Final Project Part 3: Final Report Submission

Team 3
Panshul Saraswat
Nikhil Nagendra
Raj Mehta
Vyshnavi Gokavarapu
Rutuja Lohakare

Problem Statement:
In professional basketball, particularly within the NBA, real-time data analytics are pivotal for enhancing game analysis, optimizing player performance, and boosting fan engagement. Despite the potential of live data, its effective utilization is often hampered by the complexity of processing and visualizing this data in real-time. Our project accesses live NBA data using the nba_api, which provides a rich set of game statistics through endpoints like https://nba-prod-us-east-1-mediaops-stats.s3.amazonaws.com/NBA/liveData/scoreboard/todaysScoreboard_00.json. The API outputs data in JSON format, encapsulating game details such as scores, player stats, and team performance metrics across various game periods.

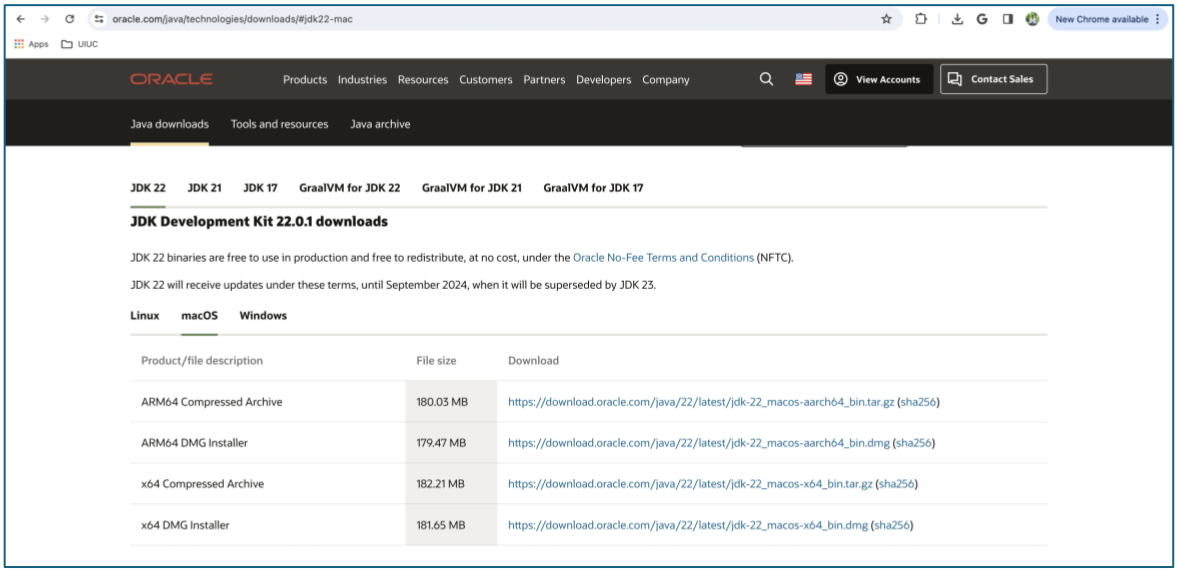
However, challenges persist in swiftly ingesting, processing, and displaying these data points. The structure of the data includes nested elements under keys like games, homeTeam, and awayTeam, detailing real-time updates like game scores, individual player performances, and game status. The goal of our proposed data pipeline—integrating Apache Kafka, Apache Spark, InfluxDB, and Grafana—is to streamline these processes. By setting up Kafka for efficient data ingestion, using Spark for real-time data analytics, storing processed data in InfluxDB, and visualizing insights through Grafana, we aim to facilitate quick, informed decision-making for coaches and teams, and enhance the interactive experience for fans. This pipeline promises to revolutionize how stakeholders interact with live game data, turning raw stats into actionable insights instantaneously.

Task 1: Data Ingestion

Apache Kafka is the platform used for handling real-time data feeds. It is a distributed event streaming platform capable of handling trillions of events a day.

Apache Kafka Requires Java to run, as Kafka itself is a Java-based system.

Install Java:
When setting up a data pipeline involving Apache Kafka, Java is a necessary dependency. Go to Oracle's official Java download page. Choose the version of Java Development Kit (JDK) you need. For most users working with Kafka and Spark, JDK 22 or the latest version will be suitable. Run the installer and check if the java has been installed or not from the terminal.



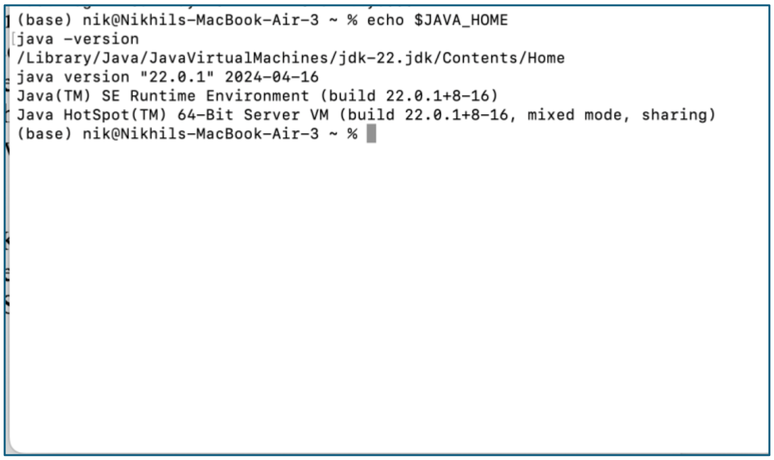
JDK Development Kit 22.0.1 downloads

JDK 22 binaries are free to use in production and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions (NFTC).

JDK 22 will receive updates under these terms, until September 2024, when it will be superseded by JDK 23.

Product/file description	File size	Download
ARM64 Compressed Archive	180.03 MB	https://download.oracle.com/java/22/latest/jdk-22_macos-aarch64_bin.tar.gz (sha256)
ARM64 DMG Installer	179.47 MB	https://download.oracle.com/java/22/latest/jdk-22_macos-aarch64_bin.dmg (sha256)
x64 Compressed Archive	182.21 MB	https://download.oracle.com/java/22/latest/jdk-22_macos-x64_bin.tar.gz (sha256)
x64 DMG Installer	181.65 MB	https://download.oracle.com/java/22/latest/jdk-22_macos-x64_bin.dmg (sha256)

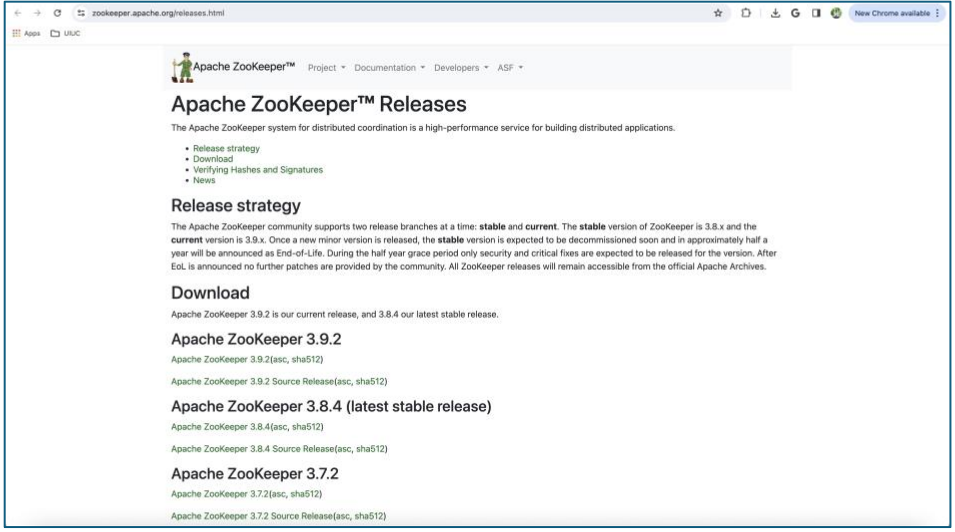
echo $JAVA_HOME
java -version



```
(base) nik@Nikhils-MacBook-Air-3 ~ % echo $JAVA_HOME
java -version
/Library/Java/JavaVirtualMachines/jdk-22.jdk/Contents/Home
java version "22.0.1" 2024-04-16
Java(TM) SE Runtime Environment (build 22.0.1+8-16)
Java HotSpot(TM) 64-Bit Server VM (build 22.0.1+8-16, mixed mode, sharing)
(base) nik@Nikhils-MacBook-Air-3 ~ %
```

Install Zookeeper:

Go to the Apache Zookeeper releases page to download the latest stable release of Zookeeper. Look for the binary archive, which typically has an extension .tar.gz. Extract the file and have it ready.



Apache ZooKeeper™ Releases

The Apache ZooKeeper system for distributed coordination is a high-performance service for building distributed applications.

- Release strategy
- Download
- Verifying Hashes and Signatures
- News

Release strategy

The Apache ZooKeeper community supports two release branches at a time: **stable** and **current**. The **stable** version of ZooKeeper is 3.8.x and the **current** version is 3.9.x. Once a new minor version is released, the **stable** version is expected to be decommissioned soon and in approximately half a year will be announced as End-of-Life. During the half year grace period only security and critical fixes are expected to be released for the version. After EoL is announced no further patches are provided by the community. All ZooKeeper releases will remain accessible from the official Apache Archives.

Download

Apache ZooKeeper 3.9.2 is our current release, and 3.8.4 our latest stable release.

Apache ZooKeeper 3.9.2

Apache ZooKeeper 3.9.2(asc, sha512)

Apache ZooKeeper 3.9.2 Source Release(asc, sha512)

Apache ZooKeeper 3.8.4 (latest stable release)

Apache ZooKeeper 3.8.4(asc, sha512)

Apache ZooKeeper 3.8.4 Source Release(asc, sha512)

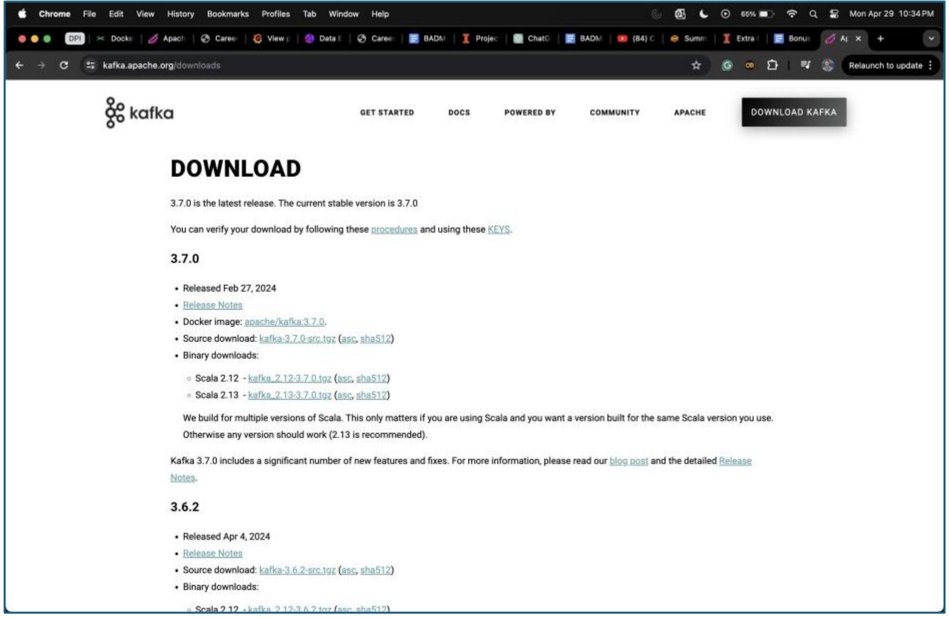
Apache ZooKeeper 3.7.2

Apache ZooKeeper 3.7.2(asc, sha512)

Apache ZooKeeper 3.7.2 Source Release(asc, sha512)

Install Kafka:

Download Apache Kafka binary file Scala 2.13from the official website.



DOWNLOAD

3.7.0 is the latest release. The current stable version is 3.7.0

You can verify your download by following these procedures and using these KEYS.

3.7.0

- Released Feb 27, 2024
- Release Notes
- Docker image: apache/kafka:3.7.0.
- Source download: kafka-3.7.0-src.tgz (asc, sha512)
- Binary downloads:
  - Scala 2.12 - kafka_2.12-3.7.0.tgz (asc, sha512)
  - Scala 2.13 - kafka_2.13-3.7.0.tgz (asc, sha512)

  We build for multiple versions of Scala. This only matters if you are using Scala and you want a version built for the same Scala version you use. Otherwise any version should work (2.13 is recommended).

Kafka 3.7.0 includes a significant number of new features and fixes. For more information, please read our blog post and the detailed Release Notes.

3.6.2

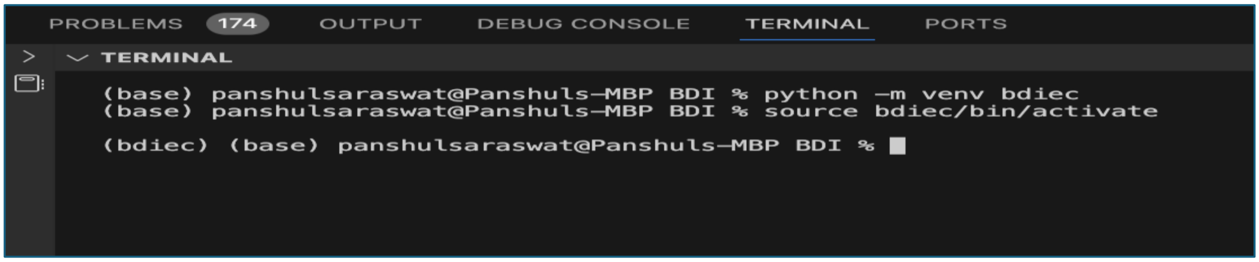
- Released Apr 4, 2024
- Release Notes
- Source download: kafka-3.6.2-src.tgz (asc, sha512)
- Binary downloads:
  - Scala 2.12 - kafka_2.12-3.6.2.tgz (asc, sha512)

Once the file is downloaded place the file in a folder. Extract the file in the same folder by giving the following command on the terminal.

```
tar -xzvf kafka_2.13-3.7.0.tar.gz
```

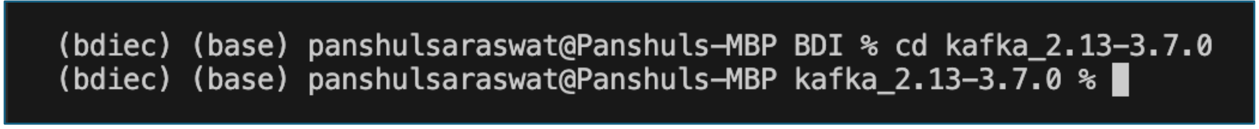
Kafka Setup:

Next objective is to create a virtual environment (venv) so that the Kafka server can be run in it.



```
(base) panshulsaraswat@Panshuls-MBP BDI % python -m venv bdiec
(base) panshulsaraswat@Panshuls-MBP BDI % source bdiec/bin/activate

(bdiec) (base) panshulsaraswat@Panshuls-MBP BDI %
```

After the successful creation of the environment venv navigate to the extracted Kafka folder in the venv.

