Home Smart Irrigation System using Spring & Apache Kafka

Trayan Iliev, IPT – IT Education Evolved, http://iproduct.org/

About Me

Trayan Iliev

– CEO of IPT – Intellectual Products & Technologies [https://iproduct.org/]

– Oracle® certified programmer 15+ Y

– End-to-end reactive full-stack apps with Go, Python, Java, ES / TypeScript, Angular, React and Vue.js

– 12+ years IT trainer

– Voxxed Days, jPrime, Java2Days, jProfessionals, DEV.BG, BGOUG, BGJUG, speaker on topics like: Reactive Robotics & IoT, Java, Spring, Reactor, SOA & REST, high-performance Java & JavaScript programming, Intelligent Agents & ML

– Lecturer @ Sofia University – courses: Fullstack React, Multimedia with Angular & TypeScript, Web Apps & Services with Spring, Multiagent Systems and Social Robotics, Practical Robotics & Smart Things, Distributed Machine Learning with Applications in Robotics & IoT, Intelligent Agents using LLM

– Robotics / smart-things/ IoT enthusiast, RoboLearn hackathons organizer

Agenda

- Advantages of smart irrigation at home and in the garden.
- Comparison of different approaches to plant irrigation. Drip irrigation.
- Planning and physical construction of the irrigation system.
- Hardware: valves, filters, pressure regulators, flow meters, humidity sensors.
- Embedded software: Arduino with ESP32
- Communication protocols: Constrained Application Protocol (CoAP)
- Persistent data logging with Apache Kafka
- Web/Kafka server integration: Eclipse Californium, Spring for Apache Kafka, Reactive WebSocket with Spring WebFlux
- Web interface: ReactJS Dashboard
- Q&A

Advantages of Smart Irrigation for Homes and Gardens - I

Water Efficiency

- Uses soil moisture sensors, weather data, or plant needs to apply water only when necessary.
- Prevents overwatering and reduces water waste, helping conserve a precious resource.

Cost Savings

- Lower water bills by cutting unnecessary usage.
- Extends the lifespan of plants and garden systems, reducing replacement or repair costs.

Healthier Plants & Lawn

- Delivers the right amount of water at the right time, promoting stronger roots and healthier growth.
- Minimizes plant stress from under- or overwatering.

Convenience & Automation

- Schedules watering automatically, even when you're away.
- Can be controlled remotely through a smartphone or voice assistant.

Advantages of Smart Irrigation for Homes and Gardens - II

🌍 Environmental Benefits

- Reduces runoff, soil erosion, and nutrient leaching caused by excess watering.
- Supports sustainable gardening practices.

📊 Data & Adaptability

- Many systems adjust in real-time based on rainfall, temperature, and evaporation rates.
- Some provide usage data and insights, helping you track and optimize water consumption.

🏡 Added Value

- Modern, eco-friendly feature that increases property appeal.
- Creates a low-maintenance garden, ideal for busy homeowners.

Comparison of Traditional & Smart Irrigation

Feature	Traditional Irrigation	Smart Irrigation
Water Usage	Often wastes water due to fixed schedules	Uses sensors & weather data to water only when needed
Plant Health	Risk of overwatering or underwatering	Consistent watering, healthier plants & soil
Cost	Higher water bills	Saves money through reduced consumption
Convenience	Manual control or timer-based	Automated, can be controlled via smartphone/voice
Adaptability	Same schedule regardless of weather	Adjusts in real time (rain, temperature, soil moisture)
Environmental Impact	Can cause runoff, erosion, and nutrient loss	Conserves water, reduces runoff, supports sustainability
Maintenance	May require frequent adjustment	Low maintenance once set up
Property Value	Standard feature	Eco-friendly upgrade, adds modern appeal

Approaches to Plant Irrigation - I

1. Flood Irrigation (Surface Irrigation)

How it works: Water is released across the soil surface and allowed to soak in.

☑ Advantages: Simple, low-cost, minimal infrastructure.

❌ Disadvantages: High water loss (evaporation, runoff, deep percolation), uneven distribution, weeds thrive.

🌍 Best suited for: Large traditional farms with access to abundant water.

2. Sprinkler Irrigation

How it works: Water is sprayed through nozzles, imitating rainfall.

☑ Advantages: Suitable for many soil types and crops, uniform coverage, less labor.

❌ Disadvantages: Loses water to wind and evaporation, requires energy (pumps, pressure), wet leaves may promote disease.

🌍 Best suited for: Lawns, gardens, and large-scale farms where water supply is adequate.

Approaches to Plant Irrigation - II

3. Drip Irrigation (Micro-Irrigation)

How it works: Water is delivered slowly and directly to plant roots through tubing, emitters, or perforated pipes.

☑ Advantages (why it's superior):

- Saves up to **50–70% water** compared to flood/sprinkler.
- Delivers water **exactly where needed**, reducing waste.
- Keeps leaves dry, lowering risk of fungal diseases.
- Supports fertigation (nutrients delivered via water).
- Improves plant growth and yield with consistent moisture.

❌ Disadvantages:

- Higher initial installation cost.
- Requires maintenance (clogging of emitters).

🌍 Best suited for: Home gardens, orchards, vineyards, vegetable farms, and areas with limited water.

Approaches to Plant Irrigation - III

4. Soaker Hoses

How it works: Porous hoses laid on the soil release water gradually.

☑ Advantages: Affordable, easy to install for home gardens.

❌ Disadvantages: Less precise than drip irrigation, not ideal for large-scale farms.

🌍 Best suited for: Small-scale home gardens, raised beds.

5. Smart Irrigation Systems (Modern Layer on Top of Any Method)

How it works: Uses sensors, timers, and weather data to optimize water delivery (can be paired with drip, sprinkler, or soaker).

☑ Advantages: Maximizes efficiency, minimizes waste, very convenient.

❌ Disadvantages: Requires tech setup and higher cost.

🌍 Best suited for: Eco-conscious homeowners, urban gardens, and sustainable farms.

Planning Phase

1. **Survey the Area**
    - Map the garden/field (size, slope, sun exposure).
    - Note crop or plant locations, soil type, and water requirements.
    - Divide into zones (plants with similar needs grouped together).
2. **Choose Irrigation Method**
    - Drip irrigation for efficient root-level watering.
    - Sprinklers for lawns or uniform coverage areas.
    - Soaker hoses for small raised beds.
    - Hybrid systems if different areas have different needs.
3. **System Design**
    - Draw a layout (pipes, emitters, sprinklers, valves, pump).
    - Calculate water demand per zone.
    - Decide control system: manual, timer-based, or smart irrigation.
4. **Budget & Material Selection**
    - List required components: mainline pipes, lateral pipes, drip emitters, sprinklers, valves, filters, fittings
    - Estimate costs (equipment, installation, maintenance).

Physical Construction Phase - I

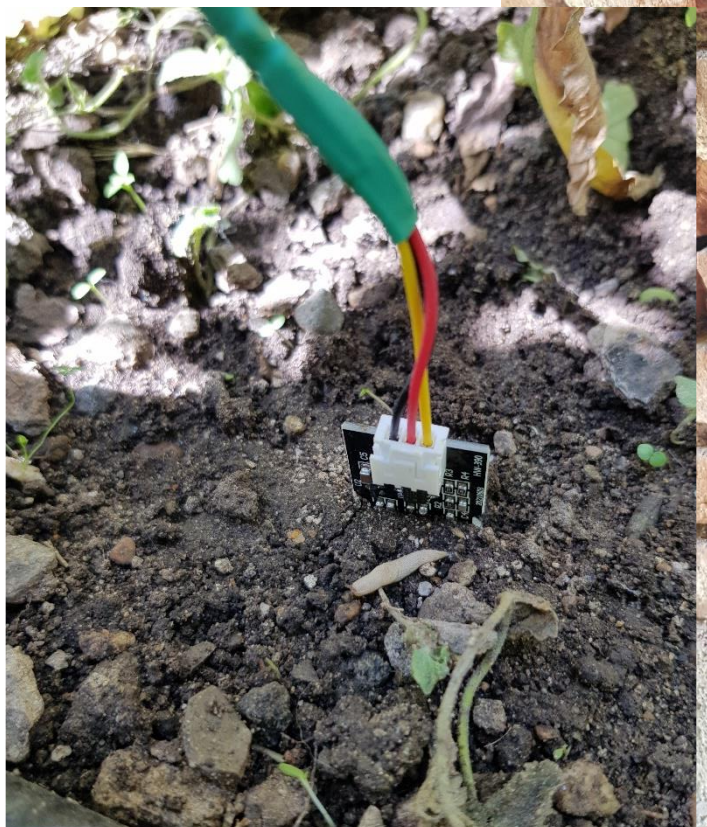
1. **Prepare the Site**
    - Clear the area of debris and weeds.
    - Level or contour if needed to avoid uneven water distribution.
2. **Install Main Line/Lines**
    - Lay the main water supply pipe from the source.
    - Ensure correct diameter for adequate flow.
3. **Set Up Control Units**
    - Install pump (if required), filter (to prevent clogging), pressure regulator, and valves.
    - Place smart controller / timer near the water source.
4. **Lay Lateral Lines & Emitters**
    - Extend smaller pipes (laterals) to plant zones.
    - Attach drip emitters / soaker hoses / sprinklers according to the design.
    - Keep drip emitters near root zones for maximum efficiency.

