# Setup Guide: DuckDB Data Integration with Grafana Using Apache and FastAPI

## Step 1: Install Required Software

Install Apache:  
On Ubuntu or Debian:  
 sudo apt update  
 sudo apt install apache2  
  
On CentOS or RHEL:  
 sudo yum install httpd  
  
Install Python, DuckDB, and FastAPI:  
Install Python (skip if already installed):  
 sudo apt install python3 python3-pip  
  
Install DuckDB, FastAPI, and Gunicorn via pip:  
 pip3 install duckdb fastapi gunicorn

## Step 2: Set Up DuckDB and Database Schema

Create DuckDB Database  
Open a Python or DuckDB shell to create a new database and a table for measurements:

import duckdb  
  
# Connect to DuckDB (this will create devices.duckdb if it doesn't exist)  
conn = duckdb.connect('devices.duckdb')  
  
# Create a measurements table  
conn.execute('''  
 CREATE TABLE IF NOT EXISTS measurements (  
 datetime TIMESTAMP,  
 longitude DOUBLE,  
 latitude DOUBLE,  
 device\_id VARCHAR,  
 value DOUBLE,  
 unit VARCHAR,  
 height DOUBLE  
 )  
''')

This will create a DuckDB database named `devices.duckdb` with a `measurements` table to store your incoming data.   
Make sure to include a `datetime` field to support time-series data for Grafana visualization.

## Step 3: Create the FastAPI Application

Create a Python file called `app.py` with the following content. This defines a FastAPI application that handles POST requests and inserts data into DuckDB:

from fastapi import FastAPI, Request, HTTPException  
import duckdb  
import json  
from datetime import datetime  
  
app = FastAPI()  
  
@app.post("/measurements")  
async def receive\_measurement(request: Request):  
 try:  
 # Open a DuckDB connection for each request  
 with duckdb.connect('devices.duckdb') as db\_connection:  
 # Parse incoming JSON data  
 data = await request.json()  
   
 # Convert timestamp if not provided  
 data['datetime'] = data.get('datetime', datetime.now().isoformat())  
  
 # Insert data into DuckDB  
 db\_connection.execute('''  
 INSERT INTO measurements (datetime, longitude, latitude, device\_id, value, unit, height)  
 VALUES (?, ?, ?, ?, ?, ?, ?)  
 ''', [data['datetime'], float(data['longitude']), float(data['latitude']),   
 data['device\_id'], float(data['value']), data['unit'], float(data['height'])])  
  
 return {"status": "success"}  
  
 except Exception as e:  
 raise HTTPException(status\_code=500, detail=str(e))

## Step 4: Run the FastAPI Application with Gunicorn

Run the following command to start the FastAPI application with Gunicorn, allowing it to handle multiple requests:  
  
 gunicorn -w 4 -k uvicorn.workers.UvicornWorker -b 127.0.0.1:8000 app:app

## Step 5: Configure Apache as a Reverse Proxy

Edit your Apache configuration file to proxy requests to the FastAPI app. Add the following configuration:  
  
<VirtualHost \*:80>  
 ServerName yourdomain.com # Replace with your actual domain or server IP  
  
 ProxyPass /measurements http://127.0.0.1:8000/measurements  
 ProxyPassReverse /measurements http://127.0.0.1:8000/measurements  
  
 ErrorLog ${APACHE\_LOG\_DIR}/error.log  
 CustomLog ${APACHE\_LOG\_DIR}/access.log combined  
</VirtualHost>  
  
Enable required modules and restart Apache:  
  
 sudo a2enmod proxy  
 sudo a2enmod proxy\_http  
 sudo systemctl restart apache2

## Step 6: Test the Setup

Use `curl` to test the setup and verify data insertion:  
  
 curl "http://yourdomain.com/measurements" -v -X POST -H "Content-Type: application/json" -d '{"longitude":"139.7449","latitude":"35.6617","device\_id":"47","value":"60","unit":"cpm","height":"111"}'

