A Internship Project Report on

Stock Market Real-Time Data Analysis Using Kafka

BY

VINAY WASUDEO YERNE

(PRN:1062222749)

MCA – II, SEM-IV

2022-2024

Dr. Vishwanath Karad MIT
World Peace University, Pune
Pune - 411038

In Partial Fulfillment of the Degree of
Master in Computer Application (M. C. A.)

Under the Guidance of

Name of Guide

Dr. Monika Gadre
Academic Year 2023-24

CERTIFICATE

This is to certify that Mr. / Ms. VINAY WASUDEO YERNE has successfully completed his/her project work entitled **“Stock Market Real-Time data Analysis Using Kafka”** in partial fulfillment of MCA – II Semester-IV program for the year A.Y. 2023-2024. He /She have worked under our guidance and direction.

Dr Dinesh Banswal
(Program Director, SOB)

Dr. Monica Gadre
(Project Guide)

Dr. Rajshree Kadam
(Associate Dean, Academics, SOB)

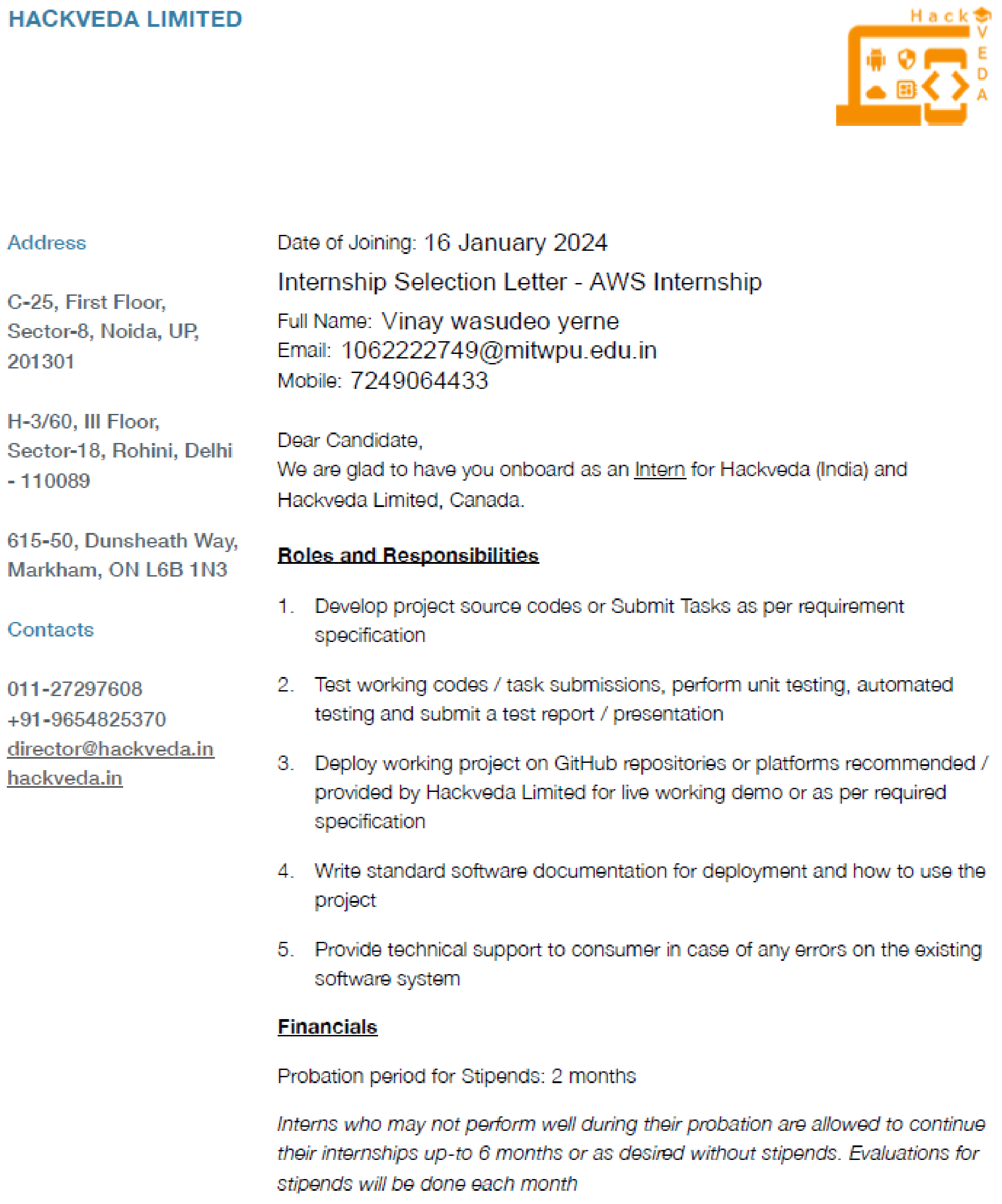
Dr. Deependra Sharma
(Dean-School of Management)

.

Examiner 1 **Examiner 2**

Date :
Place :

Company Offer Letter



HACKVEDA LIMITED

Address

C-25, First Floor,
Sector-8, Noida, UP,
201301

H-3/60, III Floor,
Sector-18, Rohini, Delhi
- 110089

615-50, Dunsheath Way,
Markham, ON L6B 1N3

Contacts

011-27297608
+91-9654825370
director@hackveda.in
hackveda.in

Date of Joining: 16 January 2024

Internship Selection Letter - AWS Internship

Full Name: Vinay wasudeo yerne
Email: 1062222749@mitwpu.edu.in
Mobile: 7249064433

Dear Candidate,
We are glad to have you onboard as an Intern for Hackveda (India) and Hackveda Limited, Canada.

Roles and Responsibilities

1. Develop project source codes or Submit Tasks as per requirement specification
2. Test working codes / task submissions, perform unit testing, automated testing and submit a test report / presentation
3. Deploy working project on GitHub repositories or platforms recommended / provided by Hackveda Limited for live working demo or as per required specification
4. Write standard software documentation for deployment and how to use the project
5. Provide technical support to consumer in case of any errors on the existing software system

Financials

Probation period for Stipends: 2 months

Interns who may not perform well during their probation are allowed to continue their internships up-to 6 months or as desired without stipends. Evaluations for stipends will be done each month

Acknowledgment

I am grateful to **HACKVEDA** and my internship supervisor MS. NEHA , for providing me with the opportunity to completemy MCA internship at their organization. Their support and guidance helped me to understand the business world and gain valuable experience in my field.

From the moment I started my internship, **Mr. Devanshu Shukla sir**, provided me with clear direction and expectations, and was always available to answer my questions and provide valuable feedback. Their expertise and guidance helped me to understand the inner workings of the company and the industry, and allowed me to make the most of my internship.

Thankyou .

Student Name:-

Vinay Wasudeo Yerne

(TASK BASED)

Chapter	Page number
CHAPTER 1 : INTRODUCTION	7-9
1.1 Existing System	10-11
1.2 Need for System	12-13
1.3 Operating Environment Hardware and Software	14
CHAPTER 2 : PROPOSED SYSTEM	
2.1 Proposed System (Introduction of system)	15
2.2 Module specifications (Scope)	16
2.3 Objectives of System	17
CHAPTER 3 : ANALYSIS & DESIGN	
3.1 Use Case Diagrams	18
3.2 Activity Diagram	19
3.6 Table specifications (Database design)	20
CHAPTER 4 : USER MANUAL	
4.1 User Interface Screens (Input)	21-26
4.2 Output Screens with data	27-34
4.3 Data Reports	35
4.4 Test Procedures and cases	36-38
4.5 Sample program code	37
4.6 Limitation and bibliography	39

CHAPTER 1:- INTRODUCTION

Cloud computing is an on-demand delivery of compute power, database, storage, applications, and other IT resources via a cloud services platform via the internet with pay-as-you-go pricing. It offers rapidaccess to flexible and low-cost IT resources, allowing users to provision theright type and size of computing resources to power their ideas or operate their IT department. Cloud computing provides a simple way to access servers, storage, databases, and a broad set of application services over the internet. A cloud services platform like Amazon Web Services owns and maintains the network-connected hardware required for these application services, while user provision and use resources via a web application.

AWS service that allows developers to create conversational interfaces for applications using voice and text. It uses the same conversational engine as Amazon Alexa, allowing developers to create sophisticated, natural language chatbots in new and existing applications. It offers deep functionality and flexibility in natural language understanding and automatic speech recognition, enabling engaging user experiences and product categories.

Amazon Web Services (AWS) is the world’s most comprehensive and broadly adopted cloud, offering over 200 fully featured services from data centers globally. Millions of customers—including the fastest-growing startups, largest enterprises, and leading government agencies—are using AWS to lower costs, become more agile, and innovate faster.

Setting up AWS Cloud for stock market data management and analysis involves creating a scalable and secure infrastructure on the Amazon Web Services (AWS) platform to store, process, and analyze vast amounts of financial data efficiently. This includes designing and deploying databases, implementing data ingestion pipelines, developing analytical models, and creating user-friendly interfaces for accessing insights and reports. Leveraging AWS services such as

Amazon S3 for storage, Amazon Redshift for data warehousing, AWS Glue for ETL (Extract, Transform, Load), and Amazon Sage Maker for machine learning, the system enables seamless data management, real-time analytics, and predictive modeling to support informed decision-making in the dynamic stock market environment.

Setting up AWS Cloud for stock market data management and analysis involves deploying a robust infrastructure capable of handling the storage, processing, and analysis of vast amounts of financial data. This infrastructure enables organizations to store historical and real-time market data securely, perform complex data analytics, and derive actionable insights to drive business strategies.

In this document, we will explore the process of setting up AWS Cloud for stock market data management and analysis, including the selection and configuration of AWS services, the design and implementation of data processing pipelines, and the development of analytical models to extract valuable insights from stock market data. Through this comprehensive approach, organizations can leverage the power of AWS Cloud to enhance their capabilities in managing and analyzing stock market data, ultimately driving better financial outcomes and staying ahead in today's competitive market environment.

AWS KAFKA:-

Amazon Managed Streaming for Apache Kafka (Amazon MSK) is a fully managed service that makes it easy for you to build and run applications that use Apache Kafka to process streaming data. Apache Kafka is an open-source platform for building real-time streaming data pipelines and applications.
Here are some key features and benefits of Amazon MSK:
1. Managed Service: Amazon MSK simplifies the setup, management, and monitoring of Apache Kafka clusters, allowing you to focus on building applications rather than managing infrastructure.
2. High Availability: Amazon MSK provides high availability by automatically replicating data across multiple availability zones within a region, ensuring durability and fault tolerance.
3. Scalability: You can easily scale your Apache Kafka clusters up or down based on your application's needs, without worrying about provisioning or managing additional resources.

4. Integration with AWS Services: Amazon MSK integrates seamlessly with other AWS services such as Amazon S3, Amazon CloudWatch, and AWS Identity and Access Management (IAM), enabling you to build end-to-end streaming data solutions.

5. Security: Amazon MSK offers built-in security features such as encryption at rest and in transit, authentication using AWS IAM, and network isolation through Amazon VPC, ensuring data confidentiality and integrity.

6. Monitoring and Management: Amazon MSK provides metrics and logs through Amazon CloudWatch, allowing you to monitor the performance and health of your Apache Kafka clusters in real-time.

1.1 Existing System :-

This section outlines the current system or processes that are in place for managing and analyzing stock market data. It may include details about any software tools, databases, or manual procedures currently used ionized software tools like Bloomberg Terminal or Reuters Eikon for real-time data and analysis. These tools are complemented by databases housing historical and real-time market data.

Tools like AWS Kafka or Terminal provide real-time and historical market data. Databases store vast data including prices, financials. Manual tasks involve validation and qualitative analysis. Data from exchanges, news, filings are integrated via APIs. Analysis includes fundamental, technical, and sentiment analysis.

1.1.1 Online Research: -

- Software Tools: Stock market data analysis often relies on specialized software tools such as Bloomberg Terminal, Reuters Eikon, or proprietary trading platforms. These tools provide real-time market data, historical data, charting, technical analysis tools, and sometimes algorithmic trading capabilities.

- Databases: Large volumes of stock market data are stored in databases for analysis. These databases may include historical price data, company financials, economic indicators, and other relevant information. Common database technologies used in this context include SQL databases like MySQL or PostgreSQL, as well as NoSQL databases like MongoDB for handling unstructured data.

- Manual Procedures: Despite the advancements in technology, some aspects of stock market analysis still require manual intervention. This could include tasks such as data validation, qualitative analysis of news or events impacting the market, or decision-making based on non-quantitative factors.

- Data Sources: The system likely integrates data from various sources such as stock exchanges, financial news outlets, regulatory filings, and third-party data providers. APIs (Application Programming Interfaces) are often used to automate data retrieval and ensure data accuracy.

- Analysis Techniques: Different analysis techniques are employed, including fundamental analysis (evaluating a company's financial health and prospects), technical analysis (examining price movements and trading volumes), and sentiment analysis (assessing market sentiment based on news articles, social media, etc.).

1.2 Need for System:-

- The current system for stock market data management and analysis may exhibit several shortcomings necessitating a new solution. Inefficiencies in data processing and analysis could lead to delays in decision-making, hindering traders and investors from capitalizing on timely market opportunities.

- Limitations in scalability may impede the system's ability to handle the ever-growing volumes of market data effectively, potentially leading to performance bottlenecks and decreased responsiveness. Furthermore, the existing system may lack integration with emerging data sources like social media, depriving users of valuable insights into market sentiment and trends.

- Additionally, there may be a need for additional features such as advanced analytics capabilities including machine learning algorithms for predictive modeling and risk assessment, which can provide a competitive edge in today's dynamic markets. Overall, the demand for a new system arises from the desire to overcome these inefficiencies, limitations, and to incorporate advanced features essential for robust and agile stock market data management and analysis.

1.2.1 Limitations of Conventional Methods:-

- Latency Issues: Despite Kafka's real-time processing capabilities, conventional methods may struggle to achieve low-latency data analysis due to inefficient data processing pipelines or resource constraints. This latency can impede timely decision-making in fast-paced market environments.

- Scalability Challenges: Conventional methods may face difficulties in effectively scaling data processing pipelines to handle the high volume of real-time data generated by stock markets. Scaling issues can lead to performance bottlenecks and decreased system responsiveness, especially during periods of high market activity.
- Complex Data Transformation: Efficiently transforming raw stock market data into a format suitable for analysis can be challenging for conventional methods. This complexity arises from the need for data normalization, aggregation, and enrichment processes, which may not be easily handled using traditional techniques.
- Limited Analytical Capabilities: Conventional methods may lack advanced analytical capabilities compared to modern data analytics platforms. This limitation restricts the types of analysis that can be performed on stock market data, potentially limiting insights and decision-making potential.
- Dependency on Legacy Systems: Conventional methods often rely on legacy systems and technologies that may not be optimized for real-time data analysis using Kafka. This reliance introduces compatibility issues and impedes the seamless integration of Kafka into existing data infrastructure.
- Risk of Data Loss: Inadequate fault tolerance mechanisms in conventional setups may pose a risk of data loss, particularly during system failures or network disruptions. This risk undermines data integrity and reliability, crucial factors in stock market data analysis.

1.3 Operating Environment Hardware and Software

•The proposed system requires hardware infrastructure capable of handling real-time data processing and analysis. This includes high-performance servers with sufficient CPU, memory, and storage resources to accommodate the volume of data generated by stock market feeds. Additionally, a distributed computing environment may be necessary for scalability and fault tolerance.

•For the operating system, compatibility with both Linux and Windows environments is recommended to cater to diverse deployment scenarios. Linux distributions like Ubuntu, CentOS, or Red Hat Enterprise Linux are popular choices for server deployments due to their stability and performance benefits.

•Regarding databases, support for both SQL and NoSQL databases is essential to accommodate various data storage and querying requirements. SQL databases such as PostgreSQL or MySQL are suitable for structured data storage, while NoSQL databases like Apache Cassandra or MongoDB can handle unstructured or semi-structured data efficiently.

•Key software dependencies include Apache Kafka for real-time data ingestion and stream processing, along with stream processing frameworks like Apache Flink or Apache Spark Streaming for advanced analytics. Additionally, analytics tools such as Apache Hadoop or Apache Hive may be utilized for batch processing and data warehousing.

•Containerization technologies like Docker and orchestration platforms like Kubernetes can facilitate deployment, management, and scalability of the system components in a distributed environment.

CHAPTER 2 -PROPOSED SYSTEM:-

2.1 Proposed System (Introduction of system):-

- The proposed system for stock market data management and analysis is a comprehensive platform designed to facilitate real-time processing, analysis, and decision-making in financial markets. It offers advanced features such as:
- Real-time Data Ingestion: Seamless integration with data sources to ingest real-time stock market data feeds from exchanges, news outlets, and other relevant sources.
- Scalable Architecture: Built on scalable infrastructure to handle high volumes of data efficiently, ensuring responsiveness during peak market activity.
- Advanced Analytics: Utilizes cutting-edge analytics techniques including machine learning and natural language processing for predictive modeling, sentiment analysis, and risk assessment.
- Flexible Data Storage: Supports both SQL and NoSQL databases for flexible data storage, enabling structured and unstructured data analysis.
- Stream Processing: Leverages stream processing frameworks like Apache Kafka and Apache Flink for real-time data processing, enabling instant insights and decision-making.
- Interactive Visualization: Provides interactive visualization tools for intuitive exploration and analysis of stock market trends, patterns, and anomalies.

