
# GWINNETT BUS – MILESTONE 2

**Team 3:**
Sam Bostian, Michael Rizig, Charlie McLarty, Brian Pruitt, Allen Roman

# OVERVIEW

Recap

Updated Dataflow

Server Process Details

Data Generation

Installation Guide

Dataflow Demonstration (Event logging, Updating, Queries)

Data Validation Demonstration (Valid vs Invalid Data Recognition and Flagging)

Next Steps

# KAFKA PRODUCER RECAP

**Step 1: Import Kafka native API for python**

```
from kafka import KafkaProducer
```

**Step 2: Define server address and desired topic on said server:**

```
TOPICNAME = 'GCPS_Bus_Monitoring'
SERVERIP = 'localhost:9092'
```

**Step 3: Connect to the server via Kafka producer object and pass IP**

```
producer = KafkaProducer(bootstrap_servers=SERVERIP)
```

**Step 4: Publish data to the topic**

```
producer.send(TOPICNAME, b'data')
producer.flush()
```

# KAFKA CONSUMER RECAP

Consumer.py: “Consumes” the real-time data from the producer.

1

**Step 1: Import Kafka native API for python**

```
from kafka import
KafkaConsumer
```

2

**Step 2: Define server address and desired topic on said server:**

```
TOPICNAME =
'GCPS_Bus_Monitoring' SERVERIP
= 'localhost:9092' #insert server
ip here
```

3

**Step 3: Connect to the server via Kafka consumer object and pass desired topic**

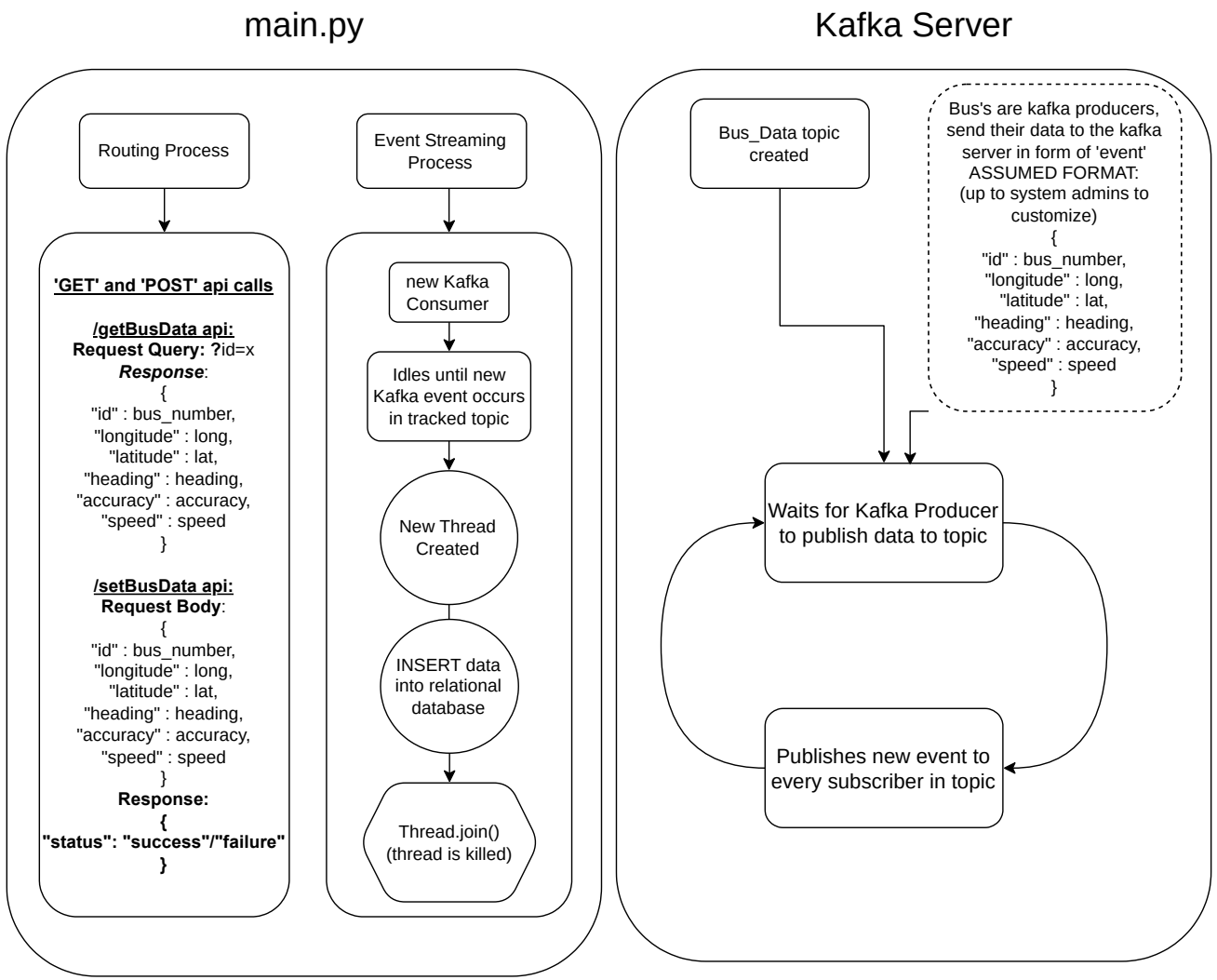
```
consumer =
KafkaConsumer(TOPICNAME,
bootstrap_servers=SERVERIP)
```

