKafkaAvroDeserializer.java:149)  [Worker-0c75d189e32130784] at com.amazonaws.services.schemaregistry.deserializers.avro.AWSKafkaAvroDeserializer.deserialize(AWSKafkaAvroDeserializer.java:114)  [Worker-0c75d189e32130784] at com.amazonaws.services.schemaregistry.kafkaconnect.AWSKafkaAvroConverter.toConnectData(AWSKafkaAvroConverter.java:116)  [Worker-0c75d189e32130784] ... 18 more  [Worker-0c75d189e32130784] Caused by: com.amazonaws.services.schemaregistry.exception.AWSSchemaRegistryException: Failed to get schema version Id = 6498496c-16e6-4ee8-a0bb-40ec9ac6b57d  [Worker-0c75d189e32130784] at com.amazonaws.services.schemaregistry.common.AWSSchemaRegistryClient.getSchemaVersionResponse(AWSSchemaRegistryClient.java:170)  [Worker-0c75d189e32130784] at com.amazonaws.services.schemaregistry.deserializers.GlueSchemaRegistryDeserializationFacade$GlueSchemaRegistryDeserializationCacheLoader.load(GlueSchemaRegistryDeserializationFacade.java:257)  [Worker-0c75d189e32130784] at com.amazonaws.services.schemaregistry.deserializers.GlueSchemaRegistryDeserializationFacade$GlueSchemaRegistryDeserializationCacheLoader.load(GlueSchemaRegistryDeserializationFacade.java:253)  [Worker-0c75d189e32130784] at com.google.common.cache.LocalCache$LoadingValueReference.loadFuture(LocalCache.java:3529)  [Worker-0c75d189e32130784] at com.google.common.cache.LocalCache$Segment.loadSync(LocalCache.java:2278)  [Worker-0c75d189e32130784] at com.google.common.cache.LocalCache$Segment.lockedGetOrLoad(LocalCache.java:2155)  [Worker-0c75d189e32130784] at com.google.common.cache.LocalCache$Segment.get(LocalCache.java:2045)  [Worker-0c75d189e32130784] at com.google.common.cache.LocalCache.get(LocalCache.java:3951)  [Worker-0c75d189e32130784] at com.google.common.cache.LocalCache.getOrLoad(LocalCache.java:3974)  [Worker-0c75d189e32130784] at com.google.common.cache.LocalCache$LocalLoadingCache.get(LocalCache.java:4935)  [Worker-0c75d189e32130784] at com.amazonaws.services.schemaregistry.deserializers.GlueSchemaRegistryDeserializationFacade.retrieveSchemaRegistrySchema(GlueSchemaRegistryDeserializationFacade.java:217)  [Worker-0c75d189e32130784] ... 23 more  [Worker-0c75d189e32130784] Caused by: software.amazon.awssdk.core.exception.SdkClientException: Unable to execute HTTP request: connect timed out  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.exception.SdkClientException$BuilderImpl.build(SdkClientException.java:98)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.exception.SdkClientException.create(SdkClientException.java:43)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.utils.RetryableStageHelper.setLastException(RetryableStageHelper.java:204)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.RetryableStage.execute(RetryableStage.java:83)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.RetryableStage.execute(RetryableStage.java:36)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.RequestPipelineBuilder$ComposingRequestPipelineStage.execute(RequestPipelineBuilder.java:206)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.StreamManagingStage.execute(StreamManagingStage.java:56)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.StreamManagingStage.execute(StreamManagingStage.java:36)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.ApiCallTimeoutTrackingStage.executeWithTimer(ApiCallTimeoutTrackingStage.java:80)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.ApiCallTimeoutTrackingStage.execute(ApiCallTimeoutTrackingStage.java:60)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.ApiCallTimeoutTrackingStage.execute(ApiCallTimeoutTrackingStage.java:42)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.ApiCallMetricCollectionStage.execute(ApiCallMetricCollectionStage.java:48)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.ApiCallMetricCollectionStage.execute(ApiCallMetricCollectionStage.java:31)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.RequestPipelineBuilder$ComposingRequestPipelineStage.execute(RequestPipelineBuilder.java:206)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.RequestPipelineBuilder$ComposingRequestPipelineStage.execute(RequestPipelineBuilder.java:206)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.ExecutionFailureExceptionReportingStage.execute(ExecutionFailureExceptionReportingStage.java:37)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.ExecutionFailureExceptionReportingStage.execute(ExecutionFailureExceptionReportingStage.java:26)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.AmazonSyncHttpClient$RequestExecutionBuilderImpl.execute(AmazonSyncHttpClient.java:193)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.handler.BaseSyncClientHandler.invoke(BaseSyncClientHandler.java:103)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.handler.BaseSyncClientHandler.doExecute(BaseSyncClientHandler.java:167)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.handler.BaseSyncClientHandler.lambda$execute$1(BaseSyncClientHandler.java:82)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.handler.BaseSyncClientHandler.measureApiCallSuccess(BaseSyncClientHandler.java:175)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.handler.BaseSyncClientHandler.execute(BaseSyncClientHandler.java:76)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.client.handler.SdkSyncClientHandler.execute(SdkSyncClientHandler.java:45)  [Worker-0c75d189e32130784] at software.amazon.awssdk.awscore.client.handler.AwsSyncClientHandler.execute(AwsSyncClientHandler.java:56)  [Worker-0c75d189e32130784] at software.amazon.awssdk.services.glue.DefaultGlueClient.getSchemaVersion(DefaultGlueClient.java:7387)  [Worker-0c75d189e32130784] at com.amazonaws.services.schemaregistry.common.AWSSchemaRegistryClient.getSchemaVersionResponse(AWSSchemaRegistryClient.java:166)  [Worker-0c75d189e32130784] ... 