MQTT & IOT

IDEO-LAB

JANUARY 2025

BY GUILLAUME ONEILL

🌐 IoT – Internet of Things (Internet des Objets)

1. Définition

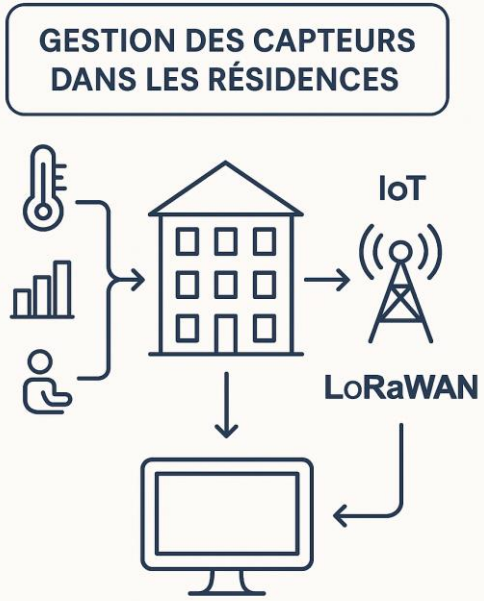
L'IoT désigne l'ensemble des objets physiques équipés de **capteurs, actionneurs, processeurs et connectivité réseau** leur permettant de :

- **Collecter** des données de leur environnement (température, pression, mouvement, localisation, etc.).
- **Communiquer** ces données via Internet ou un réseau privé.
- **Interagir** avec d'autres systèmes, applications ou plateformes.
- **Prendre des décisions locales** (edge computing) ou remonter des données vers le cloud.

IDEO·LAB

RAPPEL TECHNOLOGIES MAJEURES

- IoT
- MQTT
- LoRaWAN
- BigData


GESTION DES CAPTEURS
DANS LES RÉSIDENCES
IoT
LoRaWAN

DATA COLLECTION AND STORAGE

- Big Data

- API REST & SDK

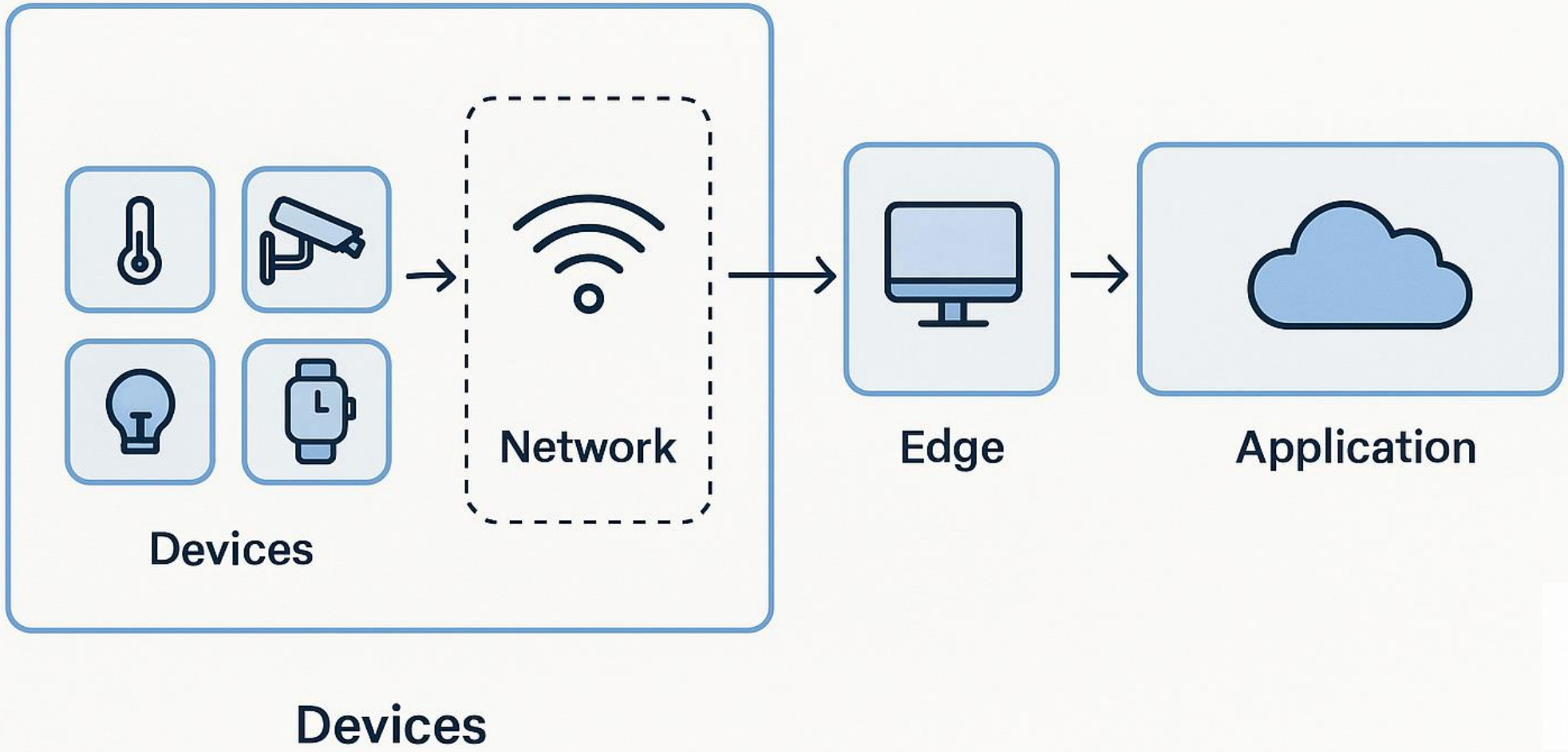

BIGDATA

AGENDA IOT

- 1. **Définition**
- 2. **Architecture technique de l’IoT**
- 3. **Standards et Protocoles clés**
- 4. **Enjeux techniques**
- 5. **Cas d’usage IoT**
- 6. **Écosystème technologique**
- 7. **Tendances futures**
- **8. Monitoring & Tuning**
- **Big Data**

IDEO·LAB

IoT (Internet des Objets)



IDEO·LAB

2. Architecture technique de l'IoT

On peut décomposer une architecture IoT en plusieurs couches :

🔹 a) Couche perception (devices)

- **Capteurs** : température, luminosité, pression, biométriques, GPS, etc.
- **Actionneurs** : moteurs, relais, LED, électrovannes.
- **Modules embarqués** : microcontrôleurs (Arduino, STM32, ESP32) ou cartes plus puissantes (Raspberry Pi, Nvidia Jetson).

🔹 b) Couche réseau

- Protocoles de communication :
  - **Sans fil courte portée** : Bluetooth LE, Zigbee, Z-Wave, NFC.
  - **Sans fil longue portée (faible consommation)** : LoRaWAN, Sigfox, NB-IoT, LTE-M.
  - **Classiques** : Wi-Fi, Ethernet, 4G/5G.
- Contraintes : **latence**, **bande passante**, **consommation énergétique**, **sécurité**.

🔹 c) Couche traitement (Edge & Cloud)

- **Edge computing** : traitement local (passerelles IoT, routeurs intelligents) pour réduire la latence.
- **Cloud computing** : stockage massif et analyse de données (AWS IoT, Azure IoT Hub, Google Cloud IoT, Oracle IoT Cloud).
- **Middleware IoT** : message brokers (MQTT, AMQP, CoAP), plateformes de gestion de flotte, APIs REST/GraphQL.

IDEO·LAB

d) Couche application

- Applications web/mobile (tableaux de bord, supervision).
- Intégration avec **IA / Machine Learning** pour la détection d'anomalies, la maintenance prédictive.
- Automatisation via **SCADA**, ERP, CRM, domotique, etc.

IDEO·LAB

3. Standards et Protocoles clés

- **MQTT** (Message Queuing Telemetry Transport) : léger, adapté aux objets peu puissants.
- **CoAP** (Constrained Application Protocol) : basé sur HTTP, optimisé IoT.
- **AMQP** : plus lourd, pour applications industrielles.
- **OPC-UA** : standard industriel pour automatisation et interopérabilité.
- **HTTP/REST & WebSockets** : classiques mais énergivores.

IDEO·LAB

4. Enjeux techniques

🔐 Sécurité

- Authentification forte (certificats, clés).
- Chiffrement TLS/DTLS.
- Gestion des mises à jour OTA (Over The Air).
- Protection contre les attaques DDoS via botnets d'objets connectés (ex. Mirai).

⚡ Consommation énergétique

- Optimisation hardware (modes sommeil).
- Protocoles basse consommation.
- Batteries longue durée ou récupération d'énergie (energy harvesting).

📡 Scalabilité

- Gestion de millions d'objets simultanément.
- Déploiement de brokers distribués.
- Cloud-native & microservices.

📊 Data Management

- Volume de données massif ("Big Data IoT").
- Besoin de traitements temps réel (Kafka, Flink, Spark Streaming).
- Stockage optimisé (TSDB : InfluxDB, TimescaleDB).

IDEO·LAB

5. Cas d'usage IoT

- **Industrie 4.0** : machines connectées, maintenance prédictive.
- **Smart Cities** : capteurs pollution, éclairage intelligent, gestion du trafic.
- **Santé** : objets médicaux connectés, télémédecine, wearables.
- **Agriculture** : capteurs d'humidité, irrigation automatisée.
- **Domotique** : thermostats intelligents, caméras, assistants vocaux.
- **Transport/Logistique** : traçabilité en temps réel (RFID + IoT).

