# UNICEF

## UNICEF – Kinder in Not: Überblick

- UNICEF ist die Kinderhilfsorganisation der Vereinten Nationen und in 192 Ländern aktiv
- Schwerpunkt: Schutz, Versorgung und Förderung von Kindern in Krisen- und Notsituationen – insbesondere bei Konflikten, Naturkatastrophen und Pandemien
- Für das Jahr 2025 erwartet UNICEF weltweit über 213 Millionen Kinder in humanitärem Bedarf
- Der globale Finanzierungsaufruf "Humanitarian Action for Children 2025" umfasst rund 9,9 Milliarden US-Dollar zur Versorgung von über 109 Millionen Kindern

## UNICEF einfach erklärt: Überblick

- UNICEF ist das Kinderhilfswerk der Vereinten Nationen, mit Hauptauftrag, die Kinderrechte für alle Kinder weltweit sicherzustellen – unabhängig von Herkunft, Religion oder Hautfarbe
- gegründet 1946, heute in über 190 Ländern aktiv
- Ziel ist es, jedem Kind eine Kindheit zu ermöglichen: Gesundheit, Bildung, Schutz und Entwicklungschancen
- UNICEF hilft in Akutfällen (Katastrophen, Konflikte) und unterstützt langfristig Entwicklung und Stabilität
- Programme kombinieren praktische Hilfe (Brunnen, Impfungen, Schulen) mit politischer Arbeit und Beratung der Regierungen
- UNICEF entwickelt keine Insellösungen, sondern übertragbare Leuchtturmprojekte mit Breitenwirkung
- Die Arbeit basiert auf langfristiger Wirkung, Transparenz und Zusammenarbeit mit lokalen Partnern
- UNICEF unterstützt die Agenda 2030 / SDGs umfassend, da Kinderrechte essenziell für nachhaltige Entwicklung sind
- UNICEF Deutschland informiert, sammelt Spenden und sensibilisiert für Kinderrechte – durch Öffentlichkeits- und Bildungsarbeit mit Ehrenamtlichen

## Aktuelle Einsatzländer und Programme (Stand: 2025)

- Afghanistan: Versorgung von rund 12 Millionen Kindern mit Gesundheitsleistungen, Ernährung, sauberem Wasser und psychosozialer Betreuung
- Sudan: Versorgung inmitten eines Bürgerkriegs; Fokus auf Wasser, Hygiene, Schutz vor Ausbeutung, akuter Mangelernährung
- Palästina / Gaza: Behandlung mangelernährter Kinder; Verteilung von RUTF (therapeutische Spezialnahrung); humanitäre Versorgung akut gefährdet
- Syrien und Nachbarländer: Mobile Gesundheits- und Bildungseinheiten, psychosoziale Hilfe und Schutzprogramme für Geflüchtete
- Demokratische Republik Kongo: Einsatz gegen Cholera, Kinderarbeit und sexualisierte Gewalt in bewaffneten Konflikten
- Libanon: Hilfe bei Energie- und Wirtschaftskrise, Unterstützung von Kindern in Armut und geflüchteten Familien

## Künftige strategische Schwerpunkte von UNICEF (ab 2025)

- Ausbau von Frühwarnsystemen und datenbasierter Risikoplanung in Krisenregionen

- Stärkere Verzahnung von Not- und Entwicklungsprogrammen (Resilienzförderung)
- Verbesserung digitaler Lernangebote für Kinder in abgelegenen oder gefährdeten Regionen
- Aufbau lokaler Kapazitäten (Health, WASH, Education), um Abhängigkeiten von Hilfslieferungen zu reduzieren
- Innovation bei der Nutzung von Daten, Satellitenbildern und KI zur besseren Zielgruppenansprache und Bedarfsplanung

