**Linux** :

[**https://markdown.data-ensta.fr/s/ypPSr0ZLm#**](https://markdown.data-ensta.fr/s/ypPSr0ZLm) **(bonne explication linux)**

* + Ubuntu, distribution la plus répandu, basée sur Debian donc relativement similaire (aide accessible, stable)
  + Linux mint, plus orienté bureautique

**Le choix de prendre une distribution linux vient principalement du fait que Linux est open source,** l’open source, désigne les logiciels soumis à des licences qui stipulent que le code de ces derniers doit être accessible à tous, **c’est donc gratuit**.

* **Système d'exploitation** :
  + **Debian** (Debian12, que je connais et que j’ai eu l’habitude d’utiliser pour les différents TP en cours), qui offre un environnement **stable** pour installer les serveurs Apache, MariaDB, PHP et le client MQTT, possibilité d’obtenir de l’aide grâce à la communauté (forum…)
    - [« En bref, Debian c’est le choix de la stabilité éternelle »](https://markdown.data-ensta.fr/s/ypPSr0ZLm)

Bien que la stabilité soit un mot assez relatif pour parler d’une distribution linux, ce qui en ressort c’est : Les mises à jour sont peu fréquentes car une distribution stable ne sera pas mise à niveau tant qu'elle n'aura pas été testée && la stabilité peut aussi faire référence a peu de bugs, plantages… (pour la distribution et les logiciels)

[Reddit](https://www.reddit.com/r/linux4noobs/comments/i91m68/what_do_people_mean_when_they_mention_stability/?tl=fr)

« [Ça conviendra très bien pour des serveurs qui doivent redémarrer une fois tout les cinq ans](https://framacolibri.org/t/la-stabilite-des-distribution-linux/12780)»

Debian **12**??? pourquoi plus que Debian 11 ou autre

Plus de fonctionnalité et d’amélioration car c’est la version la plus récente ?

**MQTT :**

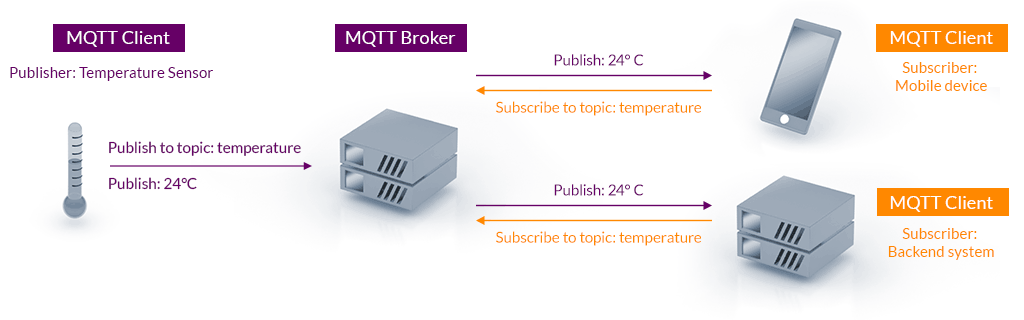
Message Queuing Telemetry Transport est un protocole de messagerie léger adapté aux situations où les clients doivent utiliser peu de code et sont connectés à des réseaux peu fiables ou limités en bande passante.

PRINCIPE de push-subscribe

Client : peut-être soit éditeur, soit abonné, soit les deux

Brockers : Le broker est **le serveur avec lequel les clients communiquent**. Il reçoit les communications qui émanent des clients et les retransmet à d’autres clients.

Un client publie uniquement quand il a des informations à transmettre, et un broker n’envoie des informations aux abonnés que quand il reçoit de nouvelles données.



**Structure** : une structure de type *capteurs/PétroleetGaz/Pression/*permet à un abonné de préciser qu’il souhaite ne recevoir que les données émanant des clients publiant dans la rubrique *Pression*.

si un broker reçoit des données publiées sur une rubrique qui n’existe pas à ce moment-là, celle-ci sera automatiquement créée, après quoi les clients pourront s’y abonner.

Pour alléger au maximum le protocole, les messages reçus ne sont pas conservés sur le broker, sauf à ce qu’ils soient indiqués comme « conservés »

un client ne peut effectuer que quatre opérations au cours de la phase de communication : publication, abonnement, désabonnement ou ping.

**Publication** — envoie un bloc de données contenant le message à envoyer.

**Abonnement** — permets au client de s’abonner à une rubrique

PING – Un client peut envoyer une commande ping vers le serveur. Les commandes ping peuvent servir à vérifier que la connexion est toujours fonctionnelle

DISCONNECT – Un abonné ou éditeur peut envoyer un message DISCONNECT au broker. Ce message informe le broker qu’il n’y a plus lieu qu’il envoie ou mette en file d’attente des messages destinés à un abonné, et qu’il ne recevra plus de données le part d’un éditeur.

