IoT-Messaging & Datapipeline Implementering

Med fokus på MQTT (Message Queuing Telemetry Transport)

Navn: Tobias Andersen
Arbejde: Netværkstekniker
Firma: Københavns Universitet

Projektetoplæg

AgriTech Farming

Forretningsmæssigt behov

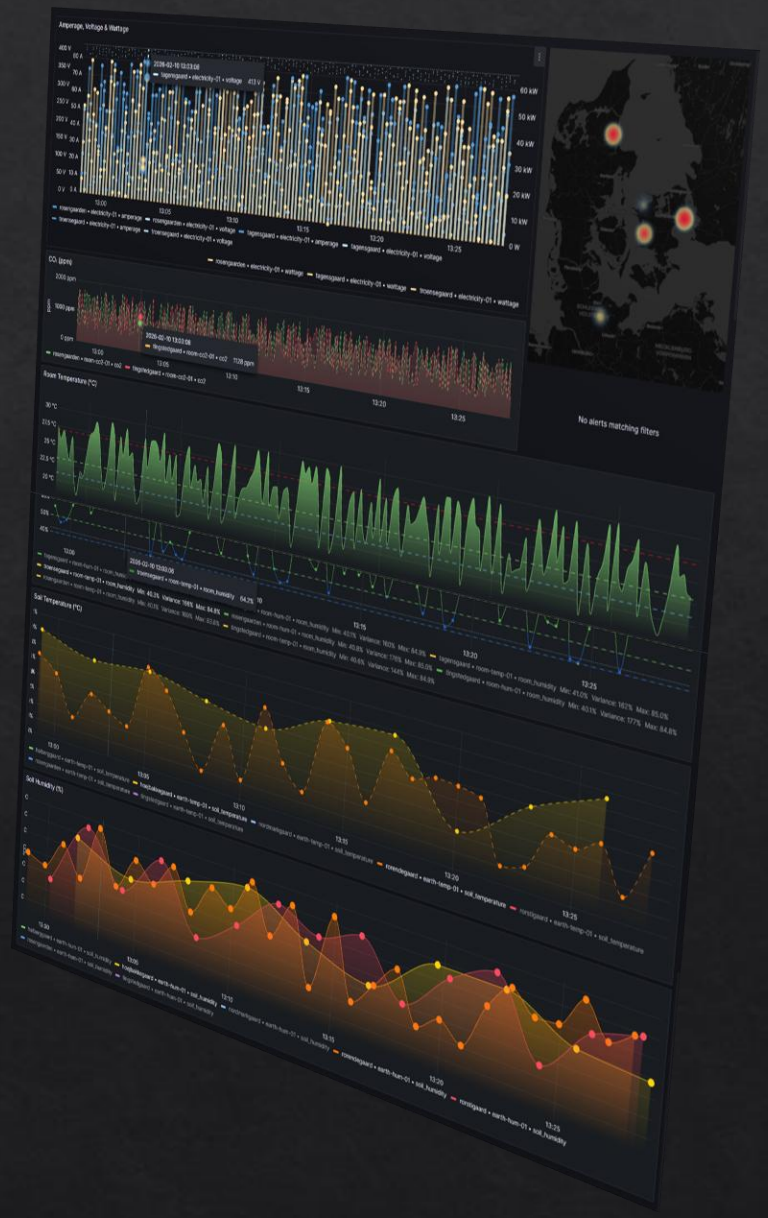
- Kontinuerlig indsamling af sensor- og maskindata fra landbrugsarealer
- Overførsel af billeddata fra droner
- Centraliseret overvågning og analyse af driftsdata

Teknisk målsætning

- Robust og skalerbar IoT-løsning baseret på eventdrevet arkitektur
- Løs kobling mellem edge-enheder og backend-systemer
- Fokus på drift, sikkerhed og udvidelsesmuligheder

Løsningsstrategi

- PoC er ikke kun en demo, men designet som en produktionsklar reference-implementering
- Arkitekturen kan skaleres fra PoC til produktion uden redesign – kun kapacitetsudvidelse



Løsningsdesign

Jf. Opstillede FM og FR/NFR

Principper
- Deterministisk
- Skalerbar
- Tenant-isoleret

Funktionelt
- Automatiseret dataindsamling
- Central administration
- Konfigurationsdrevet onboarding

Skalerbarhed
- Tenants, Farme, Lokationer
- Enheder, Sensorer, Sensortyper
- Policies

Sikkerhed
- TLS
- Default-deny policy model
- Statisk attributbaseret (ABAC) policy-evaluering

Domæne Model
- Tenant
- Farm
- Location
- Device
- Device class
- Message class
- Policies

FR: Automatiseret dataindsamling & central styring
NFR: Sikkerhed (TLS + default-deny)
NFR: Skalerbarhed (konfigurationsdrevet onboarding)

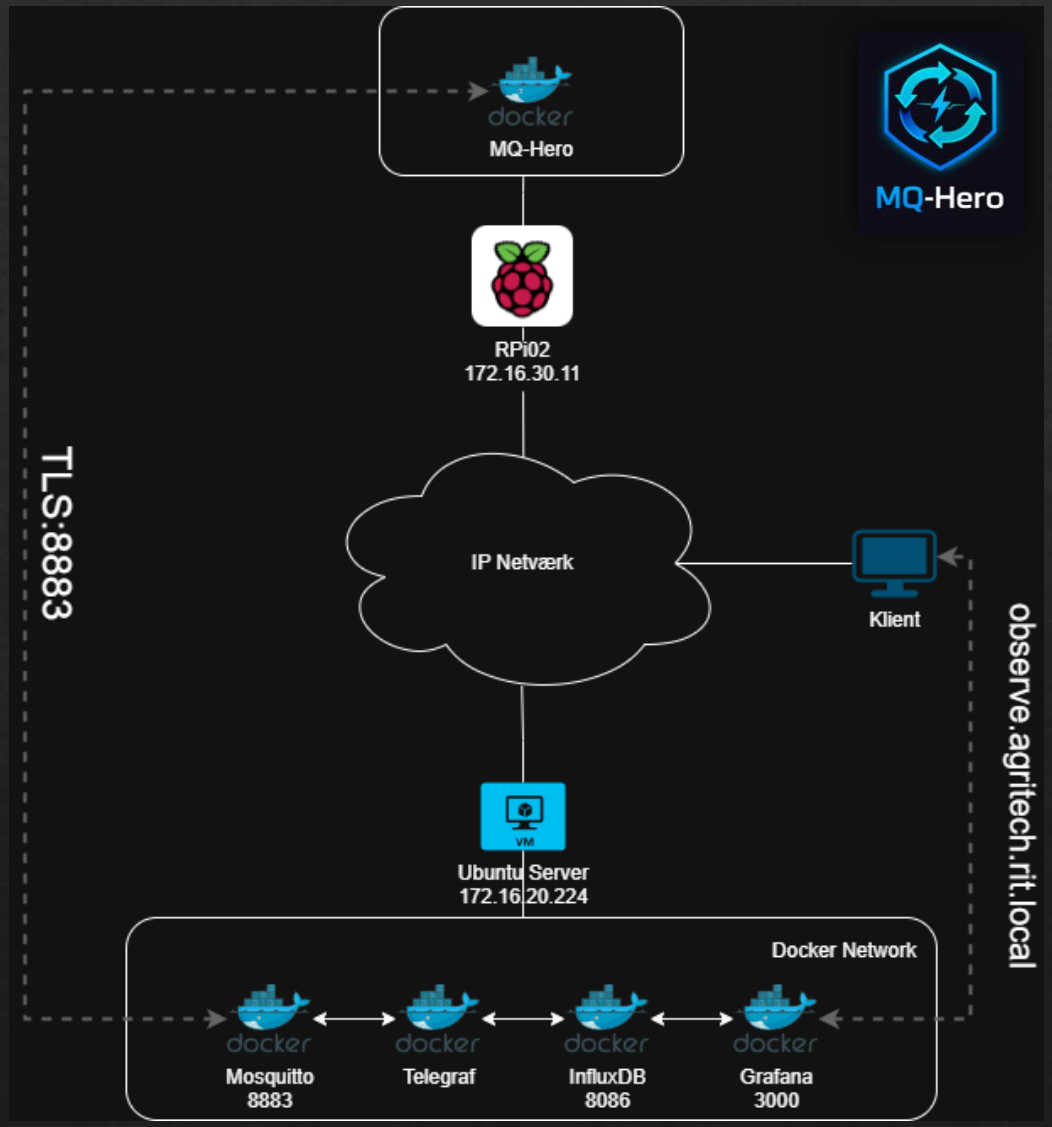
Designet er direkte afledt af opstillede FR og NFR (sikkerhed, skalerbarhed, lave driftsomkostninger).