## Wichtige Themen für ein Bewerbungsgespräch mit UNICEF

- Klare Kenntnis der UNICEF-Mission: Schutz der Kinderrechte weltweit – auch in Notlagen
- Verständnis aktueller Krisenregionen und der dortigen kindbezogenen Bedarfe
- Interesse an humanitärer Hilfe, Resilienzaufbau und Entwicklungspolitik
- Datenbasiertes Arbeiten: Einsatz von Monitoring- und Evaluierungstools, z.B. zur Wirksamkeitskontrolle von Programmen
- Technisches oder analytisches Wissen (z.B. in Data Engineering, Plattform-Monitoring, API-Anbindungen) mit konkretem Bezug zur Programmunterstützung
- Kulturelle Sensibilität, Zusammenarbeit mit lokalen Partnern, Verständnis für komplexe Einsatzkontexte

## Aktuelle Projekte im Data Engineering (BI & Controlling) bei UNICEF

- Entwicklung und Optimierung von ETL-Prozessen zur Datenintegration aus verschiedensten Quellen (z.B. Partnerorganisationen, Gesundheitsdaten, Geographic Data) für BI- und Controllingzwecke. Dies beinhaltet Datenbereinigung, Harmonisierung, Validierung und Skalierbarkeit in Python und Azure- oder Databricks-Umgebungen.
- Aufbau und Pflege eines Data Warehouse und eines Indicator Data Warehouse zur Speicherung von Monitoring- und Krisendaten für Dashboard-Anwendungen. Integration von räumlichen Daten über GeoSight, GeoRepo und ArcGIS-Plattformen.
- Umsetzung von Power BI-Dashboards für Echtzeitvisualisierung von Projektfortschritten, Hilfsleistungen und KPIs zur Unterstützung von Controlling und Steuerung in Krisenregionen.

## Supply Chain Intelligence & Dashboarding

- Data Warehouse Optimierung durch Integration von Supply-Chain- und Demand-Planungsdaten
- Entwicklung vom PowerBI Dashboards zur Steuerung und Kontrolle, u.a. für Forecast, Procurement, WHO-Health und BHA
- Ziel ist die Bereitstellung konsistenter Supply-Master-Daten zur besseren BI-Nutzung und zum Monitoring kritischer Versorgungssituationen

## Ernährung & Überwachung

- Aufbau eines Nutrition Information System (NIS) mit zentralem Repository für Surveillance-, Monitoring- und Supply-Daten.
- Integration externer Systeme wie DHIS2/HMIS in das NIS.
- Automatisierung von Datenanalysen zur Unterstützung datenbasierter Entscheidungsprozesse.
- Entwicklung interaktiver Dashboards in Power BI zur Echtzeitüberwachung, inkl. Alarmierung, Lagerbeständen, Lieferstatus und Fortschrittsanzeige.

## Mögliche zukünftige Projekte im Data Engineering (BI & Controlling)

- Processing-Pipelines für den Integration von Private-Sector-Real-Time-Daten über die Magic Box Plattform (z.B. Mobilitätsdaten, Telekom-, Klimadaten) zur Unterstützung von Entscheidungsprozessen in Krisen.
- Automatisierte Geodaten-Pipelines mit Tools wie Geowrangler zur Analyse von Infrastruktur, Internetzugang oder Kinderarmut auf subnationaler Ebene; Ausgabe in BI-Reports zur Planung und Steuerung.
- Integration und Aufbereitung von U-Report oder RapidSMS-Daten (Crowd-sourced Feedback aus Krisengebieten) zur Echtzeit-Analyse von Bedarfen und Schutzrisiken; Nutzung für Monitoring-Dashboards
- Entwicklung einer Natural Language Processing (NLP) Pipeline zur Verarbeitung von Social Media oder Nachrichten (z.B. mittels Datensätzen wie HumVI oder CrisisBench), um Ereignisse wie Gewaltlagen oder medizinische Notfälle automatisch zu klassifizieren und BI-gerecht aufzubereiten
- Aufbau einer Pipeline zur Identifikation und Kartierung informeller Siedlungen aus Satellitenbildern zur Steuerung von Hilfsmaßnahmen gegen Kinderarmut und Schutzprogramme

