Alan

- Proxmox
  - 01- Introduction
  - 02- Installation
  - 03- Serveurs applicatifs
  - 04- Utilisateurs
- Diagramme séquence MQTT
- Recette globale de la chaîne IoT

Proxmox

01- Introduction

Proxmox Virtual Environment est un hyperviseur open-source qui combine deux technologies de virtualisation majeures :

- **KVM** (Kernel-based Virtual Machine) pour la **virtualisation** complète de machines virtuelles
- **LXC** (Linux Containers) pour des **conteneurs** système légers

Grâce à cette double approche, **Proxmox** VE permet de déployer un environnement flexible, performant et hautement disponible pour héberger toutes sortes de services, du simple conteneur applicatif à l'infrastructure multi-VM complète.

1. **Debian** est réputée pour sa robustesse et constitue un excellent choix pour des services critiques. Ici, on installera **centreon**, une solution complète de supervision réseau, système et applicative.
2. **Ubuntu Server** 24.04 sert de machine virtuelle dédiée à l'hébergement du serveur web, offrant stabilité et simplicité de gestion. Ici, il servira à héberger un **broker** de messages (ex. : Mosquitto MQTT) ainsi qu'un **serveur** web.

L'utilisation d'une VM dédiée permet **d'isoler** complètement les **services critiques** et d'assurer un meilleur contrôle des performances notamment sur l'aspect **cybersécurité**.

1. Préparation

```
Navigateur ? https://192.168.42.XX:8006
```

```
Login : root@pam / mot_de_passe_proxmox
```

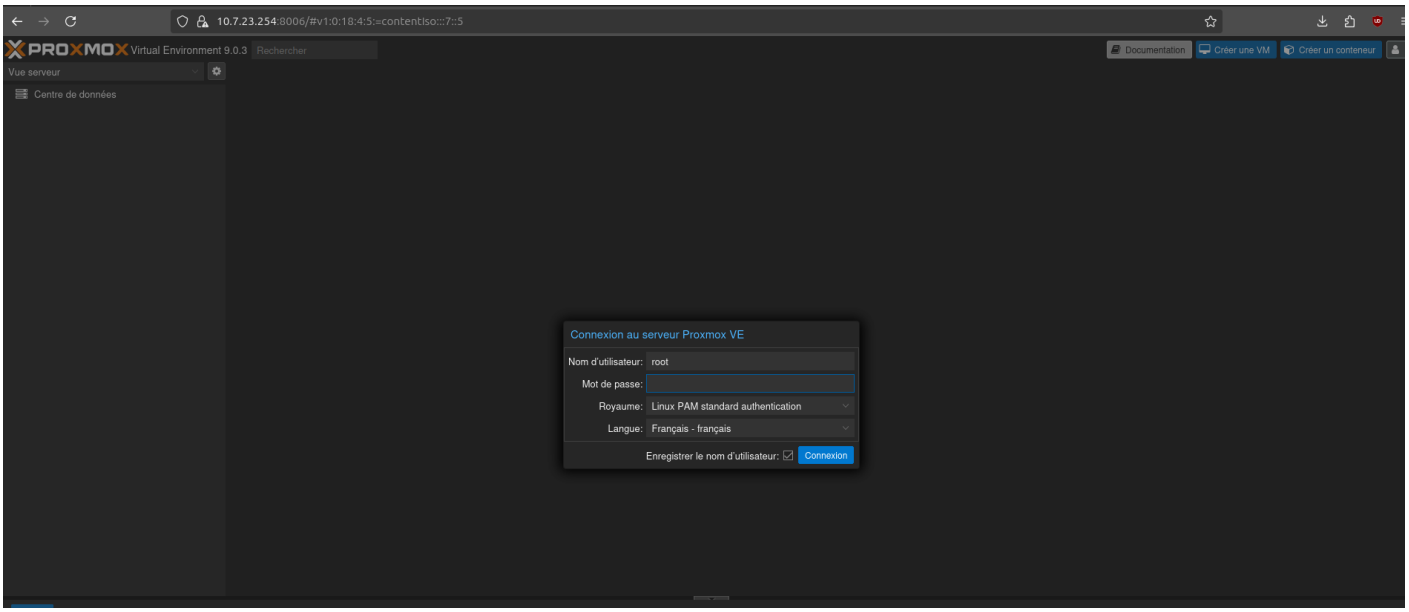

Image 1 : Page d’accueil de l'environnement Proxmox

02- Installation

2.1 Virtualisation

Avant de pouvoir installer une machine virtuelle, nous devons d'abord télécharger le support ISO sur notre **Proxmox** VE

