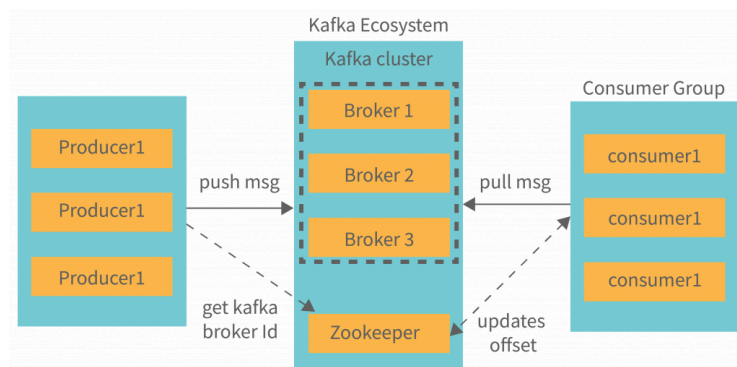
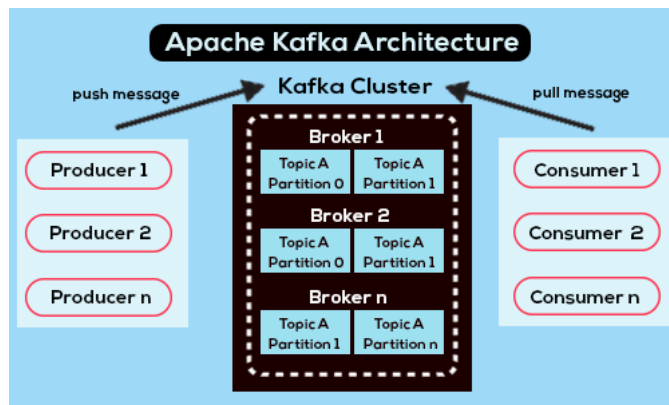


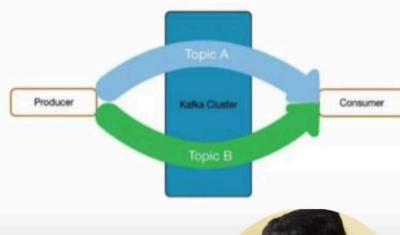
Conceptos



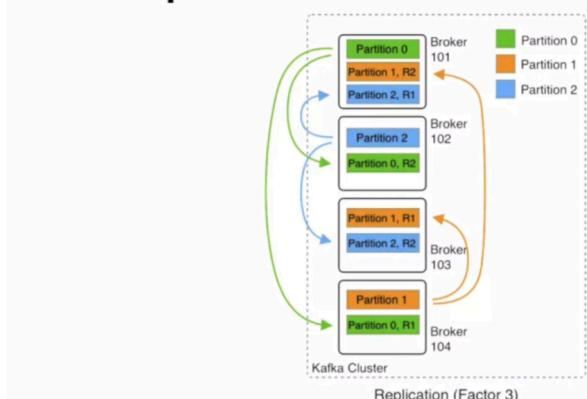
Zookeeper vigila a los broker (manager). Si un broker se cae, los demás toman esa responsabilidad de realizar las tareas necesarias.

Topics

- **Topics:** Streams of "related" Messages in Kafka
 - Is a Logical Representation
 - Categorizes Messages into Groups
- Developers define Topics
- Producer \longleftrightarrow Topic: N to N Relation
- Unlimited Number of Topics



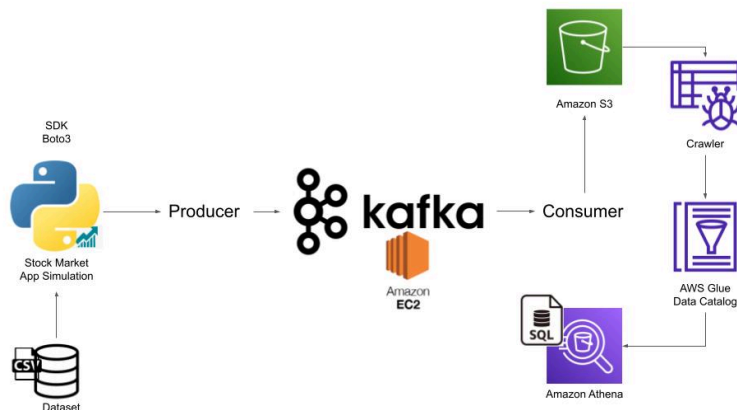
Broker Replication



Lesson Glossary

- **Broker (Kafka)** - A single member server of the Kafka cluster
- **Cluster (Kafka)** - A group of one or more Kafka Brokers working together to satisfy Kafka production and consumption
- **Node** - A single computing instance. May be physical, as in a server in a datacenter, or virtual, as an instance might be in AWS, GCP, or Azure.
- **Zookeeper** - Used by Kafka Brokers to determine which broker is the leader of a given partition and topic, as well as track cluster membership and configuration for Kafka
- **Data Partition (Kafka)** - Kafka topics consist of one or more partitions. A partition is a log which provides ordering guarantees for all of the data contained within it. Partitions are chosen by hashing key values.