- Robust Security: Implements robust security measures to protect sensitive financial data and ensure compliance with regulatory requirements.

2.2 Module specifications (Scope):-

- Data Ingestion: Module responsible for ingesting real-time stock market data from multiple sources such as exchanges, news feeds, and social media platforms.
- Data Storage: Module for storing and managing stock market data efficiently, supporting both SQL and NoSQL databases for structured and unstructured data.
- Data Processing: Module for processing raw data streams using stream processing frameworks like Apache Kafka and Apache Flink to enable real-time analytics.
- Analytics: Module incorporating advanced analytics techniques including machine learning, natural language processing, sentiment analysis, and statistical modeling for extracting insights from stock market data.
- Visualization: Module for interactive visualization of analyzed data, providing intuitive tools for exploring trends, patterns, and anomalies in stock market data.
- Alerting and Monitoring: Module for setting up alerts and monitoring systems to notify users of significant market events or deviations from predefined criteria.
- Security and Compliance: Module ensuring robust security measures to protect sensitive financial data and ensure compliance with regulatory requirements such as GDPR and FINRA.
- Integration and APIs: Module facilitating integration with external systems and providing APIs for seamless data exchange with third-party applications and services.

2.3 Objectives of System:-

The objectives of the proposed system are:-

- Real-time Insights: Enable users to access real-time stock market data and gain actionable insights for informed decision-making.

- Scalability: Provide a scalable architecture capable of handling high volumes of data to support growing user demands.

- Advanced Analytics: Implement advanced analytics techniques such as machine learning and sentiment analysis to extract valuable insights from data.

- Efficiency: Streamline data processing and analysis workflows to improve efficiency and reduce latency in decision-making processes.

- Visualization: Offer intuitive visualization tools for interactive exploration of stock market trends, patterns, and anomalies.

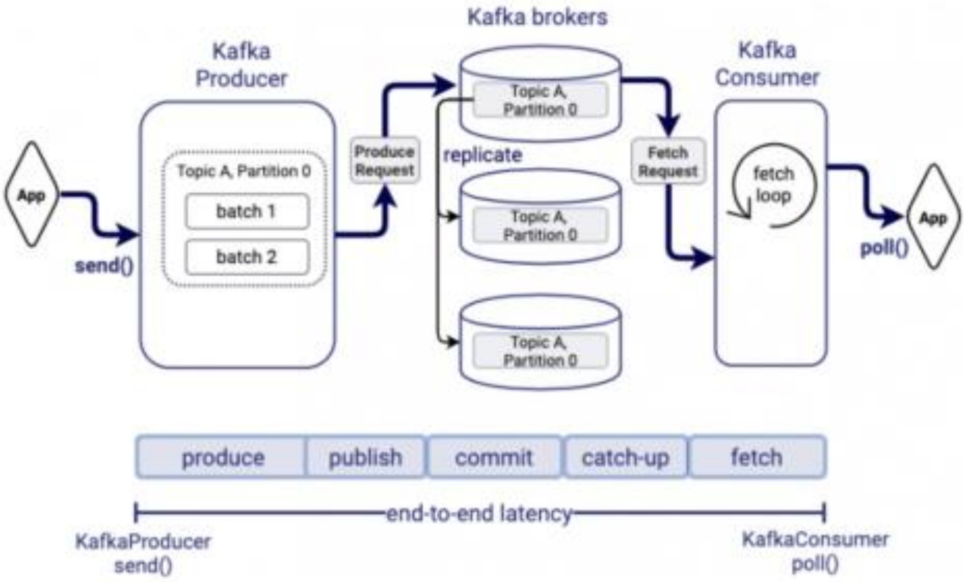
- Security and Compliance: Ensure robust security measures and compliance with regulatory requirements to protect sensitive financial data.

- User-Friendly Interface: Design a user-friendly interface to enhance user experience and facilitate ease of use for traders, analysts, and investors.

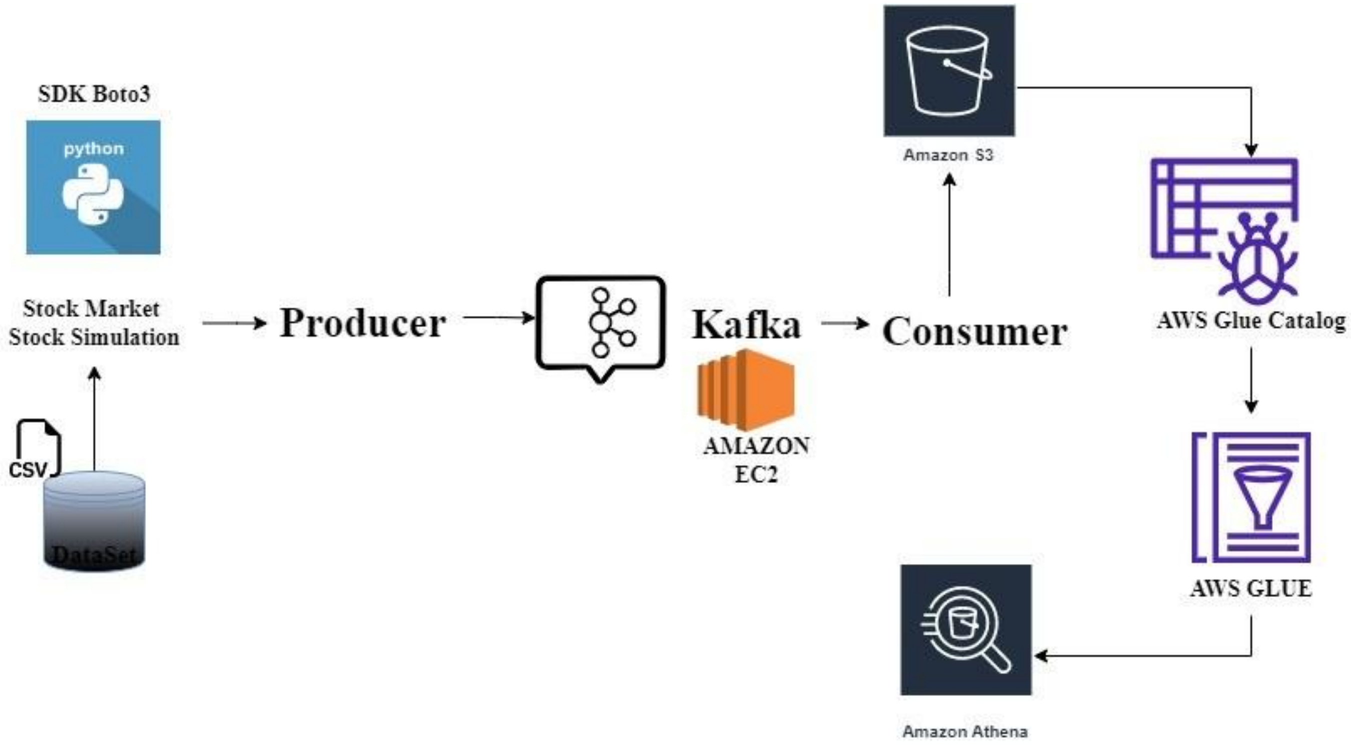
- Integration: Facilitate integration with external systems and APIs to enable seamless data exchange and interoperability with other applications.

CHAPTER 3 : ANALYSIS & DESIGN

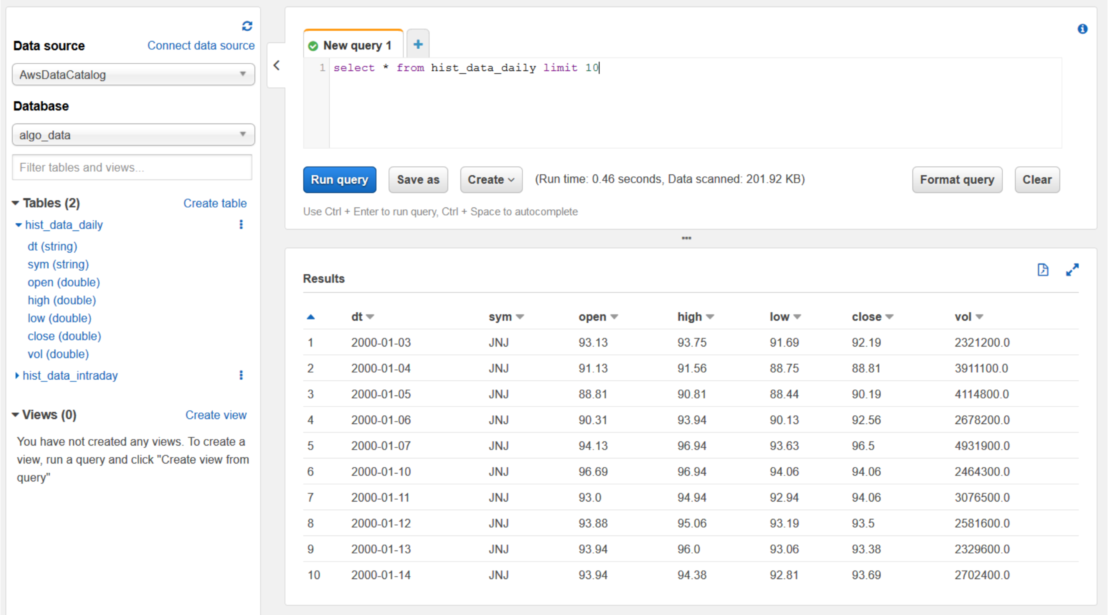
3.1 Process of Kafka Diagrams:-



3.2 Stock Market Data analysis Using Kafka Architecture.



3.3 Database :-



Data source: AwsDataCatalog

Database: algo_data

Tables (2)

- hist_data_daily
  - dt (string)
  - sym (string)
  - open (double)
  - high (double)
  - low (double)
  - close (double)
  - vol (double)
- hist_data_intraday

Views (0)

You have not created any views. To create a view, run a query and click "Create view from query"

New query 1

```
select * from hist_data_daily limit 10
```

(Run time: 0.46 seconds, Data scanned: 201.92 KB)

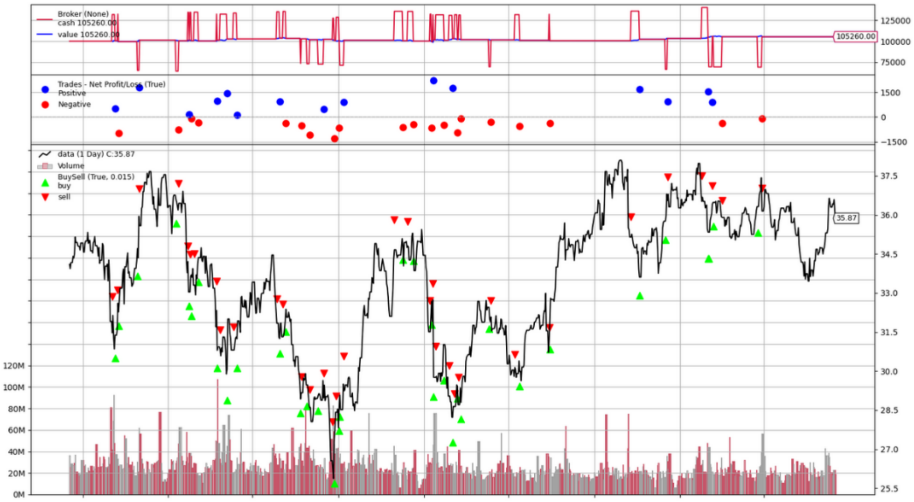
Use Ctrl + Enter to run query, Ctrl + Space to autocomplete

Results

	dt	sym	open	high	low	close	vol
1	2000-01-03	JNJ	93.13	93.75	91.69	92.19	2321200.0
2	2000-01-04	JNJ	91.13	91.56	88.75	88.81	3911100.0
3	2000-01-05	JNJ	88.81	90.81	88.44	90.19	4114800.0
4	2000-01-06	JNJ	90.31	93.94	90.13	92.56	2678200.0
5	2000-01-07	JNJ	94.13	96.94	93.63	96.5	4931900.0
6	2000-01-10	JNJ	96.69	96.94	94.06	94.06	2464300.0
7	2000-01-11	JNJ	93.0	94.94	92.94	94.06	3076500.0
8	2000-01-12	JNJ	93.88	95.06	93.19	93.5	2581600.0
9	2000-01-13	JNJ	93.94	96.0	93.06	93.38	2329600.0
10	2000-01-14	JNJ	93.94	94.38	92.81	93.69	2702400.0

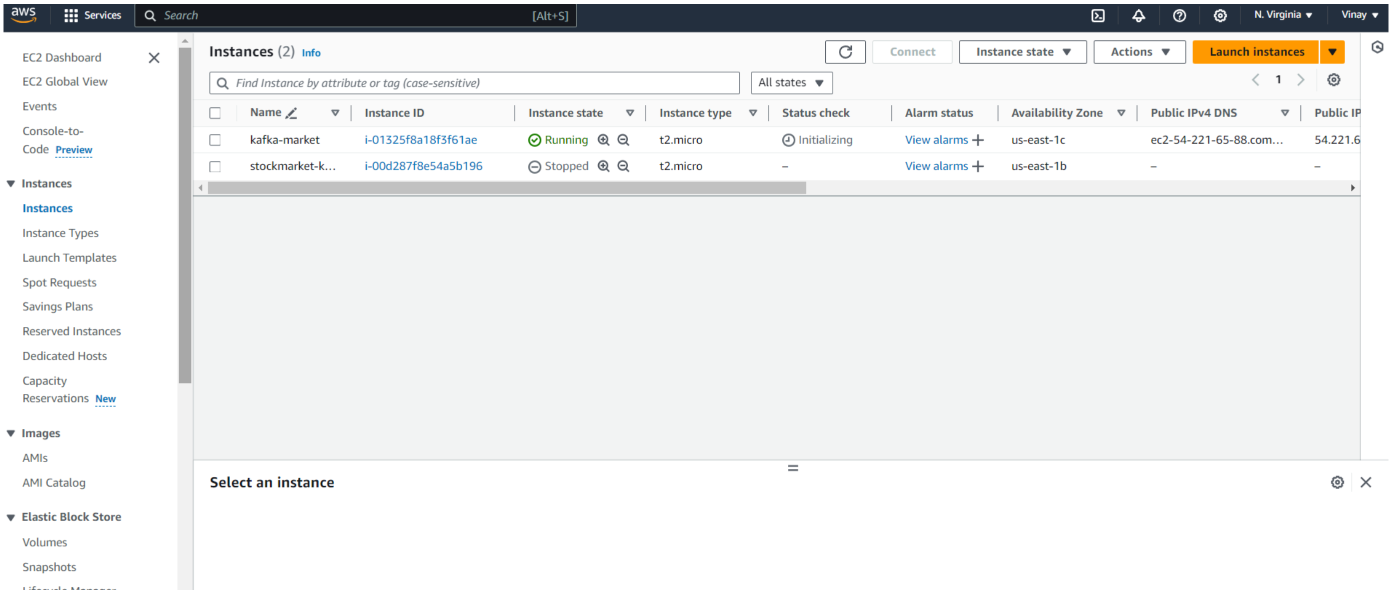
```
In [8]: from IPython.display import Image
        Image(filename='local/'+algo_name+'/model/chart.png')
```

Out[8]:



