Thank you sponsors!

INSPARK

twintos

ilionx

infoSupport
Solid Innovator

redgate

sopra steria

DATA GIANTS

BRANDER
company

SQL brander.nl
DATA brander.nl
CRM brander.nl
ERP brander.nl

Bull
Master your AI destiny

dbWatch

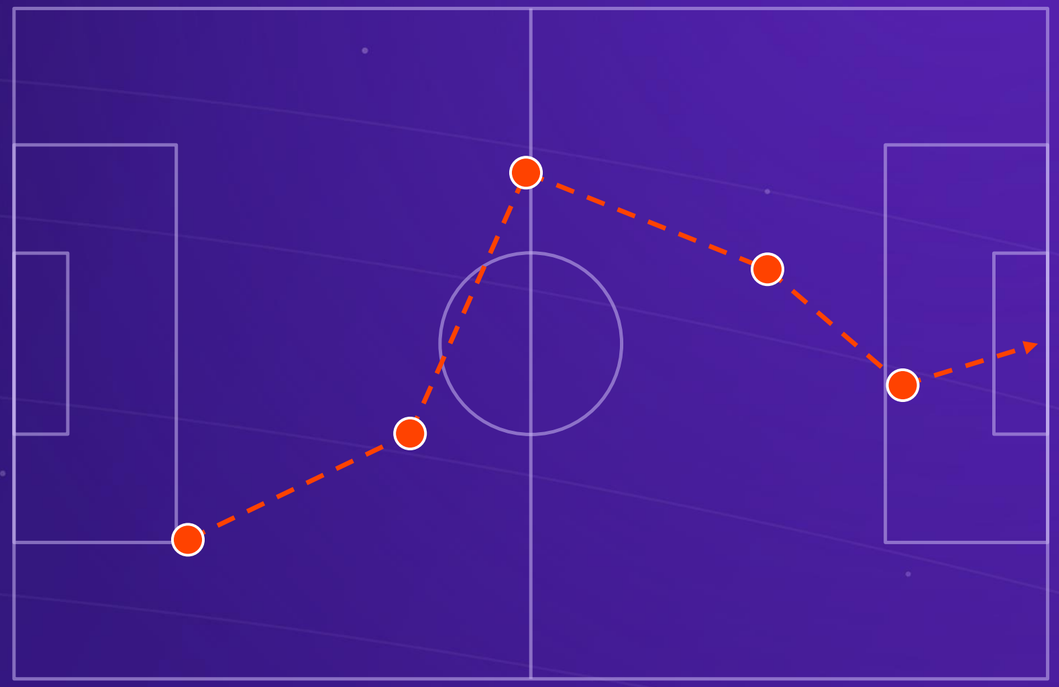
BBTG

Data Saturdays

DATA SATURDAY HOLLAND 2026

The Limitless Data Platform

The Rise of Kubernetes



Jesse Gorter

About me



Jesse Gorter

Data Architect & Principal Consultant at BBTG

- Years of Spark in a box: Synapse, Fabric and Databricks
- Football data enthusiast: Slotbal
- This platform: a side project I built in a month or two

Quick poll

Hands up

1



Who runs their data platform on Fabric, Databricks or Snowflake?

2



Who knows what it costs per month?

3



Who could move away in a week or month?

Managed SaaS platforms are great right?

I have built plenty on them, but you do give something up



Cost

Capacity and compute pricing that grows with every workload



Lock-in

Proprietary features, formats and orchestration



Control

Runtimes, networking and upgrades on someone else's schedule



Flexibility

The architecture follows the product, not your needs

Why ask this now?

A few things changed in the last years that make it worth asking



Engines

DuckDB handles millions of rows on a single machine, no cluster needed



Exit plans

Sooner or later someone asks how you would move to another platform



Formats

Delta Lake and Iceberg keep your data readable outside any one product



Kubernetes

Every big cloud offers managed Kubernetes, and it runs on-premises too

A personal experiment

Where I come from

SaaS Spark in the box

- Synapse, Fabric, Databricks
- Scaling is easy
- Cluster start-up is slow
- The product makes many design choices for you

This experiment

Kubernetes as a blank canvas

- Pick what works, drop what doesn't
- DuckDB instead of a hosted Spark cluster
- One small node, compute only when it runs
- You run into the interesting problems yourself

Picking every piece myself turned out to be the fun part.

