



# DATA ENG BRISBANE

FOR DATA ENGINEERS, BY DATA ENGINEERS.

<https://www.meetup.com/brisbane-data-engineering-meetup/>



# Building an IoT Platform using Modern Data Stack



Prefect, DuckDB, FastAPI

Brisbane Data Engineering Meetup

6 December 2023  
Francisco Liwa



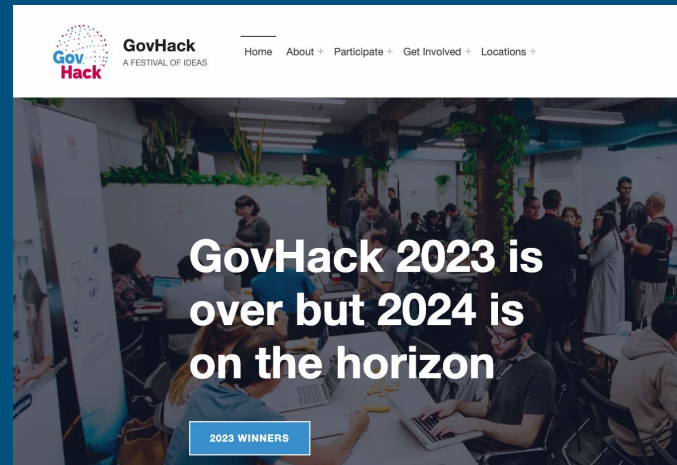
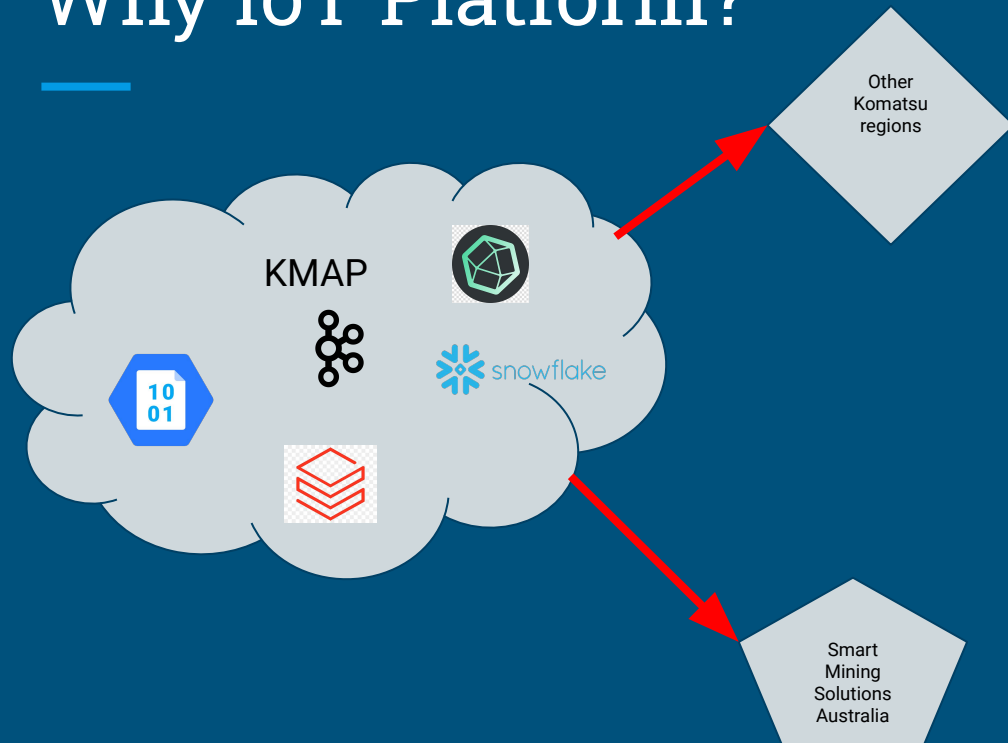
# About Me