Architecture



En principio, creamos una instancia EC2 con Amazon Linux y la siguiente configuración.

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Q Search our full catalog including 1000s of application and OS images

Quick Start

Amazon Linux
aws

macOS
Mac

Ubuntu
ubuntu

Windows
Microsoft

Red Hat
Red Hat

SUSE L
SUS

Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI
ami-03a6eaae9938c858c (64-bit (x86)) / ami-03f6c2c562b3df715 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

Description

Amazon Linux 2023 AMI 2023.2.20230920.1 x86_64 HVM kernel-6.1

Architecture

64-bit (x86)

AMI ID

ami-03a6eaae9938c858c

Verified provider

▼ Summary

Number of instances [info](#)

1

Software Image (AMI)

Amazon Linux 2023 AMI 2023.2.2...[read more](#)
ami-03a6eaae9938c858c

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

Cancel [Launch instance](#)

[Review commands](#)

▼ Instance type [info](#)

Instance type

t2.micro
Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Windows base pricing: 0.0162 USD per Hour
On-Demand SUSE base pricing: 0.0116 USD per Hour
On-Demand RHEL base pricing: 0.0716 USD per Hour
On-Demand Linux base pricing: 0.0116 USD per Hour

Free tier eligible

All generations

[Compare instance types](#)

Additional costs apply for AMIs with pre-installed software

Generamos una key para acceder a la máquina virtual.

▼ Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required

kafka-stock-market-key

Create new key pair

Usamos SSH para conectarnos a la máquina.

Inicialmente descargamos la última versión de Kafka para poder usarla en nuestra máquina virtual.

```
[ec2-user@ip-172-31-81-89 ~]$ wget https://downloads.apache.org/kafka/3.5.1/kafka_2.12-3.5.1.tgz
--2023-09-24 23:42:26-- https://downloads.apache.org/kafka/3.5.1/kafka_2.12-3.5.1.tgz
Resolving downloads.apache.org (downloads.apache.org)... 88.99.95.219, 135.181.214.104, 2a01:4f9:3a:2c57::2, ...
Connecting to downloads.apache.org (downloads.apache.org)[88.99.95.219]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 106956505 (102M) [application/x-gzip]
Saving to: 'kafka_2.12-3.5.1.tgz'

kafka_2.12-3.5.1.tgz      100%[=====] 102.00M  15.4MB/s   in 7.5s

2023-09-24 23:42:34 (13.5 MB/s) - 'kafka_2.12-3.5.1.tgz' saved [106956505/106956505]
```

Luego descomprimos con la ayuda del comando:

```
tar -xvf kafka_2.12-3.5.1.tgz
```

Ahora debemos instalar java dentro de la instancia.

```
sudo yum install java-1.8.0-openjdk / sudo yum install java-1.8.0-amazon-corretto-devel
java -version
```

En este momento ya podemos correr el zookeeper en una ventana (1) y el servidor de kafka (2) en otra.

(1) `bin/zookeeper-server-start.sh config/zookeeper.properties`

(2)

Incrementamos memoria para nuestro servidor de kafka.

```
export KAFKA_HEAP_OPTS="-Xmx256M -Xms128M"
```

Para poder conectarnos con nuestra máquina local debemos exponer la dirección IP pública de nuestro EC2.

```
##### Server Basics #####

# The id of the broker. This must be set to a unique integer for each broker.
broker.id=0

##### Socket Server Settings #####

# The address the socket server listens on. If not configured, the host name will be equal to the value of
# java.net.InetAddress.getCanonicalHostName(), with PLAINTEXT listener name, and port 9092.
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
#listeners=PLAINTEXT://:9092

# Listener name, hostname and port the broker will advertise to clients.
# If not set, it uses the value for "listeners".
advertised.listeners=PLAINTEXT://107.23.252.79:9092

# Maps listener names to security protocols, the default is for them to be the same. See the config documentation for more
#listener.security.protocol.map=PLAINTEXT:PLAINTEXT,SSL:SSL,SASL_PLAINTEXT:SASL_PLAINTEXT,SASL_SSL:SASL_SSL
```