Physical Construction Phase - II

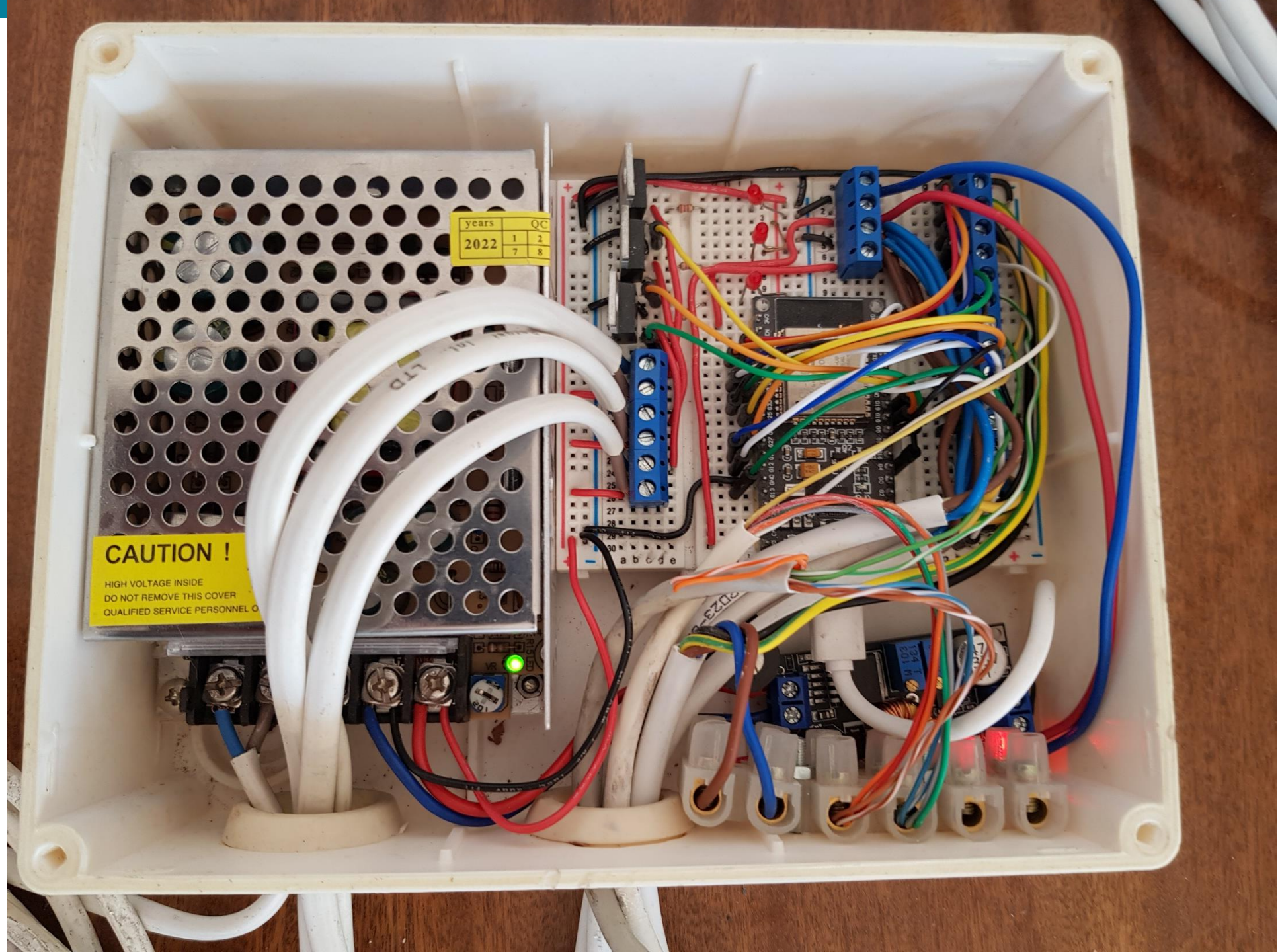
5. **Connect & Test**
    - Connect laterals to the main line with fittings.
    - Turn on the system to check flow, leaks, or pressure issues.
    - Adjust emitter/sprinkler positions for even coverage.
6. **Secure & Bury Pipes (if needed)**
    - Stake or bury pipes to avoid tripping hazards and UV damage.
    - Mulch over drip lines in gardens to reduce evaporation.
7. **Final Adjustments & Scheduling**
    - Program timers or smart controllers (time of day, duration, frequency).
    - Fine-tune based on soil moisture, plant type, and weather.

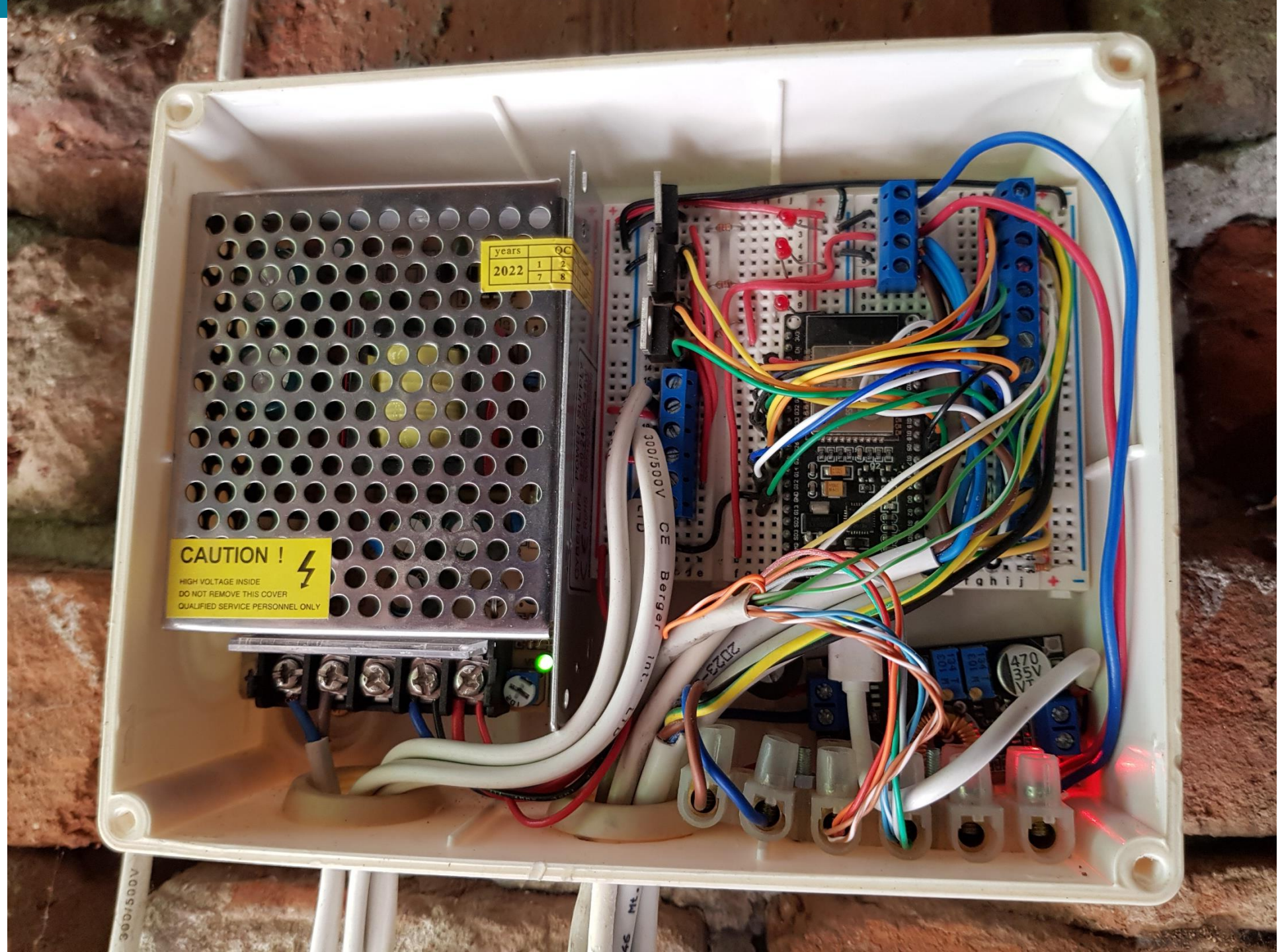
Hardware

1. Filters
2. Pressure regulators
3. Pressure gauges
4. Valves
5. Flow meters
6. Soil humidity sensors




Smart Irrigation
Sensor Data
Irrigation Control
Charts & Analytics
Flow Meter Data
Moisture Sensor Data


years
QC
2022
1
2
7
8
CAUTION !
HIGH VOLTAGE INSIDE
DO NOT REMOVE THIS COVER

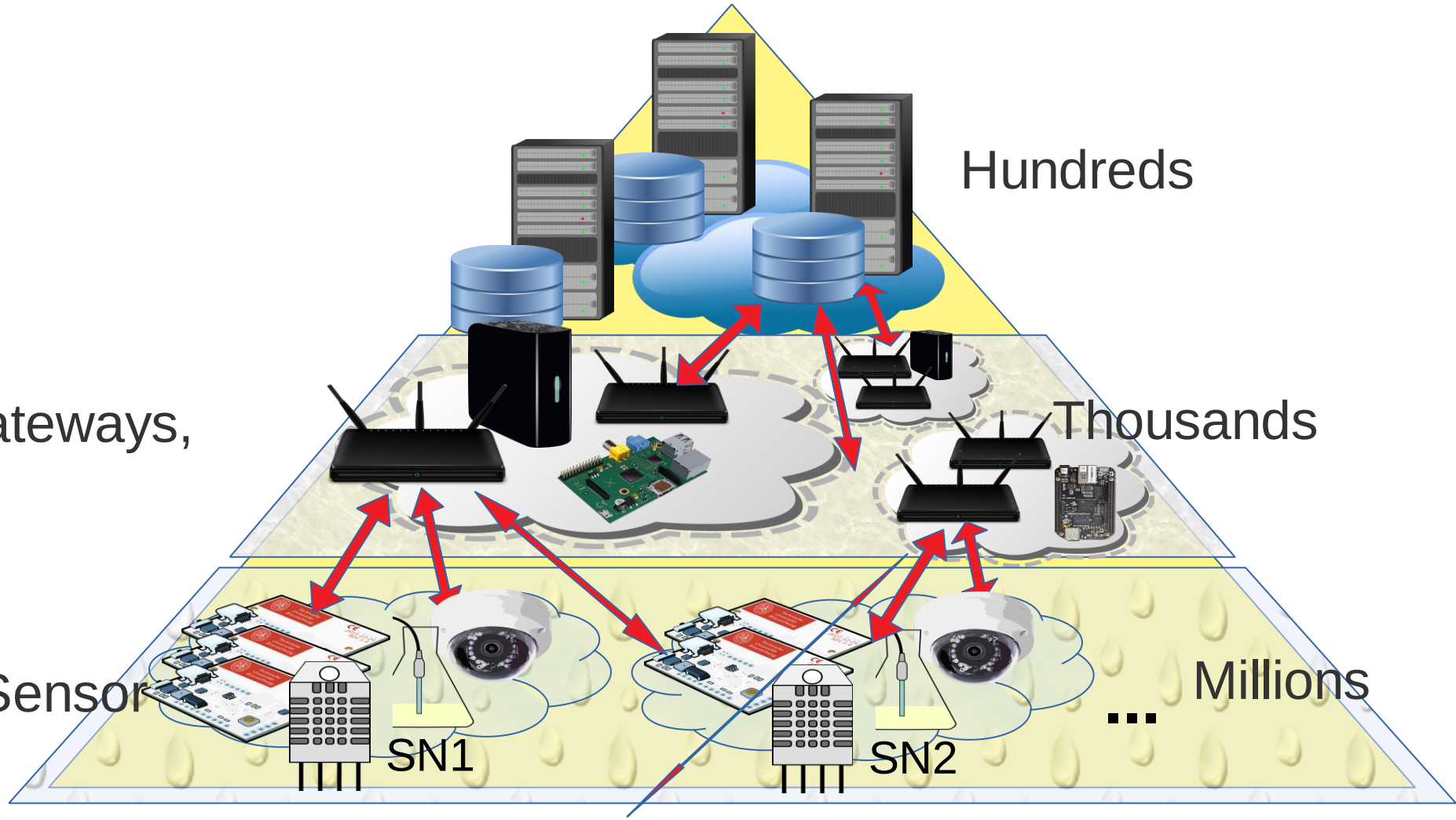

years
QC
2022
1
2
7
8
CAUTION !
HIGH VOLTAGE INSIDE
DO NOT REMOVE THIS COVER
QUALIFIED SERVICE PERSONNEL ONLY
300/500V CE Berger
470
35V

