Raspberry Pi Serveur

Installation sur un Raspberry Pi

Prérequis

Raspberry Pi OS

Vous devrez installer la dernière version complète du système d'exploitation Raspberry Pi OS

Utilisez l'application Raspberry Pi Imager

Node.js

Node.js est le framework requis pour exécuter MagicMirror.
L'installation de Node.js est très simple ; suivez simplement le guide officiel ci-dessous :

□□□ nodejs download current

```
# Download and install nvm:
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.3/install.sh | bash

# in lieu of restarting the shell
\. "$HOME/.nvm/nvm.sh"

# Download and install Node.js:
nvm install 25

# Verify the Node.js version:
node -v # Should print "v25.2.1".

# Verify npm version:
npm -v # Should print "11.6.2".
```

Installation manuelle

1. Vérifiez si `Git` est installé sur votre machine en exécutant la commande `git` (l'aide devrait s'afficher), sinon installez-le.

   ```
   sudo apt update
   sudo apt install git -y
   ```

   Vérifier

   ```
   git --version
   ```

   (Cela devrait vous afficher quelque chose comme `git version 2.30.x` ou plus récent).

2. Cloner le dépôt :

   ```
   git clone https://github.com/MagicMirrorOrg/MagicMirror
   ```

3. Accédez au dépôt :

   ```
   cd MagicMirror
   ```

4. Installez l'application : ``

   ```
   node --run install-mm
   ```

5. Faites une copie du fichier de configuration d'exemple :

   ```
   cp config/config.js.sample config/config.js
   ```

6. ## Lancer l'application :

   ```
   node --run start
   ```

Démarrage automatique de votre MagicMirror

Utilisation de PM2

PM2 est un gestionnaire de processus de production pour les applications Node.js, doté d'un équilibreur de charge intégré. Il permet de maintenir les applications actives en permanence, de les recharger sans interruption de service et de simplifier les tâches d'administration système courantes. Dans cet exemple, nous l'utiliserons pour exécuter un script shell.

Install PM2

Installez PM2 à l'aide de NPM :

```
npm install -g pm2
```

Démarrage de PM2 au boot

Pour que PM2 puisse fonctionner correctement lors du (re)démarrage de votre système d'exploitation, il doit être lancé au démarrage. Heureusement, PM2 dispose d'une fonction pratique pour cela.

```
pm2 startup
```

> PM2 va maintenant vous afficher une commande à exécuter.

Créer un script de démarrage MagicMirror²

Pour utiliser PM2 avec MagicMirror², il suffit de créer un simple script shell. Il est préférable de placer ce script en dehors du dossier MagicMirror² afin d'éviter tout problème lors d'une éventuelle mise à jour du miroir.

```
cd ~
nano mm.sh
```

Ajoutez les lignes suivantes :

```
cd ./MagicMirror
DISPLAY=:0 node --run start
```

Enregistrez et fermez le fichier à l'aide des commandes CTRL+O et CTRL+X. Assurez-vous ensuite que le script shell est exécutable en exécutant la commande suivante :

```
chmod +x mm.sh
```

Démarrez votre MagicMirror² avec PM2

Il vous suffit de démarrer votre miroir avec la commande suivante :

```
pm2 start mm.sh
```

Votre miroir devrait maintenant démarrer et apparaître sur votre écran après quelques secondes.

Activer le redémarrage du script MagicMirror²

Pour garantir le redémarrage de MagicMirror² après un redémarrage, vous devez enregistrer l'état actuel de tous les scripts exécutés via PM2. Pour ce faire, exécutez la commande suivante :

```
pm2 save
```

Et voilà ! Votre MagicMirror² devrait maintenant redémarrer après le démarrage et après toute panne.

Controlling your MagicMirror² via PM2

Avec votre MagicMirror fonctionnant via PM2, vous disposez d'outils pratiques :

Redémarrage de votre MagicMirror²

```
pm2 restart mm
```

Arrêter votre MagicMirror²

```
pm2 stop mm
```

Afficher les logs MagicMirror²

```
pm2 logs mm
```

Show the MagicMirror² process information

```
pm2 show mm
```

Rotation de l’écran à 90°

Configurer un Raspberry Pi pour afficher l’écran en **mode portrait**, tourné de **90° vers la droite**, **sans environnement graphique** (console ou mode kiosk type MagicMirror, Immich kiosk, etc.).

La rotation se fait via le fichier de démarrage `cmdline.txt` avec le pilote KMS.