IDEO·LAB

- **Hardware** : ESP32, STM32, Arduino, Raspberry Pi.
- **OS embarqués** : FreeRTOS, Zephyr, RIOT.
- **Plateformes Cloud** : AWS IoT Core, Azure IoT Hub, Google IoT Core (deprecated), Oracle IoT Cloud, IBM Watson IoT.
- **Edge AI** : TensorFlow Lite, TinyML, Nvidia Jetson.
- **Standards industriels** : OPC-UA, Modbus, BACnet.

---

7. Tendances futures

- **5G & IoT massif (mMTC)** : millions d'objets/km².
- **Edge AI** : traitement IA directement sur microcontrôleurs.
- **Blockchain + IoT** : traçabilité et sécurité renforcée.
- ** jumeaux numériques (digital twins)** : simulation en temps réel des objets physiques.
- **Green IoT** : capteurs autonomes énergétiquement.

IDEO·LAB

Standards et Protocoles clés

IDEO·LAB

AGENDA PROTOCOLES & TRANSPORT

- **MQTT** (*Message Queuing Telemetry Transport*)
- Réseau **LoRaWAN**
- **RabbitMQ**

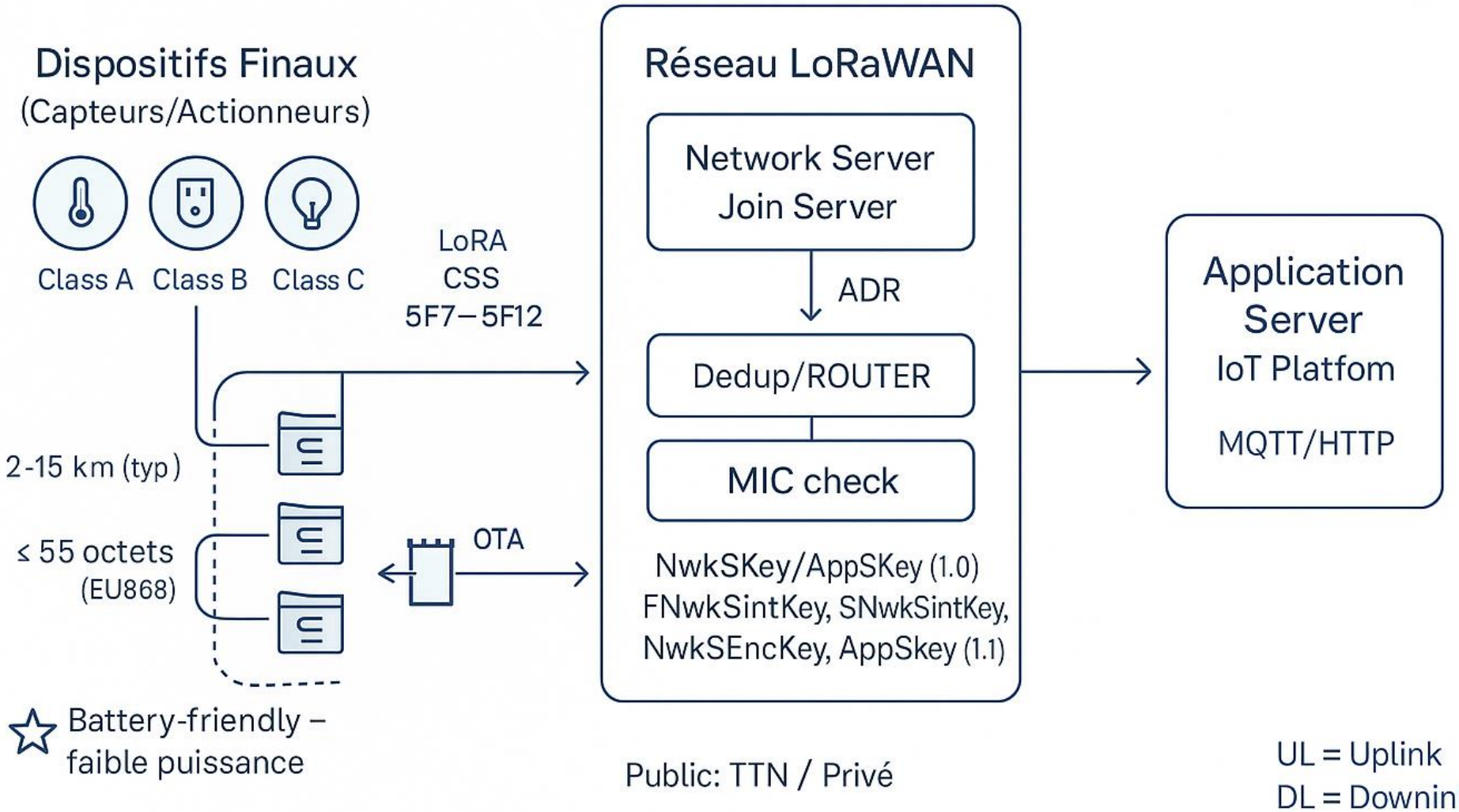
IDEO·LAB

LoRaWAN

IDEO·LAB

IDEO·LAB

LoRaWAN – Flux de fonctionnement



UL = Uplink
DL = Downink

1. Présentation générale

- **Nom complet** : *Long Range Wide Area Network*
- **Créateur** : Cycleo (acquis par Semtech en 2012) → LoRa Alliance (2015).
- **Type** : Réseau LPWAN (*Low Power Wide Area Network*).
- **Objectif** : Communication longue portée et faible consommation pour l'IoT.
- **Fréquences utilisées** : bandes ISM libres (EU868, US915, AS923, etc.).
- **Débit** : 0.3 kbps à 50 kbps (selon configuration).
- **Portée typique** :
  - Urbain : 2–5 km
  - Rural : 10–15 km (jusqu'à 30 km dans conditions optimales)
- **Taille max message** :
  - 51 octets (EU868 en DR0)
  - 242 octets (certaines régions, DR7+)

2. Architecture technique

🔹 a) Éléments principaux

1. **End Devices (capteurs/actionneurs)**
   - Fonctionnent sur piles (autonomie 5–10 ans).
   - Classes de fonctionnement :
     - **Classe A** : communications ascendantes (uplink) + fenêtre courte de réception. → *ultra low power*.
     - **Classe B** : ajout de slots de réception synchronisés (via balises gateway).
     - **Classe C** : écoute permanente (forte consommation).
2. **Gateways (passerelles LoRa)**
   - Reçoivent les signaux LoRa (radio) → envoient sur IP (Ethernet, 4G, fibre).
   - Mode "star-of-stars" : toutes les gateways couvrent la zone et forwardent vers le réseau.
3. **LoRaWAN Network Server (LNS)**
   - Gestion des devices, des clés de sécurité, et du routage.
   - Fonctions : **deduplication**, **MIC check (Message Integrity Code)**, **ADR (Adaptive Data Rate)**.
4. **Application Server**
   - Déchiffre la charge utile applicative (*AppSKey*).
   - Interface avec la plateforme IoT / Cloud (MQTT, HTTP, gRPC).

3. Couches protocolaires

- **LoRa (PHY)** : modulation radio **CSS (Chirp Spread Spectrum)** → robuste au bruit et interférences.
- **LoRaWAN (MAC + Réseau)** : encapsule LoRa et définit la sécurité, la gestion des clés, les classes, l'ADR.

---

4. Sécurité

- **Double chiffrement** (AES-128) :
  1. **NwkSKey** : sécurité réseau (intégrité, contrôle du routage).
  2. **AppSKey** : sécurité applicative (contenu des messages).
- **Activation modes** :
  - **ABP (Activation By Personalization)** : clés fixes, peu sécurisé.
  - **OTAA (Over-The-Air Activation)** : handshake dynamique → plus sécurisé (échange avec Join Server).
- Protection contre la relecture : compteur de trames (*Frame Counter*).

---

5. Communication

- **Uplink (UL)** : messages du device → gateway → serveur.
- **Downlink (DL)** : messages serveur → device (moins fréquent, limité).
- **ADR (Adaptive Data Rate)** : optimisation dynamique du débit, SF (*spreading factor*), et puissance → équilibre consommation / portée.
- **Duty Cycle** : limitation d'émission (ex : 1% en EU868).

6. Classes d'appareils

Classe	Description	Consommation	Cas d'usage
A	Uplink initié par device, petite fenêtre de réception	Très faible	Capteurs batterie (10 ans)
B	Fenêtres de réception synchronisées par balises gateway	Moyenne	Objets nécessitant plus de downlink
C	Réception continue	Élevée	Actionneurs secteur (lampadaires, vannes)

7. Performances

- **Portée** : jusqu'à 15 km (zone rurale dégagée).
- **Nombre de devices** : > 10^6 par réseau (avec clustering gateways).
- **Consommation** : quelques µA en sommeil, ≈ 40 mA en émission.
- **Latence** : variable (quelques secondes), pas adaptée au temps réel strict.

---

8. Cas d'usage

- **Smart Cities** : parkings connectés, éclairage intelligent, capteurs pollution.
- **Agriculture intelligente** : capteurs humidité/sol, suivi bétail.
- **Industrie 4.0** : maintenance prédictive, capteurs vibration/température.
- **Santé** : suivi patients (capteurs médicaux basse fréquence).
- **Logistique** : suivi palettes/containers (tracking basse fréquence).
- **Environnement** : qualité de l'air, détection d'inondations/incendies.

---

9. Écosystème

- **LoRa Alliance** : standardisation, certification.
- **Réseaux publics** :
  - *The Things Network (TTN)* : réseau communautaire mondial.
  - Opérateurs : Orange, Bouygues, KPN, Swisscom, Proximus.
- **Serveurs réseaux open-source** : ChirpStack, The Things Stack.
- **Constructeurs hardware** : Semtech (SX127x, SX130x), Kerlink, Multitech, IMST.

