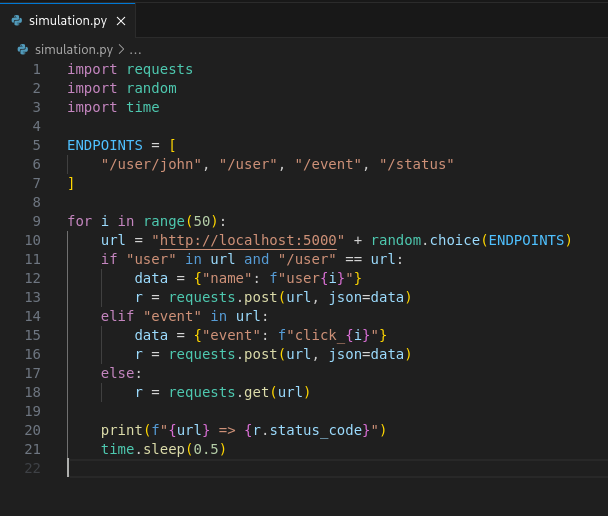
Week 1: Infrastructure and API Development

1. Develop/Use a REST API server with 5-10 endpoints.



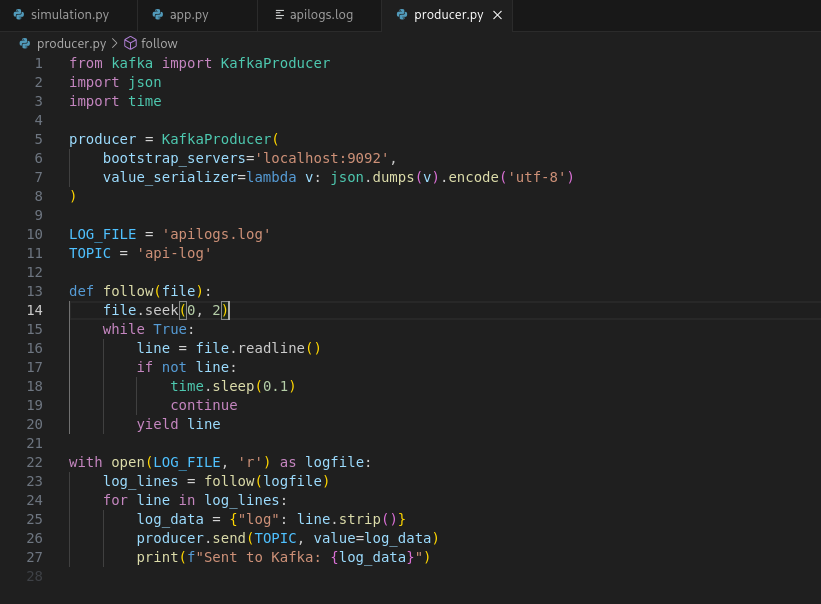
1. Create a script to simulate workload by generating API requests.

