pack:wise

Streamlit for Rapid MVPs - Experiences and Next Steps

Dresden Datenbankforum | Packwise GmbH | Datum 14.Januar 2026

Wer spricht hier?

- **Marc Siggelkow**
- **Wirtschaftsinformatik an der HTW Dresden**
- **Data Scientist / Analyst**

marc.Siggelkow@packwise.de
www.packwise.io

pack:wise

01 Packwise

02 Das Problem

03 Streamlit

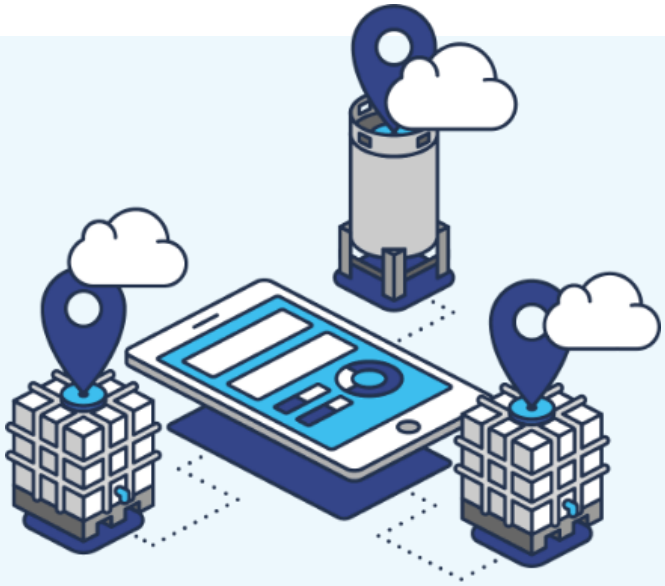
04 Examples

05 Next Steps

01 Packwise

Packwise

Containerflotten intelligenter steuern - dank **Packwise Flow**



pack:wise
digitize. manage. **match.**

- **Gegründet 2017 in Dresden**
- **Fokus: Industrial IoT (IIoT) für die Prozessindustrie.**
- **Über 60 internationale Großkunden (Chemie, Lebensmittel, Logistik).**
- **Aktive Container-Flotte in 20+ Ländern auf 5 Kontinenten.**

Mission: Making the Supply Chain transparent and autonomous.

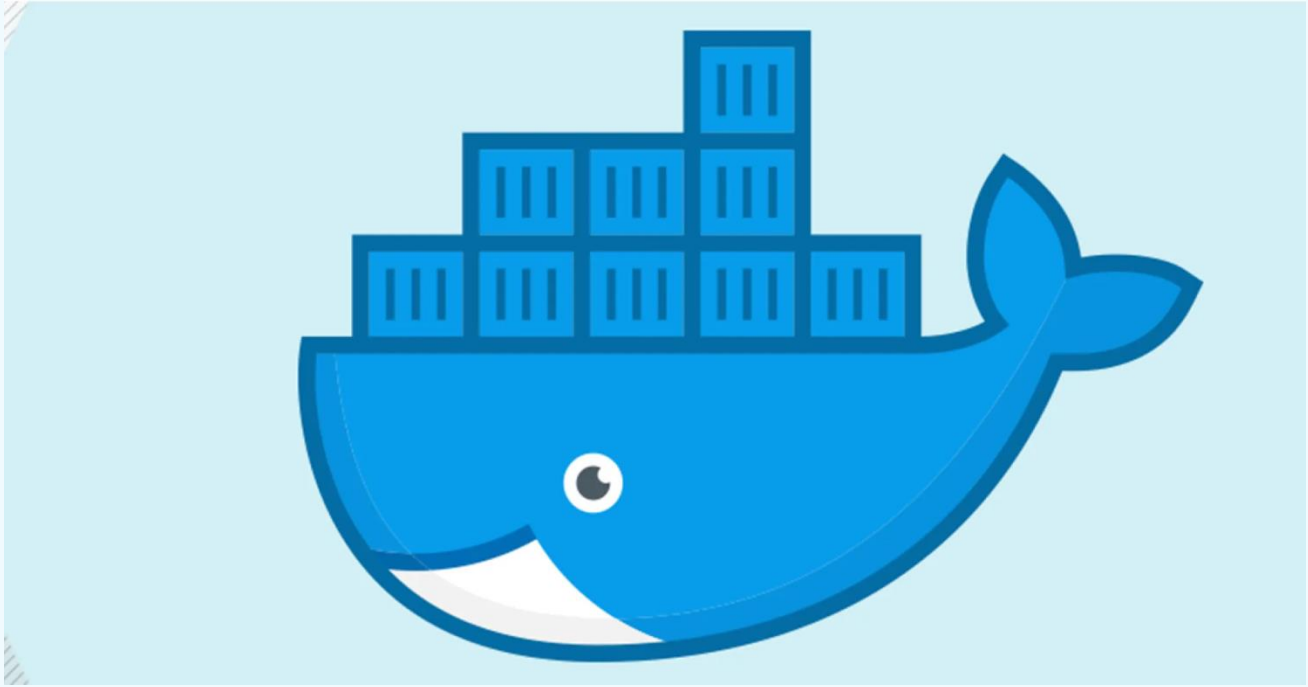
Packwise

Welche Container tracken wir?