- Allez dans le menu stockage pour identifier l'emplacement où **télécharger** les images **ISO** :

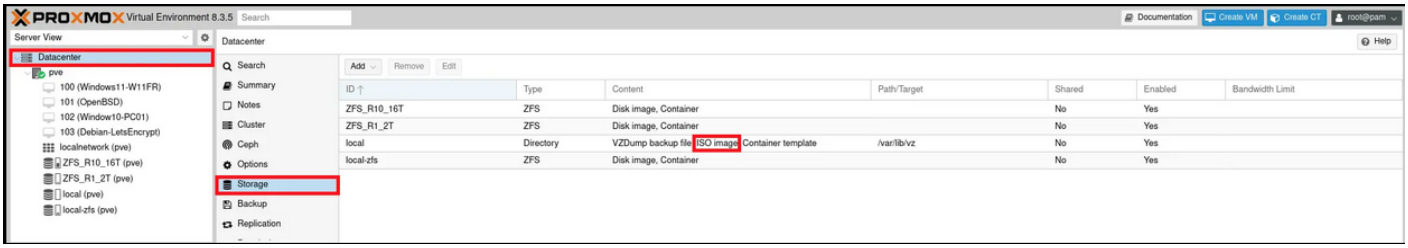

Image 2 : Exemple non contractuel

- Accédez à l'un des **stockages** identifiés précédemment et cliquez sur **Upload** :

Image 3 : Exemple non contractuel

- Téléchargez chaque fichier ISO du système pour lequel vous souhaitez **créer** une machine virtuelle :

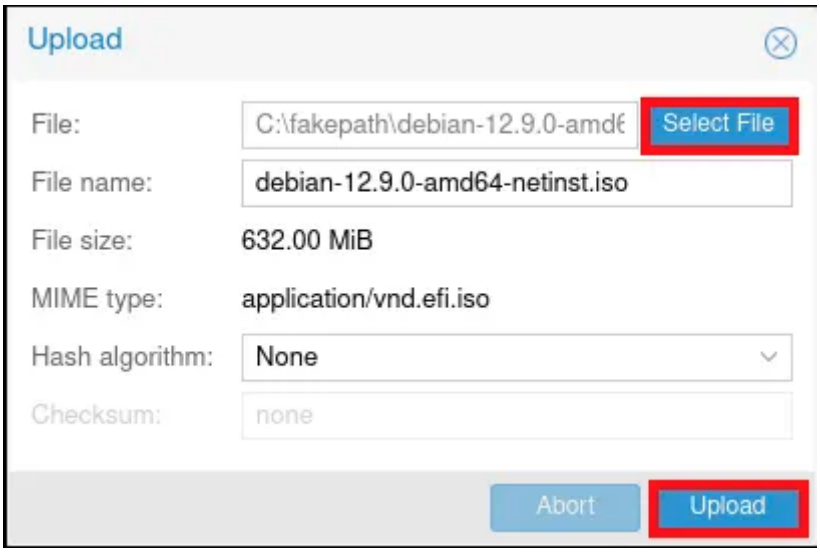

Image 4 : Exemple non contractuel

Les **images ISO** devraient apparaître après leur téléchargement**.**

Assurez-vous de faire la comparaison avec la commande suivante à saisir dans un terminal :

```
sha256sum
```