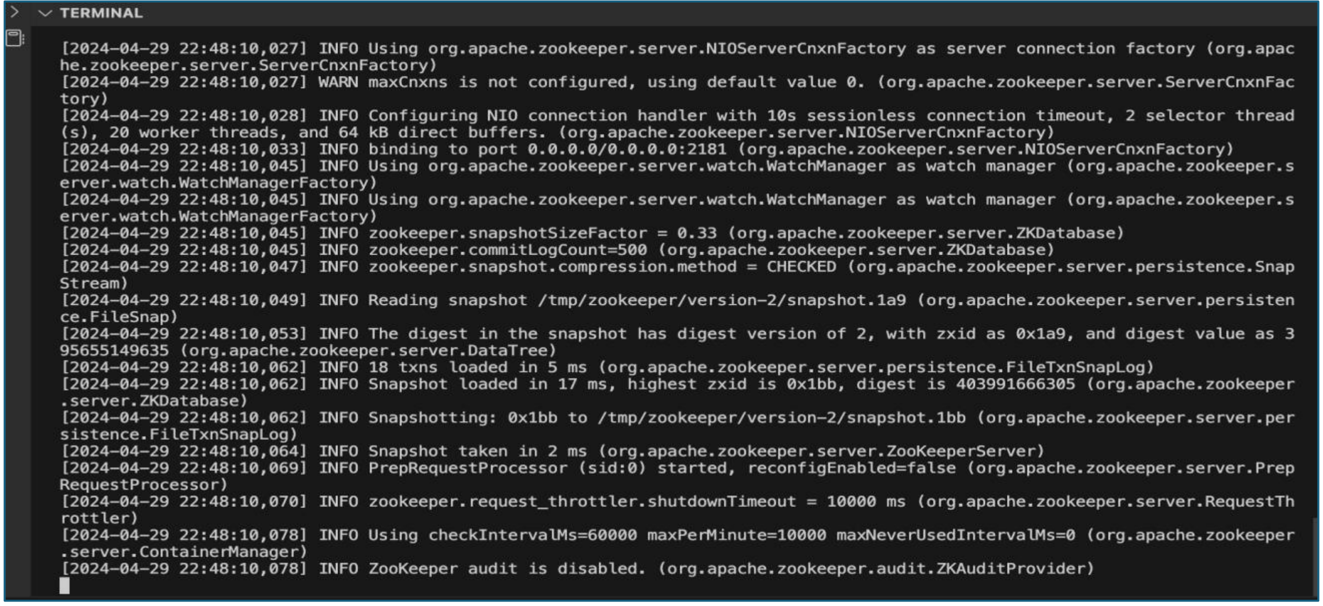


```
(bdiec) (base) panshulsaraswat@Panshuls-MBP BDI % cd kafka_2.13-3.7.0
(bdiec) (base) panshulsaraswat@Panshuls-MBP kafka_2.13-3.7.0 %
```

Now we run the below script to launch the zookeeper which is required to run Kafka.

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

After the above command is run in the terminal successfully, below results will be displayed indicating the zookeeper service is running



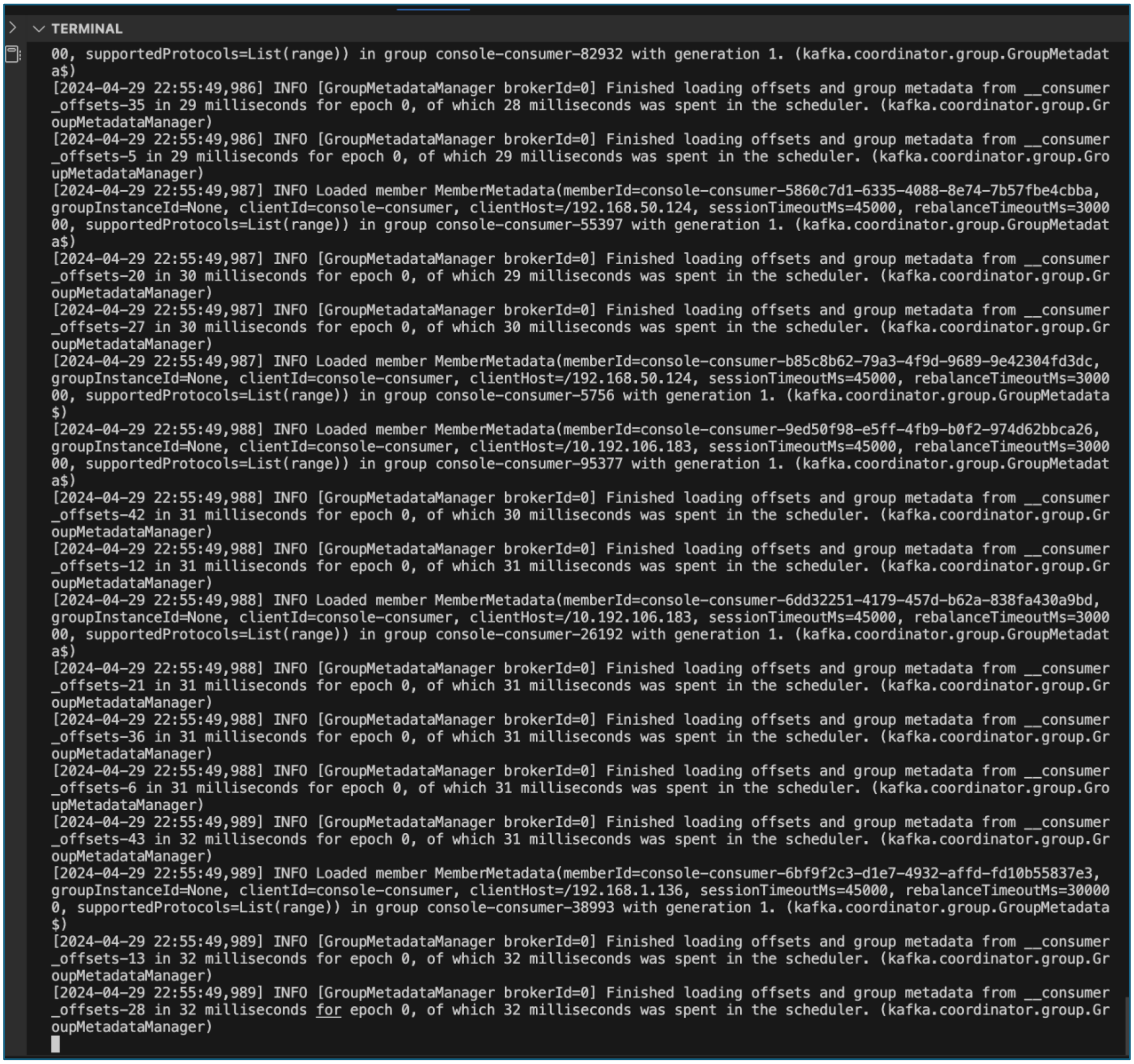
```
[2024-04-29 22:48:10,027] INFO Using org.apache.zookeeper.server.NIOServerCnxnFactory as server connection factory (org.apac
he.zookeeper.server.ServerCnxnFactory)
[2024-04-29 22:48:10,027] WARN maxCnxns is not configured, using default value 0. (org.apache.zookeeper.server.ServerCnxnFac
tory)
[2024-04-29 22:48:10,028] INFO Configuring NIO connection handler with 10s sessionless connection timeout, 2 selector thread
(s), 20 worker threads, and 64 kB direct buffers. (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2024-04-29 22:48:10,033] INFO binding to port 0.0.0.0/0.0.0.0:2181 (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2024-04-29 22:48:10,045] INFO Using org.apache.zookeeper.server.watch.WatchManager as watch manager (org.apache.zookeeper.s
erver.watch.WatchManagerFactory)
[2024-04-29 22:48:10,045] INFO Using org.apache.zookeeper.server.watch.WatchManager as watch manager (org.apache.zookeeper.s
erver.watch.WatchManagerFactory)
[2024-04-29 22:48:10,045] INFO zookeeper.snapshotSizeFactor = 0.33 (org.apache.zookeeper.server.ZKDatabase)
[2024-04-29 22:48:10,045] INFO zookeeper.commitLogCount=500 (org.apache.zookeeper.server.ZKDatabase)
[2024-04-29 22:48:10,047] INFO zookeeper.snapshot.compression.method = CHECKED (org.apache.zookeeper.server.persistence.Snap
Stream)
[2024-04-29 22:48:10,049] INFO Reading snapshot /tmp/zookeeper/version-2/snapshot.1a9 (org.apache.zookeeper.server.persisten
ce.FileSnap)
[2024-04-29 22:48:10,053] INFO The digest in the snapshot has digest version of 2, with zxid as 0x1a9, and digest value as 3
95655149635 (org.apache.zookeeper.server.DataTree)
[2024-04-29 22:48:10,062] INFO 18 txns loaded in 5 ms (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
[2024-04-29 22:48:10,062] INFO Snapshot loaded in 17 ms, highest zxid is 0x1bb, digest is 403991666305 (org.apache.zookeeper
.server.ZKDatabase)
[2024-04-29 22:48:10,062] INFO Snapshotting: 0x1bb to /tmp/zookeeper/version-2/snapshot.1bb (org.apache.zookeeper.server.per
sistence.FileTxnSnapLog)
[2024-04-29 22:48:10,064] INFO Snapshot taken in 2 ms (org.apache.zookeeper.server.ZooKeeperServer)
[2024-04-29 22:48:10,069] INFO PrepRequestProcessor (sid:0) started, reconfigEnabled=false (org.apache.zookeeper.server.Prep
RequestProcessor)
[2024-04-29 22:48:10,070] INFO zookeeper.request_throttler.shutdownTimeout = 10000 ms (org.apache.zookeeper.server.RequestTh
rottler)
[2024-04-29 22:48:10,078] INFO Using checkIntervalMs=60000 maxPerMinute=10000 maxNeverUsedIntervalMs=0 (org.apache.zookeeper
.server.ContainerManager)
[2024-04-29 22:48:10,078] INFO ZooKeeper audit is disabled. (org.apache.zookeeper.audit.ZKAuditProvider)
```

Note: Do not terminate the zookeeper terminal as we need to keep this service running

Next step is run a kafka server, open another terminal and navigate to extracted Kafka directory and give the below command to run a script which will launch the Kafka server.

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

Once the command is run successfully, below results will be shown indicating Kafka server is now setup and is now ready for data to be sent to it.



```
00, supportedProtocols=List(range)) in group console-consumer-82932 with generation 1. (kafka.coordinator.group.GroupMetadat
a$)
[2024-04-29 22:55:49,986] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer
_offsets-35 in 29 milliseconds for epoch 0, of which 28 milliseconds was spent in the scheduler. (kafka.coordinator.group.Gr
oupMetadataManager)
[2024-04-29 22:55:49,986] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer
_offsets-5 in 29 milliseconds for epoch 0, of which 29 milliseconds was spent in the scheduler. (kafka.coordinator.group.Gro
upMetadataManager)
[2024-04-29 22:55:49,987] INFO Loaded member MemberMetadata(memberId=console-consumer-5860c7d1-6335-4088-8e74-7b57fbe4cbba,
groupInstanceId=None, clientId=console-consumer, clientHost=/192.168.50.124, sessionTimeoutMs=45000, rebalanceTimeoutMs=3000
00, supportedProtocols=List(range)) in group console-consumer-55397 with generation 1. (kafka.coordinator.group.GroupMetadat
a$)
[2024-04-29 22:55:49,987] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer
_offsets-20 in 30 milliseconds for epoch 0, of which 29 milliseconds was spent in the scheduler. (kafka.coordinator.group.Gr
oupMetadataManager)
[2024-04-29 22:55:49,987] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer
_offsets-27 in 30 milliseconds for epoch 0, of which 30 milliseconds was spent in the scheduler. (kafka.coordinator.group.Gr
oupMetadataManager)
[2024-04-29 22:55:49,987] INFO Loaded member MemberMetadata(memberId=console-consumer-b85c8b62-79a3-4f9d-9689-9e42304fd3dc,
groupInstanceId=None, clientId=console-consumer, clientHost=/192.168.50.124, sessionTimeoutMs=45000, rebalanceTimeoutMs=3000
00, supportedProtocols=List(range)) in group console-consumer-5756 with generation 1. (kafka.coordinator.group.GroupMetadata
$)
[2024-04-29 22:55:49,988] INFO Loaded member MemberMetadata(memberId=console-consumer-9ed50f98-e5ff-4fb9-b0f2-974d62bbca26,
groupInstanceId=None, clientId=console-consumer, clientHost=/10.192.106.183, sessionTimeoutMs=45000, rebalanceTimeoutMs=3000
00, supportedProtocols=List(range)) in group console-consumer-95377 with generation 1. (kafka.coordinator.group.GroupMetadat
a$)
[2024-04-29 22:55:49,988] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer
_offsets-42 in 31 milliseconds for epoch 0, of which 30 milliseconds was spent in the scheduler. (kafka.coordinator.group.Gr
oupMetadataManager)
[2024-04-29 22:55:49,988] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer
_offsets-12 in 31 milliseconds for epoch 0, of which 31 milliseconds was spent in the scheduler. (kafka.coordinator.group.Gr
oupMetadataManager)
[2024-04-29 22:55:49,988] INFO Loaded member MemberMetadata(memberId=console-consumer-6dd32251-4179-457d-b62a-838fa430a9bd,
groupInstanceId=None, clientId=console-consumer, clientHost=/10.192.106.183, sessionTimeoutMs=45000, rebalanceTimeoutMs=3000
00, supportedProtocols=List(range)) in group console-consumer-26192 with generation 1. (kafka.coordinator.group.GroupMetadat
a$)
[2024-04-29 22:55:49,988] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer
_offsets-21 in 31 milliseconds for epoch 0, of which 31 milliseconds was spent in the scheduler. (kafka.coordinator.group.Gr
oupMetadataManager)
[2024-04-29 22:55:49,988] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer
_offsets-36 in 31 milliseconds for epoch 0, of which 31 milliseconds was spent in the scheduler. (kafka.coordinator.group.Gr
oupMetadataManager)
[2024-04-29 22:55:49,988] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer
_offsets-6 in 31 milliseconds for epoch 0, of which 31 milliseconds was spent in the scheduler. (kafka.coordinator.group.Gro
upMetadataManager)
[2024-04-29 22:55:49,989] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer
_offsets-43 in 32 milliseconds for epoch 0, of which 31 milliseconds was spent in the scheduler. (kafka.coordinator.group.Gr
oupMetadataManager)
[2024-04-29 22:55:49,989] INFO Loaded member MemberMetadata(memberId=console-consumer-6bf9f2c3-d1e7-4932-affd-fd10b55837e3,
groupInstanceId=None, clientId=console-consumer, clientHost=/192.168.1.136, sessionTimeoutMs=45000, rebalanceTimeoutMs=30000
0, supportedProtocols=List(range)) in group console-consumer-38993 with generation 1. (kafka.coordinator.group.GroupMetadata
$)
[2024-04-29 22:55:49,989] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer
_offsets-13 in 32 milliseconds for epoch 0, of which 32 milliseconds was spent in the scheduler. (kafka.coordinator.group.Gr
oupMetadataManager)
[2024-04-29 22:55:49,989] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer
_offsets-28 in 32 milliseconds for epoch 0, of which 32 milliseconds was spent in the scheduler. (kafka.coordinator.group.Gr
oupMetadataManager)
```

In an another terminal deactivate virtual environment venv, as this terminal will be later used to run a python file.

