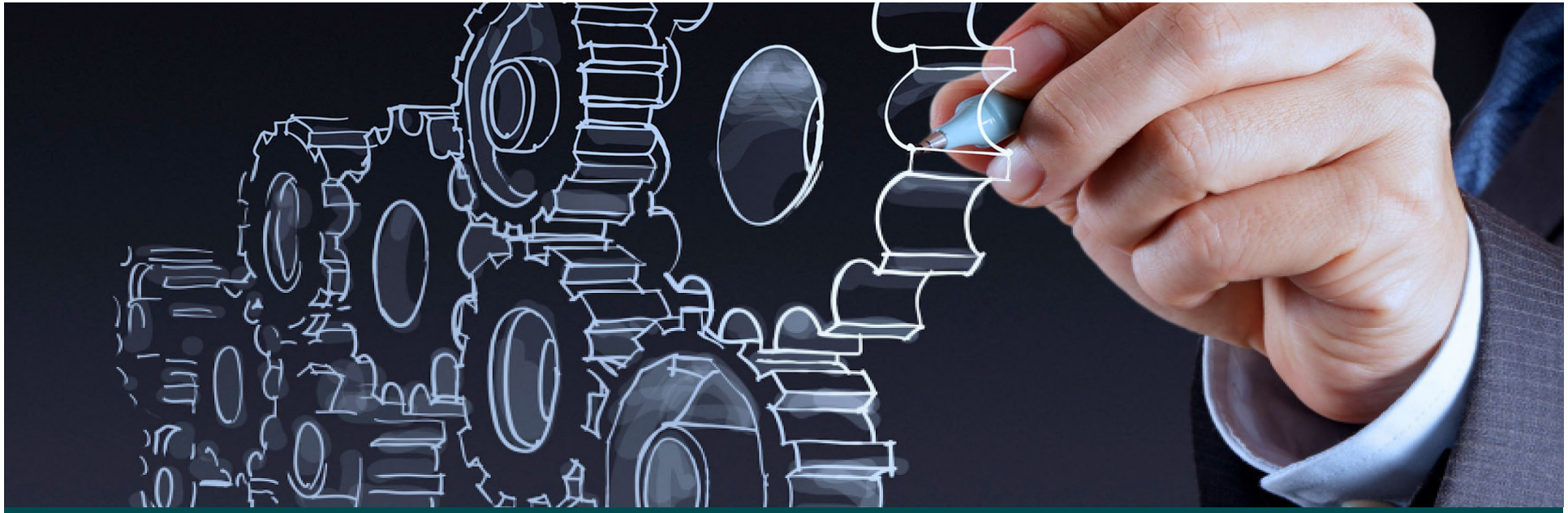
Alligator Company

Data Works!



Data Warehouse Automation mit Verstand

Alligator Company - Data Works!

Torsten Glunde / CEO

Data Warehouse Automation und Modernisierung

Data Warehousing und Business Intelligence seit 2002.

Kernkompetenzen:
Moderne Datenplattformen, Data Warehouse Automation & Virtualization, Analytics Engineering, DWH in der Cloud

Methoden:
ELT/ETL, SQL, CI/CD, Datavault, Information Modeling, ELM, BEAM

Gitlab: https://gitlab.com/alligatorcompany

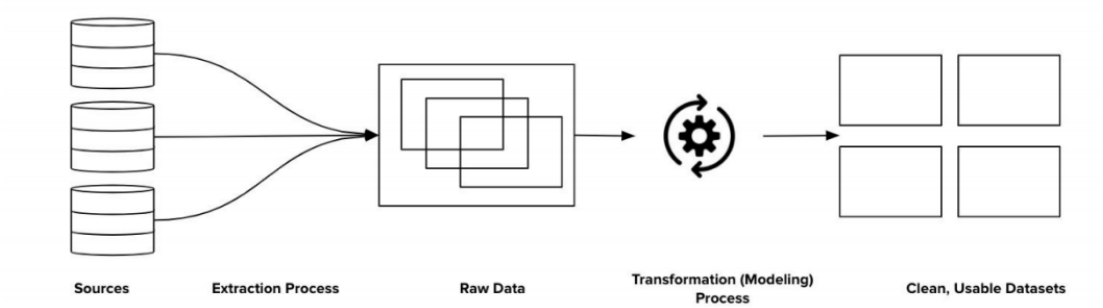
Focus Shifting - Automated Cloud DBMS include MPP