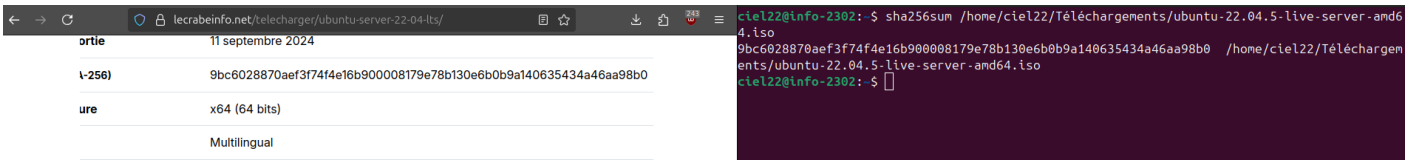

Image 5 : Comparaison pour ubuntu

L’intérêt de cette commande est de comparer **l'empreinte** qu'elle retourne avec celle présente sur le site, ici nous avons exactement le même retour ce qui signifie qu'il n'y a pas eu d’erreur lors de la **transmission**.

2.2 Création d'une Machine Virtuelle

- Configurez les paramètres selon vos besoins, puis cliquez sur Finish

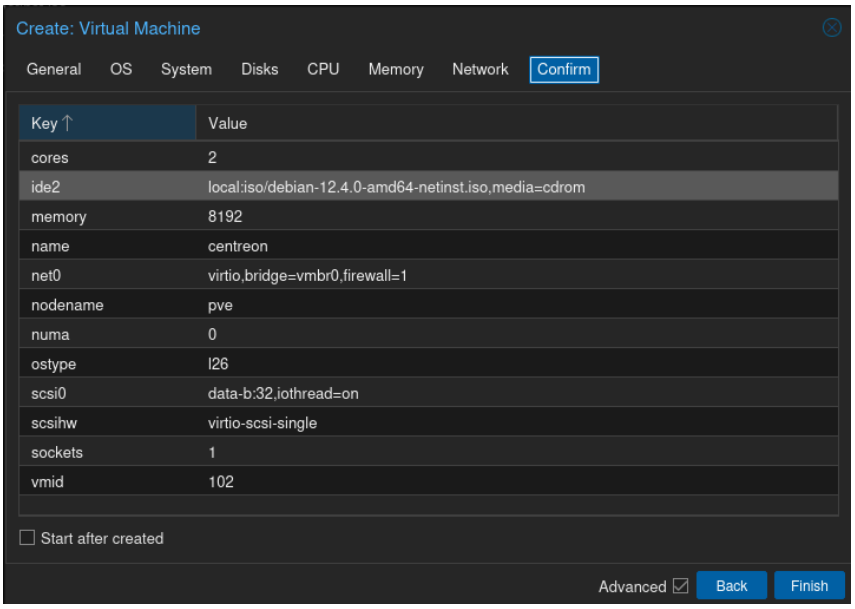

Image 6 : Exemple de paramètres pour une VM

2.3 Installation de Debian

- Une fois créée, démarrez la machine virtuelle depuis le menu **Console**. Elle démarrera automatiquement à partir du média d'installation de **Debian** :