Cloud, Fog and Mist Computing

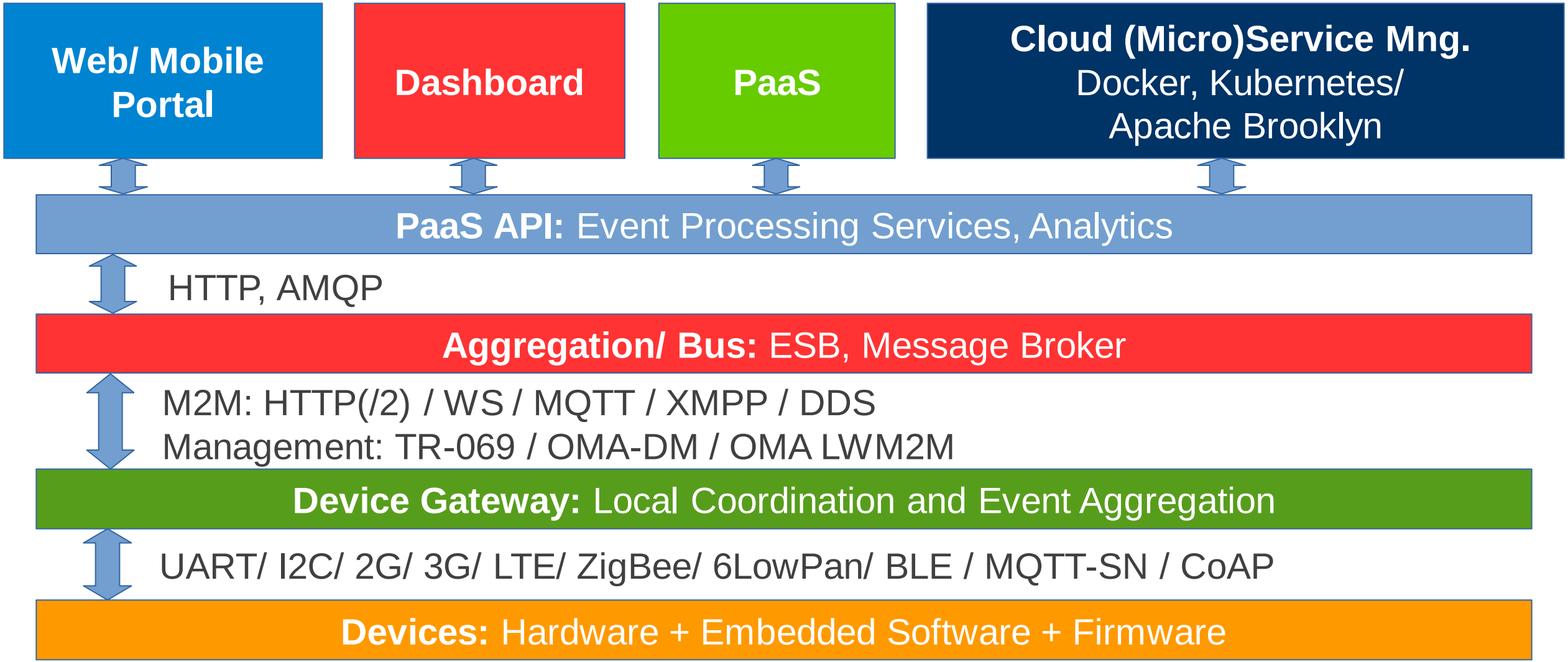
Cloud Computing
(Data-centers)

Fog Computing
(Fog Nodes, Edge Gateways, Cloudlets)

Mist Computing
(Smart IoT Devices, Sensor Networks)



IoT Services Architecture



Constrained Application Protocol (CoAP)

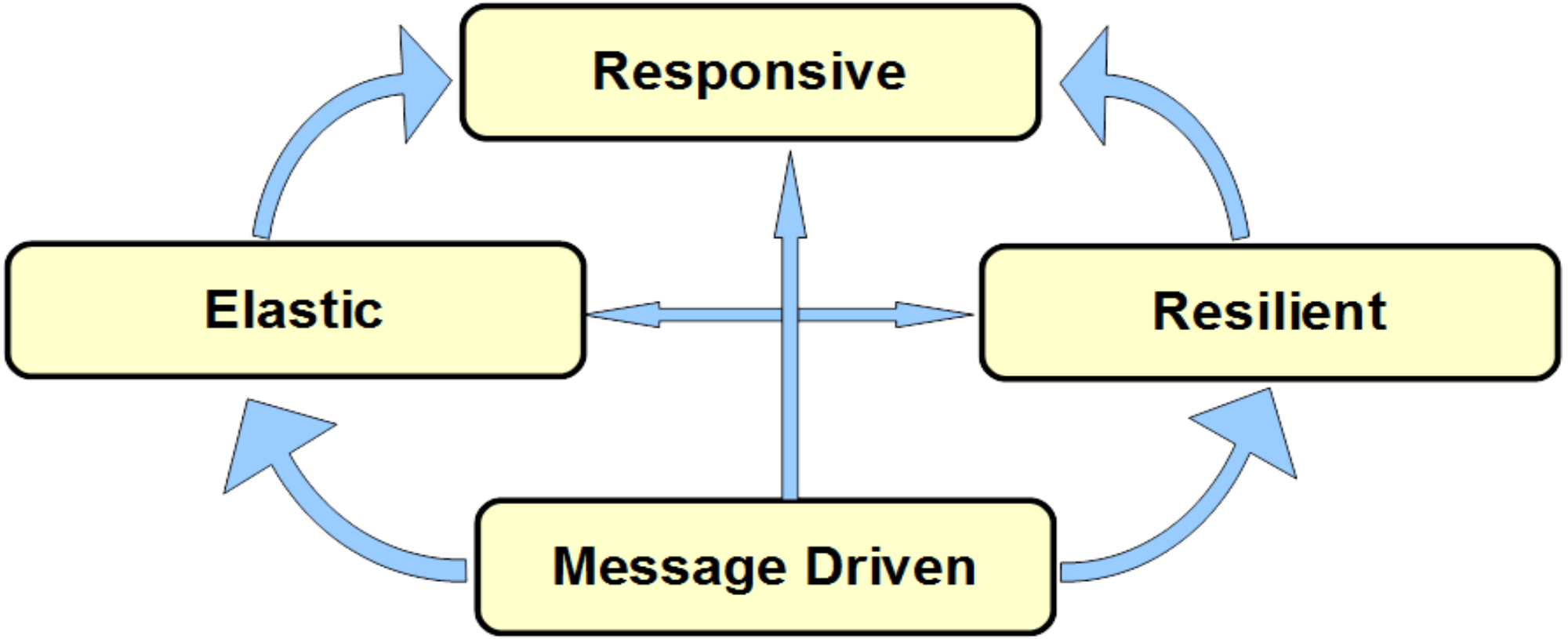
https://en.wikipedia.org/wiki/Constrained_Application_Protocol

- CoAP (RFC 7252) is an application-layer protocol that is intended for use in resource-constrained Internet devices, such as wireless sensor network nodes. CoAP is designed to easily translate to HTTP for simplified integration with the web.
- Multicast, low overhead, and simplicity are important for Internet of things (IoT) and machine-to-machine (M2M) communication, which tend to be embedded and have much less memory and power supply than traditional Internet devices have. Therefore, efficiency is very important.
- CoAP is used between devices on the same network (e.g., low-power, lossy networks),and between devices on different networks both joined by internet.
- CoAP can run on most devices that support UDP or a UDP analogue.
- Tutorial: https://www.startertutorials.com/blog/constrained-application-protocol.html

Apache Kafka

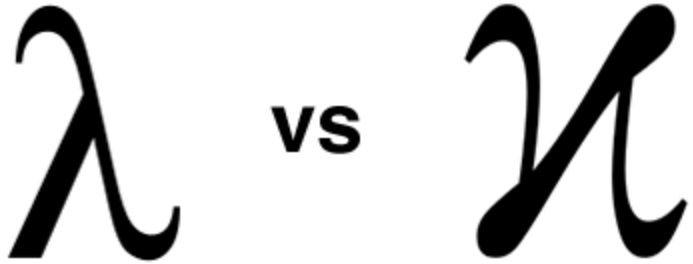
Reactive Manifesto

http://www.reactivemanifesto.org



Kappa Architecture

Query = K (New Data) = K (Live streaming data)

