



# AWS IoT Core

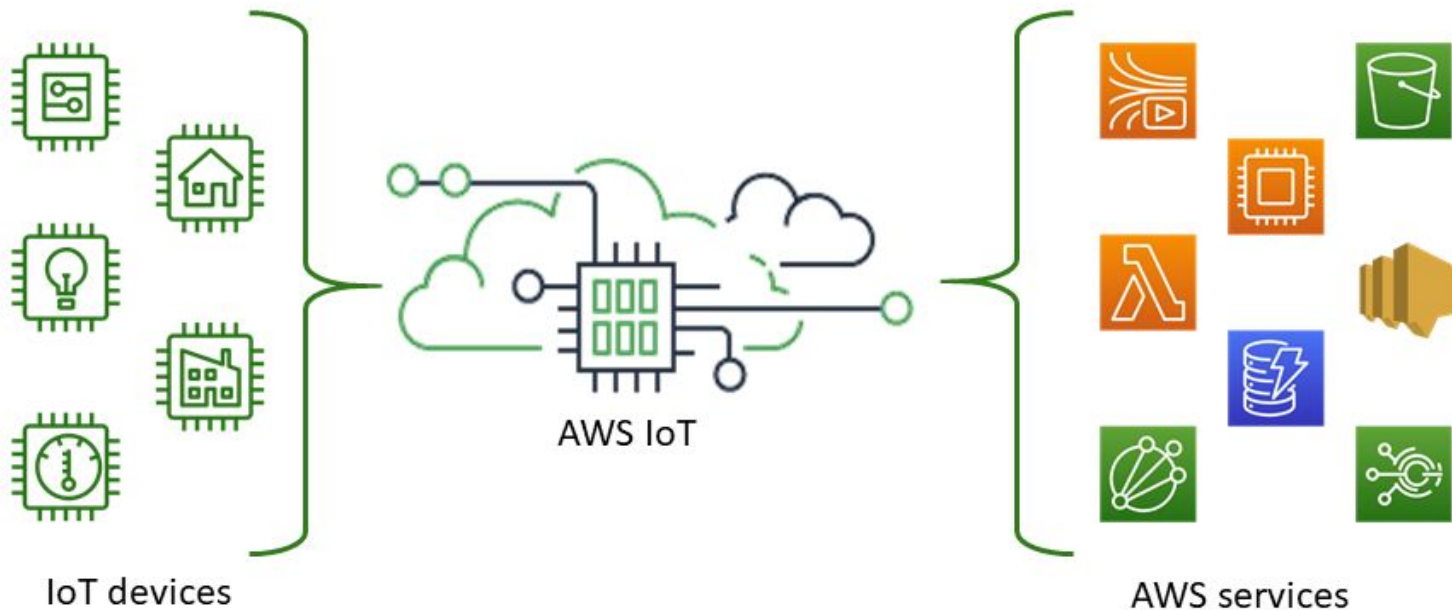
Workshop CLD

# Contenu

- 01 ● AWS Iot Core
- 02 ● Autres services AWS IoT
- 03 ● Protocoles supportés
- 04 ● Stratégie
- 05 ● Analyse de cas
- 06 ● Proof of concept (demo)
- 07 ● Essayez par vous même!