THE QUESTION

Should Kubernetes become the foundation of your modern data platform?

What are we going to see today?

How the platform works, what I learned, and then we run it

The platform

- Why Kubernetes, and what AKS does for you
- The setup
- The architecture: Medallion

What I learned

- What it costs
- What went wrong along the way
- When Kubernetes is a good fit, and when not

Live demo

- A Job running on AKS
- StatsBomb corrects the World Cup final

Six Kubernetes terms

You will hear these for the rest of the talk

Container an app and its dependencies, packaged as an image
`this lab: the platform image`

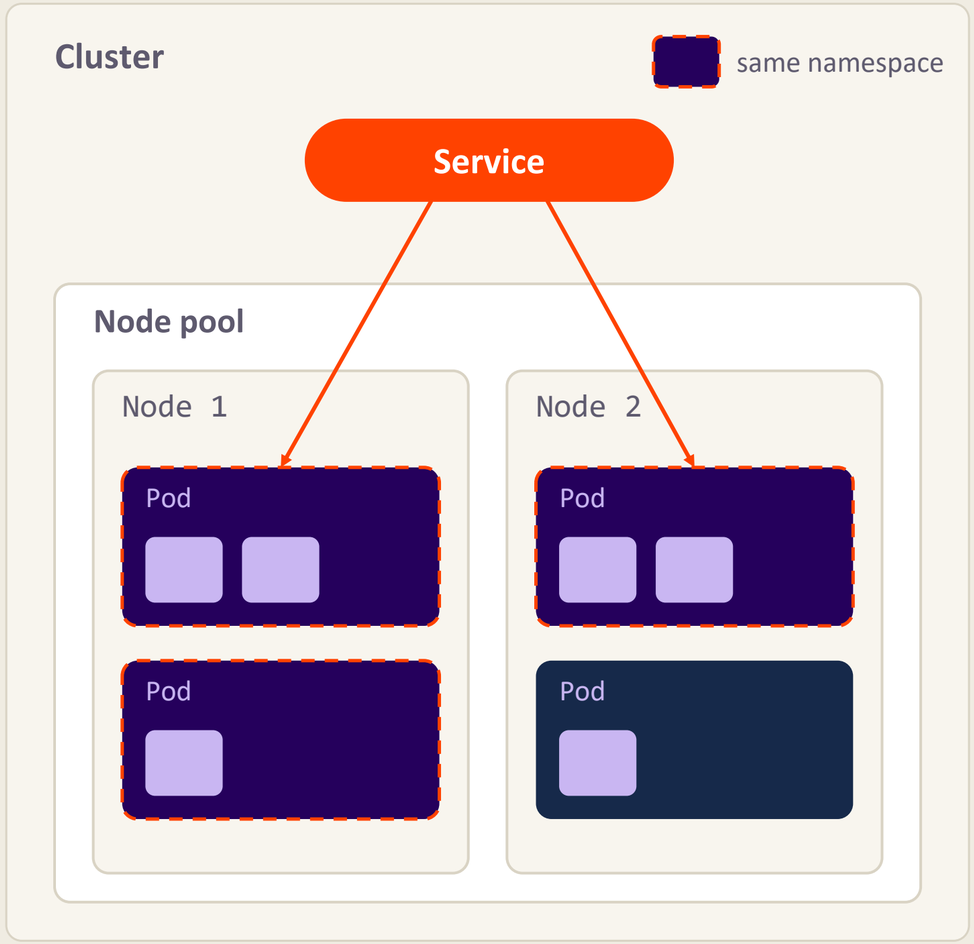
Pod the smallest thing Kubernetes runs: one or more containers
`this lab: one pod per Job run`

Node a virtual machine that runs pods
`this lab: 1 × Standard_B2s`

Node pool a group of nodes with the same VM size (AKS)
`this lab: node pool sys`

Namespace a logical boundary for resources inside the cluster
`this lab: namespace dataplatform`

Service a stable address in front of a set of pods
`this lab: none, batch Jobs serve no traffic`



But isn't Kubernetes hard?