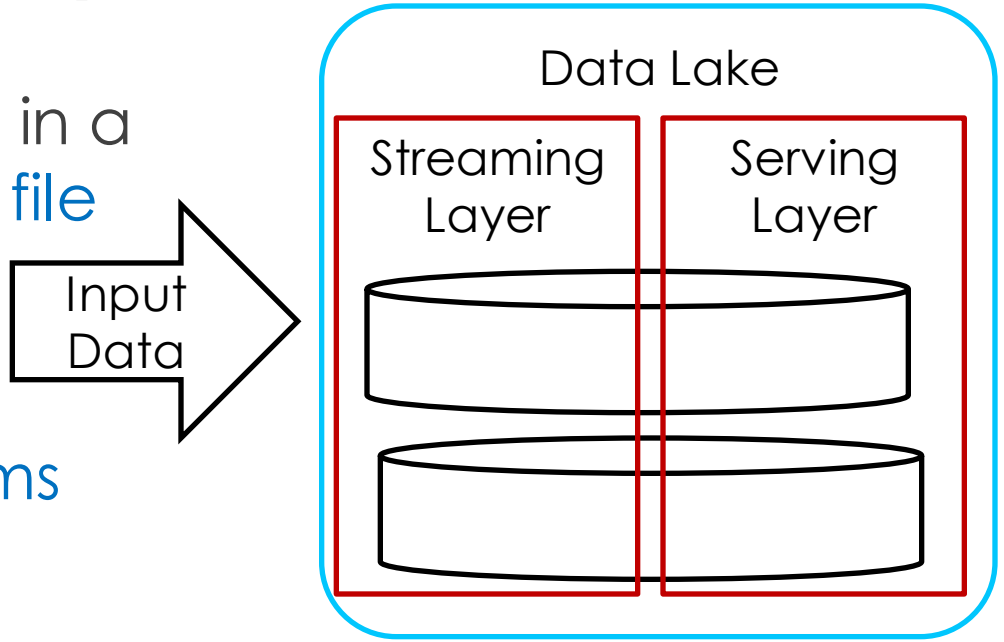


- Proposed by Jay Kreps in 2014
- Real-time processing of distinct events
- Drawbacks of Lambda architecture:
  - It can result in coding overhead due to comprehensive processing
  - Re-processes every batch cycle which may not be always beneficial
  - Lambda architecture modeled data can be difficult to migrate
- Canonical data store in a Kappa Architecture system is an append-only immutable log (like Kafka, Pulsar)

Kappa Architecture II

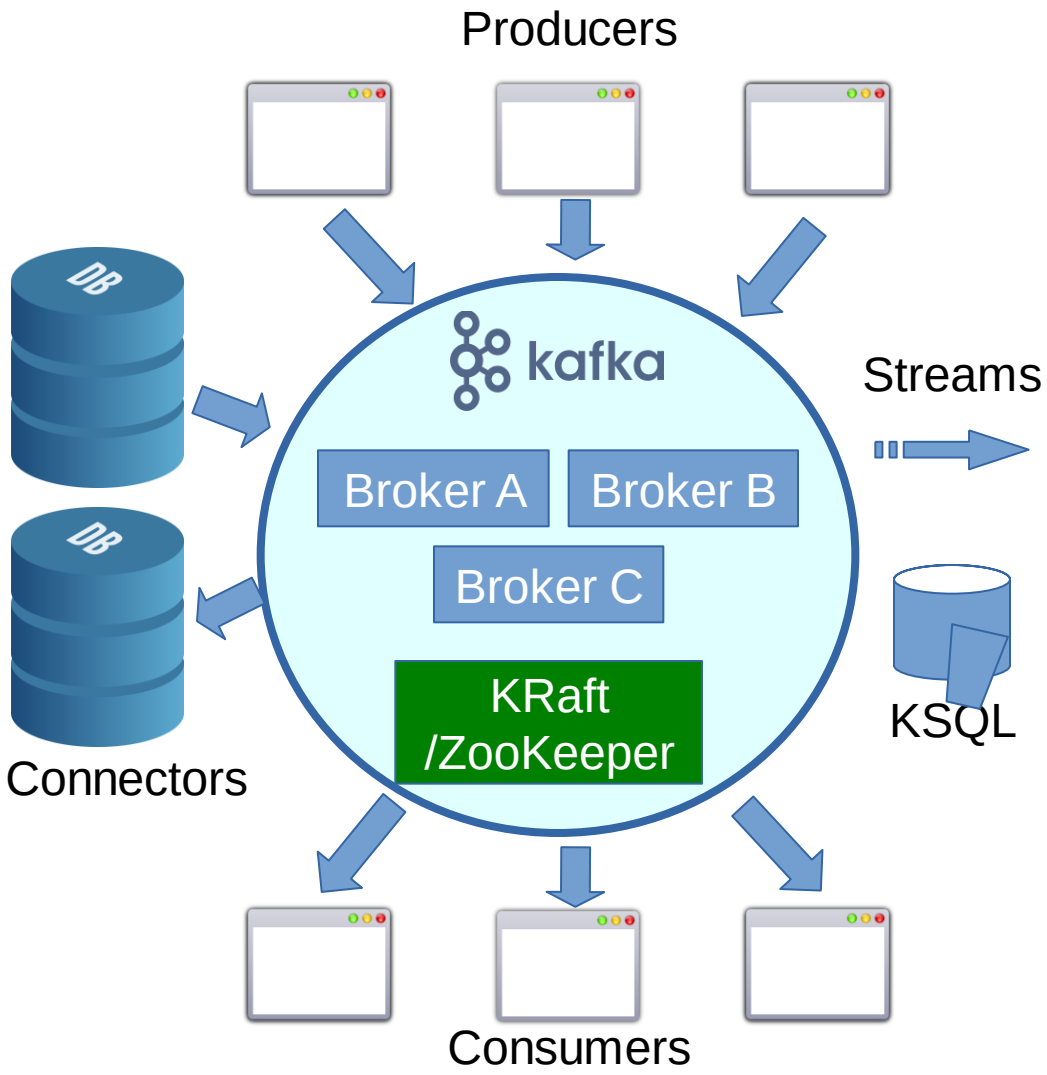
Query = K (New Data) = K (Live streaming data)

- Multiple data events or queries are logged in a queue to be catered against a distributed file system storage or history.
- The order of the events and queries is not predetermined. Stream processing platforms can interact with database at any time.
- It is resilient and highly available as handling terabytes of storage is required for each node of the system to support replication.
- Machine learning is done on the real time basis



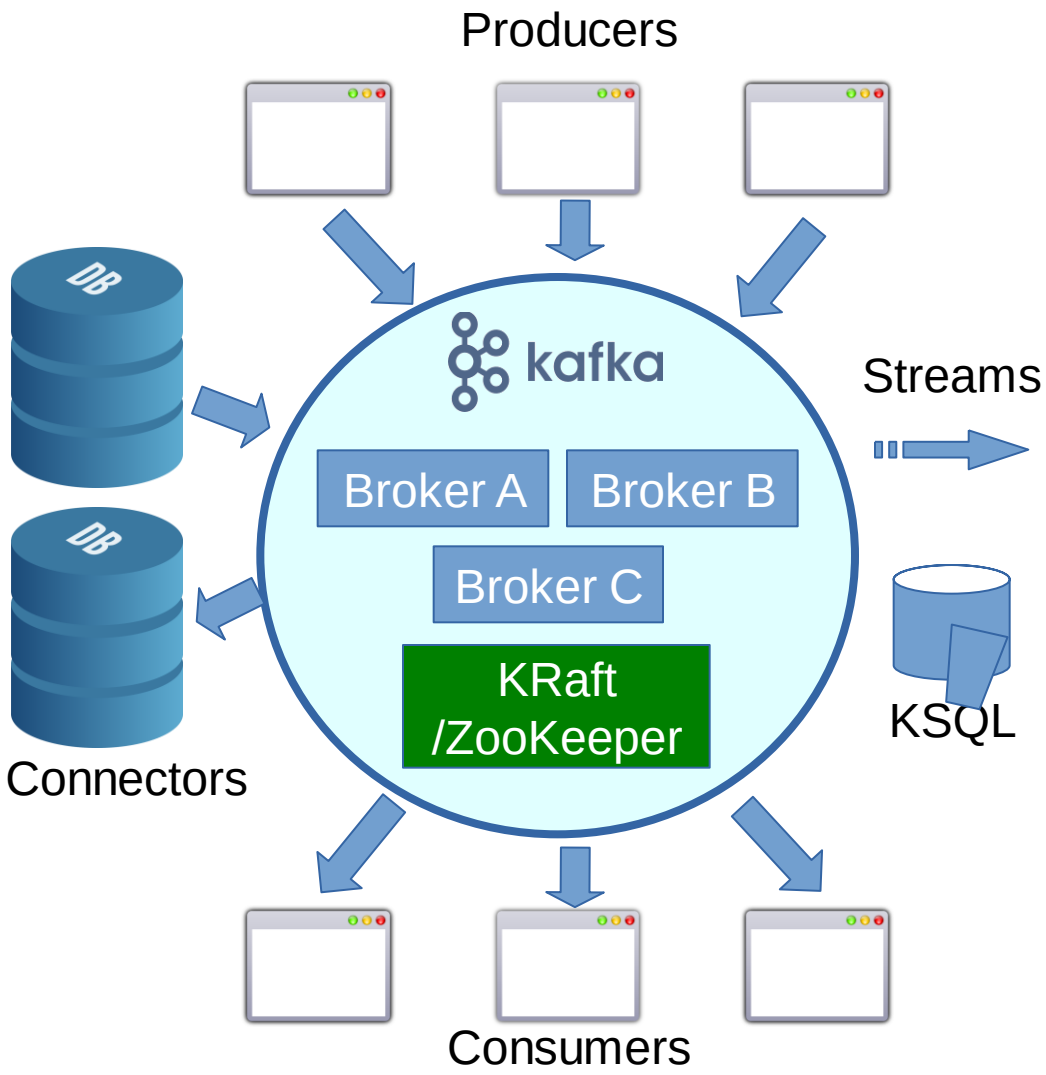
Kafka Main Concepts

- Kafka is run as a cluster on one or more servers (brokers) that can span multiple datacenters.
- The Kafka cluster stores streams of records in categories called topics.
- Each record consists of a key, value, and timestamp.