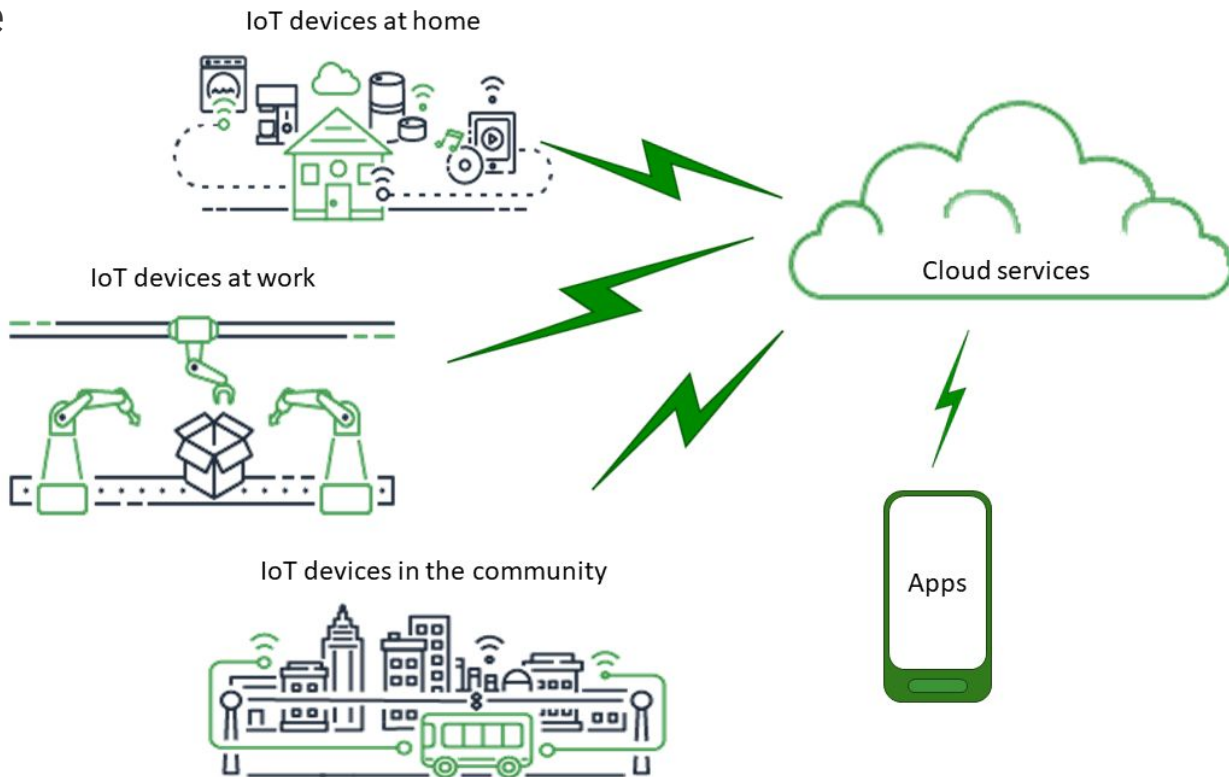
# AWS IoT Core

Cloud pour l'IoT



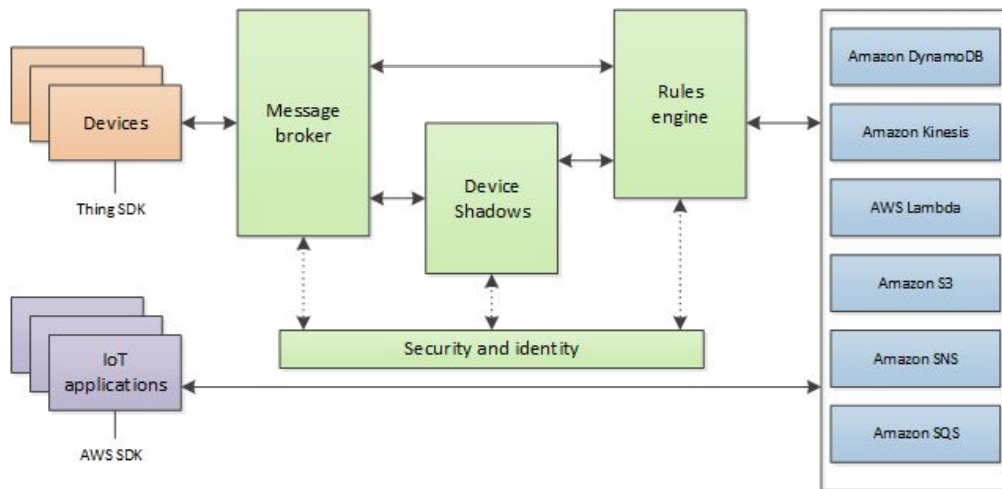
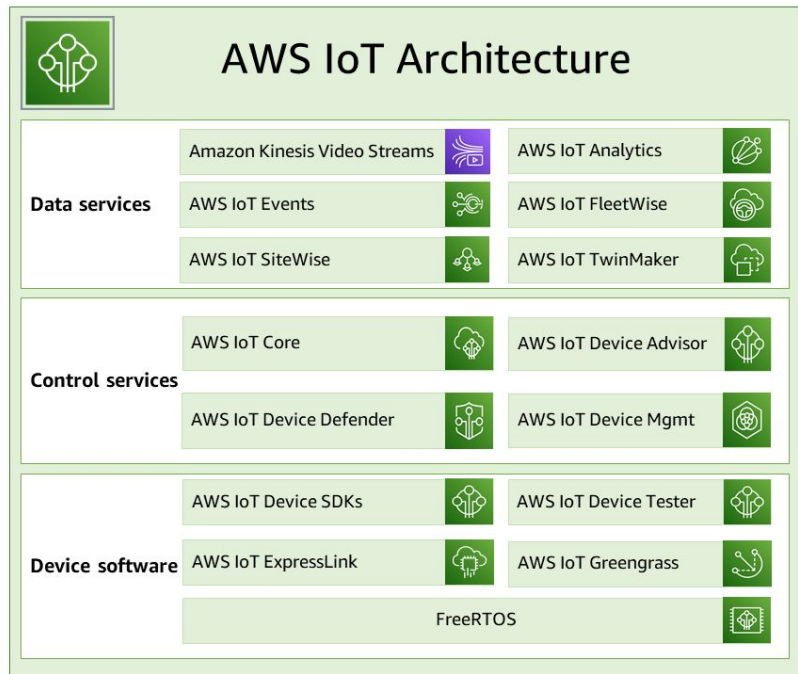
# AWS IoT Core

Écosystème riche



# Autres services AWS IoT

Services fournis par AWS



# Protocoles supportés

AWS IoT Prend en charge 2 principaux protocoles nativement:

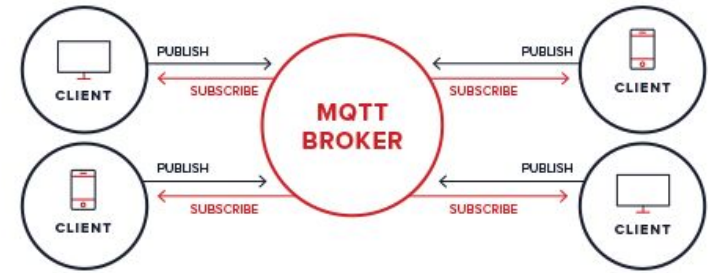
- MQTT (Message Queuing Telemetry Transport)
  - publish/subscribe
- HTTPS
  - Seulement publish



MQTT est préféré pour les applications IoT en raison de sa faible utilisation de la bande passante et de sa capacité à gérer des connexions intermittentes.

# MQTT

Protocole de type publish/subscribe



1. Les **publishers** publient des messages sur un **"Topic"**
2. Les **subscribers** reçoivent ces messages, en s'abonnant au **"Topic"** correspondant.
3. Un **Broker** central s'occupe de la gestion des Topics et de la livraison des messages aux Subscribers.
4. MQTT permet 2 niveaux de QoS: **"at most once"** (best effort) et **"exactly once"**. Cela permet une personnalisation en fonction des besoins de l'application.

# MQTT

## Sécuriser les communications



Pour établir une connexion MQTT sécurisée avec AWS IoT Core:

1. Un appareil utilise son certificat X.509 pour lancer une session TLS.
2. L'appareil est alors reconnu par AWS IoT Core comme une entité authentifiée
3. Ses autorisations (déterminées par sa politique AWS IoT) dictent les actions qu'il peut alors effectuer.