… nicht diese

Packwise

Welche Container tracken wir?



... auch nicht diese

Packwise

Welche Container tracken wir?







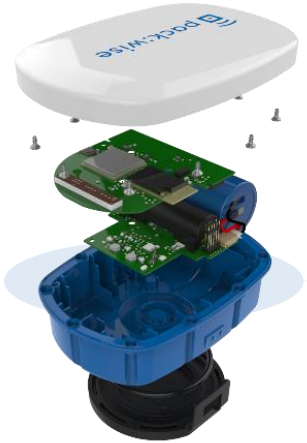

... und viele mehr!

pack:wise *flow*

pack:wise *smart cap*

Packwise

Die Smartcap – das IoT-Device



Packwise Smart Cap
Create a unique data set

Pressure

Location

Temperature

Damage

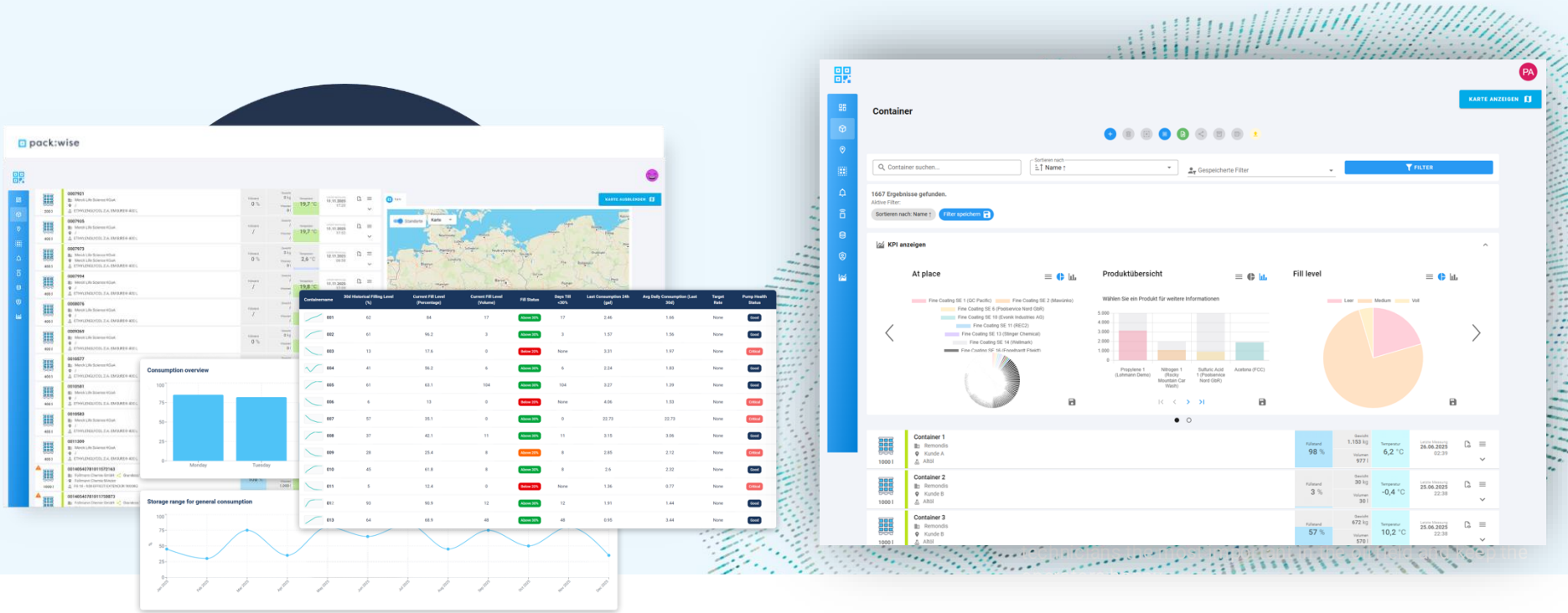
Movement

Filllevel

Packwise Flow
Turn it into valueable insights

Packwise

Packwiseflow – die Software Anwendung



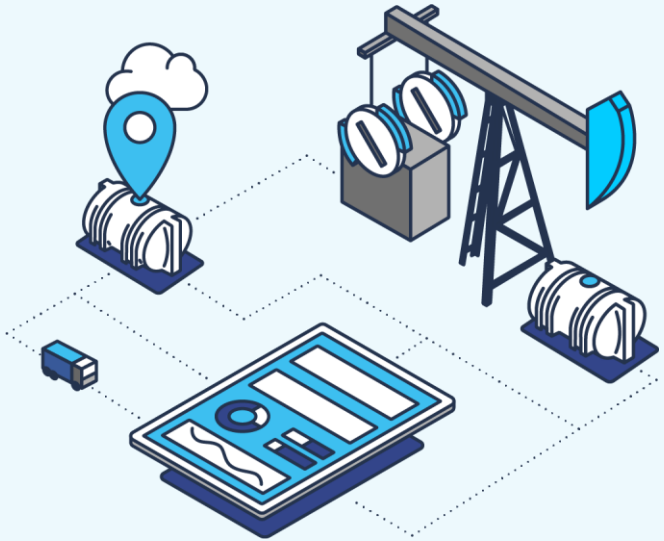
02

Das Problem

Das Problem

Der Ausgangszustand

- **Erreicht:** Volle Transparenz über Position (GPS) und Füllstand (Radar).
- **Problem:** Reine Darstellung von Ist-Zuständen ohne Kontext.
- **Status:** „Data Rich, Insight Poor“ – Wir haben die Daten, aber noch keine Antworten.
- **Ziel:** Den Sprung vom passiven Monitoring zur aktiven Logistiksteuerung schaffen.



Das Problem

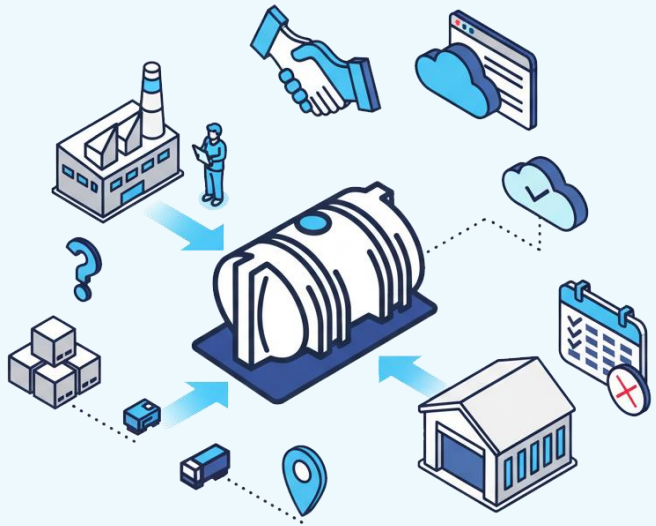
Status Que

- **Deskriptive Daten sind vorhanden:** Wir wissen, *wo* der Container ist und *wie viel* drin ist (Status Quo).
- **Die „So-What?“-Lücke:** Kunden sehen Punkte auf einer Karte, können aber ihre Supply Chain noch nicht aktiv steuern.