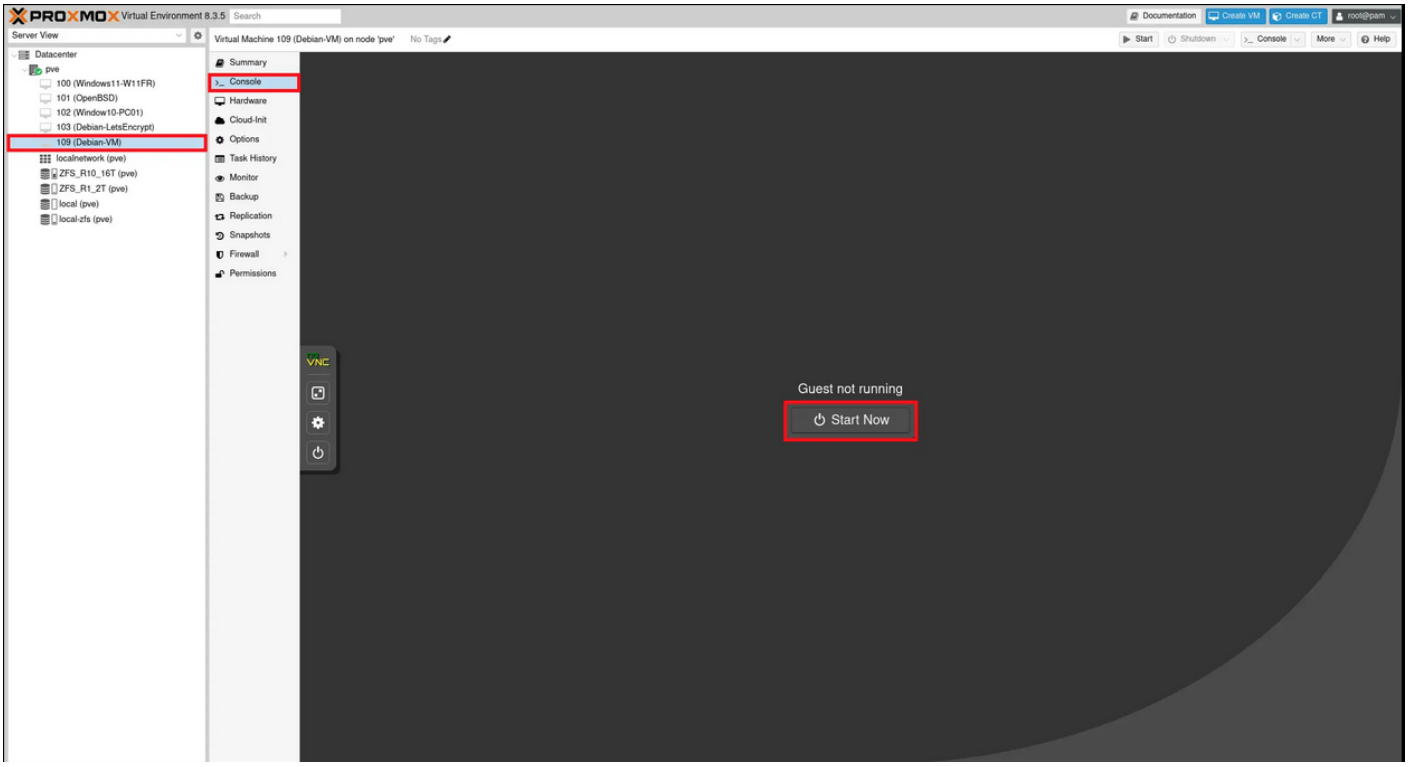

Image 7 : Exemple non contractuel

03- Serveurs applicatifs

Dans le cadre du projet **SIGACS** (Système informatique de gestion automatisée d'un complexe de serres), trois machines virtuelles ont été déployées sur **Proxmox** VE pour gérer la branche applicative du système. Cette architecture est conforme à la répartition des tâches *E1* (hébergement web, base de données, broker MQTT). Les VM sont connectées au réseau 192.168.42.0/24 via le bridge `vmbr0`, intégrées avec le routeur pfSense (192.168.42.55) gérant NTP et la sécurité.

3.1 VM ubuntu-broker-server

Le rôle de la VM est de récupérer des données des capteurs, le broker MQTT central lui reçoit les publications des capteurs.

Paramètre	Valeur
OS	Ubuntu Server 24.04 LTS
CPU	1 vCPU
RAM	8 Go
Disque	32 Go (local-lvm)
Réseau	IP 192.168.42.130/26, bridge vmbr0, gw 192.168.42.129

3.2 VM ubuntu-web-serveur

Le rôle de la VM est d'héberger et de sauvegarder des données. Elle reçoit les mesures MQTT du broker, les stocke en base de données relationnelle, puis les transmets avec un site web.

Paramètre	**Valeur**
OS	Ubuntu Server 24.04 LTS
CPU	1 vCPU

Paramètre	Valeur
RAM	8 Go
Disque	32 Go (local-lvm)
Réseau	IP 192.168.42.131/26, bridge vmbr0, gw 192.168.42.129

3.3. VM debian-centreon

Le rôle de la VM est la génération d'alertes ainsi que de la surveillance au niveau du réseau.

