Comparaison Plateformes IoT : Cloud vs Edge Management

Réalisé Par Les étudiants :

- *Mohamed Aziz Aydi*
- *Mahdi Boughariou*
- *Taha Chelly*

- *Classe : T-IIF/TD1*

1. Architecture Générale et Philosophie

Le choix architectural repose sur une distinction fondamentale entre les deux plateformes : AWS gère la **donnée** (approche Data-Centric), tandis que balena gère le **cycle de vie du logiciel** sur le matériel (approche Software-Centric).

AWS IoT Core : Le Tissu Événementiel ("Event Fabric")

AWS IoT Core n'est pas un produit monolithique mais un ensemble de services managés "Serverless". L'architecture est conçue pour l'ingestion massive et le routage de messages.

- **Broker MQTT (Géré) :** Support complet de MQTT 5.0 (gestion avancée des codes d'erreur et expiration des messages) et MQTT 3.1.1. Il est conçu pour l'hyperscale, gérant des millions de connexions simultanées avec une réplication automatique sur plusieurs zones de disponibilité.
- **Rules Engine (Moteur de Règles) :** Composant critique permettant de router les données sans provisionner de serveur. Il utilise une syntaxe SQL pour filtrer et envoyer les données vers d'autres services AWS (DynamoDB, Kinesis, Lambda) ou vers des applications externes.
- **Device Shadow :** Un document JSON persistant qui maintient l'état "rapporté" et l'état "désiré" de l'objet, permettant aux applications de communiquer avec des appareils même lorsqu'ils sont hors ligne.
- **Approche Edge :** Repose sur **AWS IoT Greengrass**, un runtime logiciel à installer sur l'OS de l'appareil pour exécuter des fonctions Lambda ou des conteneurs Docker localement.

balenaCloud : L'Infrastructure DevOps Physique

balenaCloud est une plateforme complète de gestion de flotte ("FleetOps") qui intègre le système d'exploitation, le moteur de conteneur et le cloud de gestion.

- **balenaOS :** Un système d'exploitation Linux minimaliste et durci (basé sur Yocto). Sa particularité est son système de fichiers en lecture seule (Read-Only RootFS) qui empêche la corruption des données lors des coupures de courant brutales.
- **balenaEngine :** Un moteur de conteneurs compatible Docker, mais optimisé pour l'IoT (images plus légères, gestion économe de la mémoire et du disque).
- **Le Superviseur :** Un agent autonome présent sur chaque appareil qui gère le téléchargement des mises à jour, l'application des configurations et le maintien de l'état souhaité, assurant le fonctionnement de l'appareil même sans connexion internet.
- **Architecture Réseau :** Utilise un VPN natif pour chaque appareil, permettant un accès SSH et web direct sans configuration complexe de pare-feu.

2. Étapes d'Onboarding et Outils

AWS IoT Core : Identité et Certificats

L'onboarding chez AWS est centré sur la sécurité cryptographique stricte.

- **Méthode :** Chaque objet doit posséder un certificat X.509 unique pour s'authentifier.
- **Fleet Provisioning :** Permet aux appareils de se connecter initialement avec un certificat "Bootstrap" temporaire pour demander et générer automatiquement leurs certificats de production uniques lors de la première connexion.
- **Outils :** Utilisation de l'AWS CLI, des SDKs (Python, C++, Java), et intégration possible avec des composants matériels sécurisés (Secure Elements) pour le stockage des clés.

balenaCloud : Flash & Go

L'onboarding chez balena privilégie la rapidité de déploiement et l'expérience développeur.

- **Méthode :** L'utilisateur télécharge une image OS pré-configurée depuis le tableau de bord. Cette image contient déjà les identifiants de la flotte.
- **Outils :** Logiciel **balenaEtcher** pour flasher les cartes SD ou SSD.
- **Preloading :** Possibilité de "précharger" les conteneurs applicatifs dans l'image OS avant le flashage. L'appareil est fonctionnel dès le premier démarrage sans avoir besoin de télécharger des gigaoctets de données via le réseau.

3. Sécurité : Authentification et Gestion TLS

Fonctionnalité	AWS IoT Core	balenaCloud
Modèle de Sécurité	**Zero Trust / Responsabilité Partagée**. AWS sécurise le cloud, l'utilisateur doit sécuriser l'OS et le matériel.	**Security by Design**. L'OS est verrouillé et géré par balena. La surface d'attaque est minimisée par défaut.
Authentification	**Mutual TLS (mTLS)** strict avec certificats X.509. Le client et le serveur s'authentifient mutuellement.	**Clés API et VPN**. Authentification via tokens API rotatifs et tunnel VPN chiffré (TLS).
Gestion des Rôles	**AWS IAM Policies**. Granularité extrême (ex: droit de publier uniquement sur un topic précis).	**Permissions d'équipe**. Rôles utilisateur (Developer, Operator) et isolation par "Flotte".
Accès Distant	**Secure Tunneling**. Nécessite l'utilisation d'un proxy local et une configuration spécifique côté client.	**Natif**. Accès SSH direct via le dashboard et URL publique sécurisée pour chaque appareil.
Audit et Surveillance	**AWS IoT Device Defender**. Audit des configurations et détection d'anomalies comportementales (IDS) en temps réel.	**Support Access**. Accès temporaire auditable accordé aux ingénieurs balena pour le débogage, sécurisé cryptographiquement.