---

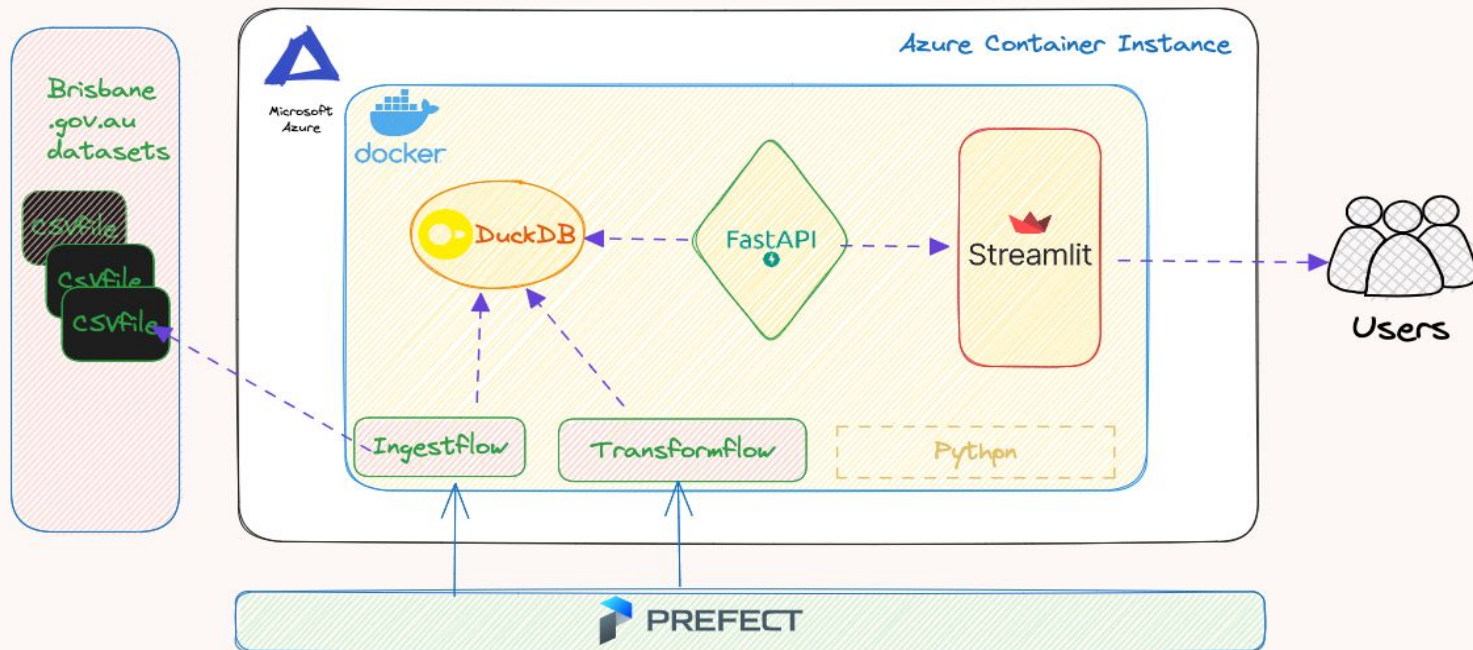


- Almost 20+ years in Software Engineering (NEC, IBM , TOSHIBA, SAMSUNG)
- 6+ years in Data Engineering space
- Love working with Startups - ( co-founded 3 in Philippines )
- Currently working as Senior Data Engineer at Komatsu Australia ( Smart Mining Solutions) - **Software and Data Engineering team lead**
- Trekking
- Basketball
- Chess

# Why IoT Platform?







# The GOAL



# Modern Data Stack (for my use case only)

---

- **PREFECT** - Workflow orchestration
-  **DuckDB** - Database / Data warehouse
-  **FastAPI** - API Layer
-  **Streamlit** - Visualization
-  - Deployment env