2010er Jahre

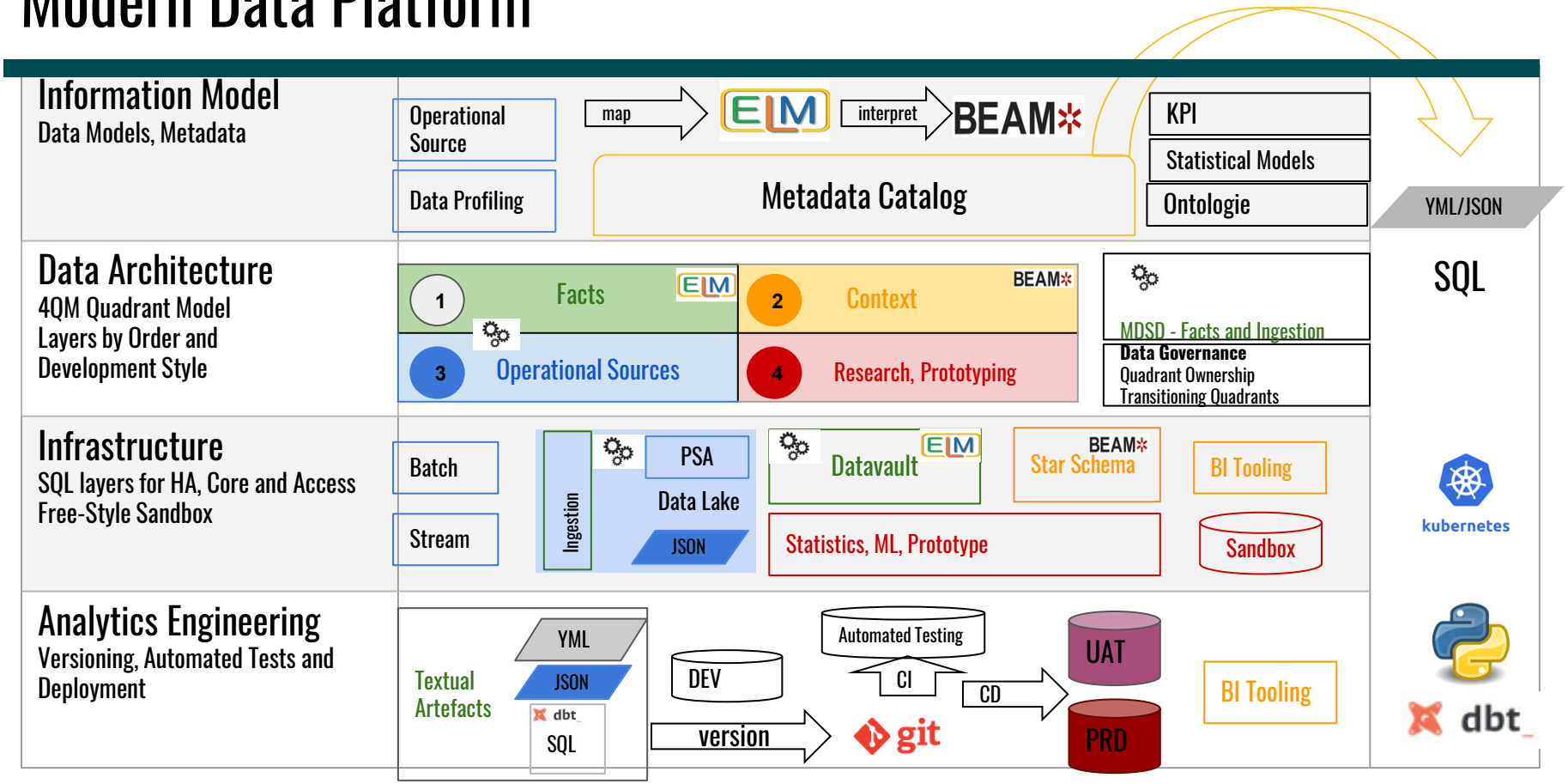
- Wie können wir die bestehende ETL Infrastruktur skalieren?
- Kostenreduktion durch Hadoop?
- Data Warehouse / Data Lake Diskussionen

Letzte 5 Jahre:

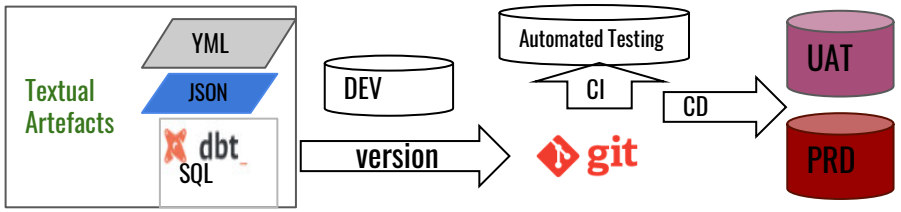
- Cloud Data Warehouse Hersteller sind MPP-fähig
- Cloud Storage Kosten sinken auch für DBMS
- Hadoop Projekte scheitern an “manuellem” MPP
- ELT vereinfacht Skalierung durch SQL Transformation



Modern Data Platform



Analytics Engineering (DBT)



- Alles ist ein SELECT statement (model)
  - Geschäftslogik in SQL - boilerplate code entfällt
  - Idempotent → wiederholbare Läufe
  - Transformation code is verfügbar über Versionierung
  - Collaboration über GIT
  - Materialisierungen sind konfigurierbar
- Materialisierungen
  - View, Table, Ephemeral, Incremental, Snapshot
  - … erweiterbar
- Lineage über {{ ref(..) }}
  - Schema Interpolation (Sandboxing)
  - Automatische Lineage und Dokumentation
- Testing framework
  - Test Annahmen über unsere Daten
  - Annahmen in yml definiert
  - Kompilierung in SQL
  - Tests
    - unique, nulls, values, relationship
  - Query tests
- Dokumentation
  - Beschreibungen (yml)
  - Modelabhängigkeiten
  - Model SQL
  - Quellen
  - Spalten und Tabellenstatistiken

No Middleware to buy and maintain
Versioning and accessibility through SQL in GIT
Transparenz und Kollaboration