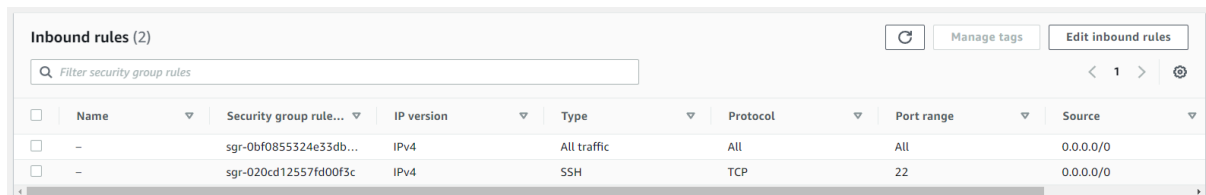
Utilizamos el siguiente comando para ingresar al archivo.

```
sudo nano config/server.properties
```

Luego corremos el servidor.

```
bin/kafka-server-start.sh config/server.properties
```

Agregamos una nueva regla al security group de nuestra EC2 para poder acceder desde nuestra máquina local.



	Name	Security group rule...	IP version	Type	Protocol	Port range	Source
<input type="checkbox"/>	-	sgr-0bf0855324e33db...	IPv4	All traffic	All	All	0.0.0.0/0
<input type="checkbox"/>	-	sgr-020cd12557fd00f3c	IPv4	SSH	TCP	22	0.0.0.0/0

En este punto, ya podemos crear el **topic** en una nueva ventana de la instancia, usando el siguiente comando.

```
bin/kafka-topics.sh --create --topic demo_testing2 --bootstrap-server 174.129.108.126:9092 --replication-factor 1 --partitions 1
```

```
[ec2-user@ip-172-31-81-89 kafka_2.12-3.5.1]$ bin/kafka-topics.sh --create --topic demo_testing2 --bootstrap-server 107.23.252.79:9092 --replication-factor 1 --partitions 1
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could collide. To avoid issues it is best to use either, but not both.
Created topic demo_testing2.
```

Luego creamos el **producer** con el siguiente comando.

```
bin/kafka-console-producer.sh --topic demo_testing2 --bootstrap-server 174.129.108.126:9092
```

En una ventana nueva creamos el **consumer**.

```
bin/kafka-console-consumer.sh --topic demo_testing2 --bootstrap-server 174.129.108.126:9092
```

Producer

```
[ec2-user@ip-172-31-81-89 kafka_2.12-3.5.1]$ bin/kafka-console-producer.sh --topic demo_testing2 --bootstrap-server 107.23.252.79:9092
hello world
>fvf
>fbfn
>hnhn
>jjjj
```

Consumer

```
[ec2-user@ip-172-31-81-89 kafka_2.12-3.5.1]$ bin/kafka-console-consumer.sh --topic demo_testing2 --bootstrap-server 107.23.252.79:9092
hello world
fvf
fbfn
hnhn
jjjj
```

Ahora, usamos nuestro notebook en Python para conectarnos al **producer** y poder enviar mensajes al **consumer**.

```
{ "hello": "world" }

producer = KafkaProducer(bootstrap_servers = ['54.208.198.32:9092'],
                          value_serializer = lambda x: dumps(x).encode('utf-8'))

producer.send('demo_testing2', value={'hello': 'world'})

<kafka.producer.future.FutureRecordMetadata at 0x19d08289810>
```

Creamos la conexión al consumer en Python y desde ahí podemos ver lo que llega desde el producer.

```
consumer = KafkaConsumer('demo_testing2',
                          bootstrap_servers = ['54.208.198.32:9092'],
                          value_deserializer = lambda x: loads(x.decode('utf-8')))

for c in consumer:
    print(c.value)

{'hello': 'world'}
{'name': 'Jhon'}
```

Creamos un bucket en S3.

The screenshot shows the AWS S3 console interface. At the top, there's a section titled 'Buckets (1) Info' with a description and a 'Learn more' link. Below this are buttons for 'Copy ARN', 'Empty', 'Delete', and 'Create bucket'. A search bar is present with the text 'Find buckets by name'. Below the search bar is a table with columns: Name, AWS Region, Access, and Creation date. The table contains one entry: 'kafka-stock-market-bucket-1' in the 'US East (N. Virginia) us-east-1' region, with 'Bucket and objects not public' access and a creation date of 'September 25, 2023, 10:59:15 (UTC-05:00)'.

Debemos crear un user para conectarnos al S3.

