



Get to zero carbon, fast.

InsightGlobal

Walmart

DOORDASH

BlackRock

Figma

BAILLIE GIFFORD

7Wise

airbnb

Match Group

Grover

Sonder



BLOCK

Misfits Market

athenahealth

ZAXBY'S



monzo

CSWR  
Central States Water Resources

BEAUTYCOUNTER

EVERLANE

BOMBAS

TaskUs

BOOM

Klarna

shopify

dataiku

BOB'S DISCOUNT FURNITURE

WARBY PARKER

Spotify

GOJO | Purell  
GOJO, MAKERS OF PURELL™

okta

sweetgreen

Oakley Capital

FORTRESS

General Mills

stripe

servicenow

sunrun

THOMABRAVO

SKIMS

kainos

BainCapital

Wolt



InsightGlobal

Walmart

DOORDASH

BlackRock

Figma

BAILLIE GIFFORD

7Wise

airbnb



Grover

Sonder



BLOCK

Misfits Market

ath

CSWR  
Central States Water Resources

BEAUTYCOUNTER

EVERLANE

B

Klarna

shopify

data  
iku

BO

GOJO | Purell  
GOJO, MAKERS OF PURELL™

okta

sweetgreen

Oakley

General  
Mills

stripe

servicenow

sunrun

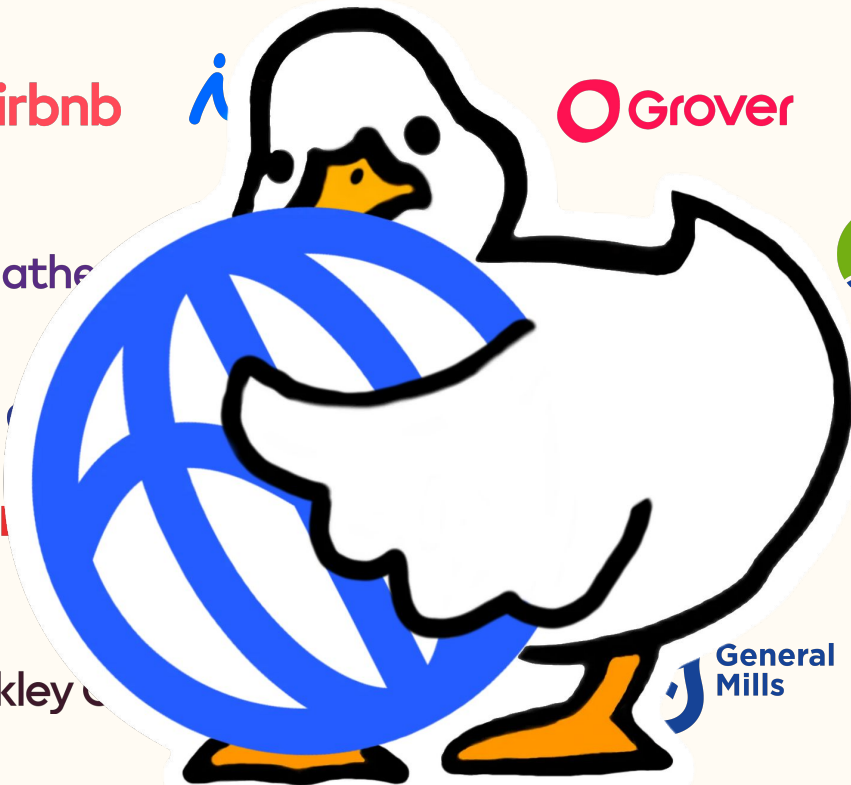
THOMABRAVO

SKIMS

kainos

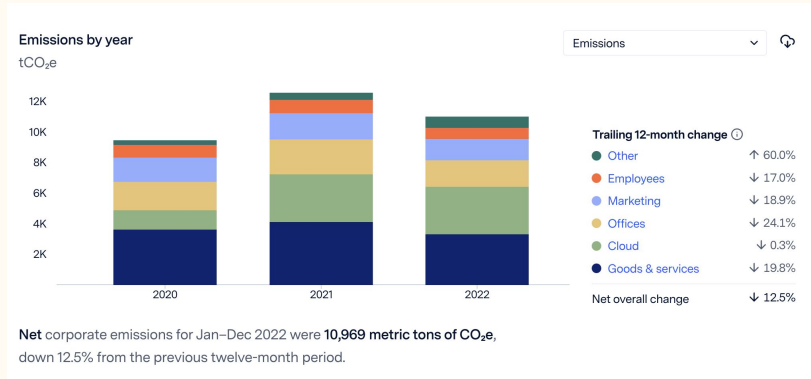
BainCapital

Wolt



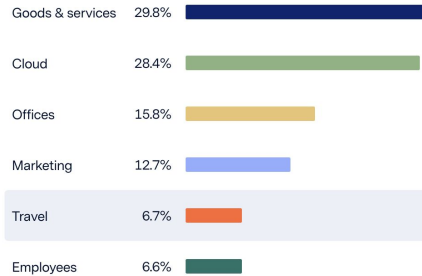
# Carbon footprint data

# User needs: fast aggregates



## Emissions by category

Goods & services, cloud, and offices were the biggest drivers of your emissions during Jan–Dec 2022.



