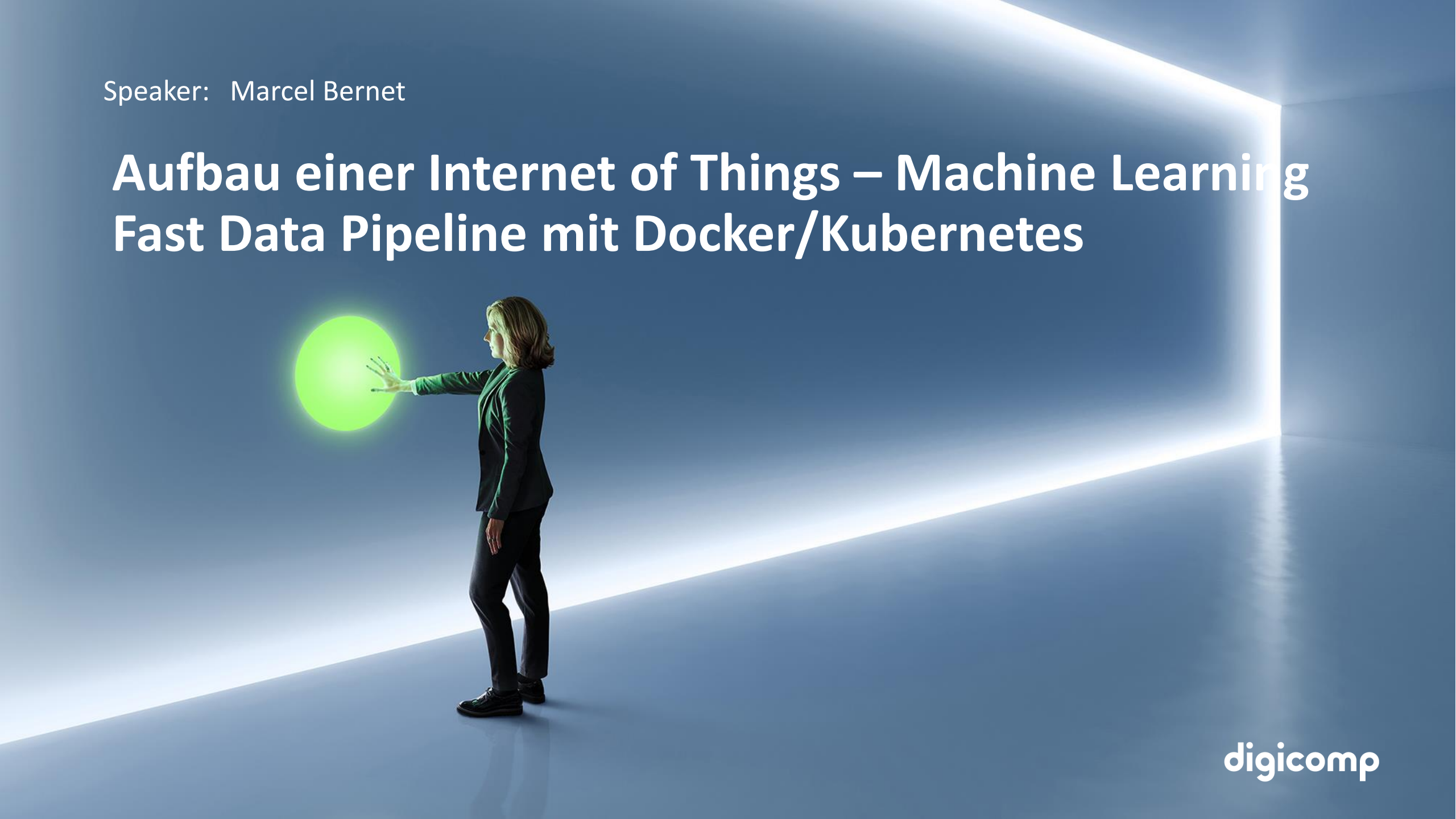
Speaker: Marcel Bernet

Aufbau einer Internet of Things – Machine Learning Fast Data Pipeline mit Docker/Kubernetes