## Relevante Aufgabenbereiche für BI & Controlling im UNICEF Kontext

- Konzeption, Entwicklung und Wartung robuster ETL-Prozesse für Datenintegration, Datenqualität und Automatisierung
- Aufbau, Administration und Weiterentwicklung von Data Warehouses und BI-Plattformen zur zentralen Datenanalyse
- Erstellung und Pflege von Dashboards und Reports für Controlling, Monitoring und Entscheidungsträger
- Dokumentation, Standardisierung und Schulung von Data-Engineering-Prozessen und Datenquellen
- Enge Zusammenarbeit mit Monitoring- und Analytik-Teams sowie regionübergreifende Koordination mit Data Nodes und Partnern

## Beispiele für datenorientierte Entscheidungen im Kontext von UNICEF – Kinder in Not

- Analyse von Konflikt- und Fluchtdaten, um Krisengebiete zu identifizieren, in denen Kinder akut gefährdet sind, etwa durch Vertreibung, Gewalt oder Ausbeutung
- Einsatz von Gesundheits- und Versorgungsdaten zur Entscheidung, wo mobile Kliniken oder temporäre Impfzentren eingerichtet werden sollen, um Kinder in schwer zugänglichen Gebieten zu erreichen
- Aggregation von Bildungsunterbrechungsdaten nach Naturkatastrophen (z.B. Erdbeben, Überschwemmungen), um über temporäre Schulräume oder digitale Lernangebote für betroffene Kinder zu entscheiden
- Auswertung von Schutzbedarfsanalysen, um Programme zum Schutz vor Kindersoldaten-Rekrutierung, sexueller Gewalt oder Zwangsarbeit in besonders gefährdeten Regionen zu priorisieren
- Analyse von Meldungen aus lokalen Partnernetzwerken, um Kinderrechtsverletzungen (z.B. Misshandlung, Kinderehen) zu erkennen und schnell Hilfsmaßnahmen einzuleiten
- Verknüpfung von Erhebungsdaten mit Monitoring-Systemen zur Bewertung der Wirksamkeit früherer Notfallmaßnahmen, etwa zur Entscheidung über Verlängerung oder Anpassung von Hilfsprogrammen für Kinder

# Technisches

## Data Warehouse

### Grundlegendes

- Zentrales, strukturiertes System zur langfristigen Speicherung und Analyse großer Datenmengen aus verschiedenen Quellen
- Optimiert für komplexe Abfragen und Business-Intelligence-Anwendungen
- Fokus liegt auf strukturierte, sauberen und konsistenten Daten, die für Business Intelligence (BI), Reporting und Data Analytics genutzt werden
- Daten kommen meist aus operativen Systemen (ERP, CRM), externen Quellen oder Sensoren
- Data Warehouses sind optimiert für schnelle, komplexe Abfragen (OLAP – Online Analytical Processing)
- Beispiele: MS SQL Server, Amazon Redshift, Google BigQuery

### Eigenschaften

- Themenorientierung: Daten werden nach Themen (Kunden, Produkte, Verkäufe) organisiert, nicht nach einzelnen Transaktionen
- Integration: Daten aus verschiedenen Quellen werden vereinheitlicht (gleiche Formate, Zeitstempel, Kodierungen)
- Zeitbezug: Historische Daten werden gespeichert, um Trends und Entwicklungen analysieren zu können
- Nicht-flüchtig: Daten werden nicht ständig verändert, sondern ergänzend gespeichert

### Architektur

- ETL-Prozess (Extract, Transform, Load): Daten werden extrahiert, aufbereitet und ins DW geladen
- Datenbanken/Storage: Relationale oder spaltenbasierte Datenbanken, die für große Datenmengen und schnelle Abfragen optimiert sind
- Metadatenmanagement: Beschreibung der Datenquellen, Transformationen und Datenqualität
- Front-End Tools: BI-Tools, Dashboards, Reporting-Systeme, die auf das DW zugreifen

