Software Requirements

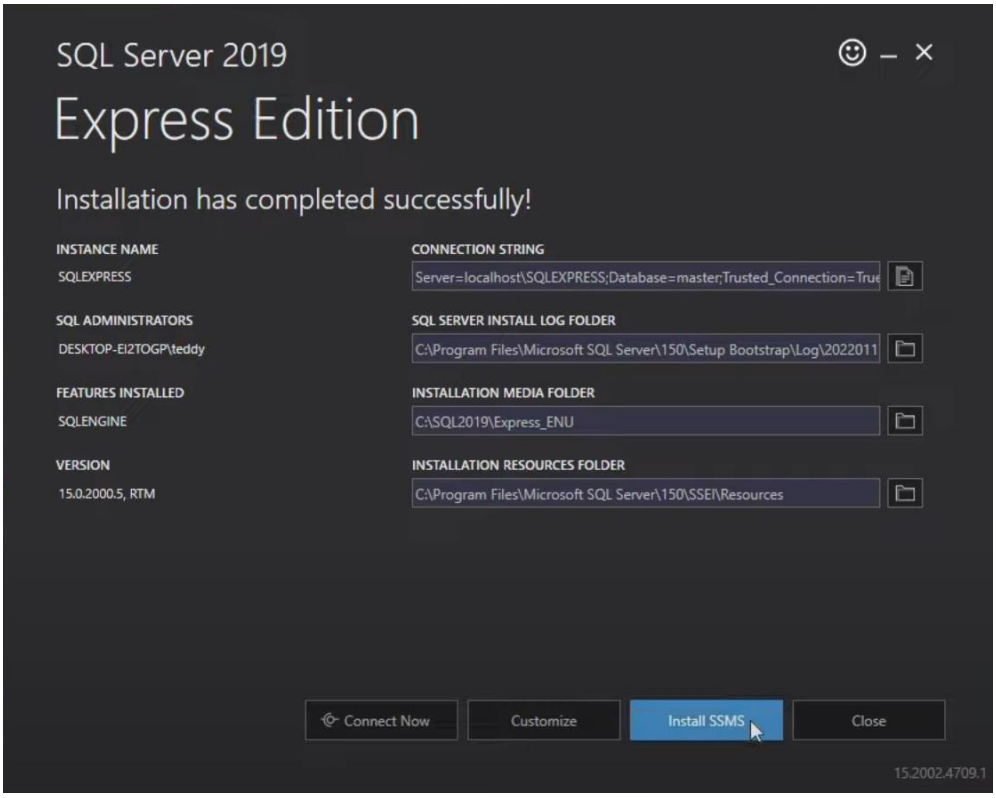
1. SQL Server: https://www.microsoft.com/en-us/sql-server/sql-server-downloads

Developer

SQL Server 2022 Developer is a full-featured free edition, licensed for use as a development and test database in a non-production environment.

Download now

   a. Microsoft SQL Server Management Studio (click Install SSMS when installing SQL Server)



2. Python Version 3.7 or later
    a. Required library: pyodbc → pip install pyodbc
3. Apache Kafka Scala 2.12 or 2.13: https://kafka.apache.org/downloads

SUPPORTED RELEASES

3.9.0

- Released November 6, 2024
- Release Notes
- Docker image: apache/kafka:3.9.0.
- Docker Native image: apache/kafka-native:3.9.0.
- Source download: kafka-3.9.0-src.tgz (asc, sha512)
- Binary downloads:
    - Scala 2.12 - kafka_2.12-3.9.0.tgz (asc, sha512)
    - Scala 2.13 - kafka_2.13-3.9.0.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).

4. Apache Zookeeper 3.9.3: https://zookeeper.apache.org/releases.html

Apache ZooKeeper™ Project Documentation Developers ASF

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 3.9.3 is our current release, and 3.8.4 our latest stable release.