## Step 7: Pull Data into Grafana from DuckDB

To visualize the data stored in DuckDB on Grafana, follow these steps:  
  
1. \*\*Install the Simple JSON Data Source Plugin in Grafana\*\*  
  
 In Grafana, go to \*\*Configuration > Plugins\*\*, search for "Simple JSON" and install it. Alternatively, you can install it from the command line:  
  
 grafana-cli plugins install grafana-simple-json-datasource  
  
 After installation, restart Grafana:  
  
 sudo systemctl restart grafana-server  
  
2. \*\*Create the Python API to Serve Data to Grafana\*\*  
  
 Create a new Python file `grafana\_duckdb\_api.py` with the following code, modified to avoid connection locking:

from fastapi import FastAPI, HTTPException  
import duckdb  
from datetime import datetime  
  
app = FastAPI()  
  
@app.get("/search")  
async def search():  
 return ["value", "longitude", "latitude", "height"]  
  
@app.post("/query")  
async def query(request: dict):  
 target = request["targets"][0]["target"]  
   
 try:  
 with duckdb.connect('devices.duckdb') as db\_connection:  
 data = db\_connection.execute(f'''  
 SELECT datetime, {target} FROM measurements ORDER BY datetime  
 ''').fetchall()  
  
 return [{  
 "target": target,  
 "datapoints": [  
 [row[1], int(datetime.strptime(row[0], "%Y-%m-%d %H:%M:%S").timestamp() \* 1000)]  
 for row in data  
 ]  
 }]  
 except Exception as e:  
 raise HTTPException(status\_code=500, detail=str(e))  
  
@app.post("/annotations")  
async def annotations(request: dict):  
 return []

3. \*\*Run the API with Uvicorn\*\*  
  
 Run the API with the following command:  
  
 uvicorn grafana\_duckdb\_api:app --host 0.0.0.0 --port 8001  
  
4. \*\*Add the Simple JSON Data Source in Grafana\*\*  
  
 - In Grafana, go to \*\*Configuration > Data Sources\*\* and click \*\*Add data source\*\*.  
 - Select \*\*Simple JSON\*\* from the list.  
 - Set the \*\*URL\*\* to `http://localhost:8001` (or the appropriate IP/hostname if running on another server).  
 - Click \*\*Save & Test\*\* to confirm that Grafana can connect to your API.  
  
5. \*\*Create a Dashboard in Grafana\*\*  
  
 - Go to \*\*+ > Dashboard\*\* in Grafana and add a new panel.  
 - In the \*\*Query\*\* section, select the \*\*Simple JSON\*\* data source you configured.  
 - Choose a metric (e.g., "value" or "height") from the dropdown to visualize.  
 - Adjust the \*\*Visualization\*\* type and settings as desired to display your data.  
  
 Your Grafana dashboard should now visualize data stored in DuckDB in real time.

## Additional Step: Viewing DuckDB Data in DBeaver

To view and manage your DuckDB data in DBeaver, follow these steps:  
  
1. \*\*Install DBeaver\*\*:  
 Download and install DBeaver from https://dbeaver.io/.  
  
2. \*\*Add a DuckDB Driver in DBeaver\*\*:  
 - Open DBeaver, go to \*\*Database > Driver Manager\*\*, and click \*\*New\*\*.  
 - Name the driver `DuckDB`, set \*\*Driver Type\*\* to `Generic`, and add the DuckDB JDBC JAR file from https://github.com/duckdb/duckdb/releases.  
 - Click \*\*OK\*\* to save.  
  
3. \*\*Create a New Database Connection\*\*:  
 - Go to \*\*Database > New Database Connection\*\* and select \*\*DuckDB\*\*.  
 - Provide the path to `devices.duckdb` and click \*\*Finish\*\*.  
  
4. \*\*Explore Data\*\*:  
 - Expand the DuckDB connection in \*\*Database Navigator\*\*, right-click on `measurements`, and choose \*\*View Data\*\*.  
 - Run SQL queries in the SQL Editor to explore and analyze your data.  
  
This enables quick access to DuckDB data directly within DBeaver.