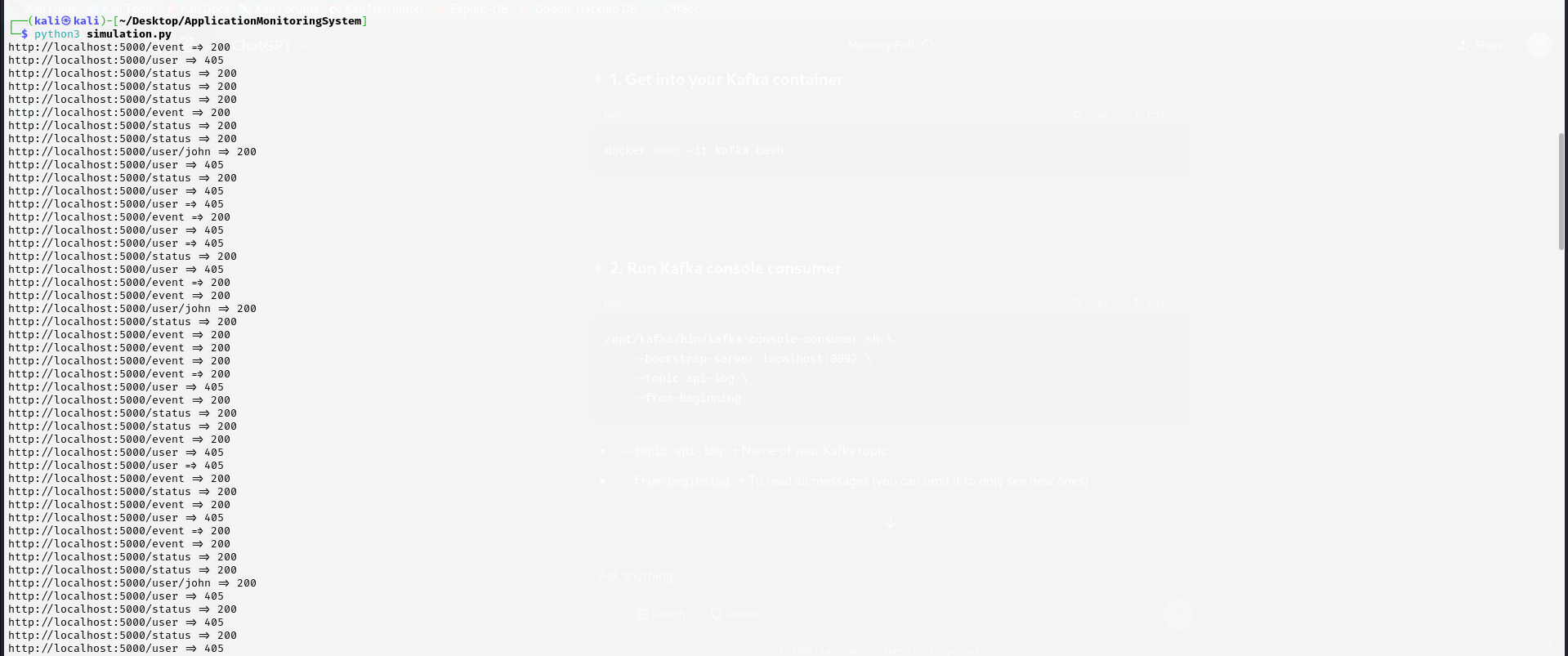


1. Install and configure Apache Kafka with topics for different log types.

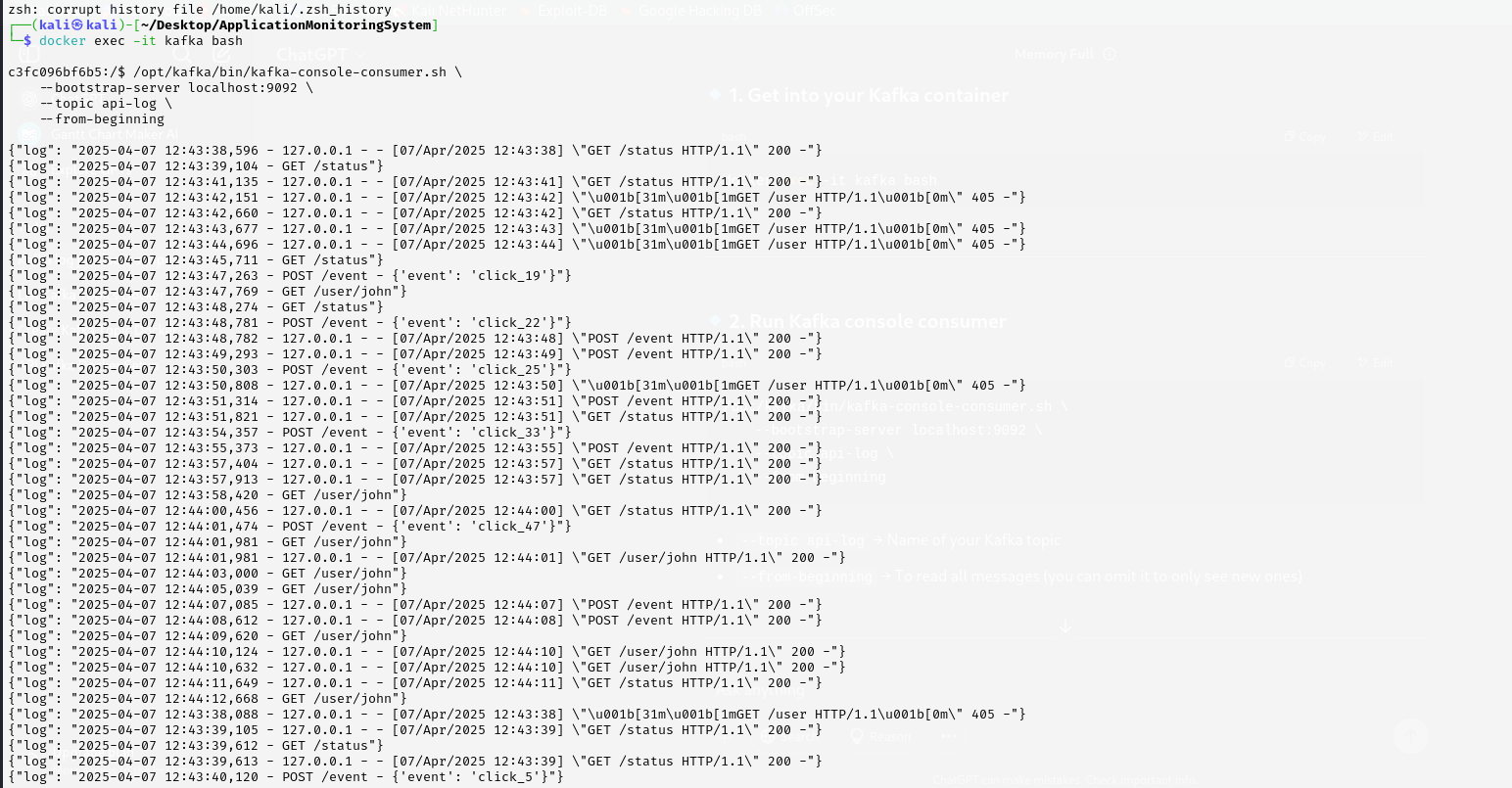


1. Implement a producer that pushes logs into Kafka topics.

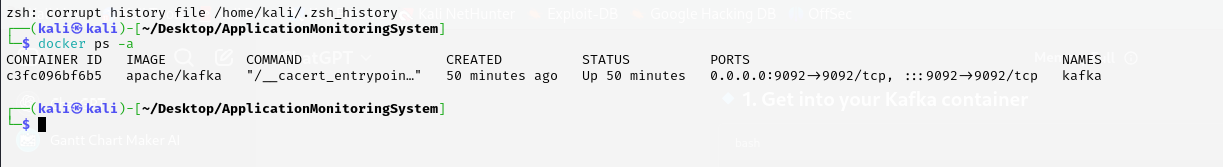








1. Set up Docker containers for all components.



**Week 2: Log Processing and Storage**

Consumer.py file:

**from kafka import KafkaConsumer**

**import psycopg2**

**import json**

**import re**

**from datetime import datetime**

**conn = psycopg2.connect(**

**dbname="mydb",**

**user="myuser",**

**password="mypassword",**

**host="localhost",**

**port="5432"**

**)**

**cursor = conn.cursor()**

**# Ensure table exists**

**cursor.execute("""**

**CREATE TABLE IF NOT EXISTS logs (**

**id SERIAL PRIMARY KEY,**

**timestamp TIMESTAMPTZ,**

**method VARCHAR(10),**

**endpoint TEXT,**

**status\_code INT,**

**raw\_log TEXT**

**);**

**""")**

**conn.commit()**

**# Kafka setup**

**consumer = KafkaConsumer(**

**"api-log",**

**bootstrap\_servers=["localhost:9092"],**

**value\_deserializer=lambda m: json.loads(m.decode("utf-8")),**

**auto\_offset\_reset="earliest",**

**enable\_auto\_commit=True,**

**group\_id="log-consumer-group"**

**)**

**# Log parser**

**def parse\_log(raw\_log):**

**# Regex to match HTTP logs**

**http\_log\_match = re.search(r'(\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2},\d+) - .\*?\"(GET|POST|PUT|DELETE|HEAD) (.\*?) HTTP.\*?\" (\d{3})', raw\_log)**

**simple\_match = re.search(r'(\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2},\d+) - (GET|POST|PUT|DELETE|HEAD) (/.+)', raw\_log)**