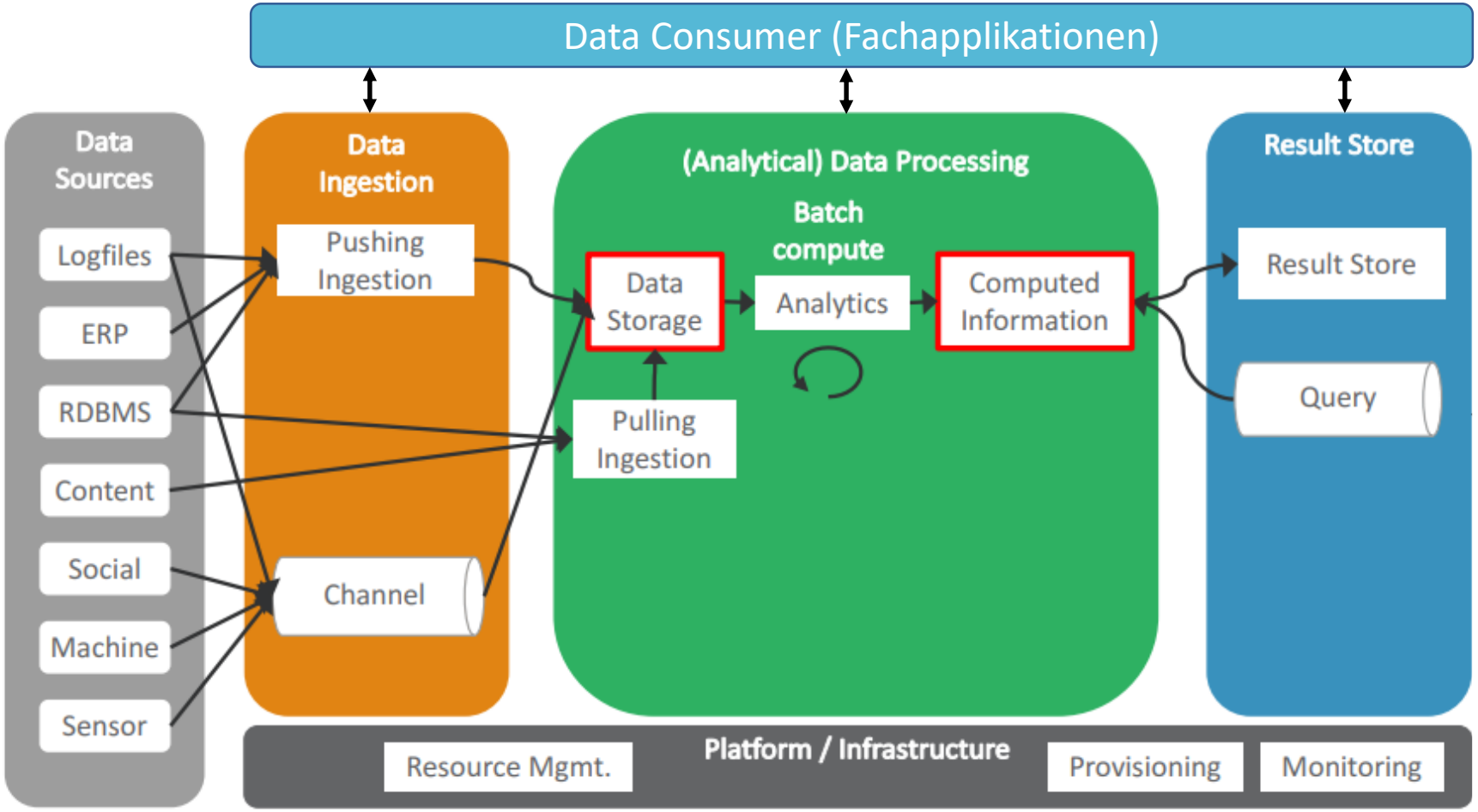
digicomp

Vorstellung Marcel Bernet

★ Marcel Bernet

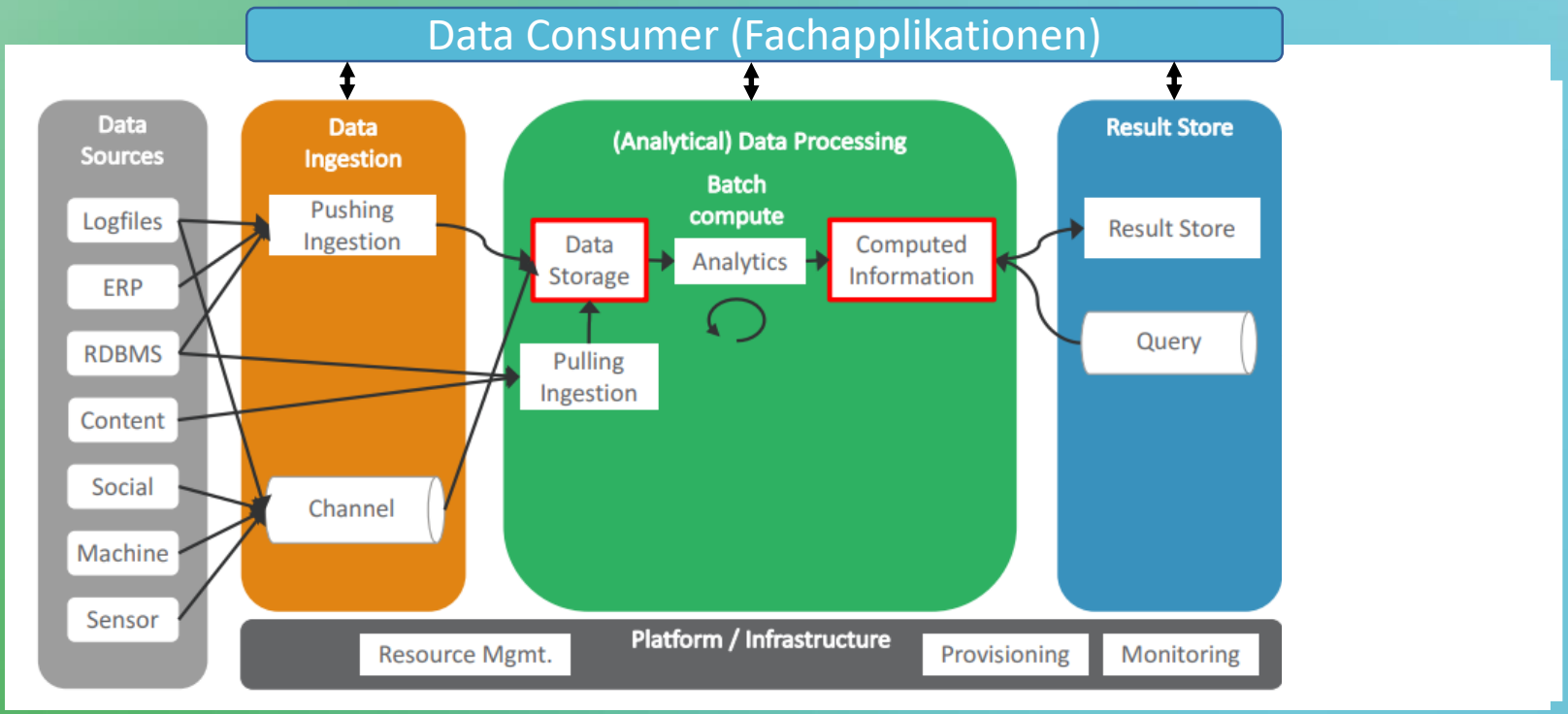
- IT-Consultant, IT-Architekt (Beratung, Schulung)
- Seit über 20 Jahren Selbständig.
- Themenkreise: Digitalisierung, Internet of Things, Big Data, Machine Learning, Microservices, DevOps, Containerisierung
- Ehemaliges Mitglied Expertenkommissionen swissICT, eCH
- Ehemaliger CH-open Präsident
- https://github.com/mc-b/

Big Picture: Fast Data Pipeline



Agenda

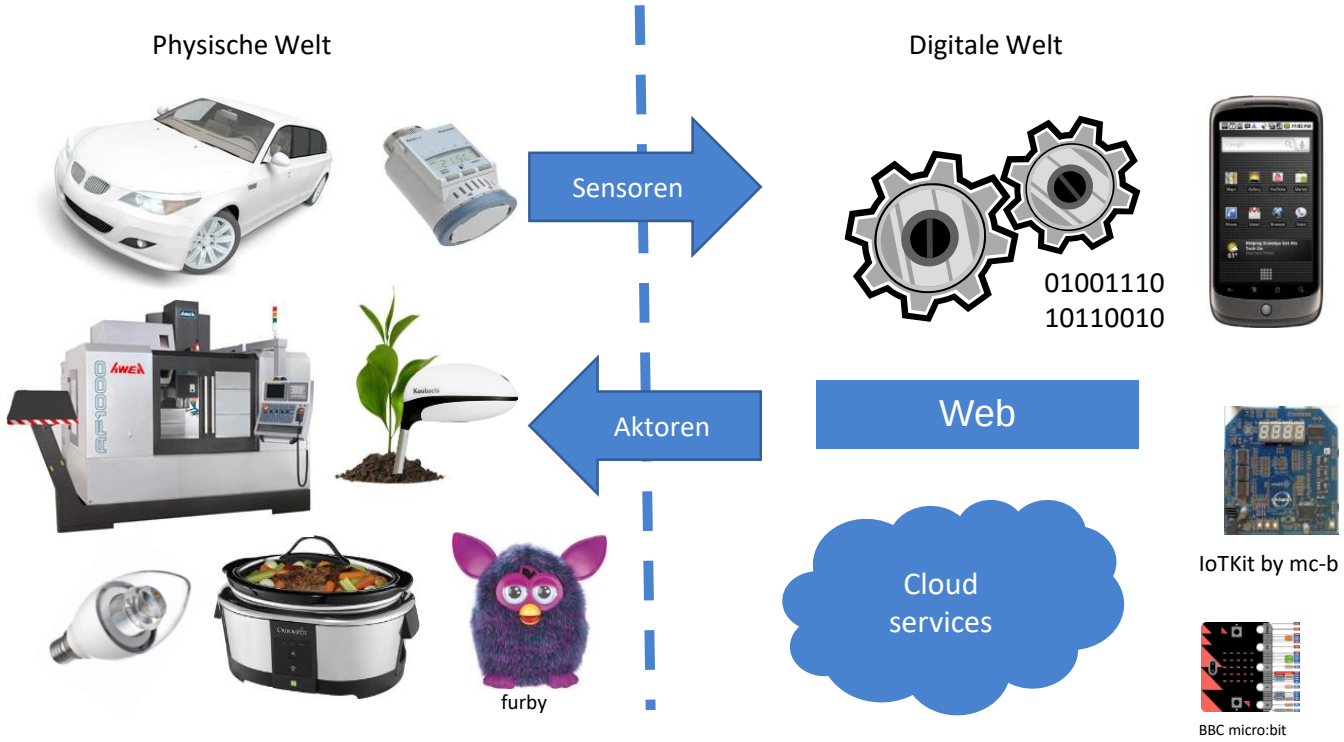
- ★ Internet of Things (Data Sources)
- ★ Machine Learning (Analytical Data Processing)
- ★ Docker/Kubernetes (Platform / Infrastructure)
- ★ Fast Data Pipeline
- ★ Erweiterung: IoT und Prozesse (Data Consumer)



Internet of Things (IoT)

Data Sources

Internet of Thing (IoT)



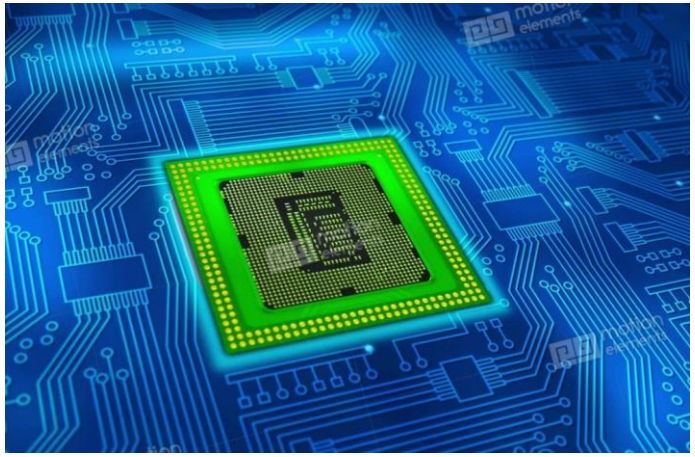
★ Das «Internet der Dinge» vereint die physische mit der digitalen Welt.

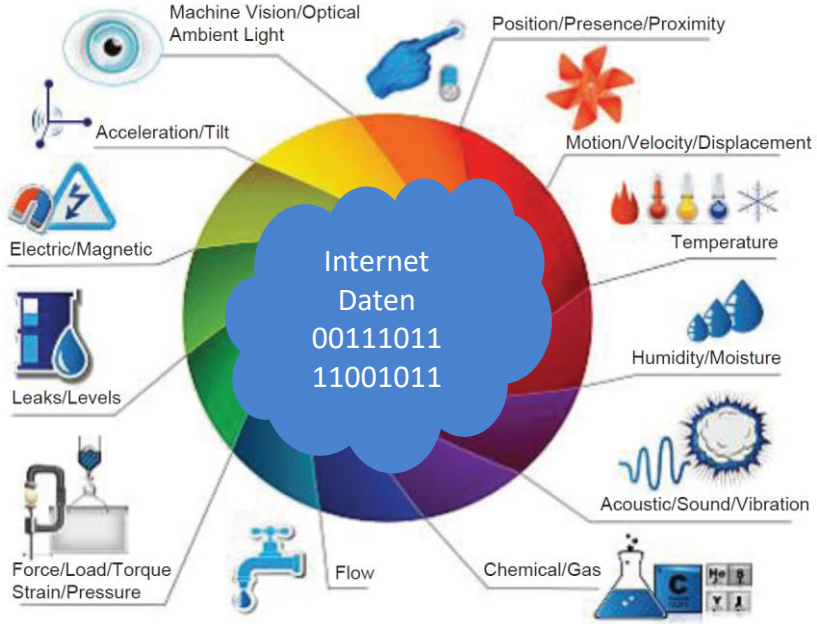
★ Sensoren nehmen die physische Welt war und Aktoren wirken auf sie.