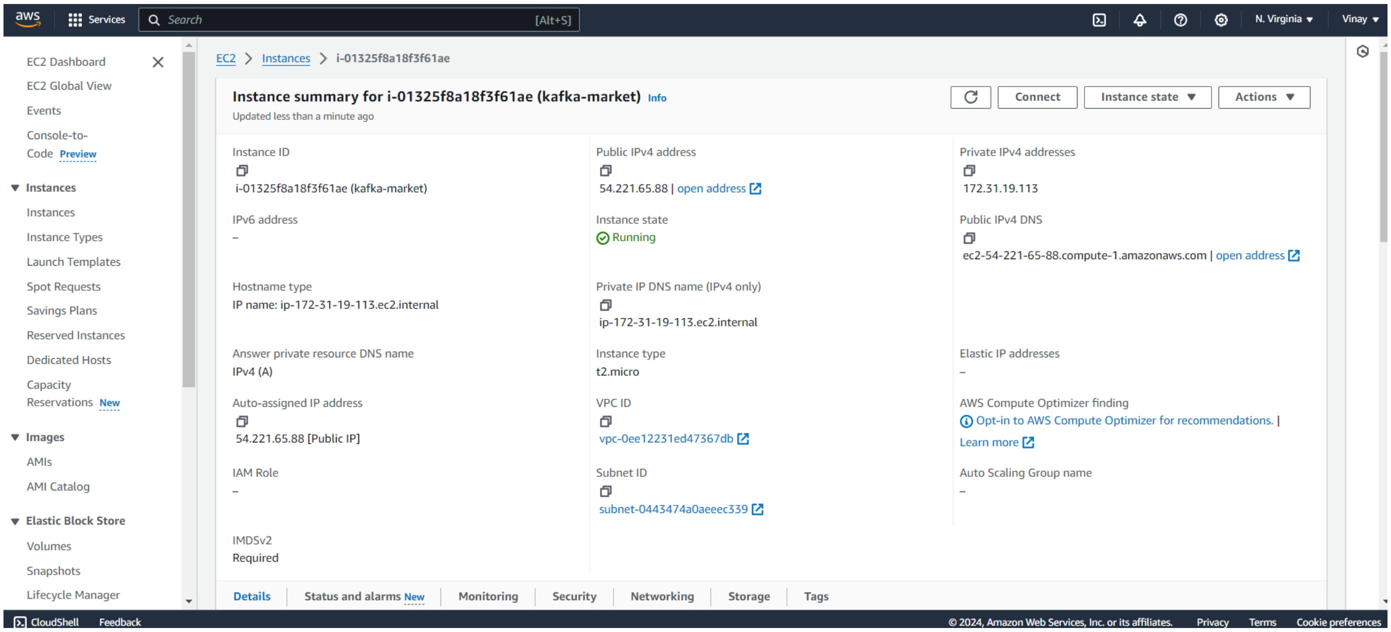
CHAPTER 4:- USER MANUAL

4.1 User Interface Screens (Input):-



Instances (2) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IP
kafka-market	i-01325f8a18f3f61ae	Running	t2.micro	Initializing	View alarms +	us-east-1c	ec2-54-221-65-88.com...	54.221.6
stockmarket-k...	i-00d287f8e54a5b196	Stopped	t2.micro	–	View alarms +	us-east-1b	–	–



Instance summary for i-01325f8a18f3f61ae (kafka-market) Info

Updated less than a minute ago

Instance ID
i-01325f8a18f3f61ae (kafka-market)

Public IPv4 address
54.221.65.88 | open address

Private IPv4 addresses
172.31.19.113

IPv6 address
–

Instance state
Running

Public IPv4 DNS
ec2-54-221-65-88.compute-1.amazonaws.com | open address

Hostname type
IP name: ip-172-31-19-113.ec2.internal

Private IP DNS name (IPv4 only)
ip-172-31-19-113.ec2.internal

Answer private resource DNS name
IPv4 (A)

Instance type
t2.micro

Elastic IP addresses
–

Auto-assigned IP address
54.221.65.88 [Public IP]

VPC ID
vpc-0ee12231ed47367db

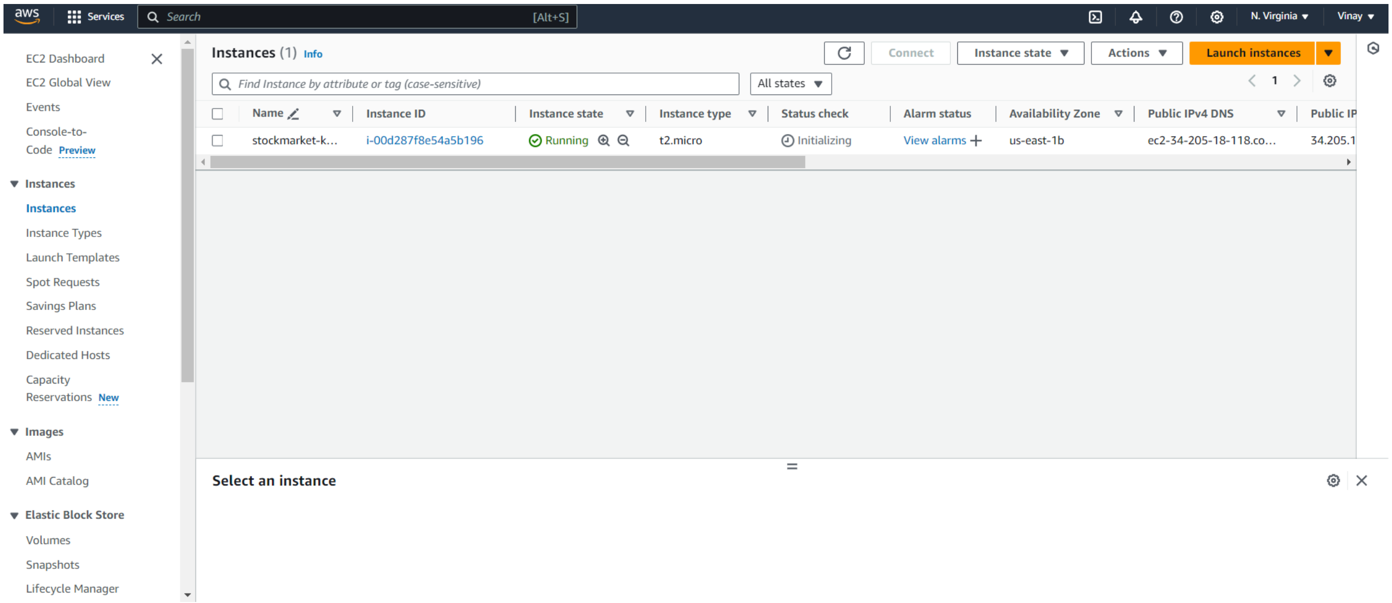
AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations. | Learn more

IAM Role
–

Subnet ID
subnet-0443474a0aeeec339

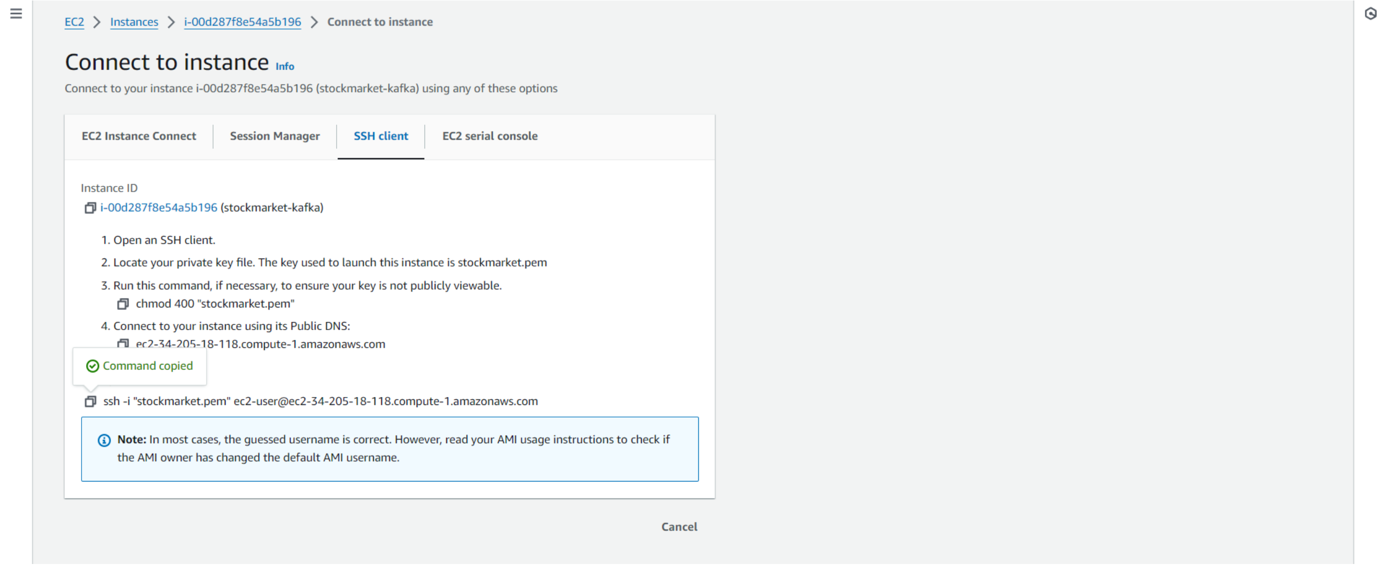
Auto Scaling Group name
–

IMDSv2
Required



Instances (1) Info

□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IP
□	stockmarket-k...	i-00d287f8e54a5b196	✓ Running	t2.micro	Initializing	View alarms +	us-east-1b	ec2-34-205-18-118.co...	34.205.1



EC2 > Instances > i-00d287f8e54a5b196 > Connect to instance

Connect to instance Info

Connect to your instance i-00d287f8e54a5b196 (stockmarket-kafka) using any of these options

Instance ID

i-00d287f8e54a5b196 (stockmarket-kafka)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is stockmarket.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.

```
chmod 400 "stockmarket.pem"
```

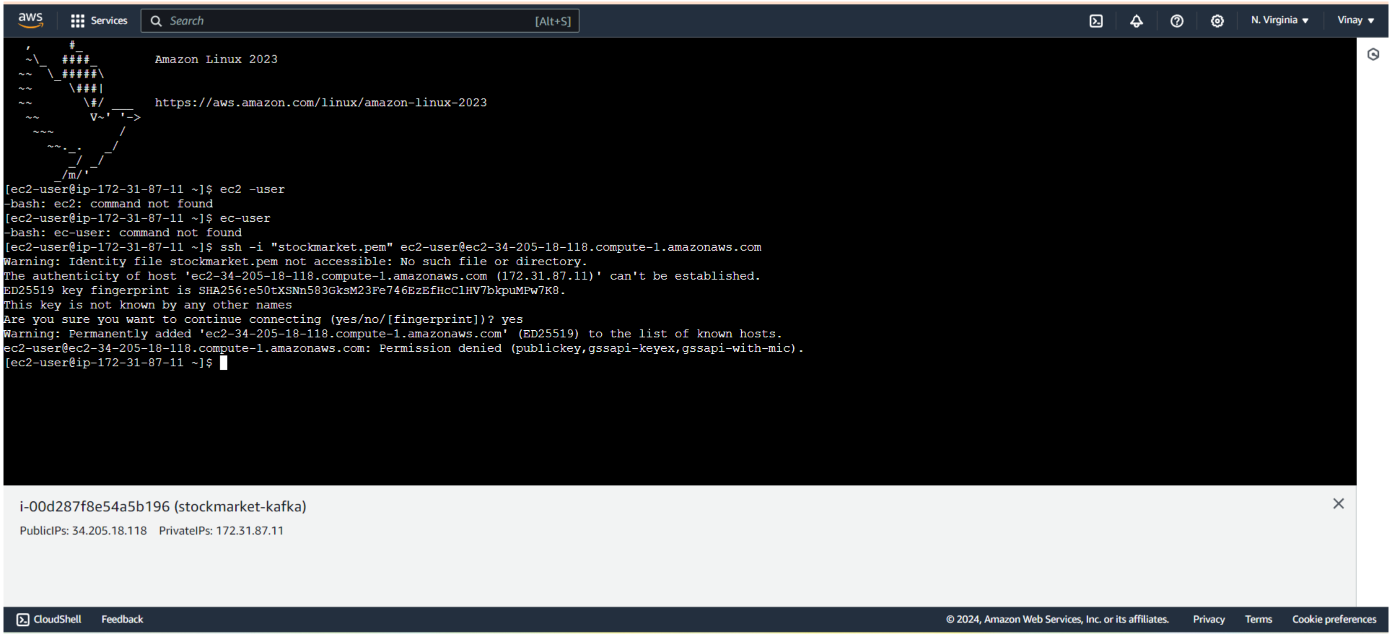
4. Connect to your instance using its Public DNS:

```
ec2-34-205-18-118.compute-1.amazonaws.com
```

✓ Command copied

```
ssh -i "stockmarket.pem" ec2-user@ec2-34-205-18-118.compute-1.amazonaws.com
```

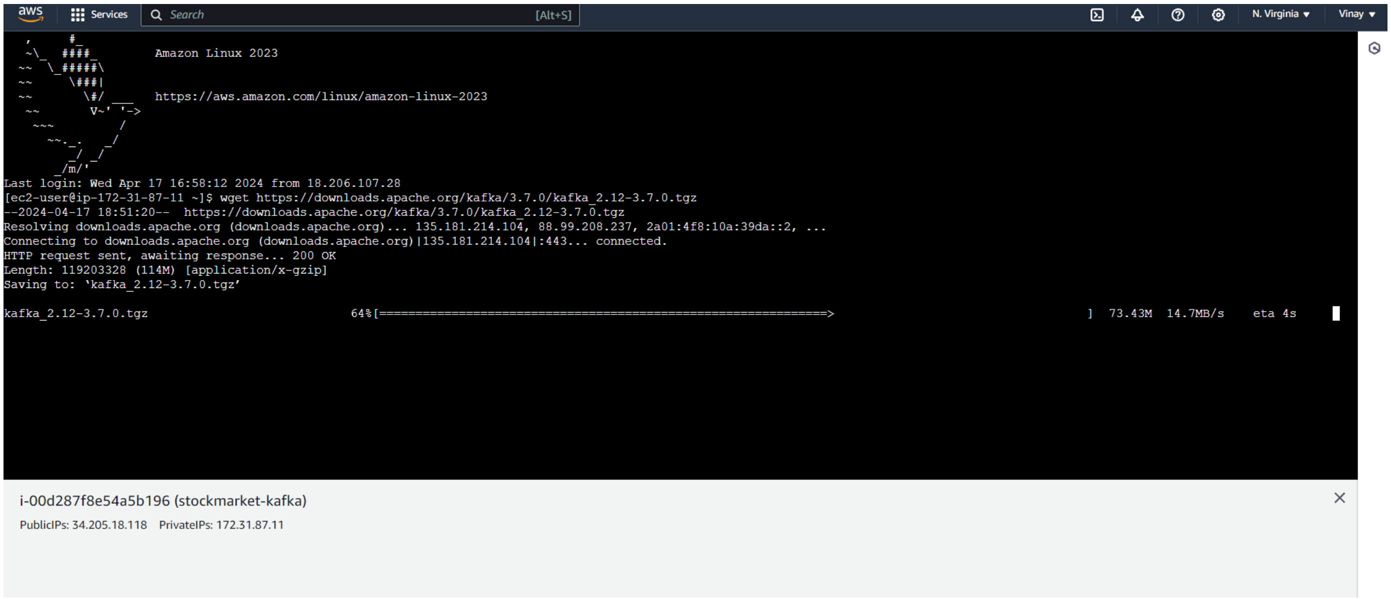
Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.



```
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-87-11 ~]$ ec2 -user
-bash: ec2: command not found
[ec2-user@ip-172-31-87-11 ~]$ ec-user
-bash: ec-user: command not found
[ec2-user@ip-172-31-87-11 ~]$ ssh -i "stockmarket.pem" ec2-user@ec2-34-205-18-118.compute-1.amazonaws.com
Warning: Identity file stockmarket.pem not accessible: No such file or directory.
The authenticity of host 'ec2-34-205-18-118.compute-1.amazonaws.com (172.31.87.11)' can't be established.
ED25519 key fingerprint is SHA256:e50tXSNn583GksM23Fe746EzEfHcClHV7bkpuMPw7K8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-34-205-18-118.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
ec2-user@ec2-34-205-18-118.compute-1.amazonaws.com: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[ec2-user@ip-172-31-87-11 ~]$
```

i-00d287f8e54a5b196 (stockmarket-kafka)