33 more  [Worker-0c75d189e32130784] Caused by: java.net.SocketTimeoutException: connect timed out  [Worker-0c75d189e32130784] at java.base/java.net.PlainSocketImpl.socketConnect(Native Method)  [Worker-0c75d189e32130784] at java.base/java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl.java:412)  [Worker-0c75d189e32130784] at java.base/java.net.AbstractPlainSocketImpl.connectToAddress(AbstractPlainSocketImpl.java:255)  [Worker-0c75d189e32130784] at java.base/java.net.AbstractPlainSocketImpl.connect(AbstractPlainSocketImpl.java:237)  [Worker-0c75d189e32130784] at java.base/java.net.SocksSocketImpl.connect(SocksSocketImpl.java:392)  [Worker-0c75d189e32130784] at java.base/java.net.Socket.connect(Socket.java:609)  [Worker-0c75d189e32130784] at java.base/sun.security.ssl.SSLSocketImpl.connect(SSLSocketImpl.java:305)  [Worker-0c75d189e32130784] at java.base/sun.net.NetworkClient.doConnect(NetworkClient.java:177)  [Worker-0c75d189e32130784] at java.base/sun.net.www.http.HttpClient.openServer(HttpClient.java:507)  [Worker-0c75d189e32130784] at java.base/sun.net.www.http.HttpClient.openServer(HttpClient.java:602)  [Worker-0c75d189e32130784] at java.base/sun.net.www.protocol.https.HttpsClient.<init>(HttpsClient.java:266)  [Worker-0c75d189e32130784] at java.base/sun.net.www.protocol.https.HttpsClient.New(HttpsClient.java:373)  [Worker-0c75d189e32130784] at java.base/sun.net.www.protocol.https.AbstractDelegateHttpsURLConnection.getNewHttpClient(AbstractDelegateHttpsURLConnection.java:207)  [Worker-0c75d189e32130784] at java.base/sun.net.www.protocol.http.HttpURLConnection.plainConnect0(HttpURLConnection.java:1187)  [Worker-0c75d189e32130784] at java.base/sun.net.www.protocol.http.HttpURLConnection.plainConnect(HttpURLConnection.java:1081)  [Worker-0c75d189e32130784] at java.base/sun.net.www.protocol.https.AbstractDelegateHttpsURLConnection.connect(AbstractDelegateHttpsURLConnection.java:193)  [Worker-0c75d189e32130784] at java.base/sun.net.www.protocol.https.HttpsURLConnectionImpl.connect(HttpsURLConnectionImpl.java:168)  [Worker-0c75d189e32130784] at software.amazon.awssdk.http.urlconnection.UrlConnectionHttpClient$RequestCallable.call(UrlConnectionHttpClient.java:209)  [Worker-0c75d189e32130784] at software.amazon.awssdk.http.urlconnection.UrlConnectionHttpClient$RequestCallable.call(UrlConnectionHttpClient.java:197)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.util.MetricUtils.measureDurationUnsafe(MetricUtils.java:64)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.MakeHttpRequestStage.executeHttpRequest(MakeHttpRequestStage.java:77)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.MakeHttpRequestStage.execute(MakeHttpRequestStage.java:56)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.MakeHttpRequestStage.execute(MakeHttpRequestStage.java:39)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.RequestPipelineBuilder$ComposingRequestPipelineStage.execute(RequestPipelineBuilder.java:206)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.RequestPipelineBuilder$ComposingRequestPipelineStage.execute(RequestPipelineBuilder.java:206)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.RequestPipelineBuilder$ComposingRequestPipelineStage.execute(RequestPipelineBuilder.java:206)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.RequestPipelineBuilder$ComposingRequestPipelineStage.execute(RequestPipelineBuilder.java:206)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.ApiCallAttemptTimeoutTrackingStage.execute(ApiCallAttemptTimeoutTrackingStage.java:73)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.ApiCallAttemptTimeoutTrackingStage.execute(ApiCallAttemptTimeoutTrackingStage.java:42)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.TimeoutExceptionHandlingStage.execute(TimeoutExceptionHandlingStage.java:78)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.TimeoutExceptionHandlingStage.execute(TimeoutExceptionHandlingStage.java:40)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.ApiCallAttemptMetricCollectionStage.execute(ApiCallAttemptMetricCollectionStage.java:50)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.ApiCallAttemptMetricCollectionStage.execute(ApiCallAttemptMetricCollectionStage.java:36)  [Worker-0c75d189e32130784] at software.amazon.awssdk.core.internal.http.pipeline.stages.RetryableStage.execute(RetryableStage.java:81)  [Worker-0c75d189e32130784] ... 56 more |
| --- |