Systemets Arkitektur

Løst Koblet og Eventdrevet Datapipeline

I PoC'en er der deployet 29 sensorer fordelt på 8 sensortyper og 9 gårde med 10 underlokationer. Det demonstrerer, at topologi, policy-model og MQTT-routing fungerer deterministisk på tværs af flere sites uden ændringer i arkitekturen.

- 🟩 Hardware Laget (IoT-Edge/RPi)
- ↓
- 🟩 Runtime Laget (Docker Containerization)
- ↓
- 🟥 Sensor Laget (MQ-Hero/Python)
- ↓
- 🟧 MQTT Klient Laget (paho-mqtt)
- ↓
- 🟥 Protokol Laget (MQTT)
- ↓
- 🟩 Broker Laget (MQTT Broker)
- ↓
- 🟩 Ingest / Transformations Laget (Telegraf)
- ↓
- 🟩 Storage Laget (InfluxDB)
- ↓
- 🟩 Observability & Analytics Laget (Grafana)
- ↓
- 🟩 Presentations Laget (Browser)

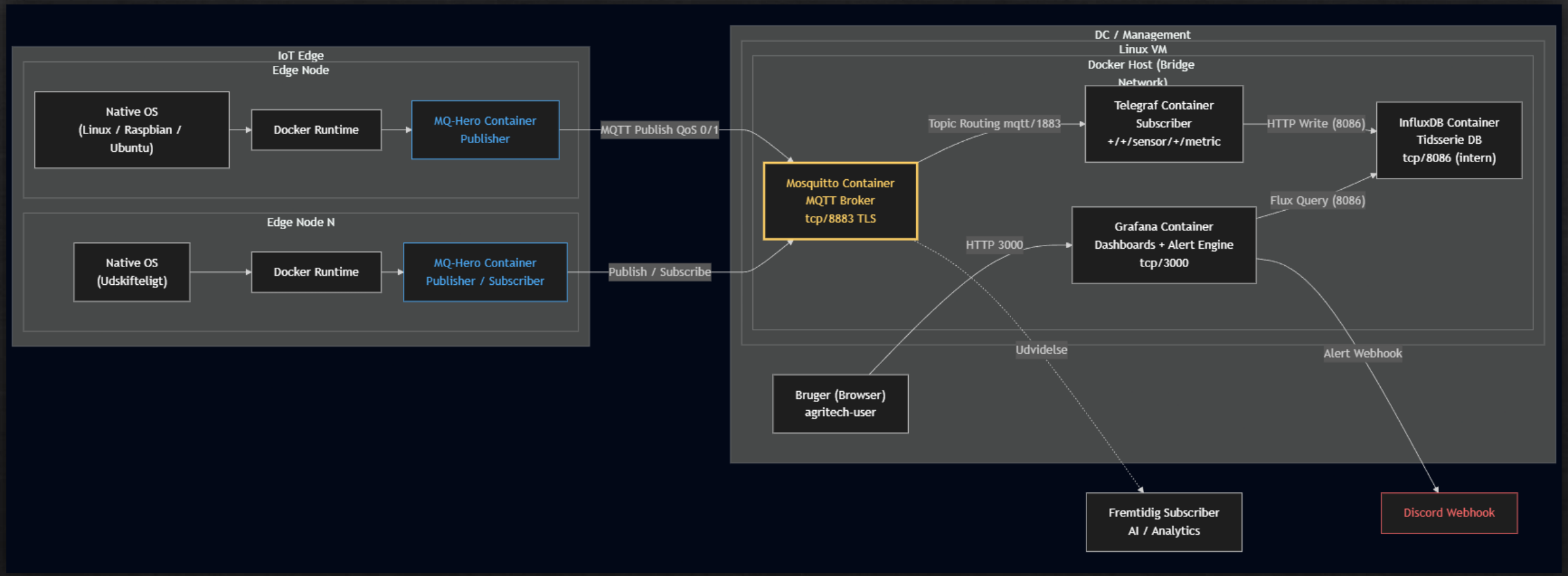


Næste lag ↓
Ikke udskifteligt 🟥
Kan udskiftes 🟧
Nemt udskifteligt 🟩

Løsningen kan flyttes fra PoC til produktion uden arkitekturændringer – kun skalering.

Systemets Helhed

En Hurtig Gennemgang



Gennemgang af Systemet

Systemet gennemgås lag for lag

Edge → Runtime → App → Client → Protocol → Broker → Ingest → Storage → Observability → Presentation

- Hardware Laget (IoT-Edge/Rpi)
- ↓
- Runtime Laget (Docker Containerization)
- ↓
- Sensor Laget (MQ-Hero/Python)
- ↓
- MQTT Klient Laget (paho-mqtt)
- ↓
- Protokol Laget (MQTT)
- ↓
- Broker Laget (MQTT Broker)
- ↓
- Ingest / Transformations Laget (Telegraf)
- ↓
- Storage Laget (InfluxDB)
- ↓
- Observability & Analytics Laget (Grafana)
- ↓
- Presentations Laget (Browser)

Løsningen gennemgås lag for lag for systemet.

Næste emne ↓

Emne som er behandlet

Det emne vi tager fat i

Emne som ikke er behandlet

Hardware Laget (Edge)

Platform
- Raspberry Pi 4 (dedikeret IoT edge-enhed)
- Headless deployment (ingen GUI)
- Linux (Raspian OS)
- Lavt hardware-mæssigt fodaftryk
- Statisk IP via Netplan

Runtime isolation
- Containerization
- Services m.v. isoleret fra OS

Overvågning
- SNMPv2+3 konfigureret
- Bredt OID træ
  - Usikkert, men fint til PoC
- Kun tilladte management hosts

Firewall
- Ingen åbne services mod internettet
- Containere isoleret
- Host-baseret firewall

Netplan Configuration

```yaml
network:
  version: 2
  ethernets:
    eth0:
      dhcp4: false
      addresses:
        - 172.16.30.11/24
      routes:
       - to:  172.16.0.0/12
         via: 172.16.20.1
      nameservers:
        addresses:
          - 172.16.20.2
          - 172.16.20.3
      optional: false
  wifis:
    wlan0:
      dhcp4: false
      addresses:
        - 192.168.0.201/24
      routes:
        - to:  192.168.0.0/24
          via: 192.168.0.1
      nameservers:
        addresses: [1.1.1.1, 8.8.8.8]
      access-points:
        "mit-ssid":
          password: "***"
      optional: true
      regulatory-domain: "DK"
```

ufw opsætning

```
sudo apt update && sudo apt upgrade -y && sudo apt install ufw -y

sudo ufw --force reset
sudo ufw default deny incoming
sudo ufw default allow outgoing

sudo ufw allow from 172.16.0.0/12 to any port 22 proto tcp
sudo ufw allow from 172.16.0.0/12 to any port 8883 proto tcp
sudo ufw allow from 172.16.0.0/12 to any port 3000 proto tcp
sudo ufw allow from 127.0.0.1 to any port 8086 proto tcp
sudo ufw deny 8086
systemctl enable ufw
systemctl start ufw
sudo ufw enable
```

snmpd.sh

```
SYS_NAME="ftp01"
SYS_LOCATION="ZBC-DC"
sudo apt update
sudo apt upgrade -y
sudo apt install ufw snmp snmpd
sudo truncate -s 0 /etc/snmp/snmpd.conf


sudo tee /etc/snmp/snmpd.conf >/dev/null <<EOF
agentAddress udp:161
sysServices 72
rocommunity RITtest 127.0.0.1/32
rocommunity RITadmin 172.16.20.7/32
sysLocation ${SYS_LOCATION}
sysContact patr121@zbc.dk,tobi801j@zbc.dk
sysName ${SYS_NAME}-agritech
# Specify proper OIDs...
view altingv2c included .1
view altingv3 included .1
access notConfigGroup "" v2c noauth exact altingv2c none none
access notConfigGroup "" usm authPriv exact altingv3 none none
EOF

if ! sudo grep -q 'snmpacc' /var/lib/snmp/snmpd.conf; then
  sudo net-snmp-create-v3-user -ro \
    -a SHA -A 'KOde12345!!?' \
    -x AES -X 'KOde12345!!?' \
    snmpacc
fi

sudo ufw allow 22
sudo ufw allow from 172.16.20.0/24 to any port 161 proto udp

sudo systemctl enable snmpd
sudo systemctl restart snmpd

sudo ufw --force enable
sudo ufw reload
```

Runtime Laget (Containeriseret Miljø)

Isolation

- Applikationen kører isoleret fra host-OS
- Reduceret angrebsflade

Reproducerbarhed

- Deterministisk dependency installation
- Ens runtime i PoC og produktion
- Immutable container image