Some of it was easier than I thought, some of it was not

Easier than expected

- AKS runs the control plane, I only manage my own nodes
- Two YAML manifests describe every Job
- A few scripts start, run and stop everything
- The same image runs on my laptop and in AKS

Still real work

- No SQL endpoint at this moment
- No catalog or lineage out of the box
- Monitoring is still on my list
- Need to have Kubernetes skills in your organization
- Setting up AKS properly with private networking takes time and skills – but so does Databricks

When is Kubernetes realistic?

You don't have to go SaaS. But you do own everything that breaks

Good fit

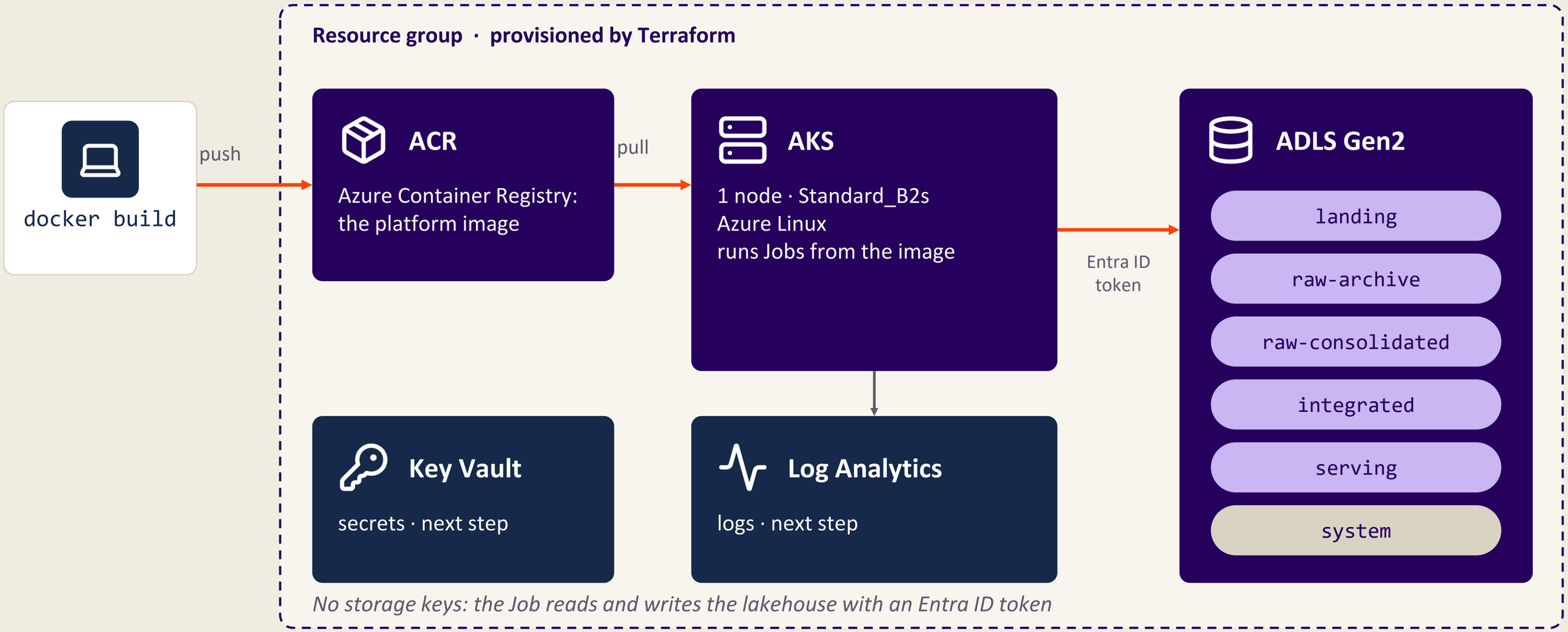
- You want control over cost, runtime and network
- Portability matters (exit strategy)
- The team is comfortable with containers and infrastructure as code

Think twice

- No one wants to own platform operations
- You need governance, catalog and lineage out of the box
- Many analysts expect a notebook-and-click experience
- If you like low-code
- If portability is not an issue