## Travel →

Emissions from employee business travel (flights, hotels, meals, ground transportation).



# Medium-sized data

- 12% of customers have footprints with > 1m rows
- Largest customer has >15m rows → ~750mb parquet



# Medium-sized data

- 12% of customers have footprints with > 1m rows
- Largest customer has >15m rows → ~750mb parquet



# Medium-sized data

- 12% of customers have footprints with > 1m rows
- Largest customer has >15m rows → ~750mb parquet
- Fits on one machine, but non-trivial to query performantly





Existing solution: Postgres



# Existing solution: Postgres

- 162GiB table / 252GiB of Postgres database size



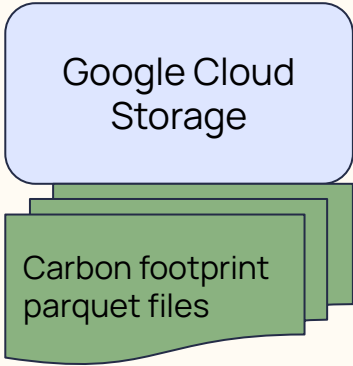
# Existing solution: Postgres

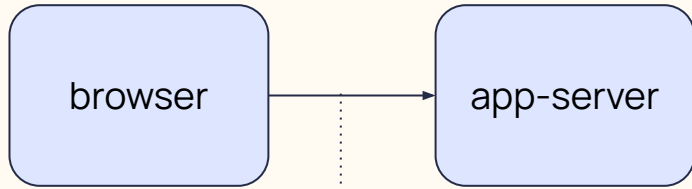
- 162GiB table / 252GiB of Postgres database size
- Painful migrations
- Arbitrary analytic queries don't scale