Drift & Skalering

- Kan deployes på Linux/Windows hosts
- Kan skaleres horisontalt

Host- og Docker-init

```
sudo apt update && sudo apt upgrade -y && sudo apt install git -y
curl -fsSL https://get.docker.com | sudo sh
sudo usermod -aG docker $USER
sudo shutdown -r now

git clone https://github.com/blue-hexagon/mq_hero && cd ./mq_hero
docker build -t mqhero:latest .
docker run -d --name myapp --restart unless-stopped mqhero:latest
```

Dockerfile for MQ-Hero IoT-Edge

```
FROM python:3.14-slim

# Prevent Python from writing .pyc files
ENV PYTHONDONTWRITEBYTECODE=1
ENV PYTHONUNBUFFERED=1

WORKDIR /app

RUN apt-get update && apt-get install -y \
    build-essential \
    && rm -rf /var/lib/apt/lists/*

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY src/ src/
COPY main.py .
COPY .env .
COPY ops/observability/mosquitto/config/certs/ca.crt .

CMD ["python", "main.py"]
```

Runtime Laget (Containeriseret Miljø)

Containeriseret Observability Stack

- Isoleret netværk (iot_net, bridge network)
- Broker, Ingestion og Storage adskilt i separate services
- Persistent volumes til data- og logbevaring
- RO mounts fra filsystemet (Git)
- TLS-beskyttet MQTT-trafik (Edge-DC)

Observability-Stack Docker Compose File (Pt. 1/2)

```yaml
version: "3.8"

networks:
  iot_net:
    driver: bridge

services:
  mosquitto:
    image: eclipse-mosquitto:2.0
    container_name: mosquitto
    ports:
      - "1883:1883"
    volumes:
      - ./mosquitto/mosquitto.conf:/mosquitto/config/mosquitto.conf:ro
      - ./mosquitto/config/passwords:/mosquitto/config/passwords:ro
      - ./mosquitto/config/acl:/mosquitto/config/acl:ro
      - mosquitto_data:/mosquitto/data
      - mosquitto_logs:/mosquitto/log
    networks:
      - iot_net

  influxdb:
    image: influxdb:2.7
    container_name: influxdb
    ports:
      - "8086:8086"
    volumes:
      - influxdb_data:/var/lib/influxdb2
    environment:
      DOCKER_INFLUXDB_INIT_MODE: setup
      DOCKER_INFLUXDB_INIT_USERNAME: admin
      DOCKER_INFLUXDB_INIT_PASSWORD: admin123
      DOCKER_INFLUXDB_INIT_ORG: iot
      DOCKER_INFLUXDB_INIT_BUCKET: mqtt_metrics
      DOCKER_INFLUXDB_INIT_ADMIN_TOKEN: Sv3ndeT02qkdencauCucumfbEr
    networks:
      - iot_net
```

Observability-Stack Docker Compose File (Pt. 2/2)

```yaml
  telegraf:
    image: telegraf:1.30
    container_name: telegraf
    depends_on:
      - mosquitto
      - influxdb
    volumes:
      - ./telegraf/telegraf.conf:/etc/telegraf/telegraf.conf:ro
    networks:
      - iot_net

  grafana:
    image: grafana/grafana:10.2.3
    container_name: grafana
    ports:
      - "3000:3000"
    depends_on:
      - influxdb
    volumes:
      - grafana_data:/var/lib/grafana
      - ./grafana/provisioning:/etc/grafana/provisioning
      - ./grafana/dashboards:/var/lib/grafana/dashboards
    networks:
      - iot_net

volumes:
  influxdb_data:
  grafana_data:
  mosquitto_data:
  mosquitto_logs:
```

Gennemgang af observability container-miljøet, gennemgåes senere.

Sensor Laget (MQ-Hero)

Domænespecifikt Control-Plane (MQ-Hero DSL) [1/2]

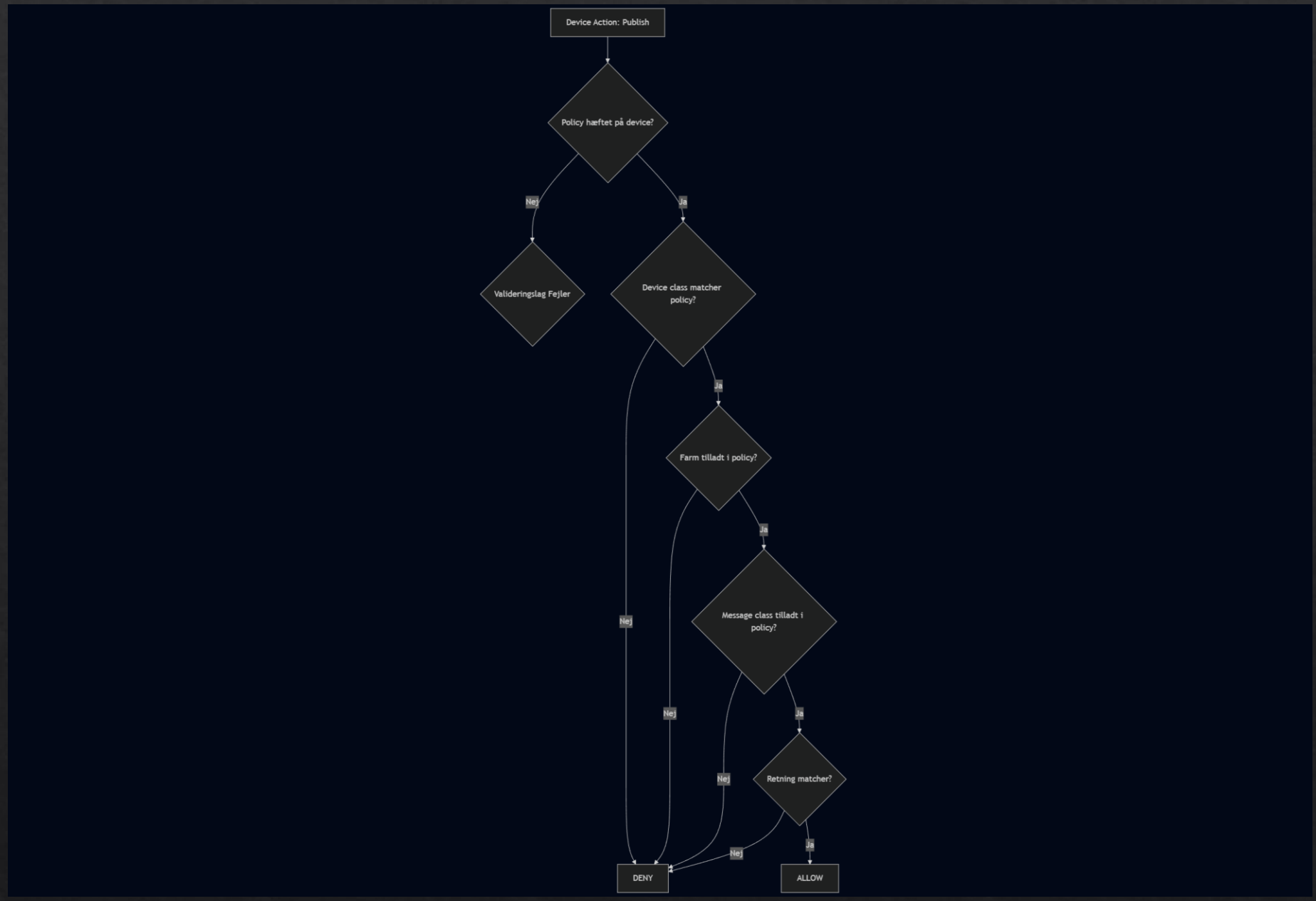
```yaml
schema_version: "2.0"
tenants:
  rit_customer_agritech:
    meta:
      short_name: "agtech"
      full_name: "AgriTech Solutions"
      api_version: 1
      description: "IoT-implementation for AgriTech Solution."
      mqtt:
        - id: "prod"
          username: "simulator"
          password: "KOde12345!!?"
          ipv4: "172.16.20.224"
          port: 8883
          keepalive: 120

    topology:
      farms:
        - id: "rorendegaard"
          name: "Rorendegaard"
          city: "Taastrup"
          devices:
            - id: "earth-hum-01"
              class: "sensor"
              driver: "sen0193"
              type: "soil_humidity"
              location: "rorendegaard.cropfield"
              interval: 60
              policies: [ "sensor-pub" ]
```

Domænespecifikt Control-Plane (MQ-Hero DSL) [2/2]

```yaml
definitions:
  device_classes:
    sensor:
      - id: "soil_humidity"
        driver: "sen0193"
        unit: "%"
        ranges:
          normal: "20-60 %"
          low: "< 20 %"
          high: "> 70 %"
        default_interval: 180

  message_classes:
    - id: metric
      topic: metrics
      qos: 0
      retain: false

  locations:
    - name: "rorendegaard.cropfield"
      latitude: 56.72814
      longitude: 10.11240

policies:
  - name: "sensor-pub"
    farms:
      - rorendegaard
    device_classes: [ "sensor" ]
    message_classes: [ "metric" ]
    direction: PUB
```

Der gives her et bevidst minimalt eksempel med henblik på at forklare designet.