4

**Step 4: Parse data:**

```
for messages in consumer:
process(messages['value'])
```

# SERVER PROCESS OVERVIEW



# SYNTHETIC DATA GENERATION

- Gwinnett County map converted to directed graph with saved coordinates data of edges and nodes

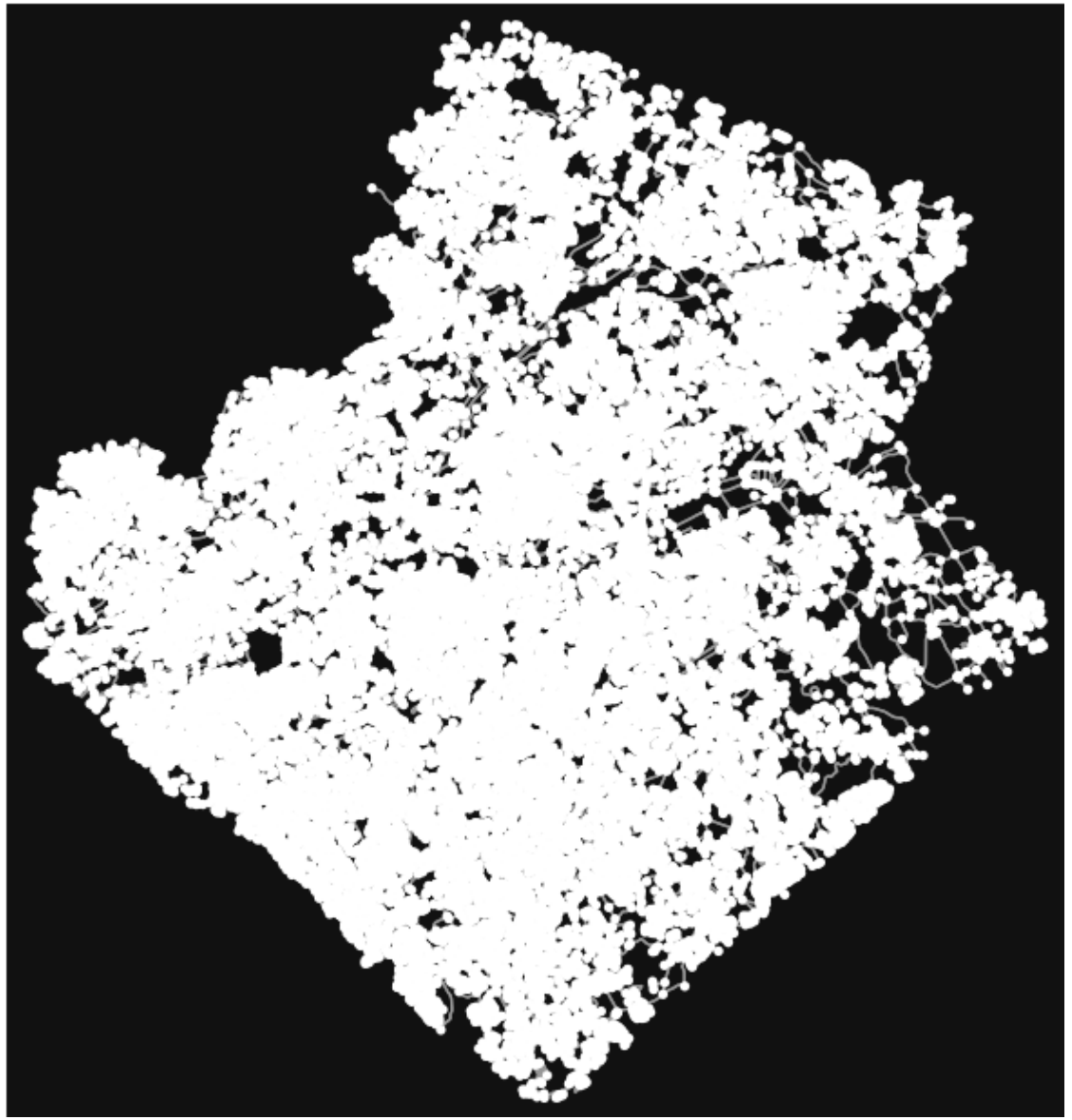
- Bus calculates distance traveled off speed (m/s) * 5 second's along current edge