The platform on Azure

Terraform provisions everything in one resource group



What Kubernetes gives a data platform

Most of what I needed was already built in

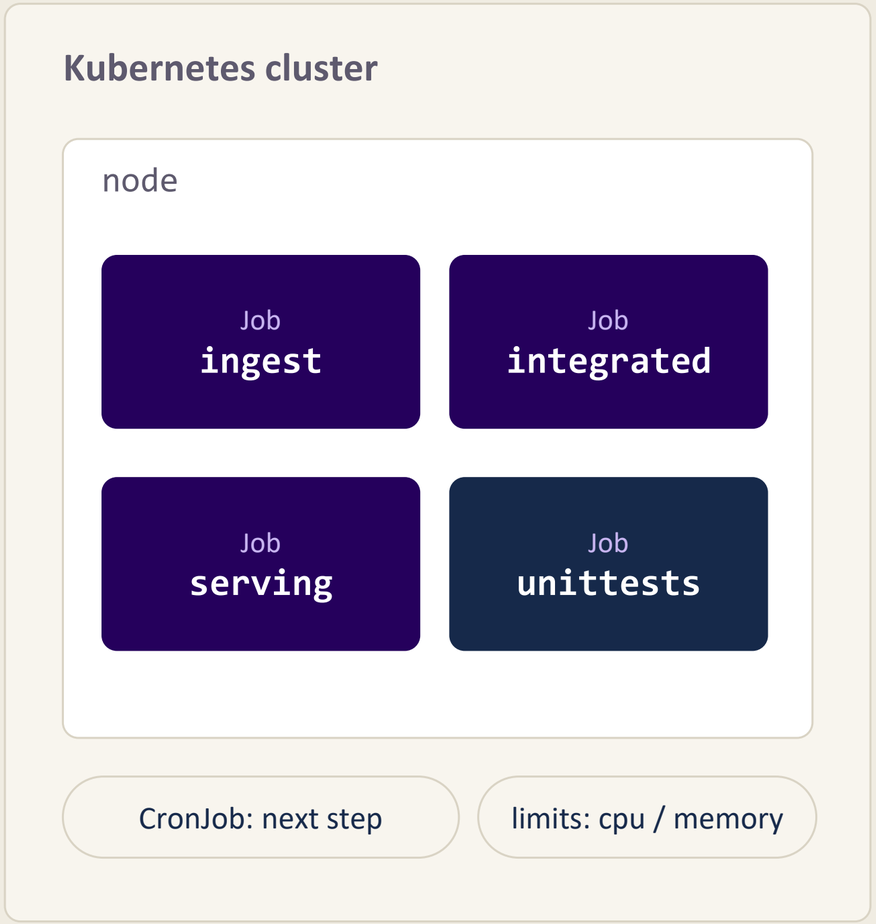
Containers
every workload is a reproducible image

Jobs & CronJobs
batch processing and scheduling built in

Resources
requests and limits per workload, nodes that scale

Identity
each workload can get its own identity

Portable
AKS, OpenShift, or on-premises Kubernetes: same manifests



FROM JSON TO POWER BI

The lakehouse

The data: StatsBomb open data

Real matches, deeply nested JSON, and plenty of surprises

- Competitions and matches
- Lineups
- Events: thousands per match
- 360 freeze frames