### Betrieb

- Monitoring und Performanceoptimierung: Abfragen, Speicherplatz und Ladeprozesse werden überwacht und optimiert (Indexierung, Partitionierung)
- Datensicherheit und Datenschutz: Zugriffsrechte, Verschlüsselung und Compliance (z.B. DSGVO) sind zentral
- Datenqualität: Daten werden validiert, Duplikate bereinigt und Fehlerquellen minimiert
- Wartung und Updates: Regelmäßige Wartung der Datenbanksoftware, Anpassungen an neue Datenquellen und Business-Anforderungen
- Backup und Recovery: Strategien zur Sicherung und Wiederherstellung der Daten sind essenziell

## Datenstrukturen

- Methoden oder Formate, wie Daten intern dargestellt und organisiert werden, damit man effizient darauf zugreifen, sie verarbeiten und speichern kann
- Bilden die Grundlage für alle Datenpipelines, Datenbanken und Speicherlösungen
- Beispiele:
  - Tabellen und relationale Strukturen
  - Arrays und Listen (JSON, Parquet)
  - Key-Value-Stores (NoSQL Datenbanken)

### Wichtigkeit

- Effizienz: Daten müssen schnell geladen, transformiert und abgefragt werden können
- Speicherplatz: Optimale Datenstrukturen sparen Speicherplatz (z.B. Komprimierung bei Spaltenformaten)
- Kompatibilität: Unterschiedliche Systeme nutzen unterschiedliche Formate – Data Engineers sorgen für reibungslose Datenübergabe
- Skalierbarkeit: Große Datenmengen erfordern effiziente Strukturen, damit Verarbeitung auch in der Cloud funktioniert
- Beim Datenimport werden Rohdaten in passende Strukturen überführt
- Im ETL-Prozess werden Daten transformiert und strukturiert (z.B. JSON in Tabellen umwandeln)
- Für Datenanalyse und Machine Learning werden oft spezielle Formate/Strukturen gewählt, die schnell verarbeitet werden können

## Datenqualität

- Vollständigkeit → Sind alle erforderlichen Datenfelder vorhanden und ausgefüllt?
- Korrektheit → Entsprechen die Daten den realen, erwarteten Werten (z.B. stimmen Postleitzahlen mit Städten überein)
- Konsistenz → Sind die Daten in verschiedenen Systemen oder Tabellen widerspruchsfrei (z.B. gleicher Kundennamen in allen Datensätzen)
- Aktualität → Sind die Daten aktuell bzw. zeitgerecht verarbeitet (z.B. keine veralteten Transaktionen im Reporting)
- Eindeutigkeit → Gibt es doppelte Datensätze, wo es keine geben sollte (z.B. doppelte Kunden-IDs)
- Validität → Entsprechen die Daten den erwarteten Formaten oder Regeln (z.B. E-Mail-Adressen im gültigen Format, Zahlen in numerischen Feldern)

## Entwicklung von Datenladestrecken

### Wichtiges

- Datenquellenvielfalt verstehen:
  - UNICEF arbeitet mit sehr unterschiedlichen Quellsystemen:
    - Administrative Daten (z.B. Personal, Finanzen)
    - Feldberichte (z.B. Gesundheitsdaten, Bildungsstatistiken)
    - Externe Quellen (Regierungsdaten, Partnerorganisationen)
  - Unterschiedliche Formate (CSV, JSON, Excel, APIs, Datenbanken) und Strukturen
- Datenintegration & Harmonisierung:
  - Unterschiedliche Datenformate und -strukturen müssen vereinheitlicht werden (z.B. Datumsformate, Maßeinheiten)
  - Terminologien und Klassifikationen (z.B. Länder-, Altersgruppen-Codes) müssen abgestimmt werden
- Datenqualität sicherstellen:
  - Fehlerhafte, unvollständige oder doppelte Daten sind häufig
  - Validierung und Bereinigung sind zentral (z.B. fehlende Werte, Inkonsistenzen)
- Automatisierung & Wiederholbarkeit:
  - Datenladestrecken müssen regelmäßig laufen (z.B. täglich, wöchentlich)
  - Automatisierte Jobs, die bei Fehlern Alarm schlagen und selbstständig neu starten
- Skalierbarkeit & Performance:
  - Datenmengen können schnell wachsen, vor allem bei Feldberichten oder Sensoren
  - Effiziente Ladeprozesse, z.B. inkrementelles Laden (nur neue/geänderte Daten)
- Sicherheit & Datenschutz:
  - Sensible Daten (z.B. persönliche Gesundheitsdaten) müssen geschützt werden
  - Zugriffskontrollen, Verschlüsselung, Compliance mit Datenschutzbestimmungen (z.B. DSGVO)
- Transparenz & Dokumentation:
  - Klare Dokumentation der ETL-Prozesse, Datenherkunft (Lineage) und Transformationsregeln
  - Besonders wichtig bei internationalen Organisationen für Audits und Reporting