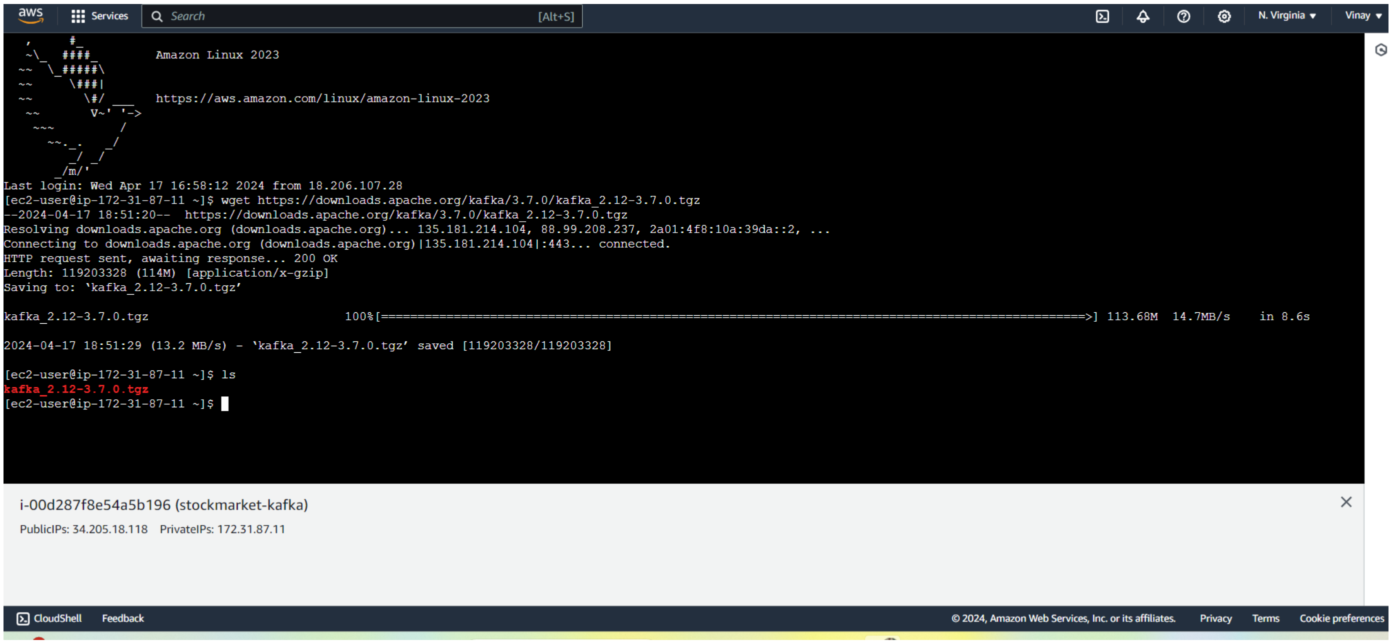
PublicIPs: 34.205.18.118   PrivateIPs: 172.31.87.11



```
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Wed Apr 17 16:58:12 2024 from 18.206.107.28
[ec2-user@ip-172-31-87-11 ~]$ wget https://downloads.apache.org/kafka/3.7.0/kafka_2.12-3.7.0.tgz
--2024-04-17 18:51:20--  https://downloads.apache.org/kafka/3.7.0/kafka_2.12-3.7.0.tgz
Resolving downloads.apache.org (downloads.apache.org)... 135.181.214.104, 88.99.208.237, 2a01:4f8:10a:39da::2, ...
Connecting to downloads.apache.org (downloads.apache.org)|135.181.214.104|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 119203328 (114M) [application/x-gzip]
Saving to: ‘kafka_2.12-3.7.0.tgz’

kafka_2.12-3.7.0.tgz                     64%[===============================================================>                                   ]  73.43M  14.7MB/s    eta 4s
```

i-00d287f8e54a5b196 (stockmarket-kafka)

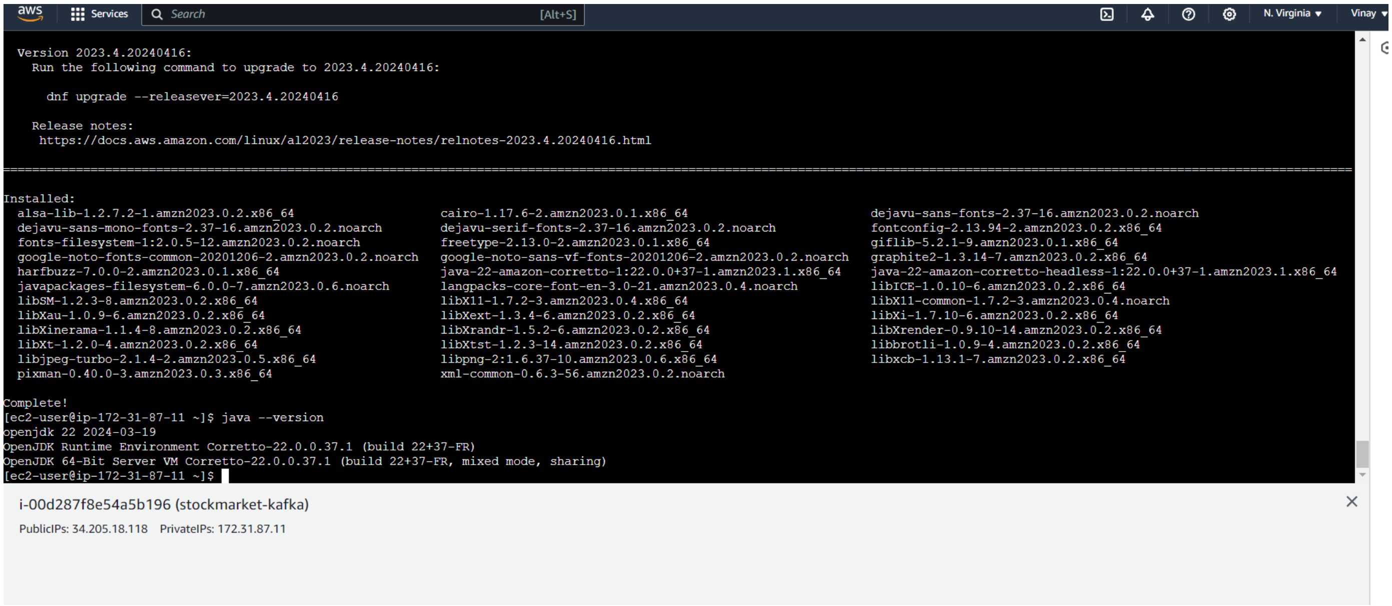
PublicIPs: 34.205.18.118   PrivateIPs: 172.31.87.11



```
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Wed Apr 17 16:58:12 2024 from 18.206.107.28
[ec2-user@ip-172-31-87-11 ~]$ wget https://downloads.apache.org/kafka/3.7.0/kafka_2.12-3.7.0.tgz
--2024-04-17 18:51:20--  https://downloads.apache.org/kafka/3.7.0/kafka_2.12-3.7.0.tgz
Resolving downloads.apache.org (downloads.apache.org)... 135.181.214.104, 88.99.208.237, 2a01:4f8:10a:39da::2, ...
Connecting to downloads.apache.org (downloads.apache.org)|135.181.214.104|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 119203328 (114M) [application/x-gzip]
Saving to: ‘kafka_2.12-3.7.0.tgz’

kafka_2.12-3.7.0.tgz                       100%[=========================================================================================>] 113.68M  14.7MB/s    in 8.6s

2024-04-17 18:51:29 (13.2 MB/s) - ‘kafka_2.12-3.7.0.tgz’ saved [119203328/119203328]

[ec2-user@ip-172-31-87-11 ~]$ ls
kafka_2.12-3.7.0.tgz
[ec2-user@ip-172-31-87-11 ~]$ 
```

i-00d287f8e54a5b196 (stockmarket-kafka)

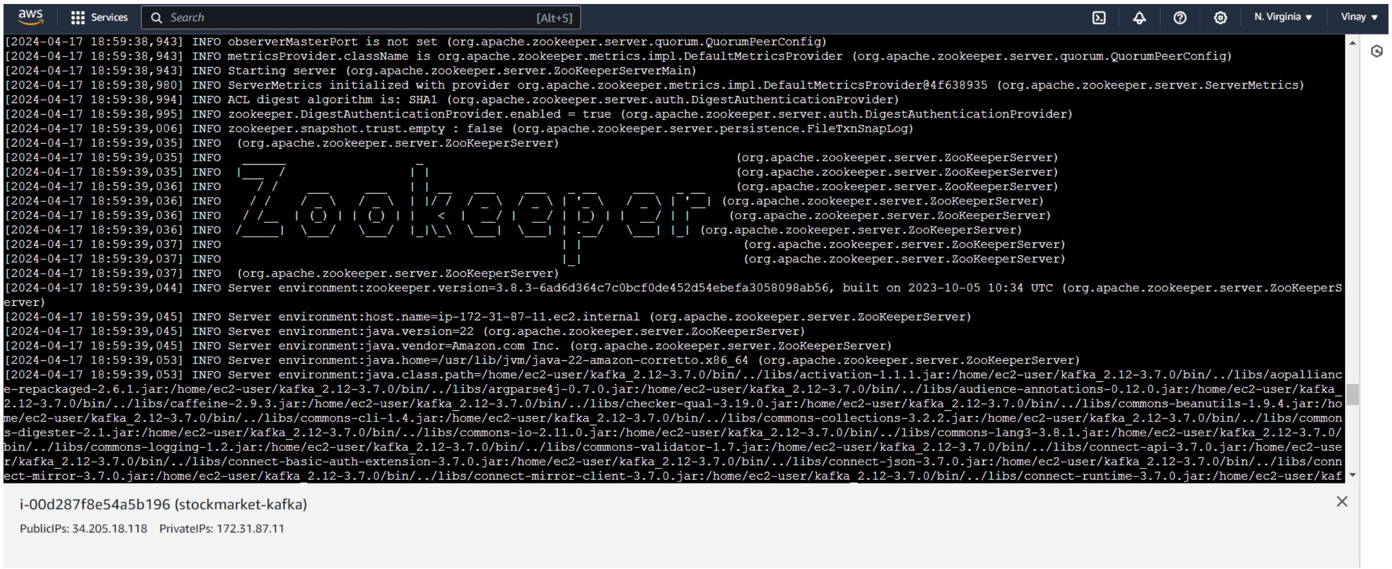
PublicIPs: 34.205.18.118   PrivateIPs: 172.31.87.11



```
  Version 2023.4.20240416:
    Run the following command to upgrade to 2023.4.20240416:

      dnf upgrade --releasever=2023.4.20240416

    Release notes:
     https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.4.20240416.html

==============================================================================================================================================================================

Installed:
  alsa-lib-1.2.7.2-1.amzn2023.0.2.x86_64                   cairo-1.17.6-2.amzn2023.0.1.x86_64                          dejavu-sans-fonts-2.37-16.amzn2023.0.2.noarch
  dejavu-sans-mono-fonts-2.37-16.amzn2023.0.2.noarch       dejavu-serif-fonts-2.37-16.amzn2023.0.2.noarch              fontconfig-2.13.94-2.amzn2023.0.2.x86_64
  fonts-filesystem-1:2.0.5-12.amzn2023.0.2.noarch          freetype-2.13.0-2.amzn2023.0.1.x86_64                       giflib-5.2.1-9.amzn2023.0.1.x86_64
  google-noto-fonts-common-20201206-2.amzn2023.0.2.noarch  google-noto-sans-vf-fonts-20201206-2.amzn2023.0.2.noarch    graphite2-1.3.14-7.amzn2023.0.2.x86_64
  harfbuzz-7.0.0-2.amzn2023.0.1.x86_64                     java-22-amazon-corretto-1:22.0.0+37-1.amzn2023.1.x86_64     java-22-amazon-corretto-headless-1:22.0.0+37-1.amzn2023.1.x86_64
  javapackages-filesystem-6.0.0-7.amzn2023.0.6.noarch      langpacks-core-font-en-3.0-21.amzn2023.0.4.noarch           libICE-1.0.10-6.amzn2023.0.2.x86_64
  libSM-1.2.3-8.amzn2023.0.2.x86_64                        libX11-1.7.2-3.amzn2023.0.4.x86_64                          libX11-common-1.7.2-3.amzn2023.0.4.noarch
  libXau-1.0.9-6.amzn2023.0.2.x86_64                       libXext-1.3.4-6.amzn2023.0.2.x86_64                         libXi-1.7.10-6.amzn2023.0.2.x86_64
  libXinerama-1.1.4-8.amzn2023.0.2.x86_64                  libXrandr-1.5.2-6.amzn2023.0.2.x86_64                       libXrender-0.9.10-14.amzn2023.0.2.x86_64
  libXt-1.2.0-4.amzn2023.0.2.x86_64                        libXtst-1.2.3-14.amzn2023.0.2.x86_64                        libbrotli-1.0.9-4.amzn2023.0.2.x86_64
  libjpeg-turbo-2.1.4-2.amzn2023.0.5.x86_64                libpng-2:1.6.37-10.amzn2023.0.6.x86_64                      libxcb-1.13.1-7.amzn2023.0.2.x86_64
  pixman-0.40.0-3.amzn2023.0.3.x86_64                      xml-common-0.6.3-56.amzn2023.0.2.noarch

Complete!
[ec2-user@ip-172-31-87-11 ~]$ java --version
openjdk 22 2024-03-19
OpenJDK Runtime Environment Corretto-22.0.0.37.1 (build 22+37-FR)
OpenJDK 64-Bit Server VM Corretto-22.0.0.37.1 (build 22+37-FR, mixed mode, sharing)
[ec2-user@ip-172-31-87-11 ~]$ 
```

i-00d287f8e54a5b196 (stockmarket-kafka)

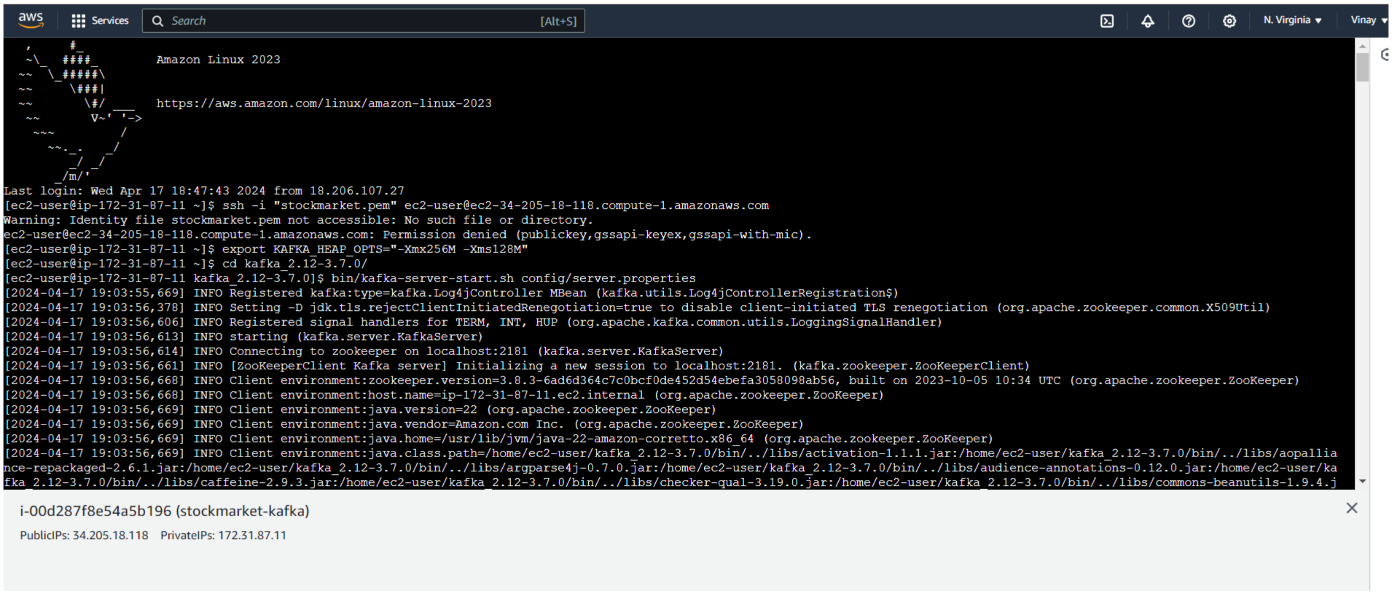
PublicIPs: 34.205.18.118   PrivateIPs: 172.31.87.11



```
[2024-04-17 18:59:38,943] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-04-17 18:59:38,943] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-04-17 18:59:38,943] INFO Starting server (org.apache.zookeeper.server.ZooKeeperServerMain)
[2024-04-17 18:59:38,980] INFO ServerMetrics initialized with provider org.apache.zookeeper.metrics.impl.DefaultMetricsProvider@4f638935 (org.apache.zookeeper.server.ServerMetrics)
[2024-04-17 18:59:38,994] INFO ACL digest algorithm is: SHA1 (org.apache.zookeeper.server.auth.DigestAuthenticationProvider)
[2024-04-17 18:59:38,995] INFO zookeeper.DigestAuthenticationProvider.enabled = true (org.apache.zookeeper.server.auth.DigestAuthenticationProvider)
[2024-04-17 18:59:39,006] INFO zookeeper.snapshot.trust.empty : false (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
[2024-04-17 18:59:39,035] INFO  (org.apache.zookeeper.server.ZooKeeperServer)
[2024-04-17 18:59:39,035] INFO   ______                  _                                          (org.apache.zookeeper.server.ZooKeeperServer)
[2024-04-17 18:59:39,035] INFO  |___  /                 | |                                         (org.apache.zookeeper.server.ZooKeeperServer)
[2024-04-17 18:59:39,036] INFO     / /    ___     ___   | |   __   ___    ___   _ __     ___   _ __   (org.apache.zookeeper.server.ZooKeeperServer)
[2024-04-17 18:59:39,036] INFO    / /    / _ \   / _ \  | |  / /  / _ \  / _ \ | '_ \   / _ \ | '__| (org.apache.zookeeper.server.ZooKeeperServer)
[2024-04-17 18:59:39,036] INFO   / /__  | (_) | | (_) | |   <    |  __/ |  __/ | |_) | |  __/ | |     (org.apache.zookeeper.server.ZooKeeperServer)
[2024-04-17 18:59:39,036] INFO  /_____|  \___/   \___/  |_|\_\   \___|  \___|  | .__/   \___| |_| (org.apache.zookeeper.server.ZooKeeperServer)
[2024-04-17 18:59:39,037] INFO                                                  | |                     (org.apache.zookeeper.server.ZooKeeperServer)
[2024-04-17 18:59:39,037] INFO                                                  |_|                     (org.apache.zookeeper.server.ZooKeeperServer)
[2024-04-17 18:59:39,037] INFO  (org.apache.zookeeper.server.ZooKeeperServer)
[2024-04-17 18:59:39,044] INFO Server environment:zookeeper.version=3.8.3-6ad6d364c7c0bcf0de452d54ebefa3058098ab56, built on 2023-10-05 10:34 UTC (org.apache.zookeeper.server.ZooKeeperS
erver)
[2024-04-17 18:59:39,045] INFO Server environment:host.name=ip-172-31-87-11.ec2.internal (org.apache.zookeeper.server.ZooKeeperServer)
[2024-04-17 18:59:39,045] INFO Server environment:java.version=22 (org.apache.zookeeper.server.ZooKeeperServer)
[2024-04-17 18:59:39,045] INFO Server environment:java.vendor=Amazon.com Inc. (org.apache.zookeeper.server.ZooKeeperServer)
[2024-04-17 18:59:39,053] INFO Server environment:java.home=/usr/lib/jvm/java-22-amazon-corretto.x86_64 (org.apache.zookeeper.server.ZooKeeperServer)
[2024-04-17 18:59:39,053] INFO Server environment:java.class.path=/home/ec2-user/kafka_2.12-3.7.0/bin/../libs/activation-1.1.1.jar:/home/ec2-user/kafka_2.12-3.7.0/bin/../libs/aopallianc
e-repackaged-2.6.1.jar:/home/ec2-user/kafka_2.12-3.7.0/bin/../libs/argparse4j-0.7.0.jar:/home/ec2-user/kafka_2.12-3.7.0/bin/../libs/audience-annotations-0.12.0.jar:/home/ec2-user/kafka_
2.12-3.7.0/bin/../libs/caffeine-2.9.3.jar:/home/ec2-user/kafka_2.12-3.7.0/bin/../libs/checker-qual-3.19.0.jar:/home/ec2-user/kafka_2.12-3.7.0/bin/../libs/commons-beanutils-1.9.4.jar:/ho
me/ec2-user/kafka_2.12-3.7.0/bin/../libs/commons-cli-1.4.jar:/home/ec2-user/kafka_2.12-3.7.0/bin/../libs/commons-collections-3.2.2.jar:/home/ec2-user/kafka_2.12-3.7.0/bin/../libs/common
s-digester-2.1.jar:/home/ec2-user/kafka_2.12-3.7.0/bin/../libs/commons-io-2.11.0.jar:/home/ec2-user/kafka_2.12-3.7.0/bin/../libs/commons-lang3-3.8.1.jar:/home/ec2-user/kafka_2.12-3.7.0/
bin/../libs/commons-logging-1.2.jar:/home/ec2-user/kafka_2.12-3.7.0/bin/../libs/commons-validator-1.7.jar:/home/ec2-user/kafka_2.12-3.7.0/bin/../libs/connect-api-3.7.0.jar:/home/ec2-use
r/kafka_2.12-3.7.0/bin/../libs/connect-basic-auth-extension-3.7.0.jar:/home/ec2-user/kafka_2.12-3.7.0/bin/../libs/connect-json-3.7.0.jar:/home/ec2-user/kafka_2.12-3.7.0/bin/../libs/conn
ect-mirror-3.7.0.jar:/home/ec2-user/kafka_2.12-3.7.0/bin/../libs/connect-mirror-client-3.7.0.jar:/home/ec2-user/kafka_2.12-3.7.0/bin/../libs/connect-runtime-3.7.0.jar:/home/ec2-user/kaf
```