Bonus:

PDM ( Python Dependency Manager )





# Our Data

The screenshot shows the Brisbane City Council Data portal. The header includes the council logo, navigation links (Home, Insights, Data, More, Login), and a breadcrumb trail: Home > Datasets > Telemetry sensors — Rainfall and Stream Heights. The main content area features a blue sidebar with the dataset title and a Creative Commons Attribution 4.0 license icon. The main text describes the dataset as raw, unprocessed data from rainfall and stream height gauges. It lists four bullet points: collect real-time information on water levels and rainfall, provide information to trigger flood alerts and warnings, provide information to trigger flooded road signage/flashings lights, and provide historic flood information for use in hydrology and hydraulic modelling. Below the text are tags for 'api', 'flood', 'gauge', 'hydrometric', 'rainfall', 'stream height', 'telemetry', and 'water'. The 'Data and Resources' section contains two download options: 'Telemetry sensors — Rainfall and Stream heights — CSV' and 'Telemetry sensors — Rainfall and Stream heights — metadata — CSV', each with 'Preview' and 'Download' buttons.

Brisbane City Council

Home Insights Data More Login

Home > Datasets > Telemetry sensors — Rainfall and Stream Heights

Dataset Categories Activity Stream

### Telemetry sensors — Rainfall and Stream Heights

Creative Commons Attribution 4.0

Rainfall and Stream Height gauges owned by Brisbane City Council. This is raw, unprocessed data. Council installs and maintains telemetry gauges at various locations across Brisbane as part of its hydrometric network. These gauges form part of the Bureau of Meteorology Flood Warning Network. This Brisbane City Council gauge information (raw data) is passed onto the Bureau of Meteorology which they display on their [public website](#) via various interfaces.

The hydrometric gauges:

- collect real-time information on water levels and rainfall
- provide information to trigger flood alerts and warnings
- provide information to trigger flooded road signage/flashings lights
- provide historic flood information for use in hydrology and hydraulic modelling.

api flood gauge hydrometric rainfall stream height telemetry water

#### Data and Resources

Telemetry sensors — Rainfall and Stream heights — CSV Preview Download

Telemetry sensors — Rainfall and Stream heights — metadata — CSV Preview Download

Site - <https://www.data.brisbane.qld.gov.au/data/dataset/telemetry-sensors-rainfall-and-stream-heights>

## Metadata and Raw sensor data

<https://www.data.brisbane.qld.gov.au/data/dataset/01af4647-dd69-4061-9c68-64fa43bfaac7/resource/117218af-4adc-4f8e-927a-0fe43c46cd-b4/download/rainfall-and-stream-heights-metadata-20231205t110000.csv>

<https://www.data.brisbane.qld.gov.au/data/dataset/01af4647-dd69-4061-9c68-64fa43bfaac7/resource/78c37b45-ecb5-4a99-86b2-f7a514f0f447/download/gauge-data-20231205t183704.csv>