Apache ZooKeeper 3.9.3

Apache ZooKeeper 3.9.3(asc, sha512)

Apache ZooKeeper 3.9.3 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 (3.7 is EoL since 2nd of February, 2024)

Apache ZooKeeper 3.7.2(asc, sha512)

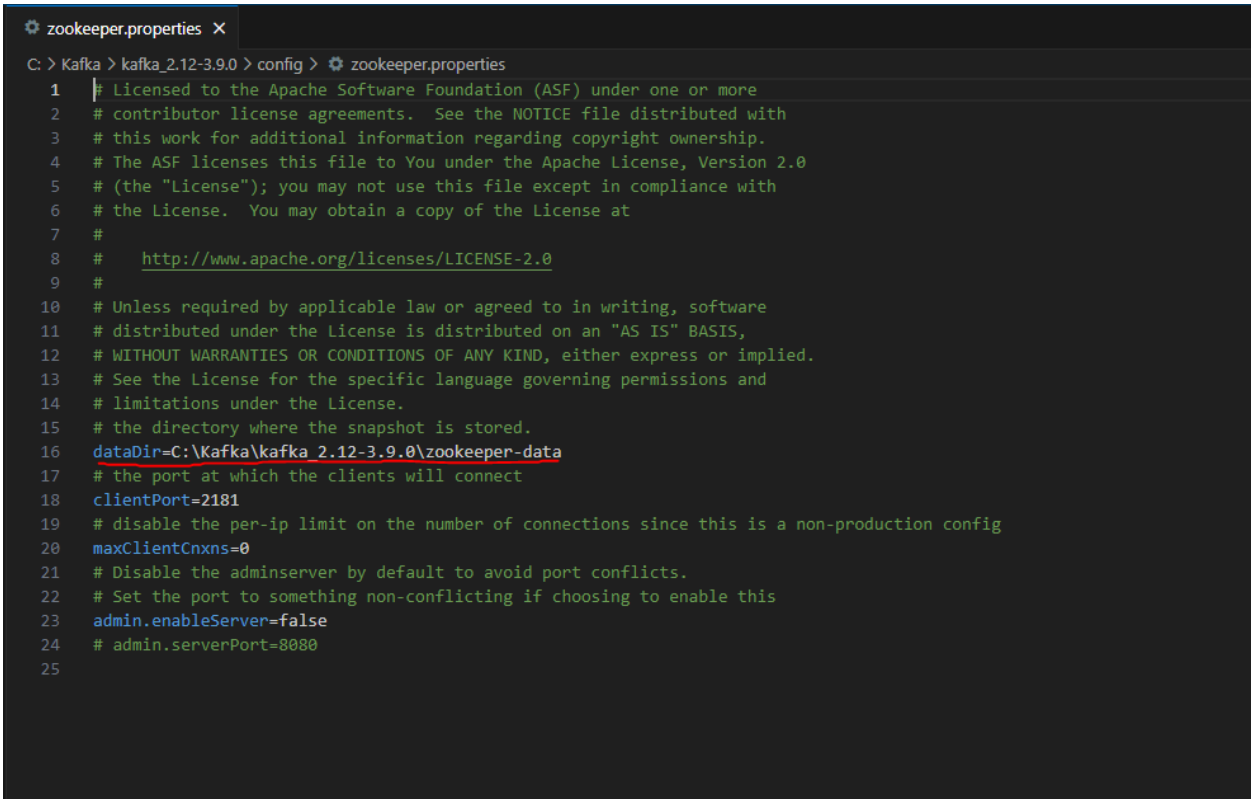
Apache ZooKeeper 3.7.2 Source Release(asc, sha512)

Older releases are available in the archive.