i-00d287f8e54a5b196 (stockmarket-kafka)

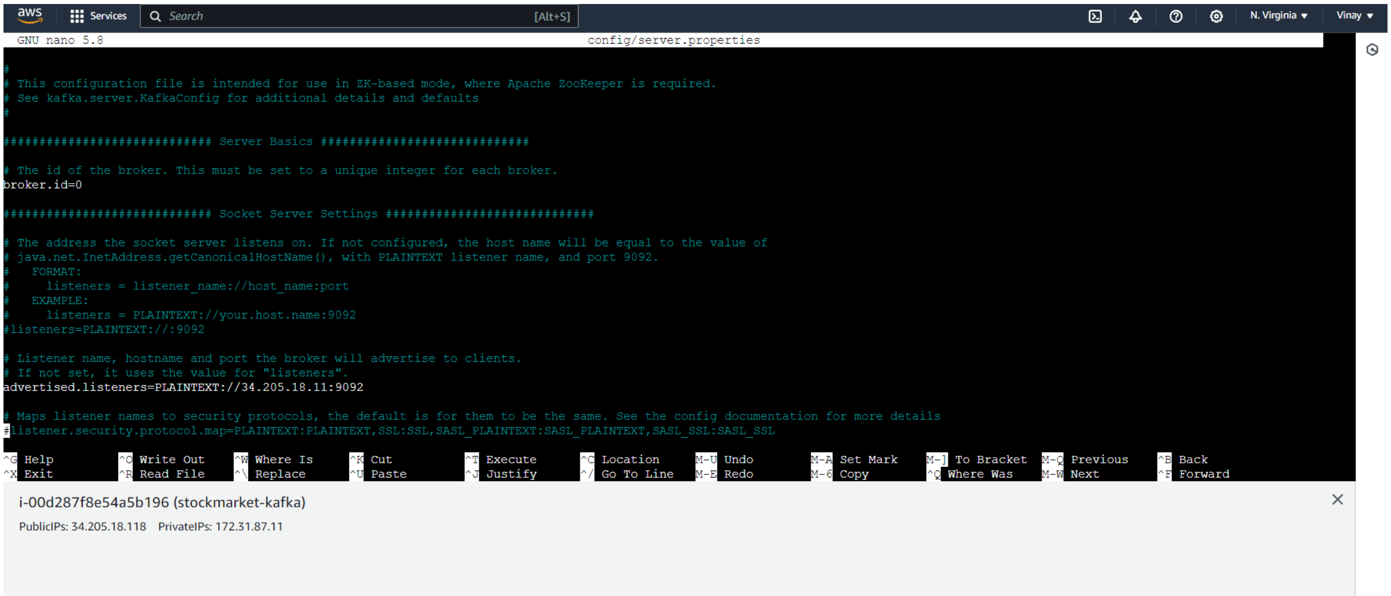
PublicIPs: 34.205.18.118 PrivateIPs: 172.31.87.11



```
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Wed Apr 17 18:47:43 2024 from 18.206.107.27
[ec2-user@ip-172-31-87-11 ~]$ ssh -i "stockmarket.pem" ec2-user@ec2-34-205-18-118.compute-1.amazonaws.com
Warning: Identity file stockmarket.pem not accessible: No such file or directory.
ec2-user@ec2-34-205-18-118.compute-1.amazonaws.com: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[ec2-user@ip-172-31-87-11 ~]$ export KAFKA_HEAP_OPTS="-Xmx256M -Xms128M"
[ec2-user@ip-172-31-87-11 ~]$ cd kafka_2.12-3.7.0/
[ec2-user@ip-172-31-87-11 kafka_2.12-3.7.0]$ bin/kafka-server-start.sh config/server.properties
[2024-04-17 19:03:55,669] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration$)
[2024-04-17 19:03:56,378] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiation=true to disable client-initiated TLS renegotiation (org.apache.zookeeper.common.X509Util)
[2024-04-17 19:03:56,606] INFO Registered signal handlers for TERM, INT, HUP (org.apache.kafka.common.utils.LoggingSignalHandler)
[2024-04-17 19:03:56,613] INFO starting (kafka.server.KafkaServer)
[2024-04-17 19:03:56,614] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer)
[2024-04-17 19:03:56,661] INFO [ZooKeeperClient Kafka server] Initializing a new session to localhost:2181. (kafka.zookeeper.ZooKeeperClient)
[2024-04-17 19:03:56,668] INFO Client environment:zookeeper.version=3.8.3-6ad6d364c7c0bcf0de452d54ebefa3058098ab56, built on 2023-10-05 10:34 UTC (org.apache.zookeeper.ZooKeeper)
[2024-04-17 19:03:56,668] INFO Client environment:host.name=ip-172-31-87-11.ec2.internal (org.apache.zookeeper.ZooKeeper)
[2024-04-17 19:03:56,669] INFO Client environment:java.version=22 (org.apache.zookeeper.ZooKeeper)
[2024-04-17 19:03:56,669] INFO Client environment:java.vendor=Amazon.com Inc. (org.apache.zookeeper.ZooKeeper)
[2024-04-17 19:03:56,669] INFO Client environment:java.home=/usr/lib/jvm/java-22-amazon-corretto.x86_64 (org.apache.zookeeper.ZooKeeper)
[2024-04-17 19:03:56,669] INFO Client environment:java.class.path=/home/ec2-user/kafka_2.12-3.7.0/bin/../libs/activation-1.1.1.jar:/home/ec2-user/kafka_2.12-3.7.0/bin/../libs/aopallia
nce-repackaged-2.6.1.jar:/home/ec2-user/kafka_2.12-3.7.0/bin/../libs/argparse4j-0.7.0.jar:/home/ec2-user/kafka_2.12-3.7.0/bin/../libs/audience-annotations-0.12.0.jar:/home/ec2-user/ka
fka_2.12-3.7.0/bin/../libs/caffeine-2.9.3.jar:/home/ec2-user/kafka_2.12-3.7.0/bin/../libs/checker-qual-3.19.0.jar:/home/ec2-user/kafka_2.12-3.7.0/bin/../libs/commons-beanutils-1.9.4.j
```

i-00d287f8e54a5b196 (stockmarket-kafka)

PublicIPs: 34.205.18.118 PrivateIPs: 172.31.87.11

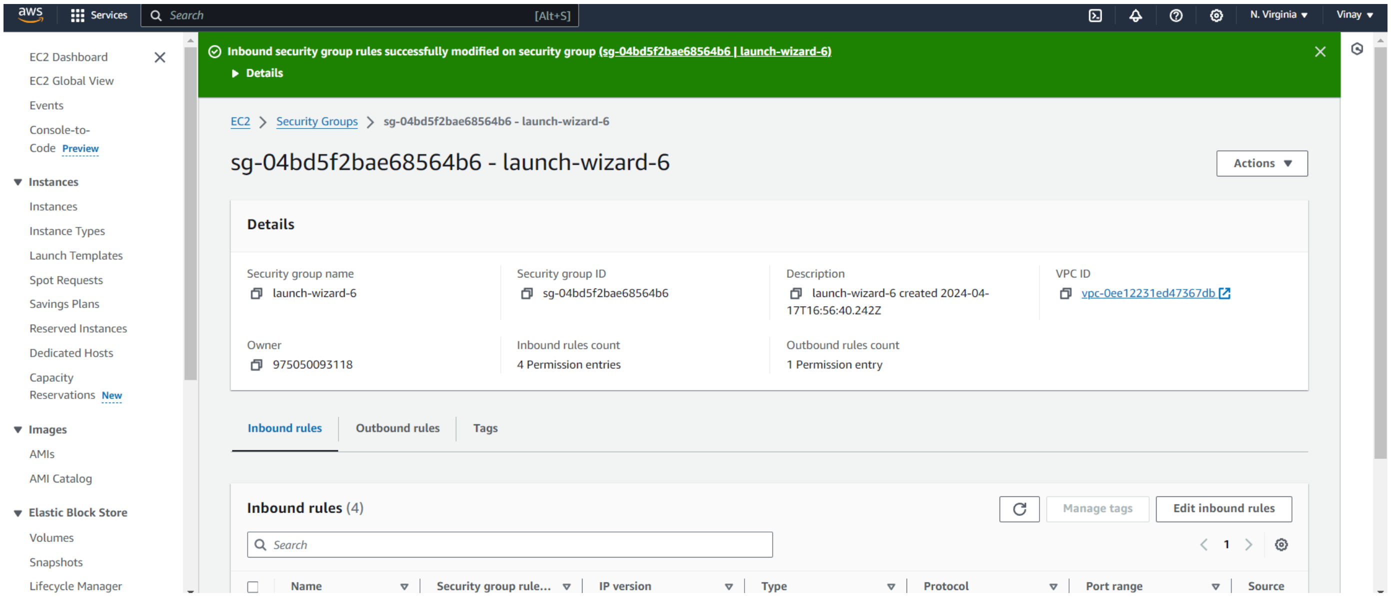


```
#
# This configuration file is intended for use in ZK-based mode, where Apache ZooKeeper is required.
# See kafka.server.KafkaConfig for additional details and defaults
#

############################# 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://34.205.18.11:9092

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

i-00d287f8e54a5b196 (stockmarket-kafka)

PublicIPs: 34.205.18.118 PrivateIPs: 172.31.87.11

4.2 Output Screen:-



Inbound security group rules successfully modified on security group (sg-04bd5f2bae68564b6 | launch-wizard-6)

sg-04bd5f2bae68564b6 - launch-wizard-6

Details

Security group name: launch-wizard-6

Security group ID: sg-04bd5f2bae68564b6

Description: launch-wizard-6 created 2024-04-17T16:56:40.242Z

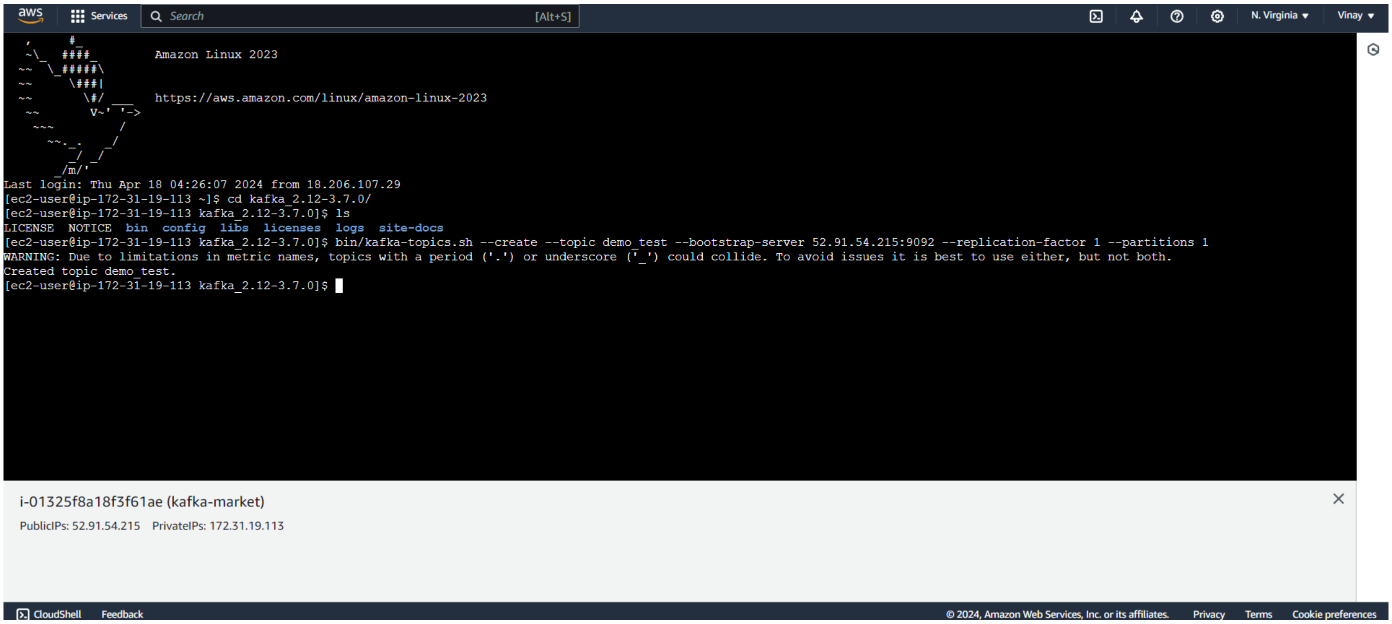
VPC ID: vpc-0ee12231ed47367db

Owner: 975050093118

Inbound rules count: 4 Permission entries

Outbound rules count: 1 Permission entry

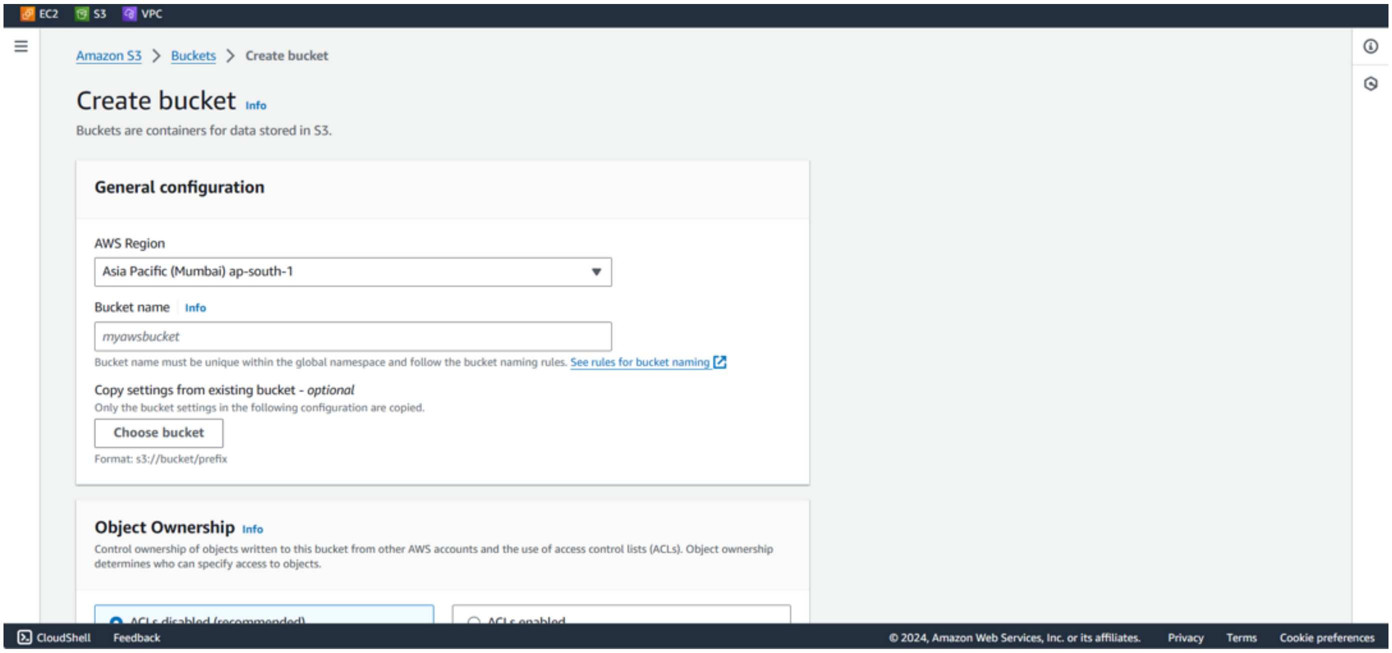
Inbound rules (4)