Paramètre	Valeur
OS	Debian 12.1.3
CPU	1 vCPU
RAM	4 Go
Disque	32 Go (local-lvm)
Réseau	IP 192.168.42.132/26, bridge vmbr0, gw 192.168.42.129

04- Utilisateurs

Dans **Proxmox** VE, la gestion des **utilisateurs** permet de contrôler les accès à l'hyperviseur et aux ressources virtualisées. Il est essentiel, dans une logique de **cybersécurité**, de ne pas utiliser le compte `root` pour les opérations courantes, mais de créer des comptes dédiés avec des droits limités au strict nécessaire.

4.1 Création d'un utilisateur

Accédez à `Datacenter` → `Permissions` → `Users`, puis cliquez sur `Add` :

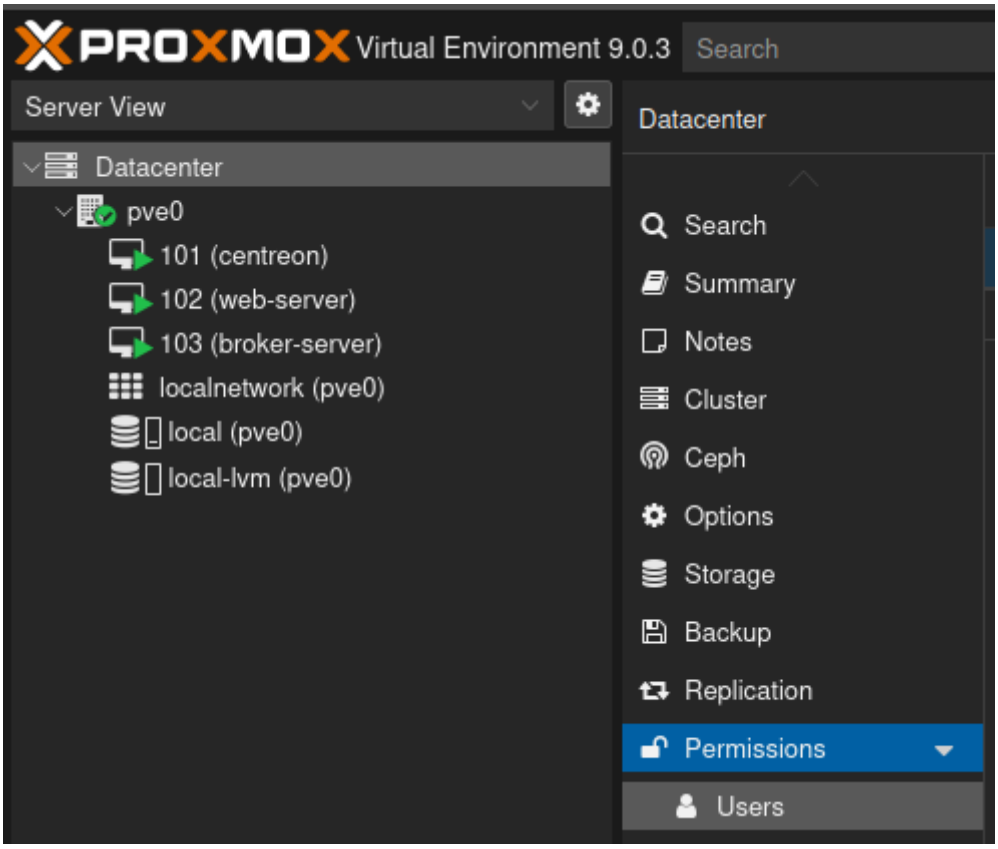

Image 8 : Exemple d'une page d'accueil

Ensuite vous renseignez les champs suivants :

Champ	Valeur exemple
User name	`admin-sigacs`
Realm	`pve`

Champ	Valeur exemple
Password	*(mot de passe fort)*
Group	*(optionnel)*

4.3 Attribution des rôles

Accédez à `Datacenter` → `Permissions` → `Add`, puis cliquez sur `User Permission` :