### Probleme

- Heterogenität der Quellsysteme:
  - Problem: Unterschiedliche Formate, fehlende Standardisierung erschweren die Integration
  - Lösung:
    - Entwicklung modularer Pipelines, die Daten aus verschiedenen Formaten aufnehmen und in ein einheitliches Zwischenformat (z.B. Parquet, JSON) transformieren.
    - Nutzung von Data Catalogs, um Quellen und Formate zu dokumentieren
- Datenqualität:
  - Problem: Unvollständige oder fehlerhafte Daten führen zu falschen Analysen
  - Lösung:
    - Automatisierte Validierung auf Vollständigkeit, Konsistenz, Wertebereiche
    - Alerts bei Datenanomalien, damit Teams schnell reagieren können
- Komplexe Transformationen: Oft müssen Daten stark angepasst werden, was Fehlerquellen erhöht

- Fehlende Automatisierung: Manuelle Prozesse sind fehleranfällig und nicht skalierbar
- Performance-Engpässe:
  - Problem: Große Datenmengen können Ladeprozesse verlangsamen
  - Lösung:
    * Inkrementelles Laden von Daten, um nur Änderungen zu verarbeiten
    * Nutzung von spaltenorientierten Speicherformaten (Parquet, ORC) für schnellere Analysen
- Sicherheitsanforderungen:
  - Problem: Sensible Daten müssen geschützt sein, was zusätzliche Komplexität bringt
  - Lösung:
    * Datenklassifikation und Anonymisierung oder Pseudonymisierung sensibler Daten
    * Compliance-Checks und Auditing-Prozesse, um Datenschutzbestimmungen einzuhalten

### Kontext UNICEF

- UNICEF arbeitet oft in schwierig zugänglichen Regionen, wo Daten nicht standardisiert oder unvollständig sind
- Daten kommen aus verschiedenen Ländern mit unterschiedlichen IT-Systemen und Sprachen
- Rechtliche und ethische Anforderungen im Umgang mit sensiblen Daten (z.B. von Kindern) sind besonders streng
- Häufig ist eine enge Zusammenarbeit mit lokalen Partnern nötig, um Datenquellen zu verstehen und zu verbessern

## API Anbindung

- Schnittstelle, über die verschiedene Software-Systeme Daten austauschen und Funktionen aufrufen können.
- API-Anbindung bedeutet, dass Systeme über definierte Protokolle (z.B. REST, SOAP) miteinander kommunizieren, Daten senden oder empfangen
- UNICEF integriert Daten aus vielen internationalen Partnern, Regierungen und eigenen Systemen. APIs sind zentral für:
  - Feld-Datenerfassung: Mobile Apps in Krisengebieten senden Gesundheits- oder Bildungsdaten per API an zentrale Datenbanken
  - Partner-Daten: Automatisierter Austausch mit NGOs, WHO, UN-Systemen
  - Monitoring & Reporting: Echtzeit-Updates für Fortschrittsberichte und Entscheidungsfindung

## Plattform Monitoring

- Plattform-Monitoring bezeichnet die kontinuierliche Überwachung von IT-Systemen, Infrastruktur und Anwendungen, um deren Verfügbarkeit, Performance und Sicherheit sicherzustellen.
- Ziel: Probleme frühzeitig erkennen, Ausfälle vermeiden und schnelle Fehlerbehebung ermöglichen.