Sensor Laget (MQ-Hero)



Sensor Laget (MQ-Hero)

Domænespecifikt Control-Plane (MQ-Hero DSL) [1/2]

```yaml
schema_version: "2.0"
tenants:
  rit_customer_agritech:
    meta:
      short_name: "agtech"
      full_name: "AgriTech Solutions"
      api_version: 1
      description: "IoT-implementation for AgriTech Solution."
      mqtt:
        - id: "prod"
          username: "simulator"
          password: "KOde12345!!?"
          ipv4: "172.16.20.224"
          port: 8883
          keepalive: 120

    topology:
      farms:
        - id: "rorendegaard"
          name: "Rorendegaard"
          city: "Taastrup"
          devices:
            - id: "earth-hum-01"
              class: "sensor"
              driver: "sen0193"
              type: "soil_humidity"
              location: "rorendegaard.cropfield"
              interval: 60
              policies: [ "sensor-pub" ]
```

Domænespecifikt Control-Plane (MQ-Hero DSL) [2/2]

```yaml
definitions:
  device_classes:
    sensor:
      - id: "soil_humidity"
        driver: "sen0193"
        unit: "%"
        ranges:
          normal: "20-60 %"
          low: "< 20 %"
          high: "> 70 %"
        default_interval: 180

  message_classes:
    - id: metric
      topic: metrics
      qos: 0
      retain: false
```

Sensor Definition

```python
@SensorFactory.register("mh_z19b.py")
class MHZ19B(SensorModel):
    async def read_metrics(self):
        return {
            "co2": random.randint(420, 1800)
        }
```


Fortsat fra sidste slide

MQTT Klient Laget (paho-mqtt)

API-lag mellem MQ-Hero og MQTT-protokollen

- paho-mqtt er Python-implementering af MQTT v3.1.1/v5 klient
- Håndterer CONNECT / SUBSCRIBE / PUBLISH / QoS handshake m.v.
- Abstraherer netværksforbindelsen fra applikationslogik
  - Persistent session
  - Automatisk reconnect med backoff
  - Buffer ved netværksudfald
  - Kontrol af QoS in-flight

paho-mqtt klient implementering

```python
class MqttClient:

    def create(self, broker: MqttBroker, client_id: str) -> None:
        client = mqtt.Client(client_id=client_id, clean_session=False)
        client.reconnect_delay_set(min_delay=1, max_delay=30)

        if broker.mqtt_username:
            client.username_pw_set(broker.mqtt_username, broker.mqtt_password)
        client.tls_set(ca_certs="ca.crt", tls_version=ssl.PROTOCOL_TLSv1_2)
        client.tls_insecure_set(False)

        client.on_connect = self.on_connect
        client.on_disconnect = self.on_disconnect
        client.on_message = self.on_message
        client.max_queued_messages_set(1000)
        client.max_inflight_messages_set(200)

        self.client = client
        self.broker = broker

    def connect(self) -> None:
        if not self.client or not self.broker:
            raise RuntimeError("MQTT client not initialized")

        self.client.connect(self.broker.mqtt_host,
        self.broker.mqtt_port,
        keepalive=self.broker.keepalive)
        self.client.loop_start()

    def publish(self, topic: str, payload: dict, qos: int = 1, retain: bool = False):
        self.client.publish(topic, json.dumps(payload), qos=qos, retain=retain)

    def subscribe(self, topic: str, qos: int = 1):
        self.client.subscribe(topic, qos=qos)

    @staticmethod
    def on_message(client, userdata, msg):
        try:
            data = json.loads(msg.payload.decode())
        except Exception:
            data = msg.payload.decode()
        print(f"[MQTT] {msg.topic}: {data}")
```

Denne kodeblok repræsenterer kildekoden men er skåret ned grundet pladsmangel

Protokol Laget (MQTT)

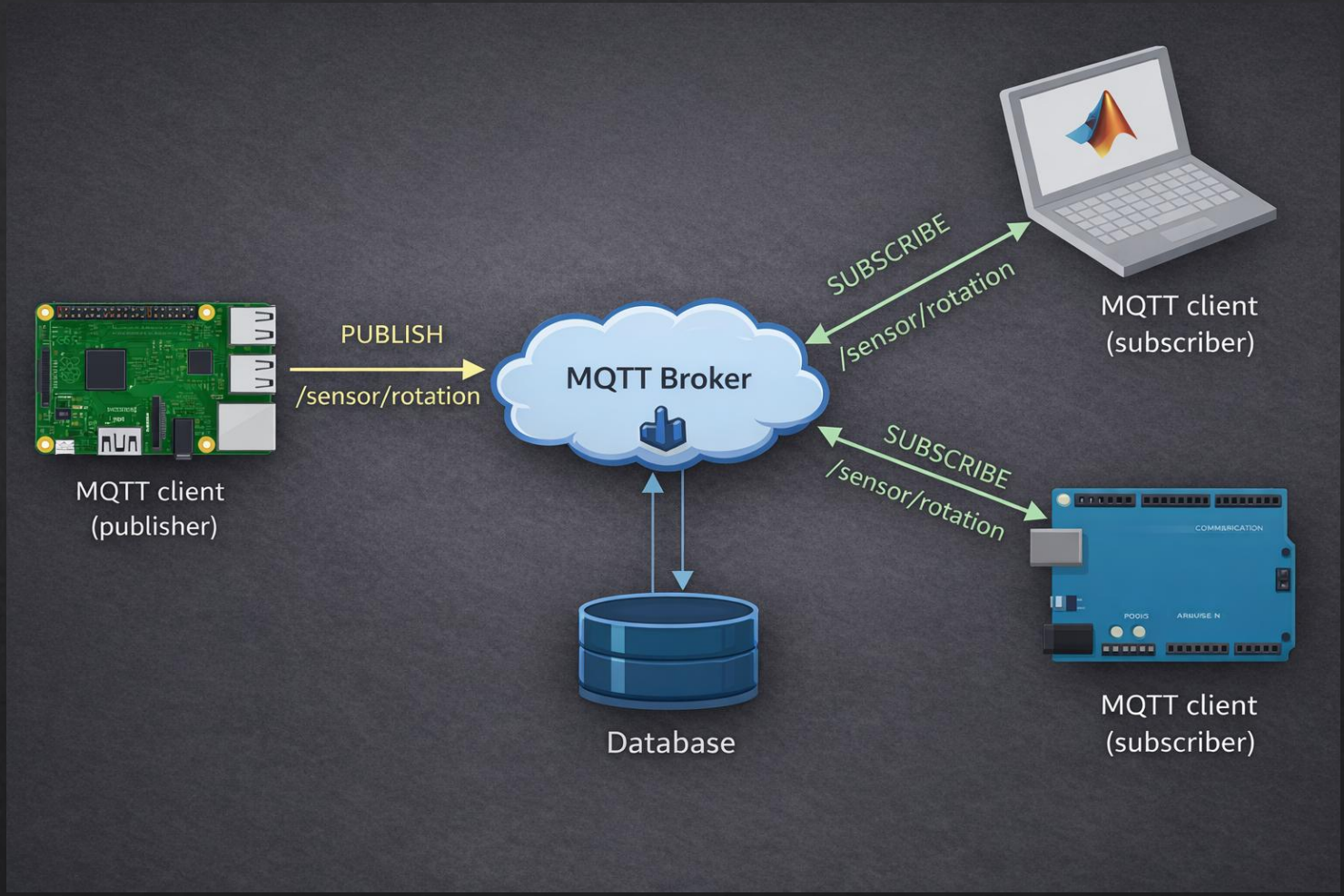
Hvorfor MQTT er valgt som transport protokol?

Arkitektur-valg:
- PUB/SUB – løs kobling mellem systemkomponenter
- Broker-medieret kommunikation
  - Central routing og isolation mellem enheder
- Topic-baseret routing – applikationslag, ikke netværks-lag.
- Broker er central – ingen afhængigheder mellem enheder

Robusthed i feltmiljø:
- Stateful forbindelser mellem klient og broker
- Buffering og QoS
- Link-flapping resistent (felt-udstyr)
- Designet til sporadisk forbindelsestab (offline-tolerant)

Edge-egenskaber:
- Meget lavt overhead
- Egnet til resource-begrænsede enheder

MQTT er velegnet som stabilt transport- og routinglag i en eventdrevet datapipeline.



MQTT fungerer som en stabil kontrakt mellem systemkomponenter.