- Data collector receives all updates and sends to the producer

- In Progress:

  Better bus route logic (from school to random location in the school radius and back)

  Edit the time to simulate a real-life scenario of each bus traveling in morning before school to track whether bus is running on time



Gwinnett county represented as a strongly connected graph

# SQL AND KAFKA INSTALL DOCUMENTATION

MS SQL SERVER FOR LINUX RH9

Install:

sudo yum install -y mssql-server-selinux

Setup:

sudo /opt/mssql/bin/mssql-conf setup

Run/ ensure status

systemctl status mssql-server

connect:

sqlcmd -S RHEL9.xVM -U SA -P HootyHoo!

KAFKA:

```
$ sudo wget https://dlcdn.apache.org/kafka/3.8.0/kafka_2.13-3.8.0.tgz
```

After installing:

```
$ tar -xzf kafka_2.13-3.8.0.tgz
```

```
$ cd kafka_2.13-3.8.0
```

Generate a Cluster UUID

```
$ KAFKA_CLUSTER_ID="$(bin/kafka-storage.sh random-uuid)"
```

Format Log Directories

```
$ bin/kafka-storage.sh format -t $KAFKA_CLUSTER_ID -c
config/kraft/server.properties
```

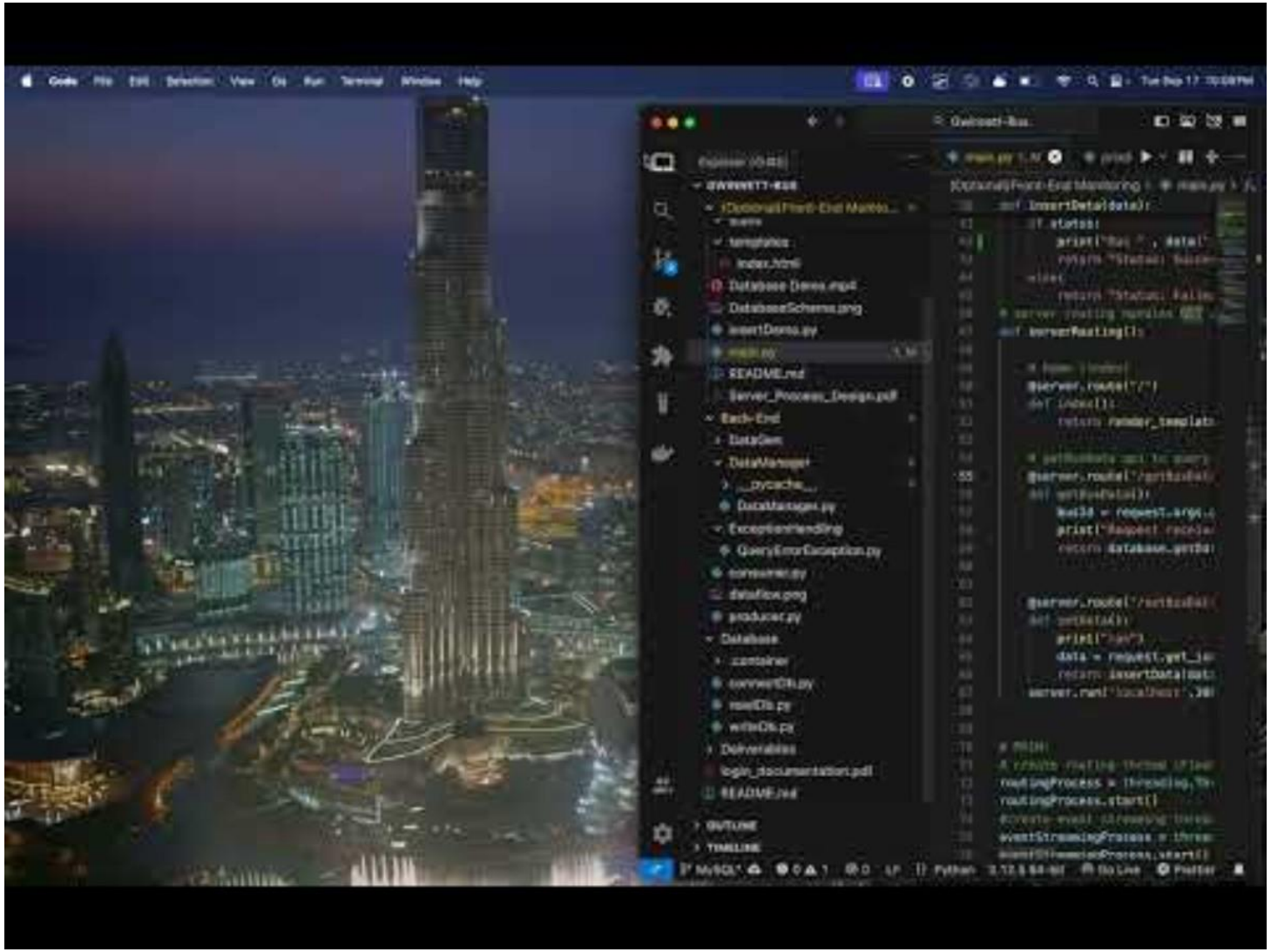
Start the Kafka Server

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

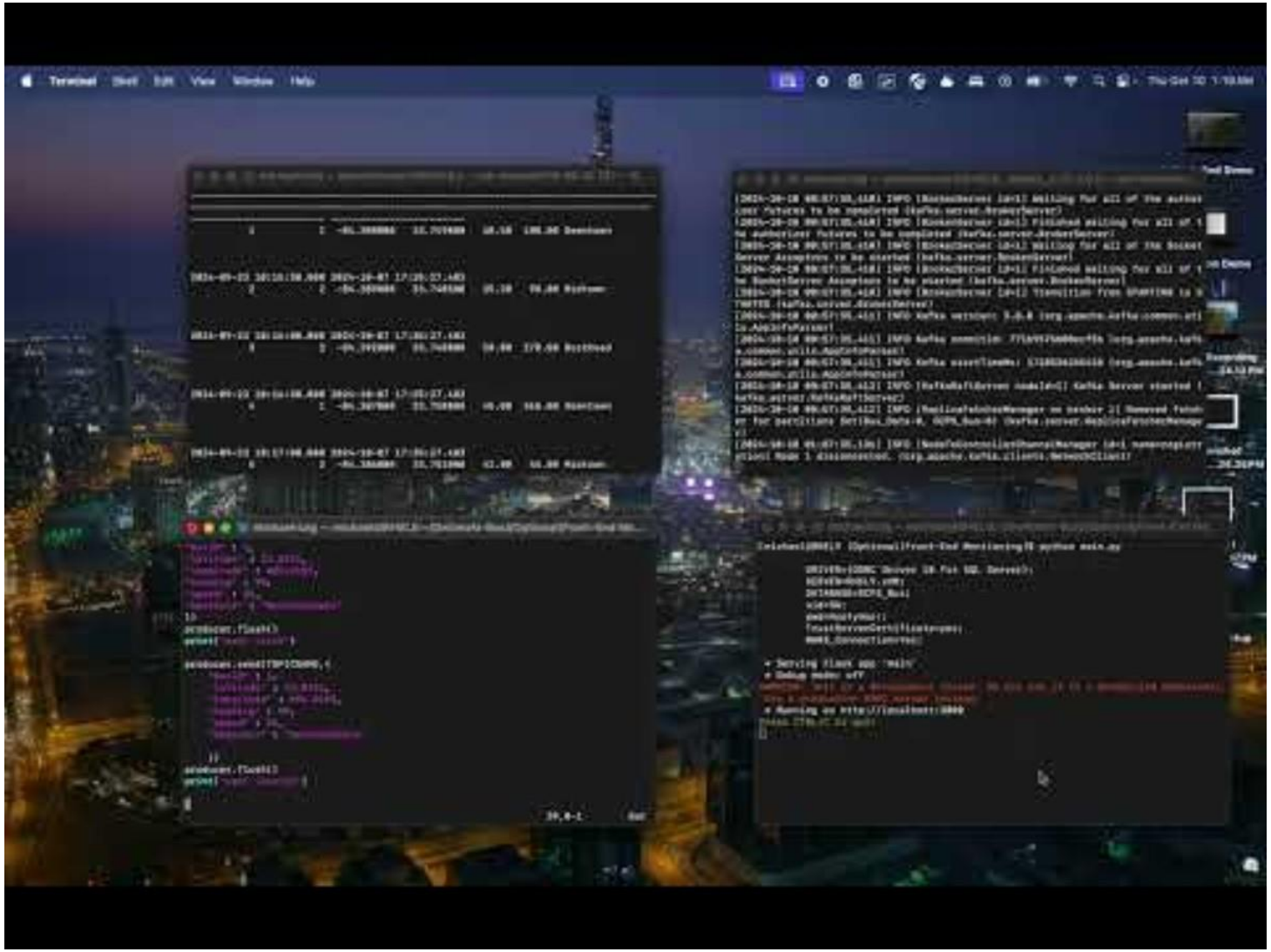
```
Create a topic:
```

```
$ bin/kafka-topics.sh --create --topic TOPICNAME --bootstrap-server
localhost:9092
```

# DATA FLOW



# DATA VALIDATION



# NEXT STEPS

Testing/QA: Running unit testing in scale, utilizing up to 2000 simultaneous bus inputs to simulate a realistic workload scenario and possibly test more for growth.

Containerization: Utilizing Podman to containerize our project for simplicity, scalability and deployment.

Bonus:

Front End: Revamping the front end to highlight stale data, and filtering for bus id, late status, geofence, etc.