## Wichtige Bereiche beim Plattform-Monitoring

### Infrastruktur-Monitoring

- Überwachung von Hardware-Komponenten
- Wichtige Metriken: CPU-Auslastung, RAM-Nutzung, Festplattenplatz, Netzwerkauslastung, Latenz

### Anwendungs-Monitoring

- Überwachung der Verfügbarkeit und Leistung von Software-Anwendungen und Services
- Wichtige Kennzahlen: Antwortzeiten, Fehlerquoten, Transaktionsvolumen

### Log-Management

- Sammlung und Analyse von System- und Anwendungs-Logs
- Hilft, Ursachen von Fehlern zu finden und Sicherheitsvorfälle zu erkennen

### Sicherheits-Monitoring

- Überwachung von Zugriffen, verdächtigen Aktivitäten und Angriffen
- Einbindung von Intrusion Detection Systems (IDS) und Security Information and Event Management (SIEM)

### Benutzer- und Zugriffsmonitoring

- Überwachung von Nutzeraktivitäten und Berechtigungen, um Missbrauch zu verhindern

## Wichtige Funktionen und Methoden

- Alerting: Automatische Benachrichtigung bei Überschreitung von Schwellenwerten (z.B. CPU $> 90\%$)
- Dashboards: Übersichtliche Visualisierung der Systemzustände in Echtzeit
- Trendanalysen: Historische Daten zur Kapazitätsplanung und Problemprävention
- Self-Healing: Automatisierte Reaktionen auf bestimmte Probleme (z.B. Neustart eines Dienstes)
- SLA-Überwachung: Kontrolle, ob Service-Level-Agreements eingehalten werden

## Tools für Plattform-Monitoring (Beispiele)

- Prometheus & Grafana: Open-Source-Monitoring und Visualisierung
- Visualisierung von Metriken und Zeitreihendaten aus verschiedenen Datenquellen
- Erstellung interaktiver Dashboards zur Überwachung von Systemen und Anwendungen
- Alarmierung und Benachrichtigungen bei Grenzwertüberschreitungen

### Herausforderungen im Plattform-Monitoring

- Datenflut: Große Datenmengen müssen sinnvoll ausgewertet werden
- False Positives: Zu viele oder falsche Alarme können zu Alarmmüdigkeit führen
- Integration heterogener Systeme: Unterschiedliche Plattformen und Technologien müssen zusammen überwacht werden
- Skalierbarkeit: Monitoring-Lösung muss mitwachsen, z.B. bei Cloud-Plattformen
- Sicherheit: Monitoring-Daten müssen geschützt und nur autorisierten Personen zugänglich sein

### Plattform-Monitoring im Kontext von UNICEF

- UNICEF betreibt oft verteilte Systeme und globale Datenplattformen. Monitoring ist entscheidend, um Ausfälle in Krisengebieten schnell zu erkennen und Datenverluste zu vermeiden
- Monitoring unterstützt die Einhaltung von Datenschutz und Sicherheit, vor allem bei sensiblen Daten von Kindern
- Echtzeit-Überwachung ermöglicht schnelles Reagieren auf technische oder sicherheitsrelevante Probleme

## MLOps

### Grundlegendes zu MLOps

- MLOps ist die Praxis, ML-Modelle effizient, reproduzierbar und skalierbar zu entwickeln, zu deployen und im Betrieb zu halten
- Es kombiniert Methoden aus DevOps und Data Science, um den gesamten ML-Lifecycle zu automatisieren und zu überwachen
- Ziel: Schnelle und sichere Bereitstellung von ML-Modellen in produktiven Umgebungen sowie kontinuierliche Verbesserung und Wartung