Protokol Laget (MQTT)

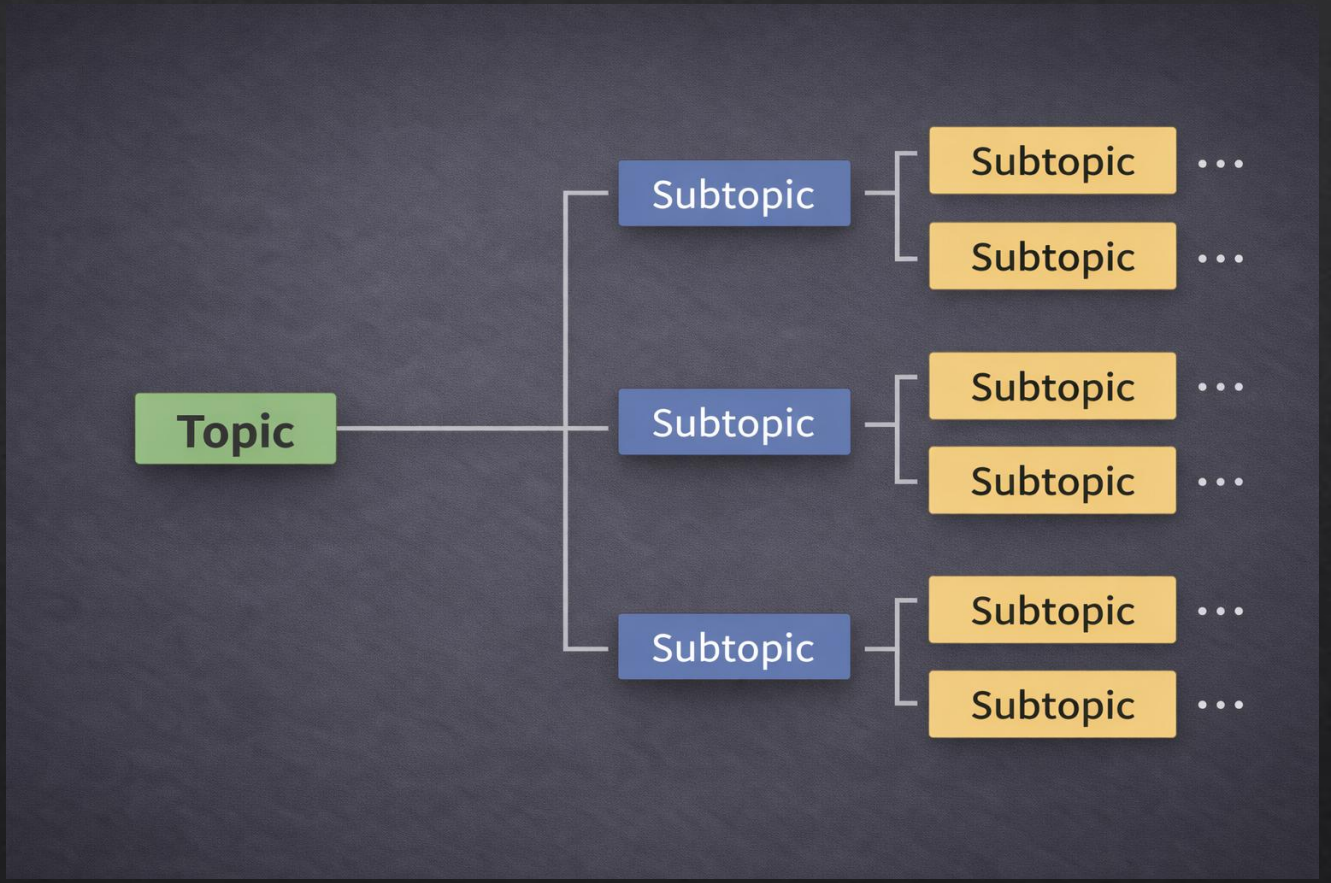
Topics som Routing-grundlag

Hvad er MQTT Topics?

- Applikations-lag routing-mekanisme.
- Hierakisk adresseringsstruktur (tekstbaseret sti) som bruges af:
  - MQTT-klienten til at sende data.
  - MQTT-brokeren til at route beskeder mellem producers og consumers.
- Hver klient kan publicere og/eller abbonere på et eller flere topics.

MQTT Brokerens Rolle

- Hver klient publicerer og/eller abonnerer på én eller flere routing paths (topics) hvor den kan sende data eller modtage data fra fx andre klienter via MQTT-brokeren.
- MQTT-brokeren håndhæver hvilke topics klienter må publicere og/eller abbonere på (RO/WO/RW).



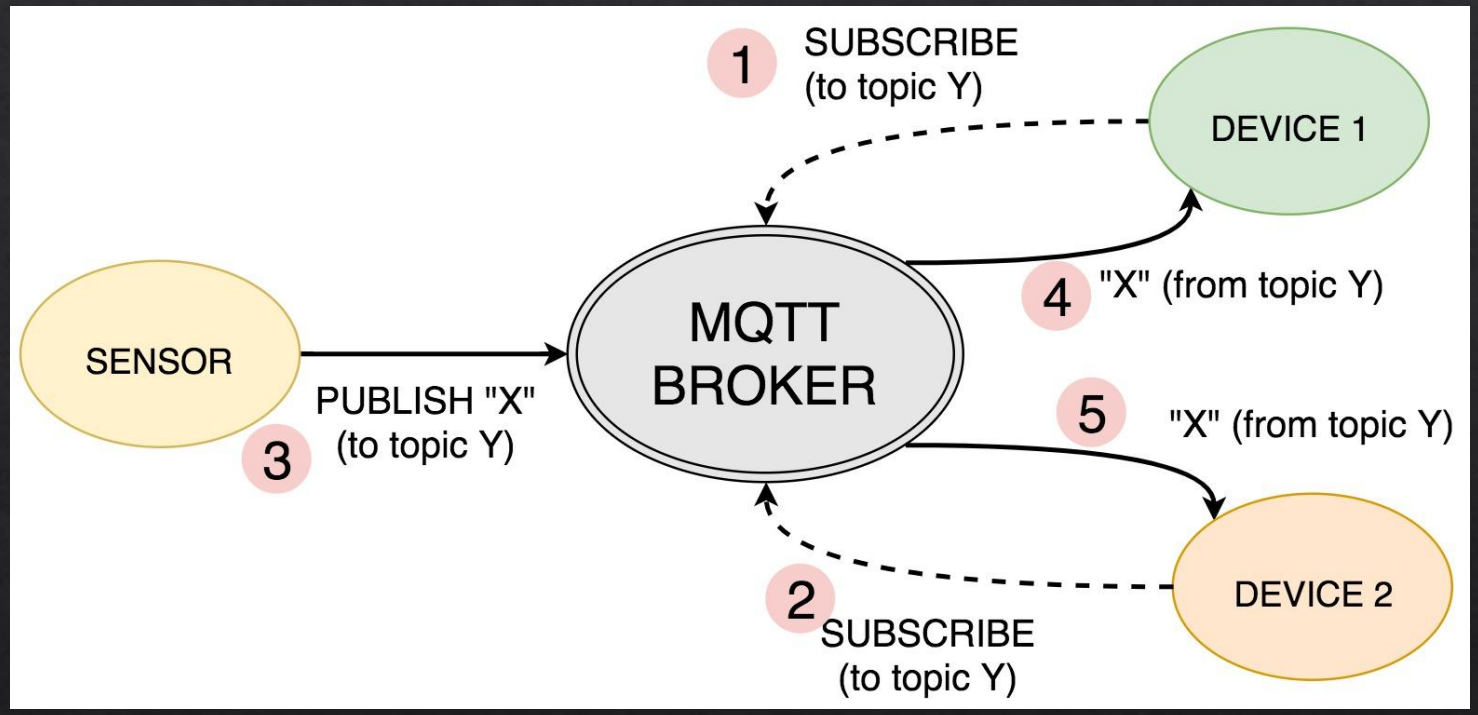
Hierarkisk topic-struktur muliggør routing, skalering og adgangskontrol

Protokol Laget (MQTT)

Publish/Subcribe Adfærd

MQTT Pub/Sub Behaviour

- Publishers og subscribers har ingen viden om hinanden – de kender kun til topics.
- Subscribere skal melde ud til brokeren at de ønsker at modtage beskeder på et bestemt topic.
- Publishers sender beskeder ud til brokeren, men ved ikke hvem der modtager dem.
- MQTT brokeren matcher topics og leverer beskeder til alle autoriserede abbonenter, der har en aktiv subscription.