Prérequis

- Raspberry Pi OS récent (avec le pilote vidéo **KMS** activé par défaut).
- Accès au Raspberry Pi en **SSH** ou clavier/écran.
- Un écran branché sur la sortie **HDMI** du Raspberry Pi.

> Dans l’exemple ci-dessous, on utilise la sortie **HDMI-A-1**, qui est la plus courante.

Vérifier le nom de la sortie HDMI

Sur le Raspberry Pi, lancer :

```
kmsprint | grep Connector
```

Exemple de résultat :

```
Connector 0 (33) HDMI-A-1 (connected)
```

- Le nom de la sortie HDMI à utiliser est ici : **HDMI-A-1**.
- Si tu obtiens un autre nom (par ex. `HDMI-A-2`), il faudra l’utiliser à la place.

Dans notre cas, on part sur **HDMI-A-1**.

Éditer le fichier `cmdline.txt`

Sur Raspberry Pi OS **Bookworm** (versions récentes), le fichier se trouve ici :

```
sudo nano /boot/firmware/cmdline.txt
```

> Sur des versions plus anciennes (Bullseye et avant), le chemin peut être : `/boot/cmdline.txt`

Tu vas voir **une seule ligne très longue**.
⚠ **Important : ne surtout pas ajouter de retour à la ligne. Toute la configuration doit rester sur UNE seule ligne.**

Ajoute un paramètre `video=...` contenant la résolution + la rotation.

Par exemple, pour un écran en **1920x1080 à 60 Hz**, tourné de **90° vers la droite** :

```
... quiet splash video=HDMI-A-1:1920x1080M@60,rotate=90
```

Conseils :

- Garde tous les autres paramètres déjà présents (ne supprime rien).
- Si un `video=...` existe déjà, remplace-le par ta version avec `rotate=90` au bon endroit.
- Ne mets pas d'espace dans `video=HDMI-A-1:1920x1080M@60,rotate=90` (tout doit rester collé après le `=`).

Sauvegarder et redémarrer

Dans `nano` :

- Sauvegarder : `Ctrl + O`, puis **Entrée**
- Quitter : `Ctrl + X`

Puis redémarrer le Raspberry Pi :

```
sudo reboot
```

Au redémarrage, l'affichage doit être :

- en **portrait**,
- tourné de **90° vers la droite** (côté droit vers le bas).

Dépannage / ajustements

A. L'écran ne s'affiche pas correctement

Tu peux essayer une autre résolution plus standard :

```
video=HDMI-A-1:1280x720M@60,rotate=90
```

ou

```
video=HDMI-A-1:1024x768M@60,rotate=90
```

B. Retour à l’orientation normale

Pour revenir en mode paysage classique, il suffit de :

- enlever `,rotate=90`, ou
- supprimer entièrement le paramètre `video=HDMI-A-1:...` ajouté.

Puis redémarrer.

Résumé

- **Orientation 90° vers la droite** = `rotate=90`
- Fichier à modifier :
  - `/boot/firmware/cmdline.txt` (Bookworm)
  - ou `/boot/cmdline.txt` (anciennes versions)
- Tout doit tenir sur **une seule ligne**.
- Exemple complet :

```
console=serial0,115200 console=tty1 root=PARTUUID=xxxxx-xx rootfstype=ext4 fsck.repair=yes rootwait quiet splash video=HDMI-A-1:1920x1080M@60,rotate=90
```

Tailscale (Tunnel VPN pour la gestion à distance)

Tailscale est un service VPN qui vous permet d'accéder à vos appareils et applications partout dans le monde, de manière sécurisée et simple. Il établit des connexions point à point chiffrées grâce au protocole open source WireGuard, ce qui signifie que seuls les appareils de votre réseau privé peuvent communiquer entre eux.

Installer Tailscale sur le cadre photo numérique me permet de m'y connecter et de le gérer en toute sécurité.

Vous pouvez l'installer facilement en exécutant la commande suivante :

```
curl -fsSL https://tailscale.com/install.sh | sh
```

Après l'installation, connectez la machine au réseau tailnet.

```
sudo tailscale up
```

Suivre les instructions données pour se connecter au compte maitre.

Installation et configuration de vdirsyncer

Synchroniser automatiquement un calendrier iCloud (CalDAV) vers un fichier `.ics` local, exploitable par le module **calendar** de MagicMirror.