- **Das versteckte Potentzial:** Die DB füllt sich mit wertvollen Zeitreihen, aber die geschäftskritischen Antworten (Umlaufgeschwindigkeit, Bedarfsprognose) fehlen.

Das Problem

Der nächste Schritt

- **Nicht-artikulierte Anforderungen:** Kunden können ihren Bedarf erst benennen, wenn sie erste Analysen sehen.
- **Explorationsphase:** Welche KPIs (z. B. Umlaufgeschwindigkeit, Restlaufzeit-Prognose) liefern den größten Business-Value?
- **Das Risiko des „Over-Engineering“:** Monate an Entwicklung für ein Feature zu investieren, das am Ende niemand nutzt.



03

Streamlit

Streamlit

Was ist das?

Definition: Open-Source Python-Framework zur Erstellung interaktiver Web-Apps für Data Science.
Das Paradigma: Scripting statt Callbacks

- Keine Trennung von Frontend (HTML/CSS/JS) und Backend.
- Die gesamte App ist ein einziges Python-Skript.
- Reaktives Ausführungsmodell: Bei jeder Nutzerinteraktion wird das Skript von oben nach unten neu ausgeführt.

Core Features:

- **UI als Code:** Widgets (Slider, Buttons, Inputs) werden direkt als Variablen definiert (x = st.slider(...)).
- **Native Integration:** Nahtlose Unterstützung für Pandas, Plotly, Altair, Matplotlib und PyTorch.