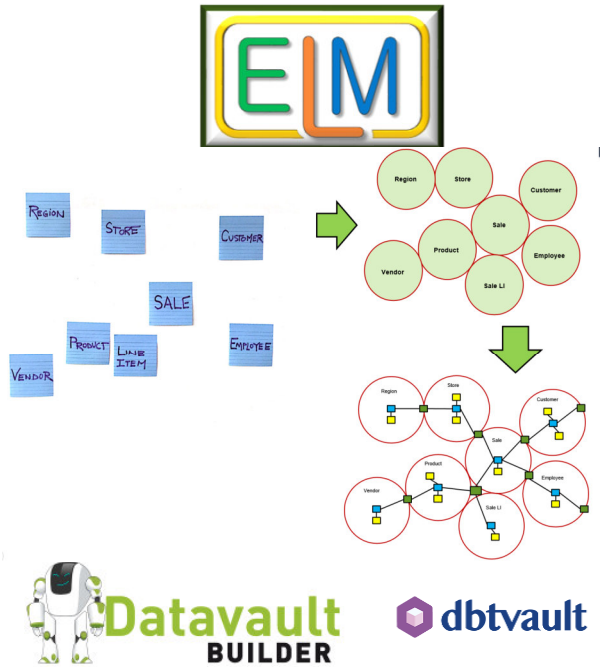
Continuous Integration

Automation
- Geschwindigkeit
- Reproduzierbarkeit
- Modellgetrieben
- Versionierung

Disposable Environments
- Entwickler können isoliert arbeiten
- Inkrementelle bilden und unabhängig versionieren

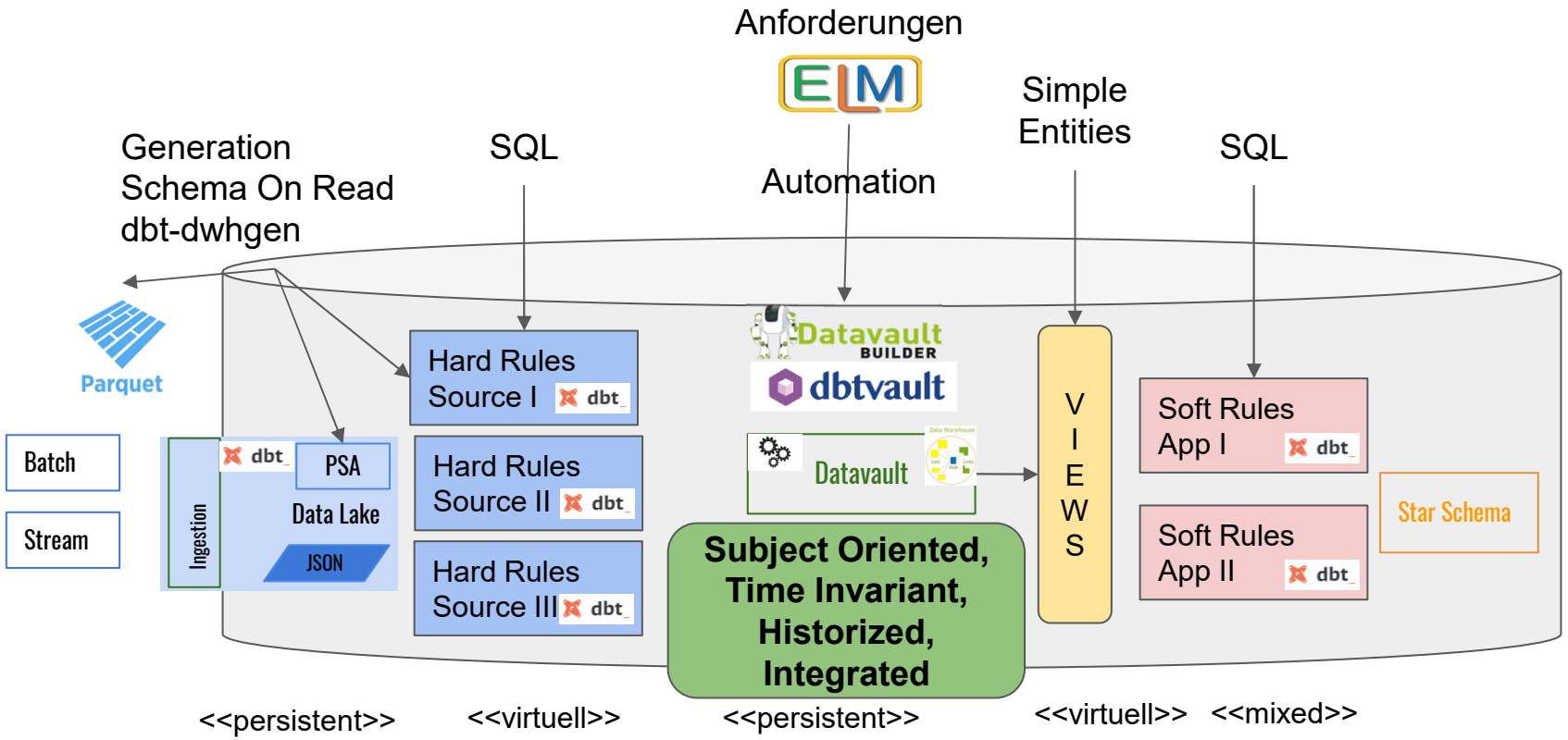
Simplify Operations
- k3d + k3s + Kustomize
- Okteto
- Argo Workflow, Events, CD, Rollouts