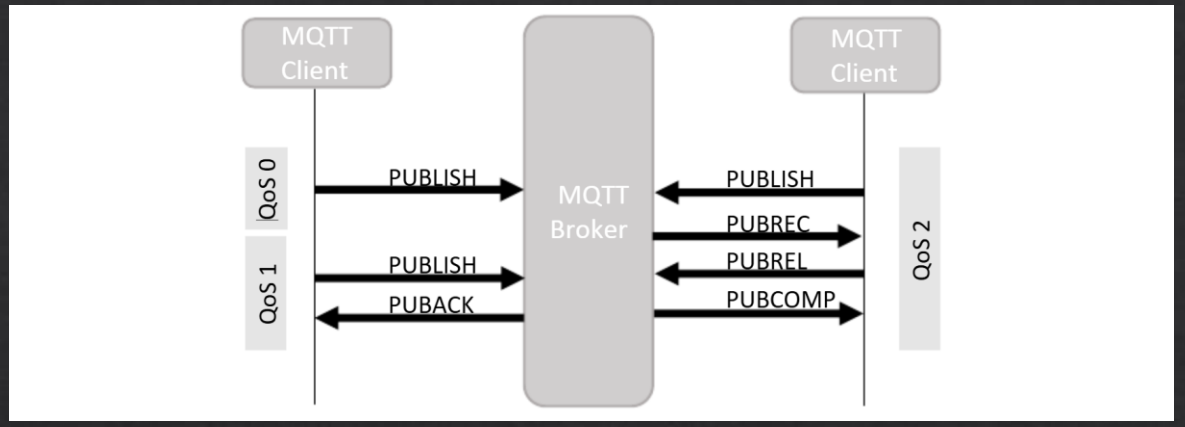
Publish X to Y → Broker Routes → Subscribers Listening on Y Recieves X

Protokol Laget (MQTT)

Reliability: QoS 1, 2 & 3

QoS — Tre Forskellige Grader

- Best Effort / at most once:
  - Minimal overhead
  - Højfrekvente Metrics
- At least once:
  - Lav overhead
  - Enestående Alerts
- Præcis én gang:
  - Høj overhead
  - Kritiske Kommandoer
  - Anvendes sjældent
  - Klienten skal bruge persistent session (clean_session=False) og stabilt client_id og broker skal have persistence slået til.



< Message Header : Fixed length >

Bits	7	6	5	4	3	2	1	0
Byte 1	Message Type				DUP Flag	QoS Level		RETAIN
Byte 2	Remaining Length							

Message Type - 4 bits :

Mnemonic	Enumeration	Description
Reserved	0000 (0)	Reserved
CONNECT	0001 (1)	Client request to connect to Server
CONNACK	0010 (2)	Connect Acknowledgement
PUBLISH	0011 (3)	Publish message
PUBACK	0100 (4)	Publish Acknowlegement
PUBREC	0101 (5)	Publish Recieved (assured delivery part 1)
PUBREL	0110 (6)	Publish Release (assured delivery part 2)
PUBCOMP	0111 (7)	Publish Complete (assured delivery part 3)
SUBSCRIBE	1000 (8)	Client Subscribe Request
SUBACK	1001 (9)	Subscribe Acknowlegement
UNSUBSCRIBE	1010 (10)	Client Unsubscribe Request
UNSUBACK	1011 (11)	Unsubscribe Acknowlegement
PINGREQ	1100 (12)	PING Request
PINGRESP	1101 (13)	PING Response
DISCONNECT	1110 (14)	Client is disconnecting
Reserved	1111 (15)	Reserved

DUP Flag (Duplicate Flag) - 1 bit :

Value	Description
0	This is not a duplicate message
1	This is a duplicate message, meaning 'I am trying to send the same message again'

https://www.researchgate.net/figure/Latency-Performance-Evaluation-of-the-Secure-Application-Layer-Protocols-in-MQTT-and-CoAP_fig3_349339

Protokol Laget (MQTT)

Reliability: LWT (Last Will and Testament)

Last Will and Testament

- Device dør -> LWT
- Status topics
- Heartbeat-systemer

Såfremt:

- TCP-forbindelsen dør uventet
- Keepalive timeout overskrides
- Klienten crasher
- Netværk ryger

Sendes LWT af broker, på vegne af klienten.

*Bør altid sættes med **qos=1** og **will_retain=True***

LWT Konfiguration

```yaml
CONNECT:
  client_id: earth-hum-01
  will_topic: rit/rorendegaard/sensor/earth-hum-01/status
  will_payload: "offline"
  will_qos: 1
  will_retain: true
```

< Message Header : Fixed length >

Bits	7	6	5	4	3	2	1	0
Byte 1	Message Type				DUP Flag	QoS Level		RETAIN
Byte 2	Remaining Length							

Message Type - 4 bits :

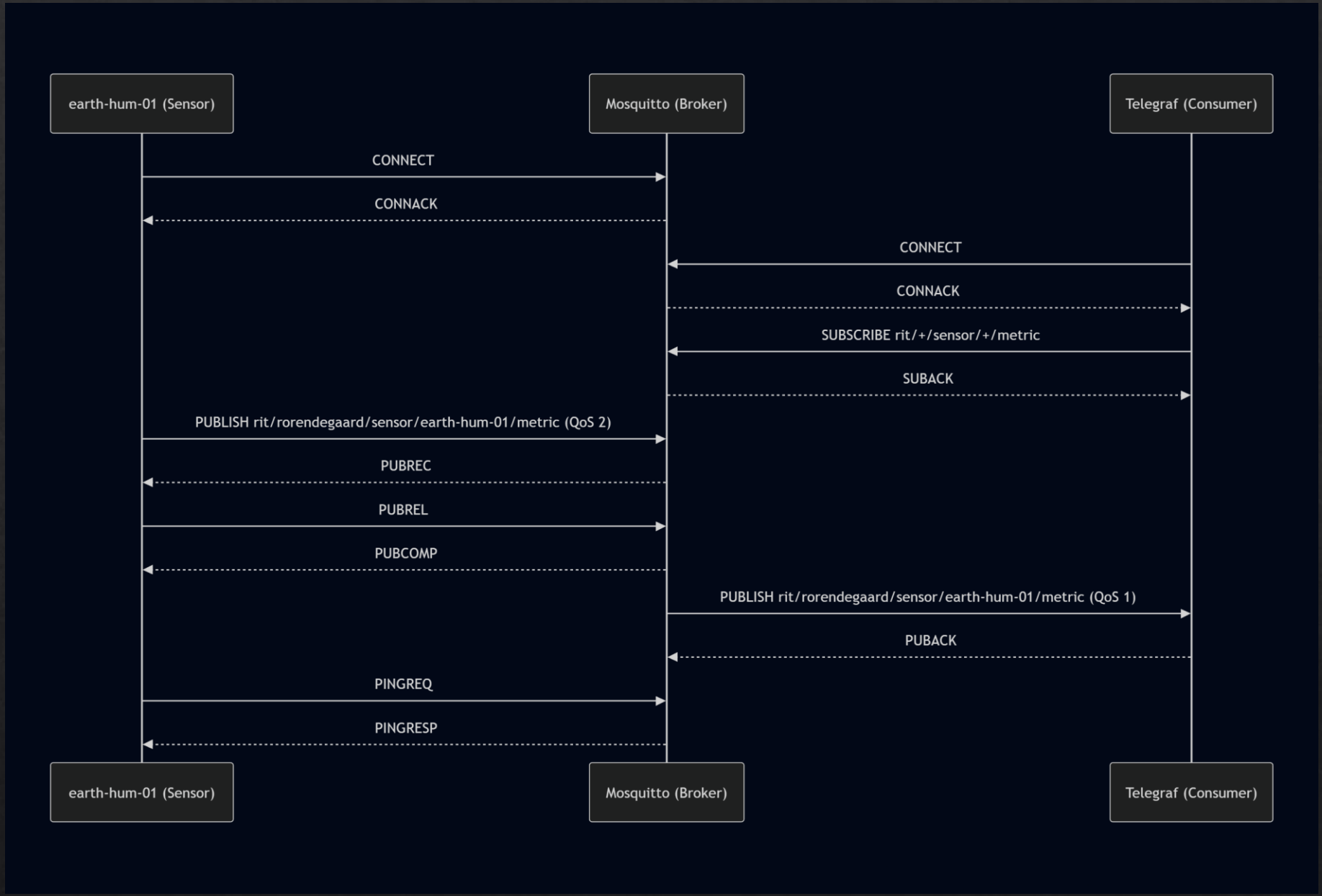
Mnemonic	Enumeration	Description
Reserved	0000 (0)	Reserved
CONNECT	0001 (1)	Client request to connect to Server
CONNACK	0010 (2)	Connect Acknowledgement
PUBLISH	0011 (3)	Publish message
PUBACK	0100 (4)	Publish Acknowlegement
PUBREC	0101 (5)	Publish Recieved (assured delivery part 1)
PUBREL	0110 (6)	Publish Release (assured delivery part 2)
PUBCOMP	0111 (7)	Publish Complete (assured delivery part 3)
SUBSCRIBE	1000 (8)	Client Subscribe Request
SUBACK	1001 (9)	Subscribe Acknowlegement
UNSUBSCRIBE	1010 (10)	Client Unsubscribe Request
UNSUBACK	1011 (11)	Unsubscribe Acknowlegement
PINGREQ	1100 (12)	PING Request
PINGRESP	1101 (13)	PING Response
DISCONNECT	1110 (14)	Client is disconnecting
Reserved	1111 (15)	Reserved