Streamlit

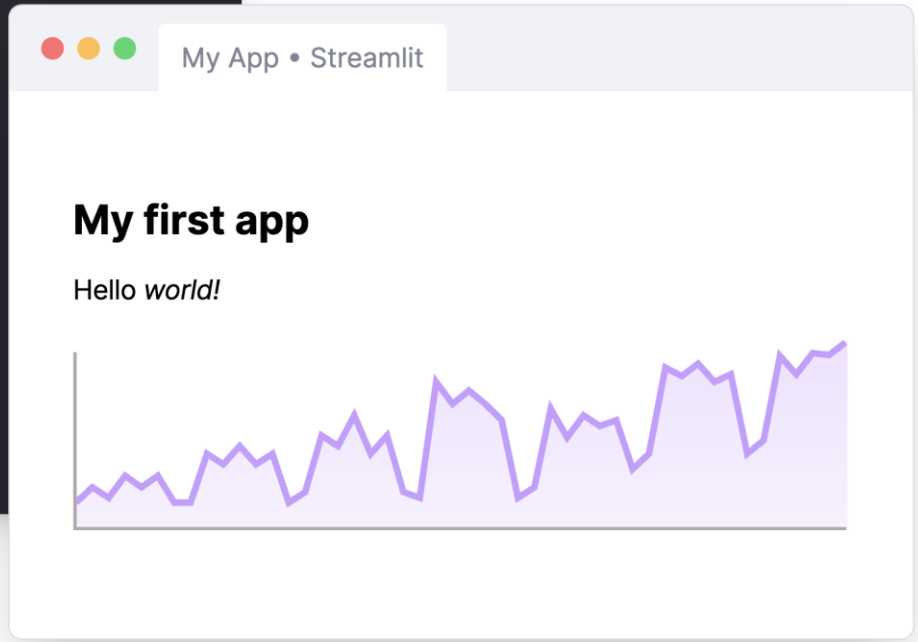
Simple Streamlit App

MyApp.py

```
import streamlit as st
import pandas as pd

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

df = pd.read_csv("my_data.csv")
st.line_chart(df)
```



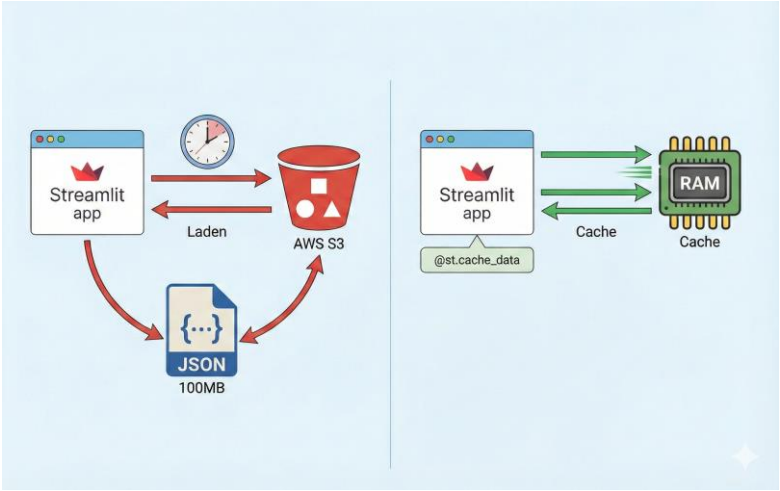
Streamlit

st.cache_data

- **Das Problem:** Große Datensätze (z.B. 100MB+ Json File von S3) führen bei jedem Re-Run zu Latenzen.
- **Naive Lösung:** Daten bei jedem Klick neu laden -> Schlechte UX, hohe S3-Kosten, Server-Last.

Die Lösung: @st.cache_data

- Speichert das Ergebnis einer Funktion im RAM.
- **Hashing-Prinzip:** Streamlit prüft die Eingabeparameter. Sind sie gleich, wird das Ergebnis sofort aus dem Cache geladen.