# Exploratory Data Analysis (EDA)

```
"Sensor ID","Location ID","Location Name","Sensor Type","Unit of Measure","Latitude","Longitude"
"E1809","540801","Rachele Close, Forest Lake","Rainfall","mm","-27.631476","152.953555"
"E1531","540118","Bancroft Park, Kelvin Grove","Stream Height AHD","m","-27.444481","153.005314"
"E1515","540099","Chadston Close, Kenmore Hills","Rainfall","mm","-27.504712","152.924885"
"E1512","540117","ABQ-2 Mt Coot -tha","Rainfall","mm","-27.463952","152.947533"
"E2020","540071","Corinda High School, Corinda","Rainfall","mm","-27.546892","152.988372"
"E1886","540800","Rosewood Place, Murarrie","Rainfall","mm","-27.465266","153.094268"
"E2111","40788","Johnson Rd (Adermann Br), Forestdale","Stream Height AHD","m","-27.655308","153.000406"
"E1594","540286","Mouth of Breakfast Ck, Newstead","Stream Height AHD","m","-27.441564","153.047296"
"E1561","540124","Burralong St, Deagon","Stream Height AHD","m","-27.334180","153.058401"
"E1560","540124","Burralong St, Deagon","Rainfall","mm","-27.334180","153.058401"
```

```
"Measured","E1809","E1531","E1515","E1512","E2020","E1886","E2111","E1594"
"2023-12-04T18:40:00","0","-","0","0","0","0","-","0.04","-0.12","0","0","-"
"2023-12-04T18:45:00","0","-","0","0","0","0","-","0.04","-","0","0","-"
"2023-12-04T18:50:00","0","-","0","0","0","0","-","-0.01","-0.17","0","0","-"
"2023-12-04T18:55:00","0","-","0","0","0","0","-","-","-","0","0","-"
"2023-12-04T19:00:00","0","-","0","0","0","0","-","-","-0.22","0","0","-"
"2023-12-04T19:05:00","0","-","0","0","0","0","-","-0.06","-0.27","0","0","-"
"2023-12-04T19:10:00","0","-","0","0","0","0","-","-0.11","-","0","0","-"
"2023-12-04T19:15:00","0","-","0","0","0","0","-","-0.16","-0.32","0","0","-"
"2023-12-04T19:20:00","0","-","0","0","0","0","-","-","-","0","0","-"
```

metadata					sensor data
1st file	"Measured","E1809", "2023-09-16T19:50:00","0.24"	....		"Measured","E1809", "2023-09-17T19:40:00","0.20"	
	next file	"Measured","E1809", "2023-09-16T20:00:00","0.20"	....		"Measured","E1809", "2023-09-17T19:50:00","0.21"

Note:

Automatically collected telemetry data for rainfall and stream height gauges owned by Brisbane City Council.

The dataset includes raw gauge readings in 5-minute increments covering a 24-hour rolling period. The data is updated every 10 minutes. Dataset includes gauge readings, descriptions and location details.



# Our ELT pipeline -



# PREFECT

Prefect orchestrates workflows — it simplifies the creation, scheduling, and monitoring of complex data pipelines. With Prefect, you define workflows as Python code and let it handle the rest.

## Basic concepts in PREFECT:

- **Task** - a python function that handle workload
- **Flows** - series of tasks ( inorder) . Think of Airflow DAGs. calls tasks, and other flows ( subflows )
- **Server** - orchestration engine - could be locally of or hosted in the cloud
- **Agents** - daemon process running on local/cloud/container which executes flows
- **Deployments** - A server-side concept that encapsulates flow metadata, allowing it to be scheduled and triggered via API.
- **Pools** - prioritize and manage deployment runs and control the infrastructure they run on.

<https://www.prefect.io/>

# How to PREFECT?

Note: Prefect requires Python 3.8 or newer.

1. Install  
`pip install prefect`
2. Write your 'flows' in python
3. Start Server  
`prefect server start` -> Self-hosted  
`prefect cloud login` -> Prefect Cloud
4. Deploy flow and run deployment  
`-`python flow.py``  
`- prefect deployment run 'hello_universe/first-deploy'`
5. Start agent pool  
`prefect agent start --pool "default-agent-pool"`

```
from prefect import flow, task
```

```
@task(log_prints=True)
def say_hello(name: str):
    print(f"Hello {name}!")
```

```
@flow
def hello_universe(names: list[str]):
    for name in names:
        say_hello(name)
```

```
if __name__ == "__main__":
    # create your first deployment to automate your flow
    hello_universe.deploy(name="your-first-deployment")
```

# Our Database -



# DuckDB

- An in-memory/in-process OLAP SQL database
- SQL on top of your csv and parquet files
- No dependency cli - **duckdb cli**
- Python API
- Lightweight
- <https://shell.duckdb.org/>
- <https://duckdb.org/docs/archive/0.9.2/api/cli>

## Why DuckDB?



### Simple and portable

- In-process, serverless
- C++11, no dependencies, single-file build
- APIs for Python, R, Java, Julia, Swift, ...
- Runs on Windows, Linux, macOS, OpenBSD, ...