11. Limites

- Débit très bas → pas adapté aux vidéos ou flux lourds.
- Latence → non adapté au contrôle en temps réel (robotique critique).
- Réglementation stricte (duty cycle limité).
- Interférences possibles si saturation de bande ISM.

---

12. Tendances futures

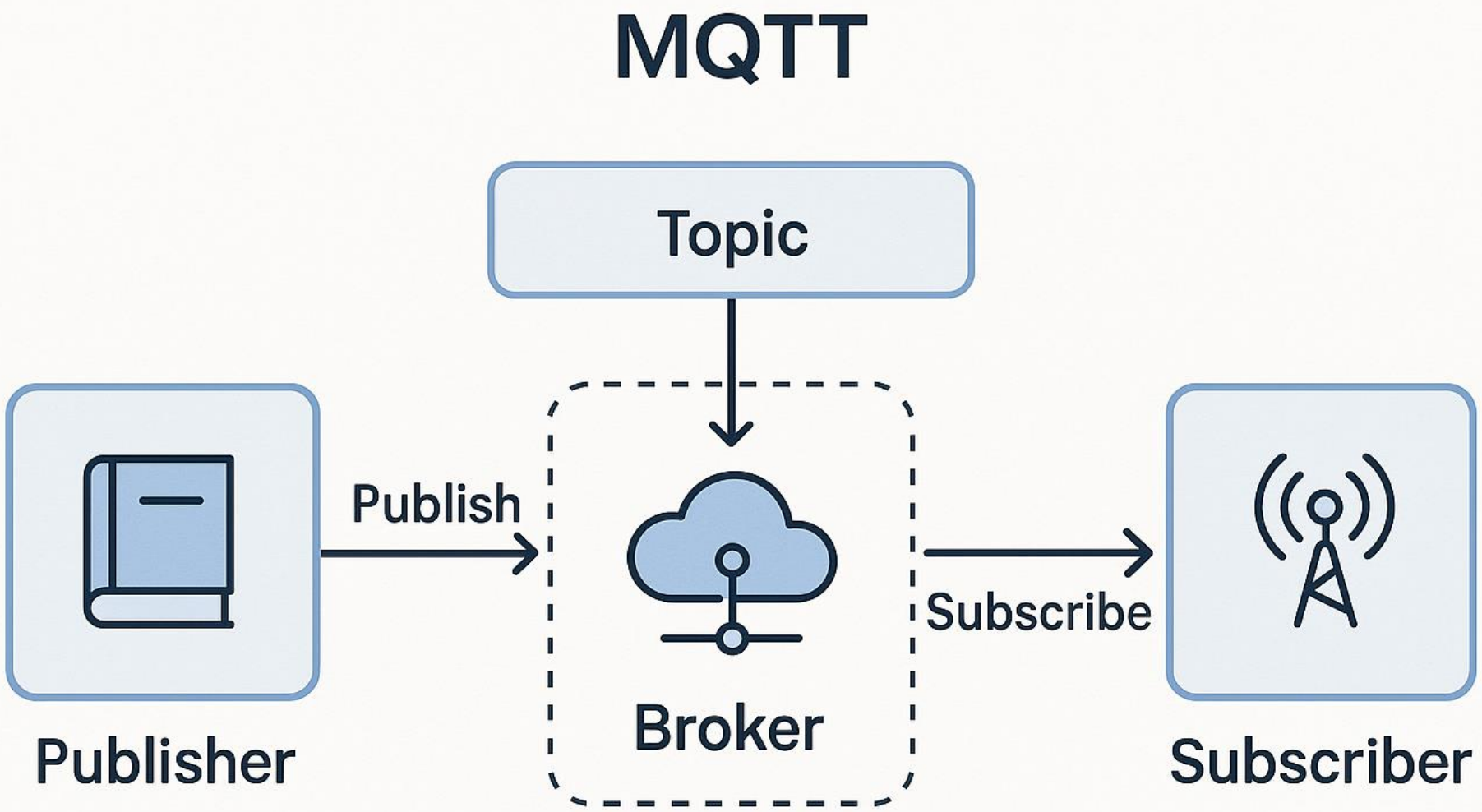
- **LoRaWAN 1.1+** : sécurité renforcée, nouveaux jeux de clés.
- **Satellite + LoRa** (LoRaWAN direct-to-satellite, ex : Lacuna Space, Swarm).
- **Edge AI + LoRa** : traitement embarqué intelligent.
- **Hybridation 5G / LoRaWAN** : combiner temps réel (5G) et bas débit longue portée (LoRa).

10. Comparaison avec autres LPWAN

Techno	Débit	Portée	Conso	Licence	Avantages	Limites
LoRaWAN	0.3–50 kbps	2–15 km	Très faible	ISM (gratuit)	Longue portée, open, écosystème large	Débit faible, latence
Sigfox	100 bps	10–50 km	Très faible	Propriétaire	Portée énorme, coût faible	Fermé, dépend opérateur
NB-IoT	26–250 kbps	1–10 km	Faible	LTE (licence opérateur)	Intégré réseau 4G/5G, QoS	Consommation > LoRa
LTE-M	200–1000 kbps	1–10 km	Moyenne	LTE	Voix/temps réel, mobilité	Besoin opérateur

MQTT

IDEO·LAB


MQTT
Topic
Publish
Broker
Subscribe
Publisher
Subscriber

AGENDA MQTT

- 1. **Présentation générale**
- 2. **Principes fondamentaux**
- 3. **Caractéristiques techniques**
- 4. **Qualité de Service (QoS)**
- 5. **Gestion des sessions**
- 6. **Fonctionnalités avancées**
- 7. **Sécurité**
- 8. **Écosystème logiciel**
- 9. **Performances**
- 10. **Cas d’usage**
- 11. **Comparaison avec autres protocoles IoT**
- 12. **Tendances et évolutions**

IDEO·LAB

1. Présentation générale

- **Nom complet** : MQTT = *Message Queuing Telemetry Transport*
- **Type** : Protocole de messagerie léger basé sur le modèle *publish/subscribe*
- **Origine** : Créé en 1999 par IBM & Eurotech pour superviser des pipelines pétroliers sur satellite (réseau instable + faible bande passante).
- **Norme** : OASIS standard (2014), ISO/IEC 20922 (2016).
- **Domaines d'application** : IoT, M2M, domotique, SCADA, mobile, capteurs, monitoring en temps réel.

---

2. Principes fondamentaux

🔹 Modèle Publish / Subscribe

- **Publisher** : envoie un message sur un *topic* (ex : `sensors/temperature/lab1`).
- **Broker** : serveur central qui reçoit tous les messages et les redistribue.
- **Subscriber** : s'abonne à un ou plusieurs *topics* pour recevoir les messages correspondants.

👉 Avantages :

- Découplage fort (les publishers et subscribers ne se connaissent pas).
- Scalabilité (un message peut être reçu par 1, 10 ou 10 000 abonnés).
- Optimisé pour les réseaux contraints (bande passante, énergie, latence).

IDEO·LAB

3. Caractéristiques techniques

- **Transport** : TCP/IP (par défaut), UDP possible avec MQTT-SN.
- **Port par défaut** :
  - `1883` (non chiffré)
  - `8883` (chiffré TLS/SSL)
- **Taille minimale d'un message** : 2 octets.
- **Latence** : < 100 ms sur réseaux classiques.
- **Overhead réseau** : très faible (idéal pour LPWAN, 2G, GPRS, LoRaWAN via passerelle).

---

4. Qualité de Service (QoS)

MQTT définit 3 niveaux de fiabilité de livraison :

- **QoS 0 – At most once** : *Fire and forget* (pas d'accusé de réception).
- **QoS 1 – At least once** : message reçu au moins une fois (duplication possible).
- **QoS 2 – Exactly once** : message reçu une seule fois, garanti (plus coûteux).

👉 Utilisation typique :

- **QoS 0** : capteurs envoyant beaucoup de données (ex : température toutes les 10s).
- **QoS 1** : notifications critiques (alarme incendie).
- **QoS 2** : transactions financières, systèmes de facturation.

IDEO·LAB

5. Gestion des sessions

- **Clean Session** : si activée, pas de stockage de l'état (abonnements temporaires).
- **Persistent Session** : le broker garde en mémoire les abonnements et messages en attente si le client se déconnecte.

---

6. Fonctionnalités avancées

- **Last Will and Testament (LWT)** :
  Le client peut définir un message qui sera publié automatiquement par le broker si la connexion est interrompue brutalement.
  Exemple : un capteur tombe en panne → publication sur `status/device123` = `"offline"`.
- **Retained Messages** :
  Le broker garde en mémoire le dernier message d'un topic pour l'envoyer immédiatement aux nouveaux abonnés.
- **Wildcards dans les topics** :
  - `+` (un niveau) → `sensors/+/lab1`
  - `#` (multi-niveaux) → `sensors/#`

IDEO·LAB

7. Sécurité

- **Authentification** : login/mot de passe, certificats X.509, OAuth 2.0 (selon broker).
- **Chiffrement** : TLS/SSL (MQTT over TLS = MQTTS).
- **ACL (Access Control List)** : restriction fine des topics par utilisateur/appareil.

⚠️ Faiblesses fréquentes :

- Beaucoup de déploiements laissent MQTT ouvert en clair (`1883`) → vulnérable aux interceptions & injections.
- Botnets (ex : **Mozi**) exploitent MQTT mal configuré.