```
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Thu Apr 18 04:26:07 2024 from 18.206.107.29
[ec2-user@ip-172-31-19-113 ~]$ cd kafka_2.12-3.7.0/
[ec2-user@ip-172-31-19-113 kafka_2.12-3.7.0]$ ls
LICENSE  NOTICE  bin  config  libs  licenses  logs  site-docs
[ec2-user@ip-172-31-19-113 kafka_2.12-3.7.0]$ bin/kafka-topics.sh --create --topic demo_test --bootstrap-server 52.91.54.215:9092 --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_test.
[ec2-user@ip-172-31-19-113 kafka_2.12-3.7.0]$
```

i-01325f8a18f3f61ae (kafka-market)

PublicIPs: 52.91.54.215 PrivateIPs: 172.31.19.113


EC2
S3
VPC
Amazon S3 > Buckets > Create bucket
Info
Choose bucket
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Create bucket

Buckets are containers for data stored in S3.

General configuration

AWS Region: Asia Pacific (Mumbai) ap-south-1

Bucket name: myawsbucket

Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming

Copy settings from existing bucket - *optional*

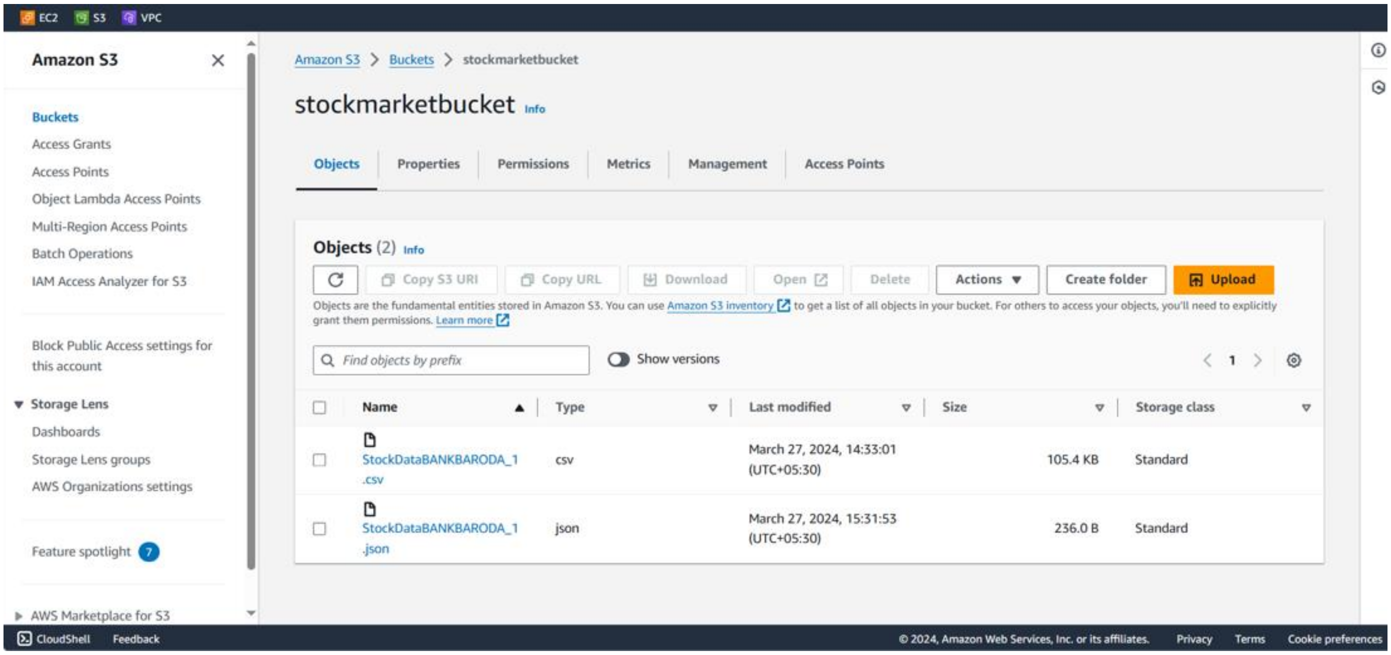
Only the bucket settings in the following configuration are copied.

Format: s3://bucket/prefix

Object Ownership

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

● ACLs disabled (recommended) ○ ACLs enabled

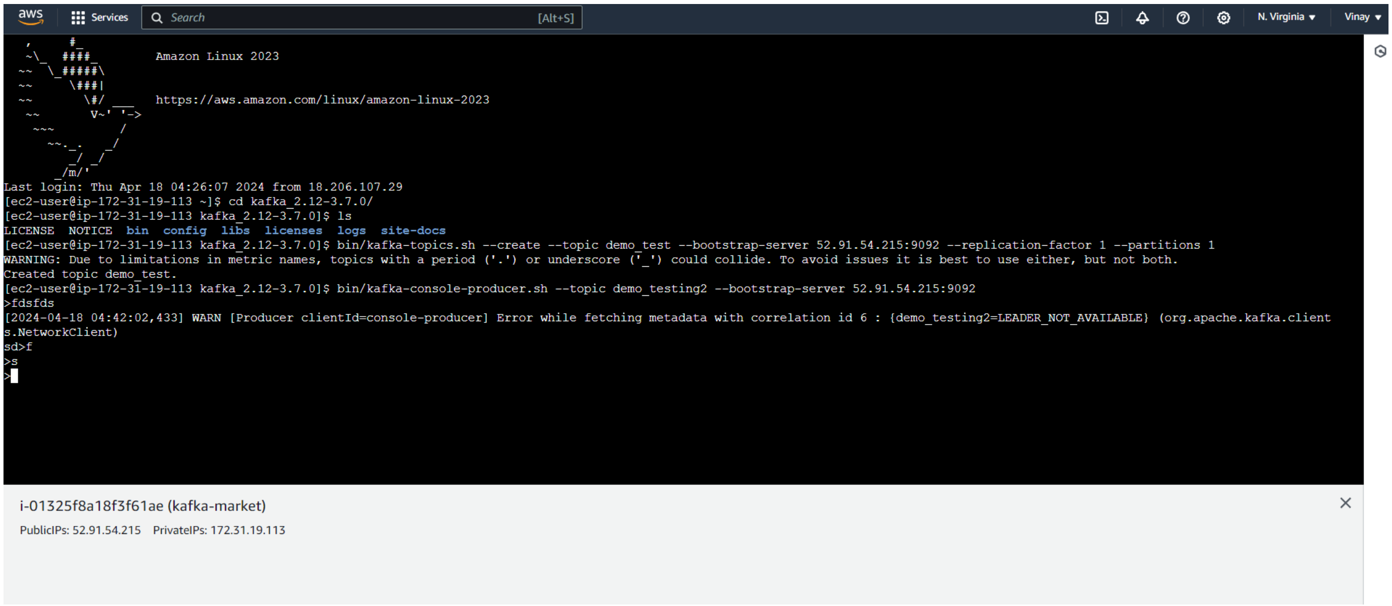


stockmarketbucket

Objects (2)

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

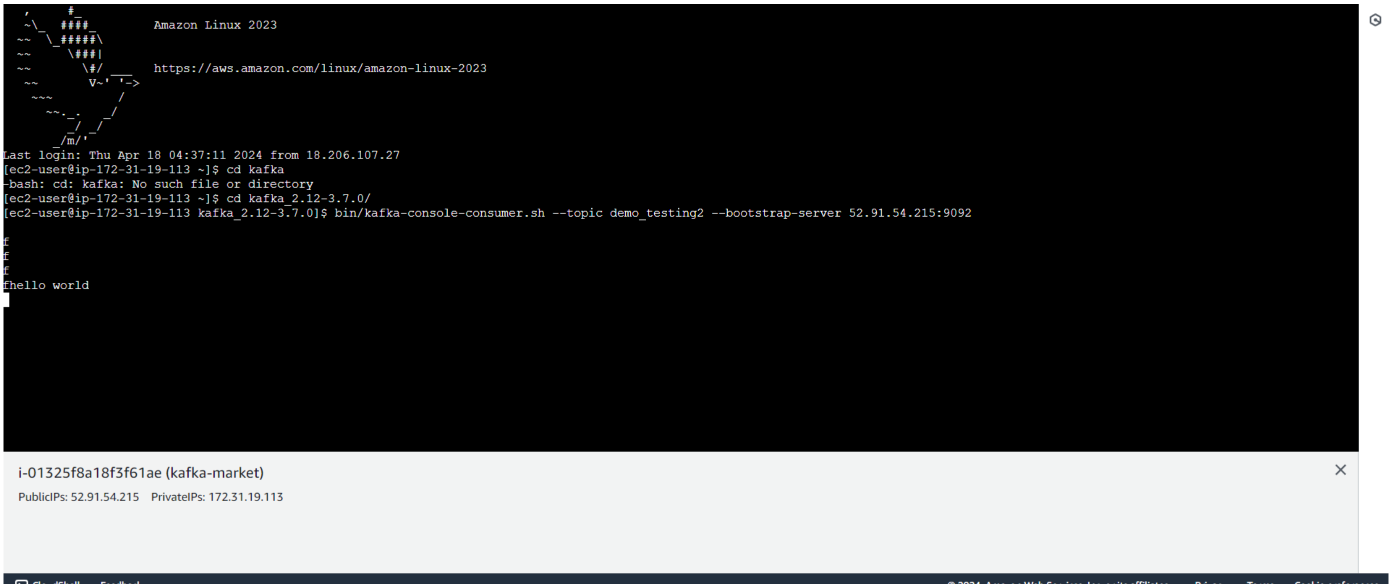
	Name	Type	Last modified	Size	Storage class
□	StockDataBANKBARODA_1.csv	csv	March 27, 2024, 14:33:01 (UTC+05:30)	105.4 KB	Standard
□	StockDataBANKBARODA_1.json	json	March 27, 2024, 15:31:53 (UTC+05:30)	236.0 B	Standard



```
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Thu Apr 18 04:26:07 2024 from 18.206.107.29
[ec2-user@ip-172-31-19-113 ~]$ cd kafka_2.12-3.7.0/
[ec2-user@ip-172-31-19-113 kafka_2.12-3.7.0]$ ls
LICENSE  NOTICE  bin  config  libs  licenses  logs  site-docs
[ec2-user@ip-172-31-19-113 kafka_2.12-3.7.0]$ bin/kafka-topics.sh --create --topic demo_test --bootstrap-server 52.91.54.215:9092 --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_test.
[ec2-user@ip-172-31-19-113 kafka_2.12-3.7.0]$ bin/kafka-console-producer.sh --topic demo_testing2 --bootstrap-server 52.91.54.215:9092
>fdsfds
[2024-04-18 04:42:02,433] WARN [Producer clientId=console-producer] Error while fetching metadata with correlation id 6 : {demo_testing2=LEADER_NOT_AVAILABLE} (org.apache.kafka.client
s.NetworkClient)
sd>f
>s
>
```

i-01325f8a18f3f61ae (kafka-market)

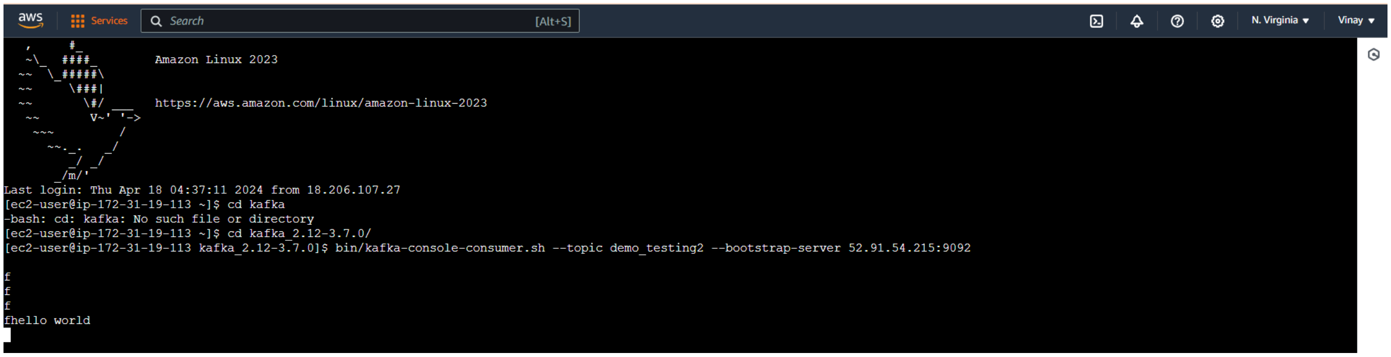
PublicIPs: 52.91.54.215   PrivateIPs: 172.31.19.113



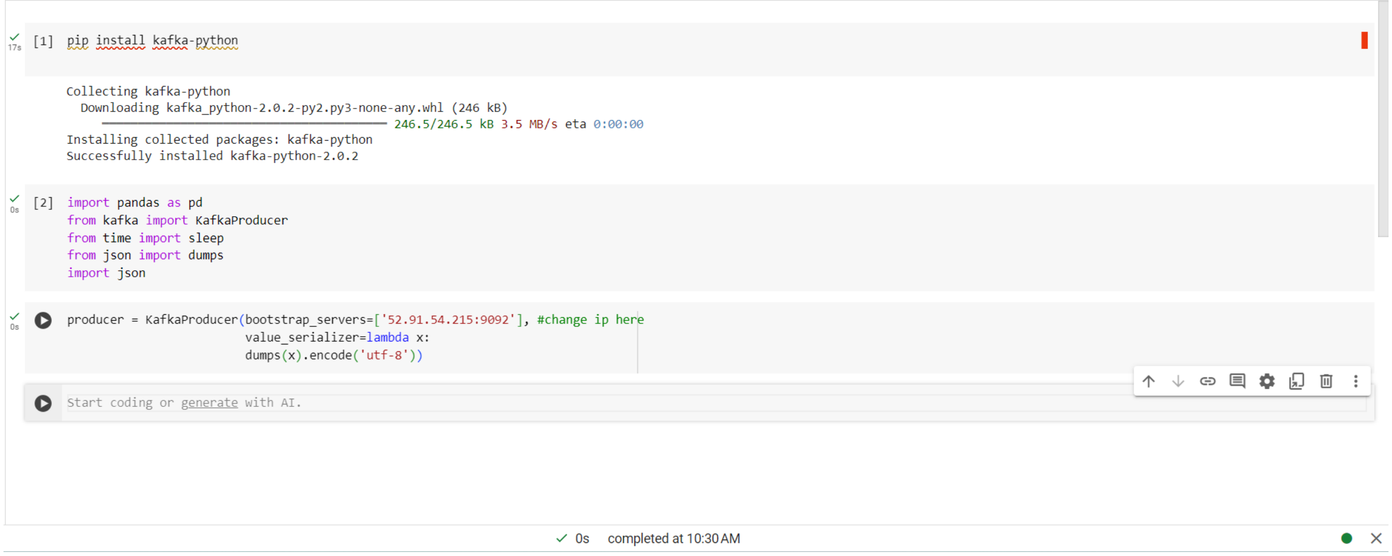
```
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Thu Apr 18 04:37:11 2024 from 18.206.107.27
[ec2-user@ip-172-31-19-113 ~]$ cd kafka
-bash: cd: kafka: No such file or directory
[ec2-user@ip-172-31-19-113 ~]$ cd kafka_2.12-3.7.0/
[ec2-user@ip-172-31-19-113 kafka_2.12-3.7.0]$ bin/kafka-console-consumer.sh --topic demo_testing2 --bootstrap-server 52.91.54.215:9092