Streamlit

Statemanagement

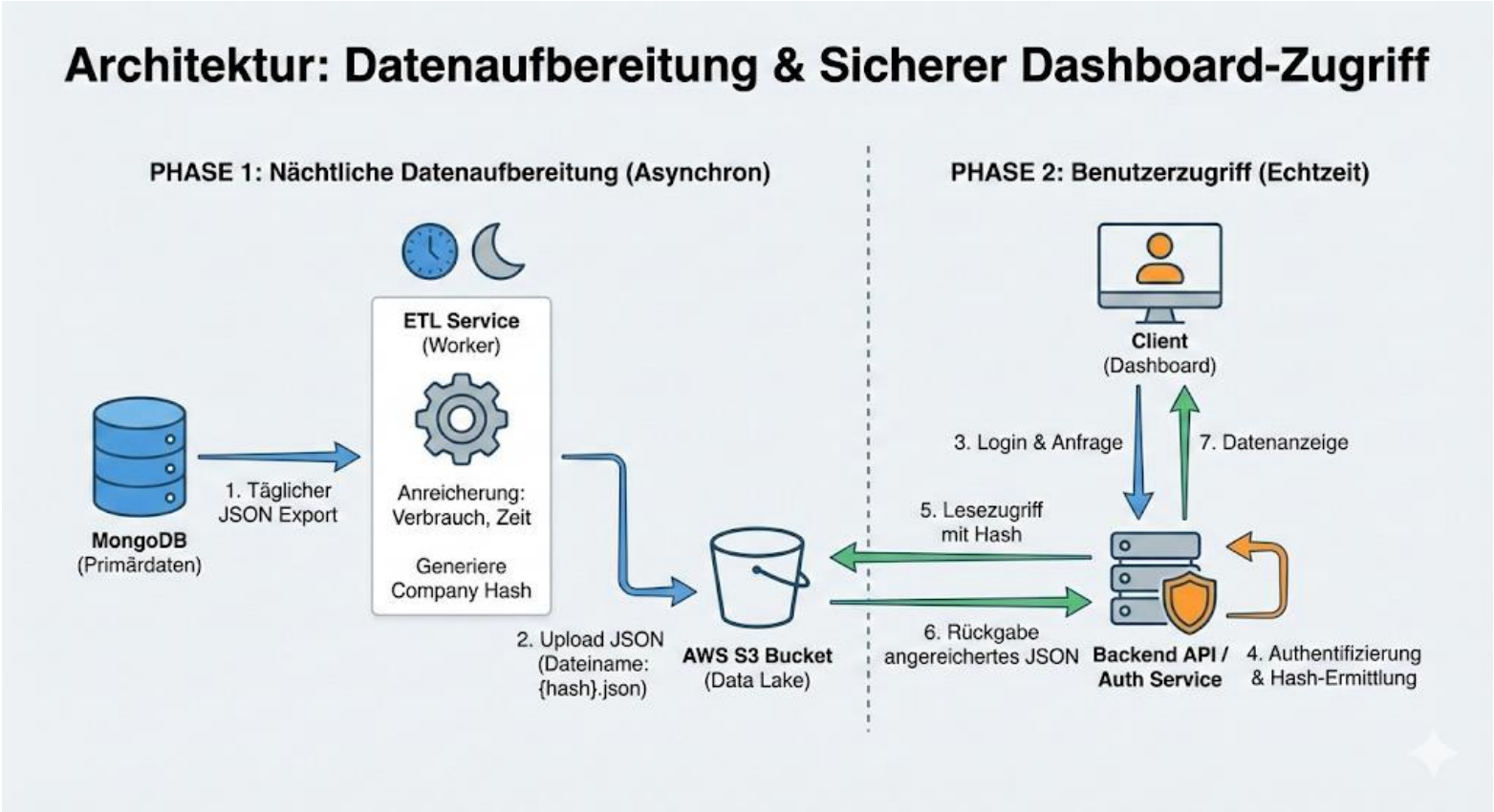
- **Das Re-Run Prinzip:** Jede Interaktion (Button, Slider, Input) löst eine komplette Neu-Ausführung des Python-Skripts aus.

- **Die Herausforderung:** Lokale Variablen gehen bei jedem Re-Run verloren.

- **Die Lösung: st.session_state**
  - Ein Dictionary-ähnliches Objekt, das über die gesamte Nutzer-Session hinweg persistent bleibt.
  - Ermöglicht komplexe Workflows (z.B. Multi-Step-Filter, Login-Status, Caching von User-Eingaben).