The screenshot shows the AWS IAM console 'Create user' wizard, specifically the 'Review and create' step. The left sidebar shows the progress: 'Step 1: Specify user details', 'Step 2: Set permissions', and 'Step 3: Review and create'. The main content area is titled 'Review and create' and includes a sub-header 'User details' with fields for 'User name' (jhon-admin-account), 'Console password type' (None), and 'Require password reset' (No). Below this is a 'Permissions summary' section showing a table with columns 'Name', 'Type', and 'Used as'. The table lists 'AdministratorAccess' as an 'AWS managed - job function' with 'Permissions policy' used. At the bottom, there's a 'Tags - optional' section with a note that no tags are associated with the resource and a button to 'Add new tag'. At the very bottom right are 'Cancel', 'Previous', and 'Create user' buttons.

Creamos una key dentro del user para poder acceder.

Retrieve access keys [Info](#)

Access key
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
AKIASVHOO4CD6YBBRQ6K	***** Show

Realizamos esta configuración en nuestra máquina.

```
PS C:\Program Files\Amazon\AWSCLIV2> aws configure
AWS Access Key ID [*****RQ6K]:
AWS Secret Access Key [*****x/hX]:
Default region name [None]: us-east-1
Default output format [us-east-1]: None
```

En este punto, ya podemos enviar datos al S3. Lo hacemos de la siguiente forma.

KafkaConsumer - Apuntamos al S3 enviando lo que recibimos en el consumer.

```
for count, i in enumerate(consumer):
    with s3.open('s3://kafka-stock-market-bucket-1/stock_market_{}.json'.format(count), 'w') as file:
        json.dump(i.value, file)
```

KafkaProducer - Usamos el producer para tomar registros aleatorios del dataset y enviarlos al consumer.

```
while True:
    dick_stock = df.sample(1).to_dict(orient='records')[0]
    producer.send('demo_testing2', value=dick_stock)
```

En el bucket quedan cargados los archivos json.

[Amazon S3](#) > [Buckets](#) > kafka-stock-market-bucket-1

kafka-stock-market-bucket-1 [Info](#)

[Objects](#) | [Properties](#) | [Permissions](#) | [Metrics](#) | [Management](#) | [Access Points](#)

Objects (20)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Copy S3 URI Copy URL Download Open Delete **Actions** Create folder Upload

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	stock_market_0.json	json	September 25, 2023, 12:21:49 (UTC-05:00)	200.0 B	Standard
<input type="checkbox"/>	stock_market_1.json	json	September 25, 2023, 12:21:49 (UTC-05:00)	198.0 B	Standard
<input type="checkbox"/>	stock_market_10.json	json	September 25, 2023, 12:21:51 (UTC-05:00)	197.0 B	Standard
<input type="checkbox"/>	stock_market_11.json	json	September 25, 2023, 12:21:51 (UTC-05:00)	182.0 B	Standard
<input type="checkbox"/>	stock_market_12.json	json	September 25, 2023, 12:21:52 (UTC-05:00)	188.0 B	Standard
<input type="checkbox"/>	stock_market_13.json	json	September 25, 2023, 12:21:52 (UTC-05:00)	176.0 B	Standard
<input type="checkbox"/>	stock_market_14.json	json	September 25, 2023, 12:21:52 (UTC-05:00)	190.0 B	Standard
<input type="checkbox"/>	stock_market_15.json	json	September 25, 2023, 12:21:52 (UTC-05:00)	180.0 B	Standard

Nos vamos al servicio AWS Glue y creamos un nuevo Crawler.

AWS Glue > Crawlers > Add crawler

Step 1
Set crawler properties

Step 2
Choose data sources and classifiers

Step 3
Configure security settings

Step 4
Set output and scheduling

Step 5
Review and create

Choose data sources and classifiers

Data source configuration

Is your data already mapped to Glue tables?

☒ Not yet
Select one or more data sources to be crawled.

☐ Yes
Select existing tables from your Glue Data Catalog.

Data sources (1) [Info](#)

The list of data sources to be scanned by the crawler.

Type	Data source	Parameters
<input type="radio"/> S3	s3://kafka-stock-market-bucket-1/	Recrawl all

[Edit](#) [Remove](#) [Add a data source](#)

► **Custom classifiers - optional**

A classifier checks whether a given file is in a format the crawler can handle. If it is, the classifier creates a schema in the form of a StructType object that matches that data format.

[Cancel](#) [Previous](#) [Next](#)

Add data source

Data source
Choose the source of data to be crawled.

Network connection - optional
Optionally include a Network connection to use with this S3 target. Note that each crawler is limited to one Network connection so any other S3 targets will also use the same connection (or none, if left blank).

[Clear selection](#) [Add new connection](#)

Location of S3 data

☒ In this account
☐ In a different account

S3 path
Browse for or enter an existing S3 path.