Convention over Configuration
- DBT und DVB standard Datenarchitektur
- Test Driven Development – DBT seeds

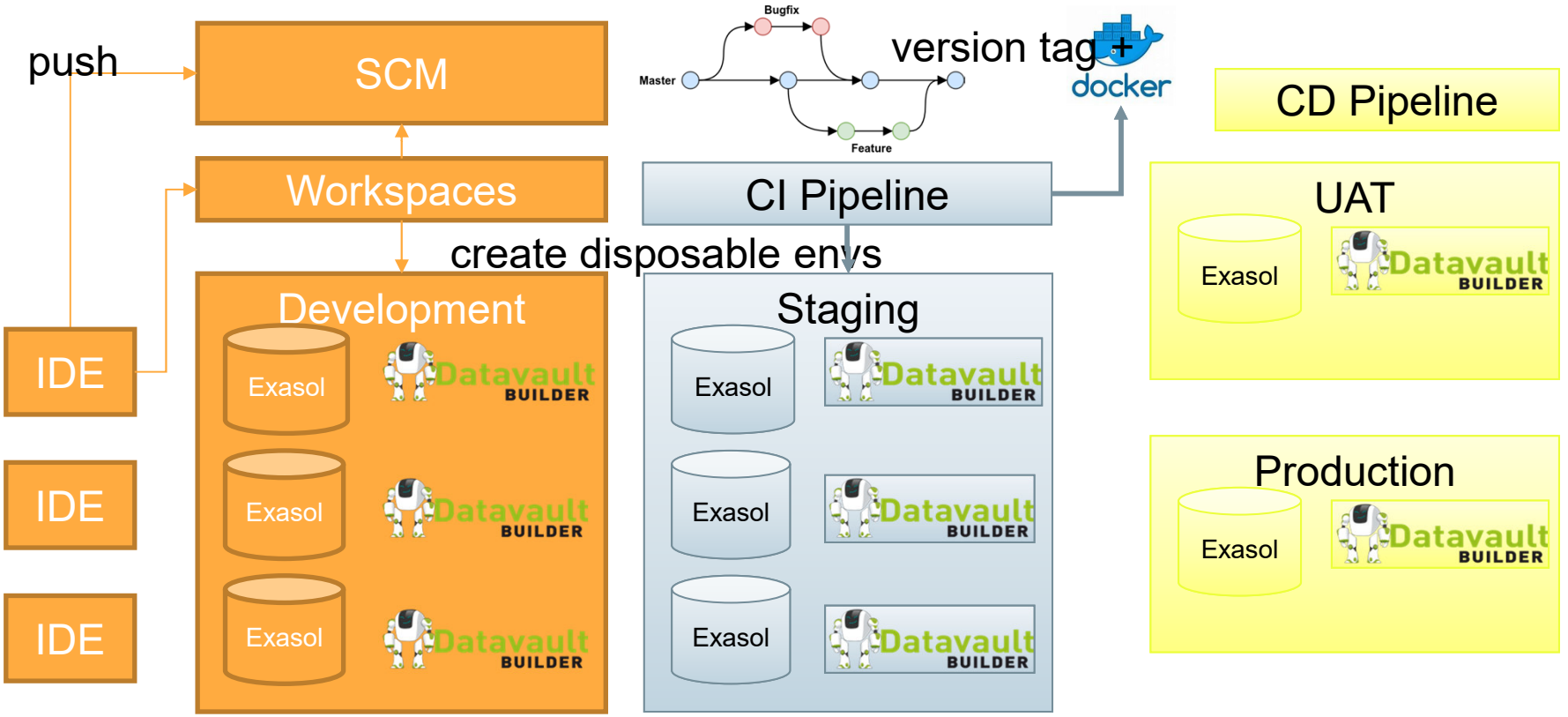


http://datawarehouseautomation.guide

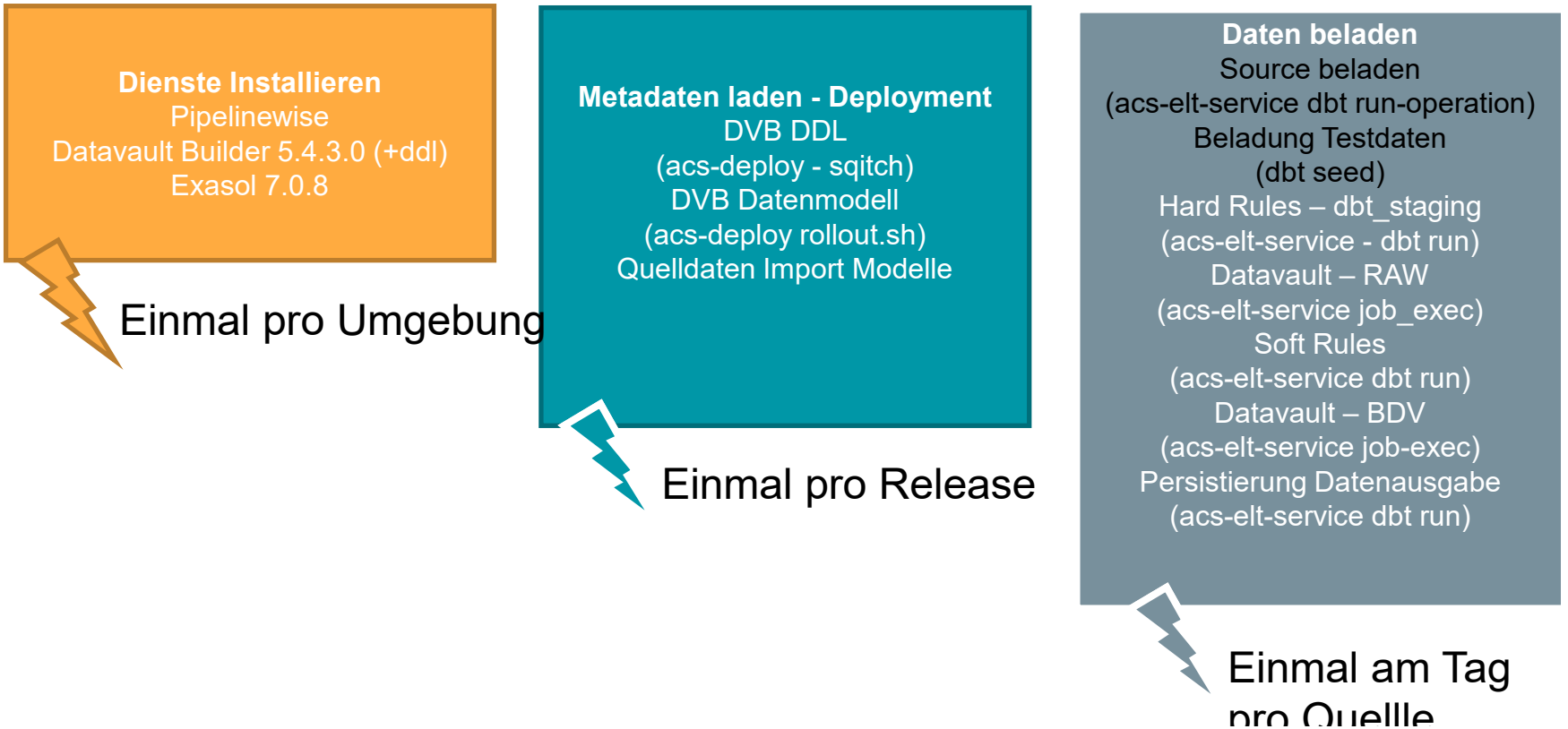
DBT and Datavault together - on Steroids: Divide and Conquer



Continuous Integration – disposable envs



CI-CD Pipeline



Werkzeugauswahl

DBMS	
Postgres https://www.postgresql.org/	Open Source, lizenzfrei, JSON, Column Möglich
Exasol https://www.exasol.com/en/	Column, In-Memory, MPP, deutscher Hersteller, Closed Source, On-Prem möglich
Snowflake https://www.snowflake.com/	Column, In-Memory, MPP + Storage/Compute, Cloud only,, Closed Source, Amerikanischer Hersteller

Data Integration	
DBT https://www.getdbt.com/	Open Core, Cloud native, SQL Modelle, ELT
DBTVault	Open Core, ELT metadata based, DBT native