# Stratégie

## Avantages de cette solution

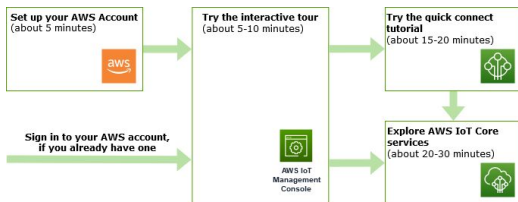
### Cloud

On a tous les avantages du Cloud

- Evolutivité et flexibilité
- Sécurité
- Scalabilité
- etc..

### Rapidité

Premiers résultats en quelques minutes



### AWS

Ecosystème complet

Permet de facilement concilier une solution IoT avec les autres solutions AWS.

### Plus d'infra physique


Plus besoin de ses serveurs

Des soucis en moins...

# Stratégie



## Quand choisir AWS IoT?

- Quand il ne s'agit pas d'une application IoT de hobby / de petite ampleur
  - Votre réseau IoT se développera au fil du temps
  - votre application IoT génère une grande quantité de données qui doivent être stockées et traitées
  - La sécurité est une préoccupation majeure pour votre application IoT
  - Votre application implique la gestion d'un grand nombre d'appareils
  - Vos appareils sont répartis dans différentes régions ou dans des régions isolées avec une faible bande passante
- 

# Analyse de cas

La ville de Lausanne dispose d'une infrastructure "on premise" pour gérer les parkings payants de la ville:

- Gestion des tarifs des parkings souterrains
- Gestion et affichage de la place disponible
- Statistiques basiques



# Analyse de cas

Quels avantages AWS IoT apporterait à la ville de Lausanne?

- Étendre le système à toutes les places de parc de la ville
- Informations sur la disponibilité en temps réel
- Analyse prédictive
- Instaurer des tarifs flexibles
- Une planification urbaine “Data-driven”:
- Intégration dans un environnement cloud riche