Streamlit

Die Architektur




pack:wise
04
Examples

Oil & Gas

Messbare Mehrwerte & Effizienzgewinne

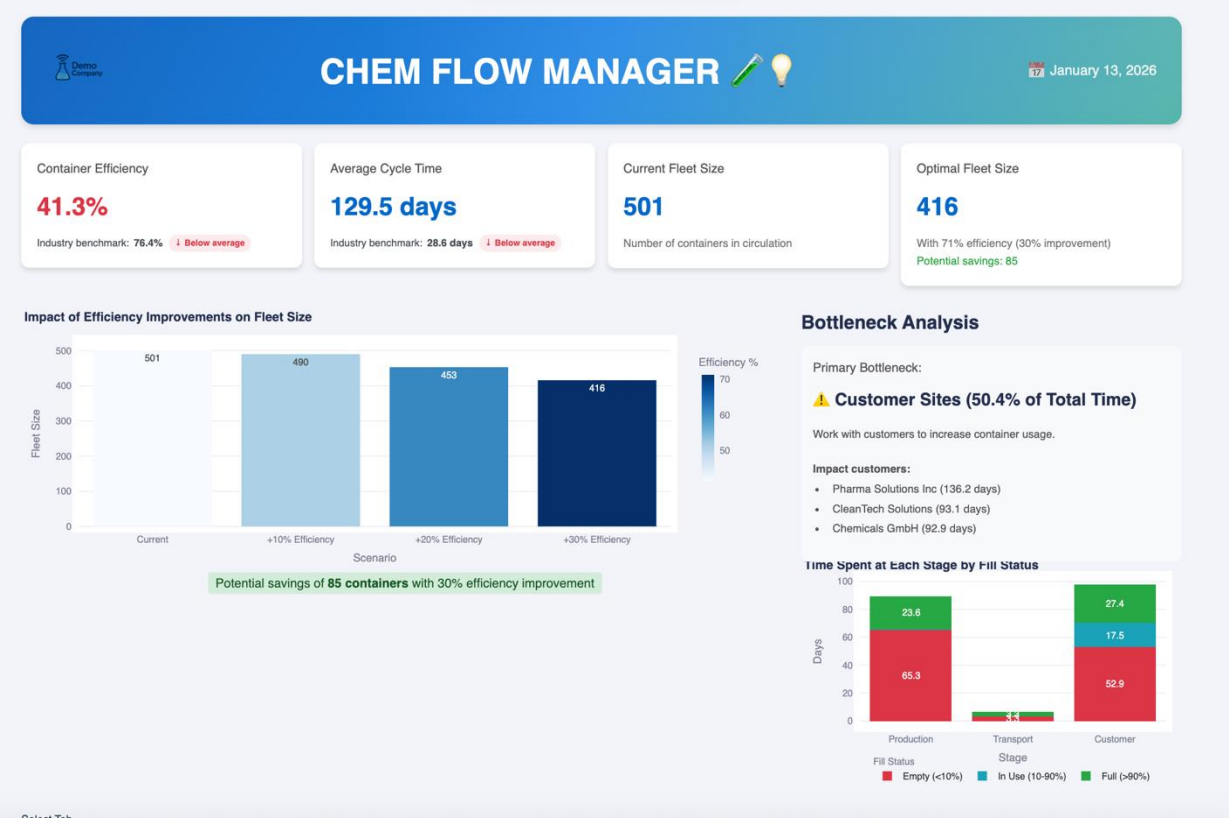
TANK	PRODUCT	PLACE	MAX VOL.	FILL. HIST.	FILL %	FILL VOL.	FILL STATUS	PUMP STATUS	CONS. LAST DAY	DAYS TILL 30%	DATE
TANK001	Hydraulic Fracturing Fluid	Eagle Ford Drilling Platform A	135		74	100	ABOVE 30%	NORMAL	3	20	202(10:
TANK043	Hydraulic Fracturing Fluid	West Texas Crude Site 2	135		79	106	ABOVE 30%	NORMAL	8	8	202(10:
LOWCONS085	Drilling Lubricant DL-50	Woodford Shale Platform TX	135		97	131	ABOVE 30%	NORMAL	1	14	202(10:
TANK042	Drilling Lubricant DL-50	Delaware Basin Rig West-1	135		29	39	BELOW 30%	NORMAL	7	0	N
TANK041	Corrosion Inhibitor CI-200	Austin Chalk Platform Alpha	135		12	16	BELOW 20%	NORMAL	3	0	N
TANK040	Drilling Mud Additive	Haynesville Rig Houston-1	135		82	111	ABOVE 30%	NORMAL	5	14	202(10:
BROKEN086	Completion Fluid CF-100	Delaware Basin Rig West-1	135		48	65	ABOVE 30%	TO BE CHECKED	0	113	202(10:
TANK039	Hydraulic Fracturing Fluid	Permian Basin Site Beta	135		24	32	BELOW 30%	NORMAL	5	0	N