### Prinzipien von MLOps

- Automatisierung: Automatisierung aller Schritte von Datenaufbereitung, Modelltraining, Testing bis hin zum Deployment
- Reproduzierbarkeit: Sicherstellen, dass Modelle und Ergebnisse jederzeit nachvollziehbar und reproduzierbar sind
- Kontinuierliche Integration und Deployment (CI/CD): Automatisierte Pipelines, die Code, Daten und Modelle integrieren und aktualisieren
- Skalierbarkeit: Modelle und Infrastruktur sollen einfach skalierbar sein, z.B. bei steigender Datenmenge oder Nutzerzahlen
- Monitoring und Feedback: Überwachung der Modelle im Betrieb, um Leistungseinbußen (z.B. durch Datenverschiebung) frühzeitig zu erkennen

### Konzepte in MLOps

- Versionierung:
  - Versionierung von Code, Daten und Modellen (z.B. mit Git, DVC, MLflow)
- Pipeline-Orchestrierung:
  - Automatisierte Abläufe für Datenvorverarbeitung, Training, Evaluation und Deployment (z.B. mit Apache Airflow, Kubeflow Pipelines)
- Testen:

  - Unit Tests, Integrationstests, Modelltests (z.B. Validierung der Genauigkeit, Fairness)
- Feature Store:
  - Zentralisierte Speicherung und Verwaltung von Features zur Wiederverwendung und Konsistenz
- Governance und Compliance:
  - Dokumentation, Auditing und Einhaltung von Datenschutz und ethischen Richtlinien

### Deployment im MLOps

- Batch Deployment: Modelle werden periodisch aktualisiert und für Stapelverarbeitung genutzt
- Online Deployment: Modelle laufen in Echtzeit als Service (z.B. REST API) und beantworten Anfragen live
- Canary Releases / Blue-Green Deployment: Neue Modelle werden schrittweise ausgerollt, um Risiken zu minimieren
- Containerisierung: Verwendung von Docker und Kubernetes für portables, skalierbares Deployment

### Betrieb und Monitoring

- Modell-Monitoring:
  - Überwachung von Modellmetriken wie Genauigkeit, Latenz, Antwortzeiten
- Daten-Monitoring:
  - Überwachung der Eingabedaten auf Drift, Anomalien oder veränderte Verteilungen
- Alerting:
  - Automatische Benachrichtigung bei Leistungsabfall oder Fehlern
- Automatisiertes Retraining:
  - Modelle werden bei Bedarf automatisch neu trainiert und deployed, z.B. bei Datenverschiebung
- Logging und Auditing:
  - Nachvollziehbarkeit aller Aktionen und Entscheidungen im ML-System

### CI/CD Pipelines

- CI: Automatisches Testen und Bauen bei jeder Codeänderung
- CDelivery: Automatisches Ausliefern in eine Staging-Umgebung
- CDeployment: Automatisches Ausliefern bis in die Produktion (ohne manuelle Eingriffe)

### Automatisierte Jobs mit MLFlow

- MLflow ist eine Open-Source-Plattform zur Verwaltung des Machine-Learning-Lifecycles mit Fokus auf Automatisierung und Reproduzierbarkeit.
- Es unterstützt vier Kernkomponenten:
  - MLflow Tracking: Protokollierung von Experimenten, Parametern, Metriken und Artefakten.
  - MLflow Projects: Strukturierung und Verpackung von ML-Code zur reproduzierbaren Ausführung.
  - MLflow Models: Verwaltung und Bereitstellung von ML-Modellen in verschiedenen Formaten und Umgebungen.

  - MLflow Registry: Zentralisierte Modell-Repository mit Versionierung, Genehmigung und Lifecycle-Management.
- Automatisierung wird durch standardisierte Pipelines erreicht, in denen Training, Evaluation und Deployment orchestriert werden.
- Integration mit CI/CD-Tools ermöglicht automatische Ausführung von Experimenten und Updates.
- MLflow erleichtert die Nachverfolgbarkeit und Reproduzierbarkeit, was manuelle Fehler reduziert und schnelle Iterationen erlaubt.
- Unterstützt verschiedene ML-Frameworks (z.B. TensorFlow, PyTorch, Scikit-learn) und Deployment-Optionen (z.B. REST APIs, Batch).