DUP Flag (Duplicate Flag) - 1 bit :

Value	Description
0	This is not a duplicate message
1	This is a duplicate message, meaning 'I am trying to send the same message again'

https://www.researchgate.net/figure/Latency-Performance-Evaluation-of-the-Secure-Application-Layer-Protocols-in-MQTT-and-CoAP_fig3_349339

MQTT Pakke Flow Illustration



MQTT Topic Routing & ACLs

Topic Routing

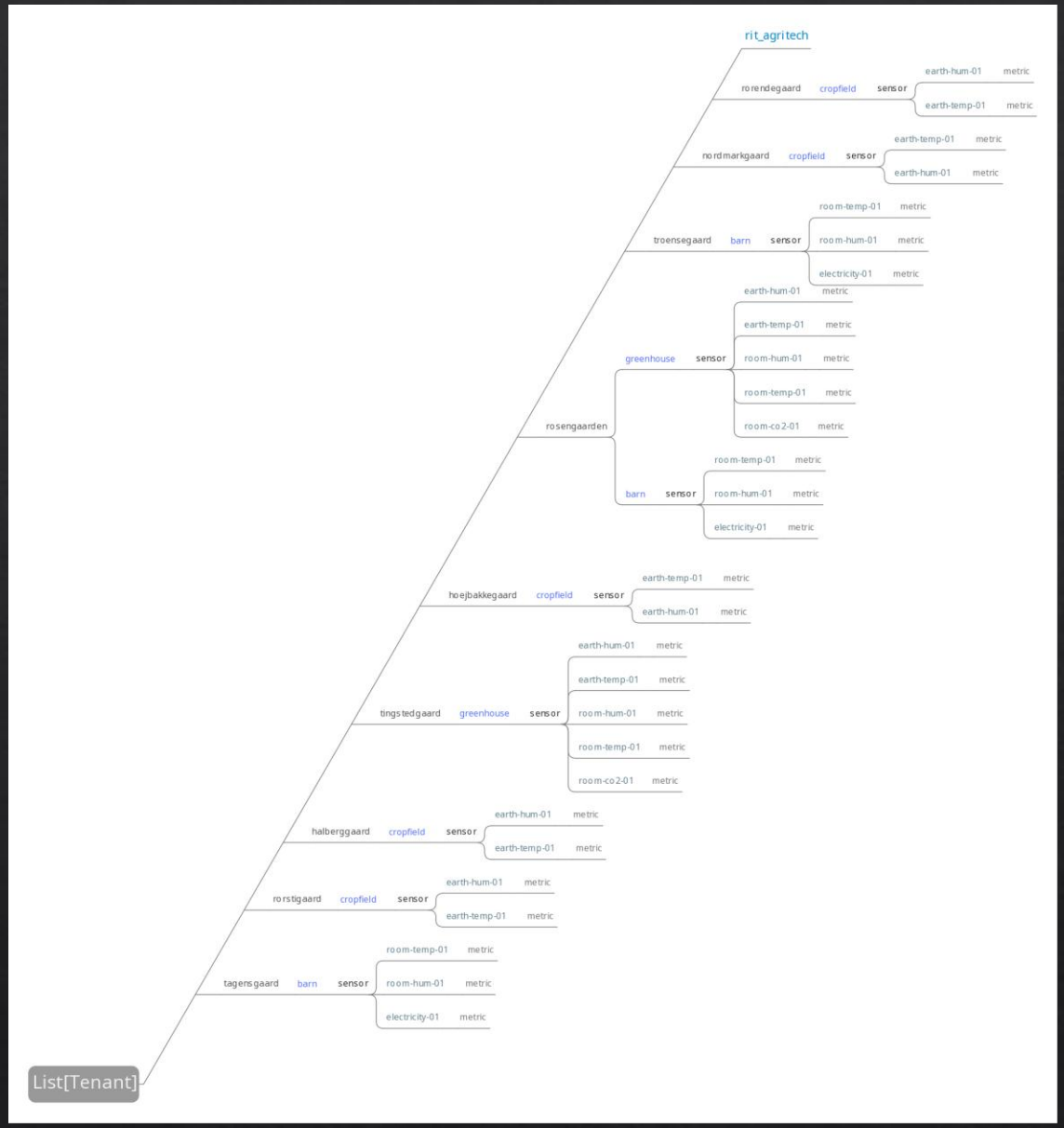
- MQTT topics er bygget op som hierakiske stier, afskilt af skråstreg.
- Topics er ikke bare ”strings/tekst”, de har semantiske betydning, fx:
- rit/rorendegaard/sensor/earth-hum-01/metric
- Jordfugtighedsmåling fra sensor earth-hum-01 på Rørendegård

Eksempler

Vi antager: **company / farm / devicetype / devicename / datatype**

Fuldt eksempel: **rit/rorendegaard/sensor/earth-hum-01/alert**

- **rit/#** → Alle data for alle companies
- **rit/+/sensor/#** → Alle sensor-data for alle companies
- **rit/agritech/sensor/+/alert** → Alle alerts fra alle sensorer i agritech



Broker Laget (Mosquitto)

Docker - Mosquitto

```yaml
version: "3.8"

networks:
  iot_net:
    driver: bridge

services:
  mosquitto:
    image: eclipse-mosquitto:2.0
    container_name: mosquitto
    ports:
      - "1883:1883"
      - "8883:8883"
    volumes:
      - ./mosquitto/mosquitto.conf:/mosquitto/config/mosquitto.conf:ro
      - ./mosquitto/config/passwords:/mosquitto/config/passwords:ro
      - ./mosquitto/config/acl:/mosquitto/config/acl:ro
      - ./mosquitto/config/certs:/mosquitto/config/certs:ro
      - mosquitto_data:/mosquitto/data
      - mosquitto_logs:/mosquitto/log
    networks:
      - iot_net
    restart: unless-stopped

volumes:
  mosquitto_data:
  mosquitto_logs:
```

Bruger Generering til Mosquitto

```
docker run --rm -it -v "$PWD:/mosquitto" eclipse-mosquitto:2.0 mosquitto_passwd -c /mosquitto/passwords telegraf
docker run --rm -it -v "$PWD:/mosquitto" eclipse-mosquitto:2.0 mosquitto_passwd -c /mosquitto/passwords simulator
```

Mosquitto ACL

```
user simulator
topic write rit_customer_agritech/rorendegaard/sensor/earth-hum-01/metric
topic write rit_customer_agritech/rorendegaard/sensor/earth-temp-01/metric
topic write rit_customer_agritech/nordmarkgaard/sensor/earth-hum-01/metric
topic write rit_customer_agritech/nordmarkgaard/sensor/earth-temp-01/metric

user telegraf
topic read rit_customer_agritech/+/sensor/+/metric

user admin
topic readwrite #
```

mosquitto.conf

```
persistence true
persistence_location /mosquitto/data/
log_dest file /mosquitto/log/mosquitto.log

log_dest stdout
log_type all

allow_anonymous false
password_file /mosquitto/config/passwords
acl_file /mosquitto/config/acl/aclfile.acl

# ================================
# Plain MQTT
# ================================
listener 1883
protocol mqtt

# ================================
# TLS MQTT
# ================================
listener 8883
protocol mqtt

# Mounted in Docker from root/ops/observability/data/certs/
cafile /mosquitto/config/certs/ca.crt
certfile /mosquitto/config/certs/mosquitto.crt
keyfile /mosquitto/config/certs/mosquitto.key

require_certificate false
```

Broker Laget: Kryptering

mosquitto.cnf

```
[req]
default_bits       = 2048
prompt             = no
default_md         = sha256
req_extensions     = req_ext
distinguished_name = dn

[dn]
C  = DK
O  = RIT
CN = mosquitto

[req_ext]
subjectAltName = @alt_names

[alt_names]
DNS.1 = mosquitto
DNS.2 = localhost
IP.1  = 127.0.0.1
IP.2  = 172.16.20.224
```

OpenSSL Certifikat Generering

```
openssl genrsa -out ca.key 4096

openssl req -x509 -new -nodes `
  -key ca.key `
  -sha256 `
  -days 3650 `
  -out ca.crt `
  -subj "/C=DK/O=RIT/CN=MQHeroCA"

openssl genrsa -out mosquitto.key 2048

openssl req -new `
  -key mosquitto.key `
  -out mosquitto.csr `
  -config mosquitto.cnf

openssl x509 -req `
  -in mosquitto.csr `
  -CA ca.crt `
  -CAkey ca.key `
  -CAcreateserial `
  -out mosquitto.crt `
  -days 825 `
  -sha256 `
  -extensions req_ext `
  -extfile mosquitto.cnf
```

Ingest / Transformations Laget (Telegraf)