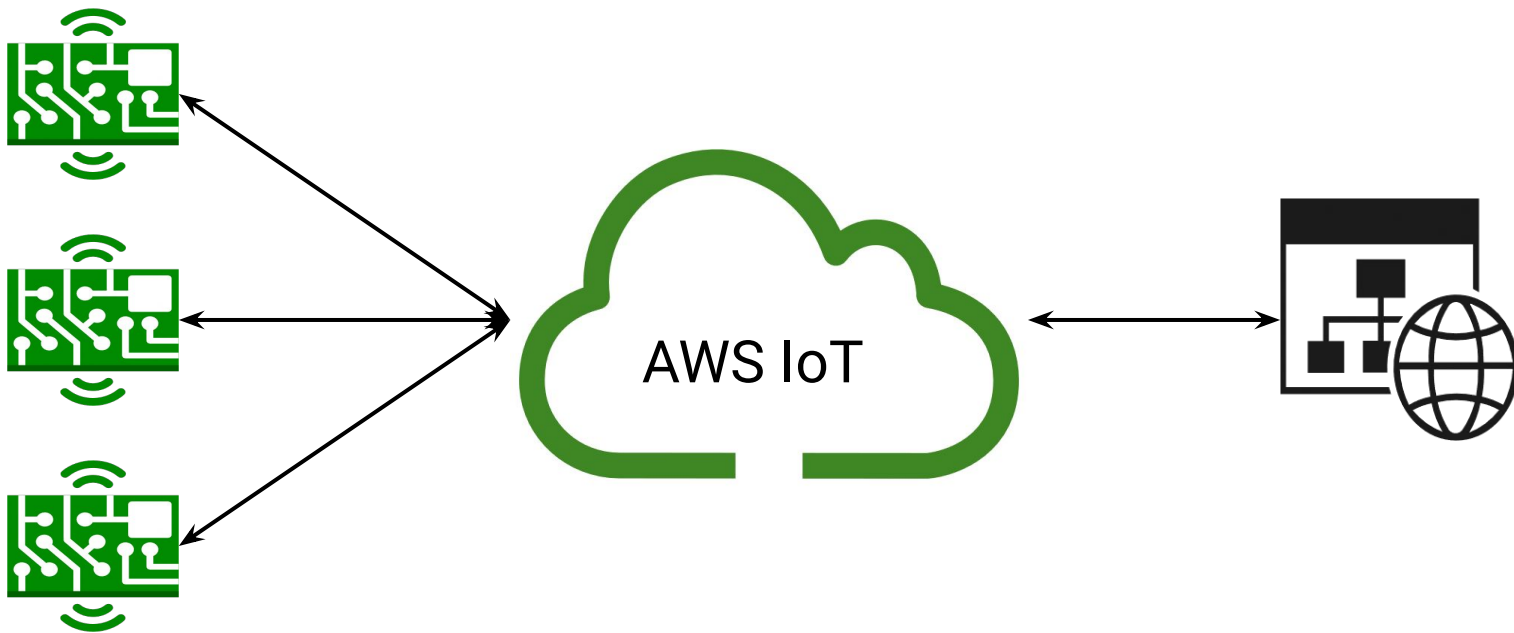
# Analyse de cas

Migrer sur le cloud serait-il rentable?

- Dans le pire des cas AWS IOT ne devrait pas coûter plus de 600.- par mois
  - Si on a 10'000 devices qui travaille 24h/24
  - Qui envoient un message toutes les 5 minutes
- Avec une infrastructure AWS ceci ne devrait pas dépasser les 3500.- par mois
- Sachant qu'il faudrait employer 2 personnes, au moins, juste pour maintenir l'infrastructure autour des serveurs qu'il nous serait nécessaire d'acheter, installer, alimenter, et stocker. On peut voir que cette solution semble économiquement viable