IoT: Zusammengefasst

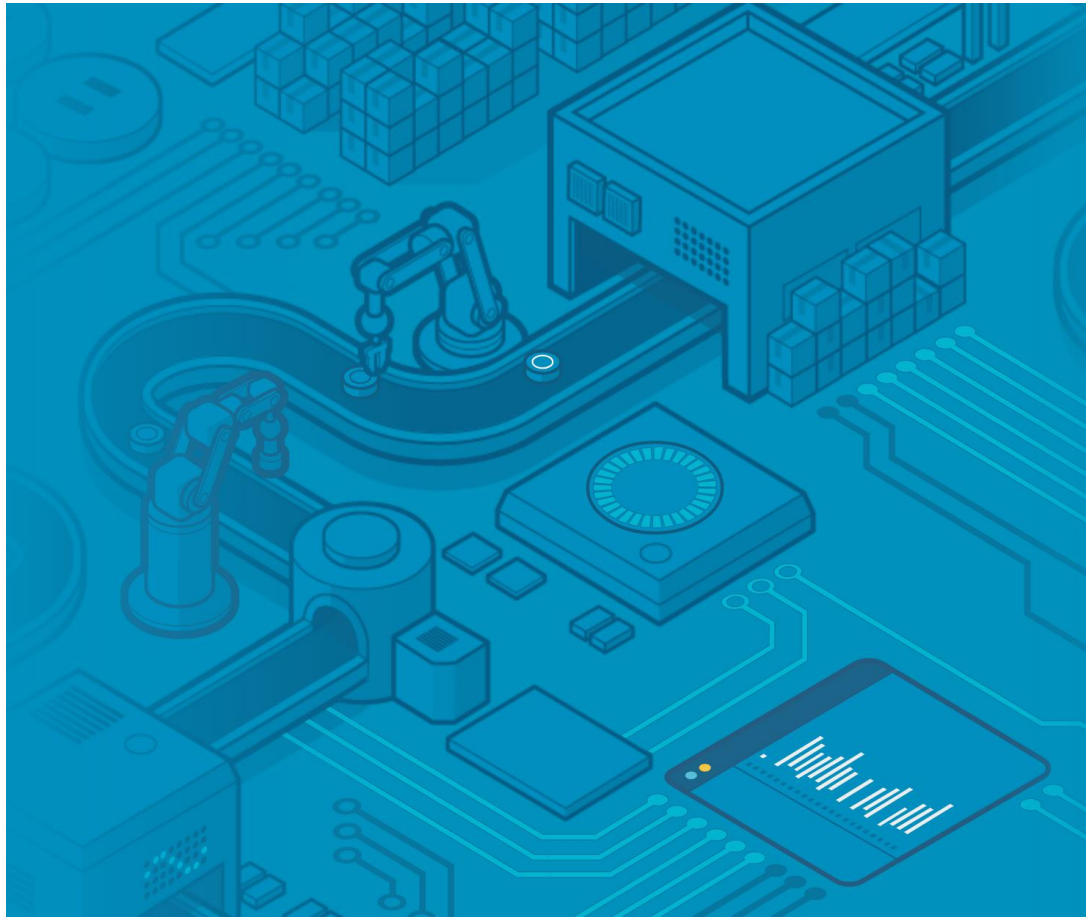
DEV {}</> Day by digicomp

- ★ Mikrocontroller MCU (ab CHF 2.52 Einzelpreis)
- ★ Geräte (Sensoren/Aktoren) sind mit dem Internet verbunden
- ★ Es entstehen **Daten, Daten, Daten**



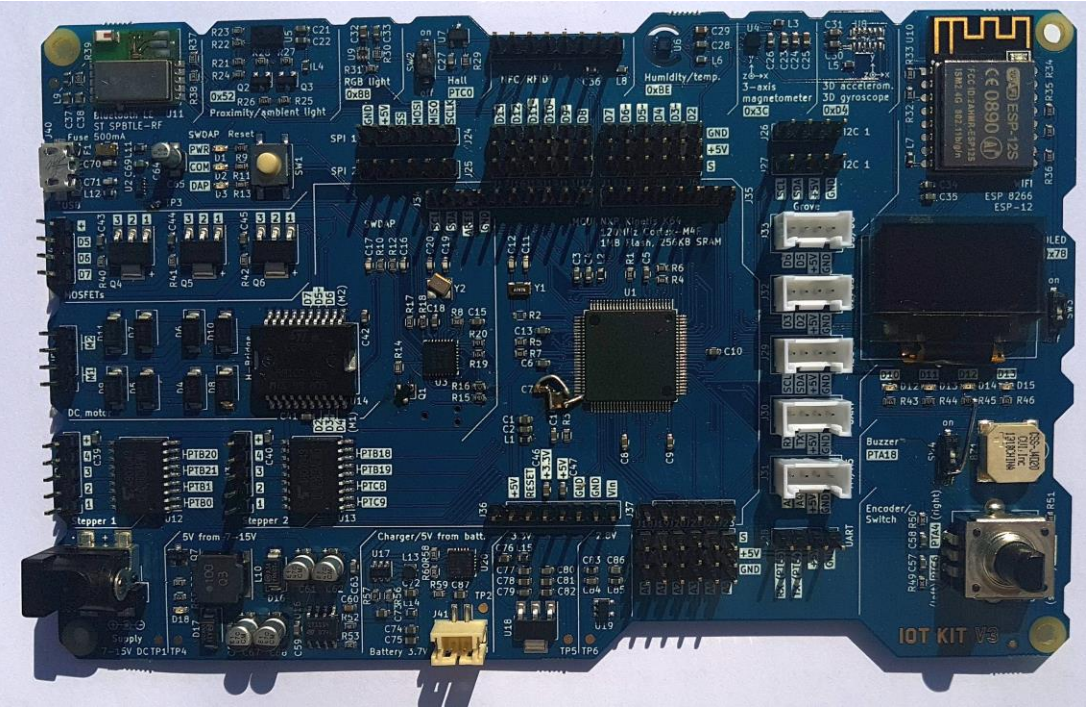


Entwicklungsumgebung - https://os.mbed.com/



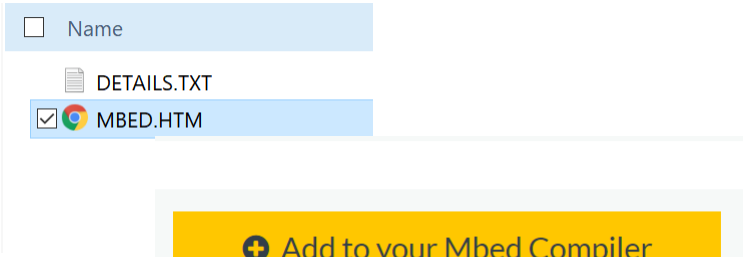
- ★ Mbed OS ist das führende Open-Source-RTOS für das Internet der Dinge und beschleunigt die Erstellung und den Einsatz von IoT-Geräten auf Basis von Arm-Prozessoren.
- ★ Mit Mbed OS können Sie IoT-Software in C++ mit unserer kostenlosen Online-IDE entwickeln, optimierten Code mit dem Arm C / C ++ - Compiler generieren und auf hunderten von Hardwareplattformen ausführen.
- ★ Der Mbed OS-Stack enthält TLS, Netzwerk, Speicher und weitere Treiber und wird durch Tausende von Codebeispielen und Bibliotheken erweitert.