**if http\_log\_match:**

**timestamp = datetime.strptime(http\_log\_match.group(1), '%Y-%m-%d %H:%M:%S,%f')**

**method = http\_log\_match.group(2)**

**endpoint = http\_log\_match.group(3)**

**status\_code = int(http\_log\_match.group(4))**

**elif simple\_match:**

**timestamp = datetime.strptime(simple\_match.group(1), '%Y-%m-%d %H:%M:%S,%f')**

**method = simple\_match.group(2)**

**endpoint = simple\_match.group(3)**

**status\_code = None**

**else:**

**return None # Unrecognized format**

**return {**

**"timestamp": timestamp,**

**"method": method,**

**"endpoint": endpoint,**

**"status\_code": status\_code,**

**"raw\_log": raw\_log**

**}**

**# Consume and insert logs**

**print("📡 Listening to Kafka...")**

**for message in consumer:**

**raw\_log = message.value.get("log")**

**print("🔹 Raw:", raw\_log)**

**parsed = parse\_log(raw\_log)**

**if parsed:**

**cursor.execute("""**

**INSERT INTO logs (timestamp, method, endpoint, status\_code, raw\_log)**

**VALUES (%s, %s, %s, %s, %s)**

**""", (**

**parsed["timestamp"],**

**parsed["method"],**

**parsed["endpoint"],**

**parsed["status\_code"],**

**parsed["raw\_log"]**

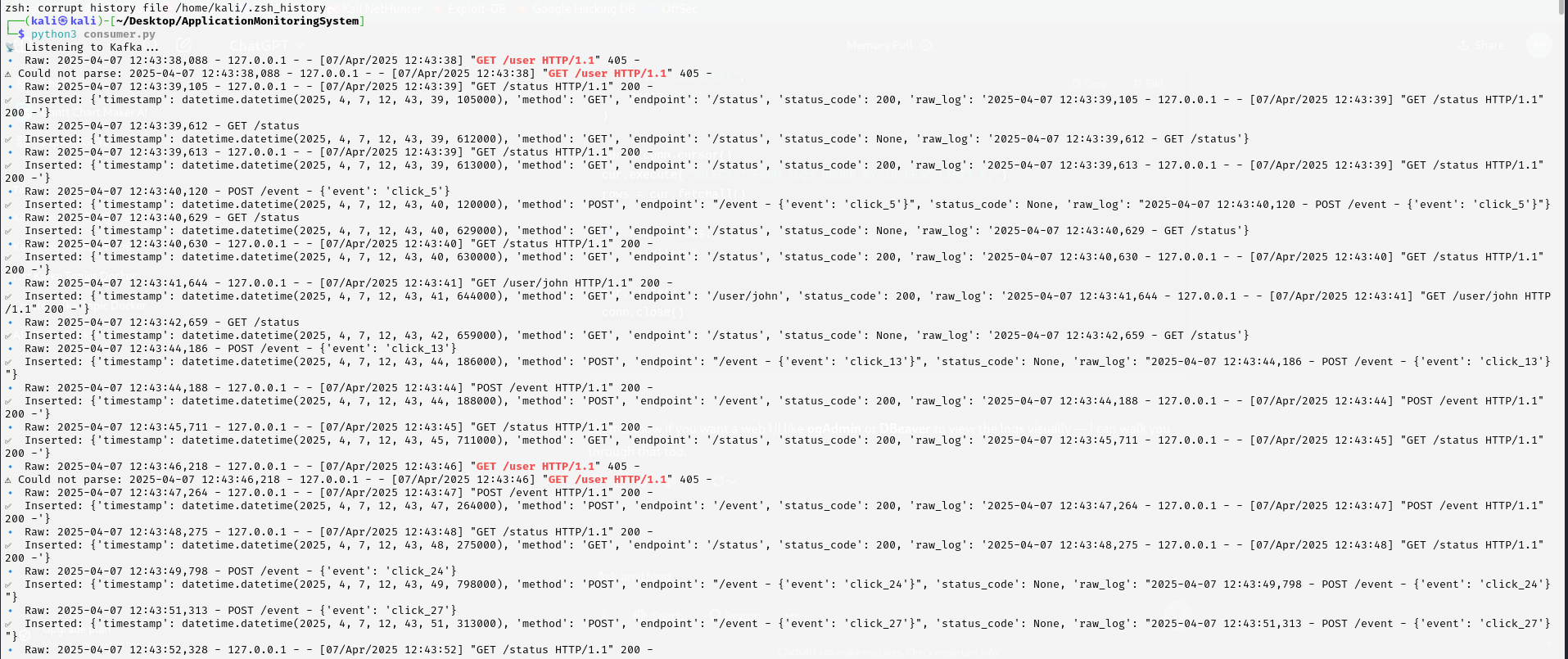
**))**

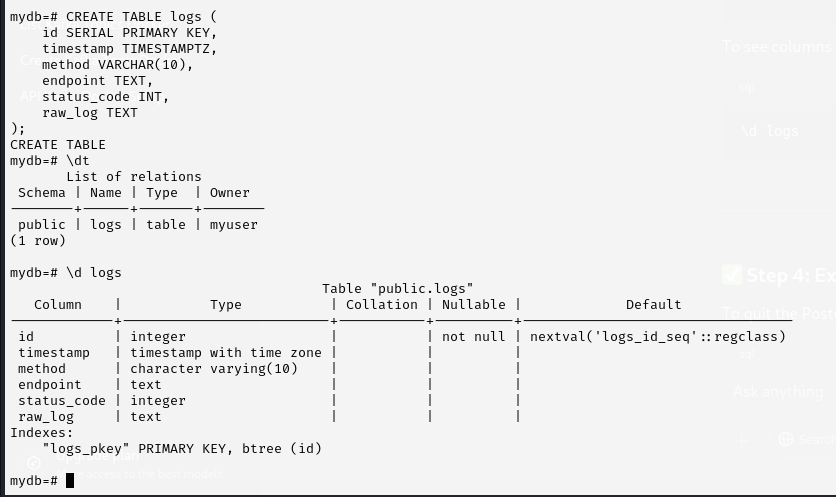