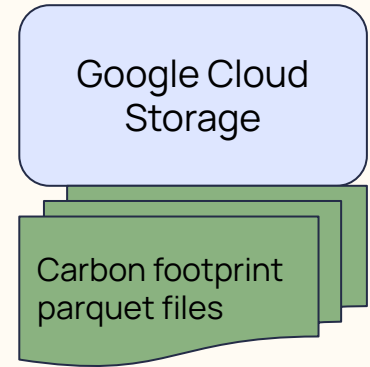
# Architecture

browser



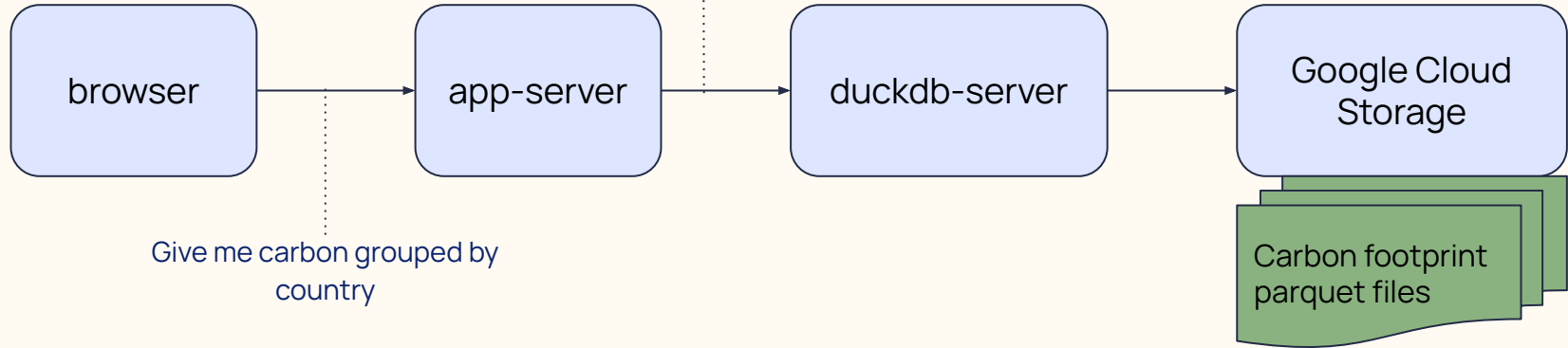


Give me carbon grouped by country



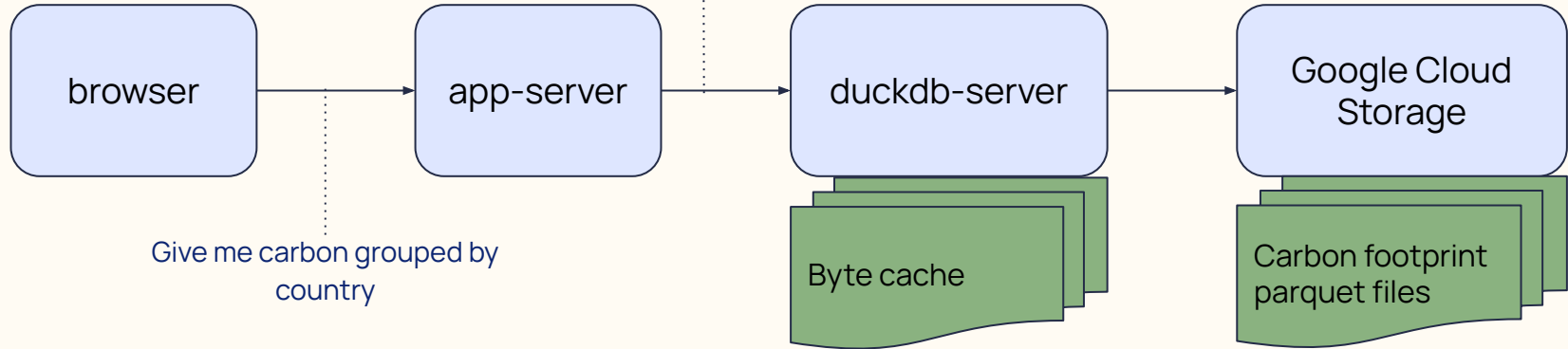
```
SELECT sum(kgco2e), country
FROM footprint
GROUP BY country
```

footprint:  
gs://watershed/airbnb/footprint.parquet



```
SELECT sum(kgco2e), country
FROM footprint
GROUP BY country
```

footprint:  
gs://watershed/airbnb/footprint.parquet





# Why we love this

- It's fast!
  - 10x faster than Postgres with lots of indexes
  - 100s of ms for our P99 data size
- No more giant table!
- Less adhoc caching
- Parquet ❤️



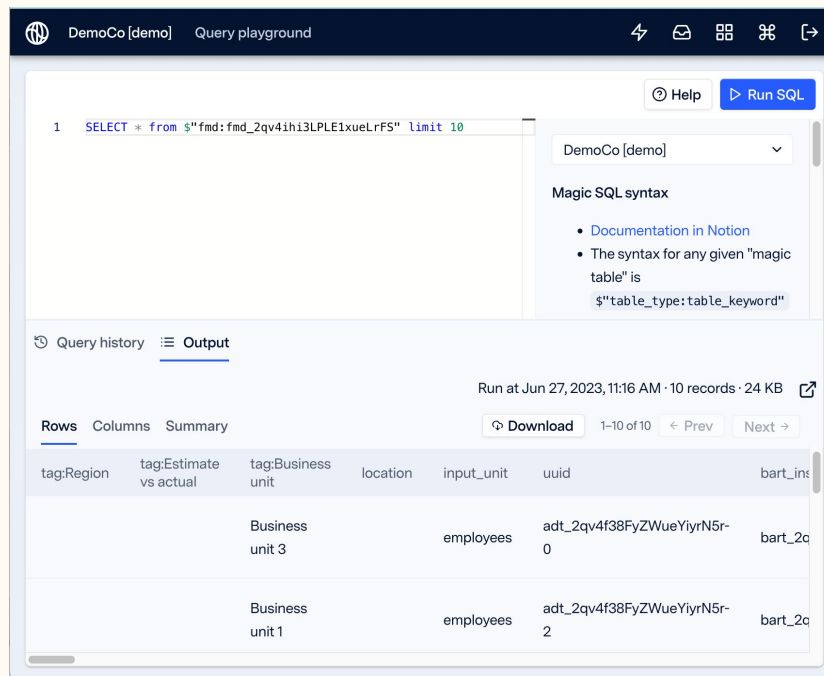
# Other uses

- Data pipeline: activity data → carbon footprint data
- Query any parquet file
- ~75k duckdb-server queries per day



# Other uses

- Data pipeline: activity data → carbon footprint data
- Query any parquet file
- ~75k duckdb-server queries per day



The screenshot shows a web-based query playground interface for DuckDB. The top bar includes the logo, the text "DemoCo [demo] Query playground", and navigation icons. The main area contains a SQL query editor with the text: `1 SELECT * from $"fmd: fmd_2qv4ih13LPLE1xueLrFS" limit 10`. To the right of the editor is a "Run SQL" button and a "Help" icon. Below the editor, there is a "Magic SQL syntax" section with a dropdown menu set to "DemoCo [demo]" and a list of links: "Documentation in Notion" and "The syntax for any given 'magic table' is '\$'table\_type:table\_keyword'".

Below the editor, there is a "Query history" and "Output" section. The "Output" section shows the results of the query, including a "Download" button, a "1-10 of 10" indicator, and "Prev" and "Next" navigation arrows. The results are displayed in a table with the following columns: tag:Region, tag:Estimate vs actual, tag:Business unit, location, input\_unit, uuid, and bart\_ins.

tag:Region	tag:Estimate vs actual	tag:Business unit	location	input_unit	uuid	bart_ins
		Business unit 3		employees	adt_2qv4f38FyZWueYiyRN5r-0	bart_2c
		Business unit 1		employees	adt_2qv4f38FyZWueYiyRN5r-2	bart_2c



# Thank you!

[jessica@watershed.com](mailto:jessica@watershed.com)