Datavault Builder https://datavault-builder.com/	Datavault Automation, Web-based Data Modeling, Cloud native, CORE Integration, Scheduling, REST-API

Schedule	
Airflow https://airflow.apache.org/	Open Source, Python script based workflow definition, DAG
DBT Cloud https://cloud.getdbt.com/login/?next=/	Cloud-only, subscription, Add-On zu DBT Open Core
Argo Workflow	CNCF cloud native DAG implementation

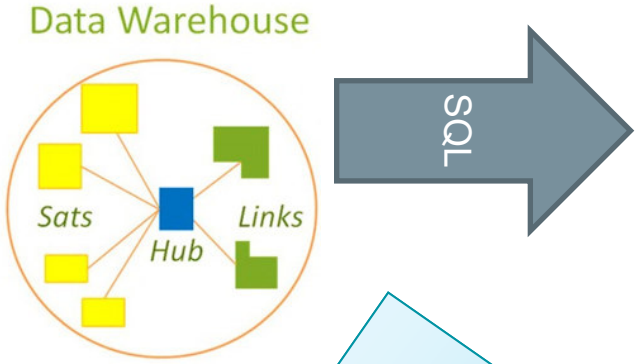
Frontend	
Apache Superset https://superset.incubator.apache.org/	Adhoc Analyse und Reporting, Web-based, Cloud-native, SQL-Semantic Layer
Looker https://looker.com/	Closed Source, Cloud-native, LookML semantic model, SQL based
Tableau https://tableau.com/	Dashboarding and

Empfehlung:
Text-based artefacts only (SQL, Python, JSON, R, Spark)
DBMS in Abhängigkeit von der Größe, Column, In-Memory, Cloud
Enterprise Integration (Authentication, Authorization) wichtige Treiber für Schedule and Frontend