---

8. Écosystème logiciel

🔹 Brokers MQTT

- **Mosquitto** (Eclipse) – open source, très populaire.
- **EMQX** – open source/enterprise, très scalable (cluster).
- **HiveMQ** – commercial, orienté haute disponibilité & entreprise.
- **RabbitMQ** (avec plugin MQTT).
- **AWS IoT Core**, **Azure IoT Hub**, **IBM Watson IoT**, **Google Cloud IoT** (supportaient MQTT).

🔹 Clients MQTT

- Langages supportés : Python (Paho), Node.js (mqtt.js), Java, C/C++, Go, Rust.
- Libs populaires : Eclipse Paho, AsyncMQTT, Mosquitto Clients.

9. Performances

- **Scalabilité** : +10 millions de connexions simultanées (EMQX benchmark).
- **Consommation énergétique** : très faible (idéal capteurs sur batterie).
- **Tolérance réseau** : reconnecte automatiquement, buffers messages en cas de coupure.

---

10. Cas d'usage

- **Domotique** : Home Assistant, OpenHAB, Jeedom → pilotage maison connectée.
- **Industrie 4.0** : monitoring machines, maintenance prédictive.
- **Smart Cities** : éclairage public, capteurs pollution.
- **Santé** : dispositifs médicaux connectés.
- **Transport/logistique** : suivi GPS de flottes en temps réel.

IDEO·LAB

11. Comparaison avec autres protocoles IoT

Protocole	Modèle	Consommation	Sécurité	Cas d'usage
MQTT	Pub/Sub	Très faible	TLS, ACL	IoT générique, domotique
CoAP	REST/UDP	Faible	DTLS	Capteurs contraints, IPv6
AMQP	Message Queue	Moyen	TLS, Auth	Banque, industrie critique
HTTP	Request/Response	Élevée	TLS	Web, API classiques

IDEO·LAB

12. Tendances et évolutions

- MQTT v5 (2019) :
  - Raison de déconnexion.
  - Partage de charge (*shared subscriptions*).
  - Propriétés dans les messages (métadonnées enrichies).
- **MQTT-SN** (Sensor Networks) : version optimisée pour réseaux sa
- **Intégration IA/Edge Computing** : filtrage et pré-analyse des dor
- **Interopérabilité OPC-UA / MQTT** dans l'industrie 4.0.

IDEO·LAB

BROKERS MQTT

IDEO·LAB

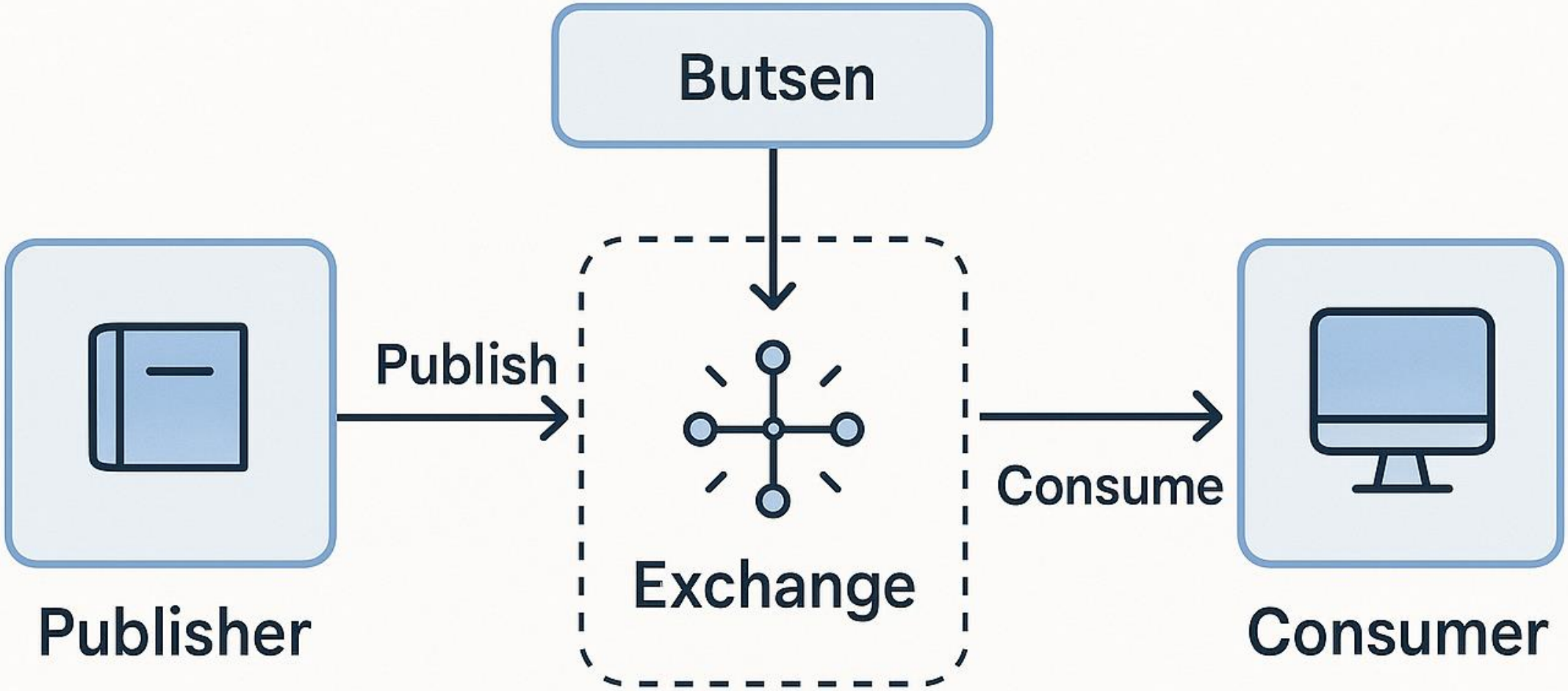
AGENDA BROKERS MQTT

- 1. RabbitMQ avec plugin MQTT

IDEO·LAB

RabbitMQ

RABBITMQ



1. Présentation générale

- **RabbitMQ** : Broker de messages open source (créé en 2007, basé sur Erlang).
- **Spécialité** : Implémentation du protocole **AMQP 0-9-1** (Advanced Message Queuing Protocol).
- **Forces** : fiabilité, persistance des messages, clustering, haute disponibilité.
- **Extensible** : via **plugins** → STOMP, AMQP 1.0, MQTT, WebSockets, etc.
- **Plugin MQTT** : permet d'utiliser **MQTT (v3.1.1 et v5)** comme protocole de communication, tout en stockant les messages et en gérant les flux via RabbitMQ.

👉 En résumé : **RabbitMQ** = **moteur de messagerie robuste**, et avec le plugin MQTT, il devient un **broker MQTT d'entreprise**.

---

2. Architecture technique

🔹 Couches logicielles

1. **Clients MQTT** → publient/s'abonnent sur des *topics*.
2. **Plugin MQTT RabbitMQ** → convertit MQTT en AMQP interne.
3. **RabbitMQ Core** → gestion des **queues**, **exchanges**, **routing** et **persistances**.
4. **Subscribers MQTT ou AMQP** → reçoivent les messages.

👉 Ainsi, RabbitMQ agit comme un **pont MQTT ↔ AMQP**, avec plus de puissance que les brokers MQTT purs (ex : Mosquitto).

IDEO·LAB

3. Protocoles supportés

- **AMQP 0-9-1** (par défaut RabbitMQ).
- **MQTT 3.1.1 & 5.0** (via plugin).
- **STOMP** (plugin).
- **HTTP API** (REST).
- **WebSockets** (pour MQTT over WS).

---

4. Installation du plugin MQTT

- Activer le plugin :

```bash
rabbitmq-plugins enable rabbitmq_mqtt
rabbitmq-plugins enable rabbitmq_web_mqtt   # si usage WebSocket
```

- Ports par défaut :
  - `1883` → MQTT non sécurisé.
  - `8883` → MQTT over TLS.
  - `15675` → Web MQTT (via WebSockets).

IDEO·LAB

5. Mapping MQTT ↔ AMQP

Le plugin traduit les concepts MQTT en **objets RabbitMQ** :

Concept MQTT	Équivalent RabbitMQ
Topic MQTT (`sensors/temp/lab1`)	Exchange `amq.topic` + Routing key
Publisher MQTT	Producteur AMQP
Subscriber MQTT	Consommateur AMQP
QoS MQTT	Ack & redelivery RabbitMQ
Retained Messages	Non natif, simulé via exchange durable

👉 Cela permet d'utiliser MQTT tout en bénéficiant de la puissance des **exchanges et queues RabbitMQ**.

---

6. Qualité de Service (QoS)

- MQTT **QoS 0/1/2** sont mappés vers RabbitMQ.
- **QoS 2** : converti en un ACK robuste → pas natif, mais RabbitMQ gère via persistance.
- Persistance possible (messages stockés sur disque).

IDEO·LAB

7. Sécurité

RabbitMQ offre plus de sécurité que Mosquitto classique :

- **TLS/SSL** pour MQTT (port 8883).
- **Authentification** : login/password, certificats X.509.
- **ACL (Access Control List)** : contrôle par *vhosts* et permissions fines.
- **Federation & Shovel plugins** : réplication sécurisée inter-sites.

---

8. Haute disponibilité & Scalabilité

- **Cluster RabbitMQ** : plusieurs nœuds RabbitMQ avec réplication des queues.
- **Sharding** : distribution automatique des messages sur plusieurs brokers.
- **Federation** : communication inter-datacenters (multi-sites).
- **Plugins de monitoring** : `rabbitmq_management` avec UI web + Prometheus/Grafana.