Create a python script in which you provide the data source and the Kafka topic for it to be written to.

The kafka-python library, which is a Kafka client for Python, allows for easy interaction with the Kafka system. This library can be installed using below command.

pip install kafka-python

Below is the script which used to send NBA data from NBA api. This code also contains a scheduler which ensures that the data is updated every 60 seconds in the Kafka topic/server. The scheduler can be removed as well depending on the use case.

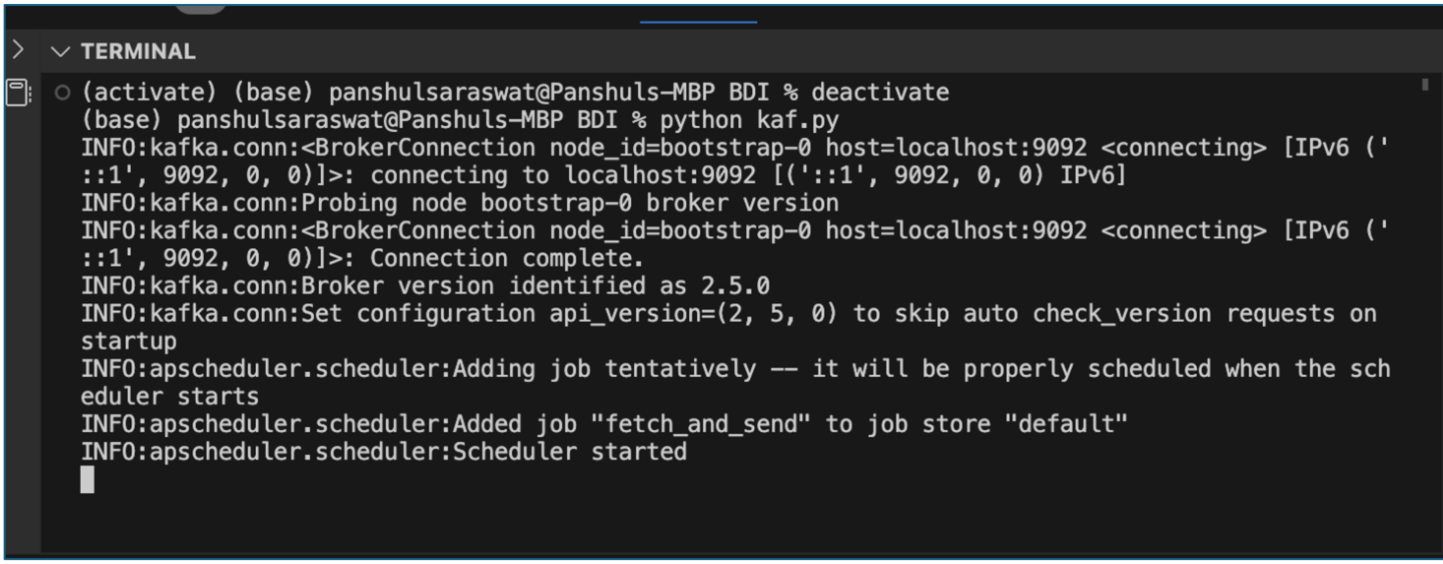
Code:

```
from kafka import KafkaProducer
import json
import time
from nba_api.live.nba.endpoints import scoreboard
from apscheduler.schedulers.blocking import BlockingScheduler
import logging

# Set up logging
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

# Kafka Configuration
kafka_broker = 'localhost:9092'
kafka_topic = 'nba-scoreboard-1' <topic name

# Create a Kafka producer
producer = KafkaProducer(
bootstrap_servers=[kafka_broker],
value_serializer=lambda x: json.dumps(x).encode('utf-8')
)

def fetch_nba_scores():
try:
# Fetch today's NBA scoreboard
games = scoreboard.ScoreBoard()
data = games.get_dict()  # Fetch the games data as a dictionary
return data
except Exception as e:
logger.error(f"Failed to fetch NBA scores: {e}")
return None

def send_data_to_kafka(data):
try:
if data:
producer.send(kafka_topic, data)
producer.flush()
logger.info("Data sent to Kafka successfully.")
else:
logger.info("No data to send.")
except Exception as e:
logger.error(f"Failed to send data to Kafka: {e}")

def fetch_and_send():
data = fetch_nba_scores()  # Fetch data from NBA API
send_data_to_kafka(data)   # Send the data to Kafka

def main():
scheduler = BlockingScheduler()
scheduler.add_job(fetch_and_send, 'interval', minutes=1)  # Adjust the interval as needed
try:
scheduler.start()
except (KeyboardInterrupt, SystemExit):
pass
```

```
if __name__ == "__main__":
main()
```

Save the python file and run in the terminal as shown below.

Python kaf.py

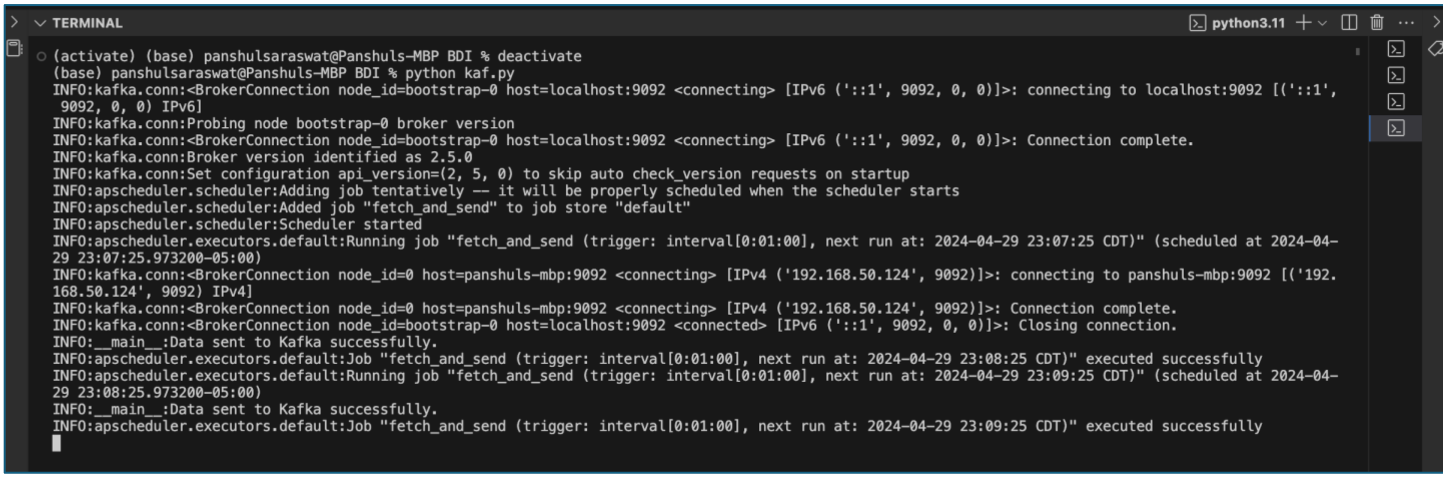


TERMINAL

```
○ (activate) (base) panshulsaraswat@Panshuls-MBP BDI % deactivate
  (base) panshulsaraswat@Panshuls-MBP BDI % python kaf.py
  INFO:kafka.conn:<BrokerConnection node_id=bootstrap-0 host=localhost:9092 <connecting> [IPv6 ('
  ::1', 9092, 0, 0)]>: connecting to localhost:9092 [('::1', 9092, 0, 0) IPv6]
  INFO:kafka.conn:Probing node bootstrap-0 broker version
  INFO:kafka.conn:<BrokerConnection node_id=bootstrap-0 host=localhost:9092 <connecting> [IPv6 ('
  ::1', 9092, 0, 0)]>: Connection complete.
  INFO:kafka.conn:Broker version identified as 2.5.0
  INFO:kafka.conn:Set configuration api_version=(2, 5, 0) to skip auto check_version requests on
  startup
  INFO:apscheduler.scheduler:Adding job tentatively -- it will be properly scheduled when the sch
  eduler starts
  INFO:apscheduler.scheduler:Added job "fetch_and_send" to job store "default"
  INFO:apscheduler.scheduler:Scheduler started
  █
```

Once the data is sent successfully you can see a print statement in the terminal:

INFO:__main__:Data sent to Kafka successfully.

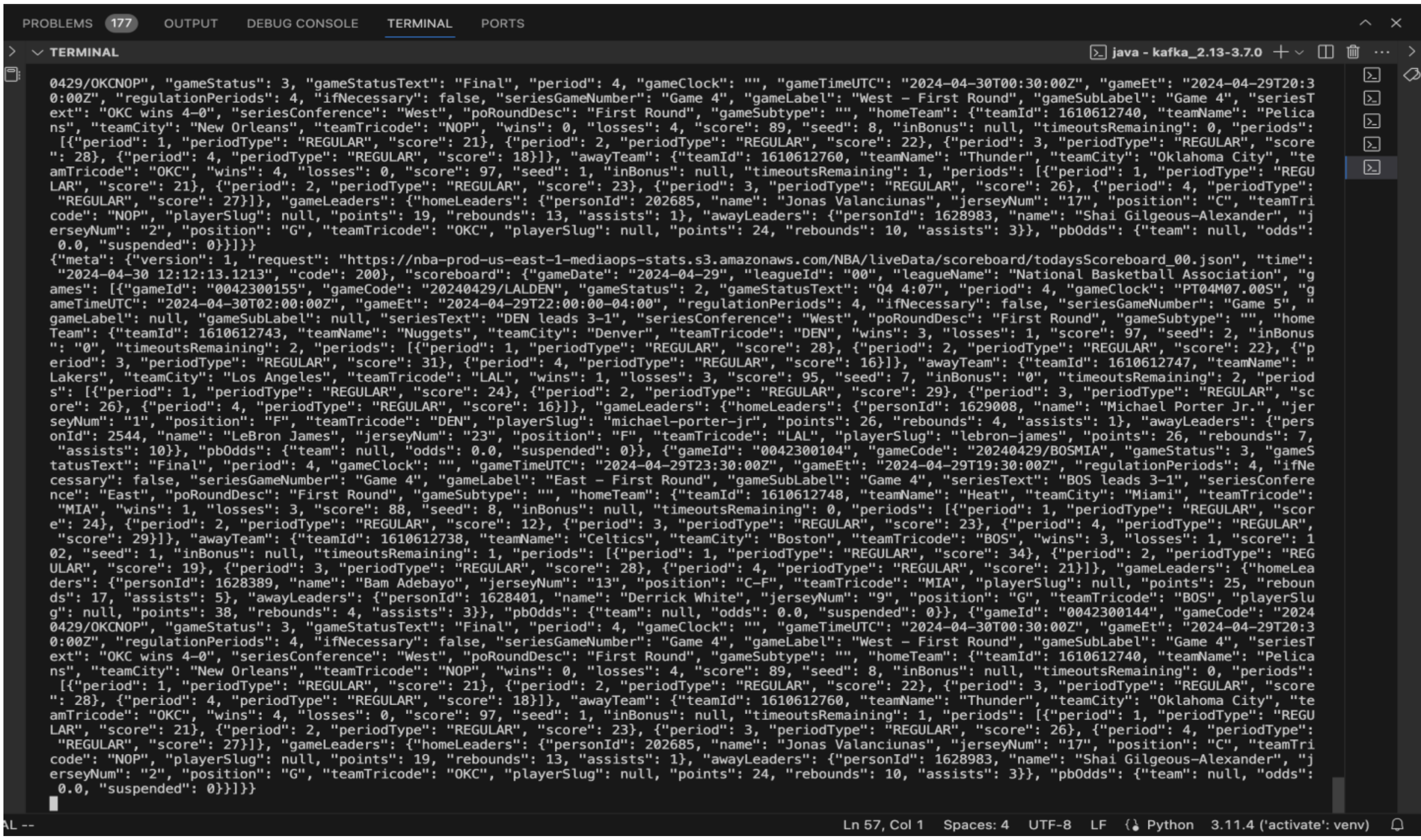


```
○ (activate) (base) panshulsaraswat@Panshuls-MBP BDI % deactivate
  (base) panshulsaraswat@Panshuls-MBP BDI % python kaf.py
  INFO:kafka.conn:<BrokerConnection node_id=bootstrap-0 host=localhost:9092 <connecting> [IPv6 ('::1', 9092, 0, 0)]>: connecting to localhost:9092 [('::1',
   9092, 0, 0) IPv6]
  INFO:kafka.conn:Probing node bootstrap-0 broker version
  INFO:kafka.conn:<BrokerConnection node_id=bootstrap-0 host=localhost:9092 <connecting> [IPv6 ('::1', 9092, 0, 0)]>: Connection complete.
  INFO:kafka.conn:Broker version identified as 2.5.0
  INFO:kafka.conn:Set configuration api_version=(2, 5, 0) to skip auto check_version requests on startup
  INFO:apscheduler.scheduler:Adding job tentatively -- it will be properly scheduled when the scheduler starts
  INFO:apscheduler.scheduler:Added job "fetch_and_send" to job store "default"
  INFO:apscheduler.scheduler:Scheduler started
  INFO:apscheduler.executors.default:Running job "fetch_and_send (trigger: interval[0:01:00], next run at: 2024-04-29 23:07:25 CDT)" (scheduled at 2024-04-
  29 23:07:25.973200-05:00)
  INFO:kafka.conn:<BrokerConnection node_id=0 host=panshuls-mbp:9092 <connecting> [IPv4 ('192.168.50.124', 9092)]>: connecting to panshuls-mbp:9092 [('192.
  168.50.124', 9092) IPv4]
  INFO:kafka.conn:<BrokerConnection node_id=0 host=panshuls-mbp:9092 <connecting> [IPv4 ('192.168.50.124', 9092)]>: Connection complete.
  INFO:kafka.conn:<BrokerConnection node_id=bootstrap-0 host=localhost:9092 <connected> [IPv6 ('::1', 9092, 0, 0)]>: Closing connection.
  INFO:__main__:Data sent to Kafka successfully.
  INFO:apscheduler.executors.default:Job "fetch_and_send (trigger: interval[0:01:00], next run at: 2024-04-29 23:08:25 CDT)" executed successfully
  INFO:apscheduler.executors.default:Running job "fetch_and_send (trigger: interval[0:01:00], next run at: 2024-04-29 23:09:25 CDT)" (scheduled at 2024-04-
  29 23:08:25.973200-05:00)
  INFO:__main__:Data sent to Kafka successfully.
  INFO:apscheduler.executors.default:Job "fetch_and_send (trigger: interval[0:01:00], next run at: 2024-04-29 23:09:25 CDT)" executed successfully
  █
```