Setup Kafka

Instructions found here: https://www.geeksforgeeks.org/how-to-install-and-run-apache-kafka-on-windows/

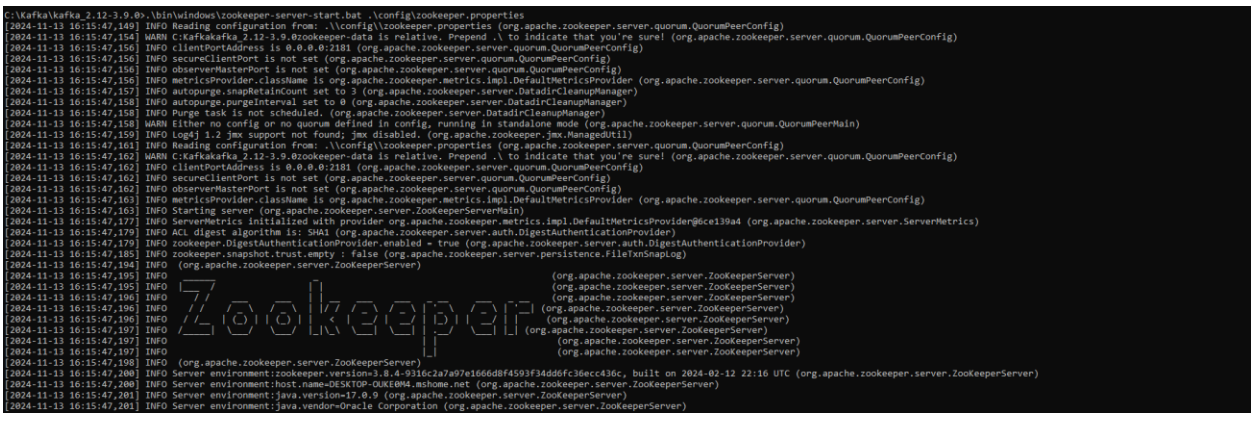
1. Extract Apache Kafka your desired directory.
2. Copy the path of the Kafka folder. Now go to *config* inside kafka folder and open *zookeeper.properties* file. Copy the path against the field *dataDir* and add */zookeeper-data* to the path.



3. Now in the same folder *config* open *server.properties* and scroll down to *log.dirs* and paste the path. To the path add */kafka-logs*

4. Now open command prompt and change the directory to the kafka folder. First start zookeeper using the command given below:

   .\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties

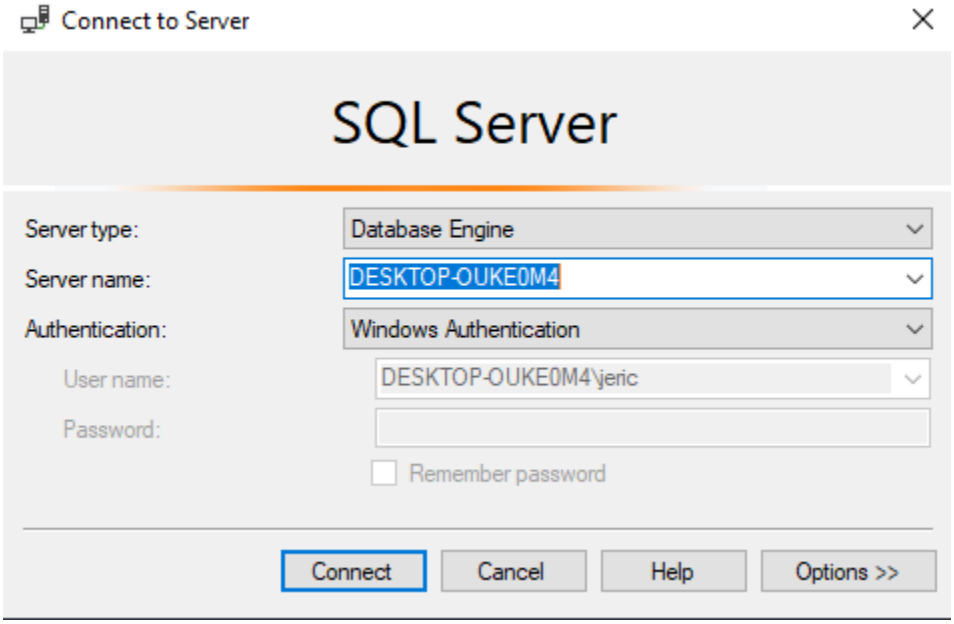


5. Now open another command prompt and change the directory to the kafka folder. Run kafka server using the command:

   .\bin\windows\kafka-server-start.bat .\config\server.properties

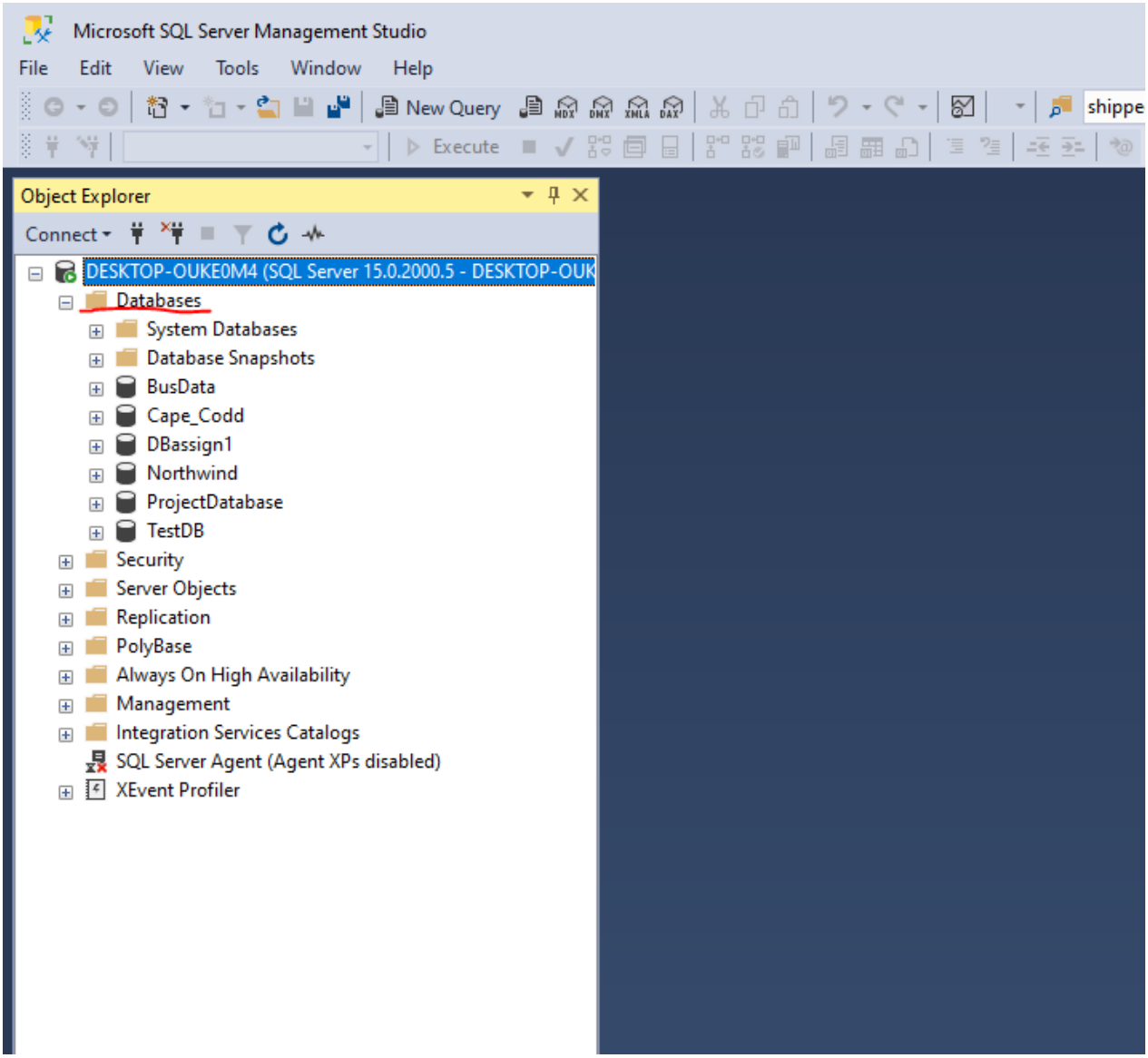
Setup Microsoft SQL Server Management Studio