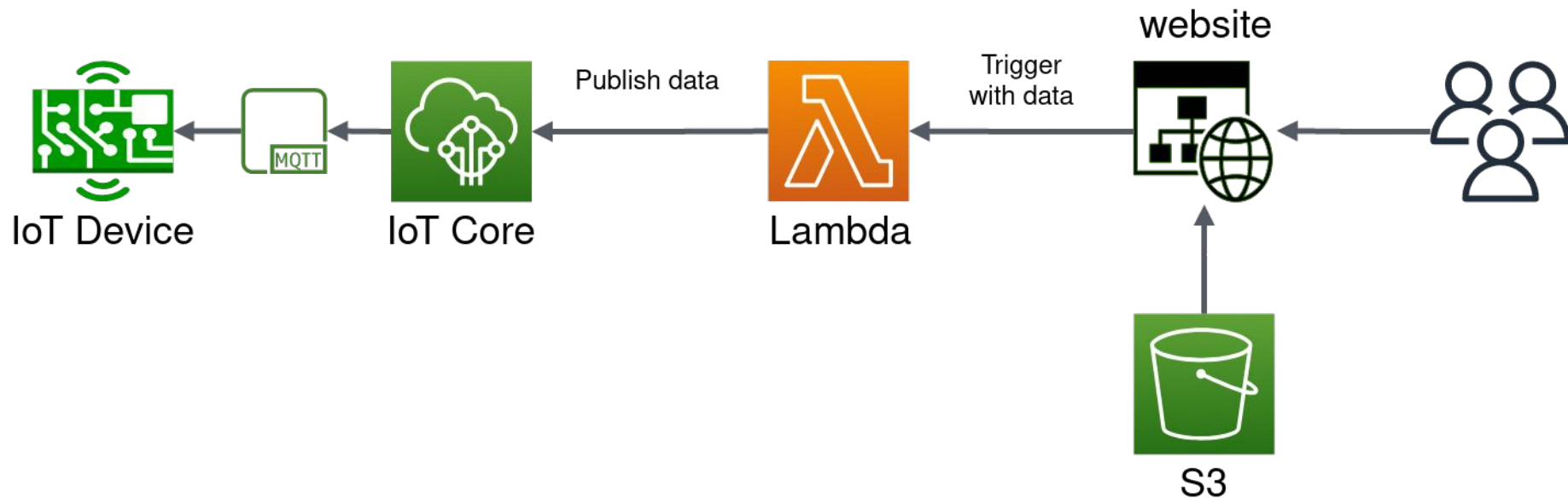
# Proof of concept

Vue d'ensemble



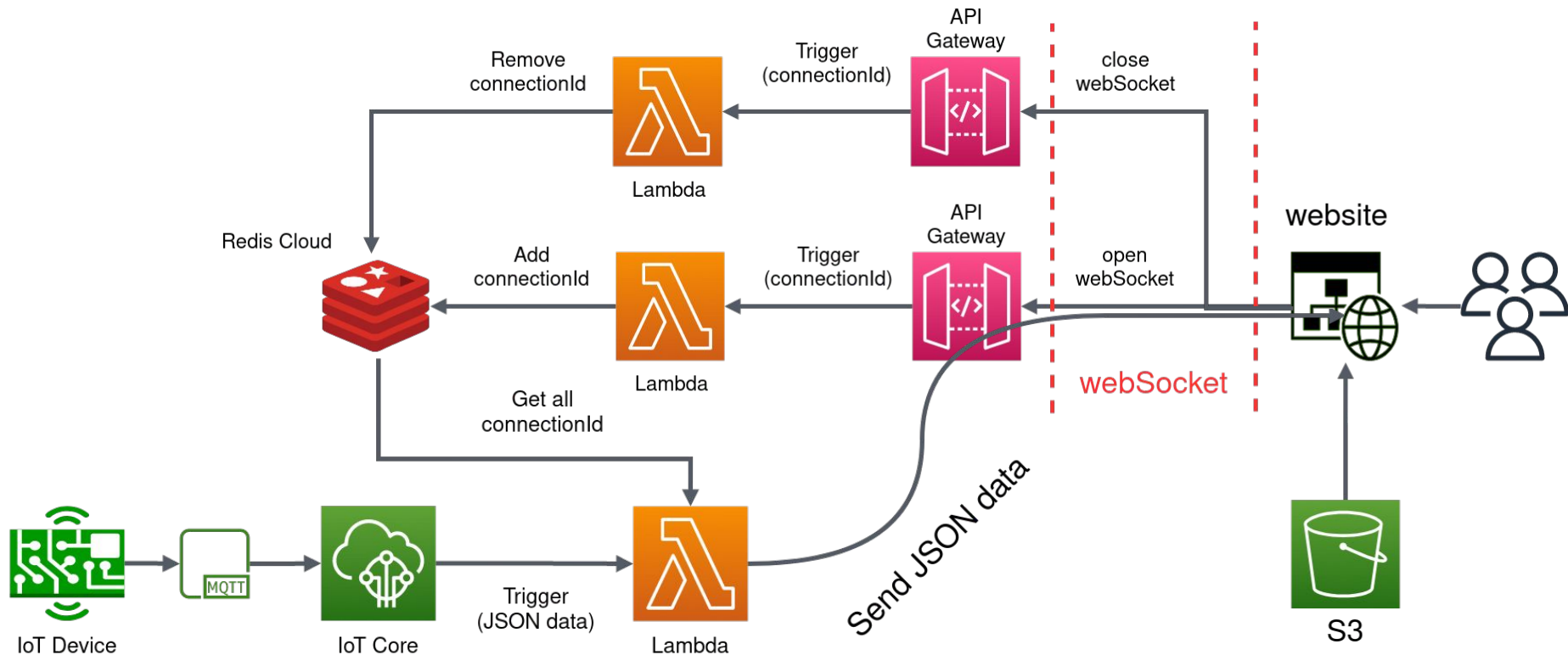
# Proof of concept

Notre infrastructure : Device subscription



# Proof of concept

## Notre infrastructure : Device Publication



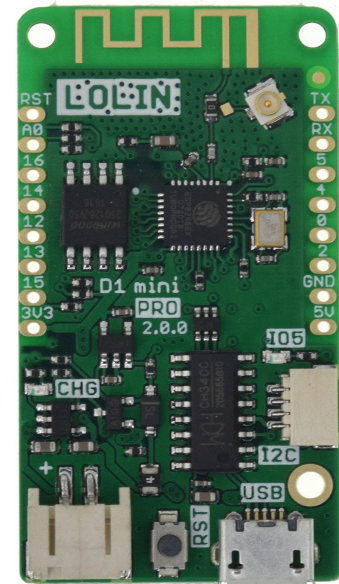
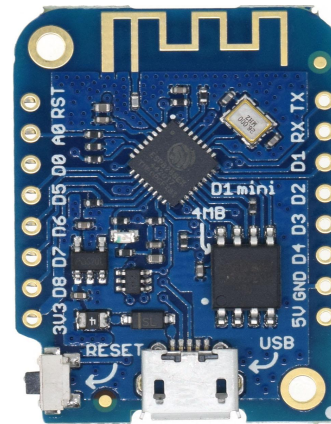


# Proof of concept

## Devices utilisés

### WEMOS D1 mini et D1 mini pro

- Basé sur MCU ESP-8266EX (32 bits)
- 1 coeur à 80 MhZ
- RAM : 112 kB
- WiFi 802.11 b/g/n/e/i
- 5-10 \$ sur aliexpress

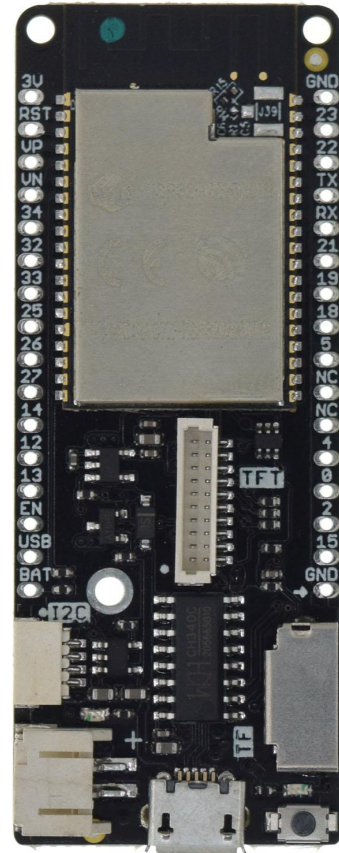


# Proof of concept

## Devices utilisés

### WEMOS D32 pro

- Basé sur MCU ESP32-WROVER (32 bits)
- 2 coeurs à 240 MhZ
- RAM : 520 kB
- WiFi 802.11 b/g/n/e/i
- Bluetooth 4.2 BR/EDR et BLE
- 15 \$ sur aliexpress



# Essayez par vous-même!



<http://workshop-grl-static-website.s3-website-us-east-1.amazonaws.com/>



# Références



Documentation officielle AWS: <https://docs.aws.amazon.com/iot/>

MQTT: <https://mqtt.org/>