f
f
f
fhello world
```

i-01325f8a18f3f61ae (kafka-market)

PublicIPs: 52.91.54.215   PrivateIPs: 172.31.19.113



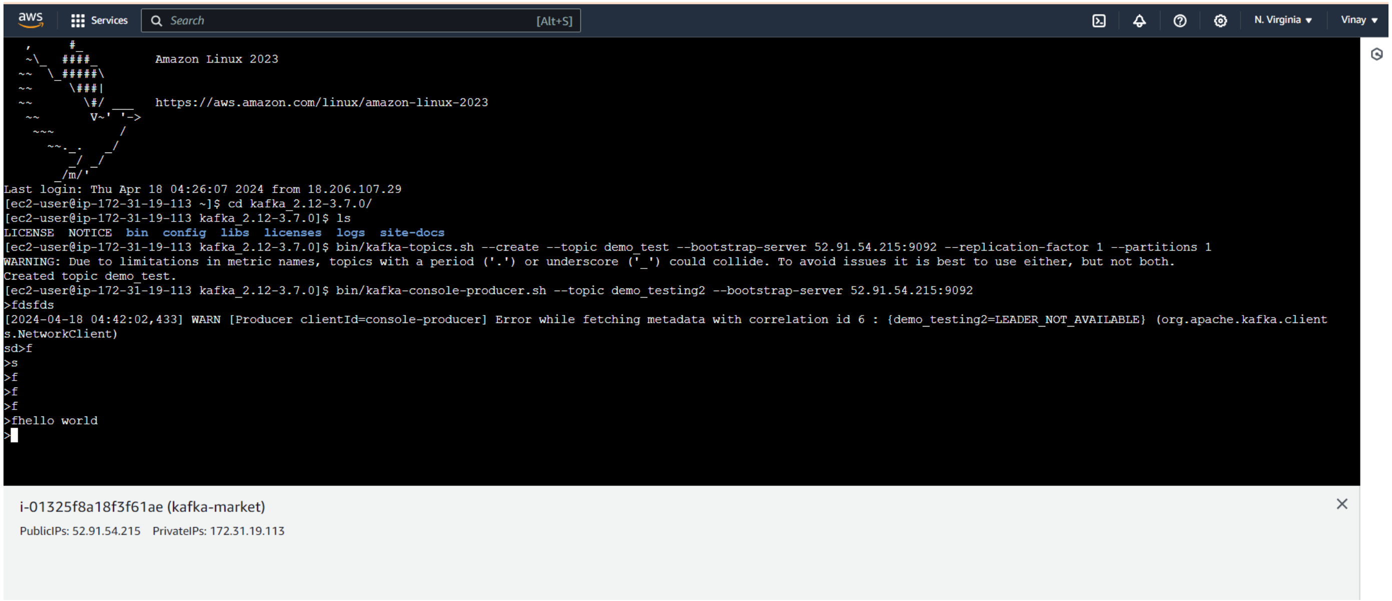
```
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Thu Apr 18 04:37:11 2024 from 18.206.107.27
[ec2-user@ip-172-31-19-113 ~]$ cd kafka
-bash: cd: kafka: No such file or directory
[ec2-user@ip-172-31-19-113 ~]$ cd kafka_2.12-3.7.0/
[ec2-user@ip-172-31-19-113 kafka_2.12-3.7.0]$ bin/kafka-console-consumer.sh --topic demo_testing2 --bootstrap-server 52.91.54.215:9092

f
f
f
fhello world
```



```
[1] pip install kafka-python

    Collecting kafka-python
      Downloading kafka_python-2.0.2-py2.py3-none-any.whl (246 kB)
                                    ━━━━━━━━━━━━━━━━━━━━━━━━━ 246.5/246.5 kB 3.5 MB/s eta 0:00:00
    Installing collected packages: kafka-python
    Successfully installed kafka-python-2.0.2
```

```
[2] import pandas as pd
    from kafka import KafkaProducer
    from time import sleep
    from json import dumps
    import json
```

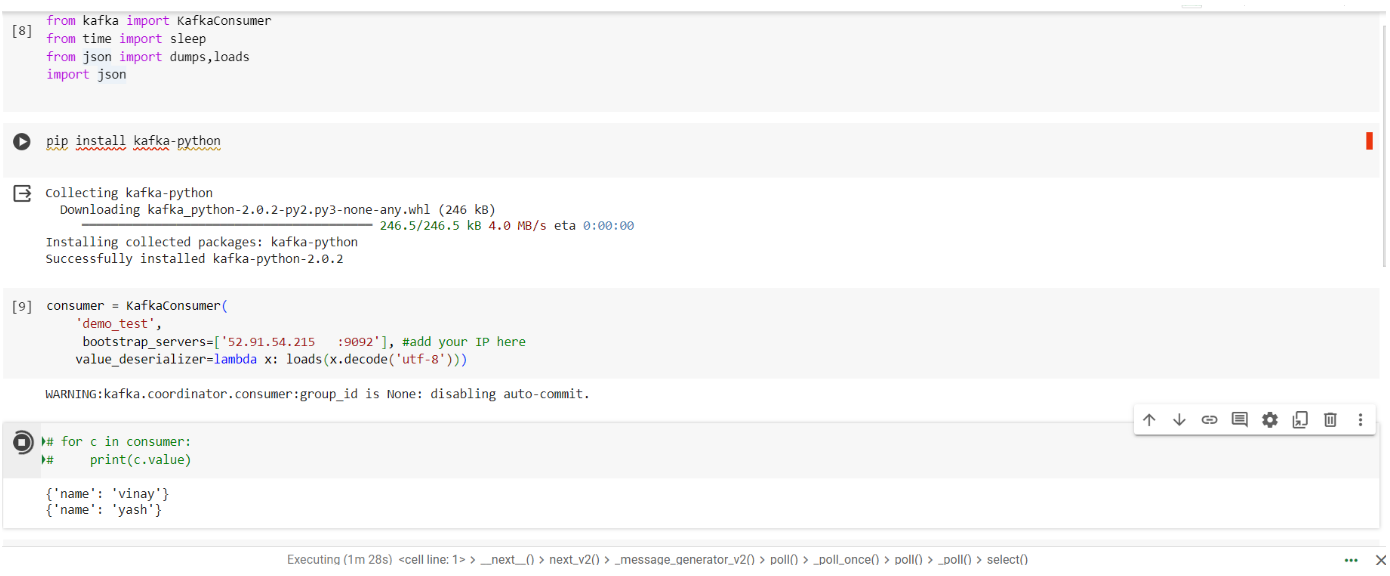
```
producer = KafkaProducer(bootstrap_servers=['52.91.54.215:9092'], #change ip here
                         value_serializer=lambda x:
                         dumps(x).encode('utf-8'))
```



```
  ,     #_
  ~\_  ####_        Amazon Linux 2023
 ~~  \_#####\
 ~~     \###|
 ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
  ~~       V~' '->
   ~~~         /
     ~~._.   _/
        _/ _/
      _/m/'
Last login: Thu Apr 18 04:26:07 2024 from 18.206.107.29
[ec2-user@ip-172-31-19-113 ~]$ cd kafka_2.12-3.7.0/
[ec2-user@ip-172-31-19-113 kafka_2.12-3.7.0]$ ls
LICENSE  NOTICE  bin  config  libs  licenses  logs  site-docs
[ec2-user@ip-172-31-19-113 kafka_2.12-3.7.0]$ bin/kafka-topics.sh --create --topic demo_test --bootstrap-server 52.91.54.215:9092 --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_test.
[ec2-user@ip-172-31-19-113 kafka_2.12-3.7.0]$ bin/kafka-console-producer.sh --topic demo_testing2 --bootstrap-server 52.91.54.215:9092
>fdsfds
[2024-04-18 04:42:02,433] WARN [Producer clientId=console-producer] Error while fetching metadata with correlation id 6 : {demo_testing2=LEADER_NOT_AVAILABLE} (org.apache.kafka.client
s.NetworkClient)
sd>f
>s
>f
>f
>f
>fhello world
>
```

i-01325f8a18f3f61ae (kafka-market)

PublicIPs: 52.91.54.215   PrivateIPs: 172.31.19.113



```
[8]  from kafka import KafkaConsumer
     from time import sleep
     from json import dumps,loads
     import json
```

```
pip install kafka-python
```

```
Collecting kafka-python
  Downloading kafka_python-2.0.2-py2.py3-none-any.whl (246 kB)
                                          246.5/246.5 kB 4.0 MB/s eta 0:00:00
Installing collected packages: kafka-python
Successfully installed kafka-python-2.0.2
```

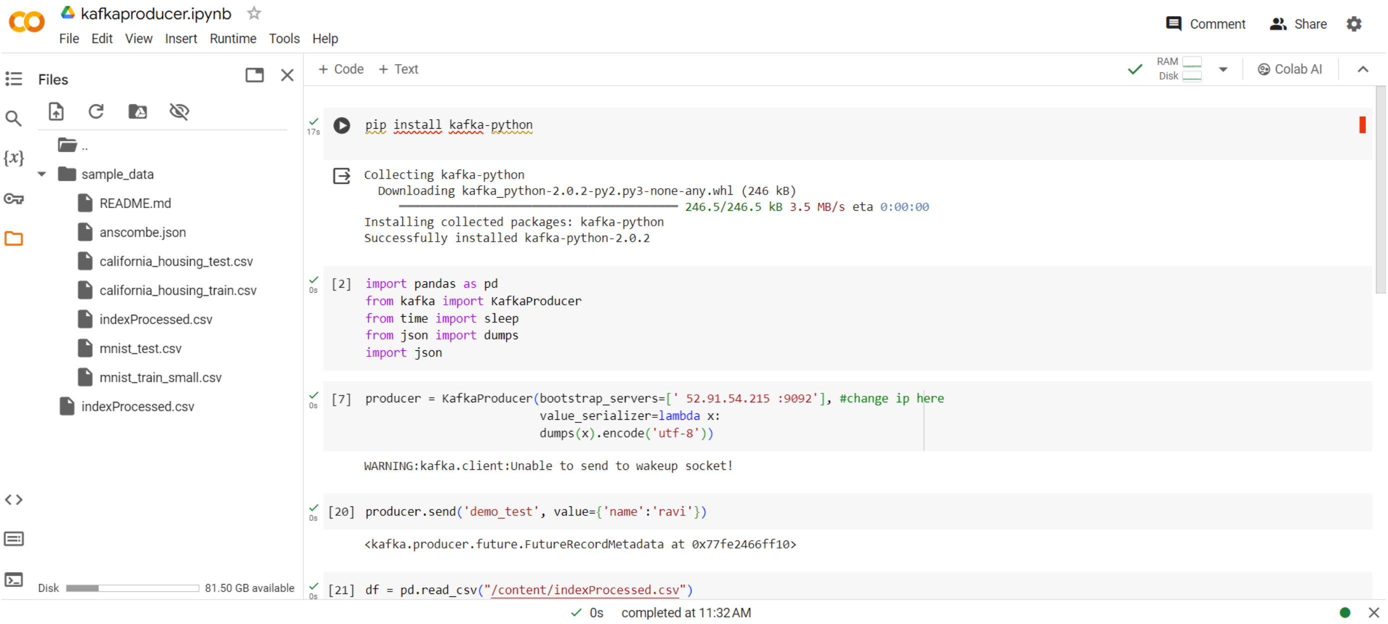
```
[9]  consumer = KafkaConsumer(
         'demo_test',
          bootstrap_servers=['52.91.54.215   :9092'], #add your IP here
         value_deserializer=lambda x: loads(x.decode('utf-8')))
```

```
WARNING:kafka.coordinator.consumer:group_id is None: disabling auto-commit.
```

```
# for c in consumer:
#     print(c.value)
```

```
{'name': 'vinay'}
{'name': 'yash'}
```

Executing (1m 28s) <cell line: 1> › __next__() › next_v2() › _message_generator_v2() › poll() › _poll_once() › poll() › _poll() › select()



```
pip install kafka-python
```

```
Collecting kafka-python
  Downloading kafka_python-2.0.2-py2.py3-none-any.whl (246 kB)
                                        246.5/246.5 kB 3.5 MB/s eta 0:00:00
Installing collected packages: kafka-python
Successfully installed kafka-python-2.0.2
```

```
[2] import pandas as pd
    from kafka import KafkaProducer
    from time import sleep
    from json import dumps
    import json
```

```
[7] producer = KafkaProducer(bootstrap_servers=[' 52.91.54.215 :9092'], #change ip here
                             value_serializer=lambda x:
                             dumps(x).encode('utf-8'))
```

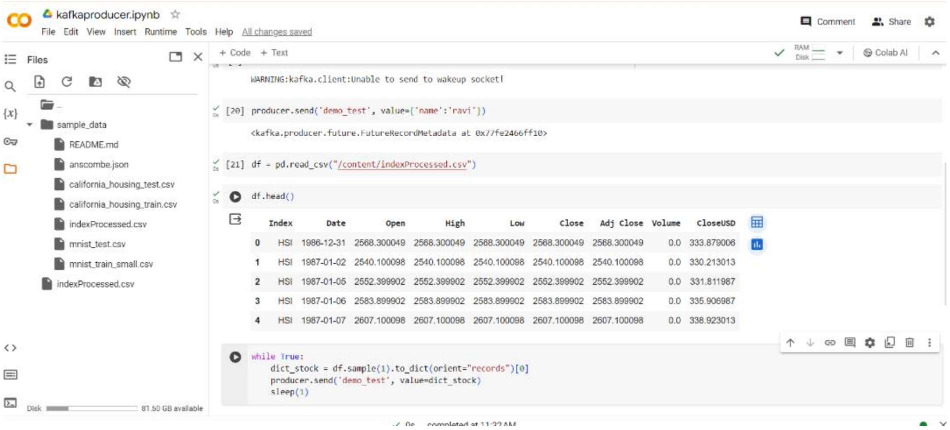
```
WARNING:kafka.client:Unable to send to wakeup socket!
```

```
[20] producer.send('demo_test', value={'name':'ravi'})
```

```
<kafka.producer.future.FutureRecordMetadata at 0x77fe2466ff10>
```

```
[21] df = pd.read_csv("/content/indexProcessed.csv")
```

✓ 0s completed at 11:32 AM



```
WARNING:kafka.client:Unable to send to wakeup socket!
```

```
[20] producer.send('demo_test', value={'name':'ravi'})
```

```
<kafka.producer.future.FutureRecordMetadata at 0x77fe2466ff10>
```

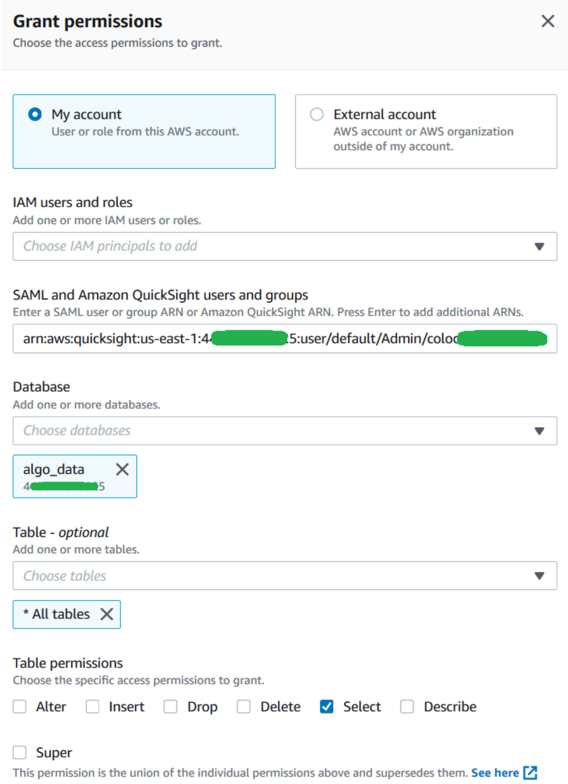
```
[21] df = pd.read_csv("/content/indexProcessed.csv")
```

```
df.head()
```

	Index	Date	Open	High	Low	Close	Adj Close	Volume	CloseUSD
0	HSI	1986-12-31	2568.300049	2568.300049	2568.300049	2568.300049	2568.300049	0.0	333.879006
1	HSI	1987-01-02	2540.100098	2540.100098	2540.100098	2540.100098	2540.100098	0.0	330.213013
2	HSI	1987-01-05	2552.399902	2552.399902	2552.399902	2552.399902	2552.399902	0.0	331.811987
3	HSI	1987-01-06	2583.899902	2583.899902	2583.899902	2583.899902	2583.899902	0.0	335.906987
4	HSI	1987-01-07	2607.100098	2607.100098	2607.100098	2607.100098	2607.100098	0.0	338.923013

```
while True:
    dict_stock = df.sample(1).to_dict(orient="records")[0]
    producer.send('demo_test', value=dict_stock)
    sleep(1)
```

✓ 0s completed at 11:32 AM



Grant permissions

Choose the access permissions to grant.