4. Coûts, Modèle Économique et Limites

AWS IoT Core : Pay-As-You-Go (OPEX Variable)

Facturation à l'usage réel. Très avantageux pour les faibles volumes de données, mais complexe à prédire à grande échelle.

- **Messagerie :** Environ 1,00 $ par million de messages (comptabilisé par bloc de 5 Ko).
- **Connectivité :** Coût faible basé sur le temps de connexion (minutes).
- **Services Annexes :** Coûts additionnels pour l'utilisation du Device Shadow, Registry, Rules Engine et Device Defender.
- **Limites Techniques :**
    - Taille maximale d'un message MQTT : **128 Ko** (ce qui bloque le transfert direct d'images ou d'audio via MQTT).
    - **Point d'attention 2025 :** Fin de vie (**EOL**) de l'interface **Fleet Hub** prévue pour le 18 octobre 2025, forçant les utilisateurs à développer leurs propres interfaces de visualisation.

balenaCloud : Abonnement (OPEX Fixe)

Facturation au nombre d'appareils gérés. Coût prédictible.

- **Plan Gratuit :** Les 10 premiers appareils sont gratuits avec toutes les fonctionnalités incluses.
- **Plan Prototype :** Environ 109 $/mois (incluant 20 appareils). Appareils supplémentaires facturés à l'unité (env. 3 $/mois).
- **Plan Production :** Tarifs dégressifs avec le volume.
- **Avantages :** Inclut l'infrastructure de mise à jour (OTA), le VPN, et le stockage des conteneurs. Pas de coût de "messagerie" interne.
- **Limites :** Moins adapté aux microcontrôleurs (MCU) sans OS (type ESP32 nu), car balena nécessite un processeur capable de faire tourner Linux (Raspberry Pi, Jetson, Gateway).

5. Cas d'Usage Récents (2024-2025)

A. Industrie Automobile : Brembo & AWS (2024)

- **Contexte :** Collecte de données télémétriques à haute fréquence sur les systèmes de freinage pour l'entraînement de modèles d'Intelligence Artificielle.
- **Pourquoi AWS?** Utilisation d'**AWS IoT FleetWise** pour filtrer intelligemment les données à bord du véhicule avant l'envoi. La capacité d'ingérer des pétaoctets de données directement dans un Data Lake S3 pour analyse immédiate était critique. balena aurait été limité pour la partie "Data Pipeline" massive.

B. Déploiement Massif LoRaWAN : Seeed Studio & balena

- **Contexte :** Gestion de 200 000 passerelles LoRaWAN pour l'agriculture et la ville intelligente à travers le monde.
- **Pourquoi balena?** Besoin critique de mises à jour OS fiables sur des connexions cellulaires coûteuses. La technologie de **Deltas Binaires** de balena a permis de ne transférer que les différences de code (quelques Ko au lieu de centaines de Mo), économisant des milliers de dollars en frais 4G et garantissant la stabilité du parc.

C. Edge AI & Robotique : GPU Containerization (2025)

- **Contexte :** Déploiement d'applications complexes (Inférence IA, Vision par ordinateur) sur des modules NVIDIA Jetson à la périphérie.
- **Pourquoi balena?** Capacité unique à conteneuriser des applications nécessitant un accès matériel direct (GPU) via balenaEngine, tout en garantissant que l'OS hôte reste immuable et sécurisé. AWS Greengrass est souvent plus complexe à configurer pour ce niveau d'interaction matérielle bas niveau.

6. Tableau Comparatif Final

Critère	AWS IoT Core	balenaCloud
Philosophie	Gestion de la **Donnée** (Ingestion, Routage)	Gestion du **Logiciel** (OS, Conteneurs, OTA)
Type de Service	Serverless (PaaS)	Container-as-a-Service (CaaS) pour IoT
Cible Matérielle	Tout appareil (du MCU au Serveur) via SDK	Appareils Linux (Raspberry Pi, Gateways, IPC)
Mises à jour (OTA)	Via Jobs (nécessite scripting manuel)	**Natif & Atomique** (Deltas binaires économes)
Sécurité	Authentification mTLS, Politiques IAM fines	OS Read-Only, VPN dédié, Conteneurs isolés
Interface Gestion	Console AWS complexe, **Fin de Fleet Hub en 2025**	Dashboard unifié "Developer Friendly"
Modèle Coût	Variable (au message). Difficile à prévoir.	Fixe (à l'appareil). Simple et prédictible.
Complexité	Élevée (Expertise Cloud Architect requise)	Moyenne (Expertise Docker/Linux requise)
Recommandation	Pour **Big Data**, Télémétrie massive, MCU	Pour **Gateways**, Edge AI, Appareils Linux

Conclusion :
L'approche optimale pour 2025 est souvent hybride : utiliser balenaCloud pour gérer la santé, la sécurité et les mises à jour des appareils (Plan de Contrôle), et utiliser le SDK AWS IoT au sein de l'application balena pour envoyer les données télémétriques vers le cloud (Plan de Données).