### **When the wrong message format is sent.**

(Escape sequences are not placed properly)

| Producer\_IAM.java:12: error: ';' expected  String value = "{"firstName\":\"Aditya\",\"lastName\":\"Dhanraj\",\"age\":\"20\",\"city\":\"hyderabad\",\"state\":\"telangana\",\"country\":\"India\"}";  ^  Producer\_IAM.java:12: error: illegal character: '\'  String value = "{"firstName\":\"Aditya\",\"lastName\":\"Dhanraj\",\"age\":\"20\",\"city\":\"hyderabad\",\"state\":\"telangana\",\"country\":\"India\"}";  ^  2 errors |
| --- |

### **When the proper jar is not available in plugin jar/zip.**

|  |
| --- |

### The IAM setup is not properly set up in the Kafka client.

### Proper access to MSKC is not provided in IAM.

## **Clean up:**

Note: This process/steps are useful for avoiding extra costing for your AWS account.

Step1: Delete the kafka connect

* Open AWS console, search and open MSK .
* From the left box, under MSK clusters, select Connectors.
* Choose the Connector that you want to delete by selecting the Radio button before it.
* Now, click on Delete on the top right corner of the console.

Step2: Delete the Kafka cluster.

* Open AWS console, search and open MSK .
* From the left box, under MSK connect, select Clusters.
* Choose the Cluster that you want to delete by selecting the Radio button before it.
* Now, click on the Actions dropdown on the top right corner of the page.
* Finally, click on Delete cluster.

Step3: Delete EC2 instance.

* Open AWS console, search and open EC2 .
* Choose the instance that you want to delete by selecting the checkbox before it.
* Now, click Terminate instance.

Step4: Delete VPC Endpoints.

