ANALYTICS

A Serverless Query Engine f
Spare Parts

An open-source implementation of a Data I
DuckDB and AWS Lambdas

Ciro Greco
Apr 27, 2023 11 min read

A duck in the cloud. Photo by László Glatz on Unsplash

In this post we will show how to build a simple end-to-end application in the cloud on a serverless infrastructure. The purpose is simple: we want to show that we can develop directly against the cloud while minimizing the cognitive overhead of designing and building infrastructure. Plus, we will put together a design that minimizes costs compared to modern data warehouses, such as Big Query or Snowflake.

As data practitioners we want (and love) to build ap
top of our data as seamlessly as possible. Whether
BI, Data Science or ML all that matters is the final a
how fast you can see it working end-to-end. The inf
often gets in the way though.

Imagine, as a practical example, that we need to bu
customer-facing analytics application for our produc
Because it's client-facing we have performance con
need to respect, such as low latency.

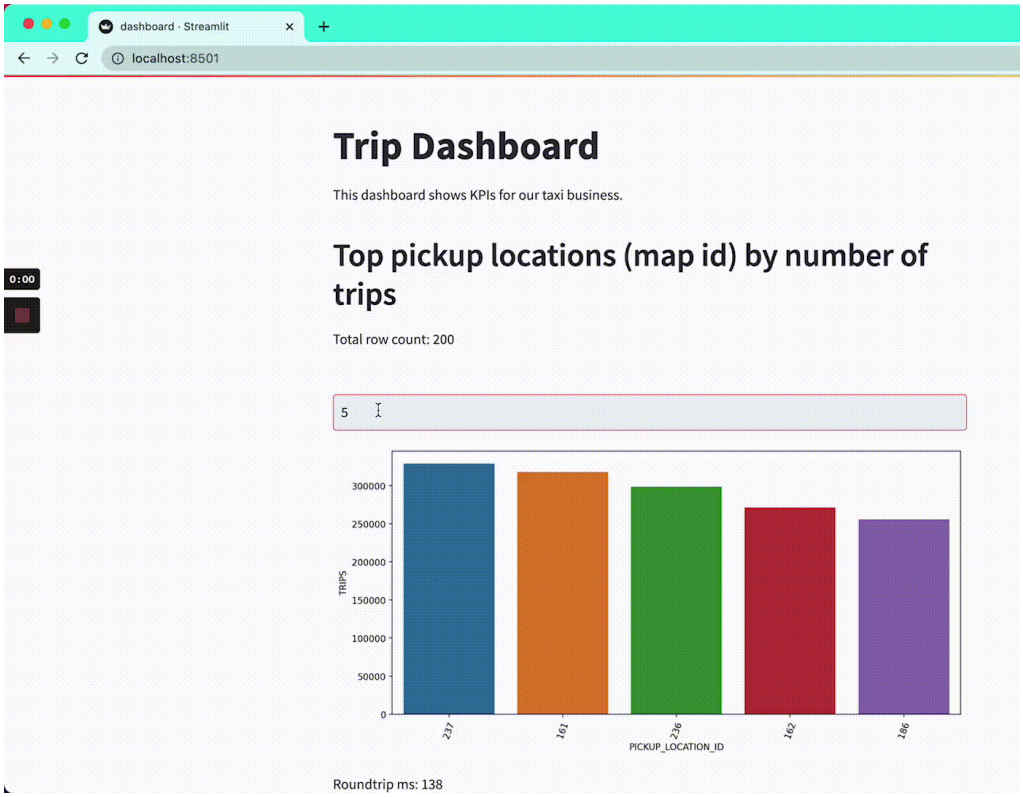
We can start developing it directly in the cloud, but
immediately bring us to some infra questions: wher
it? How big a machine do we need? Because of the
requirement, do we need to build a caching layer? I
we do it?

Alternatively, we can develop our app locally. It will
more intuitive from the developer experience point
only postpones the infra questions, since in the enc
to find a way to go from our local project to actual
Moreover, the data will need to leave the cloud env
machine, which is not exactly secure and auditable.

To make the cloud experience as smooth as possib
designed a data lake architecture where data are si

simple cloud storage (AWS S3) and a serverless infrastructure that embeds DuckDB works as a query engine. At the end of the cycle, we will have an analytics app that can be used to both visualize and query the data in real time with virtually no infra costs.