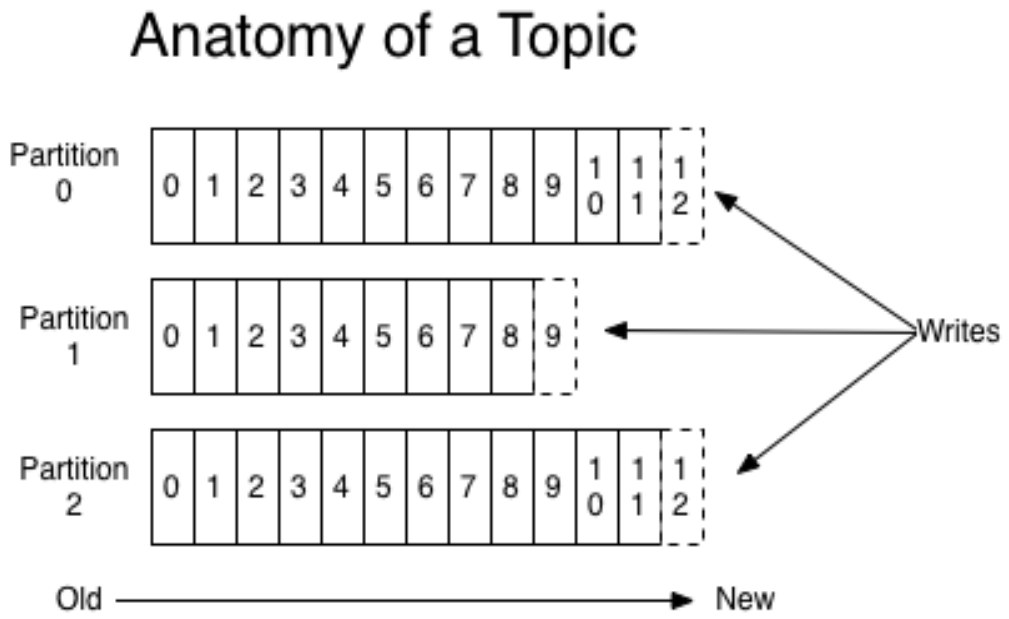
Kafka Core APIs

- The Producer API - publish a stream of records to one or more Kafka topics.
- The Consumer API - subscribe to one or more topics and process the stream of records produced to them.
- The Streams API - a stream processor, consuming an input stream from one or more topics and producing an output stream to one or more output topics, effectively transforming the input streams to output streams.
- The Connector API allows building and running reusable producers or consumers that connect Kafka topics to existing applications or data systems – e.g. connector to a DB might capture every change in a table



Topics and Logs

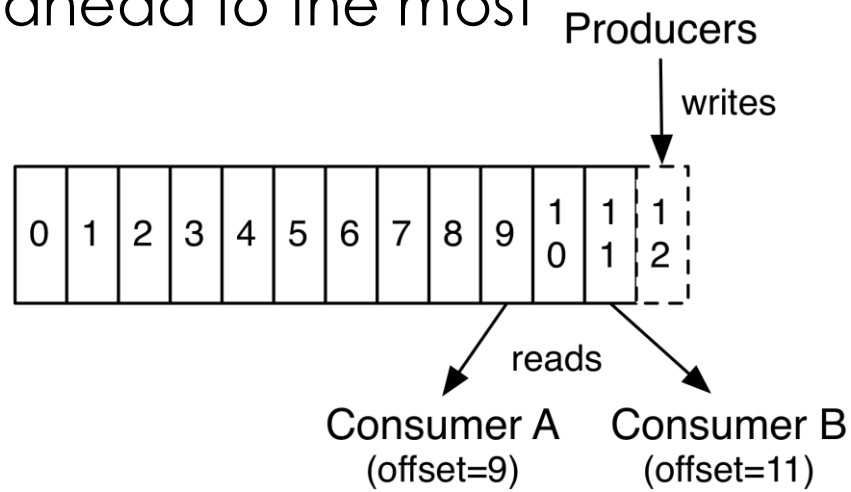
- Topic = stream of records
- A topic is a category or feed name to which records are published. Topics in Kafka are always multi-subscriber; that is, a topic can have zero, one, or many consumers that subscribe to the data
- For each topic, the Kafka cluster maintains a partitioned log --->
- Each partition is an ordered, immutable sequence of records that is continually appended to - a structured commit log
- The records in the partitions are each assigned a sequential id number called the offset that uniquely identifies each record within the partition.



Source: https://github.com/apache/kafka/blob/trunk/docs/images/log_anatomy.png

Consumers Offset and Data Retention

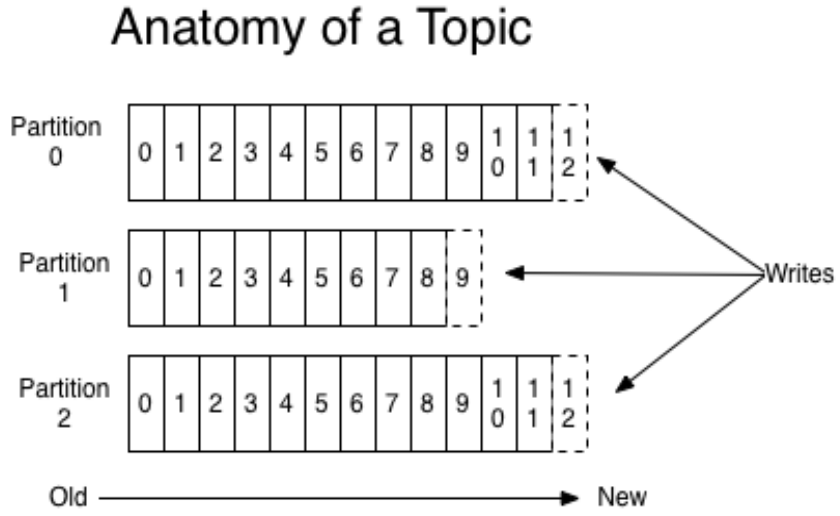
- Kafka cluster durably persists all published records - whether or not they have been consumed, using configurable retention period
- Kafka performance is effectively constant with respect to data size
- Offset or position of that consumer in the log – controlled by the consumer: normally a consumer will advance its offset linearly as it reads records, but, since the position is controlled by the consumer it can consume records in any order. E.g. a consumer can reset to an older offset to reprocess data from the past or skip ahead to the most recent record and start consuming from "now".
- Kafka consumers are very cheap - they can come and go without much impact on the cluster or on other consumers.



Source: https://github.com/apache/kafka/blob/trunk/docs/images/log_consumer.png

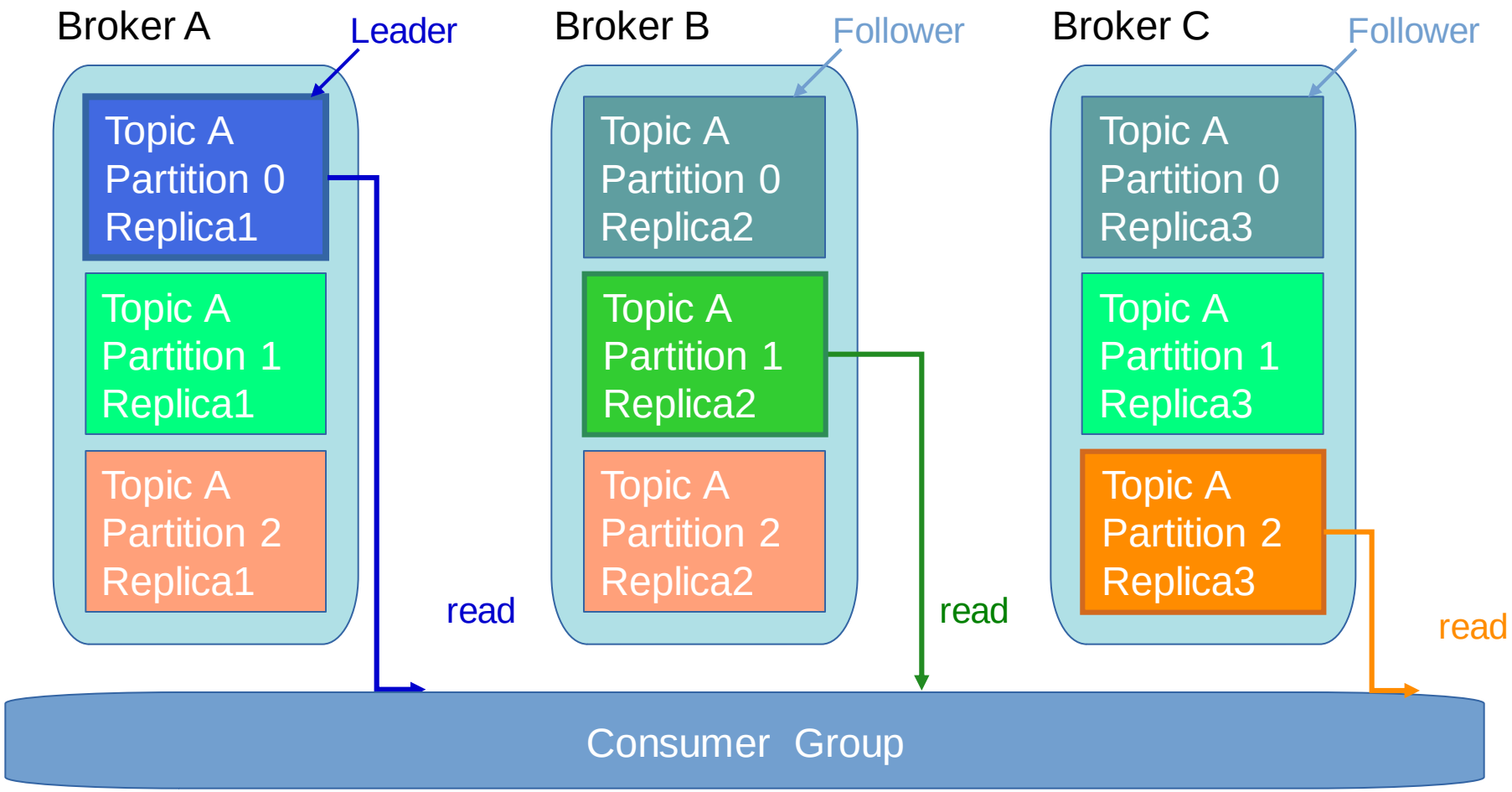
Kafka Partitions and Distribution

- The partitions in the log serve several purposes:
  1. Allow the log to scale beyond a size that will fit on a single server. Each individual partition must fit on the servers that host it, but a topic may have many partitions so it can handle an arbitrary amount of data.
  2. Act as the unit of parallelism—more on that in next slide