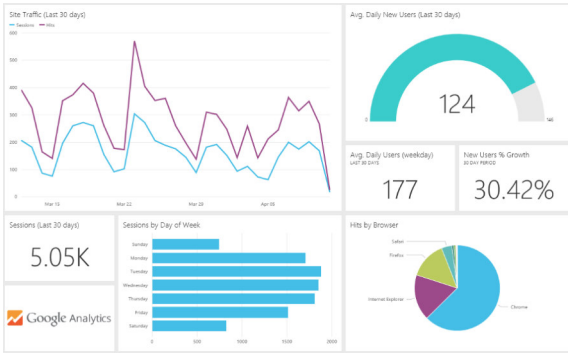
4QM – DWH Automation

Automation

BI Tools, ETL/ELT, Frontend

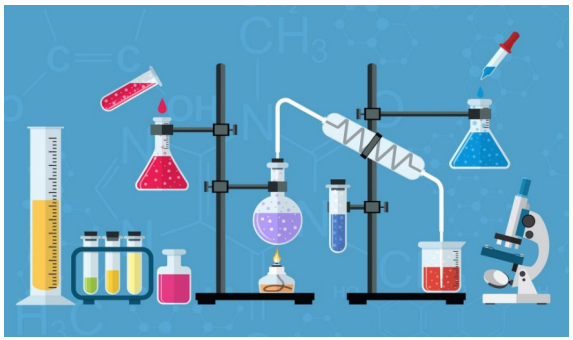


SQL



SQL

SQL



Learning from Twelve-Factor-App - https://12factor.net/

1. Codebase, one codebase
2. Dependencies, explicitly declared and isolated
3. Config, use environment variables
4. Backing Services, attached resources
5. Build, release, run, separation
6. Processes
7. Port binding, Export services with ports
8. Concurrency, scale out shared nothing
9. Disposability, fast startup and graceful shutdown
10. Dev/prod parity
11. Logs, event streams
12. Admin processes, on-off processes

- Docker images
  - Secure Containment, Port binding, Stdout Logging, Environment Variables
- Kubernetes
  - YAML declarations
  - Concurrency, Disposability
  - Jobs for management tasks, one-off processes
  - Services Port binding
  - API abstraction helps env similarity
- Follow Standards
  - no exec into container - processes only via service ports or Kubernetes API
- Declarative over Imperative
  - SQL, YAML, JSON - Simple text artefacts simple to version and collaborate

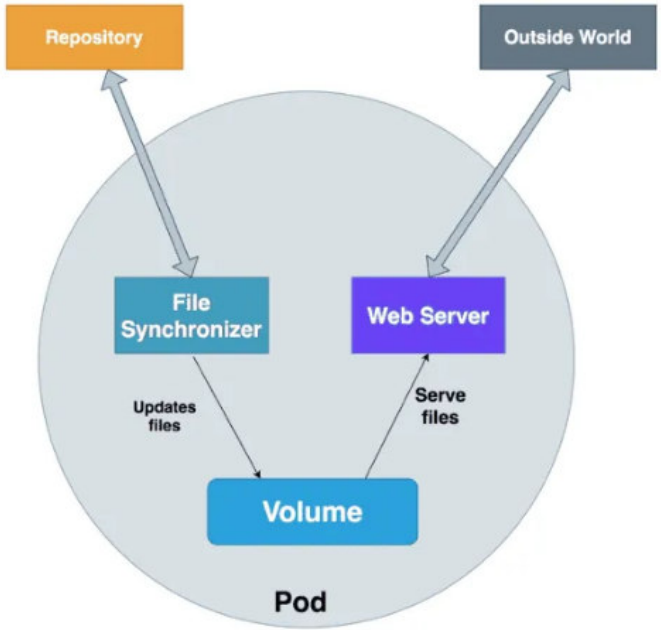
Vor- und Nachteile

- DBTVault (kein Werkzeugbruch, sehr gute Unterstützung Snowflake) vs. DVB (visuelle Darstellung, weniger technisch, stärker mit SQLServer (DBT nur community-modul), gute Unterstützung Snowflake), DVB monatliche Subscription, DBT komplett frei, Support DBT zukaufen
- DBTVaut: yaml-Deskriptoren
- DVB: REST-API – Json payload

- Dataspot (Governance+Fachbereichsintegration) vs. Konzeptionelles Modell (Datenmodellierung ERWin/Innovator/PowerDesigner)
- Input für Werkzeuge aus Modellierung “Automation der Automation”

Deployment

- Quellsystem Strukturen
  - Kafka: YAML, Manifest für CRD → kubectl
  - Import Tabellen Exasol: Sqitch.org → acs-deploy docker container
- Hard Rules
  - DBT views: dbt run, Strukturänderungen werden ersetzt
- Persistent Staging
  - DBT incremental model, generische Struktur, Full-Refresh bei Strukturänderungen
- Datavault Builder JSON-Metadaten
  - Python script, docker container → acs-deploy docker container
- Soft Rules
  - DBT views + table, strukturen werden ersetzt → acs-deploy docker container
- Versionierung DWH-Projekt docker image → Side-Car Container im Pod für Deploy-Image