Brocker MQTT :

[Mosquitto](https://mosquitto.org) : open-source, léger, relativement simple d’utilisation (apparemment) /bien documenté

EMQX : serverless gratuit/version payante, plus lourd en ressources

hiveMQ : serverless gratuit/version payante

*Je n’ai pas trouvé de différence significative a part le fait que Mosquitto semble fait pour des projet plus simple que les 2 autres et que cela me suffit (qui comportent notamment des versions payante plus avancé)*

Client MQTT (librairies): J’ai l’intention d’utiliser un script php ce qui réduit les possibilité

phpMQTT : simple, basique

Mosquitto-PHP : irait bien avec le brocker avec plus de fonctionnalité (QoS, sécurité…) mais donc plus complexe pour la mise en œuvre

*Je suppose que le meilleur choix serait la librairie Mosquitto-PHP mais le système MQTT étant complètement nouveau pour moi je pense qu’il serait dans un premier temps plus simple et plus sage d’utilisé phpMQTT.*

[Paessler](https://www.paessler.com/fr/it-explained/mqtt#:~:text=Plus%20connu%20sous%20l%27acronyme,ou%20limités%20en%20bande%20passante.)

[MQTT](https://mqtt.org)

Serveur WEB :

**Apache m’est imposer**, en effet Apache est un logiciel de server web qui permet d’établir une connexion entre un serveur et les navigateurs des visiteurs du site web (Firefox, Google Chrome, Safari, etc.) tout en délivrant des fichiers entre eux (structure client-serveur). Le visiteur du site web souhaite charger une page, le navigateur envoie une requête au serveur et Apache renvoie une réponse avec tous les fichiers demandés (texte, images, etc.). Les serveurs web traitent les fichiers écrits dans différents langages de programmation tels que PHP, Python, Java et autres.

Ils les transforment en fichiers HTML statiques et diffusent ces fichiers dans le navigateur des utilisateurs web

Apache est très rependu et fonctionne très bien, relativement simple dans sa configuration. On peut le comparer par exemple a **Nginx** qui est aussi assez populaire car il permet de mieux gérer les trafics élevés, mais est semblerait-il moins simple de prise en main. Ici, de toute façon, apache suffit amplement.

[hostinger](https://www.hostinger.fr/tutoriels/quest-ce-quapache-serveur-web-apache)

**Script intégration des données dans la BDD :**

**Script :**

Python, C/C++ (beaucoup plus compliqué), ***PHP***… DONC pourquoi du php ? *(Tous sont compatibles avec linux, mqtt) simple preference : je connais déjà le langage (utilisation en TP), permet d’éviter de multiplier les environnement (serveur apache pour exécuter le php)*

Lancement du script en boucle pour collecter les données et les intégrer à la BDD (A REVOIR PAS NECESSAIRE ?) :

Systemd : « activer » services en permanences, systemctl pour administrer ces services (arrêter, redémarrer, actif, logs…) redémarrage auto

Supervisor : Permet la même chose, possibilité d’avoir une interface web (facilité rapidité) **mais nécessite une installation**

***CHOIX SYSTEMD car semble assez simple de prise en main pour la mise en œuvre et ne requiers pas d’installation supplémentaire.***

[Systemd](https://documentation.suse.com/en-us/sle-micro/6.0/html/Micro-systemd-basics/index.html), [créer un systemd](https://linuxhandbook.com/create-systemd-services/)

[Supervisor s1](https://supervisord.org/introduction.html), [s2](https://www.elao.com/blog/infra/utiliser-supervisor-pour-controler-ses-services-applicatifs)

**Generation de mail :**

Besoin d’un serveur **SMTP ????**

Utiliser la librairie phpMailer dans le script PHP lors de l’intégration des données dans la base ?

Comparer la valeur à intégrer dans la BDD avec le seuil du capteur qui envoie la donnée dans la BDD ?

**GROS QUESTIONNEMENT POUR LA SECURITE DE TOUTE MON INSTALLATION ? DES CHOSES A FAIRE QUE JE NE CONNAIS PAS… ?**

Logiciel schéma MCD et MLD :

* Looping (utilisé)
  + Pas une appli mais simplement un exécutable donc pratique
  + Permet de réaliser le MLD automatiquement (crow foot)
  + Permet d’obtenir le code SQL de la base (tables, attributs)
  + Assez intuitif
* JMerise
  + Aussi un executable
  + Permet d’obtenir le MLD ni le code SQL
  + Moins intuitif
* Lucidapp
  + Intéressant car c’est en ligne donc pas de logiciel a installé et accessible assez facilement mais ne propose pas de MCD mais uniquement des MLD et avec un choix restreint en version gratuite

**Éditeur/IDE**

* ***Visual Studio Code*** : Gratuit, léger et rapide, avec de nombreuses extensions pour *PHP.*

**NetBeans :** Gratuit, bon support php.

**Choix VS Code, utilisations en TP donc simple préférence**

VIRTUALISATION DES SERVEURS

Les intérêts sont :

* utilisation optimale des ressources d'un parc de machines (répartition des machines virtuelles sur les machines physiques en fonction des charges respectives) ;
* installation, déploiement et migration facile des machines virtuelles d'une machine physique à une autre, notamment dans le contexte d'une mise en production à partir d'un environnement de qualification ou de pré-production, livraison facilitée ;
* économie sur le matériel par mutualisation (consommation électrique, entretien physique, surveillance, support, compatibilité matérielle, etc.)
* installation, tests, développements, cassage et possibilité de recommencer sans casser le système d'exploitation hôte ;
* sécurisation et/ou isolation d'un réseau (cassage des systèmes d'exploitation virtuels, mais pas des systèmes d'exploitation hôtes qui sont invisibles pour l'attaquant, tests d'architectures applicatives et réseau) ;
* isolation des différents utilisateurs simultanés d'une même machine (utilisation de type site central) ;
* allocation dynamique de la puissance de calcul en fonction des besoins de chaque application à un instant donné ;
* diminution des risques liés au dimensionnement des serveurs lors de la définition de l'architecture d'une application, l'ajout de puissance (nouveau serveur, etc.) étant alors transparent.