1. Launch Microsoft SQL Server Management Studio

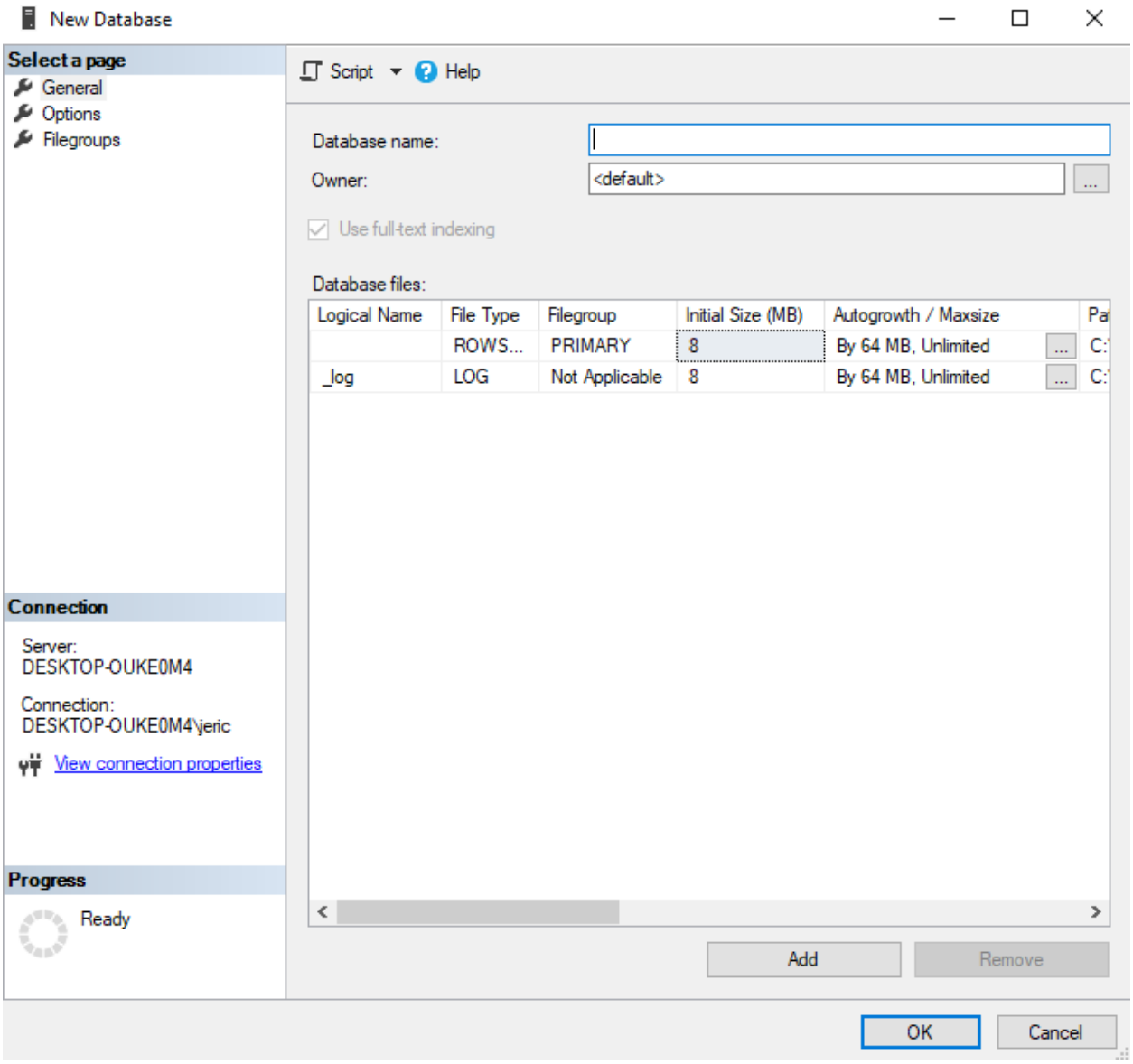


2. Take note of the Server name and connect.