* Open AWS console, search and open VPC.
* From the left table, click on Endpoints.
* Choose the Endpoints that you want to delete by selecting the checkbox before it.
* Now, click on the Actions dropdown on the top right corner of the page.
* Finally, click on Delete VPC endpoints.

Step5: Delete Schema and Registry from Glue

| Registry Deletion:   * Open AWS console, search and open Glue. * In the left navigation pane, under Data catalog, choose Schema registries. * Choose a registry from the list, by checking a box. * In the Action menu, choose Delete registry. * Enter the text Delete in the field to confirm deletion. * Choose Delete. |
| --- |
| Schema Deletion:   * Open AWS console, search and open Glue. * In the left navigation pane, under Data catalog, choose Schemas. * Choose a schema from the list, by checking a box. * In the Action menu, choose Delete registry. * Enter the text Delete in the field to confirm deletion. * Choose Delete. |

Step6: Delete S3 logs.

* Open AWS console, search and open Glue.
* Click on the S3 Bucket, where the logs that you want to delete.
* Choose log objects from the list, by checking a box.
* Now, click on Delete.
* Enter the text “permanently delete” in the field to confirm deletion.
* Choose Delete.

## **What is IAM?**

IAM gives secure access to company resource-like emails, databases, data, and applications-to verified entities, ideally with a bare minimum of interference. The goal is to manage access so that the right people can do their jobs and the wrong people, like hackers, are denied entry.There are two parts to granting secure access to an organization’s resources: Identity management and access management.

User: The IAM user represents the human user or workload who uses the IAM user to interact with AWS. A user in AWS consists of a name and credentials.

User groups: An IAM user group is a collection of IAM users. User groups let you specify permissions for multiple users, which can make it easier to manage the permissions for those users.

Role: An IAM *role* is an IAM identity that you can create in your account that has specific permissions. An IAM role is similar to an IAM user, in that it is an AWS identity with permission policies that determine what the identity can and cannot do in AWS.

Policies: A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when an IAM principal (user or role) makes a request.

Features:

* Shared access to your AWS account.
* Granular permissions
* Secure access to AWS resources for applications that run on Amazon EC2
* Multi-factor authentication (MFA).
* Identity federation.
* Identity information for assurance.
* Integrated with many AWS services.
* PCI DSS Compliance.
* Free to use.

|  | **What is Kafka(AWS MSK Kafka cluster)?** |
| --- | --- |

It publishes and subscribes to a stream of records and also is used for fault-tolerant storage. The applications are designed to process the records of timing and usage.

Log partitions of different servers are replicated in Kafka. It stores, reads and analyzes the streaming data where developers and users contribute the coding updates. It is used for messaging, website activity tracking, log aggregation and commit logs. It can be used as a database, but it does not possess a data model or indexes.

Broker: A Broker is a Kafka server that runs in a Kafka Cluster. Kafka Brokers form a cluster. The Kafka Cluster consists of many Kafka Brokers on many servers.

Topic: It refers to a category or a common name used to store and publish a particular stream of data.

Kafka Partitioning: Partitioning takes the single topic log and breaks it into multiple logs, each of which can live on a separate node in the Kafka cluster.

Kafka Cluster: A Kafka cluster is a system that consists of several Brokers, Topics, and Partitions for both.

For more information go the link available under References -> [Kafka](https://docs.google.com/document/d/12fHUe3dPwudTJr1TCzO19YWe9rqJoXIv/edit#heading=h.3s49zyc)

|  | **What is Kafka Connect?** |
| --- | --- |

Kafka Connect acts as a mediator between the apache Kafka and different or other data-driven systems. The Kafka connector is nothing but a tool for reliable as well as scalable streaming solutions.

The Kafka Connect acts as a moderator in between the Kafka system and the external analytics world. But the scope of the Kafka connector is low. It is majorly focusing on the streaming data copy in and out from Kafka. The major focus is on the developer front to write or create high-quality code, reliable and high recital Kafka connector plugins, etc.

Plugins: It is an AWS resource that contains the code that defines your connector logic. You upload a JAR file (or a ZIP file that contains one or more JAR files) to an S3 bucket, and specify the location of the bucket when you create the plugin.