👉 RabbitMQ dépasse Mosquitto ou EMQX pour des **scénarios industriels lourds**.

IDEO·LAB

9. Cas d'usage typiques

1. **IoT Entreprise**
   - Capteurs MQTT → RabbitMQ → Big Data (Kafka, Spark, Hadoop).
   - Fiabilité & persistance → ne pas perdre de messages critiques.
2. **Industrie 4.0**
   - Machines connectées MQTT → RabbitMQ → ERP / MES.
   - Intégration avec SCADA, OPC-UA.
3. **Applications hybrides**
   - Clients mobiles MQTT.
   - Back-end applicatif en AMQP (Java, Python, Go).
4. **Smart Cities**
   - Réseaux de capteurs distribués, avec agrégation fiable.

IDEO·LAB

10. Points forts vs Mosquitto / EMQX

Critère	RabbitMQ + MQTT	Mosquitto	EMQX
Robustesse	Haute (persistance, clustering)	Moyenne	Haute
Performance	Moins rapide que Mosquitto pur	Très rapide	Très scalable
Protocoles multiples	Oui (MQTT, AMQP, STOMP, WS)	Non (juste MQTT)	Oui (MQTT, CoAP, etc.)
UI & Monitoring	UI web native + Prometheus	Basique	UI avancée
Cas idéal	Entreprises, cloud hybride	IoT léger/domotique	IoT massif (10M devices)

IDEO·LAB

11. Limitations

- Plus lourd que Mosquitto (besoin de plus de ressources CPU/RAM).
- Latence légèrement plus élevée que brokers MQTT purs.
- Retained Messages MQTT pas totalement équivalents (simulés).
- Pas optimisé pour **ultra-haute charge IoT (10M devices)** → EMQX plus adapté.

---

12. Tendances et évolutions

- Intégration accrue avec **Kubernetes (RabbitMQ Operator)**.
- **MQTT v5 natif** avec propriétés avancées (raison de déconnexion, shared subscriptions).
- Ponts **RabbitMQ ↔ Kafka** pour ingestion Big Data IoT.
- Utilisation conjointe avec **Elixir/NATS** pour systèmes distribués.

MONITORING

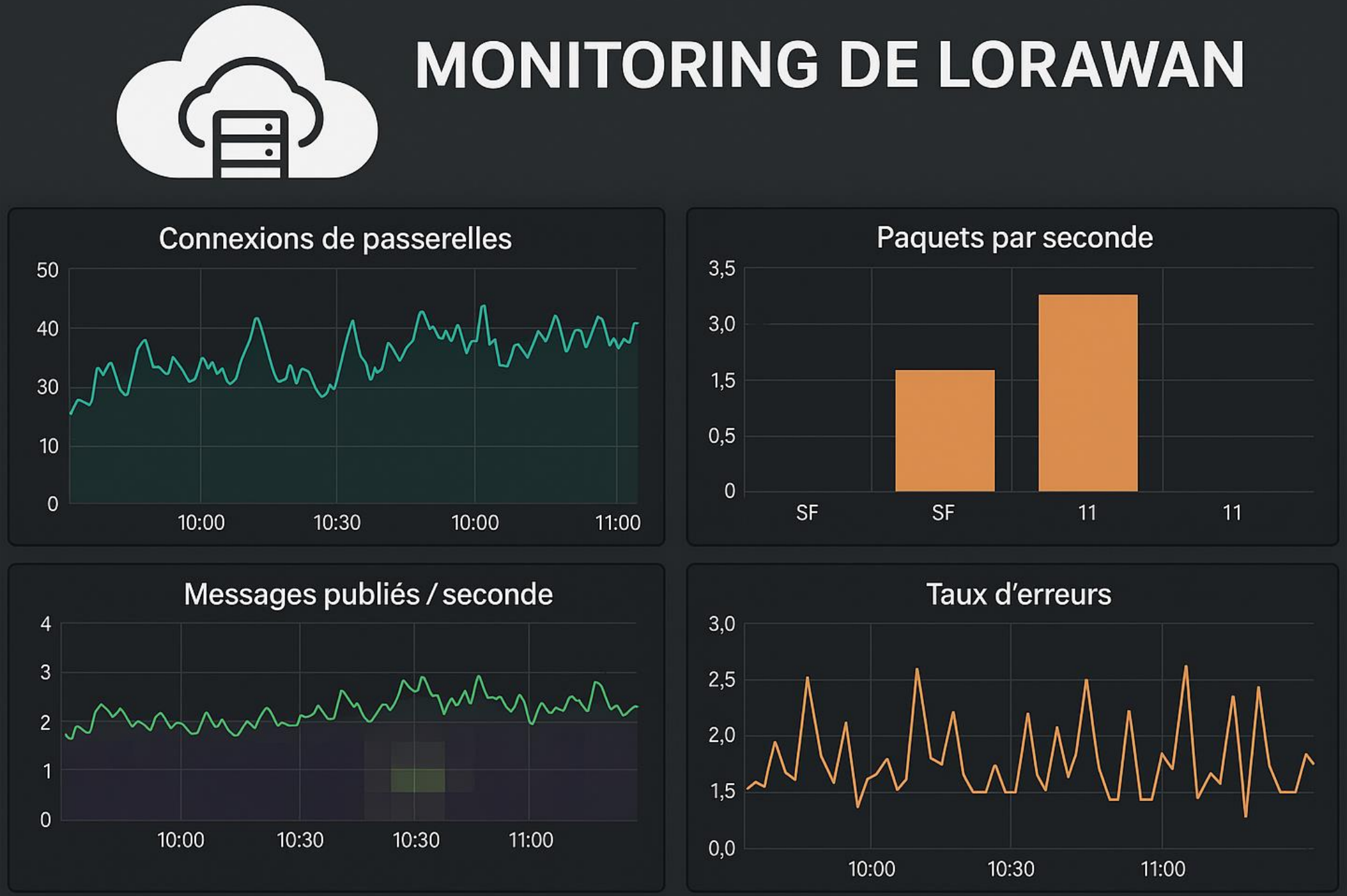
AGENDA MONITORING

- 📡 Monitoring de LoRaWAN
- 📡 Monitoring de MQTT

LoraWAN

IDEO·LAB

MONITORING DE LORAWAN



1. Pourquoi monitorer LoRaWAN ?

- Vérifier **l'état des gateways** et leur couverture.
- Surveiller **la qualité des liaisons radio** (SNR, RSSI).
- Détecter des problèmes de **capacité** (trop de capteurs saturant une gateway).
- Assurer la **sécurité** (détection d'anomalies, clés invalides).
- Mesurer **l'efficacité énergétique** des capteurs (batterie, duty cycle).
- Assurer la **QoS des applications IoT** (latence, taux de perte de paquets).

2. Points de monitoring clés

🔹 a) Niveau Device (Capteurs/Actionneurs)

- **État de la batterie** (autonomie restante).
- **Uplink success rate** (taux de transmission réussie).
- **Downlink latency** (délai de réception des commandes).
- **Compteurs de trames** (*Frame Counters*) → détection de pertes ou relectures.
- **ADR (Adaptive Data Rate)** : vérifier si l'optimisation fonctionne.

🔹 b) Niveau Gateway

- **Disponibilité de la gateway** (ping, état connexion IP).
- **Nombre de paquets reçus/transmis**.
- **RSSI / SNR moyen par device**.
- **Taux de collisions** (si plusieurs devices émettent simultanément).
- **Charge CPU/RAM** si gateway Linux embarqué.

🔹 c) Niveau Réseau (Network Server)

- **Uptime du LNS (LoRaWAN Network Server)**.
- **Déduplication** (éviter doublons d'une même trame reçue par plusieurs gatew
- **Messages MIC invalides** (intégrité échouée).
- **Nombre de devices connectés / actifs**.
- **Distribution des Spreading Factors (SF7–SF12)**.

🔹 d) Niveau Application Server

- **Latence bout-en-bout** (device → gateway → NS → App Server).
- **Disponibilité des API (MQTT/HTTP)**.
- **QoS per-device** (ex : perte de 10% sur capteur X).
- **Alerting** en cas de seuils dépassés.

3. Outils de Monitoring LoRaWAN

🔹 Solutions open-source

- **ChirpStack**
  - Dashboard intégré pour suivre devices, gateways, uplinks/downlinks.
  - Export métriques vers **Prometheus + Grafana**.
- **The Things Stack (TTN/TTS)**
  - Monitoring en temps réel via console TTN.
  - Intégrations natives (Datadog, AWS IoT, InfluxDB).

🔹 Monitoring industriel

- **Kerlink Wanesy Management Center** (supervision de gateways LoRaWAN).
- **Actility ThingPark** (solution opérateur LoRaWAN).
- **Loriot Network Management** (suivi réseau et devices).

🔹 Intégrations classiques

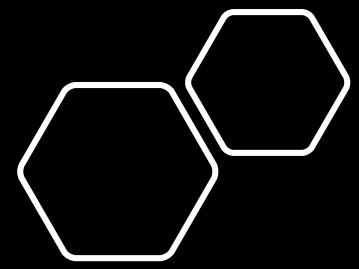
- **Prometheus + Grafana** : métriques de ChirpStack, TTS ou brokers MQTT.
- **ELK Stack (Elasticsearch, Logstash, Kibana)** : analyse logs et corrélation anomalies.
- **Datadog / Zabbix** : supervision classique infra + LoRaWAN metrics.