Librairies MQTT pour arduino : <https://www.arduino.cc/reference/en/libraries/mqtt/>

Exemple de déploiement ESP32: [Building an AWS IoT Core device using AWS Serverless and an ESP32 | AWS Compute Blog](#)

Notre code source: [https://github.com/DrC0okie/HEIG\\_CLD\\_Workshop](https://github.com/DrC0okie/HEIG_CLD_Workshop)

Appareils utilisés:

WEMOS D32 pro: [https://www.wemos.cc/en/latest/d32/d32\\_pro.html](https://www.wemos.cc/en/latest/d32/d32_pro.html)

WEMOS D1 mini: [https://www.wemos.cc/en/latest/d1/d1\\_mini\\_3.1.0.html](https://www.wemos.cc/en/latest/d1/d1_mini_3.1.0.html)

WEMOS D1 mini pro: [https://www.wemos.cc/en/latest/d1/d1\\_mini\\_pro.html](https://www.wemos.cc/en/latest/d1/d1_mini_pro.html)





# merci!



Pour nous contacter :

Anthony David - [anthony.david@heig-vd.ch](mailto:anthony.david@heig-vd.ch)

Alice Grunder - [alice.grunder@heig-vd.ch](mailto:alice.grunder@heig-vd.ch)

Timothée Van Hove - [anthony.david@heig-vd.ch](mailto:anthony.david@heig-vd.ch)