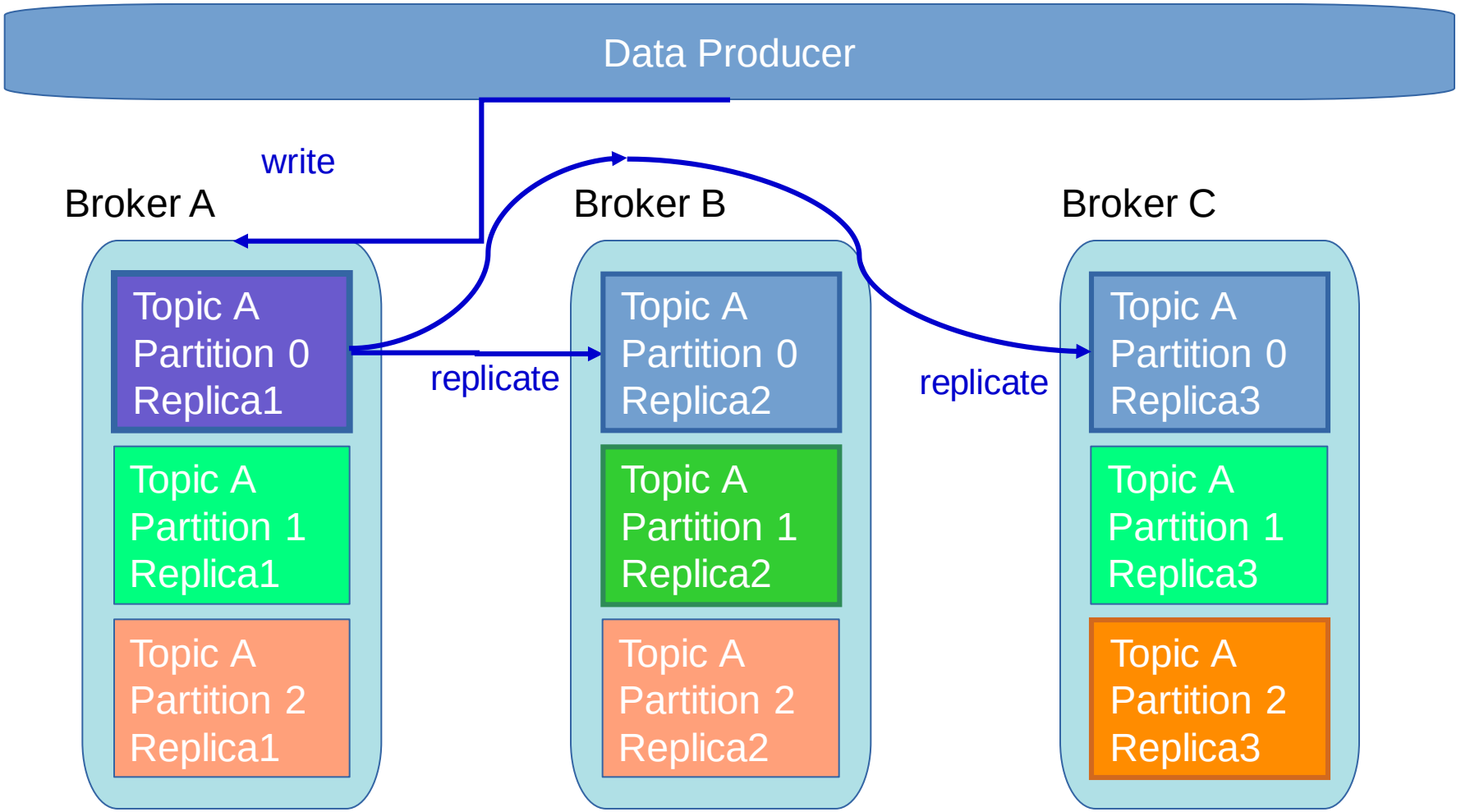


Source: https://github.com/apache/kafka/blob/trunk/docs/images/log_anatomy.png

Distribution of Partitions and Read Balancing

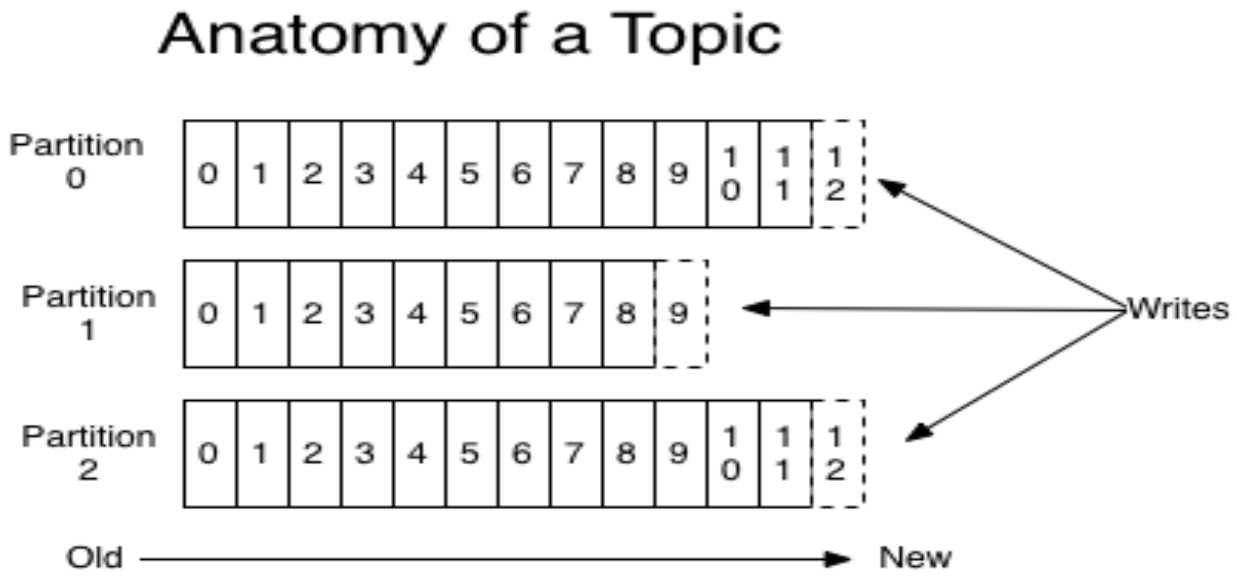


Data Replication



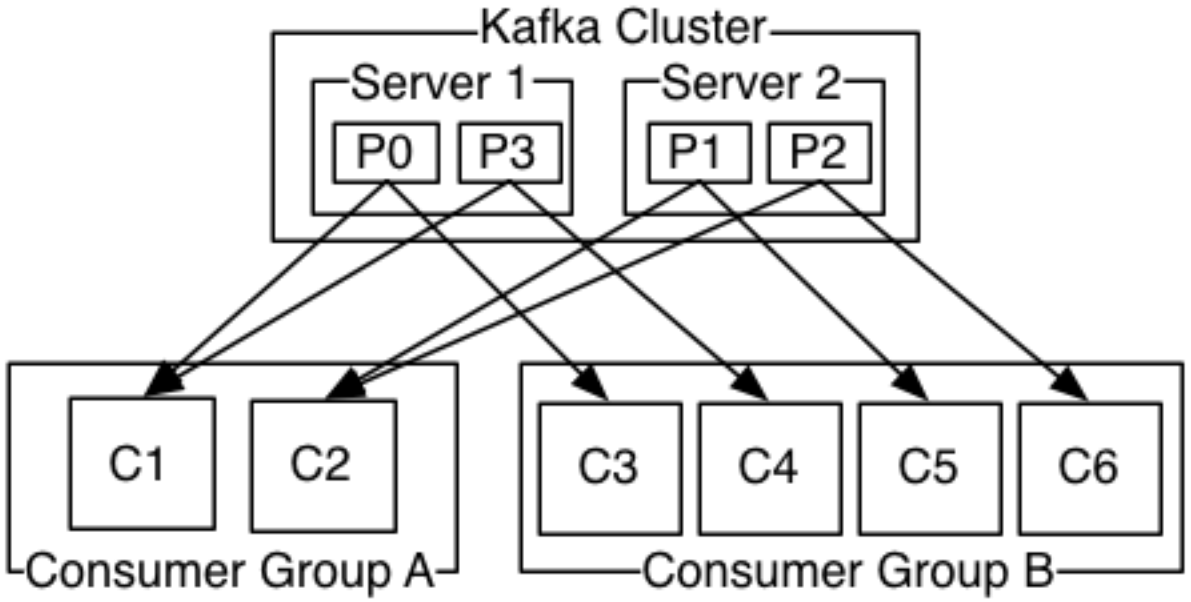
Kafka Producers

Producers publish data to the topics of their choice. The producer is responsible for choosing which record to assign to which partition within the topic. This can be done in a round-robin fashion simply to balance load or it can be done according to some semantic partition function (e.g. based key's hash value)

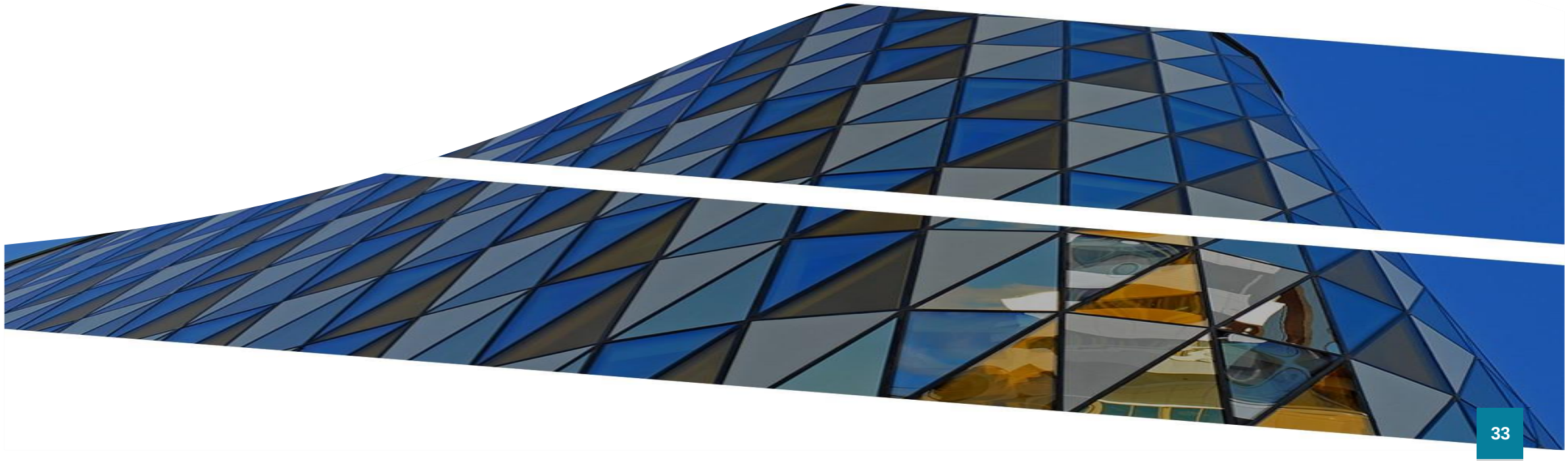


Kafka Consumers

- Consumers label themselves with a consumer group, and each record published to a topic is delivered to one consumer instance within each consumer group. Can be separate processes/machines
- Same consumer group --> records will effectively be load balanced
- Different consumer groups, --> broadcast to all the consumers