[more](#) →



### Feature-rich

- Transactions, persistence
- Extensive SQL support
- Direct Parquet, CSV, and JSON querying
- Joins, aggregates, window functions

[more](#) →



### Fast

- Optimized for analytics
- Vectorized and parallel engine
- Larger than memory processing
- Parallel Parquet, CSV, and NDJSON loaders

[more](#) →



### Free and extensible

- Free & open-source
- Permissive MIT License
- Flexible extension mechanism

[more](#) →

<https://duckdb.org/>

# How to make your DB quack!

## 1. Install or Download

- `brew install duckdb`
- `https://github.com/duckdb/duckdb`
  - `/releases/download/v0.9.2/duckdb_cli-osx-universal.zip`
- `pip install duckdb==0.9.2`

## 2. Create a .duckdb file

- `touch demo.duckdb`

## 3. Connect to duckdb

```
import duckdb
duckdb.connect(database=db, read_only=is_shared)
```

## 4. Query Data

```
./duckdb
D.open db.datahack.duckdb
D.databases
D.tables
```

```
import duckdb

def get_db_conn(db=':memory:', is_shared=False):
    return duckdb.connect(database=db, read_only=is_shared)

conn = get_db_conn('db/datahack.duckdb')

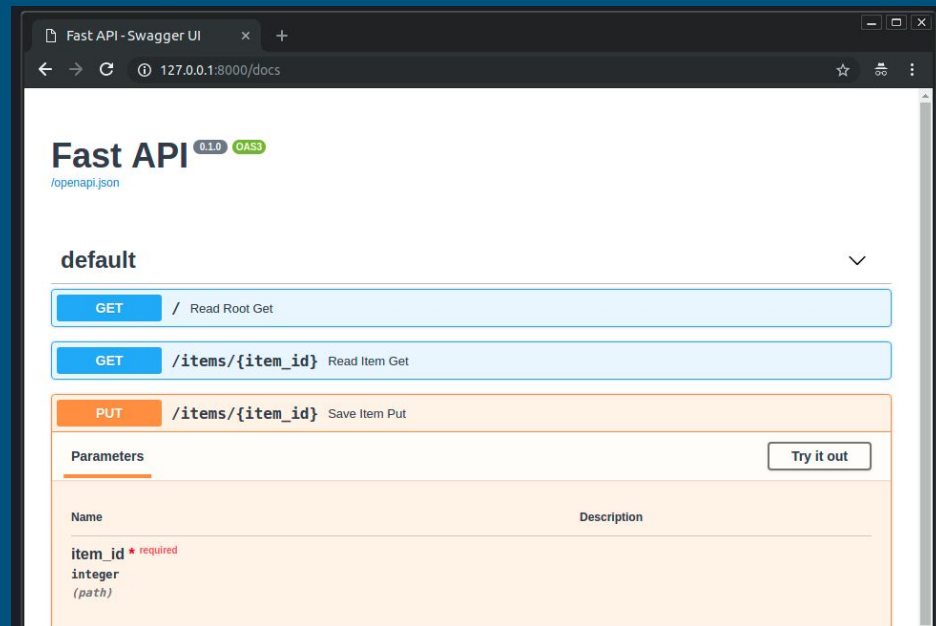
conn.execute(
    """
    DROP TABLE IF EXISTS gauge_sensors_metadata;
    CREATE TABLE gauge_sensors_metadata AS
    SELECT *
    FROM read_csv_auto

('src/data/metadata/rainfall-and-stream-heights-metadata-20230917t110000.csv')
    """
)
```