Workers: A worker is a Java virtual machine (JVM) process that runs the connector logic. Each worker creates a set of tasks that run in parallel threads and do the work of copying the data.

For more information go the link available under References -> [Kafka Connect](https://docs.google.com/document/d/12fHUe3dPwudTJr1TCzO19YWe9rqJoXIv/edit#heading=h.3s49zyc)

## **What is Glue?**

Glue is a serverless data integration service that makes it easy for analytics users to discover, prepare, move, and integrate data from multiple sources. You can use it for analytics, machine learning, and application development. It also includes additional productivity and data ops tooling for authoring, running jobs, and implementing business workflows.

Features:

* Discover and organize data:

1. Unify and search across multiple data stores.
2. Unify and search across multiple data stores.
3. Manage schemas and permissions.
4. Connect to a wide variety of data sources.
5. Visually transform data with a drag-and-drop interface.
6. Visually transform data with a drag-and-drop interface.
7. Clean and transform streaming data in transit .
8. Deduplicate and cleanse data with built-in machine learning.
9. Built-in job notebooks.
10. Edit, debug, and test ETL code.
11. Define, detect, and remediate sensitive data.

* Schemas: With Amazon Cloud Directory, schemas define what types of objects can be created within a directory (users, devices, and organizations), enforce validation of data for each object class, and handle changes to the schema over time.
* ETL: ETL, which stands for extract, transform and load, is a data integration process that combines data from multiple data sources into a single, consistent data store that is loaded into a data warehouse or other target system.
* Registry: Schema Registry provides a serving layer for your metadata. It provides a RESTful interface for storing and retrieving Avro schemas. It stores a versioned history of all schemas, provides multiple compatibility settings and allows evolution of schemas according to the configured compatibility setting.

For more information go the link available under References -> [Glue](https://docs.google.com/document/d/12fHUe3dPwudTJr1TCzO19YWe9rqJoXIv/edit#heading=h.meukdy)

## **What is VPC?**

A virtual private cloud (VPC) is a secure, isolated private cloud hosted within a public cloud. VPC customers can run code, store data, host websites, and do anything else they could do in an ordinary private cloud, but the private cloud is hosted remotely by a public cloud provider. (Not all private clouds are hosted in this fashion.) VPCs combine the scalability and convenience of public cloud computing with the data isolation of private cloud computing.

A public cloud is shared cloud infrastructure. Multiple customers of the cloud vendor access that same infrastructure, although their data is not shared – just like every person in a restaurant orders from the same kitchen, but they get different dishes. Public cloud service providers include AWS, Google Cloud Platform, and Microsoft Azure, among others.

A private cloud, however, is single-tenant. A private cloud is a cloud service that is exclusively offered to one organization. A virtual private cloud (VPC) is a private cloud within a public cloud; no one else shares the VPC with the VPC customer.

Features:

* IP Addressing: IP addresses enable resources in your VPC to communicate with each other and with resources over the internet.
* Routing:With this feature, you can route all incoming and outgoing traffic flowing to/from an internet gateway or virtual private gateway to a specific Amazon EC2 instance’s elastic network interface.
* Subnets:A subnet is a range of IP addresses in your VPC. A subnet must reside in a single Availability Zone. After you add subnets, you can deploy AWS resources in your VPC.
* Gateways and Endpoints:A gateway connects your VPC to another network. For example, use an internet gateway to connect your VPC to the internet. Use a VPC endpoint to connect to AWS services privately, without the use of an internet gateway or NAT device.
* Security groups: It acts as a virtual firewall for your EC2 instances to control incoming and outgoing traffic. Inbound rules control the incoming traffic to your instance, and outbound rules control the outgoing traffic from your instance. When you launch an instance, you can specify one or more security groups.
* For more information go the link available under References -> [VPC](https://docs.google.com/document/d/12fHUe3dPwudTJr1TCzO19YWe9rqJoXIv/edit#heading=h.2szc72q)

## **What is EC2?**

EC2 provides the broadest and deepest instance choice to match your workload’s needs. General purpose, compute optimized, memory optimized, storage optimized, and accelerated computing instance types are available that provide the optimal compute, memory, storage, and networking balance for your workloads. Processors from Intel, AMD, NVIDIA and AWS power these instance types and provide additional performance and cost optimizations.