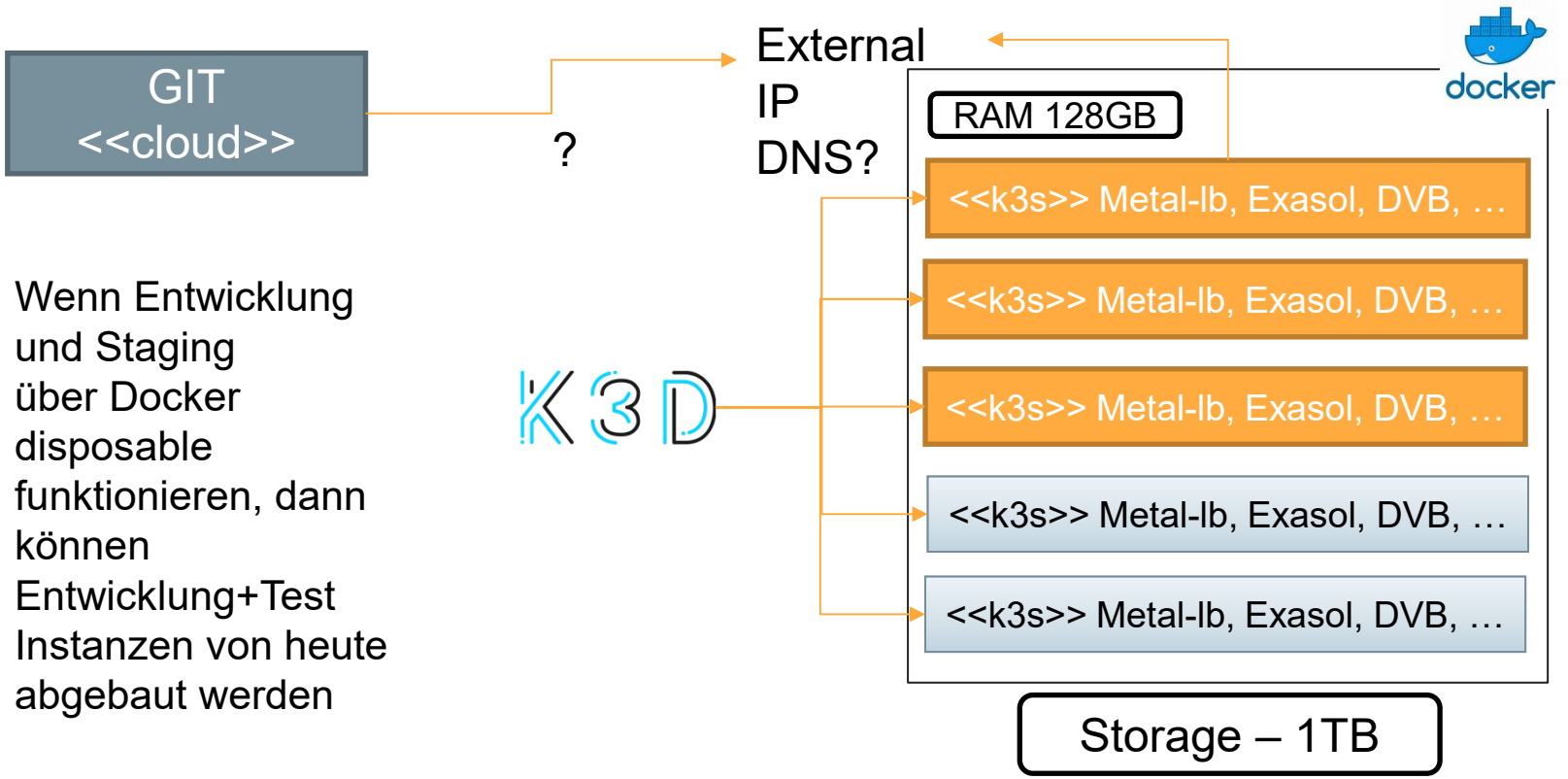
Sicherheitsfragen, CI

- Secrets Verschlüsselung
  - Secrets können versionsverwaltet werden
  - Cluster spezifisch

- Docker Images Security Scan
- Datentests per Seeds
  - Soft Rules: DBT Modelle sollen per SEEd getestet werden
    - STAGING → DVB → DBT
    - SEEDs (CSV) aus Kafka Datenströmen
  - Über Argo oder Airflow leicht auf Kubernetes umzusetzen - kann von jedem CI ausgeführt werden