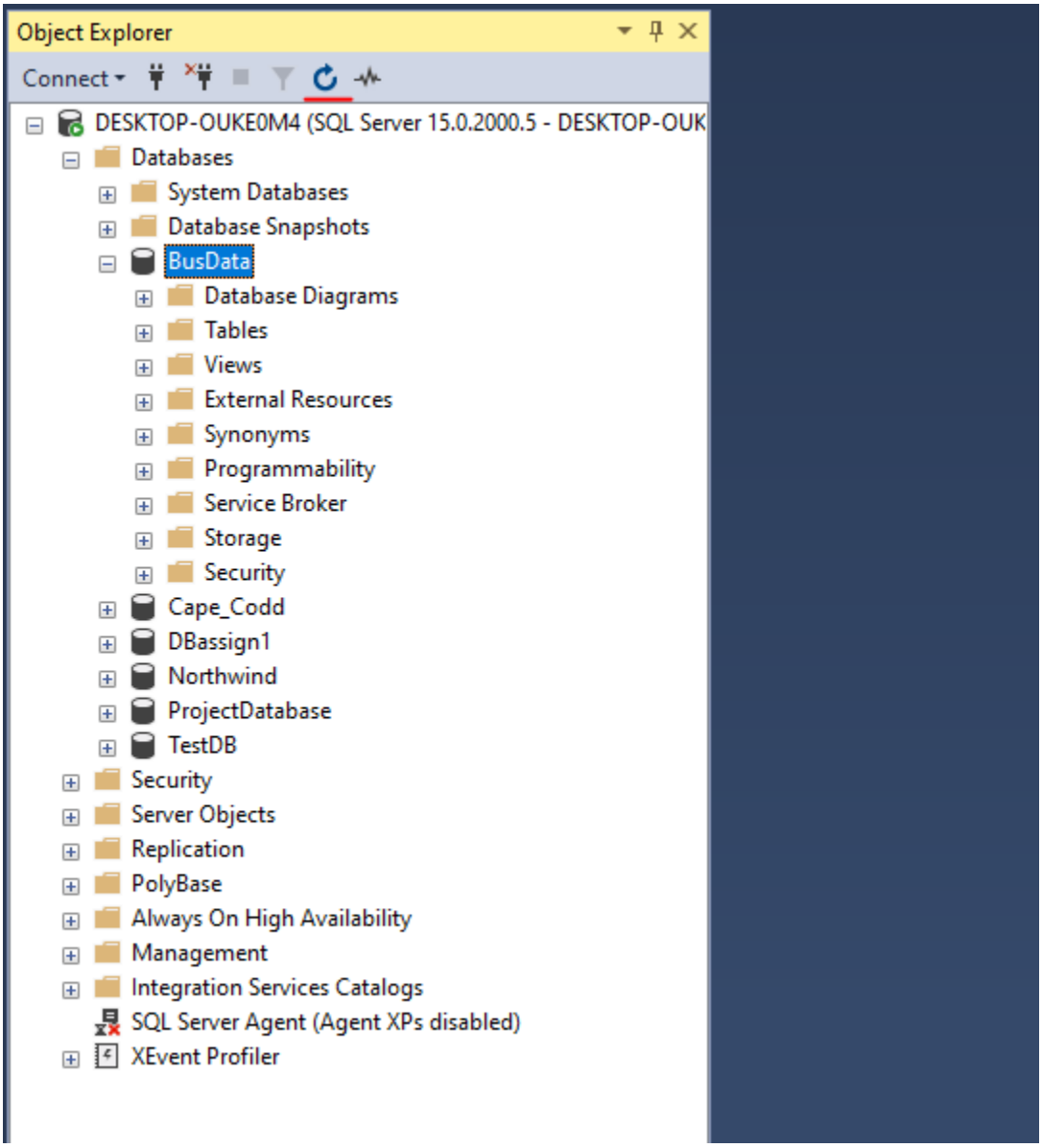
3. Create a new database, right click on ‘Databases’ and click create new ‘Database’…