Kafka Streams API



Kafka Streams

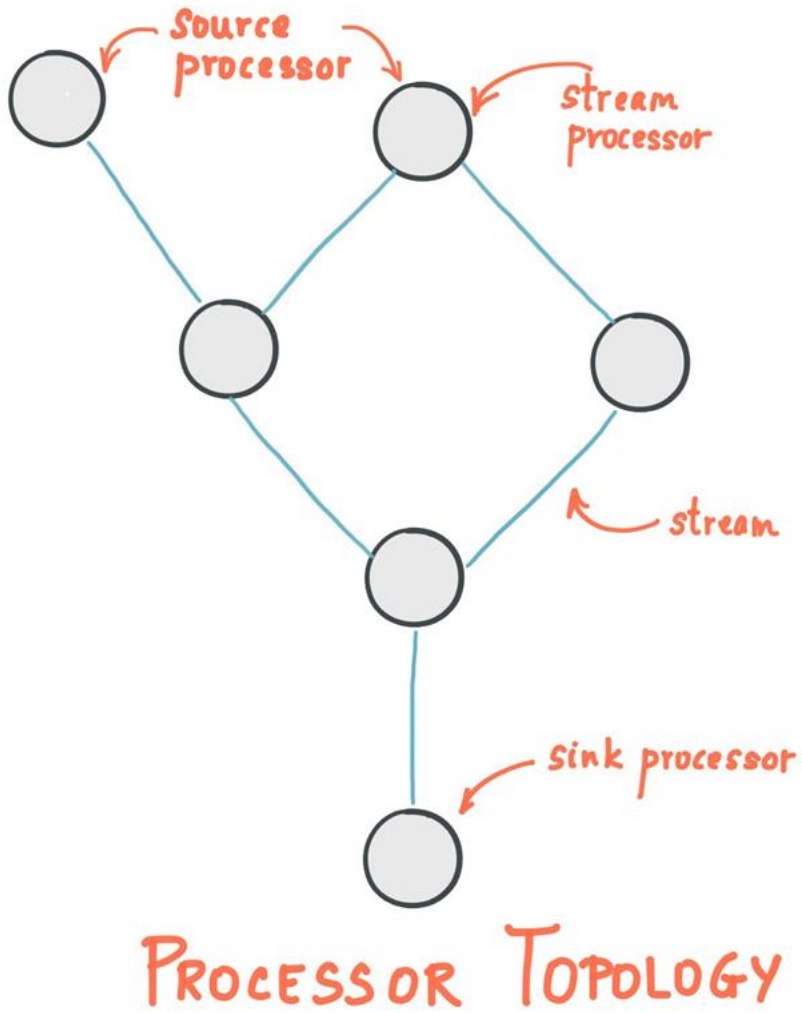
- By combining storage and low-latency subscriptions, streaming applications can treat both past and future data the same way. That is a single application can process historical, stored data but rather than ending when it reaches the last record it can keep processing as future data arrives. This is a generalized notion of stream processing that subsumes batch processing as well as message-driven applications ==> Kappa architecture
- Likewise for streaming data pipelines the combination of subscription to real-time events make it possible to use Kafka for very low-latency pipelines; but the ability to store data reliably make it possible to use it for critical data where the delivery of data must be guaranteed or for integration with offline systems that load data only periodically or may go down for extended periods of time for maintenance. The stream processing facilities make it possible to transform data as it arrives.

Why you'll love using Kafka Streams?

- Elastic, highly scalable, fault-tolerant
- Deploy to containers, VMs, bare metal, cloud
- Equally viable for small, medium, & large use cases
- Fully integrated with Kafka security
- Write standard Java and Scala applications
- Exactly-once processing semantics
- No separate processing cluster required
- Develop on Mac, Linux, Windows

Source: *https://kafka.apache.org/documentation/streams/*

Kafka Stream Processing - DAG



Source: *https://kafka.apache.org/documentation/streams/*

Demo

Available @ Github:

https://github.com/iproduct/ipt-smart-irrigation

Demo Main Features

1. **Real-time Sensor Data**: Displays live data from flowmeters and moisture sensors, including calculated water volume.
2. **Irrigation Control**: Allows individual control of irrigation valves through a WebSocket connection.
3. **Zone Management**: A dedicated page for CRUD (Create, Read, Update, Delete) operations on irrigation zones, interacting with a backend REST API.
4. **Historical Data Visualization**: Uses Echarts to display historical flow volume and moisture sensor data in separate, resizable charts.
5. **Responsive Design**: The dashboard is designed to adapt to various screen sizes.

Smart Irrigation

Zone: Front Yard (Valve 0)

Current Readings

Flow Volume 1: 3.26 L / 20 L

Moisture 1: 72

Moisture 2: 56

Irrigation Control

Valves Status

Valve 0: Closed

OPEN VALVE 0 | CLOSE VALVE 0

Zone: Tomatos Garden (Valve 1)

Current Readings

Flow Volume 2: 13.41 L / 30 L

Moisture 3: 71

Moisture 4: 74

Irrigation Control

Valves Status

Valve 1: Closed

OPEN VALVE 1 | CLOSE VALVE 1

Zone: Balcony Pots (Valve 2)