~4,200

matches, thousands of events each

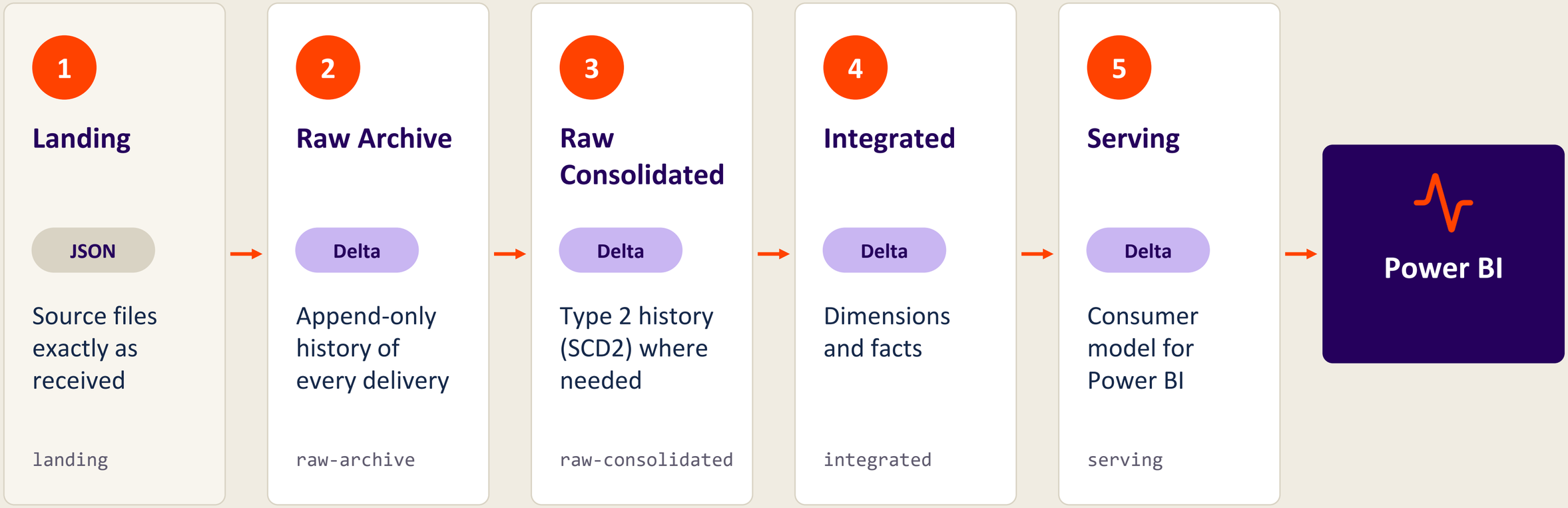
landing layout

```
statsbomb/
├─ competitions/competitions.json
├─ matches/<competition_id>/<season_id>.json
├─ lineups/<match_id>.json
├─ events/<match_id>.json
└─ three-sixty/<match_id>.json
```

Events carry a nested struct per event type: pass, shot, carry, …

Five layers, one direction

Every layer is its own storage container



Data only moves to the right. No layer ever writes back.

Let's start with the result

Real football data, one small Kubernetes node, €44 in September

Players

Per 90 minutes, for players with at least 270 minutes

Competition: All

Team: All

Goals vs xG per 90 (min. 270 minutes)



Player comparison

player_display_name	Appearances	Minutes Played	Goals per 90
Agnes Beever-Jones	22	880	1.23
Cristiane	4	299	1.21
Victoria Dellaperuta	11	607	1.19
Jennifer Echegini	17	851	1.06
Zlatan Ibrahimović	56	4,389	1.03
Jairzinho	7	623	1.01
Salma Paralluelo	32	1,984	1.00
Joseba Llorente	7	452	1.00
Francesco Totti	13	456	0.99
Folarin Balogun	4	274	0.99
Imanol Agirretxe	20	1,372	0.98
Ewa Pajor	22	1,769	0.97
Niclas Füllkrug	10	473	0.95
Khadija Shaw	24	2,027	0.93
André Hahn	15	775	0.93
Sam Kerr	49	3,821	0.92
Gelson	5	396	0.91
Lautaro Martínez	12	498	0.90
Gonzalo Higuaín	49	3,821	0.90
Total	**121,214**	**8,834,732**	**0.12**

Delta Lake stores, DuckDB computes

Open format on object storage, in-process engine on one node

Delta Lake

`via deltalake / delta-rs`

- Open table format on object storage
- Transactions and schema evolution
- Time travel
- No vendor

Arrow
in between

DuckDB

`in-process SQL engine`

- No cluster to run
- Reads Delta tables as Arrow datasets
- Fast on a single node
- Plain SQL for every transformation

Need Spark later? Only a few functions touch the engine. The YAML and SQL would stay as they are.