□□ Installation

□ Méthode recommandée (via `apt`)

```
sudo apt update
sudo apt install vdirsyncer -y
```

> ❝ □□ Cela installe la version stable packagée pour Ubuntu (ex. `0.19.2`).

Vérification :

```
vdirsyncer --version
```

Résultat attendu :

```
vdirsyncer, version 0.19.2
```

⚙ Emplacement des fichiers

Fichier de configuration :

```
~/.vdirsyncer/config
```

Dossier de statut :

Il sera automatiquement créé à l'emplacement :

```
~/.vdirsyncer/status/
```

Fichiers `.ics` exportés :

Répertoire cible à créer pour MagicMirror :

```
~/MagicMirror/modules/calendars/
```

---

3. Configuration `~/.vdirsyncer/config`

Voici une version corrigée et adaptée pour Ubuntu :

```
# vdirsyncer configuration for MagicMirror

[general]
status_path = "~/.vdirsyncer/status/"

# --- CALDAV Sync 1 ---
[pair iCloud_to_MagicMirror1]
a = Mirror1
b = iCloud1
collections = ["home"]
metadata = ["displayname", "color"]

[storage Mirror1]
type = singlefile
path = /home/thymon/MagicMirror/modules/calendars/%s.ics

[storage iCloud1]
type = caldav
url = https://caldav.icloud.com/
username = sebort@icloud.com
password = abcd-efgh-ijkl-mnop   # Mot de passe d'application iCloud
read_only = true
item_types = ["VEVENT"]

# --- CALDAV Sync 2 ---
[pair iCloud_to_MagicMirror2]
a = Mirror2
b = iCloud2
collections = ["145c2a93-c79a-4347-8045-86fca9c3d201"]
metadata = ["displayname", "color"]
```

```
[storage Mirror2]
type = singlefile
path = /home/thymon/MagicMirror/modules/calendars/%s.ics

[storage iCloud2]
type = caldav
url = https://caldav.icloud.com/
username = sebou@icloud.com
password = zzzz-yyyy-xxxx-wwww   # Mot de passe d'application iCloud
read_only = true
item_types = ["VEVENT"]
```

□□ Génération du mot de passe d’application iCloud

1. Se connecter à https://appleid.apple.com
2. Section **Sécurité** → *Mots de passe spécifiques d’application* → *Générer un mot de passe...*
3. Donner un nom (ex. *vdirsyncer*)
4. Copier le mot de passe généré (ex. `abcd-efgh-ijkl-mnop`)
5. Le coller dans la configuration (`password = …`)

> ⚠ iCloud **refuse les mots de passe standards** pour CalDAV — ce mot de passe spécifique est obligatoire.

□□ Test de découverte

Découverte des calendriers disponibles :

```
vdirsyncer discover
```

Résultat attendu :

```
Discovering collections for pair iCloud_to_MagicMirror1
Found 1 collection
/home/thymon/MagicMirror/modules/calendars/home.ics
```

> Si tu vois une erreur `401 Unauthorized`, vérifie ton mot de passe d’application iCloud.

□□ Synchronisation manuelle

Pour lancer une synchronisation :

```
vdirsyncer sync
```

Le ou les fichiers `.ics` seront créés dans :

```
~/MagicMirror/modules/calendars/
```

□ Synchronisation automatique avec systemd.timer

Créer le dossier systemd utilisateur :

```
mkdir -p ~/.config/systemd/user
```

Télécharger les unités :

```
curl -o ~/.config/systemd/user/vdirsyncer.service
https://raw.githubusercontent.com/pimutils/vdirsyncer/master/contrib/vdirsyncer.service
curl -o ~/.config/systemd/user/vdirsyncer.timer
https://raw.githubusercontent.com/pimutils/vdirsyncer/master/contrib/vdirsyncer.timer
```

Modifier la fréquence (facultatif) :

```
nano ~/.config/systemd/user/vdirsyncer.timer
```

Par défaut :

```
[Timer]
OnBootSec=5m
OnUnitActiveSec=15m
```

➡ Modifier par exemple pour une synchro toutes les heures :

```
OnUnitActiveSec=1h
```

Activer et démarrer :

```
systemctl --user daemon-reload
systemctl --user enable --now vdirsyncer.timer
```