Current Readings

Flow Volume 3: 0.50 L / 15 L

Moisture 5: 92

Moisture 6: 88

Irrigation Control

Valves Status

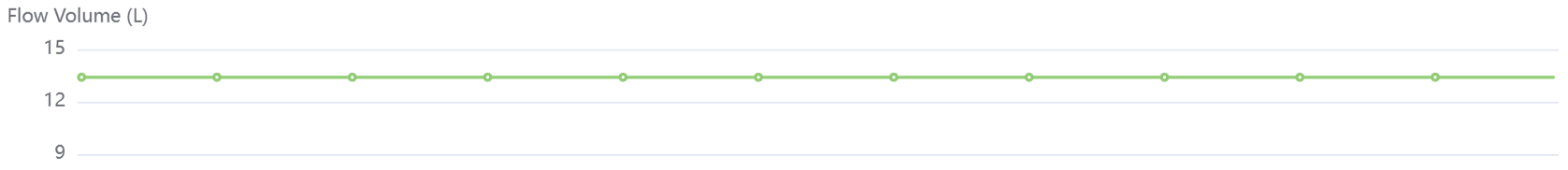
Valve 2: Closed

OPEN VALVE 2 | CLOSE VALVE 2

Charts & Analytics

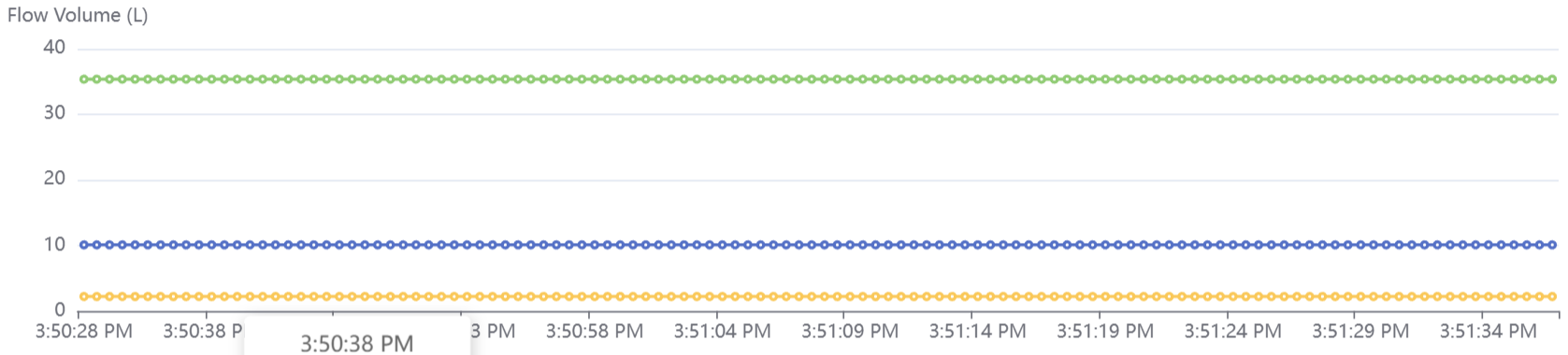
Data Visualization

Flow Meter Data

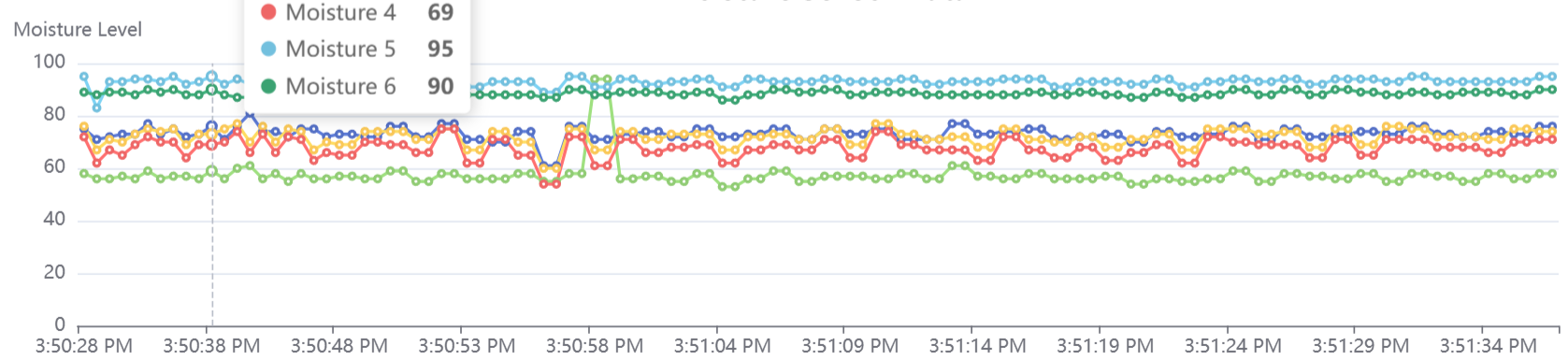


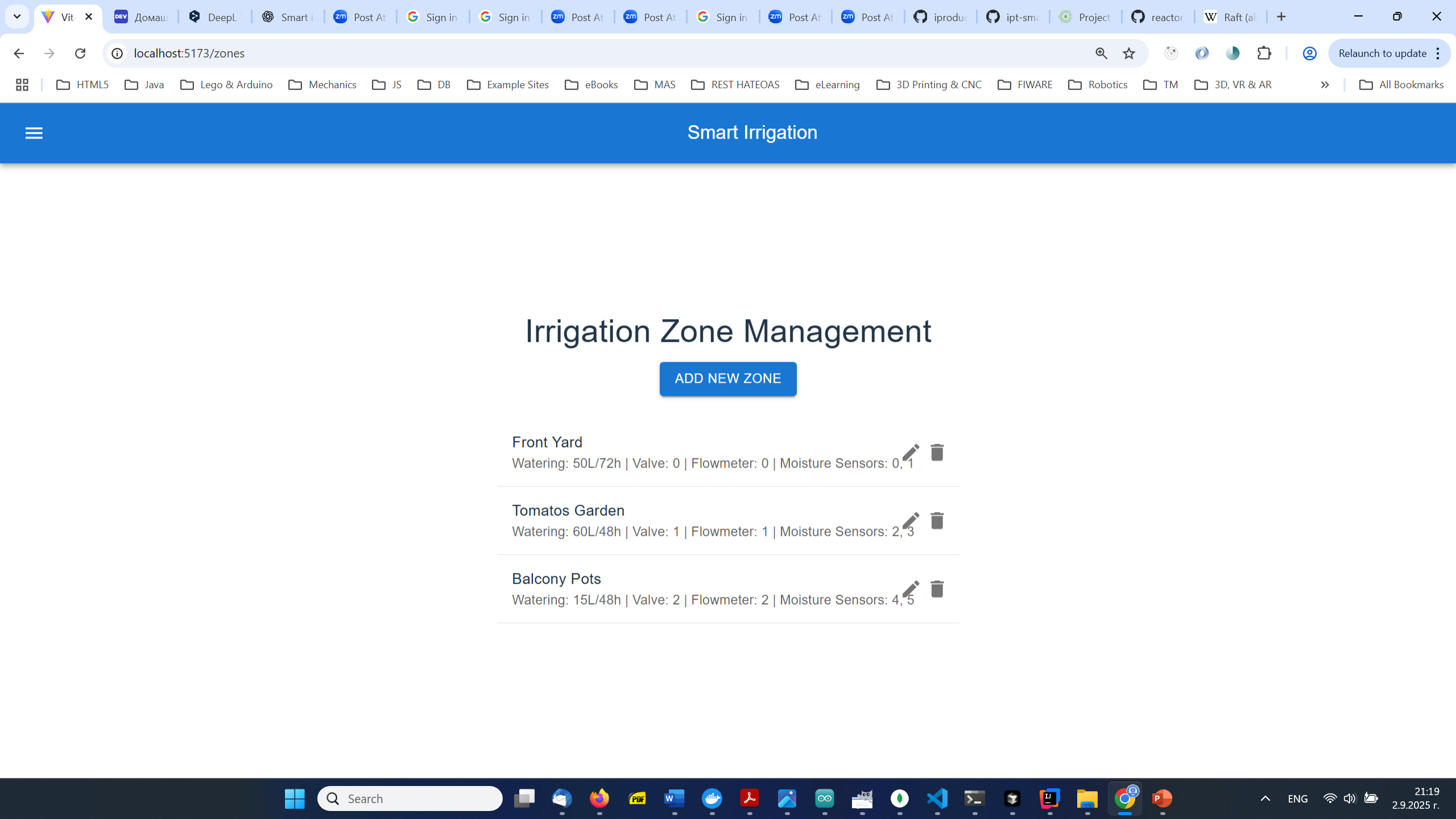
Smart Irrigation

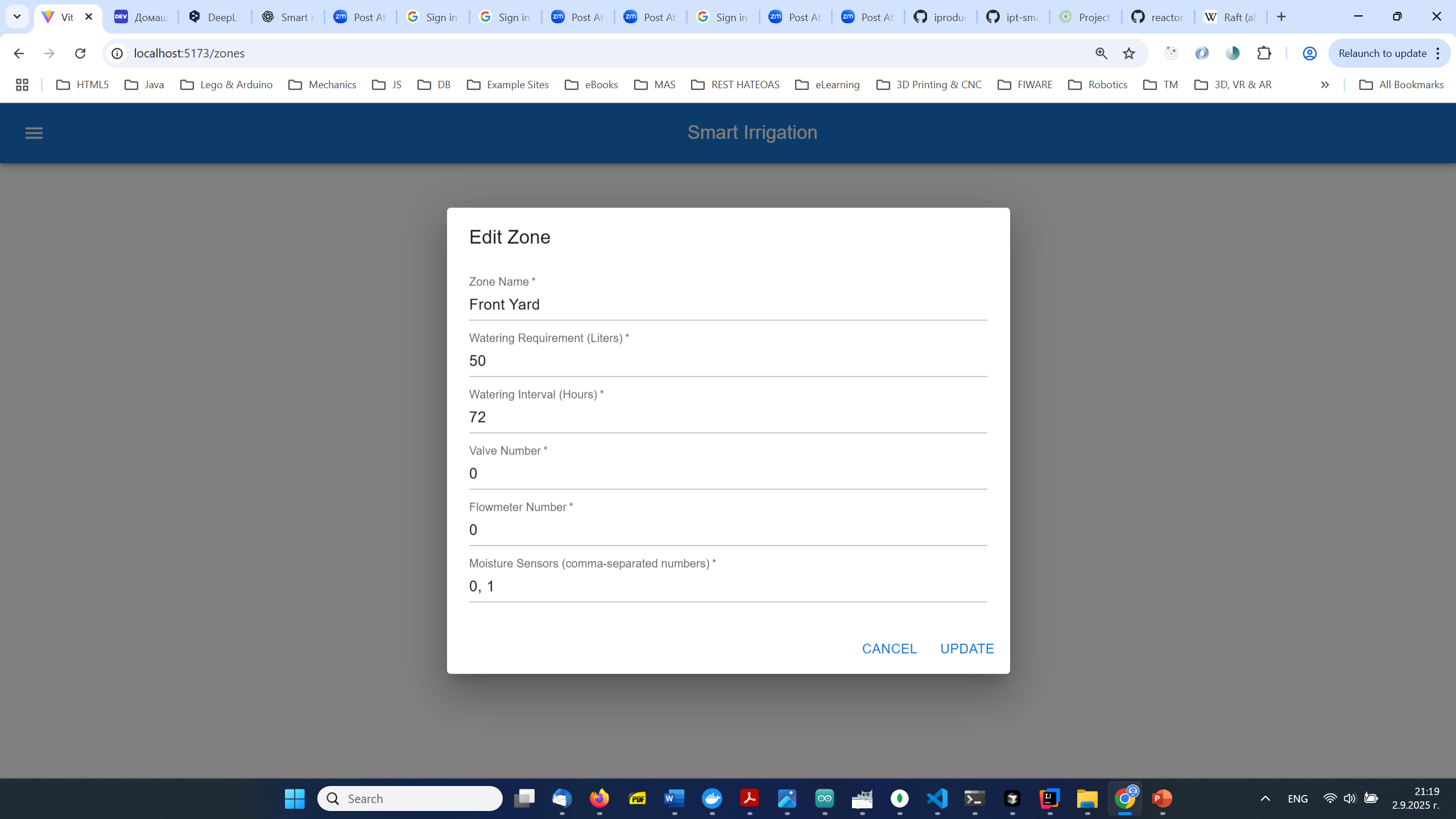
Flow Meter Data



Moisture Sensor Data




localhost:5173/zones
Relaunch to update
HTML5
Java
Lego & Arduino
Mechanics
JS
DB
Example Sites
eBooks
MAS
REST HATEOAS
eLearning
3D Printing & CNC
FIWARE
Robotics
TM
3D, VR & AR
All Bookmarks
Smart Irrigation
Irrigation Zone Management
ADD NEW ZONE
Front Yard
Watering: 50L/72h | Valve: 0 | Flowmeter: 0 | Moisture Sensors: 0, 1
Tomatos Garden
Watering: 60L/48h | Valve: 1 | Flowmeter: 1 | Moisture Sensors: 2, 3
Balcony Pots
Watering: 15L/48h | Valve: 2 | Flowmeter: 2 | Moisture Sensors: 4, 5
Search
ENG
21:19
2.9.2025 г.


localhost:5173/zones
Relaunch to update
HTML5
Java
Lego & Arduino
Mechanics
JS
DB
Example Sites
eBooks
MAS
REST HATEOAS
eLearning
3D Printing & CNC
FIWARE
Robotics
TM
3D, VR & AR
All Bookmarks
Smart Irrigation
Edit Zone
Zone Name *
Front Yard
Watering Requirement (Liters) *
50
Watering Interval (Hours) *
72
Valve Number *
0
Flowmeter Number *
0
Moisture Sensors (comma-separated numbers) *
0, 1
CANCEL
UPDATE
Search
ENG
21:19
2.9.2025 г.

Frontend Technologies

- React
- TypeScript
- Material-UI
- Echarts & echarts-for-react
- React Router
- WebSockets (for real-time data)

Backend Technologies

- **Reactive Web Services**: RESTful APIs for managing irrigation zones.
- **CoAP Server**: Communication with IoT irrigation controllers for sensor data and commands.
- **Apache Kafka Integration**: Real-time processing of sensor data using Kafka Streams and publishing of commands.
- **Reactive MongoDB Persistence**: Storage and retrieval of irrigation zone configurations.
- **WebSockets**: Real-time sensor data visualization in the frontend.

Thank's for Your Attention!

Trayan Iliev

IPT – Intellectual Products & Technologies

http://iproduct.org/

https://github.com/iproduct

https://twitter.com/trayaniliev

https://www.facebook.com/IPT.EACAD

https://www.linkedin.com/in/trayaniliev/

https://www.linkedin.com/company/ipt-iproduct