4. Métriques clés à surveiller

Niveau	Indicateurs	Objectif
Device	Batterie, uplinks, ADR	Autonomie & fiabilité capteurs
Gateway	Uptime, RSSI, SNR, collisions	Santé couverture radio
Réseau	MIC fails, duplication, SF usage	Sécurité & efficacité spectre
Application	Latence E2E, QoS per-device	SLA applicatif

5. Alerting & Automatisation

- **Exemples d'alertes :**
  - Gateway offline > 5 min.
  - Taux d'échec uplink > 20%.
  - Batterie < 10%.
  - Montée brutale du nombre de messages MIC invalides (attaque ou bug device).
- **Automatisation :**
  - Reconfiguration ADR si SF trop haut.
  - Rotation automatique des clés (sécurité).
  - Reboot remote de gateway si surcharge.



---

6. Visualisation typique

- **Dashboard Grafana LoRaWAN :**
  - Map couverture gateways (heatmap).
  - Graphes RSSI/SNR par capteur.
  - Histogramme des Spreading Factors.
  - Courbes autonomie batteries.
  - Taux de pertes uplink/downlink.

7. Tendances

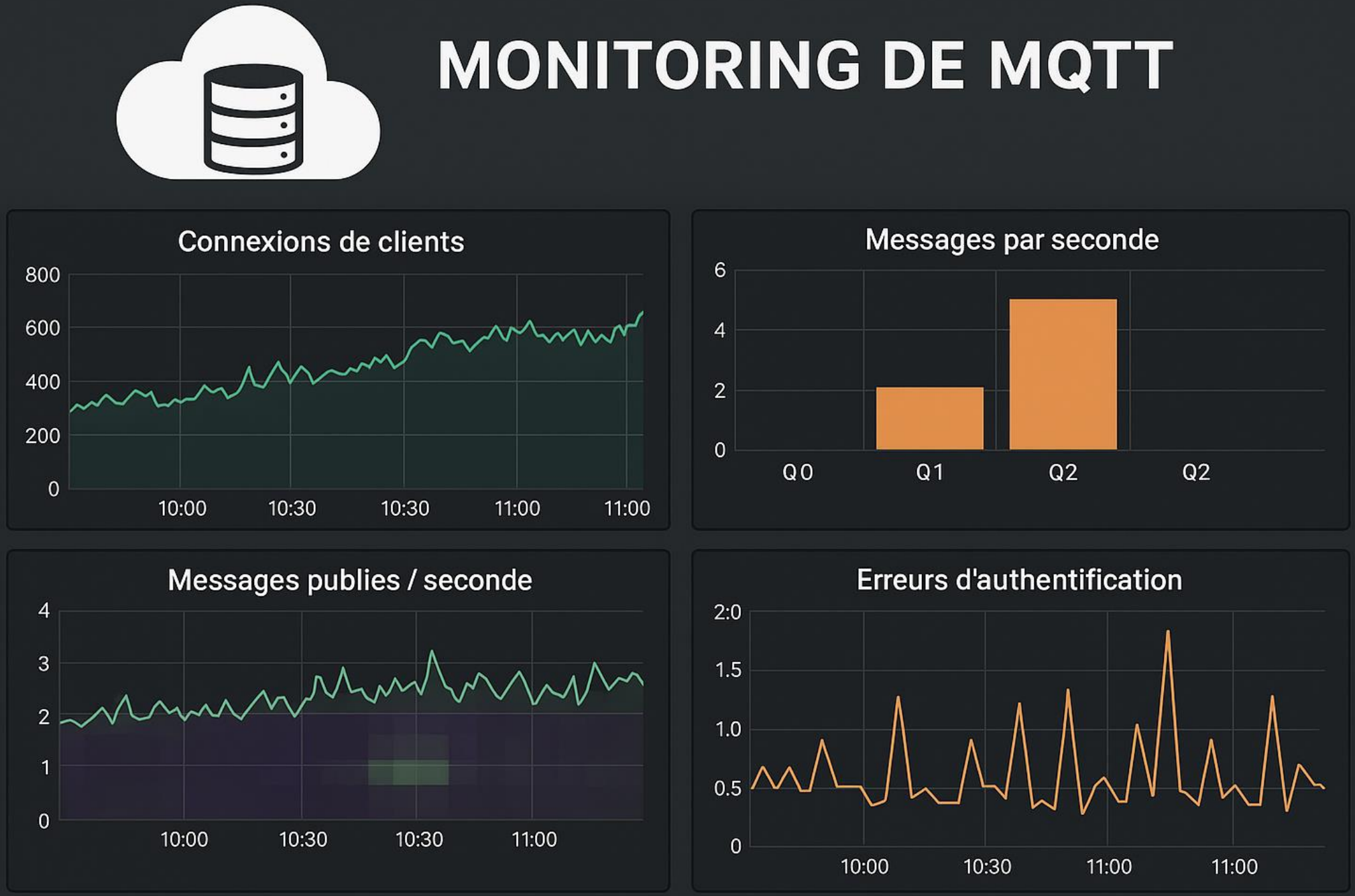
- Monitoring **IA/ML** → détection proactive d'anomalies (ex : prédire panne d'une gateway).
- Intégration **LoRaWAN + 5G monitoring** (réseaux hybrides).
- **Monitoring distribué multi-cloud** (Azure IoT Hub, AWS IoT Core avec LoRaWAN).
- Solutions **Digital Twin** pour visualiser santé réseau et devices en temps réel.

✅ En résumé : le **monitoring LoRaWAN** se fait à 4 niveaux (Device, Gateway, Réseau, Application), avec des outils comme **ChirpStack + Prometheus/Grafana** ou **TTN/TTS** côté open-source, et **Actility / Kerlink** côté industriel.

MQTT

IDEO·LAB

MONITORING DE MQTT



1. Pourquoi monitorer MQTT ?

MQTT est un protocole léger, mais il est souvent utilisé pour des systèmes critiques (IoT, industrie, finance).

Le monitoring permet de :

- Assurer la **stabilité du broker** (point central).
- Vérifier la **qualité de service (QoS)** des messages.
- Détecter des **problèmes de connectivité ou surcharge**.
- Surveiller la **sécurité** (authentification, attaques DDoS, connexions suspectes).
- Garantir le **SLA applicatif** (taux de livraison, latence).

🔹 a) Niveau Clients (publishers/subscribers)

- Nombre de connexions actives.
- Taux de **connexion/déconnexion** (stabilité réseau).
- Messages envoyés/reçus par client.
- Respect du **QoS** (0,1,2).
- Détection de clients inactifs (KeepAlive dépassé).

🔹 b) Niveau Broker MQTT

- **Charge CPU/RAM** du broker.
- Nombre de **topics actifs**.
- Débit de messages (pub/sub par seconde).
- Latence moyenne de distribution.
- Taille des files de messages en attente (surtout QoS 1 & 2).
- Sessions persistantes stockées.
- **Retained messages** : suivi de leur croissance.

🔹 c) Niveau Réseau

- Débit montant/descendant (network throughput).
- Latence moyenne entre client et broker.
- Taux de **perte de paquets** TCP/TLS.
- Saturation des ports 1883/8883.

🔹 d) Niveau Sécurité

- Tentatives d'authentification échouées.
- Clients non autorisés sur certains topics.
- Anomalies TLS (certificat expiré, handshake failed).
- Détection de scans massifs ou DoS (exemple : attaques sur Mosquitto mal sécurisé).

3. Métriques clés à surveiller

Niveau	Indicateurs	Objectif
Client	Connexions, QoS, keepalive	Santé device / app
Broker	Messages/s, topics actifs, file attente	Performance & scalabilité
Réseau	Latence, pertes paquets	Fiabilité transport
Sécurité	Auth fail, certs, attaques	Protection & conformité

4. Outils de monitoring MQTT

🔹 Brokers MQTT avec monitoring natif

- **EMQX** : dashboard intégré + export Prometheus.
- **HiveMQ** : console avancée + extensions Grafana/Kibana.
- **Mosquitto** : basique, mais logs exploitables + plugin Prometheus.
- **RabbitMQ (plugin MQTT)** : monitoring via **RabbitMQ Management UI**.

🔹 Solutions open-source

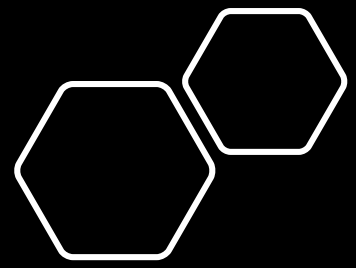
- **Prometheus + Grafana** → exporter métriques broker (EMQX, HiveMQ, Mosquitto).
- **ELK Stack (Elasticsearch + Kibana)** → analyse des logs MQTT.
- **Telegraf + InfluxDB + Grafana (TIG stack)** → suivi métriques MQTT.

🔹 Monitoring applicatif

- **Datadog / New Relic** : intégration MQTT via plugins.
- **Zabbix / Nagios** : supervision basique (ping broker, latence, uptime).

5. Alerting & Automatisation

- **Exemples d'alertes** :
  - Déconnexion d'un client critique.
  - Latence > 200 ms pour livraison QoS 1.
  - Nombre de retained messages > seuil.
  - Authentification échouée multiple → suspicion brute force.
  - Broker CPU > 80% ou mémoire saturée.
- **Automatisation possible** :
  - Redémarrage automatique du broker si surcharge.
  - Rotation des certificats TLS expirants.
  - Auto-scale de brokers en cluster (EMQX/HiveMQ en Kubernetes).



---

6. Dashboards typiques (Grafana/Datadog)

- Nombre de clients connectés (courbe dans le temps).
- Messages publiés / seconde.
- Distribution par QoS (0/1/2).
- Heatmap des topics actifs (volumes par thème).
- Latence moyenne des messages.
- Taux d'échecs d'authentification.
- Charge cluster (si plusieurs brokers).