**conn.commit()**

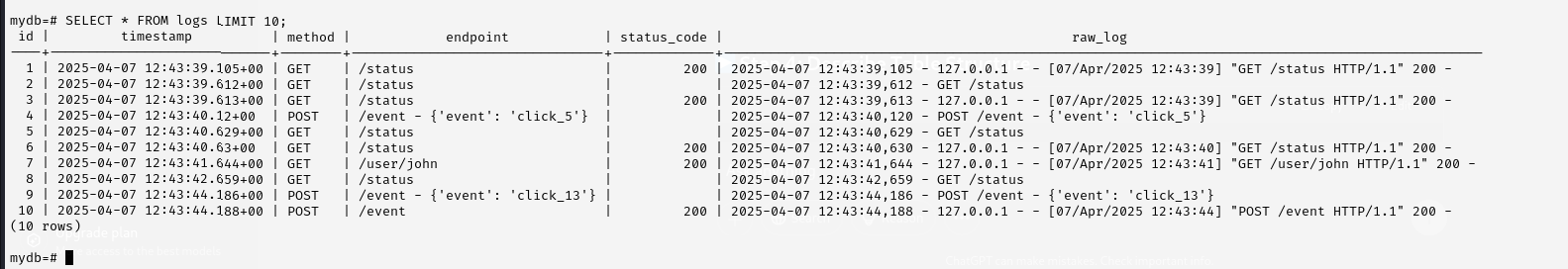
**print("✅ Inserted:", parsed)**

**else:**

**print("⚠️ Could not parse:", raw\_log)**

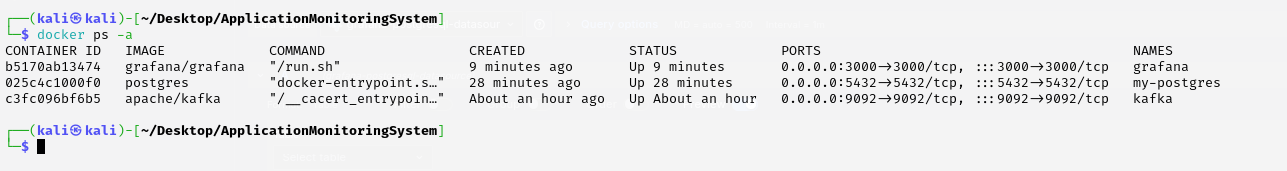


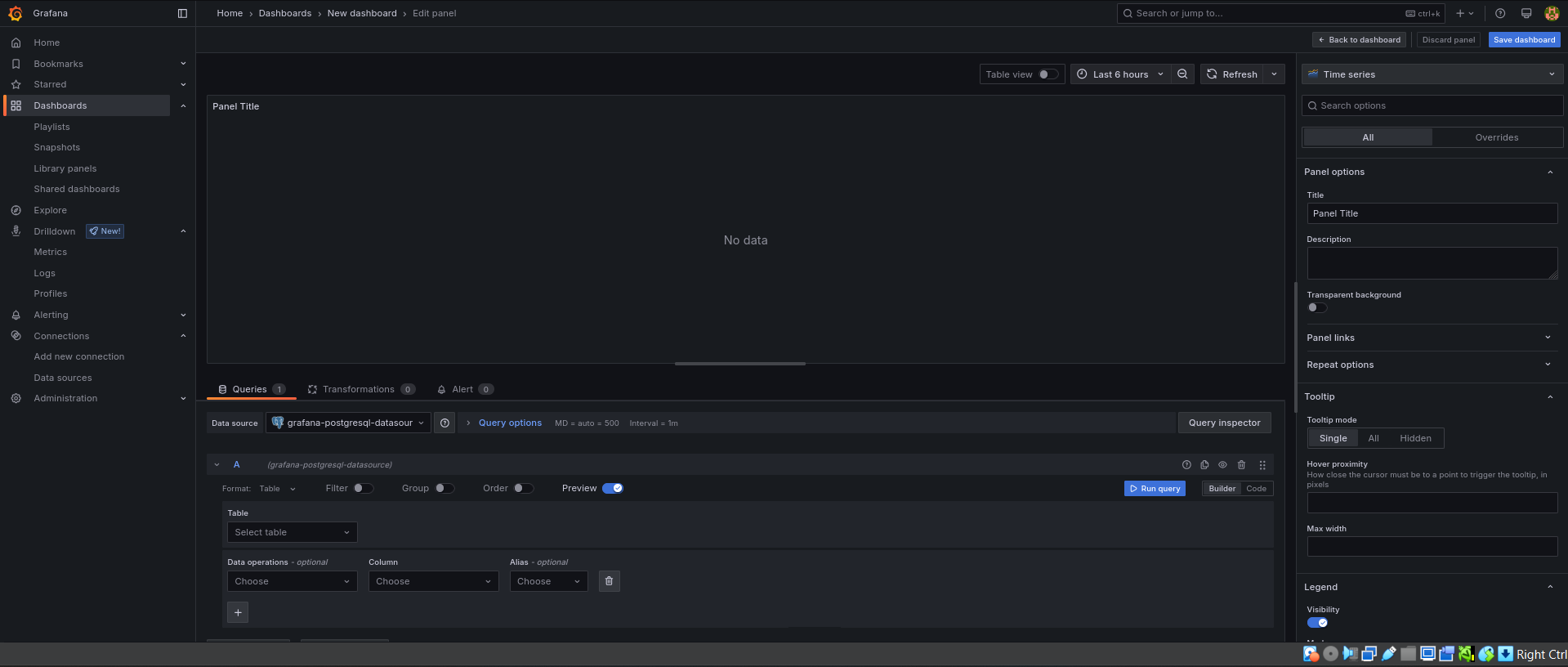




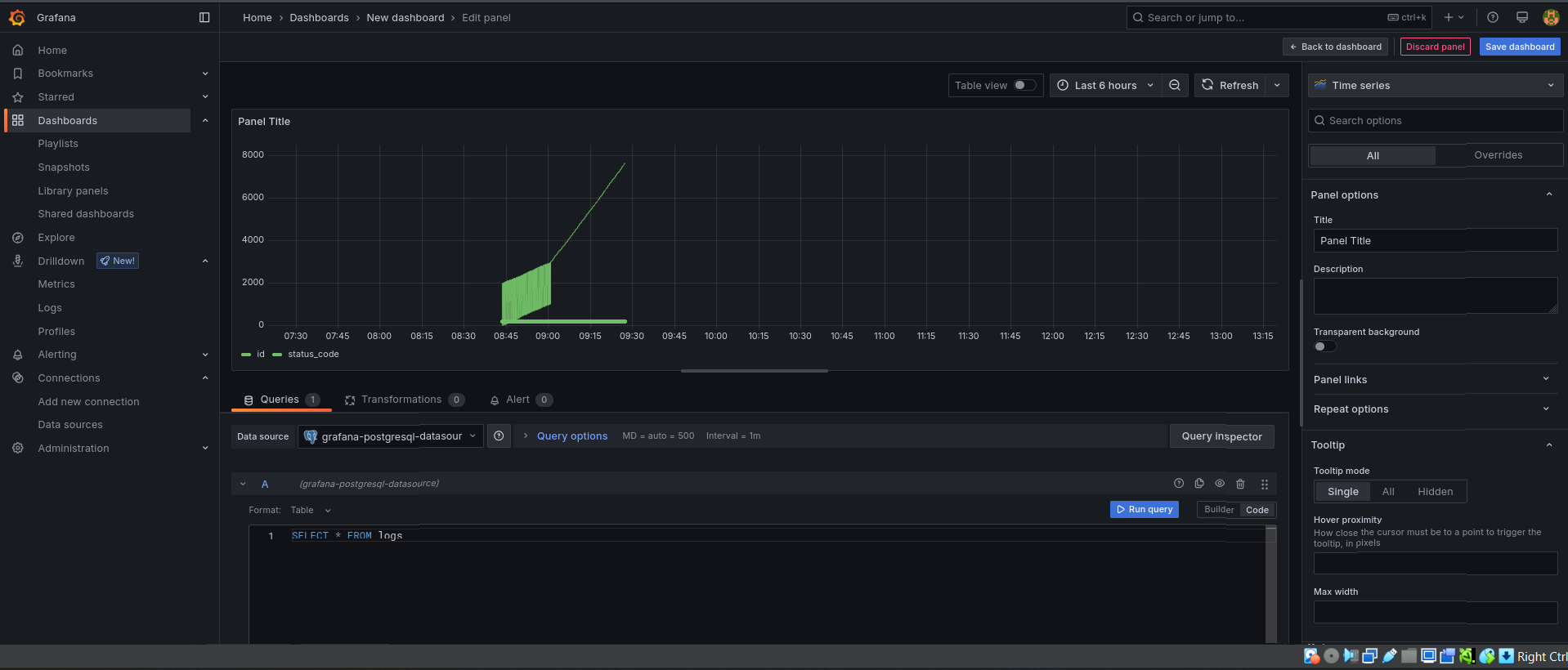
**Week 3: Visualization**

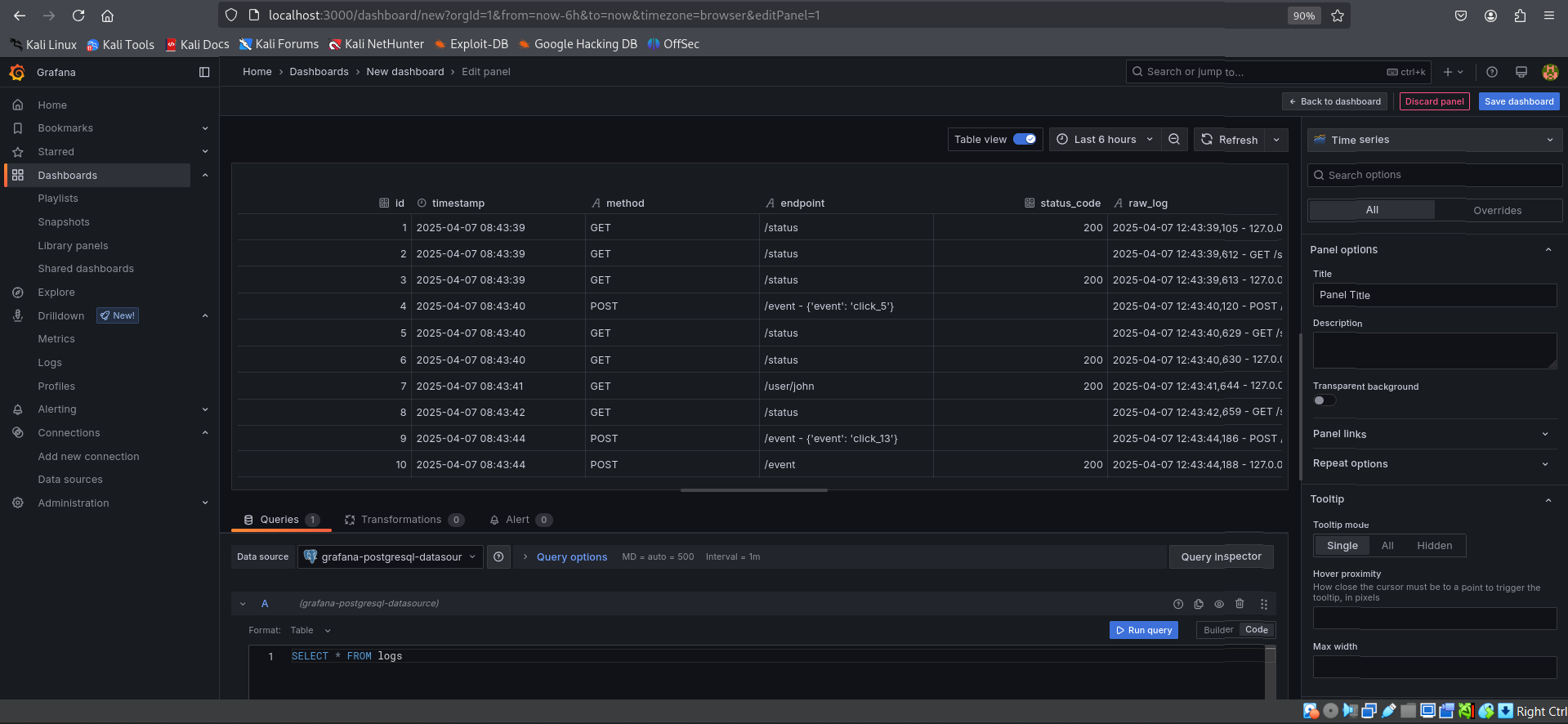
1. Connect Grafana to the database and configure dashboards.