Of course, this is a bit of a simplification, as some tweaks would be needed to run this project in real production sce What we provide is a general blueprint to leverage the se storage and compute to build a data lake with a qu the cloud. We show how to power an interactive da an (almost) free cloud endpoint, no warehouse setu lighting fast performance. In our implementation, th application is a simple Streamlit app, but that's me explanatory purposes: you can easily think of pluggi favorite BI tool.



A lightinign fast analytics app built with our system. Image from the a

Ducks go serverless

Y'all know DuckDB at this point. It is an open-source in-process SQL OLAP database built specifically for analytical queries. It is somewhat still unclear how much DuckDB is actually used in production, but for us today the killer feature is the possibility of querying parquet files directly in S3 with SQL syntax.

So most practitioners seem to be using it right now as a local engine for data exploration, *ad hoc* analysis, POCs and prototyping (with some creative ideas on how to ex purpose to cover more surface). People create note small data apps with embedded DuckDB to prototy experiment with production data locally.

The cloud is better. And if we often feel it isn't, it's l something is wrong with the tool chain we use, but very big difference between a bad idea and a good executed.

If we combine a data lake architecture, a serverless DuckDB and a bit of ingenuity we can build a very fa from spare parts: no warehouse setup, lighting fast and outrageously cheap costs – S3 most expensive pricing is $0.023 per GB, AWS Lambda is very fast a

zero when not in use, so it is a no-fat computation bill, plus AWS gives you 1M calls for free.

Buckle up, clone the repo, sing along and for more details, please refer to the README.

Architecture

This project is pretty self-contained and requires or
introductory-level familiarity with cloud services and

The idea is to start from a Data Lake where our dat
Once the data is uploaded in our S3 in a parquet fo
then trigger the lambda with a SQL query. At that p
lambda goes up, spins up a DuckDB instance in me
computes the query and gets back to the user with
which can be directly rendered as tables in the Terr

The architecture has a number of advantages mostl
from the serverless design: speed, proximity to the
line deployment are nice; plus, of course, the syste
zero when not used, so we only pay per query.

General architecture of our system. Image from the authors

Your first query engine + data lake from spare part

We provide a simple script that will create an S3 bu
populate it with a portion of the NYC TLC Trip Recor
(available under the nyc.gov terms of use), both as a
and as a hive-partitioned directory (you can run it w
Once the data are in the data lake we can set up ar
lambda: if you have the Serverless CLI setup correc
the lambda is one command of Make again.

The lambda can be invoked in any of the usual ways
a query as its main payload: when it runs, it uses Du
using an instance if on a warm start) to execute the
data lake. DuckDb does not know anything about th
and after the execution, making the lambda purely s
far as data semantics is concerned.

For instance, you can use the simple Python script i
to send this query to the lambda, and display the re

```
SELECT
  pickup_at as pickup_time,
  dropoff_at as dropoff_location,
  trip_distance
FROM read_parquet(your_s3_bucket/dataset/taxi_2019_04.parquet')
WHERE pickup_at >= '2019-04-01' AND pickup_at < '2019-04-03'
LIMIT 10
```


The results are visualized directly in your terminal. Image from the au

Et voilà! You can now query your data lake, securely
This very simple design addresses directly two of t
frictions for working in cloud data warehouses:

- The setup is significantly simplified since all the
to do is to have her AWS credential. Once the s
the user only needs access to the lambda (or a
it!): that is good, as it gives the user full query c
without access to the underlying storage.
- The performances are so good that it feels like
locally, even if we always go through the cloud.
cloud experience helps tame the too familiar fe
advantages of working on remote machines is p
coin of good developer experience.

(Almost) Free Analytics

It's all good and boujee, but let us say that we want to do a bit more than query data on the fly. Let's say that we want to build an application on top of a table. It's a very simple app, there is no orchestration and no need to calibrate the workload.

At the same time, let's say that this application needs to be responsive, it needs to be fast. Anyone who deals w directly, for instance, knows how it is important to p fresh responsive experience to the clients who war data. The one thing nobody likes is a dashboard tha minutes to load.

To see how this architecture can bridge the gap bet pipelines and real-time querying for analytics, we pi dbt DAG to simulate running some offline SQL trans over the original dataset resulting in a new artifact i lake (the equivalent of a dashboard view).

To keep things as self contained as possible, we inc version that you can run locally on your machine (se more details – nor dbt nor the engine behind it mat pattern to work). However, you can use a different r dbt and export the final artifact as a parquet file, w or BigQuery.