[View](#) [Browse S3](#)

All folders and files contained in the S3 path are crawled. For example, type s3://MyBucket/MyFolder/ to crawl all objects in MyFolder within MyBucket.

Subsequent crawler runs
This field is a global field that affects all S3 data sources.

☒ Crawl all sub-folders
Crawl all folders again with every subsequent crawl.

☐ Crawl new sub-folders only
Only Amazon S3 folders that were added since the last crawl will be crawled. If the schemas are compatible, new partitions will be added to existing tables.

☐ Crawl based on events
Rely on Amazon S3 events to control what folders to crawl.

☐ Sample only a subset of files

☐ Exclude files matching pattern

[Cancel](#) [Add an S3 data source](#)

Debemos crear un nuevo rol para acceder desde Glue hacia los demás servicios sin problema.

IAM > Roles > Create role

Step 1
Select trusted entity

Step 2
Add permissions

Step 3
Name, review, and create

Select trusted entity

Trusted entity type

☒ **AWS service**
Allow AWS services like EC2, Lambda, or others to perform actions in this account.

☐ **AWS account**
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

☐ **Web identity**
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

☐ **SAML 2.0 federation**
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

☐ **Custom trust policy**
Create a custom trust policy to enable others to perform actions in this account.

Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

Choose a use case for the specified service.

Use case
☒ **Glue**
Allows Glue to call AWS services on your behalf.

[Cancel](#) [Next](#)

Le asignamos permisos de administrador.

Permissions policy summary	
Policy name 🔗	Type
AdministratorAccess	AWS managed - job function

Configure security settings

IAM role [Info](#)

Existing IAM role

▼ [🔄](#) [View 🔗](#)

[Create new IAM role](#) [Update chosen IAM role](#)

Only IAM roles created by the AWS Glue console and have the prefix "AWSGlueServiceRole-" can be updated.

[AWS Glue](#) > [Databases](#) > Add database

Create a database

Create a database in the AWS Glue Data Catalog.

Database details

Name

Database name is required, in lowercase characters, and no longer than 255 characters.

Location - *optional*

Set the URI location for use by clients of the Data Catalog.

Description - *optional*

Descriptions can be up to 2048 characters long.

[Cancel](#) [Create database](#)

Set output and scheduling

Output configuration [Info](#)

Target database

▼ [🔄](#)

[Clear selection](#) [Add database 🔗](#)

Table name prefix - *optional*

Finalmente creamos el crawler.

Ahora ingresamos a Amazon Athena y seleccionamos el Query editor. Elegimos la base de datos creada anteriormente y realizamos un query de prueba.

The screenshot shows the Amazon Athena Query Editor interface. On the left, the 'Data' sidebar is visible with 'Data source' set to 'AwsDataCatalog' and 'Database' set to 'stock_market_kafka'. Below this, 'Tables and views' are listed, including 'kafka_stock_market_bucket_1'. The main area displays 'Query 2' with the SQL statement: `SELECT * FROM "stock_market_kafka"."kafka_stock_market_bucket_1" limit 10;`. The query status is 'Completed' with a 'Run time' of 541 ms and 'Data scanned' of 2.07 KB. The 'Results (10)' section shows a table with 10 rows of stock market data.

#	index	date	open	high	low	close	adj close	volume	closeusd
1	IXIC	2018-12-21	6573.490234	6586.680176	6304.629883	6332.990234	6332.990234	4.53412E9	6332.990234
2	399001.SZ	2021-01-05	14760.25	15149.09961	14716.79004	15147.57031	15147.57031	2173300.0	2423.6112496
3	NYA	1966-02-21	528.26001	528.26001	528.26001	528.26001	528.26001	0.0	528.26001
4	399001.SZ	2015-05-29	15920.33008	16386.06055	15299.65039	16100.4502	16100.26953	2319700.0	2576.072032

Por último, corremos nuevamente el producer y consumer con tiempo de espera de 5 segundos y validamos en Athena la cantidad de archivos json que hay en nuestro S3, con ayuda de un query.

The screenshot shows the Amazon Athena Query Editor interface. The 'Data' sidebar is the same as in the previous image. The main area displays 'Query 2' with the SQL statement: `SELECT COUNT(*) FROM "stock_market_kafka"."kafka_stock_market_bucket_1";`. The query status is 'Completed' with a 'Run time' of 397 ms and 'Data scanned' of 1.89 KB. The 'Results (1)' section shows a single row with the count of files.

#	_col0
1	10

Este valor va cambiando con el tiempo, por lo que se cumple el objetivo del laboratorio de manejar datos en tiempo real.