Instance: It is a virtual server in the AWS Cloud. With Amazon EC2, you can set up and configure the operating system and applications that run on your instance.

Key pairs: It is a set of security credentials that you use to prove your identity when connecting to an Amazon EC2 instance.

Load balancers: A load balancer serves as the single point of contact for clients. The load balancer distributes incoming application traffic across multiple targets, such as EC2 instances, in multiple Availability Zones.

AMI: An Amazon Machine Image (AMI) is a supported and maintained image provided by AWS that provides the information required to launch an instance.

Features:

* Virtual computing environments, known as *instances*
* Static IPv4 addresses for dynamic cloud computing, known as *Elastic IP addresses*
* A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using *security groups*
* Multiple physical locations for your resources, such as instances and Amazon EBS volumes, known as *Regions* and *Availability Zones*
* Secure login information for your instances using *key pairs*
* *Various configurations of CPU, memory, storage, and networking capacity for your instances, known as instance types.*

For more information go the link available under References -> [EC2](https://docs.google.com/document/d/12fHUe3dPwudTJr1TCzO19YWe9rqJoXIv/edit#heading=h.4du1wux)

## **What is S3?**

S3 (Simple Storage Service) provides object storage, which is built for storing and recovering any amount of information or data from anywhere over the internet. It provides this storage through a web services interface. While designed for developers for easier web-scale computing, it provides 99.999999999 percent durability and 99.99 percent availability of objects. It can also store computer files up to 5 terabytes in size.

Features:

* Storage management:S3 bucket names, prefixes, object tags, and S3 Inventory, you have a range of ways to categorize and report on your data, and subsequently can configure other S3 features to take action.
* Access management:To protect your data in Amazon S3, by default, users only have access to the S3 resources they create.
* Bucket: A bucket is a container for objects stored in Amazon S3. You can store any number of objects in a bucket and can have up to 100 buckets in your account.
* Objects: Objects are the fundamental entities stored in Amazon S3. Objects consist of object data and metadata. The metadata is a set of name-value pairs that describe the object.
* Regions: You can choose the geographical AWS Region where Amazon S3 stores the buckets that you create. You might choose a Region to optimize latency, minimize costs, or address regulatory requirements.

For more information go the link available under References -> [S3](https://docs.google.com/document/d/12fHUe3dPwudTJr1TCzO19YWe9rqJoXIv/edit#heading=h.279ka65)

## References

### EC2: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>

### VPC: <https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html>

### IAM: <https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html>

### MSK: <https://docs.aws.amazon.com/msk/latest/developerguide/what-is-msk.html>

### S3: <https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html>

### Glue Schema Registry: <https://docs.aws.amazon.com/glue/latest/dg/what-is-glue.html>

JSON Schema References:

<https://docs.datafabric.hpe.com/70/Kafka/KafkaSchemaRegistry/kafka-schema-registry-json-schema.html>

[Learning Journal for Kafka](https://www.youtube.com/watch?v=gg-VwXSRnmg&list=PLkz1SCf5iB4enAR00Z46JwY9GGkaS2NON&index=1)

<https://github.com/awslabs/aws-glue-schema-registry>

ProtoBuf:

<https://aws.amazon.com/blogs/big-data/introducing-protocol-buffers-protobuf-schema-support-in-amazon-glue-schema-registry/#:~:text=Introduction%20to%20Protocol%20buffers&text=AWS%20Glue%20Schema%20Registry%20supports,composite%2C%20and%20enumeration%20data%20types>.

This page will help us in Mini Project:

<https://github.com/saubury/kafka-serialization>

—----------------------------------------------------------------------------------------------------------------------------

Under progress

—-------------------------------------------------------------------------------------------------

Connector configuration for Json Schema

Note:

(Create MSK connect connector step from above content but run the connector with following configuration)

To delete the topic:

bin/kafka-topics.sh --delete --bootstrap-server $bs --topic TopicName --command-config bin/client.properties