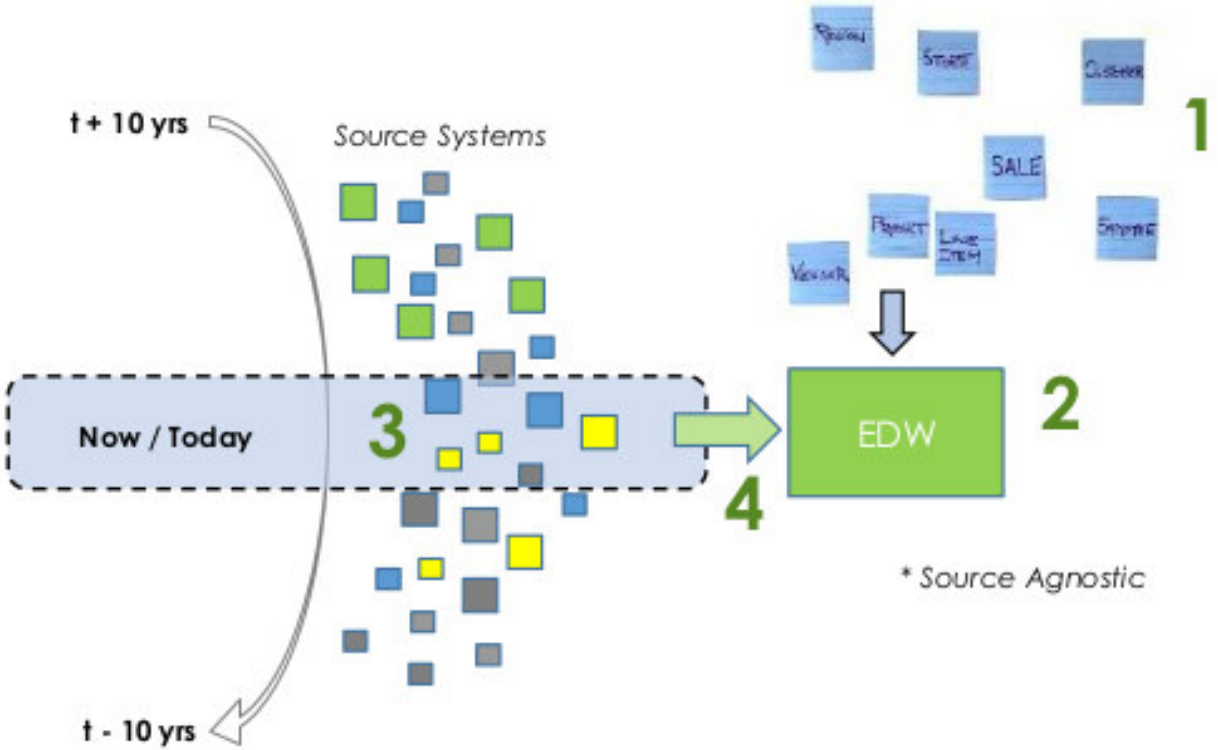
- Backup/Restore
  - Welche Layer sind persistent? Kafka und Exasol oder nur Exasol (wo liegt die PSA)
  - Metadaten müssen versioniert werden - keine zusätzlichen Datenbanken

Umgebungen Dockerhost

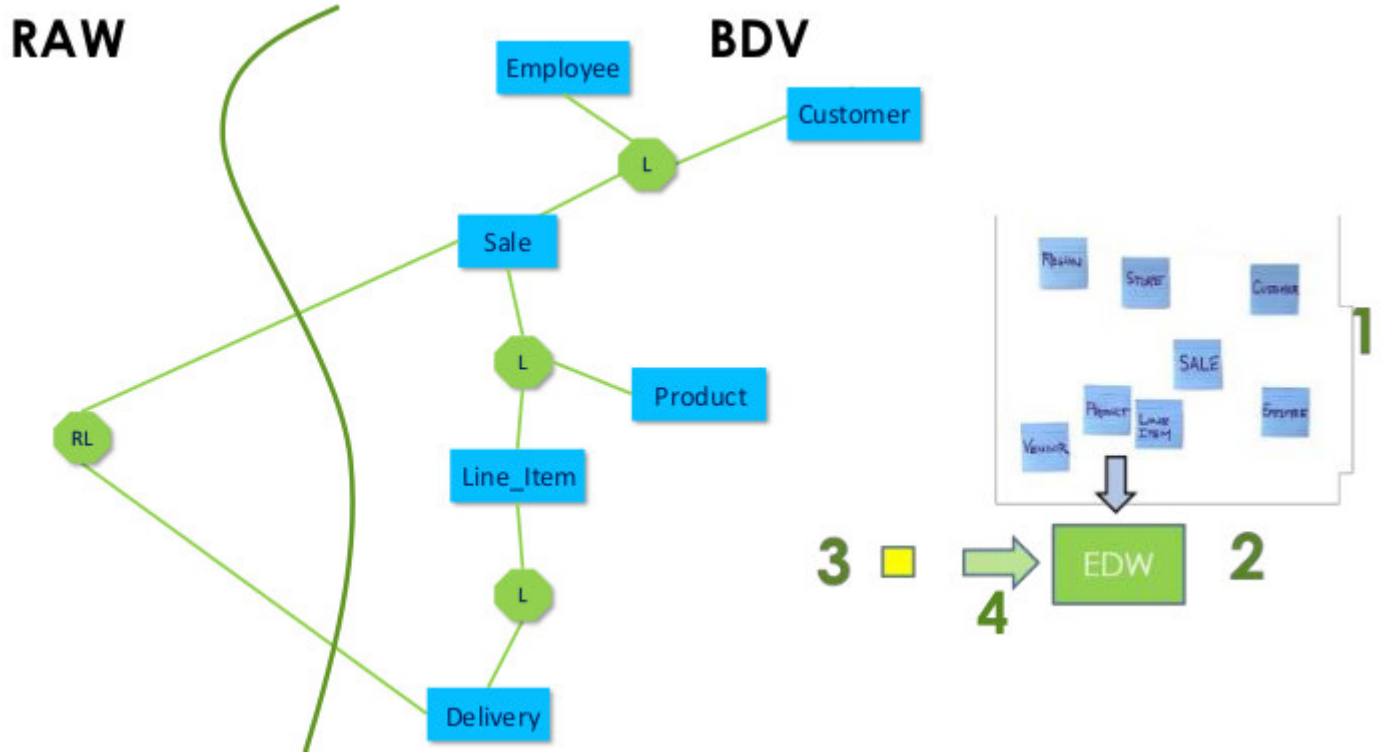


Wenn Entwicklung und Staging über Docker disposable funktionieren, dann können Entwicklung+Test Instanzen von heute abgebaut werden

EDW cannot be based on Multiple, Changing Sources



Business Driven Modeling - RAW elements are exceptions



3NF - Ensemble - New Thinking - True To The Key