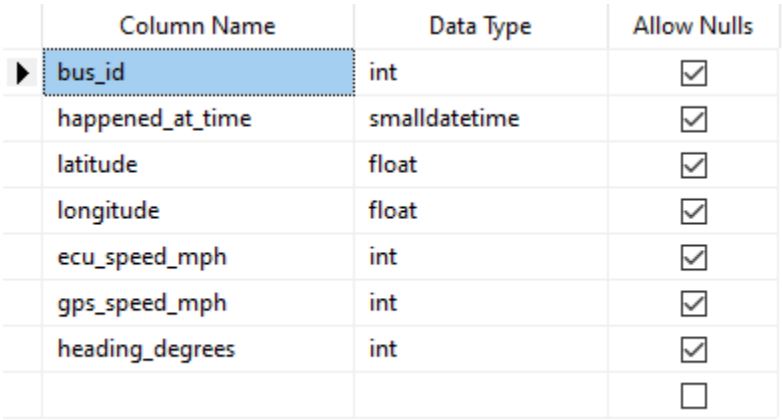
4. Name your database and click OK



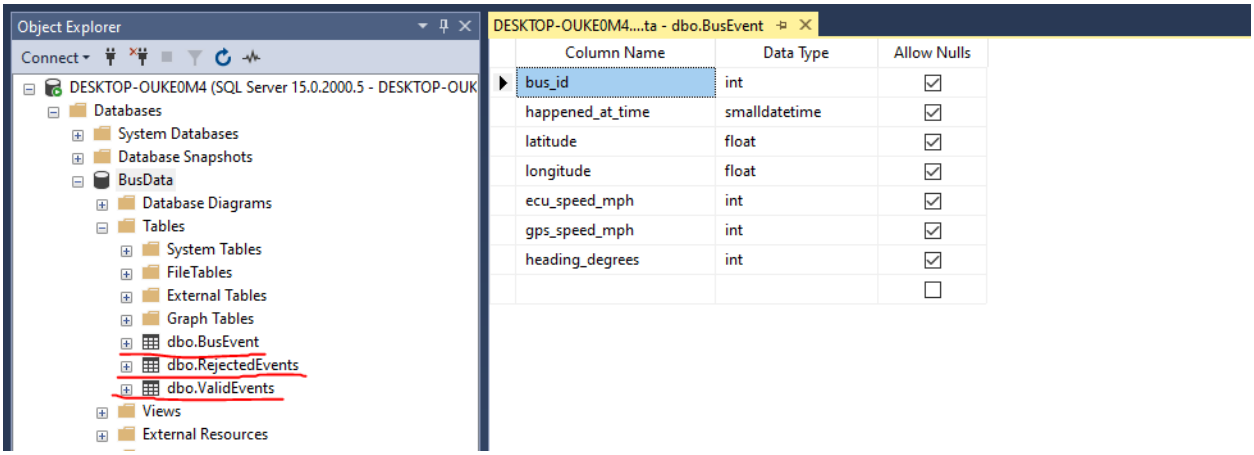
5. Refresh to see your database.