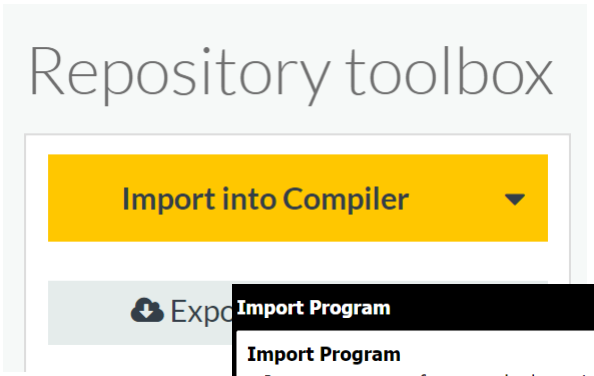
Entwicklungsboard - https://github.com/mc-b/iotkitv3

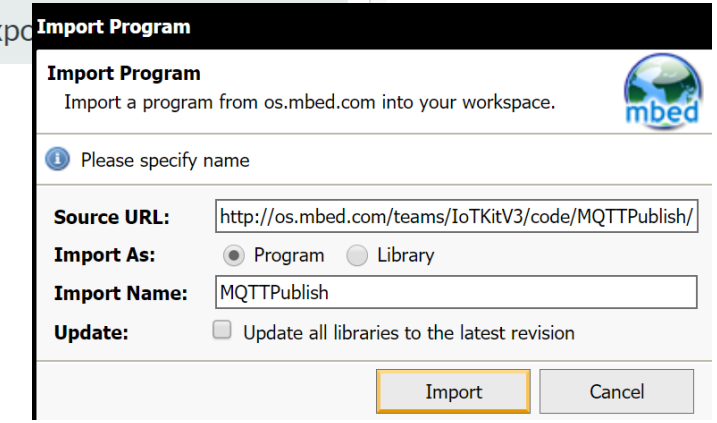


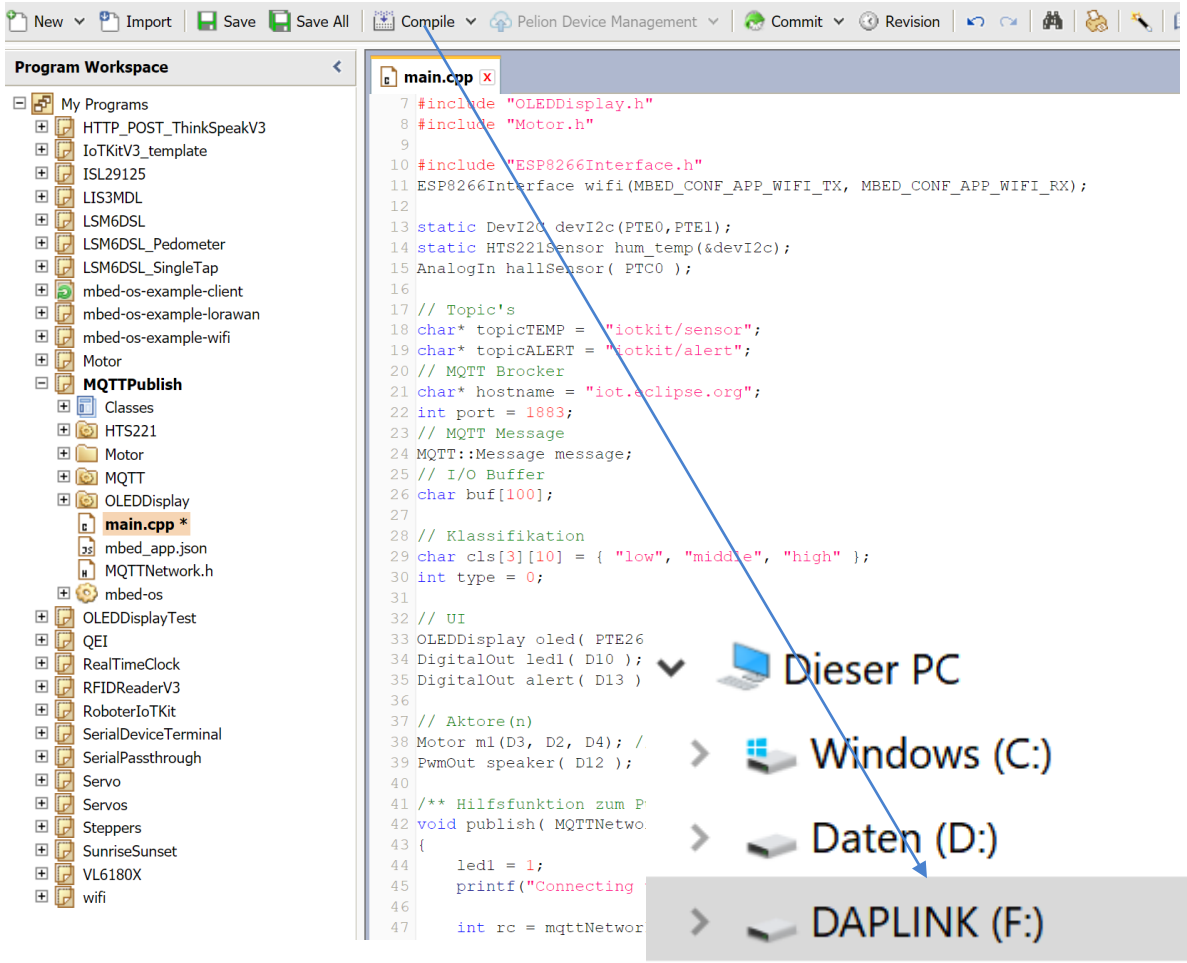
- ★ Das IoTKit V3 Board ist ein **mbed kompatibles** Entwickungsboard für die **Ausbildung**.
- ★ Der Kern basiert auf dem FRDM-K64f Board.
- ★ Es hat einen **Arduino** kompatiblen **Form Faktor** und diverse Sensoren, Aktoren und Kommunikationsmöglichkeiten on Board.
- ★ **Sensoren**: Distanz, RGB Licht, Hall, NFC/RFID, Temperatur und Luftfeuchtigkeit, Kompass, Beschleunigung und Lage, Encoder.
- ★ **Aktoren** (Driver): Elektromagnet, LED Streifen, DC Motor, Schrittmotor, OLED LCD, Buzzer
- ★ **Kommunikation/Bussysteme**: Bluetooth, WLAN, USB, UART, I^2C, SPI

Live Session: IoT Gerät Programmieren

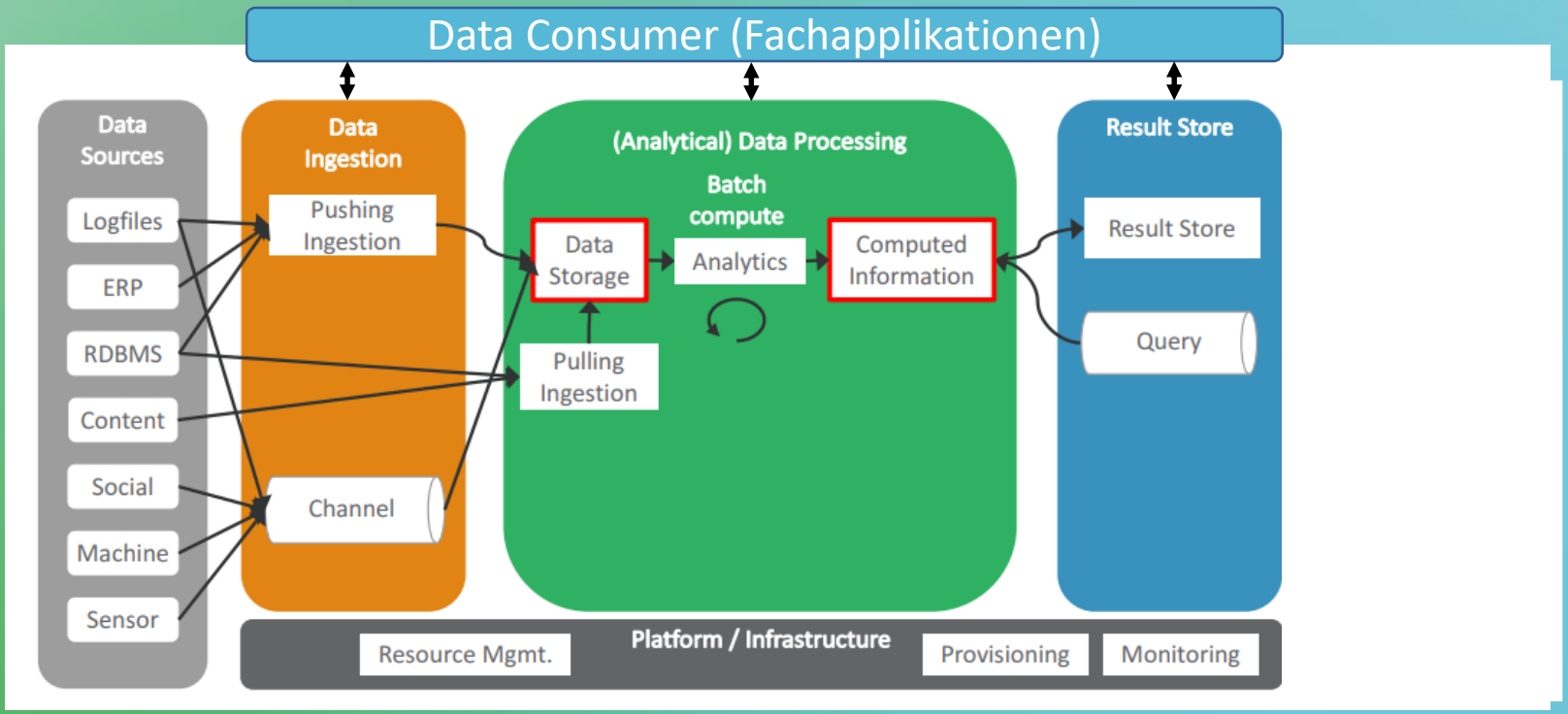








https://developer.mbed.org/getting-started/, https://developer.mbed.org/platforms/FRDM-K64F/#get-started-with-mbed



Machine Learning

(Analytical) Data Processing

(Eine) Definition

- ★ Maschinelles Lernen ist ein Oberbegriff für die **«künstliche» Generierung von Wissen aus Erfahrung.**
- ★ Ein künstliches System **lernt aus Beispielen** und kann diese **nach** Beendigung der **Lernphase verallgemeinern**.
- ★ Das heisst, es werden nicht einfach die Beispiele auswendig gelernt, sondern es «**erkennt**» **Muster** und Gesetzmässigkeiten in den Lern**daten**.