● My account
User or role from this AWS account.

○ External account
AWS account or AWS organization outside of my account.

IAM users and roles
Add one or more IAM users or roles.
Choose IAM principals to add

SAML and Amazon QuickSight users and groups
Enter a SAML user or group ARN or Amazon QuickSight ARN. Press Enter to add additional ARNs.
arn:aws:quicksight:us-east-1:4[redacted]5:user/default/Admin/coloc[redacted]

Database
Add one or more databases.
Choose databases

algo_data
4[redacted]5

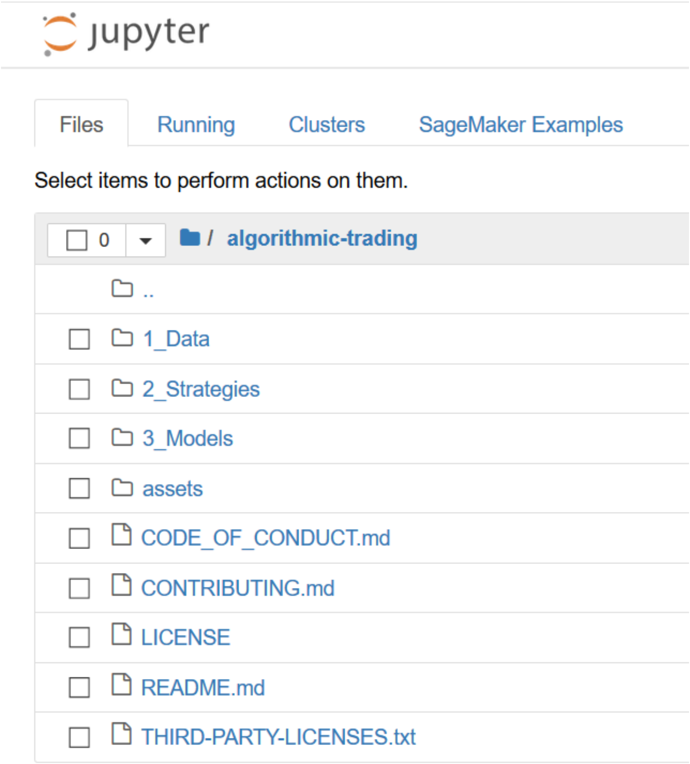
Table - *optional*
Add one or more tables.
Choose tables

* All tables

Table permissions
Choose the specific access permissions to grant.

□ Alter □ Insert □ Drop □ Delete ☑ Select □ Describe

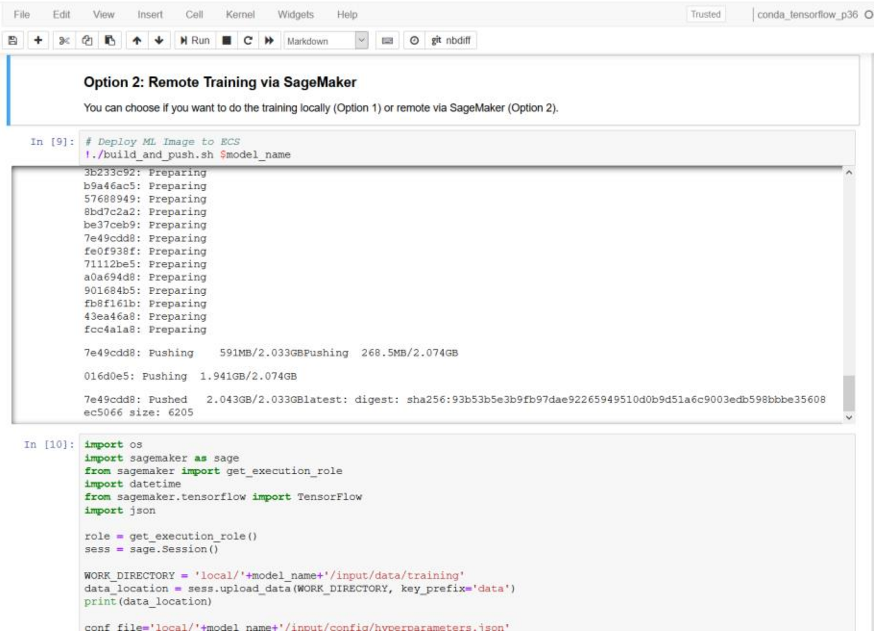
□ Super
This permission is the union of the individual permissions above and supersedes them. See here



Select items to perform actions on them.

□ 0 ▾ / algorithmic-trading

..
□ 1_Data
□ 2_Strategies
□ 3_Models
□ assets
□ CODE_OF_CONDUCT.md
□ CONTRIBUTING.md
□ LICENSE
□ README.md
□ THIRD-PARTY-LICENSES.txt



Option 2: Remote Training via SageMaker

You can choose if you want to do the training locally (Option 1) or remote via SageMaker (Option 2).

In [9]:

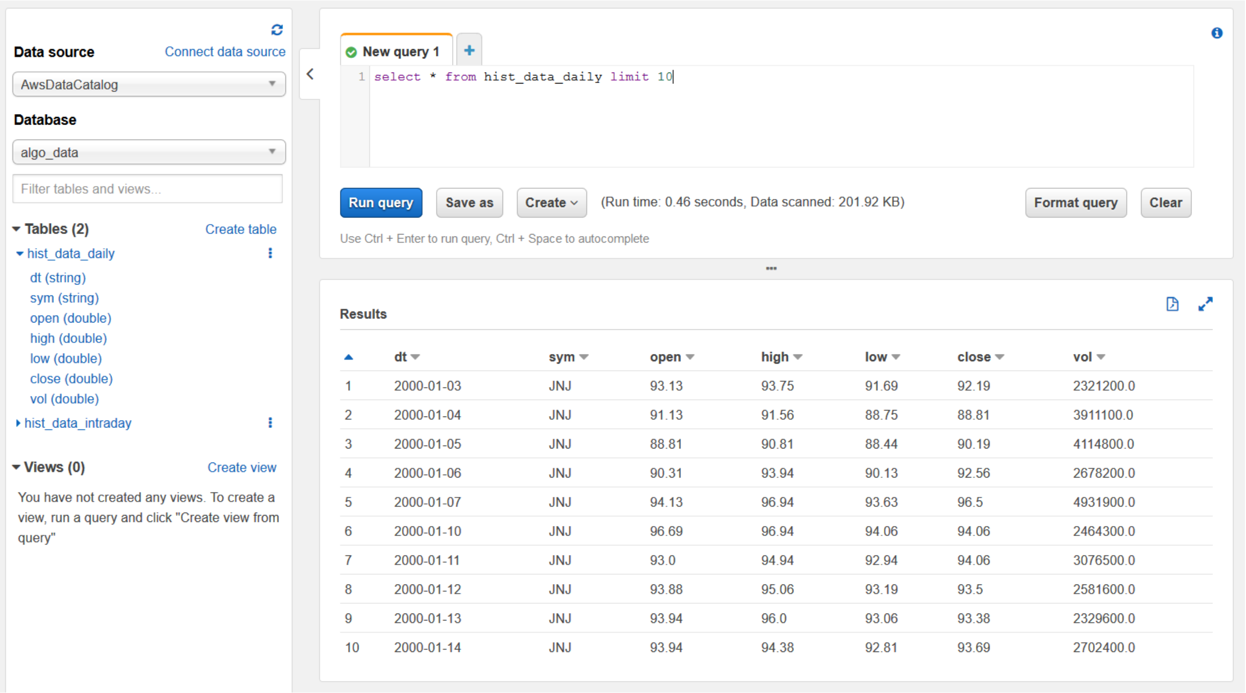
```
# Deploy ML Image to ECS
!./build_and_push.sh $model_name
```

```
3b233c92: Preparing
b9a46ac5: Preparing
57688949: Preparing
8bd7c2a2: Preparing
be37ceb9: Preparing
7e49cdd8: Preparing
fe0f938f: Preparing
71112be5: Preparing
a0a694d8: Preparing
901684b5: Preparing
fb8f161b: Preparing
43ea46a8: Preparing
fcc4a1a8: Preparing

7e49cdd8: Pushing    591MB/2.033GBPushing  268.5MB/2.074GB

016d0e5: Pushing  1.941GB/2.074GB

7e49cdd8: Pushed   2.043GB/2.033GBlatest: digest: sha256:93b53b5e3b9fb97dae92265949510d0b9d51a6c9003edb598bbbe35608
ec5066 size: 6205
```

In [10]:

```
import os
import sagemaker as sage
from sagemaker import get_execution_role
import datetime
from sagemaker.tensorflow import TensorFlow
import json

role = get_execution_role()
sess = sage.Session()

WORK_DIRECTORY = 'local/'+model_name+'/input/data/training'
data_location = sess.upload_data(WORK_DIRECTORY, key_prefix='data')
print(data_location)

conf_file='local/'+model_name+'/input/config/hyperparameters.json'
```

4.4 Test Procedures and cases:-



Data source — Connect data source

AwsDataCatalog

Database

algo_data

Filter tables and views...

Tables (2) — Create table

- hist_data_daily
  - dt (string)
  - sym (string)
  - open (double)
  - high (double)
  - low (double)
  - close (double)
  - vol (double)
- hist_data_intraday

Views (0) — Create view

You have not created any views. To create a view, run a query and click "Create view from query"

New query 1

```
select * from hist_data_daily limit 10
```

Run query | Save as | Create | (Run time: 0.46 seconds, Data scanned: 201.92 KB) | Format query | Clear

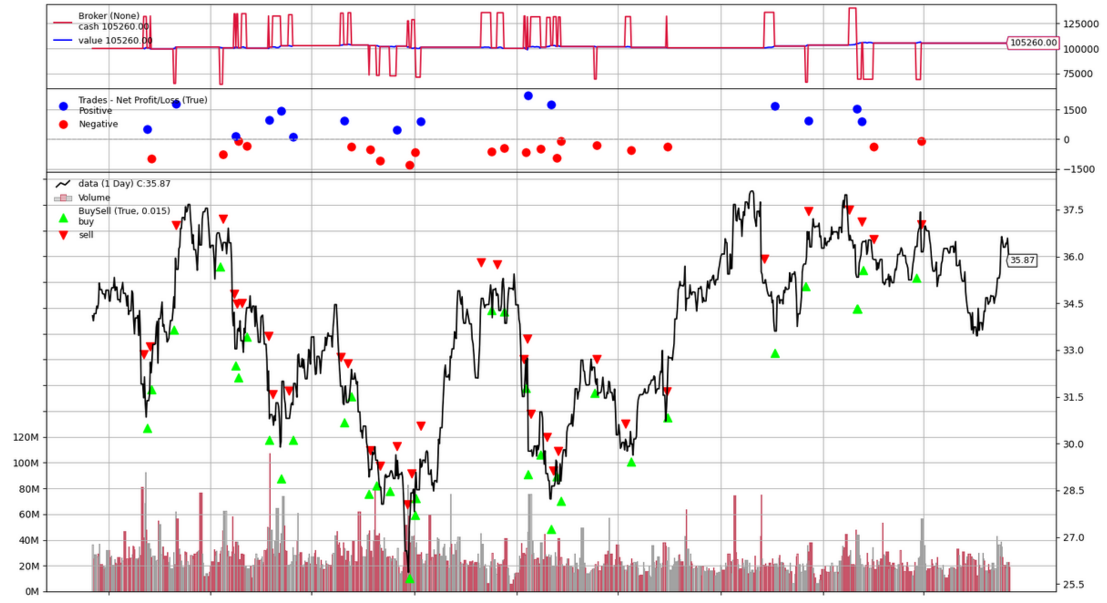
Use Ctrl + Enter to run query, Ctrl + Space to autocomplete

Results

	dt	sym	open	high	low	close	vol
1	2000-01-03	JNJ	93.13	93.75	91.69	92.19	2321200.0
2	2000-01-04	JNJ	91.13	91.56	88.75	88.81	3911100.0
3	2000-01-05	JNJ	88.81	90.81	88.44	90.19	4114800.0
4	2000-01-06	JNJ	90.31	93.94	90.13	92.56	2678200.0
5	2000-01-07	JNJ	94.13	96.94	93.63	96.5	4931900.0
6	2000-01-10	JNJ	96.69	96.94	94.06	94.06	2464300.0
7	2000-01-11	JNJ	93.0	94.94	92.94	94.06	3076500.0
8	2000-01-12	JNJ	93.88	95.06	93.19	93.5	2581600.0
9	2000-01-13	JNJ	93.94	96.0	93.06	93.38	2329600.0
10	2000-01-14	JNJ	93.94	94.38	92.81	93.69	2702400.0

```
In [8]: from IPython.display import Image
        Image(filename='local/'+algo_name+'/model/chart.png')
```

Out[8]:



4.5 **Sample Program code:-**

Command Kafka:-

```
wget https://downloads.apache.org/kafka/3.3.1/kafka_2.12-3.3.1.tgz
tar -xvf kafka_2.12-3.3.1.tgz
```

```
-----------------------
java -version
sudo yum install java-1.8.0-openjdk
java -version
cd kafka_2.12-3.3.1
```

Start Zoo-keeper:

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

Open another window to start kafka
But first ssh to to your ec2 machine as done above

Start Kafka-server:

```
----------------------------------------
```

Duplicate the session & enter in a new console --

```
export KAFKA_HEAP_OPTS="-Xmx256M -Xms128M"
cd kafka_2.12-3.3.1
bin/kafka-server-start.sh config/server.properties
```

It is pointing to private server , change server.properties so that it can run in public IP

To do this , you can follow any of the 2 approaches shared belwo --
Do a "sudo nano config/server.properties" - change ADVERTISED_LISTENERS to public ip of the EC2 instance

Create the topic:

```
-----------------------------
```

Duplicate the session & enter in a new console --

```
cd kafka_2.12-3.3.1
bin/kafka-topics.sh --create --topic demo_testing2 --bootstrap-server {Put the Public IP of your EC2 Instance:9092} --replication-factor 1 --partitions 1
```

Start Producer:

```
bin/kafka-console-producer.sh --topic demo_testing2 --bootstrap-server {Put the Public IP of your EC2 Instance:9092}
```

Start Consumer:

Duplicate the session & enter in a new console --

```
cd kafka_2.12-3.3.1
bin/kafka-console-consumer.sh --topic demo_testing2 --bootstrap-server {Put the Public IP of your EC2 Instance:9092}
```

4.5.2Kafka producer:-

```
pip install kafka-python
```

In [ ]:

```
import pandas as pd
from kafka import KafkaProducer
from time import sleep
from json import dumps
import json
```

In [ ]:

```
producer = KafkaProducer(bootstrap_servers=[':9092'], #change ip here
                         value_serializer=lambda x:
                         dumps(x).encode('utf-8'))
```

In [ ]:

```
producer.send('demo_test', value={'surnasdasdame':'parasdasdmar'})
```

In [ ]:

```
df = pd.read_csv("data/indexProcessed.csv")
```

In [ ]:

```
df.head()
```

In [ ]:

```
while True:
    dict_stock = df.sample(1).to_dict(orient="records")[0]
    producer.send('demo_test', value=dict_stock)
    sleep(1)
```

In [ ]:

```
producer.flush() #clear data from kafka server
```

4.5.3 Kafka Consumer:-

```
from kafka import KafkaConsumer
from time import sleep
from json import dumps,loads
import json
from s3fs import S3FileSystem
```

In [ ]:

```
consumer = KafkaConsumer(
    'demo_test',
     bootstrap_servers=[':9092'], #add your IP here
    value_deserializer=lambda x: loads(x.decode('utf-8')))
```

In [ ]:

```
# for c in consumer:
#     print(c.value)
```

In [ ]:

```
s3 = S3FileSystem()
```

In [ ]:

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

In [ ]:

4.6 Limitations and Bibliography

- **Limitations:**

- Integration Complexity: The system may face challenges in integrating with certain legacy systems or proprietary data sources, potentially limiting the scope of data analysis.
- Resource Constraints: Limitations in available resources, such as computing power or storage capacity, could affect the system's scalability and performance under heavy loads.
- Data Accuracy: Despite advanced analytics techniques, the accuracy of predictions and insights derived from stock market data may be influenced by factors such as data quality and market volatility.

- **Bibliography:**

- White, T. (2019). "Streaming Systems: The What, Where, When, and How of Large-Scale Data Processing." Manning Publications.
- Shukla, A., Sharma, M., & Sinha, R. (2020). "Apache Kafka Cookbook: Over 100 practical recipes on using distributed enterprise messaging to handle real-time data." Packet Publishing.
- Garg, N. (2018). "Mastering Apache Kafka: The Complete Guide to Building and Deploying Scalable and Fault-Tolerant Real-Time Streaming Applications." Packt Publishing.
- AWS Documentation. (n.d.). Retrieved from https://aws.amazon.com/documentation/
- Apache Flink Documentation. (n.d.). Retrieved from https://flink.apache.org/documentation/
- Apache Kafka Documentation. (n.d.). Retrieved from https://kafka.apache.org/documentation/