Viewing the data that is sent to Kafka:
Open a new terminal and ensure that you are in the same venv and then navigate to the Kafka folder.
Use the below command to run a script that shows the data that is there in the Kafka topic from beginning.

bin/kafka-console-consumer.sh --topic <your_topic_name> --from-beginning --bootstrap-server localhost:9092

Below you can see a screenshot of the NBA data that was sent to Kafka.



```
0429/OKCNOP", "gameStatus": 3, "gameStatusText": "Final", "period": 4, "gameClock": "", "gameTimeUTC": "2024-04-30T00:30:00Z", "gameEt": "2024-04-29T20:3
0:00Z", "regulationPeriods": 4, "ifNecessary": false, "seriesGameNumber": "Game 4", "gameLabel": "West - First Round", "gameSubLabel": "Game 4", "seriesT
ext": "OKC wins 4-0", "seriesConference": "West", "poRoundDesc": "First Round", "gameSubtype": "", "homeTeam": {"teamId": 1610612740, "teamName": "Pelica
ns", "teamCity": "New Orleans", "teamTricode": "NOP", "wins": 0, "losses": 4, "score": 89, "seed": 8, "inBonus": null, "timeoutsRemaining": 0, "periods":
 [{"period": 1, "periodType": "REGULAR", "score": 21}, {"period": 2, "periodType": "REGULAR", "score": 22}, {"period": 3, "periodType": "REGULAR", "score
": 28}, {"period": 4, "periodType": "REGULAR", "score": 18}]}, "awayTeam": {"teamId": 1610612760, "teamName": "Thunder", "teamCity": "Oklahoma City", "te
amTricode": "OKC", "wins": 4, "losses": 0, "score": 97, "seed": 1, "inBonus": null, "timeoutsRemaining": 1, "periods": [{"period": 1, "periodType": "REGU
LAR", "score": 21}, {"period": 2, "periodType": "REGULAR", "score": 23}, {"period": 3, "periodType": "REGULAR", "score": 26}, {"period": 4, "periodType":
 "REGULAR", "score": 27}]}, "gameLeaders": {"homeLeaders": {"personId": 202685, "name": "Jonas Valanciunas", "jerseyNum": "17", "position": "C", "teamTri
code": "NOP", "playerSlug": null, "points": 19, "rebounds": 13, "assists": 1}, "awayLeaders": {"personId": 1628983, "name": "Shai Gilgeous-Alexander", "j
erseyNum": "2", "position": "G", "teamTricode": "OKC", "playerSlug": null, "points": 24, "rebounds": 10, "assists": 3}}, "pbOdds": {"team": null, "odds":
 0.0, "suspended": 0}}]}}
{"meta": {"version": 1, "request": "https://nba-prod-us-east-1-mediaops-stats.s3.amazonaws.com/NBA/liveData/scoreboard/todaysScoreboard_00.json", "time":
 "2024-04-30 12:12:13.1213", "code": 200}, "scoreboard": {"gameDate": "2024-04-29", "leagueId": "00", "leagueName": "National Basketball Association", "g
ames": [{"gameId": "0042300155", "gameCode": "20240429/LALDEN", "gameStatus": 2, "gameStatusText": "Q4 4:07", "period": 4, "gameClock": "PT04M07.00S", "g
ameTimeUTC": "2024-04-30T02:00:00Z", "gameEt": "2024-04-29T22:00:00-04:00", "regulationPeriods": 4, "ifNecessary": false, "seriesGameNumber": "Game 5", "
gameLabel": null, "gameSubLabel": null, "seriesText": "DEN leads 3-1", "seriesConference": "West", "poRoundDesc": "First Round", "gameSubtype": "", "home
Team": {"teamId": 1610612743, "teamName": "Nuggets", "teamCity": "Denver", "teamTricode": "DEN", "wins": 3, "losses": 1, "score": 97, "seed": 2, "inBonus
": "0", "timeoutsRemaining": 2, "periods": [{"period": 1, "periodType": "REGULAR", "score": 28}, {"period": 2, "periodType": "REGULAR", "score": 22}, {"p
eriod": 3, "periodType": "REGULAR", "score": 31}, {"period": 4, "periodType": "REGULAR", "score": 16}]}, "awayTeam": {"teamId": 1610612747, "teamName": "
Lakers", "teamCity": "Los Angeles", "teamTricode": "LAL", "wins": 1, "losses": 3, "score": 95, "seed": 7, "inBonus": "0", "timeoutsRemaining": 2, "period
s": [{"period": 1, "periodType": "REGULAR", "score": 24}, {"period": 2, "periodType": "REGULAR", "score": 29}, {"period": 3, "periodType": "REGULAR", "sc
ore": 26}, {"period": 4, "periodType": "REGULAR", "score": 16}]}, "gameLeaders": {"homeLeaders": {"personId": 1629008, "name": "Michael Porter Jr.", "jer
seyNum": "1", "position": "F", "teamTricode": "DEN", "playerSlug": "michael-porter-jr", "points": 26, "rebounds": 4, "assists": 1}, "awayLeaders": {"pers
onId": 2544, "name": "LeBron James", "jerseyNum": "23", "position": "F", "teamTricode": "LAL", "playerSlug": "lebron-james", "points": 26, "rebounds": 7,
 "assists": 10}}, "pbOdds": {"team": null, "odds": 0.0, "suspended": 0}}, {"gameId": "0042300104", "gameCode": "20240429/BOSMIA", "gameStatus": 3, "gameS
tatusText": "Final", "period": 4, "gameClock": "", "gameTimeUTC": "2024-04-29T23:30:00Z", "gameEt": "2024-04-29T19:30:00Z", "regulationPeriods": 4, "ifNe
cessary": false, "seriesGameNumber": "Game 4", "gameLabel": "East - First Round", "gameSubLabel": "Game 4", "seriesText": "BOS leads 3-1", "seriesConfere
nce": "East", "poRoundDesc": "First Round", "gameSubtype": "", "homeTeam": {"teamId": 1610612748, "teamName": "Heat", "teamCity": "Miami", "teamTricode":
 "MIA", "wins": 1, "losses": 3, "score": 88, "seed": 8, "inBonus": null, "timeoutsRemaining": 0, "periods": [{"period": 1, "periodType": "REGULAR", "scor
e": 24}, {"period": 2, "periodType": "REGULAR", "score": 12}, {"period": 3, "periodType": "REGULAR", "score": 23}, {"period": 4, "periodType": "REGULAR",
 "score": 29}]}, "awayTeam": {"teamId": 1610612738, "teamName": "Celtics", "teamCity": "Boston", "teamTricode": "BOS", "wins": 3, "losses": 1, "score": 1
02, "seed": 1, "inBonus": null, "timeoutsRemaining": 1, "periods": [{"period": 1, "periodType": "REGULAR", "score": 34}, {"period": 2, "periodType": "REG
ULAR", "score": 19}, {"period": 3, "periodType": "REGULAR", "score": 28}, {"period": 4, "periodType": "REGULAR", "score": 21}]}, "gameLeaders": {"homeLea
ders": {"personId": 1628389, "name": "Bam Adebayo", "jerseyNum": "13", "position": "C-F", "teamTricode": "MIA", "playerSlug": null, "points": 25, "reboun
ds": 17, "assists": 5}, "awayLeaders": {"personId": 1628401, "name": "Derrick White", "jerseyNum": "9", "position": "G", "teamTricode": "BOS", "playerSlu
g": null, "points": 38, "rebounds": 4, "assists": 3}}, "pbOdds": {"team": null, "odds": 0.0, "suspended": 0}}, {"gameId": "0042300144", "gameCode": "2024
0429/OKCNOP", "gameStatus": 3, "gameStatusText": "Final", "period": 4, "gameClock": "", "gameTimeUTC": "2024-04-30T00:30:00Z", "gameEt": "2024-04-29T20:3
0:00Z", "regulationPeriods": 4, "ifNecessary": false, "seriesGameNumber": "Game 4", "gameLabel": "West - First Round", "gameSubLabel": "Game 4", "seriesT
ext": "OKC wins 4-0", "seriesConference": "West", "poRoundDesc": "First Round", "gameSubtype": "", "homeTeam": {"teamId": 1610612740, "teamName": "Pelica
ns", "teamCity": "New Orleans", "teamTricode": "NOP", "wins": 0, "losses": 4, "score": 89, "seed": 8, "inBonus": null, "timeoutsRemaining": 0, "periods":
 [{"period": 1, "periodType": "REGULAR", "score": 21}, {"period": 2, "periodType": "REGULAR", "score": 22}, {"period": 3, "periodType": "REGULAR", "score
": 28}, {"period": 4, "periodType": "REGULAR", "score": 18}]}, "awayTeam": {"teamId": 1610612760, "teamName": "Thunder", "teamCity": "Oklahoma City", "te
amTricode": "OKC", "wins": 4, "losses": 0, "score": 97, "seed": 1, "inBonus": null, "timeoutsRemaining": 1, "periods": [{"period": 1, "periodType": "REGU
LAR", "score": 21}, {"period": 2, "periodType": "REGULAR", "score": 23}, {"period": 3, "periodType": "REGULAR", "score": 26}, {"period": 4, "periodType":
 "REGULAR", "score": 27}]}, "gameLeaders": {"homeLeaders": {"personId": 202685, "name": "Jonas Valanciunas", "jerseyNum": "17", "position": "C", "teamTri
code": "NOP", "playerSlug": null, "points": 19, "rebounds": 13, "assists": 1}, "awayLeaders": {"personId": 1628983, "name": "Shai Gilgeous-Alexander", "j
erseyNum": "2", "position": "G", "teamTricode": "OKC", "playerSlug": null, "points": 24, "rebounds": 10, "assists": 3}}, "pbOdds": {"team": null, "odds":
 0.0, "suspended": 0}}]}}
```

This Completes Apache Kafka setup.

Task 2: Data Processing, and Storing

Setup Apache Spark:

PySpark is used to process the data streaming from Kafka. It handles reading, transforming, and extracting useful information from the Kafka topics. The PySpark library in Python, installed from the below command.

pip install pyspark

Spark-Kafka connector allows PySpark to read from and write to Kafka topics directly. This is crucial for integrating Kafka streams with Spark processing.
Spark-SQL-Kafka-0-10 Package must be specified in your Spark session to connect Spark with Kafka. It's included with Spark but needs to be explicitly enabled with configuration options when initializing the Spark session:

.config("spark.jars.packages", "org.apache.spark:spark-sql-kafka-0-10_2.12:<spark_version>")

Replace <spark_version> with your actual Spark version.