- ★ **Wir brauchen nicht für jedes Problem einen Programmierer!**

IoT und Machine-Learning: im Einsatz

DEV {}</> Day by digicomp

8.2 Energieeffizienz-Modul

Das Energieeffizienz-Modul vereinigt Sensoren zu Druck und Durchfluss, autarke Datenverarbeitung, ein 2/2-Wege-Absperrventil und eine Ethernet-Kommunikationsschnittstelle in sich. Die Kommunikationsparameter der Schnittstelle (übliche Feldbusse, OPC-UA, Modbus/TCP) sind offengelegt. Über die Integration einer Codesys-Steuerung sind Funktionen nachladbar. Das Modul überwacht laufend den Luftverbrauch der nachgeschalteten Anlage und kann dank Machine-Learning zwischen Ruhezustand, Betriebszustand und abnormalen Zuständen unterscheiden. Feste Grenzen für die einzelnen Zustände können ebenfalls eingestellt werden. Das Absperrventil lässt eine Auto-Stopp-Funktion zu, bei der die Druckluftzufuhr nach einer einstellbaren Zeit des Ruhezustands automatisch abgesperrt wird, um Leckage zu vermeiden. Sensordaten, Betriebszustände und Verhalten des Absperrventils sind über die Kommunikationsschnittstelle zugänglich.

Bild 6: Energieeffizienz-Modul

Quelle: Festo

- ★ Das Modul überwacht laufend den Luftverbrauch der nachgeschalteten Anlage und kann dank Machine-Learning zwischen Ruhezustand, Betriebszustand und abnormalen Zuständen unterscheiden.
- ★ Leitfaden: Welche Kriterien müssen Industrie-4.0-Produkte erfüllen?

IoT und Machine Learning: Anwendung

IoTKit: Sensordaten (mit Betriebszustand = Label)
offset = 114, value = 0xBC,21.00,51.4, low
offset = 115, value = 0xBC,26.00,51.2, middle
offset = 116, value = 0xBC,26.00,51.2, middle
offset = 117, value = 0xBC,31.10,51.2, high

```
In [25]: clf.score(X_train, y_train)
Out[25]: 1.0

In [26]: clf.score(X_test, y_test)
Out[26]: 0.9499999999999996
```

- ★ Sensordaten Klassifizieren: **Ruhezustand**, **Betriebszustand** und **abnormalen** Zuständen
- ★ Vorgehen
  - Bestimmung der Strategie, wie der Computer lernen soll.
  - Modell mit Testdaten trainieren.
  - Sensordaten an Trainiertes Modell übergeben, analog Testdaten.
  - Die Abweichung zu 1.0 liefert Rückschlüsse über den Zustand der Maschine.
  - Eine zu grosse Abweichung zu 1.0 löst den Wartungs- oder Fehlerbehebungsprozess aus.

Live Session: ML/IoT (02-2-MLTempHumSensor)

DEV {}</> Day by digicomp

Machine Learning mit IoT Testdaten

Basierend auf den Daten vom Temperatur und Luftfeuchtigssensor.

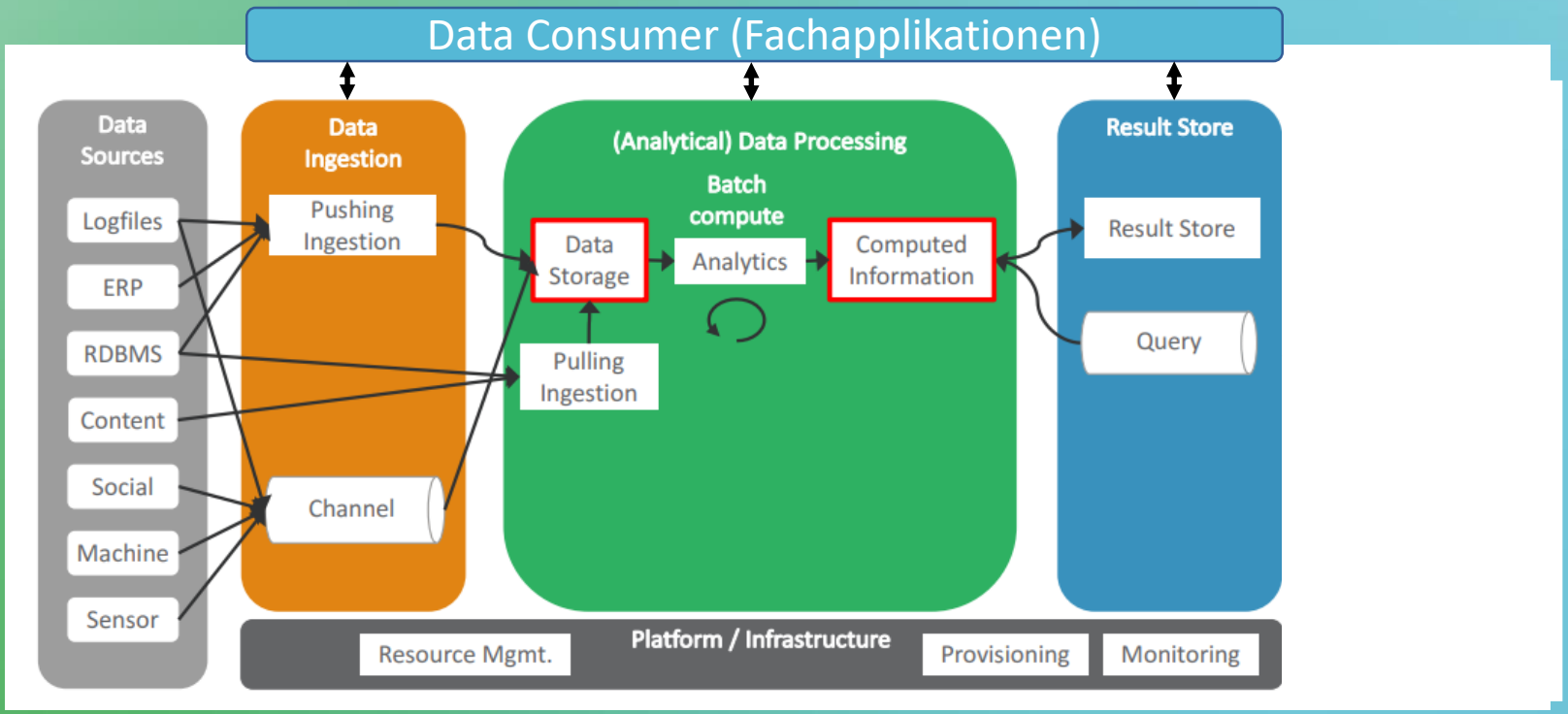
Zuerst holen wir alle Libraries wo benötigt werden

In [1]:
```
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
%pylab inline
import matplotlib.pylab as plt
import numpy as np
from distutils.version import StrictVersion
import sklearn
print(sklearn.__version__)

assert StrictVersion(sklearn.__version__ ) >= StrictVersion('0.18.1')
```

```
Populating the interactive namespace from numpy and matplotlib
0.19.2
```

Daten einlesen und evtl. Schmutz entfernen

In [2]:
```
import pandas as pd

df = pd.read_csv('data.csv', header=None, names=['sensor', 'temp', 'hum', 'class'] )
```



Docker und Kubernetes

Platform / Infrastructure

Kubernetes (K8s) im Überblick

DEV {}</>
Day
by digicomp

- ★ Das im Juli 2014 gestartete (griechisch: Steuermann) stellt die derzeit populärste Container-Cluster-/Orchestrierungs-Lösung dar.
- ★ Kubernetes ist mittlerweile bei der Cloud Native Computing Foundation (CNCF) gehostet.
- ★ **Kubernetes' Hauptaufgabe ist die Verwaltung und Orchestrierung der Container innerhalb eines Clusters, der üblicherweise aus mindestens einem Kubernetes Master und multiplen Worker Nodes besteht.**
- ★ Kubernetes wurde von Google ursprünglich als Orchestrierungs-Framework unter der Code-Bezeichnung Borg initiiert. Ziel war es, Docker- oder Rocket-(CoreOS)-Container im grossem Umfang zu deployen, zu administrieren und transparent zu orchestrieren.
- ★ Da Google Kubernetes bzw. seine Google-interne Implementierung davon selbst einsetzt, ist relativ sicher, dass K8s kein kurzlebiges Projekt sein wird.
- ★ **K8s arbeitet primär mit Docker-Containern.**

Quelle: Buch Skalierbare Container-Infrastrukturen, Kapitel 13

Kubernetes (K8s) Merkmale

- ★ Immutable (Unveränderlich) statt Mutable.
- ★ **Deklarative** statt Imperative (Ausführen von Anweisungen) **Konfiguration**.
- ★ **Selbstheilende Systeme - Neustart bei Absturz.**
- ★ Entkoppelte APIs – LoadBalancer / Ingress (Reverse Proxy).
- ★ **Skalieren (Vertikal/Horizontal) der Services** durch Änderung der Deklaration**.**
- ★ Anwendungsorientiertes statt Technik (z.B. Route 53 bei AWS) Denken.
- ★ **Abstraktion der Infrastruktur** statt in Rechnern Denken.