Oil & Gas

Messbare Mehrwerte & Effizienzgewinne

Fleet Size Manager

Messbare Mehrwerte & Effizienzgewinne



Examples

Der iterative Workflow

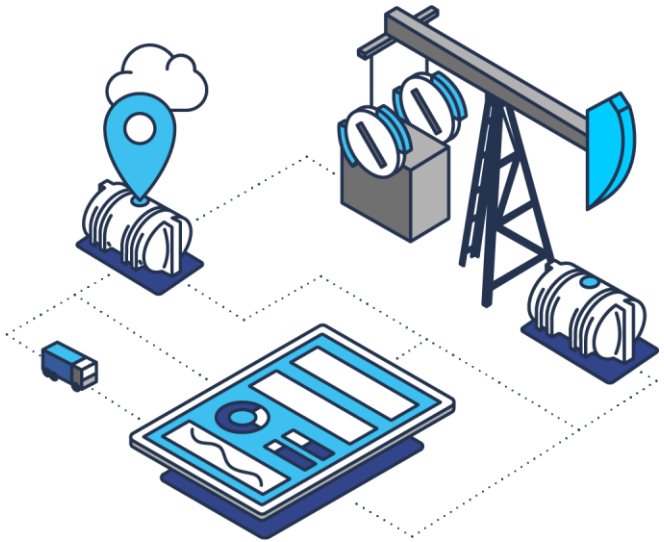
Traditioneller Weg: DB-Schema ändern → ETL anpassen → API erweitern → Frontend bauen → Deployment (Wochen).
Der Streamlit-Weg: Logik im Python-Script anpassen → Git Push → Live (Minuten).
Vorteil: Keine Datenbank-Migrationen oder Re-Seeding notwendig, da wir die Transformation „on-the-fly“ im Script machen.


pack:wise
05
Next Steps

Next Steps

Anzahl der Datenpunkte

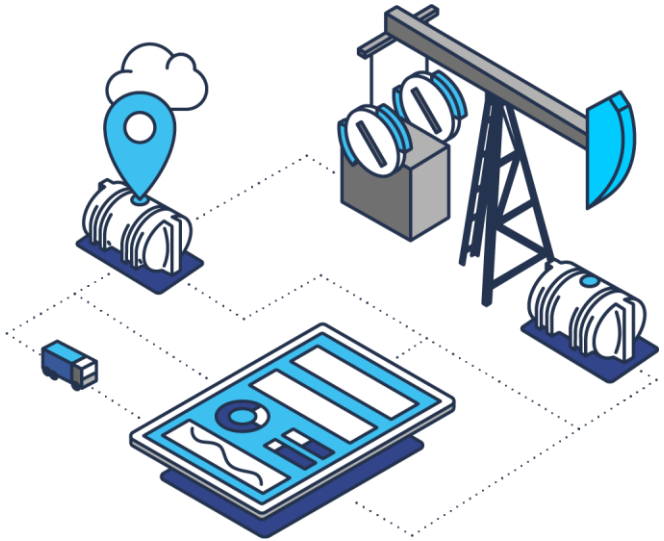
- **12.000+** aktive Smartcaps im Feld.
- **~3 Messungen** pro Tag
- **> 13 Mio.** Datenpunkte pro Jahr



Next Steps

Wenn das MVP an Grenzen stößt

- **Performance:** Millionen von Zeilen in Pandas zu prozessieren, treibt den RAM des Servers ans Limit
- **Latenz:** Der Datentransfer von S3 zum Client (Streamlit) wird bei wachsender Historie spürbar langsam.
- **Multi-Tenancy & Security:** Das Management von hunderten S3-Hashes und Customer-IDs wird im Code unübersichtlich.
- **Wartbarkeit:** „Spaghetti-Code“ in den Streamlit-Skripten durch zu viel Logik im Frontend-Bereich.