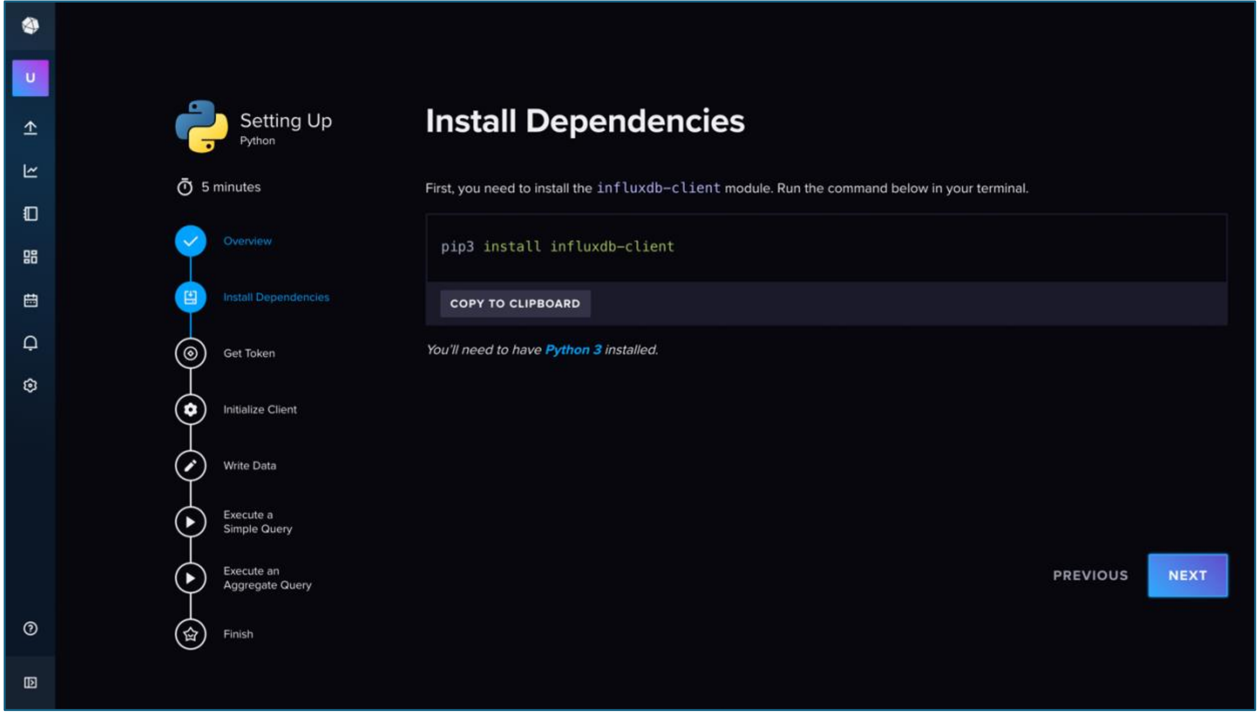
Setup Influx DB:

InfluxDB is a time-series database designed to handle high write and query loads. It is particularly useful in scenarios involving real-time analytics.
InfluxDB Client for Python: influxdb_client library, allows you to connect, write, and query data from an InfluxDB instance. This can be installed using following command.

pip install influxdb-client

Creating a bucket in InfluxDB can be done through the InfluxDB UI (User Interface). Open your web browser and navigate to the InfluxDB instance. The default URL is usually *http://localhost:8086*

Once Logged in we need to create and setup bucket in the influx db where the data is stored. As explained above install the dependency if not installed earlier.



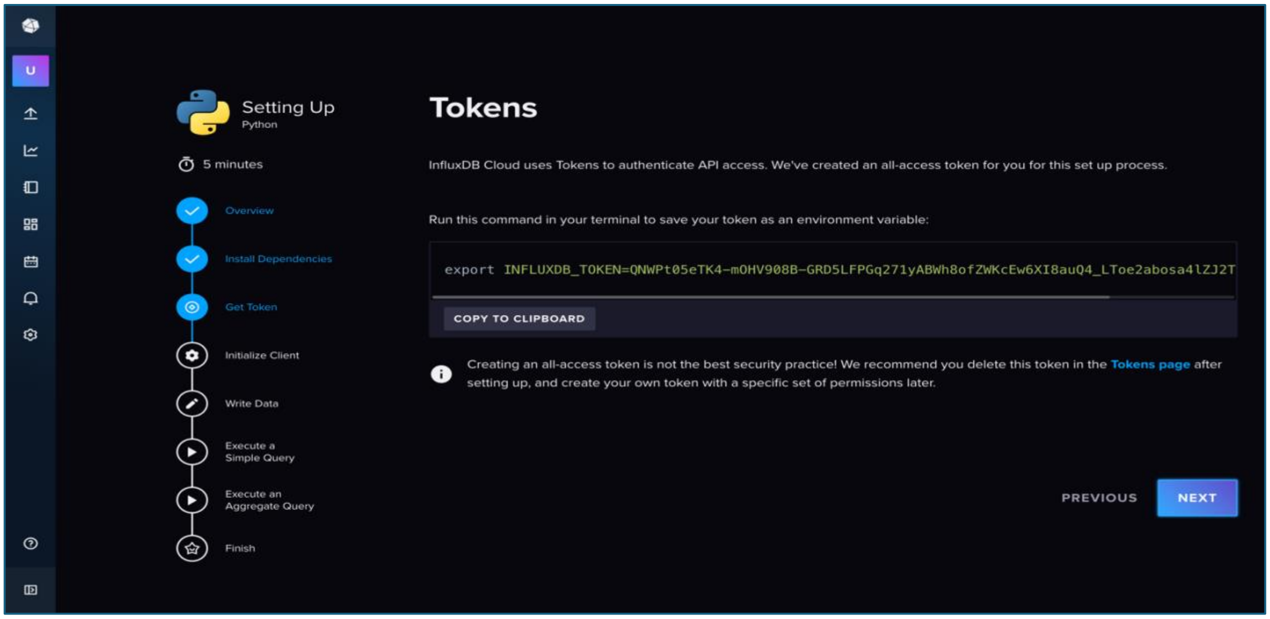
Install Dependencies

First, you need to install the influxdb-client module. Run the command below in your terminal.

```
pip3 install influxdb-client
```

You'll need to have Python 3 installed.

Create and save the token, later to be used to find the bucket



Tokens

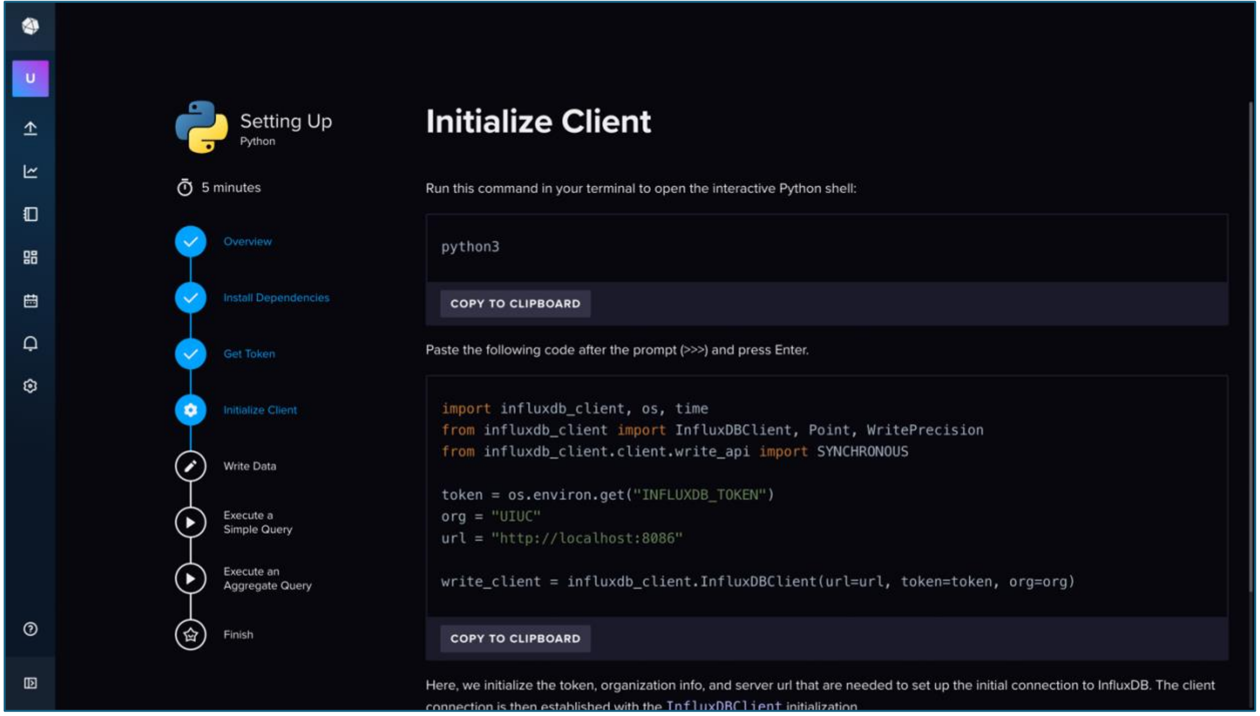
InfluxDB Cloud uses Tokens to authenticate API access. We've created an all-access token for you for this set up process.

Run this command in your terminal to save your token as an environment variable:

```
export INFLUXDB_TOKEN=QNWPt05eTK4-mOHV908B-GRD5LFPGq271yABWh8ofZWKcEw6XI8auQ4_LToe2abosa4lZJ2T
```

Creating an all-access token is not the best security practice! We recommend you delete this token in the Tokens page after setting up, and create your own token with a specific set of permissions later.

Now we create organization and set up server url as they are needed to have an initial connection, here 'UIUC' is the name given to organization.



Initialize Client

Run this command in your terminal to open the interactive Python shell:

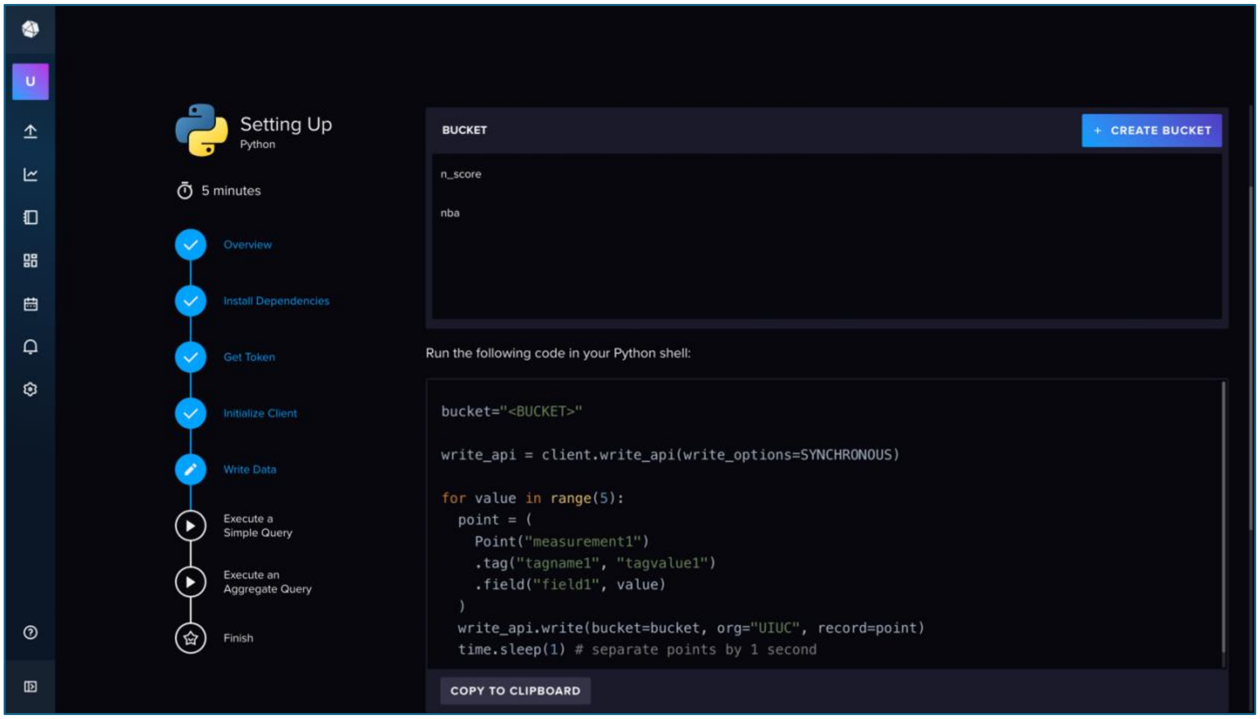
```
python3
```

Paste the following code after the prompt (>>>) and press Enter.

```
import influxdb_client, os, time
from influxdb_client import InfluxDBClient, Point, WritePrecision
from influxdb_client.client.write_api import SYNCHRONOUS

token = os.environ.get("INFLUXDB_TOKEN")
org = "UIUC"
url = "http://localhost:8086"

write_client = influxdb_client.InfluxDBClient(url=url, token=token, org=org)
```

Here, we initialize the token, organization info, and server url that are needed to set up the initial connection to InfluxDB. The client connection is then established with the InfluxDBClient initialization.

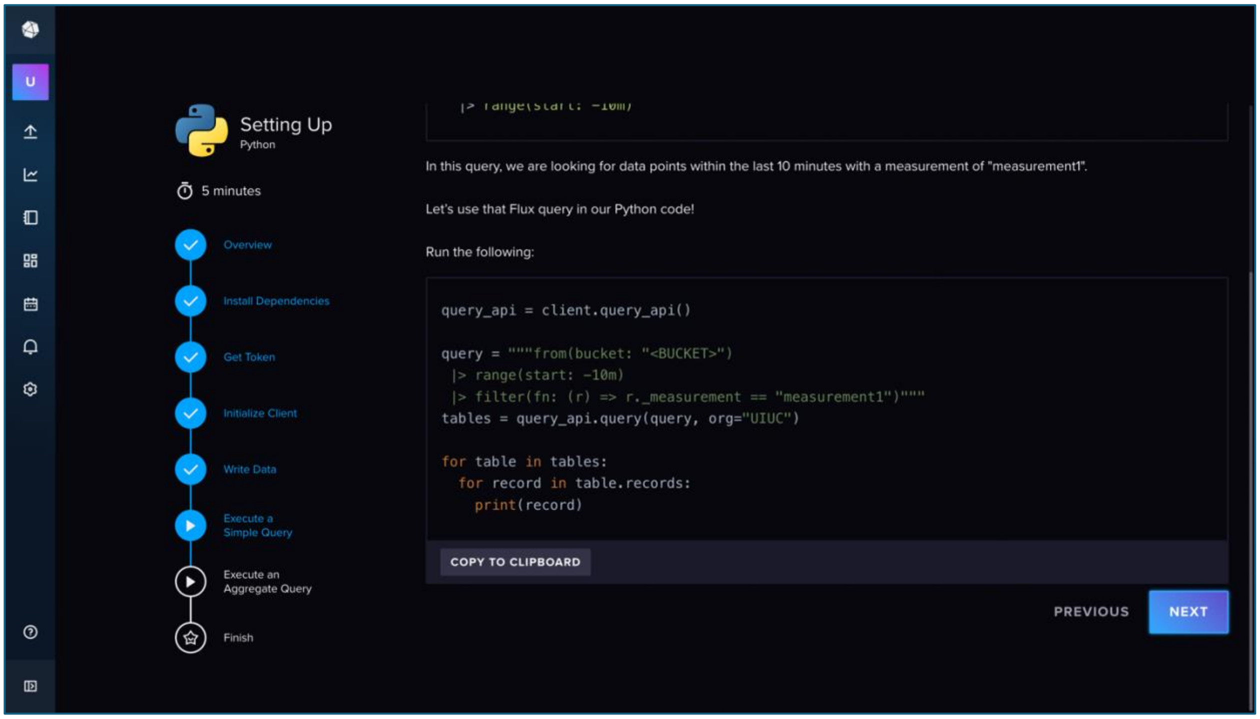
Click on create bucket, below are the 2 buckets created in this process, once done test data can be sent to check, and below images shows the same setup