# Our API - FastAPI

- Modern Python-based API Framework
- Very High performance
- Built-in Swagger documentation
- Robust and fast framework
- Fast to code
- Based on Open API

<https://fastapi.tiangolo.com/>



# How fast is FastAPI?

1. Install fastapi package

```
pip install fastapi
```

2. Install uvicorn package

```
pip install "uvicorn[standard]"
```

3. Import fastapi module

```
from fastapi import FastAPI
```

4. Run server

```
uvicorn fast-demo:app --reload
```

```
from typing import Union
```

```
from fastapi import FastAPI
```

```
app = FastAPI()
```

```
@app.get("/")
```

```
def read_root():
```

```
    return {"Hello": "World"}
```

```
@app.get("/items/{item_id}")
```

```
def read_item(item_id: int, q: Union[str, None] = None):
```

```
    return {"item_id": item_id, "q": q}
```

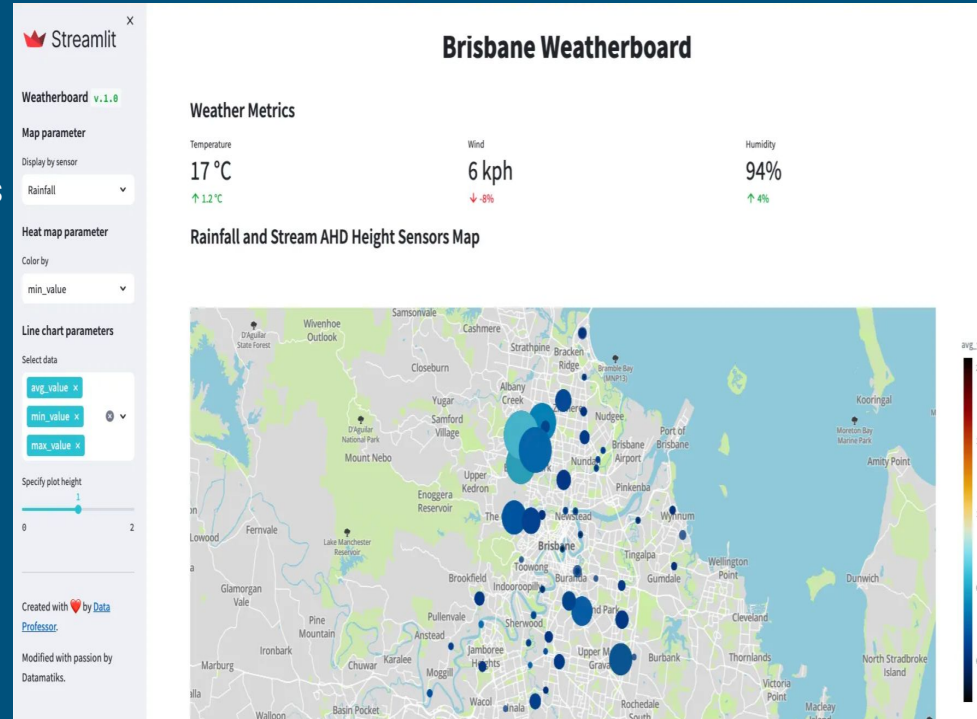


# Our Visualization

- A Python web framework for creating Data Apps
- Turns data scripts into shareable web apps in minutes.
- Opensource
- Deploy locally or cloud
- Bought by Snowflake

<https://streamlit.io/>

template- <https://youtube.com/dataprofessor/>



# How to Streamlit?

---

1. Install streamlit

```
pip install streamlit
```

2. Create streamlit app in python
3. Execute python code

```
streamlit run myapp.py
```

```
import streamlit as st
import pandas as pd
```

```
st.write("""
# My first app
Hello *world!*
""")
```

```
df =
pd.read_csv("src/data/metadata/rainfall-and-stream-heights-
metadata-20231205t110000.csv")
st.line_chart(df)
```



Thank You!