6. Expand your database you created by double clicking it, or by clicking the + next to your database
7. Create three new tables by right clicking ‘Tables’, clicking ‘New’ and then Table…
8. Each table will have the same column names and data types.
    a. bus_id: int
    b. happened_at_time: smalldatetime
    c. latitude: float
    d. longitude: float
    e. ecu_speed_mph: int
    f. gps_speed_mph: int
    g. heading_degrees: int



Column Name	Data Type	Allow Nulls
bus_id	int	☑
happened_at_time	smalldatetime	☑
latitude	float	☑
longitude	float	☑
ecu_speed_mph	int	☑
gps_speed_mph	int	☑
heading_degrees	int	☑
		☐



9. Setup for Microsoft SQL Server Management Studio is complete, take not of what the Sever Name is, what your database name is, and the name of your three tables.

Running the Source Code

1. Copy and Paste the Source Code, save and name your file.
2. Adjust the names of the server and the database to match your own.

```python
import pyodbc
import random
import time
from datetime import datetime, timedelta
from kafka import KafkaProducer
import json

# Kafka connection details
KAFKA_TOPIC = 'bus_data'
KAFKA_SERVER = 'localhost:9092'  # Update this to your Kafka server if different

# SQL Server connection details
server = 'DESKTOP-OUKE0M4'
database = 'BusData'
driver = '{ODBC Driver 17 for SQL Server}'

# Initialize Kafka Producer
producer = KafkaProducer(
    bootstrap_servers=KAFKA_SERVER,
    value_serializer=lambda v: json.dumps(v, default=str).encode('utf-8')
)

# Connect to SQL Server
connection_string = f'DRIVER={driver};SERVER={server};DATABASE={database};Trusted_Connection=yes;'
conn = pyodbc.connect(connection_string)
cursor = conn.cursor()
print("Connected to SQL Server and Kafka.")


# Generate and insert random data
1 usage
def generate_random_data():
    bus_id = random.randint( a: 1,  b: 100)
    happened_at_time = datetime.now() - timedelta(minutes=random.randint( a: 0,  b: 1000))
    latitude = round(random.uniform(-90,  b: 90), 6)
    longitude = round(random.uniform(-180,  b: 180), 6)
    ecu_speed_mph = random.randint( a: 0,  b: 120)
    gps_speed_mph = random.randint( a: 0,  b: 120)
    heading_degrees = random.randint( a: 0,  b: 360)

    row_data = {
        "bus_id": bus_id,
        "happened_at_time": happened_at_time,
        "latitude": latitude,
        "longitude": longitude,
        "ecu_speed_mph": ecu_speed_mph,
        "gps_speed_mph": gps_speed_mph,
        "heading_degrees": heading_degrees
    }
```

3. Adjust the names of the databases to match your own.

```
    # Determine if data is valid or rejected based on speed criteria
    if ecu_speed_mph > 70 or gps_speed_mph > 70:
        cursor.execute( sql: '''
            INSERT INTO dbo.RejectedEvents (bus_id, happened_at_time, latitude, longitude, ecu_speed_mph, gps_speed_mph, heading_degrees)
            VALUES (?, ?, ?, ?, ?, ?, ?)
        ''', *params: (bus_id, happened_at_time, latitude, longitude, ecu_speed_mph, gps_speed_mph, heading_degrees))
        print("An invalid record was inserted into RejectedEvents.")
    else:
        cursor.execute( sql: '''
            INSERT INTO dbo.BusEvent (bus_id, happened_at_time, latitude, longitude, ecu_speed_mph, gps_speed_mph, heading_degrees)
            VALUES (?, ?, ?, ?, ?, ?, ?)
        ''', *params: (bus_id, happened_at_time, latitude, longitude, ecu_speed_mph, gps_speed_mph, heading_degrees))

        cursor.execute( sql: '''
            INSERT INTO dbo.ValidEvents (bus_id, happened_at_time, latitude, longitude, ecu_speed_mph, gps_speed_mph, heading_degrees)
            VALUES (?, ?, ?, ?, ?, ?, ?)
        ''', *params: (bus_id, happened_at_time, latitude, longitude, ecu_speed_mph, gps_speed_mph, heading_degrees))
        print("A valid record was inserted into BusEvent and ValidEvents.")
```

4. The program will now run properly when zookeeper and a Kafka server are running.