Vérifier :

```
systemctl --user status vdirsyncer.timer
```

□□ Points importants

Élément	Ancien tuto Raspberry	Adaptation Ubuntu
Utilisateur	`pi`	`thymon`
Chemins	`/home/pi/...`	`/home/thymon/...`
Guillemets dans la config	`"Mirror"`	`Mirror` (sans guillemets)
Dates Python (`datetime.now()`)	Autorisées (ancien vdirsyncer)	□ Non supportées
systemd unit	`/etc/systemd/user`	`~/.config/systemd/user`
Commande	`sudo vdirsyncer`	□ Toujours sans `sudo`
Mot de passe iCloud	Standard	□□ Mot de passe d’application

MMM-AirQuality – Bug fetch failed / ETIMEDOUT

Symptômes

- MagicMirror démarre mais log PM2 affiche :
  - `TypeError: fetch failed`
  - `AggregateError [ETIMEDOUT]` dans `MMM-AirQuality/helper.js`
- `curl https://api.waqi.info/...` fonctionne

Objectif

◫◫ Contourner `fetch` de Node en remplaçant l’appel API par `https.request`.

Étapes rapides

1. **Aller dans le dossier du module**

```
cd ~/MagicMirror/modules/MMM-AirQuality
```

2. **Sauvegarder le helper**

```
cp helper.js helper.js.bak
```

3. **Éditer `helper.js`**

```
nano helper.js
```

4. **Remplacer tout le contenu**

```
/* MagicMirror²
 * Module: MMM-AirQuality
```

```
 *
 * By Christopher Fenner https://github.com/CFenner
 * Patched to use https.request instead of fetch (ETIMEDOUT issue).
 * MIT Licensed.
 */

const https = require("https");
const { URL } = require("url");

module.exports = {
  notifications: {
    DATA: "AIR_QUALITY_DATA",
    DATA_RESPONSE: "AIR_QUALITY_DATA_RESPONSE",
  },

  start: function () {
    console.log("AirQuality helper started ...");
  },

  loadData: function (payload) {
    const self = this;
    const urlString =
`https://${payload.config.apiBase}${payload.config.dataEndpoint}${payload.config.loca
tion}/?token=${payload.config.token}`;

    console.log(`AirQuality-Fetcher (https): ${urlString}`);

    let url;
    try {
      url = new URL(urlString);
    } catch (e) {
      console.error("AirQuality-Fetcher: invalid URL", e);
      self.sendSocketNotification(self.notifications.DATA_RESPONSE, {
        payloadReturn: null,
        status: "ERROR",
        identifier: payload.identifier,
      });
      return;
    }
```

```
const options = {
  hostname: url.hostname,
  port: 443,
  path: url.pathname + url.search,
  method: "GET",
  timeout: 10000, // 10s
};

const req = https.request(options, (res) => {
  let data = "";

  res.on("data", (chunk) => {
    data += chunk;
  });

  res.on("end", () => {
    try {
      const json = JSON.parse(data);
      self.sendSocketNotification(self.notifications.DATA_RESPONSE, {
        payloadReturn: json,
        status: "OK",
        identifier: payload.identifier,
      });
    } catch (err) {
      console.error("AirQuality-Fetcher: JSON parse error", err);
      self.sendSocketNotification(self.notifications.DATA_RESPONSE, {
        payloadReturn: null,
        status: "ERROR",
        identifier: payload.identifier,
      });
    }
  });
});

req.on("error", (err) => {
  console.error("AirQuality-Fetcher: https error", err);
  self.sendSocketNotification(self.notifications.DATA_RESPONSE, {
    payloadReturn: null,
```

```
        status: "ERROR",
        identifier: payload.identifier,
      });
    });

    req.on("timeout", () => {
      console.error("AirQuality-Fetcher: request timeout");
      req.destroy();
      self.sendSocketNotification(self.notifications.DATA_RESPONSE, {
        payloadReturn: null,
        status: "TIMEOUT",
        identifier: payload.identifier,
      });
    });

    req.end();
  },

  socketNotificationReceived: function (notification, payload) {
    switch (notification) {
      case this.notifications.DATA:
        console.log(
          `AirQuality-Fetcher: Loading data of ${payload.config.location} for module
${payload.identifier}`
        );
        this.loadData(payload);
        break;
    }
  },
};
```

5. **Redémarrer MagicMirror**

```
pm2 restart mm
```

6. **Vérifier**
   - Plus d’erreur `fetch failed / ETIMEDOUT`
   - MMM-AirQuality affiche à nouveau les données

À noter

- En cas de mise à jour du module, le patch peut être écrasé → **réappliquer `helper.js` patché**.