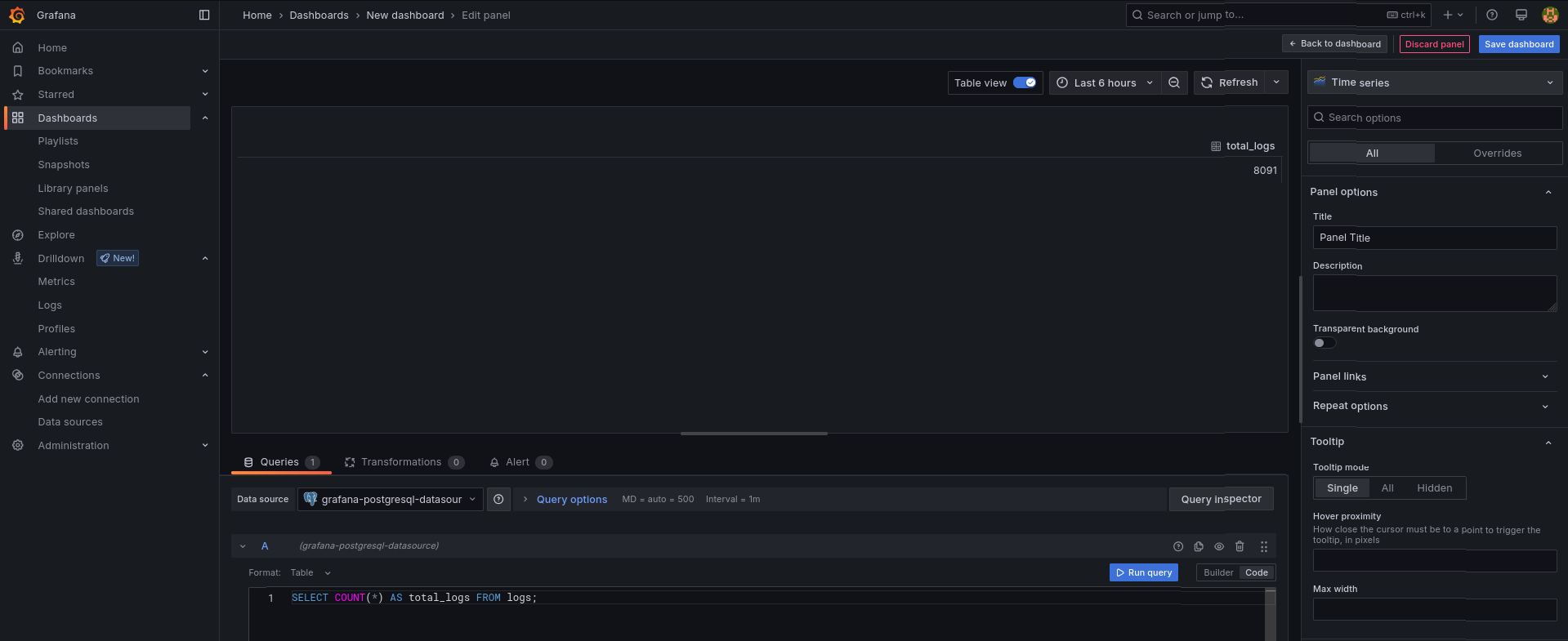


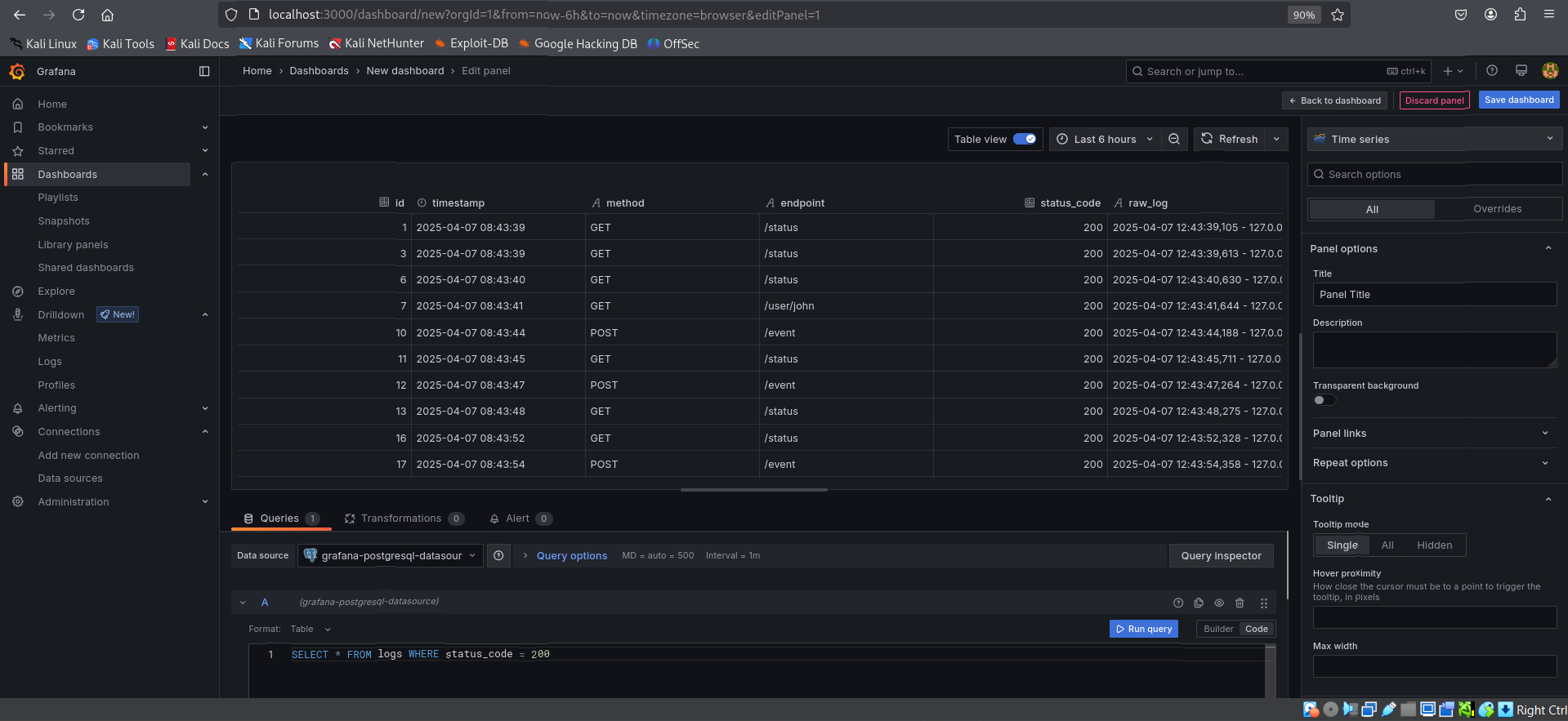


1. Write queries to extract required metrics and visualize them.









1. Implement monitoring mechanisms.