Deklaration: YAML (Dateien)

```yaml
apiVersion: v1
kind: Service
metadata:
  name: kafka
  labels:
    app: kafka
    name: kafka
spec:
  ports:
    - port: 9092
  selector:
    app: kafka
    tier: middleware
  clusterIP: None
```

```yaml
apiVersion: apps/v1beta2 # for versions before
kind: Deployment
metadata:
  name: kafka
  labels:
    app: kafka
spec:
  selector:
    matchLabels:
      app: kafka
      tier: middleware
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: kafka
        tier: middleware
    spec:
      containers:
      - name: kafka
        image: confluentinc/cp-kafka:latest
        imagePullPolicy: IfNotPresent
```

- YAML ist eine vereinfachte Auszeichnungs-sprache (englisch markup language) zur Datenserialisierung, angelehnt an XML (ursprünglich) und an die Datenstrukturen in den Sprachen Perl, Python und C sowie dem in RFC2822 vorgestellten E-Mail-Format.
- Die grundsätzliche Annahme von YAML ist, dass sich jede beliebige Datenstruktur nur mit assoziativen Listen, Listen (Arrays) und Einzelwerten (Skalaren) darstellen lässt. Durch dieses einfache Konzept ist YAML wesentlich leichter von Menschen zu lesen und zu schreiben als beispielsweise XML, ausserdem vereinfacht es die Weiterverarbeitung der Daten, da die meisten Sprachen solche Konstrukte bereits integriert haben.

Befehl um YAML Dateien Auszuwerten

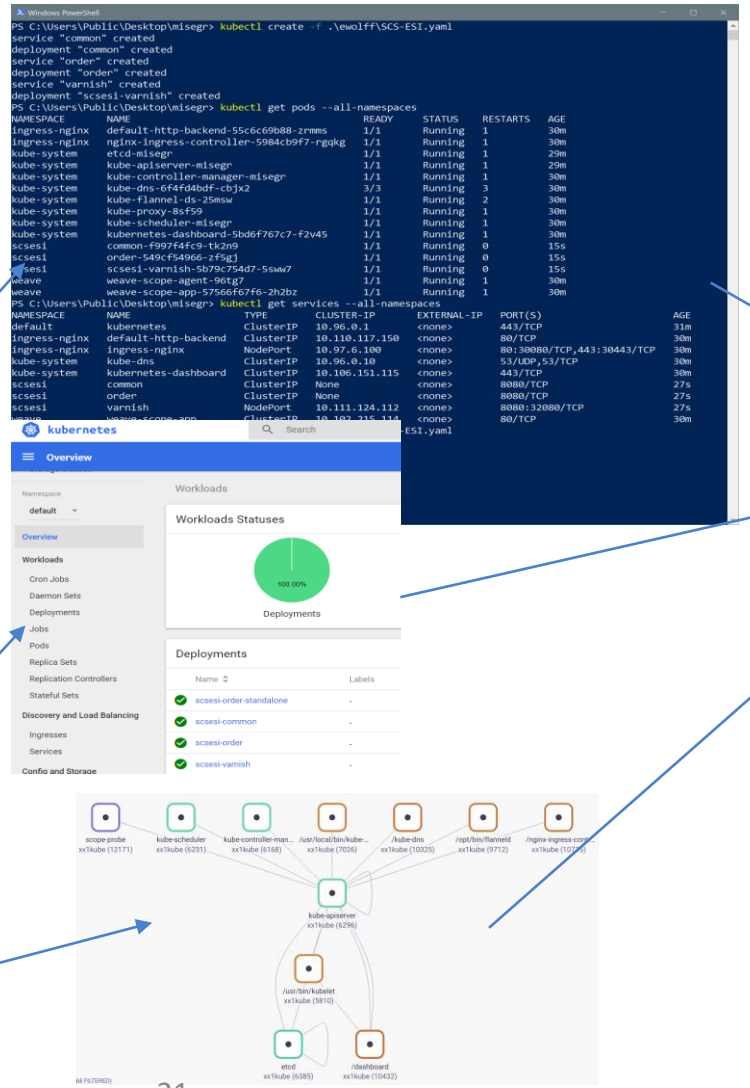
- ★ Das **kubectl**-Kommando stellt, eine der Schaltzentralen des K8s Clusters zur Administration der Ressourcen dar.
- ★ Die wichtigsten kubectl-Subkommandos in der Übersicht
  - `kubectl create -f YAML-Datei` – Erstellt eine Ressource laut YAML Datei
  - `kubectl run` – startet ein Container Image
  - `kubectl get [all]` – zeigt Resourcen an
  - `kubectl explain [pods]` – zeigt die Dokumentation zu einer Ressource an
  - `kubectl delete -f YAML-Datei` – Löscht eine Ressource laut YAML Datei
  - `kubectl exec` – startet einen Befehl im Container
  - `kubectl apply -f YAML-Datei` – führt die Änderungen in der YAML im Cluster nach
  - `kubectl cluster-info` – Liefert Informationen zum K8s Cluster

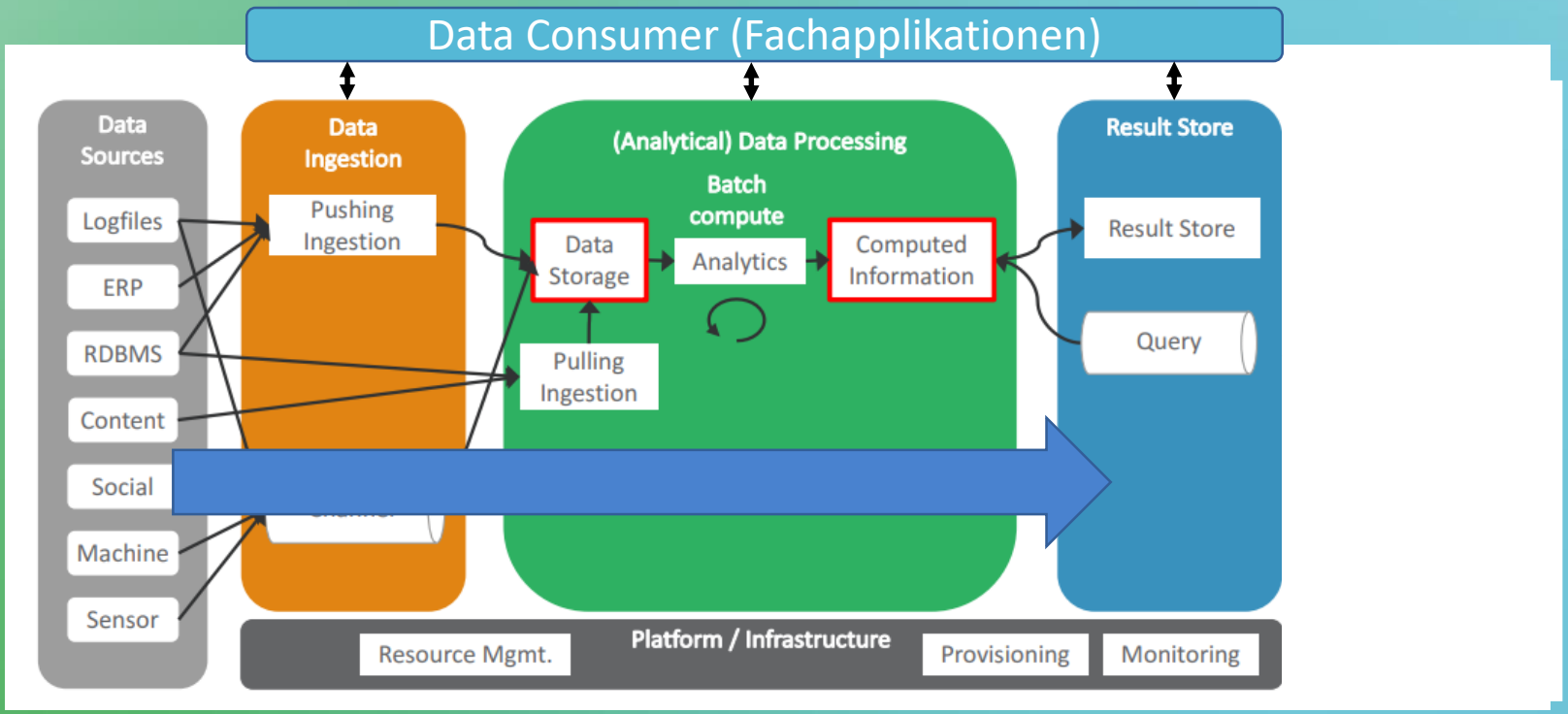
Quelle: Buch Skalierbare Container-Infrastrukturen, Kapitel 14.2