A table is a YAML file and a SQL file

The YAML says how to load it - The SQL says what comes out

`metadata/integrated/statsbomb/fact_events.yml`

```yaml
inputs:
  - raw_archive_events
strategy: partitioned
partition_key: match_id
batch_size: 100
```

`integrated/statsbomb/players.sql`

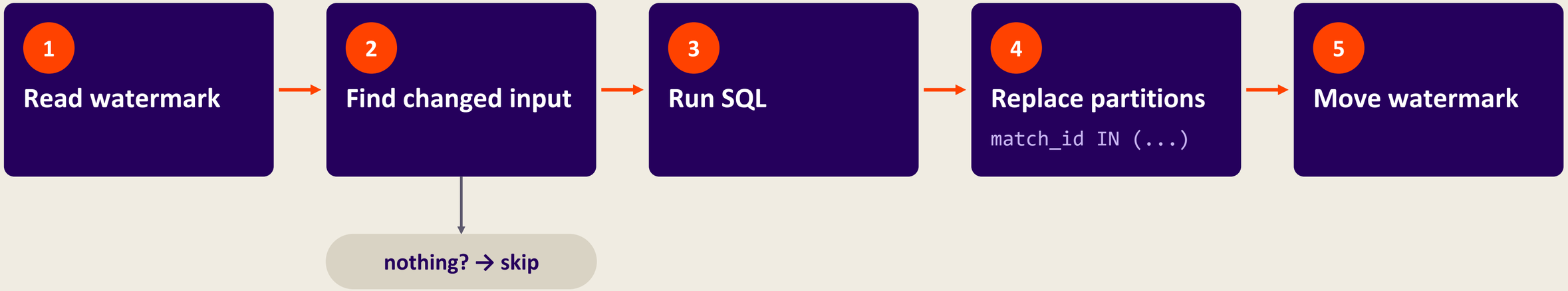
```sql
SELECT player_id, player_name,
       player_nickname, country_name
FROM raw_archive_match_lineups
QUALIFY ROW_NUMBER() OVER (
    PARTITION BY player_id
    ORDER BY match_id DESC) = 1
```



Python loads the files. Everything after that is SQL.

Run it twice, same result

Every step remembers what it already did



Every raw row records _source_file and _source_modified: a redelivered file is detected.

The load and its registration are one Delta commit.

Watermarks live in system/processing-state.

Running a pipeline twice gives the same result.

Serving: shaped for Power BI

One star model, and nothing Power BI has to fix itself

The model

`data product matchevents`

- Dimensions: match, team, player
- Facts: events, shots, players
- Freeze frames filter via shots
- Every filter flows one way

Import
mode

The numbers

`what Power BI imports`

- ~15 million events
- ~108,000 shots, all with context
- 360 aggregated per event
- Only columns the report uses

Business rules live in SQL

LET'S RUN IT

Live demo

1. A Job running on AKS
2. StatsBomb corrects the World Cup final
3. Back to the report

What bit me

The problems that cost me the most time

A rust panic in my Python pipeline

WHAT HAPPENED

Writing several event files at once crashed deltalake.

THE LESSON

Combine chunks: Delta Lake cannot handle sliced chunks of nested columns

The token that expires after an hour

WHAT HAPPENED

Long runs failed halfway: the Azure access token expired after one hour.

THE LESSON

Long jobs must refresh their token. That fix is next.

Load smart, not everything in one go

WHAT HAPPENED

Local machine and AKS lab had small compute, so needed to load smarter. Even single nodes can do a lot of work when dividing the work.

THE LESSON

Partition your data wisely, don’t try to ingest too much in one go

Takeaways

1 If portability matters to you, Kubernetes is a realistic foundation. Treat the platform as software.

2 Delta and DuckDB took me much further than I expected.

3 What made it reliable: loads you can safely rerun, and tests that describe behaviour.

THANK YOU

Questions?

Jesse Gorter · Data Architect

Let's connect on LinkedIn

Rate this session



Thank you sponsors!

INSPARK

twintos

ilionx

InfoSupport
Solid Innovator

redgate

sopra steria

DATA GIANTS

BRANDER
company

SQL brander.nl
DATA brander.nl
CRM brander.nl
ERP brander.nl

Bull
Master your AI destiny

dbWatch

BBTG

Data Saturdays