BUCKET
n_score
nba

Run the following code in your Python shell:

```
bucket="<BUCKET>"

write_api = client.write_api(write_options=SYNCHRONOUS)

for value in range(5):
  point = (
    Point("measurement1")
    .tag("tagname1", "tagvalue1")
    .field("field1", value)
  )
  write_api.write(bucket=bucket, org="UIUC", record=point)
  time.sleep(1) # separate points by 1 second
```



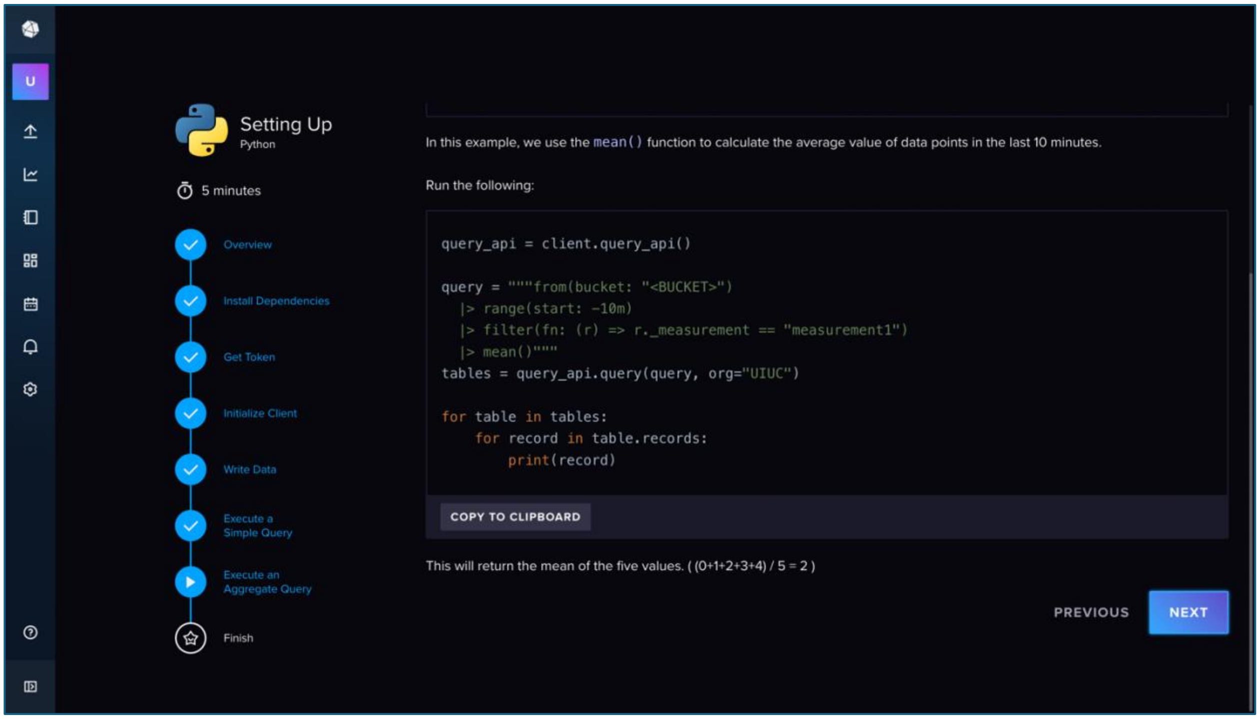
```
  |> range(start: -10m)
```

In this query, we are looking for data points within the last 10 minutes with a measurement of "measurement1".

Let's use that Flux query in our Python code!

Run the following:

```
query_api = client.query_api()

query = """from(bucket: "<BUCKET>")
 |> range(start: -10m)
 |> filter(fn: (r) => r._measurement == "measurement1")"""
tables = query_api.query(query, org="UIUC")

for table in tables:
  for record in table.records:
    print(record)
```

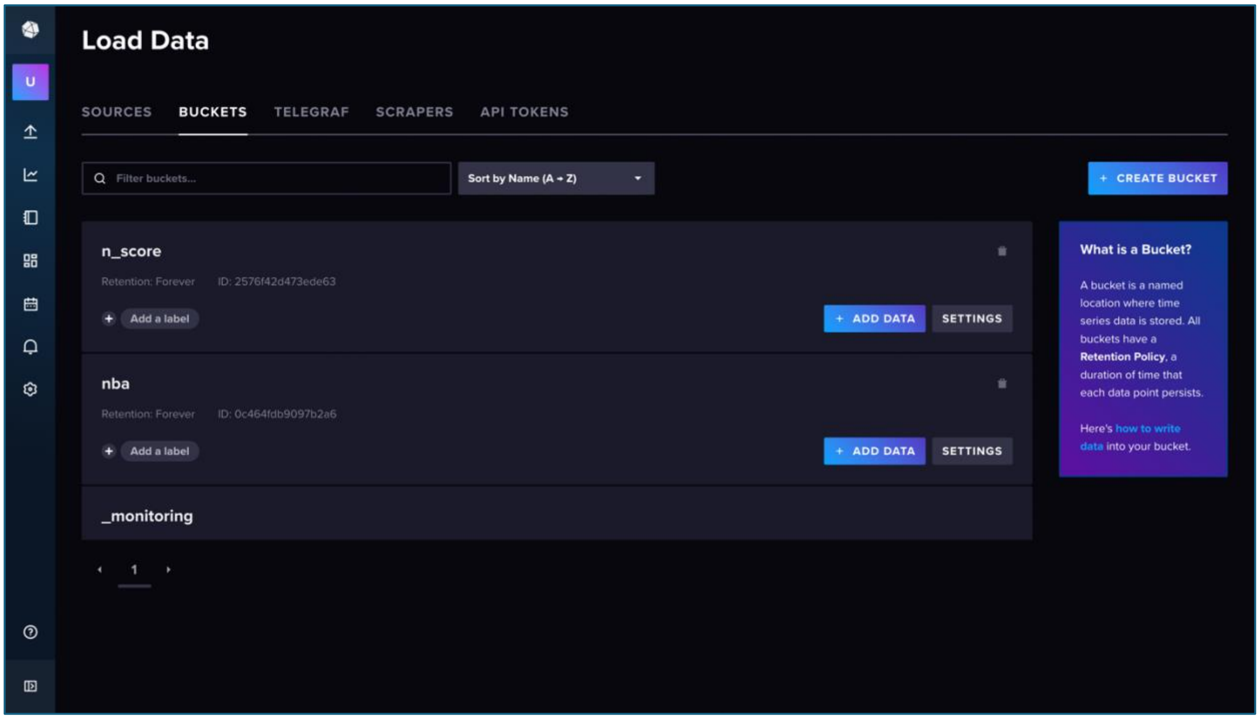


In this example, we use the `mean()` function to calculate the average value of data points in the last 10 minutes.

Run the following:

```
query_api = client.query_api()

query = """from(bucket: "<BUCKET>")
   |> range(start: -10m)
   |> filter(fn: (r) => r._measurement == "measurement1")
   |> mean()"""
tables = query_api.query(query, org="UIUC")

for table in tables:
    for record in table.records:
        print(record)
```

This will return the mean of the five values. ((0+1+2+3+4) / 5 = 2)



n_score

Retention: Forever ID: 2576f42d473ede63

nba

Retention: Forever ID: 0c464fdb9097b2a6

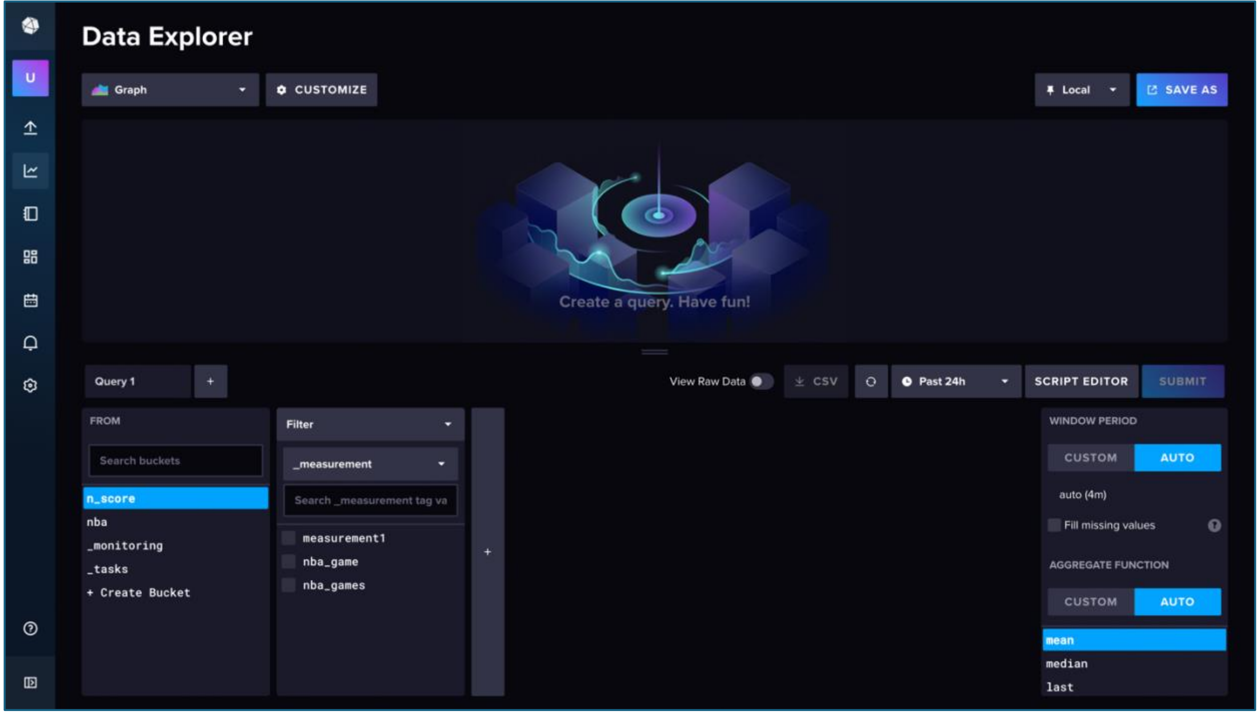
_monitoring

What is a Bucket?

A bucket is a named location where time series data is stored. All buckets have a **Retention Policy**, a duration of time that each data point persists.

Here's how to write data into your bucket.

In the data explorer, flux query can be run to retrieve the data.



From this set up below are the things to be noted:
influxdb_token = "CGJpFEeqwj1XiZe0HXFyT38_UOCuxrX-qEhIx7RyA4KY8L5HlCN-pvvdghxStvXuoN56RmrIfW0N8JGyU2B1MA=="

influxdb_org = "UIUC"

influxdb_bucket = "n_score"

influxdb_url = http://localhost:8086

Python Shell Code:
Once all the above set up is done, create a new python script which consumes the data from our kafka topic consumer and defines schema for our data to convert from json to tabular format and send it to our influx db at every minute interval, below is the code:

[Yesterday 9:23 PM] Saraswat, Panshul

Spark code :