Live Session: Docker und Kubernetes

DEV {}</> Day by digicomp

- Alle Beispiele laufen auf einer Linux Umgebung (in einer Virtuellen Maschine) mit installierter «Internet of Things», «Machine Learning», «Data Processing», Docker und Kubernetes Infrastruktur.
- **Kommandozeile** (CLI)*: Starten mittels **kubeps.bat (PowerShell)** oder **kubesh.bat (Bash)**, im Verzeichnis lernkube, und Hilfsprogramm **kubectl** zum Erstellen, Start UI, Löschen von Services, z.B.:
  - `kubectl apply -f duk/osticket`
  - `startsrv osticket`
  - `kubectl delete -f duk/osticket`
- **Dashboard**: Starten mittels **dashboard.bat**. Details siehe https://github.com/kubernetes/dashboard
- **Weave Scope:** Starten in CLI mittels **weave.bat**. Details siehe: https://github.com/weaveworks/scope





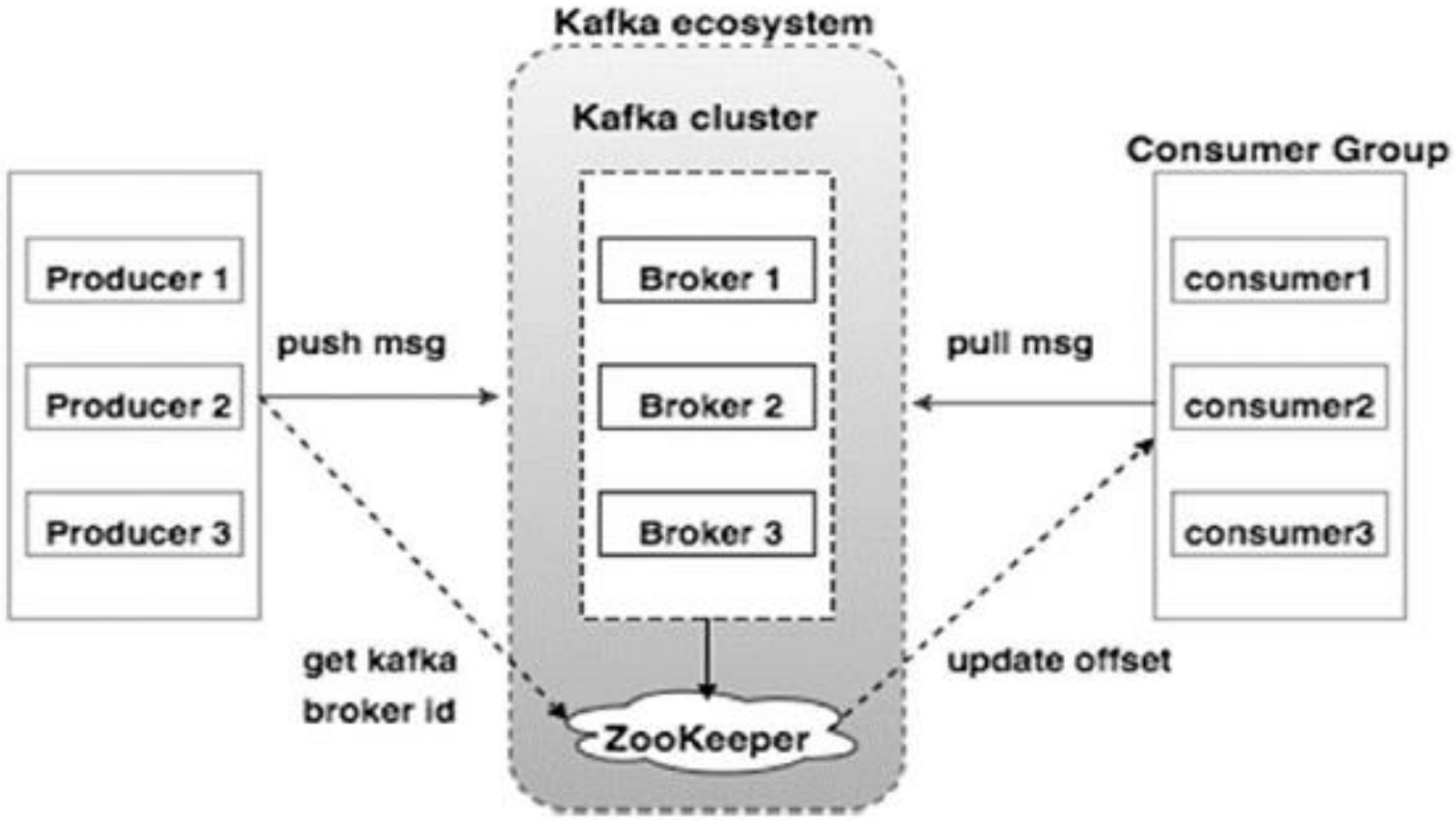
Fast Data Pipeline

Kafka - Streaming-Plattform (Messaging)

- ★ **Apache Kafka® ist eine verteilte Streaming-Plattform.**
- ★ Eine Streaming-Plattform hat drei Schlüsselfunktionen:
  - **Veröffentlichen** und abonnieren Sie Streams von Datensätzen, ähnlich einer Nachrichtenwarteschlange oder eines Enterprise-Messaging-Systems.
  - **Speichern** Sie Streams von Datensätzen auf eine fehlertolerante, dauerhafte Weise.
  - **Verarbeiten** von Datenströmen, wenn sie auftreten.
- ★ Kafka wird im Allgemeinen für zwei grosse Klassen von Anwendungen verwendet:
  - Erstellen von **Echtzeit-Streaming-Datenpipelines**, die zuverlässig Daten zwischen Systemen oder Anwendungen transportieren.
  - Erstellen von **Echtzeit-Streaming-Anwendungen**, die die Datenströme **transformieren** oder darauf reagieren.
- ★ Kafka und Kubernetes: https://www.confluent.io/confluent-operator/

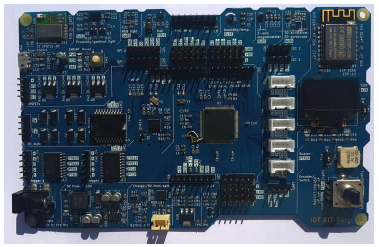
Quelle: https://kafka.apache.org/intro

Kafka – Zusammenspiel



Quelle: https://www.tutorialspoint.com/apache_kafka/apache_kafka_cluster_architecture.htm

Fast Data Pipeline im Einsatz



IoT Gerät

- **Data Sources (Sensoren)**
- mbed
- MQTT
-

Edge Gateway

- **Data Ingestion**
- Linux/Docker
- Mosquitto / Kafka
- MQTT –> TCP Stream

Data Center

- **Data Processing**
- Linux/Docker/Kubernetes
- Kafka
- TCP Stream Consumer

Data Consumer

Machine Learning

Fast Data Pipeline: IoT Gerät


```
// QoS 0
char buf[100];
sprintf(buf, "sensor: [ temp: 24.2,\n version: %f ]\n", version);
message.qos = MQTT::QOS0;
message.retained = false;
message.dup = false;
message.payload = (void*)buf;
message.payloadlen = strlen(buf)+1;
rc = client.publish(topic, message);
```

- ★ Gerät mit ARM Cortex M Prozessor. Optimiert für sehr niedrigen Stromverbrauch.
- ★ Pooling der Sensoren
- ★ Alert durch Hall Sensor (Magnetsensor)
- ★ Veröffentlichung der Sensordaten im MQTT-Format (Producer)
  - Topics: iotkit/sensor + iotkit/alert

- ★ Beispiel: https://os.mbed.com/teams/IoTKitV3/code/MQTTPublish/

Fast Data Pipeline: Gateway

Eclipse Mosquitto™

An open source MQTT broker

```
docker run --rm -ti --name mqtt-kafka-bridge devicexx/mqtt-kafka-bridge [options...]
```