7. Tendances

- Monitoring MQTT en **Kubernetes** (Helm Charts + Prometheus Operator).
- **IA / Machine Learning** pour prédire surcharge ou déconnexion massive.
- Sécurité renforcée → intégration **SIEM (Splunk, Sentinel)** pour logs MQTT.
- Suivi E2E **(device → broker → app)** via *tracing distribué* (OpenTelemetry).

✅ En résumé :

Le **monitoring MQTT** repose sur **3 couches** :

1. **Clients** (connexions, QoS, stabilité).
2. **Broker** (charge, latence, sessions, topics).
3. **Sécurité & Réseau** (TLS, auth, attaques).

TUNING TOOLS

IDEO·LAB

1. MQTT – Monitoring & Tuning

🔹 Monitoring

- **Brokers natifs** :
  - **Mosquitto** : logs basiques + plugin Prometheus.
  - **EMQX** : dashboard complet (métriques en temps réel, connexions, QoS).
  - **HiveMQ** : console avancée, cluster monitoring, extensions Grafana.
  - **RabbitMQ (plugin MQTT)** : RabbitMQ Management UI + métriques HTTP API.
- **Intégrations externes** :
  - **Prometheus + Grafana** → collecte métriques (connexions, latence, QoS, topics).
  - **ELK Stack (Elasticsearch/Kibana)** → analyse logs.
  - **Datadog / Zabbix / Nagios** → monitoring infra + protocoles.

🔹 Tuning

- Optimisation des **QoS** :
  - QoS 0 → haute fréquence, faible importance.
  - QoS 2 → peu de messages, haute criticité.
- **Retained Messages** → utiliser avec parcimonie (éviter surcharge mémoire).
- **KeepAlive** : ajuster pour éviter déconnexions inutiles.
- **Cluster & Load balancing** : scaling horizontal (HiveMQ, EMQX).
- **Sécurité** : TLS obligatoire + ACL pour limiter l'accès aux topics.

2. IoT – Monitoring & Tuning

🔹 Monitoring

- **Devices IoT** :
  - Batterie, uptime, fréquence d'émission.
  - Logs embarqués via **Edge Agents** (Telegraf, Node-RED).
- **Gateways/Edge** :
  - CPU/RAM, taux de paquets forwardés, disponibilité réseau.
  - Outils : **BalenaCloud, KubeEdge, Azure IoT Edge**.
- **Cloud/Plateformes IoT** :
  - **AWS IoT Core, Azure IoT Hub, Google IoT Core (legacy)** → dashboards natifs.
  - **Open-source** : ThingsBoard, Kaa IoT, Eclipse Kapua.

🔹 Tuning

- **Protocoles adaptés** : MQTT/CoAP pour devices contraints, HTTP/AMQP pour cloud.
- **Optimisation énergétique** : duty cycle, modes sommeil, compression payload.
- **Edge Computing** : prétraitement local pour limiter trafic Cloud.
- **Device Management** : OTA updates, firmware tuning, rotation clés sécurité.

3. LoRaWAN – Monitoring & Tuning

🔹 Monitoring

- **Network Servers (LNS)** :
  - **ChirpStack** → open-source, intégration Prometheus/Grafana.
  - **The Things Stack (TTN/TTS)** → monitoring via console + intégrations API.
  - **Actility ThingPark, Loriot, Kerlink Wanesy** → solutions opérateurs.
- **Metrics clés** :
  - RSSI, SNR, Spreading Factors (SF7–SF12).
  - Taux d'erreurs MIC (sécurité).
  - Connexions gateways, uplink/downlink latency.
  - Batterie & uplink success rate devices.

🔹 Tuning

- **ADR (Adaptive Data Rate)** : optimiser portée vs consommation.
- **Classes A/B/C** : choisir en fonction de l'usage (autonomie vs disponibilité downlink).
- **Duty Cycle** : ajuster fréquences d'émission (éviter congestion).
- **Channel plan** : configurer correctement selon région (EU868, US915…).
- **Load balancing gateways** : répartir devices pour éviter saturation.

4. Outils transverses (IoT, MQTT, LoRaWAN)

- **Grafana + Prometheus** → standard de visualisation temps réel.
- **ELK (Elastic, Logstash, Kibana)** → analyse logs & recherche anomalies.
- **Datadog / Splunk / New Relic** → SaaS pour supervision avancée.
- **OpenTelemetry** → standard de **tracing distribué** (du capteur → broker → application).
- **SIEM (Security Information & Event Management)** → Splunk, Sentinel pour sécurité IoT.

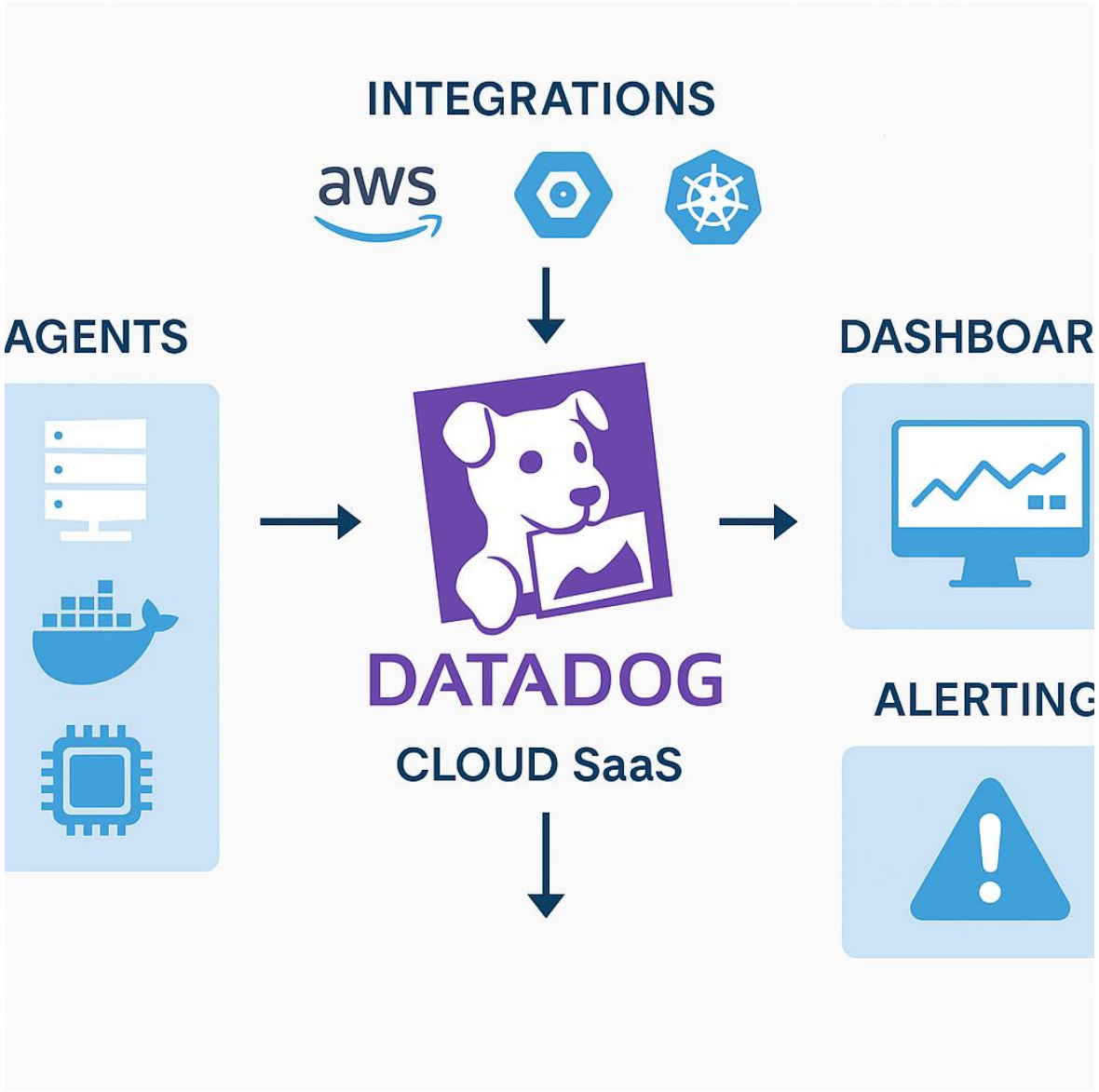
PRINCIPAUX OUTILS

- DataDog

- Grafana

DataDog

IDEO·LAB


INTEGRATIONS
aws
AGENTS
DASHBOAR
DATADOG
CLOUD SaaS
ALERTING

📄 Fiche Technique – Datadog

---

1. Présentation générale

- **Nom** : Datadog
- **Type** : SaaS de monitoring, observabilité & sécurité
- **Fondation** : 2010, New York (Olivier Pomel & Alexis Lê-Quôc, ex-IBM & Wireless Generation).
- **Cible** : Applications cloud, microservices, IoT, DevOps, infrastructures hybrides.
- **Principe** : Collecte métriques, logs & traces → centralisation dans dashboards → alertes & analyse IA.
- **Clients** : Netflix, Samsung, AirBnB, Orange, Capgemini, Airbus, BNP Paribas...

2. Architecture technique

🔹 a) Agents

- **Datadog Agent** : installé sur serveurs/VM/containers.
- Collecte métriques CPU, RAM, I/O, réseau, logs.
- Extensions (checks) pour services (MySQL, Kafka, Redis, MQTT, RabbitMQ…).