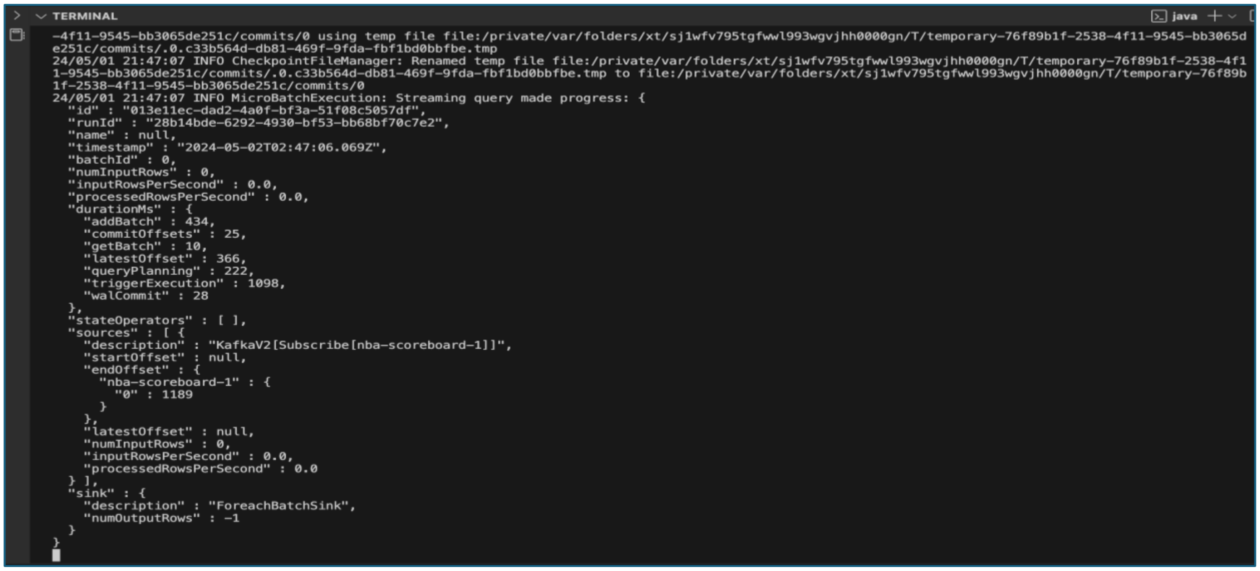
```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, from_json, explode
from pyspark.sql.types import *
from influxdb_client import InfluxDBClient, Point, WriteOptions

# InfluxDB Client Configuration
url = "http://localhost:8086"
token = "CGJpFEeqwj1XiZe0HXFyT38_UOCuxrX-qEhIx7RyA4KY8L5HlCN-pvvdghxStvXuoN56RmrIfW0N8JGyU2B1MA=="
org = "UIUC"
bucket = "n_score"
client = InfluxDBClient(url=url, token=token, org=org)
write_api = client.write_api(write_options=WriteOptions(batch_size=1000, flush_interval=10_000))

# Define the schema based on the nested JSON structure
schema = StructType([
    StructField("meta", StructType([
        StructField("version", IntegerType()),
        StructField("request", StringType()),
        StructField("time", StringType()),
        StructField("code", IntegerType())
    ])),
    StructField("scoreboard", StructType([
        StructField("gameDate", StringType()),
        StructField("leagueId", StringType()),
        StructField("leagueName", StringType()),
        StructField("games", ArrayType(StructType([
            StructField("gameId", StringType()),
            StructField("gameCode", StringType()),
            StructField("gameStatus", IntegerType()),
            StructField("gameStatusText", StringType()),
            StructField("period", IntegerType()),
            StructField("gameClock", StringType()),
            StructField("gameTimeUTC", StringType()),
            StructField("gameEt", StringType()),
            StructField("regulationPeriods", IntegerType()),
            StructField("ifNecessary", BooleanType()),
            StructField("seriesGameNumber", StringType()),
            StructField("gameLabel", StringType()),
            StructField("gameSubLabel", StringType()),
            StructField("seriesText", StringType()),
            StructField("seriesConference", StringType()),
StructField("poRoundDesc", StringType()),
StructField("gameSubtype", StringType()),
StructField("homeTeam", StructType([
    StructField("teamId", IntegerType()),
    StructField("teamName", StringType()),
    StructField("teamCity", StringType()),
    StructField("teamTricode", StringType()),
    StructField("wins", IntegerType()),
    StructField("losses", IntegerType()),
    StructField("score", IntegerType()),
    StructField("seed", IntegerType()),
    StructField("inBonus", StringType()),
    StructField("timeoutsRemaining", IntegerType()),
    StructField("periods", ArrayType(StructType([
        StructField("period", IntegerType()),
        StructField("periodType", StringType()),
        StructField("score", IntegerType())
    ])))
])),
StructField("awayTeam", StructType([
    StructField("teamId", IntegerType()),
    StructField("teamName", StringType()),
    StructField("teamCity", StringType()),
    StructField("teamTricode", StringType()),
    StructField("wins", IntegerType()),
    StructField("losses", IntegerType()),
    StructField("score", IntegerType()),
    StructField("seed", IntegerType()),
    StructField("inBonus", StringType()),
    StructField("timeoutsRemaining", IntegerType()),
    StructField("periods", ArrayType(StructType([
        StructField("period", IntegerType()),
        StructField("periodType", StringType()),
        StructField("score", IntegerType())
    ])))
])),
StructField("gameLeaders", StructType([
    StructField("homeLeaders", StructType([
        StructField("personId", IntegerType()),
        StructField("name", StringType()),
        StructField("jerseyNum", StringType()),
        StructField("position", StringType()),
        StructField("teamTricode", StringType()),
        StructField("playerSlug", StringType()),
        StructField("points", IntegerType()),
        StructField("rebounds", IntegerType()),
        StructField("assists", IntegerType())
    ])),
    StructField("awayLeaders", StructType([
        StructField("personId", IntegerType()),
        StructField("name", StringType()),
        StructField("jerseyNum", StringType()),
        StructField("position", StringType()),
        StructField("teamTricode", StringType()),
        StructField("playerSlug", StringType()),
        StructField("points", IntegerType()),
                    StructField("rebounds", IntegerType()),
                    StructField("assists", IntegerType())
                ]))
            ])),
            StructField("pbOdds", StructType([
                StructField("team", StringType()),
                StructField("odds", DoubleType()),
                StructField("suspended", IntegerType())
            ]))
        ])))
    ]))
])



# Initialize Spark Session
spark = SparkSession.builder.appName("NBA Scores Kafka Consumer").getOrCreate()



# Read from Kafka
df = spark.readStream.format("kafka") \
    .option("kafka.bootstrap.servers", "localhost:9092") \
    .option("subscribe", "nba-scoreboard-1") \
    .load()



# Select message value and convert from bytes to string
df = df.selectExpr("CAST(value AS STRING)")



# Parse the JSON string using the defined schema
parsed_df = df.select(from_json(col("value"), schema).alias("data"))



# Explode the games array into individual game records
games_df = parsed_df.select(explode(col("data.scoreboard.games")).alias("game"))



# Select relevant fields for processing
flattened_df = games_df.select(
    col("game.gameId").alias("gameId"),
    col("game.gameCode").alias("gameCode"),
    col("game.gameStatusText").alias("gameStatusText"),
    col("game.homeTeam.teamName").alias("homeTeamName"),
    col("game.homeTeam.score").alias("homeScore"),
    col("game.awayTeam.teamName").alias("awayTeamName"),
    col("game.awayTeam.score").alias("awayScore"),
    col("game.period").alias("period"),
    col("game.gameTimeUTC").alias("gameTimeUTC")
)
```

```
# Function to write each batch of DataFrame to InfluxDB
def write_to_influx(batch_df, epoch_id):
    points = []
    for row in batch_df.collect():
        point = Point("nba_games") \
            .tag("gameId", row.gameId) \
            .tag("gameCode", row.gameCode) \
            .field("gameStatusText", row.gameStatusText) \
            .field("homeTeamName", row.homeTeamName) \
            .field("homeScore", int(row.homeScore)) \
            .field("awayTeamName", row.awayTeamName) \
            .field("awayScore", int(row.awayScore)) \
            .field("period", int(row.period)) \
            .time(row.gameTimeUTC)
        points.append(point)
    write_api.write(bucket=bucket, record=points)
```

```
# Write the stream to InfluxDB
query = flattened_df.writeStream.outputMode("append") \
    .foreachBatch(write_to_influx) \
    .start()

query.awaitTermination()
```

Save the python file, and in new terminal run the spark job from the below command.

spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.5.1 spark.py



```
-4f11-9545-bb3065de251c/commits/0 using temp file file:/private/var/folders/xt/sj1wfv795tgfwwl993wgvjhh0000gn/T/temporary-76f89b1f-2538-4f11-9545-bb3065d
e251c/commits/.0.c33b564d-db81-469f-9fda-fbf1bd0bbfbe.tmp
24/05/01 21:47:07 INFO CheckpointFileManager: Renamed temp file file:/private/var/folders/xt/sj1wfv795tgfwwl993wgvjhh0000gn/T/temporary-76f89b1f-2538-4f1
1-9545-bb3065de251c/commits/.0.c33b564d-db81-469f-9fda-fbf1bd0bbfbe.tmp to file:/private/var/folders/xt/sj1wfv795tgfwwl993wgvjhh0000gn/T/temporary-76f89b
1f-2538-4f11-9545-bb3065de251c/commits/0
24/05/01 21:47:07 INFO MicroBatchExecution: Streaming query made progress: {
  "id" : "013e11ec-dad2-4a0f-bf3a-51f08c5057df",
  "runId" : "28b14bde-6292-4930-bf53-bb68bf70c7e2",
  "name" : null,
  "timestamp" : "2024-05-02T02:47:06.069Z",
  "batchId" : 0,
  "numInputRows" : 0,
  "inputRowsPerSecond" : 0.0,
  "processedRowsPerSecond" : 0.0,
  "durationMs" : {
    "addBatch" : 434,
    "commitOffsets" : 25,
    "getBatch" : 10,
    "latestOffset" : 366,
    "queryPlanning" : 222,
    "triggerExecution" : 1098,
    "walCommit" : 28
  },
  "stateOperators" : [ ],
  "sources" : [ {
    "description" : "KafkaV2[Subscribe[nba-scoreboard-1]]",
    "startOffset" : null,
    "endOffset" : {
      "nba-scoreboard-1" : {
        "0" : 1189
      }
    },
    "latestOffset" : null,
    "numInputRows" : 0,
    "inputRowsPerSecond" : 0.0,
    "processedRowsPerSecond" : 0.0
  } ],
  "sink" : {
    "description" : "ForeachBatchSink",
    "numOutputRows" : -1
  }
}
```

Here while sending the data, we look for numoutputRows= -1 which means that the data was successfully sent to InfluxDB

Task 3: Visualize with Grafana

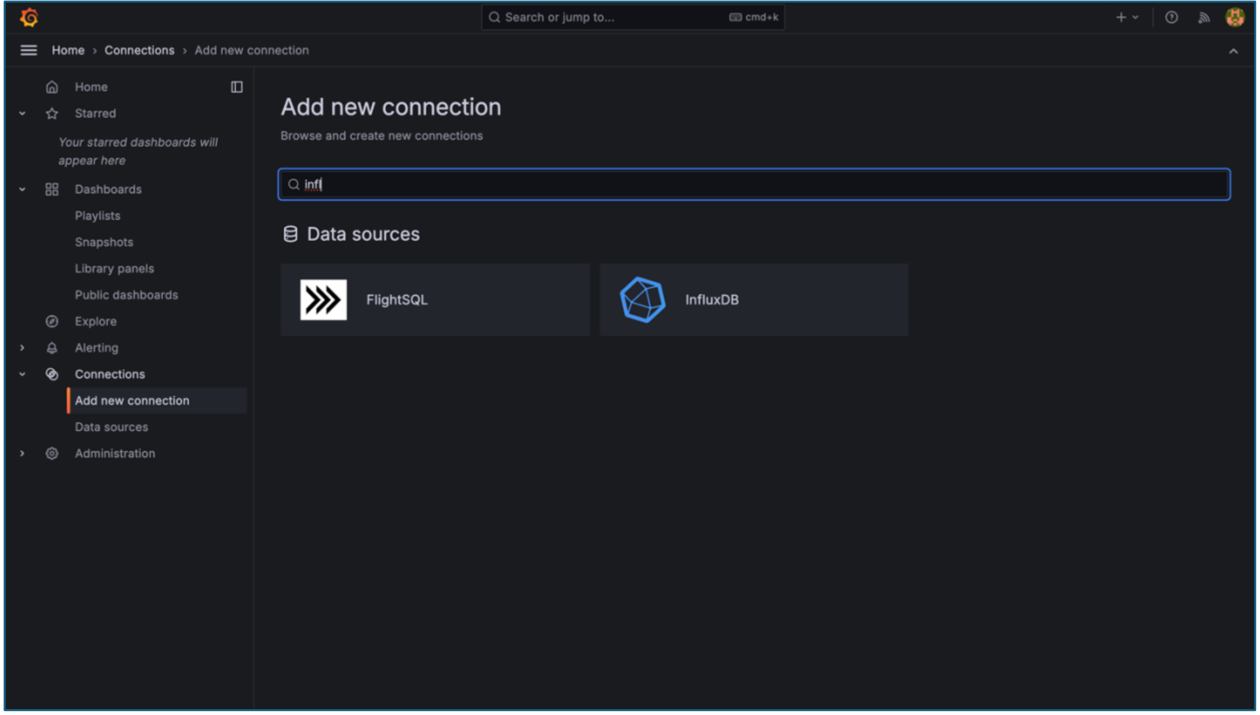
Setup Grafana:
Install and run Grafana by running below 2 commands in terminal

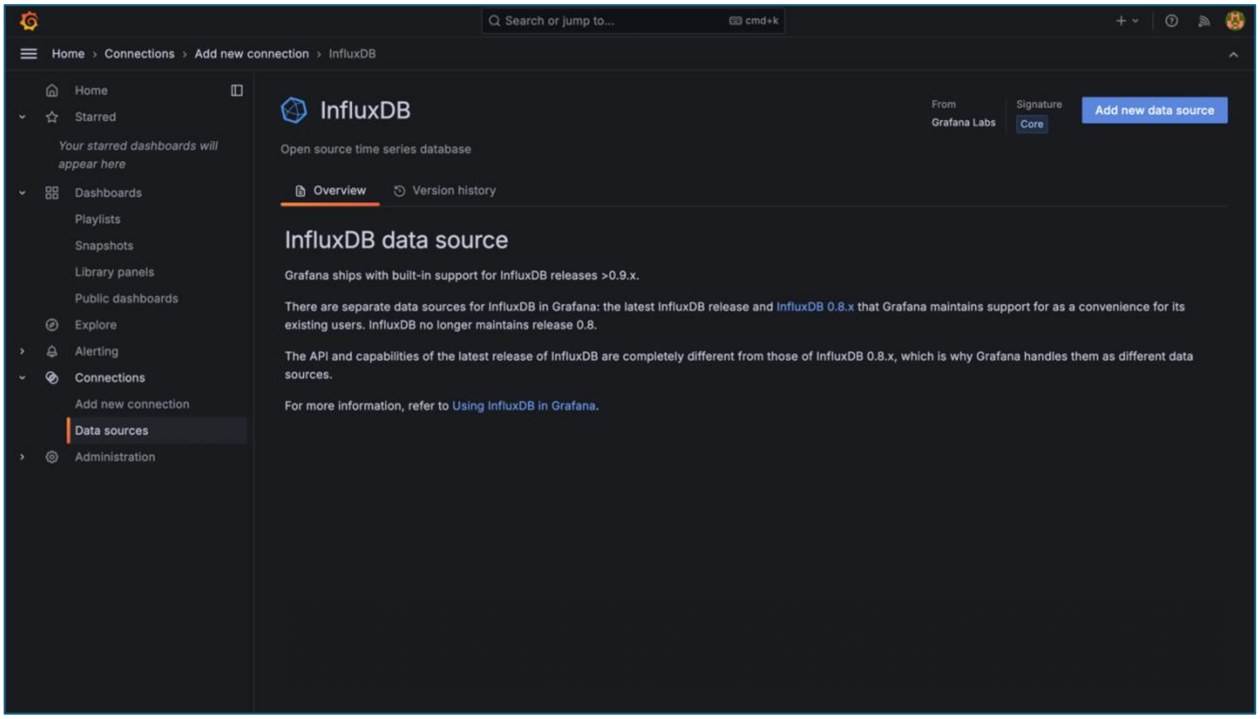
brew install grafana
brew services start grafana

Once Grafana is installed and running, open your web browser and navigate to http://localhost:3000/. The default port for Grafana is 3000.

The default login credentials are Username: admin and Password: admin. Upon first login, you will be prompted to change the password.

After logging in, you can configure data sources (like InfluxDB, MySQL, PostgreSQL, etc.) that Grafana will use to pull data for visualization. Go to Configuration > Data Sources > Add data source and select the type of source you want to add. Fill in the required connection details and settings specific to that source, as the below images showcase.