Architecture of our system paired with a dbt project. Image from the a

For the time being, we'll stick to our super simple D two nodes. The first node takes the pickup_locatior data lake and order them by the number of trips:

```
SELECT
  pickup_location_id AS location_id,
COUNT(*) AS counts
FROM read_parquet(['s3://{{ env_var('S3_BUCKET_NAME
GROUP BY 1
```

The second that gives as the top 200 pick up locati data set:

```
SELECT
  location_id,
  counts
FROM {{ ref('trips_by_pickup_location') }}
ORDER BY 2 DESC
LIMIT 200
```

We can visualize the DAG with dbt docs:

Once our pipeline is done, the final artifact is uploa
our data lake in:

```
s3:/your_s3_bucket/dashboard/my_view.parquet
```

We can then reuse the query engine we built before
second (and final) node of our DAG to visualize the
Streamlit app, simply by running in the terminal:

```
(venv) cirogreco@Ciros-MBP src % make dashboard
```

Every time we hit the dashboard, the dashboard hit
behind the scenes. If you like this simple architectu
pattern can be used in your own Streamlit app, or i
BI tool.

A few remarks on the "Reasonable Scale"

A while ago, we wrote a series of posts on what we
MLOps at Reasonable Scale where we talked about
strategies to build reliable ML applications in comp
not process data at internet scale and have a numb

constraints that truly Big-Data companies typically do not have. We mostly talked about it from the point of view of ML and MLOps because operationalizing successfully ML was a major problem for organizations at the time (maybe it still is, I am not sure), but one general observation remains: most data organizations are "Reasonable Scale" and they should design their systems around this assumption. Note that being a reasonable scale organization does not necessarily small company. The enterprise world is full of data deal with a lot of complexity within large – sometim – organizations and yet have many reasonable scale often for internal stakeholders, ranging from a few a TB.

Recently, we happily witnessed a growing debate ar whether companies need Big Data systems to deal data problems,. The most important takeaway from view remains that, if you are a Reasonable Scale org dealing with unnecessary infrastructure can be a ve burden with plenty of nefarious ramifications in you processes. You could in principle build an entire dat support low latency dashboards – maybe you could Warehouse and a caching layer -, but since your res limited, wouldn't it be nice to have a simpler and ch

In this post, we showed that the combination of da storage formats, on-demand compute and in-memo processing opens up for new possibilities at Reason The system is far from perfect and could use many improvements, but it shows that one can build an i data app with no warehouse setup, lighting fast per and virtually no costs. By removing the db from Duc combine what is fundamentally right about local ("s

processing is all you need") with what is fundamentally right about the cloud ("data is better processed elsewhere").

I am spending most of my time thinking about serverless data infra. If you are interested, you have feedback on this post or simply want to chat, drop me a line at ciro.greco@bauplanlabs.com

WRITTEN BY

Ciro Greco

See all from Ciro Greco

Analytics | AWS Lambda | Data Engineering | Ducl | Serverless

Share This Article

Related Articles

ANALYTICS

Methods for Modelling Customer Lifetime Value: The Good Stuff and the Gotchas

Part three of a comprehensive, practical guide to CLV techniques and real-world use-cases

Katherine Munro
November 17, 2023 12 min read

DATA ENGINEERING

Feature Engineer Microsoft Fabric Gen2

Fabric Madness part 3

Roger Noble
April 15, 2024 13 min re

BASKETBALL

Feature Engineering with Microsoft Fabric and PySpark

Fabric Madness part 2

Roger Noble
April 8, 2024 14 min read

DATA ENGINEERING

Data: Where Engi Science Meet

Our weekly selection Editors' Picks and orig

TDS Editors
September 29, 2022 4

ANALYTICS

SQL Mastery: Advanced Techniques for Data Professionals

Elevating Your Data Skills with Window Functions, Regex, and CTEs

Mia Dwyer

April 1, 2024 7 min read

ANALYTICS

Building a Data P… 2021

How to build a moder… platform to power you… data science…

Dave Melillo

March 7, 2021 7 min re…

ANALYTICS

Top 3 non-machine learning skills to rise in Kaggle competitions

Data, creativity, and tactic will make you climb the leaderboard

Pranay Dave

June 9, 2022 7 min read

Subscribe to Our Newsletter

Your home for data science and AI. The world's leading publication for data science, data analytics, data engineering, machine learning, and artificial intelligence professionals.