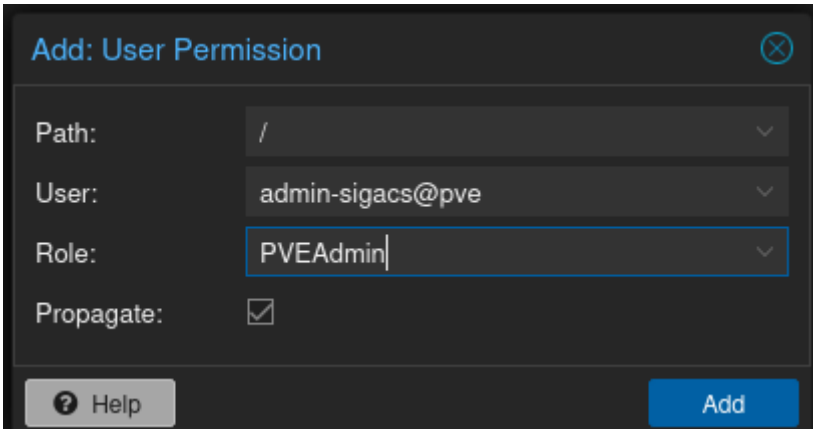

Image 9 : Exemple d'une configuration admin

L’attribution des rôles repose sur le principe du moindre privilège, qui consiste à accorder à chaque utilisateur uniquement les droits nécessaires à ses tâches. Cette approche limite les risques d’erreur, de mauvaise manipulation ou d’accès non autorisé aux ressources de l’infrastructure Proxmox. Dans le cadre du projet SIGACS, elle permet de renforcer la sécurité tout en conservant une gestion claire et adaptée des différents comptes utilisateurs.

Diagramme séquence MQTT

Bienvenue sur le projet SIGACS

Par Alan BANCQUART, Titouan LE GOUALEC, Clément HERVOUET

Diagramme de séquence — Échanges MQTT sécurisés (mTLS)

Projet SIGACS — BTS CIEL — Saint Joseph La Salle — Lorient — 2026

```
sequenceDiagram
    participant M5C as M5StickC<br/>(capteurs bac)
    participant M5S as M5Stack Core2<br/>(concentrateur serre)
    participant BR as Broker <br/>192.168.42.66:8883
    participant DB as Dashboard<br/>(Subscriber)

    %% ?? 1. Connexion WiFi ??
    Note over M5C,BR: 1. Connexion WiFi
    M5C->>M5S: WiFi.begin("Serre")
    M5S->>BR: WiFi.begin("Serre")
    BR-->>M5S: IP attribuée (DHCP)
    BR-->>M5C: IP attribuée (DHCP)

    %% ?? 2. Handshake TLS 1.2 ??
    Note over M5S,BR: 2. Handshake TLS 1.2 (mTLS) — M5Stack ? Broker
    M5S->>BR: Client Hello
    BR-->>M5S: Server Hello + server.crt
    Note over M5S: Vérification server.crt<br/>avec ca.crt embarqué
    M5S->>BR: Certificat client (client.crt)
    Note over BR: Vérification client.crt<br/>avec ca.crt (require_certificate true)
    BR-->>M5S: Tunnel TLS établi

    %% ?? 3. Connexion MQTT ??
```

```
    Note over M5S,BR: 3. Connexion MQTT
    M5S->>BR: CONNECT (clientId, user, password)
    BR-->>M5S: CONNACK (code 0 = accepté)

    %% ?? 4. Abonnement Dashboard ??
    Note over BR,DB: 4. Abonnement subscriber
    DB->>BR: SUBSCRIBE /serre/1/bac/1/#
    BR-->>DB: SUBACK

    %% ?? 5. Publication mesures (boucle 10s) ??
    Note over M5C,DB: 5. Publication mesures (toutes les 15 minutes)
    loop Toutes les 15 minutes
        M5C->>M5S: Mesure sol (MQTT local)<br/>topic: /serre/1/bac/1/sol  payload: "65.0"
        M5C->>M5S: Mesure air/temp<br/>topic: /serre/1/bac/1/air/temp  payload: "22.5"
        M5C->>M5S: Mesure air/hum<br/>topic: /serre/1/bac/1/air/hum  payload: "58.0"
        Note over M5S: Agrégation des mesures<br/>+ affichage écran TFT
        M5S->>BR: PUBLISH QoS 0 — /serre/1/bac/1/sol "65.0"
        BR->>DB: PUBLISH (redistribution)
        M5S->>BR: PUBLISH QoS 0 — /serre/1/bac/1/air/temp "22.5"
        BR->>DB: PUBLISH (redistribution)
        M5S->>BR: PUBLISH QoS 0 — /serre/1/bac/1/air/hum "58.0"
        BR->>DB: PUBLISH (redistribution)
    end

    %% ?? 6. Keep-alive ??
    Note over M5S,BR: 6. Maintien de connexion
    M5S->>BR: PINGREQ (keep-alive)
    BR-->>M5S: PINGRESP
    M5S->>BR: DISCONNECT
```