Docker Compose - Telegraf

```yaml
version: "3.8"

networks:
  iot_net:
    driver: bridge

services:
  telegraf:
    image: telegraf:1.30
    container_name: telegraf
    depends_on:
      - mosquitto
      - influxdb
    volumes:
      - ./telegraf/telegraf.conf:/etc/telegraf/telegraf.conf:ro
    networks:
      - iot_net
    restart: unless-stopped

```

telegraf.conf

```
[agent]
  interval = "10s"
  flush_interval = "10s"

[[outputs.file]]
  files = ["stdout"]

[[inputs.mqtt_consumer]]
  servers = ["tcp://mosquitto:1883"]
  topics = [
    "+/+/sensor/+/metric",
    "+/+/sensor/+/alert"
  ]
  qos = 0
  client_id = "telegraf-mqtt"
  username  = "telegraf"
  password  = "KOde12345!!?"
  persistent_session = false
  data_format = "json"
  name_override = "iot_events"
  tag_keys = [
    "farm",
    "area",
    "device_id",
    "schema"
  ]

  [[inputs.mqtt_consumer.topic_parsing]]
    topic = "+/+/sensor/+/+"
    tags = "tenant/farm/device_class/device_id/message_type"

[[outputs.influxdb_v2]]
  urls = ["http://influxdb:8086"]
  token = "Sv3ndeT02qkdencauCucumfbEr"
  organization = "iot"
  bucket = "mqtt_metrics"
  tagpass = { message_type = ["metric"] }

[[outputs.influxdb_v2]]
  urls = ["http://influxdb:8086"]
  token = "Sv3ndeT02qkdencauCucumfbEr"
  organization = "iot"
  bucket = "mqtt_alerts"
  tagpass = { message_type = ["alert"] }
```

Storage Laget (InfluxDB)

Hvorfor ikke SQL?

- Sensorer producerer kontinuerlige målinger over tid
- Specialiseret database til tidsseriedata
- Optimeret til høj write-rate og append-only data

Effektiv lagring af

- Temperatur
- Fugtighed
- Driftstilstand

Hurtige queries på

- Vis sidste 24 timer
- Gennemsnit pr. gård
- Alarmer over threshold

Docker - InfluxDB

```yaml
version: "3.8"

networks:
  iot_net:
    driver: bridge

services:
  influxdb:
    image: influxdb:2.7
    container_name: influxdb
    ports:
      - "8086:8086"
    volumes:
      - influxdb_data:/var/lib/influxdb2
    environment:
      DOCKER_INFLUXDB_INIT_MODE: setup
      DOCKER_INFLUXDB_INIT_USERNAME: admin
      DOCKER_INFLUXDB_INIT_PASSWORD: admin123
      DOCKER_INFLUXDB_INIT_ORG: iot
      DOCKER_INFLUXDB_INIT_BUCKET: mqtt_metrics
      DOCKER_INFLUXDB_INIT_ADMIN_TOKEN: Sv3ndeT02qkdencauCucumfbEr
    networks:
      - iot_net
    restart: unless-stopped

volumes:
  influxdb_data:
```

Observability & Analytics Laget

Hvorfor Grafana?

- Dashboard-baseret visualisering af tidsseriedata
- Henter data fra InfluxDB via Flux
- Realtidsvisning af sensor- og driftsdata
- Understøtter grafer, aggregering og alerting

Docker - Grafana

```yaml
version: "3.8"

networks:
  iot_net:
    driver: bridge

services:
  grafana:
    image: grafana/grafana:10.2.3
    container_name: grafana
    ports:
      - "3000:3000"
    depends_on:
      - influxdb
    volumes:
      - grafana_data:/var/lib/grafana
      - ./grafana/provisioning:/etc/grafana/provisioning
      - ./grafana/dashboards:/var/lib/grafana/dashboards
    networks:
      - iot_net
    restart: unless-stopped

volumes:
  grafana_data:
```

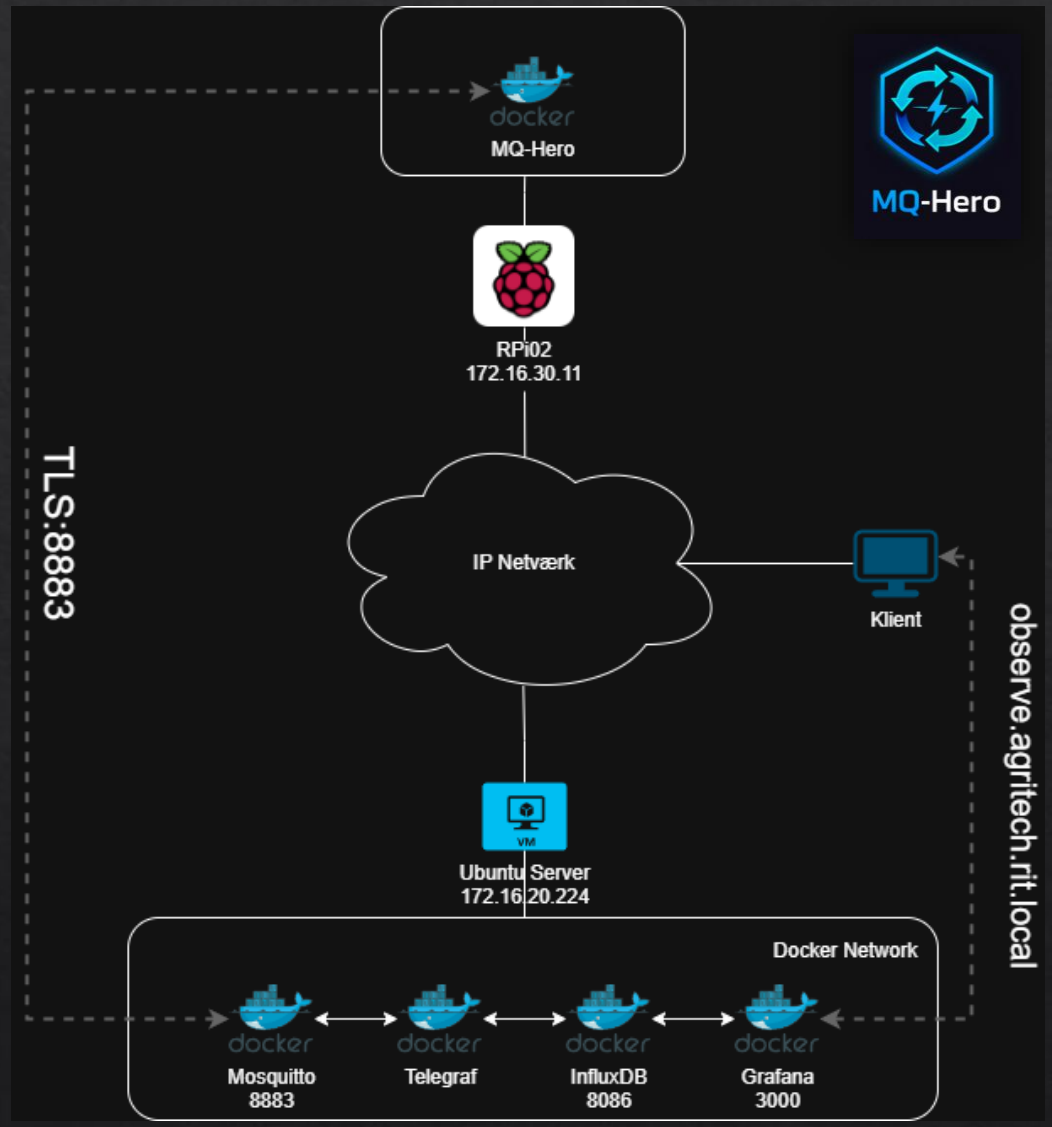
Presentations Laget (Browser / Discord)

Præsentationslaget kan enten udvides eller skiftes

- Grafana integrerer nemt alarmering med:
  - Slack
  - Teams
  - Discord
  - Telegram
  - HTTP + Webhooks

Begrænsninger

- Broker er central komponent (SPOF i PoC)
- Ingen HA cluster
- Influx retention ikke testet ved >X write rate
- Cert management manuel i PoC
- Ikke testet over WAN med høj latency
- Ikke egnet til embedded enheder
- Container-fodaftryk på RPi (v. 29 sensorer):
  - Docker image ca. 150MB
  - RAM brug: ca. 2,8% / 3800MB ~ 100MB
  - CPU brug: ca. 1%



Afrunding

Arkitekturen er lagdelt og event-drevet for at sikre løs kobling og skalerbarhed

MQTT er valgt ud fra krav om robusthed i feltmiljø

Sikkerhedsmodellen er deterministisk og default-deny for at minimere angrebsflade

Løsningen understøtter forretningskravet om central, sikker og skalerbar dataindsamling

Arkitekturen er designet, så den kan skaleres fra PoC til produktion uden redesign.