🔹 b) Collecte & intégrations

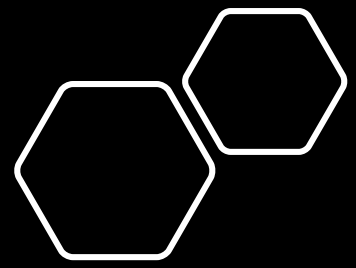
- **+600 intégrations natives** (AWS, Azure, GCP, Kubernetes, Docker, IoT, DBs, brokers).
- API REST & SDK → envoi métriques custom.
- IoT/Edge : agents embarqués + MQTT/LoRaWAN via bridge API.

🔹 c) Backend SaaS

- Stockage distribué (multi-tenant cloud).
- Analyse Big Data en temps réel (milliards de points de données).

🔹 d) Frontend

- Dashboards personnalisables.
- Visualisation multi-niveaux (infra → app → utilisateur).
- Map réseau, logs en streaming, APM traces.



🔹 Network Monitoring

- Supervision réseau L2-L7.
- Visualisation topologies cloud/hybrides.
- Anomalies trafic, latence, pertes paquets.

🔹 Security Monitoring

- Détection menaces (SIEM intégré).
- Cloud Security Posture Management (CSPM).
- Détection d'attaques (ex : brute force sur MQTT broker).

🔹 IoT Monitoring

- Suivi **MQTT, AMQP, CoAP, LoRaWAN** via intégrations ou API.
- Metrics capteurs (batterie, uplink/downlink).
- Edge agents embarqués (Linux ARM).

3. Modules principaux

🔹 Infrastructure Monitoring

- Serveurs, VMs, containers, Kubernetes, cloud providers.
- Suivi CPU, RAM, disques, réseaux, uptime.

🔹 APM (Application Performance Monitoring)

- Tracing distribué (OpenTelemetry, Jaeger, Zipkin compatible).
- Suivi requêtes web, microservices, API.
- Détection goulots d'étranglement.

🔹 Log Management

- Centralisation logs (app, système, IoT).
- Recherche temps réel (Elastic-like).
- Analyse machine learning pour anomalies.

🔹 Real User Monitoring (RUM)

- Expérience utilisateur (latence, erreurs, UX).
- Support mobile, web, IoT apps.

4. Fonctionnalités clés

- **Dashboards temps réel** (métriques, logs, traces).
- **Alerting avancé** : seuils, prédictions ML, corrélations automatiques.
- **Autodiscovery** : auto-détection de containers/pods/services.
- **Tagging & Contextualisation** : filtres par cluster, service, device, région.
- **Anomaly Detection** : algorithmes statistiques et ML.
- **Synthetic Monitoring** : tests automatisés (API, web, IoT endpoints).

---

5. Sécurité

- Données chiffrées (TLS 1.2, AES-256).
- Conformité : **GDPR, SOC 2, HIPAA, ISO 27001**.
- RBAC (Role-Based Access Control).
- Intégration SIEM / SOAR.

---

6. Scalabilité

- Monitoring **multi-cloud et hybride**.
- Support **millions de métriques/s**.
- Haute dispo (SaaS distribué multi-régions).
- Agents légers (Docker, sidecar K8s).

7. Cas d'usage typiques

- **DevOps** : CI/CD, microservices, Kubernetes, performance API.
- **IoT** : suivi capteurs via MQTT/LoRaWAN, edge devices, passerelles.
- **Finance & banque** : détection anomalies transactions (APM + SIEM).
- **Industrie 4.0** : monitoring SCADA, MES via MQTT/AMQP.
- **Smart Cities** : suivi capteurs IoT (pollution, parking, trafic).

---

8. Intégrations IoT/MQTT

- Connecteurs pour **Mosquitto, EMQX, HiveMQ, RabbitMQ.**
- Support ingestion via **MQTT → Datadog API/Agent.**
- LoRaWAN monitoring → via **ChirpStack / TTN export vers Datadog.**
- Capteurs → Gateway → Datadog agent (export MQTT/HTTP).

9. Avantages

✅ Centralisation logs + métriques + traces.

✅ Très riche en intégrations cloud & IoT.

✅ Alertes intelligentes (IA/ML).

✅ Dashboards temps réel très ergonomiques.

✅ Sécurité intégrée (SIEM).

---

10. Limites

⚠️ Coût élevé pour gros volumes (logs surtout).

⚠️ 100% SaaS (dépendance cloud, peu d'On-Premises).

⚠️ Courbe d'apprentissage pour tuning des alertes.

⚠️ Pas toujours optimal pour **temps réel ultra-critique industriel** (latence SaaS).

11. Tuning / Optimisation

- **Sampling APM** : réduire volume traces stockées.
- **Log retention tuning** : conserver logs critiques, archiver les autres.
- **Métriques custom** → envoyer seulement celles utiles.
- **Filtrage / Tagging** → regrouper par service, cluster, device.
- **Dashboards dynamiques** → adaptés par équipe (DevOps, SecOps, IoT).

---

12. Tendances

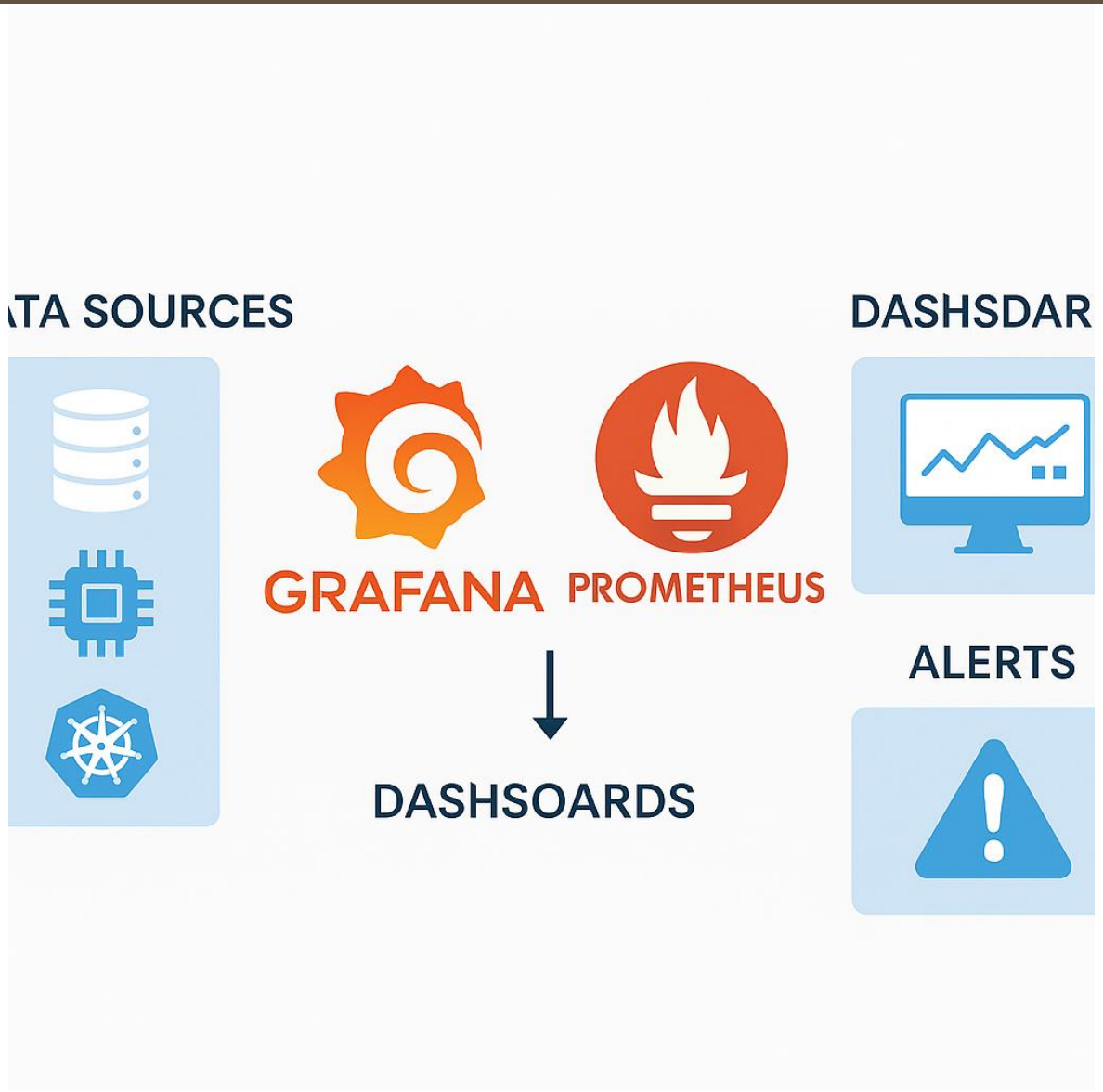
- Intégration **OpenTelemetry** → standard unifié metrics/logs/traces.
- Expansion **IoT monitoring** (Edge + LPWAN).
- **AI Ops** : analyses prédictives et remédiation automatique.
- **Security-First Monitoring** : convergence Observabilité + SIEM + SOAR.

13. Comparaison rapide

Outil	Focus	Points forts
Datadog	Full-stack observabilité	SaaS, +600 intégrations, ML
Grafana/Prometheus	Open-source metrics	Flexible, self-hosted
ELK Stack	Logs & recherche	Big data logs, extensible
Zabbix	Supervision infra	Legacy infra + réseau
Splunk	SIEM/logs	Sécurité, Big Data, $$$

Grafana

IDEO·LAB


ATA SOURCES
GRAFANA
PROMETHEUS
DASHSDAR
ALERTS
DASHSOARDS