Légende

Notation	Signification
`??` Flèche pleine	Message envoyé
`- -?` Flèche pointillée	Réponse / accusé de réception
`Note`	Action interne (vérification, calcul)
`loop`	Séquence répétée toutes les 15 minutes

Description des échanges

Phase 1 — Connexion WiFi : le M5StickC et le M5Stack Core2 rejoignent tous deux le réseau "Serre" et obtiennent chacun une adresse IP par DHCP.

Phase 2 — Handshake TLS 1.2 (mTLS) : c'est le **M5Stack Core2** (concentrateur) qui établit le tunnel chiffré avec le broker. Il vérifie l'identité du broker grâce au certificat `server.crt` signé par la CA. Le broker vérifie à son tour l'identité du M5Stack grâce au certificat client `client.crt` (option `require_certificate true`). Aucune donnée ne transite en clair à partir de cette étape.

Phase 3 — Connexion MQTT : une fois le tunnel TLS établi, le M5Stack envoie un message `CONNECT` contenant son identifiant unique, son nom d'utilisateur et son mot de passe. Le broker répond `CONNACK` avec le code 0 (connexion acceptée).

Phase 4 — Abonnement : le Dashboard s'abonne au topic générique `/serre/1/bac/1/#` pour recevoir toutes les mesures du bac 1.

Phase 5 — Publication : toutes les 15 minutes, les M5StickC publient leurs mesures vers le **M5Stack Core2** via MQTT local. Le M5Stack agrège ces données, les affiche sur son écran TFT, puis les transmet au broker Mosquitto qui les redistribue au Dashboard.

Phase 6 — Keep-alive : le M5Stack envoie régulièrement un `PINGREQ` pour maintenir la connexion active avec le broker. En fin de session, il envoie `DISCONNECT` pour clôturer proprement la connexion. Pour voir le répertoire dans sa globalité, voir le fichier [File-Tree.md].

Recette globale de la chaîne IoT

Recette

Objectif

Vérifier que la chaîne IoT fonctionne correctement, depuis la lecture des capteurs jusqu’à l’envoi des données vers le broker MQTT.

Prérequis

- M5StickC alimentées et programmées.
- M5Stack alimentée et connectée au réseau WiFi.
- Capteur DHT22 branché.
- Capteur d’humidité du sol branché.
- Serveur NTP local accessible via pfSense.
- Broker MQTT accessible sur le port 8883.

Vérifications

- Vérifier que le DHT22 remonte des valeurs cohérentes.
- Vérifier que le capteur d’humidité du sol remonte des valeurs cohérentes.
- Vérifier que les M5StickC envoient bien les données à la M5Stack via ESP-NOW.
- Vérifier que la M5Stack se synchronise correctement via le serveur NTP local.
- Vérifier que la M5Stack publie les données vers le broker MQTT.
- Vérifier que les données sont bien reçues côté broker.

Résultat attendu

- La lecture des capteurs est correcte.
- La transmission ESP-NOW fonctionne.
- La synchronisation NTP fonctionne.
- La publication MQTT fonctionne.
- La chaîne globale est opérationnelle.

Réserve

- La déconnexion du capteur d’humidité du sol n’est pas encore correctement gérée par le programme.