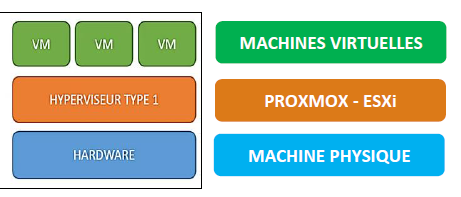
Un [hyperviseur](https://fr.wikipedia.org/wiki/Hyperviseur) de type 1 est comme un noyau système très léger et optimisé pour gérer les accès des noyaux d'OS invités à l'architecture matérielle sous-jacente. Si les OS invités fonctionnent en ayant conscience d'être virtualisés et sont optimisés pour ce fait, on parle alors de [para-virtualisation](https://fr.wikipedia.org/wiki/Paravirtualisation) (méthode indispensable sur Hyper-V de [Microsoft](https://fr.wikipedia.org/wiki/Microsoft) et qui augmente les performances sur [ESX](https://fr.wikipedia.org/wiki/VMware#VMware_ESX) de VMware par exemple). Actuellement[[Quand ?]](https://fr.wikipedia.org/wiki/Aide:Quand) l’hyperviseur est la méthode de virtualisation d'infrastructure la plus performante mais elle a pour inconvénient d’être contraignante et onéreuse, bien que permettant plus de flexibilité dans le cas de la virtualisation d'un [centre de données](https://fr.wikipedia.org/wiki/Centre_de_donn%C3%A9es).

wikipedia

[proxmox](https://www.proxmox.com/en/products/proxmox-virtual-environment/get-started)



* **l’hyperviseur de type 1, ou « *bare metal »*** : il opère directement sur le *hardware* et devient de ce fait l’outil de contrôle du système d’exploitation. Les OS invités s’exécutent alors par-dessus cet hyperviseur. **Proxmox** et ESXi sont 2 **hyperviseurs de type 1** parmi les plus connus.



YA PAS D’INTERFACE GRAPHIQUE DONC SE CO EN SSH DEPUIS UN SHUTTLE DONC CHOISIR UNE IP FIX ET DONNER A M QUERE

***Quel serveur ?***

## Recommended Hardware PROXMOX

* Intel 64 or AMD64 with Intel VT/AMD-V CPU flag.
* Memory, ***minimum 2 GB for OS and Proxmox VE services.*** Plus designated memory for guests. For Ceph or ZFS additional memory is required, approximately 1 GB memory for every TB used storage.
* Fast and redundant storage, best results with SSD disks.
* OS storage: Hardware RAID with batteries protected write cache (“BBU”) or non-RAID with ZFS and SSD cache.
* VM storage: For local storage use a hardware RAID with battery backed write cache (BBU) or non-RAID for ZFS. Neither ZFS nor Ceph are compatible with a hardware RAID controller. Shared and distributed storage is also possible.
* Redundant Gbit NICs, additional NICs depending on the preferred storage technology and cluster setup – 10 Gbit and higher is also supported.
* For PCI(e) passthrough a CPU with VT-d/AMD-d CPU flag is needed.

Recommandations matériel :

**RAM:** 2 GB  
**Processor:** 1 GHz  
**Minimum Disk Space:** 10 GB

a more realistic requirement for modern systems and desktops **is a minimum of 4 GB of RAM**, 25 GB of disk space and a CPU not older than ten years.

FOIS 2 pour les 2 VMs (Apache2, php, client MQTTss && MariaDB)

<https://www.debugpoint.com/debian-12-installation/>

2 + 2 + 2 = 6Go RAM minimum

2 cœurs devraient suffire ?

~50go stockages devraient suffire ?

Dell EMC Poweredge T40

* + - 2 \* 8 go RAM
    - HDD 500go seagate

On est large

Logiciel schéma MCD et MLD :

* Looping (utilisé)
  + Pas une appli mais simplement un exécutable donc pratique
  + Permet de réaliser le MLD automatiquement (crow foot)
  + Permet d’obtenir le code SQL de la base (tables, attributs)
  + Assez intuitif
* JMerise
  + Aussi un executable
  + Permet d’obtenir le MLD ni le code SQL
  + Moins intuitif
* Lucidapp
  + Intéressant car c’est en ligne donc pas de logiciel a installé et accessible assez facilement mais ne propose pas de MCD mais uniquement des MLD et avec un choix restreint en version gratuite