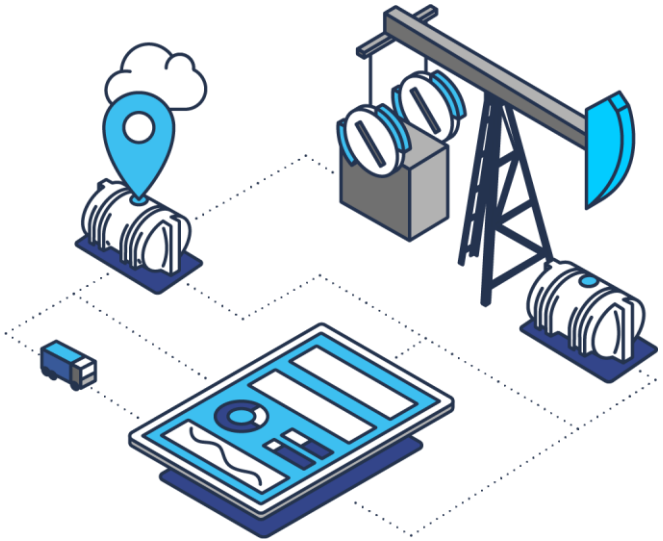


Next Steps

Wenn das MVP an Grenzen stößt

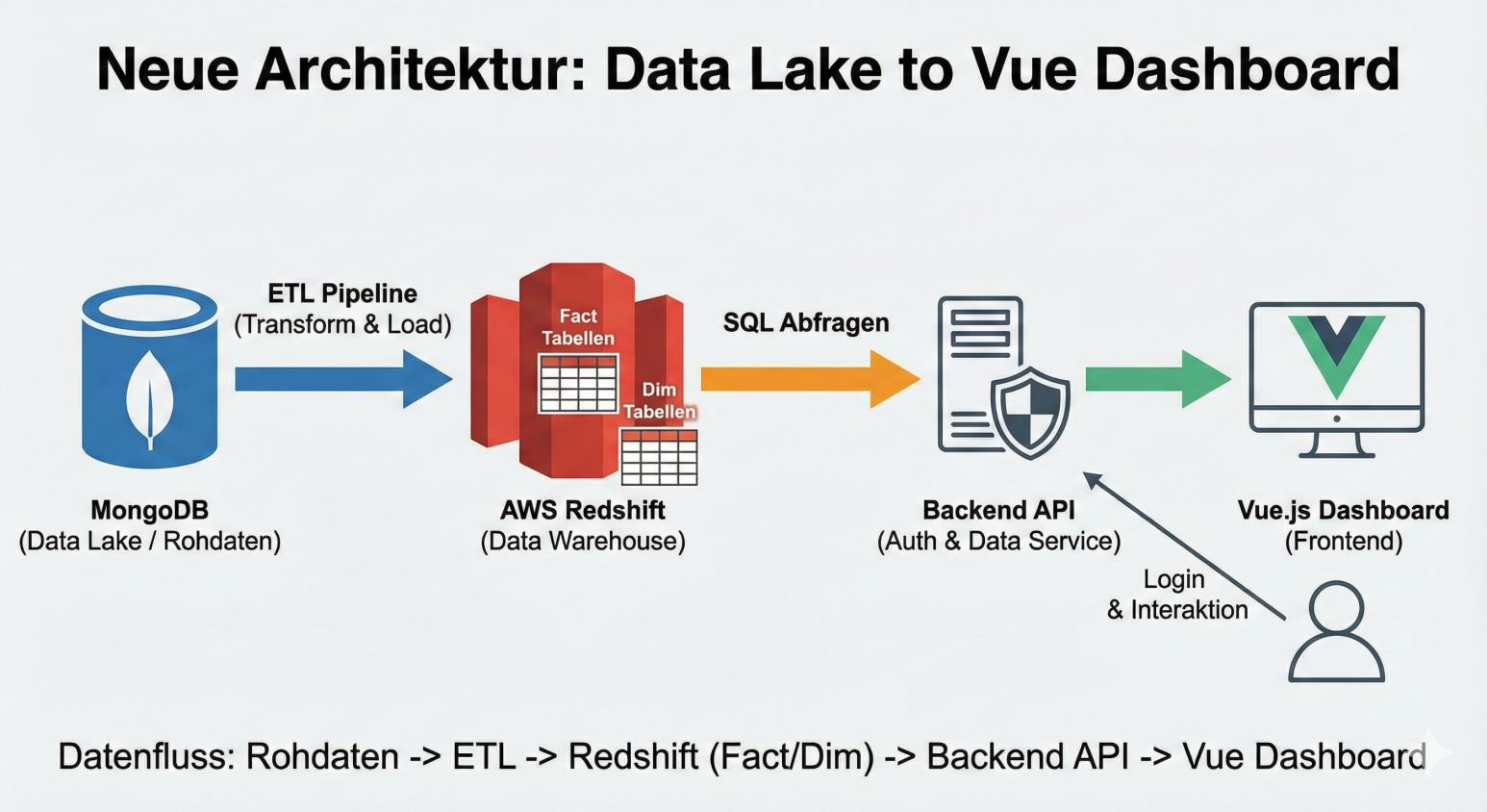
- **Storage:** Umstieg von S3-Dateien auf **AWS Redshift** (Cloud Data Warehouse).
- **Compute:** Logik wandert vom Python-Client in die Datenbank (SQL-Aggregationen).
- **ETL-Strecke:** Automatisierte Pipelines statt einfacher JSON-Dumps.

Streamlit bleibt für Data Science Prototypen, Spezial-Tools und neue MVP‘s



Next Steps

Die Architektur



Next Steps

Build to learn, not to scale

Zentrales Learning: „Scale what works – discard what doesn‘t.“ und Perfekt ist der Feind von Gut

Startup-Mentalität: Geschwindigkeit ist eine Ressource, Perfektion am Anfang ein Hindernis.

Empfehlung für Studenten: Nutzt Tools wie Streamlit, um Ideen sofort greifbar zu machen. Datenbanken sind das Fundament, aber die App ist das Fenster zum Nutzer.

pack:wise

Vielen Dank

Fragen?