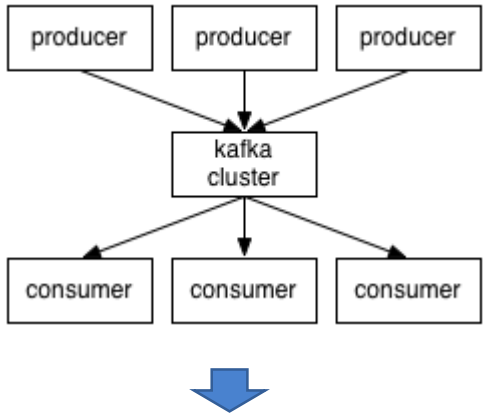
Where options are:

```
--help (-h)                : Show help
--id VAL                   : MQTT Client ID
--topics VAL               : MQTT topic filters (comma-separated)
--uri VAL                  : MQTT Server URI
--brokerlist (-b) VAL      : Kafka Broker List (comma-separated)
```

```
KStream<String, String> source = builder.stream( "broker_message" );
KStream<String, String> output = source.flatMapValues( new ValueMapper<String, Iterable<String>>()
{
    @Override
    public Iterable<String> apply(String value)
    {
        String[] values = value.split( "," );
        // Temperatur und Luftfeuchtigsensor
        if  ( "0xBC".equals( values[0].trim() ) )
        {
            System.out.println( "{" + "\"humtemp\": { \"temp\": " + values[1] + ", \"hum\": " + val
            return  ( Arrays.asList( new String[] { "{", "\"humtemp\": { \"temp\": ", values[1], ",
        }
        System.out.println( Arrays.asList( value.split( "," ) ) );
        return Arrays.asList( value.split( "," ) );
    }
} );
output.to( "iot" );
```

- ★ Cloud Service: http://iot.eclipse.org
- ★ Alternative: Gerät mit ARM Cortex A Prozessor, z.B. Raspberry Pi.
- ★ Linux und optional Docker installiert.
- ★ Mosquitto als MQTT Broker.
- ★ mqttKafkaBridge als MQTT -> Kafka Bridge
- ★ Kafka Streams zur Umwandlung und Anreicherung der Daten, z.B.:
  - Mapping MQTT Topic nach JSON
  - Anreicherung um Kunden/Edge ID der Sensordaten
- ★ Beispiele: https://github.com/mc-b/duk/tree/master/kafka
  https://github.com/mc-b/iot.kafka.git

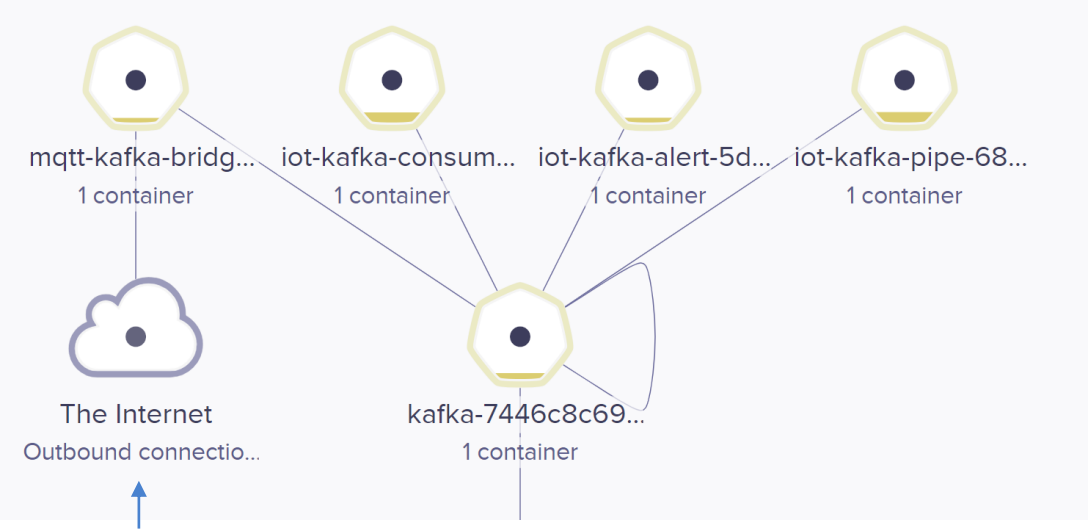
Fast Data Pipeline:Data Center

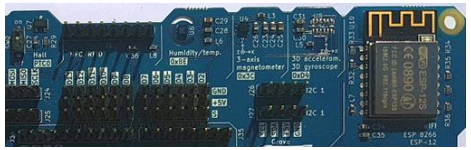



```
public static void main(String[] args) throws Exception
{
    Properties props = new Properties();
    props.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "kafka:9092");
    props.put( "group.id", "iot" );
    props.put( "enable.auto.commit", "true" );
    props.put( "auto.commit.interval.ms", "1000" );
    props.put( "key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer" );
    props.put( "value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer" );
    KafkaConsumer<String, String> consumer = new KafkaConsumer<>( props );
    consumer.subscribe( Arrays.asList( "streams-plaintext-input", "broker_message" ) );

    while (true)
    {
        ConsumerRecords<String, String> records = consumer.poll( 100 );
        for ( ConsumerRecord<String, String> record : records )
            System.out.printf( "offset = %d, value = %s%n", record.offset(), record.value() );
    }
}
```

- ★ Linux mit Docker, Kubernetes und Kafka
- ★ Hochverfügbar
- ★ Entgegenahme und Aufbereitung der Sensoren für:
  - Big Data Analysen
  - Machine Learning
  - Protokollierung
  - Speichern in Datenbank
  - …
- ★ Beispiele: https://github.com/mc-b/duk/tree/master/kafka
  https://github.com/mc-b/iot.kafka.git

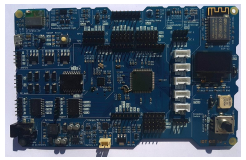
Live Session: Fast Data Pipeline



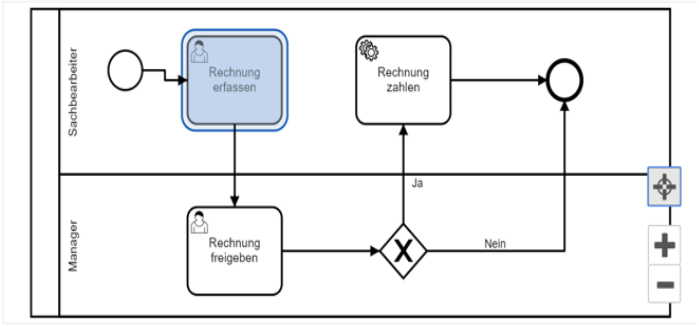


- ★ **Starten**
  - `kubectl create -f devdays/iotmldata/`
- ★ **Producer**
  - **IoTKit V3** sendet Daten an iot.eclipse.org im MQTT Format
- ★ **Gateway**
  - **mqtt-kafka-bridge** nimmt diese entgegen, wandelt diese ins ein Kafka kompatibles Format und leitet die Daten an Kafka weiter.
- ★ **Consumer**
  - **iot-kafka-consumer** bereitet die Daten im CSV-Format für Machine Learning auf.
  - **iot-kafka-alert** horcht auf Ausnahmefehler und startet einen Prozess (siehe n. Folie).
  - **iot-kafka-pipe** wandelt die Daten im JSON-Format, z.B. für Fachapplikationen auf.

Live Session: IoT und Prozesse



```
System.out.println( "AlertConsumer" );
while (true)
{
    ConsumerRecords<String, String> records = consumer.poll( 100 );
    for ( ConsumerRecord<String, String> record : records )
    {
        long offset = record.offset();
        String value = record.value();
        if ( value != null && value.startsWith( "alert" ) )
        {
            String text =  "{ \"variables\": { \"rnr\": {\"value\": " + 123 + ", \"type\": \"long\"}, " +
                           "\"rbetrag\": {\"value\": " + 100.0 + ", \"type\": \"String\"} } }";
            System.out.printf( "offset = %d, value = %s%n", offset, record.value() );
            Response rc = client.target( REST_URI )
                           .path( "RechnungStep3/start" )
                           .request(MediaType.TEXT_PLAIN)
                           .post(Entity.entity( text , MediaType.TEXT_PLAIN ) );
            System.out.println( rc.getStatus() );
        }
    }
```



- ★ IoT Geräte sendet laufend Daten an Edge / Data Center
- ★ Ein Kafka Server empfängt die Daten.
- ★ Ein Kafka Consumer filtert «Alerts» Events und startet bei Bedarf einen BPMN Prozess, z.B. Aufbietung Service Techniker.

- ★ Details:
  - https://github.com/mc-b/misegr/tree/master/bpmn
  - https://github.com/mc-b/iot.kafka

Zusammenfassung

- ★ IoT ermöglicht neue Arten von Produkten, Kundenbeziehungen, Anwendungen.
- ★ Machine Learning ermöglicht ein besseres Verständnis unserer Umwelt und der Kunden.
- ★ Docker (Container) und Kubernetes abstrahiert, vereinheitlicht Plattform und Infrastruktur.
- ★ Messaging System wie Kafka verbinden die verschiedenen Komponenten.
- ★ (BPMN) Prozesse Ergänzen die Pipeline

Kurse

- ★ Microservices-Grundlagen («MISEGR»)
- ★ Docker und Kubernetes – Übersicht und Einsatz («DUK»)
- ★ Internet of Things (IoT) im Einsatz («IOTEIN»)
- ★ Machine Learning Grundlagen («MLG»)

Abschluss

★Viel Erfolg mit IoT, Machine Learning, Docker und Kubernetes!

★ Marcel Bernet