InfluxDB

Open source time series database

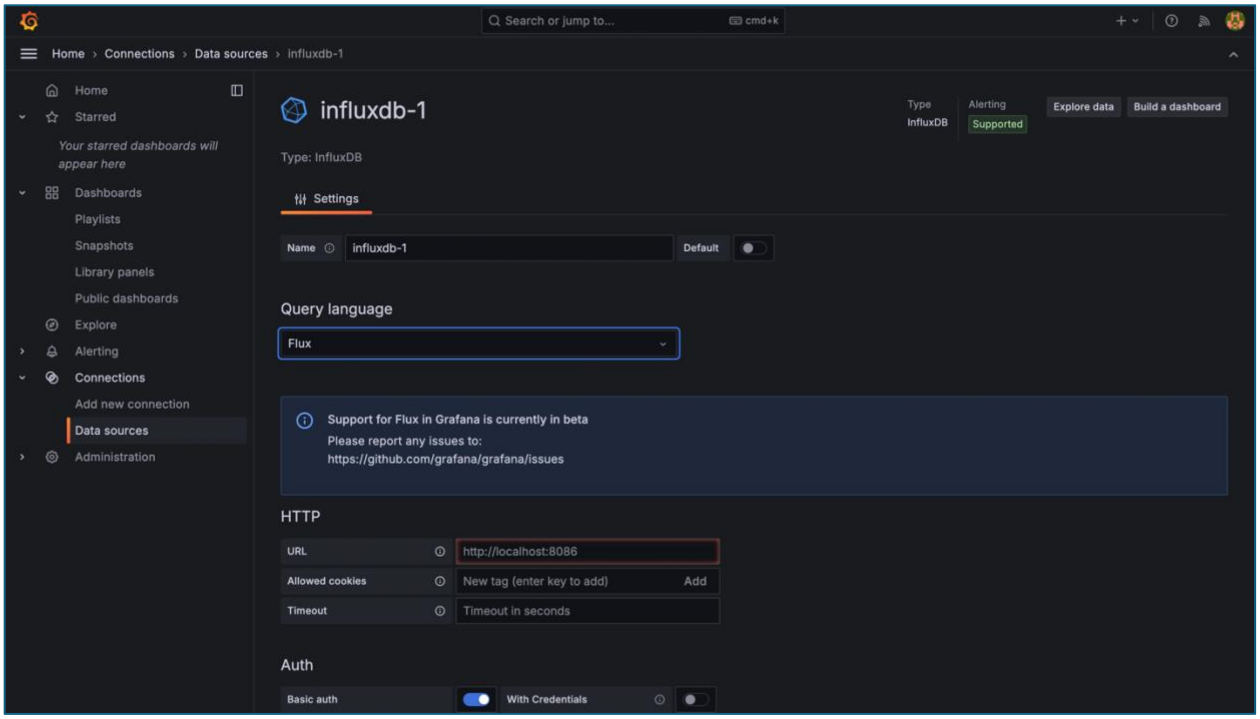
InfluxDB data source

Grafana ships with built-in support for InfluxDB releases >0.9.x.

There are separate data sources for InfluxDB in Grafana: the latest InfluxDB release and InfluxDB 0.8.x that Grafana maintains support for as a convenience for its existing users. InfluxDB no longer maintains release 0.8.

The API and capabilities of the latest release of InfluxDB are completely different from those of InfluxDB 0.8.x, which is why Grafana handles them as different data sources.

For more information, refer to Using InfluxDB in Grafana.



influxdb-1

Type: InfluxDB

Type: InfluxDB

Alerting: Supported

Name: influxdb-1

Default: off

Query language

Flux

Support for Flux in Grafana is currently in beta
Please report any issues to:
https://github.com/grafana/grafana/issues

HTTP

URL: http://localhost:8086

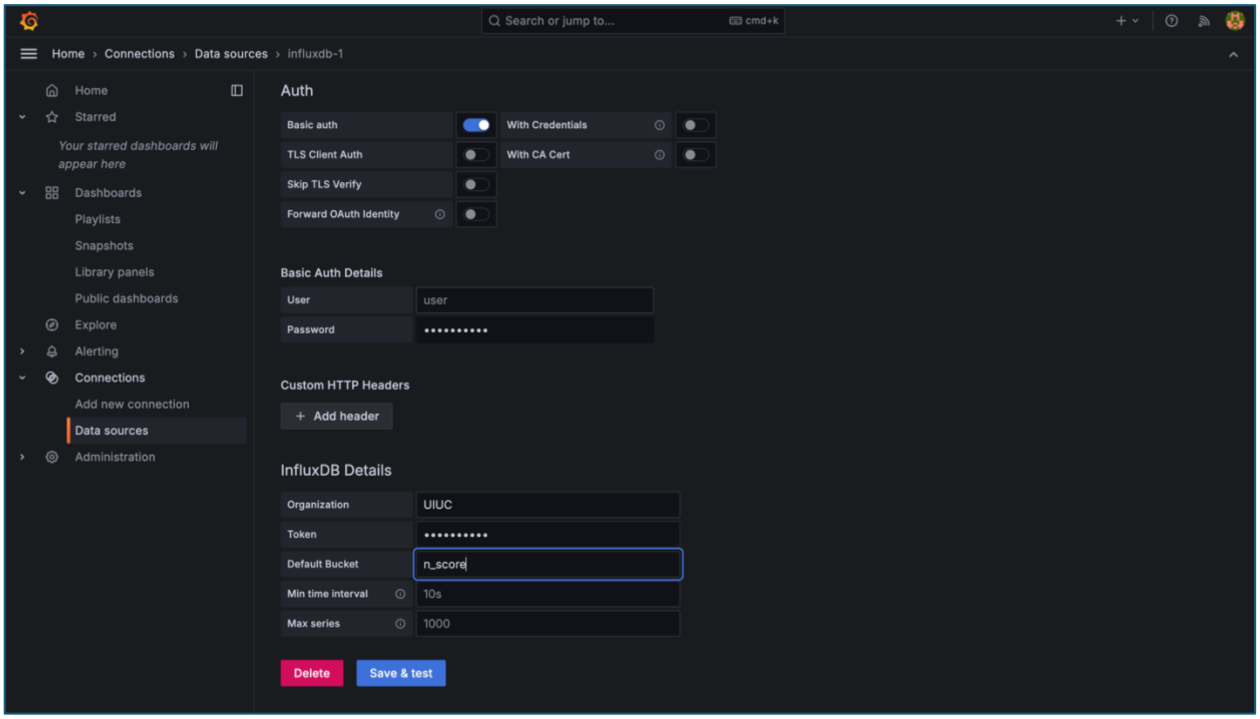
Allowed cookies: New tag (enter key to add)

Timeout: Timeout in seconds

Auth

Basic auth: on

With Credentials: off



Auth

Basic auth ● On — With Credentials ○ Off

TLS Client Auth ○ Off — With CA Cert ○ Off

Skip TLS Verify ○ Off

Forward OAuth Identity ○ Off

Basic Auth Details

User	user
Password	••••••••••

Custom HTTP Headers

InfluxDB Details

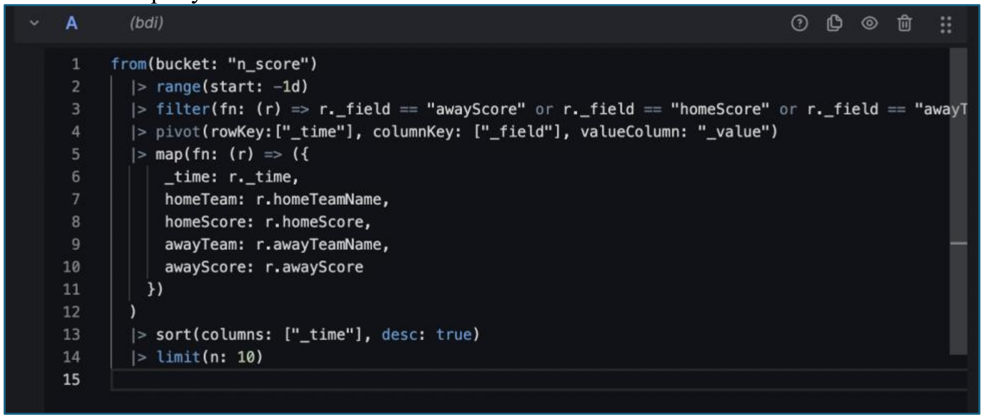
Organization	UIUC
Token	••••••••••
Default Bucket	n_score
Min time interval	10s
Max series	1000

Visulization:

Once your data sources are configured, you can start creating dashboards to visualize your data.
Go to + Create > Dashboard and begin adding panels.

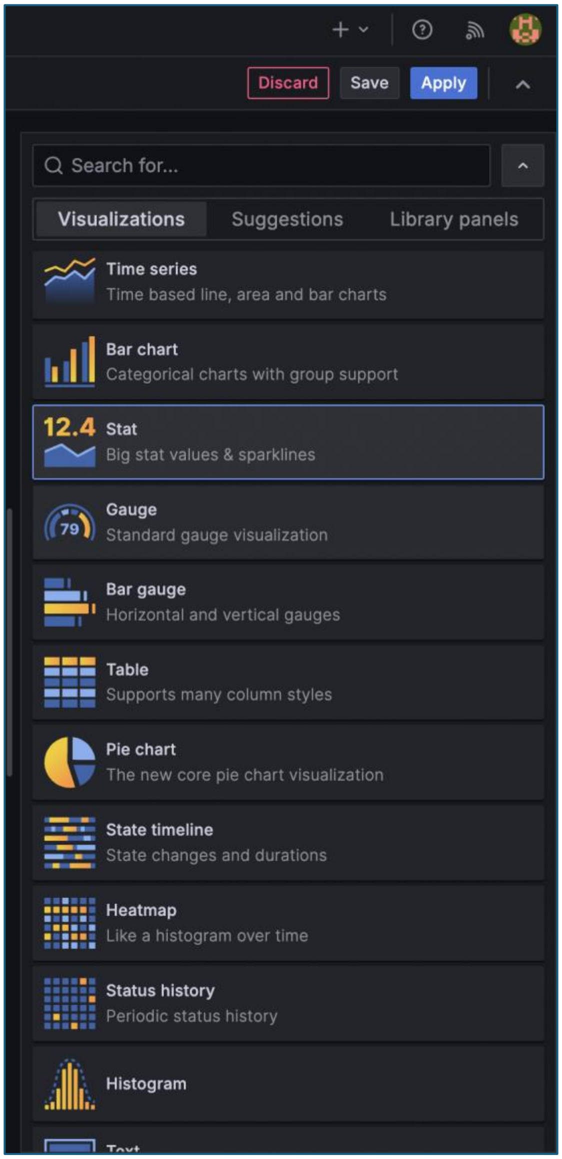
We can utilize flux query to fetch the data we need in our desired format to create and add visuals

Below is the query used:

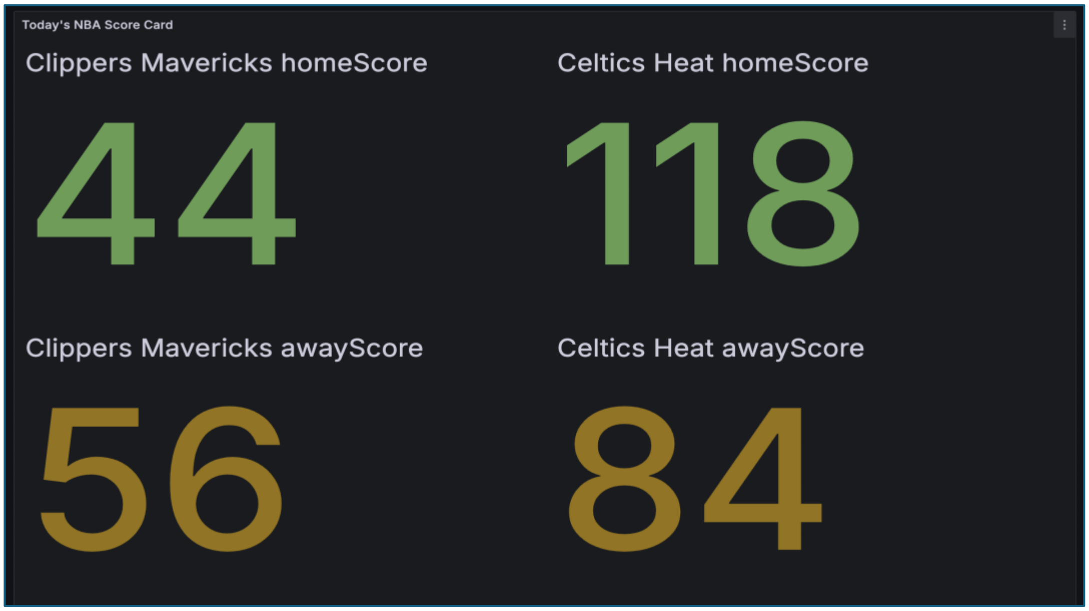


```
from(bucket: "n_score")
  |> range(start: -1d)
  |> filter(fn: (r) => r._field == "awayScore" or r._field == "homeScore" or r._field == "awayT
  |> pivot(rowKey:["_time"], columnKey: ["_field"], valueColumn: "_value")
  |> map(fn: (r) => ({
      _time: r._time,
      homeTeam: r.homeTeamName,
      homeScore: r.homeScore,
      awayTeam: r.awayTeamName,
      awayScore: r.awayScore
    })
  )
  |> sort(columns: ["_time"], desc: true)
  |> limit(n: 10)

```

After the query is run, select the desired visualization from the right hand side and configure it to your use case.


Discard
Save
Apply
Search for...
Visualizations
Suggestions
Library panels
Time series
Time based line, area and bar charts
Bar chart
Categorical charts with group support
12.4
Stat
Big stat values & sparklines
79
Gauge
Standard gauge visualization
Bar gauge
Horizontal and vertical gauges
Table
Supports many column styles
Pie chart
The new core pie chart visualization
State timeline
State changes and durations
Heatmap
Like a histogram over time
Status history
Periodic status history
Histogram
Text

Once done Dashboard is ready, this dashboard updates the live score of the NBA game at every one-minute interval



Today's NBA Score Card

Clippers Mavericks homeScore

44

Celtics Heat homeScore

118

Clippers Mavericks awayScore

56

Celtics Heat awayScore

84

Challenges:

- New Technology Learning Curve
- Dependency Setup Challenges
- Complexities with GTFS Data
- Influx DB Data Storage and Querying
- Real-Time NBA Data Limitations