BTS Services informatiques aux organisations — SESSION 2026
ANNEXE 7-1-A : Fiche descriptive de réalisation professionnelle (recto)
Épreuve E6 - Administration des systèmes et des réseaux (option SISR) - Coefficient 4

DESCRIPTION D'UNE RÉALISATION PROFESSIONNELLE			**N° réalisation : 2**
Nom, prénom	EMILIEN Noam		
Numéro de candidat	02428526184		
☑ Épreuve ponctuelle ☐ Contrôle en cours de formation		**Date :**	Mai 2026

Contexte de la réalisation professionnelle

Intitulé de la réalisation professionnelle

Administration sécurisée d'un serveur Linux - Gestion des utilisateurs, authentification par clé SSH et supervision du service avec pm2

Période de réalisation :	Mai 2026	**Lieu :**	Domicile
Modalité :		☑ Seul ☐ En équipe	

Compétences travaillées

☐ Concevoir une solution d'infrastructure réseau
☐ Installer, tester et déployer une solution d'infrastructure réseau
☑ Exploiter, dépanner et superviser une solution d'infrastructure réseau

Conditions de réalisation (ressources fournies, résultats attendus)

Ressources fournies :
Raspberry Pi 4 Model B (4 Go RAM)
MacBook Air (macOS Big Sur 11) : poste client administrateur.

Résultats attendus :
Sécurisation complète de l'accès SSH au serveur Raspberry Pi hébergeant le tableau d'affichage handball (projet 1).
Création d'un compte utilisateur limité (arbitre), authentification par clé asymétrique ed25519, désactivation de l'authentification par mot de passe, configuration d'accès simplifiés par alias via ~/.ssh/config, supervision du service Express via pm2 (monitoring, logs, gestion d'incidents), authentification HTTP Basic

Description des ressources documentaires, matérielles et logicielles utilisées

Ressources matérielles :
- Raspberry Pi 4 Model B 4 Go RAM (serveur), MacBook Air macOS Big Sur 11 (client SSH).
- Commutateur réseau, borne WiFi Aruba, câbles Ethernet, adaptateur PoE prêtés par l'UIMM Pôle Formation Poitou-Charentes.

Logicielles :
Raspberry Pi OS 64-bit (Debian Trixie), OpenSSH Server (Pi), OpenSSH Client (Mac), pm2 (gestionnaire de processus Node.js), Node.js 20.19.2.
Utilitaires Linux : adduser, groups, nano, cat, chmod, systemctl, ssh-keygen, ssh-copy-id.

Documentaires :

- https://www.raspberrypi.com/documentation/
- https://nodejs.org/en/docs
- https://expressjs.com/fr/4x/api.html
- https://developer.mozilla.org/fr/docs/Web/API/Fetch_API
- https://pm2.keymetrics.io/docs/usage/quick-start/

Modalités d'accès aux productions et à leur documentation

[Documentation technique complète : **PDF joint au dossier numérique**.]

Démonstration en direct sur le matériel physique lors de l'épreuve :

Pour accéder au serveur, il est nécessaire d'être connecté au réseau WiFi privé **HANDBALL** (mot de passe : 1erOJO2k28)

Connexion SSH sécurisée par clé (depuis le mac) :
ssh arbitre-handball (compte avec droits limités, via alias ~/.ssh/config)
ssh pi-admin (compte admin, via alias aussi ~/.ssh/config)
Mot de passe pour sudo (compte admin pi) : 1erOJO2k28

Paire de Clés créée dans le dossier ~/.ssh/ sur le client (Ordi Mac) avec la commande :
ssh-keygen -t ed25519 “cle_arbitre”
Clé privée reste sur le poste du client et la clé publique est partagée au serveur dans son fichier /home/pi/.ssh/authorized_keys pour le compte pi et /home/arbitre/.ssh/authorized_keys pour le compte arbitre.
Le serveur n’autorise la connexion SSH qu’à celui faisant une requête tout en possédant la clé privée correspondante à la clé publique qu’il possède.

Fichiers de configuration consultables sur le Pi :
/etc/ssh/sshd_config (PasswordAuthentication no, PubkeyAuthentication yes)
/home/arbitre/.ssh/authorized_keys (clé publique ed25519)
/home/pi/.ssh/authorized_keys (clé publique ed25519)

Accès à l'interface de contrôle : http://192.168.10.34:3000/control.html
→ Le navigateur déclenche automatiquement une fenêtre d'authentification HTTP Basic.
Identifiant : arbitre / **Mot de passe :** handball86

→ Identifiants corrects : Accès accordé : interface arbitre (scores, chronomètre, pénalités).
→ Identifiants incorrects : Accès refusé, popup réaffichée.

BTS Services informatiques aux organisations — SESSION 2026
ANNEXE 7-1-A : Fiche descriptive de réalisation professionnelle (verso)
Épreuve E6 - Administration des systèmes et des réseaux (option SISR) - Coefficient 4

Descriptif de la réalisation professionnelle, y compris les productions réalisées et schémas explicatifs

1. Contexte et problématique

Le Raspberry Pi 4 héberge le serveur web Express du tableau d'affichage handball (projet 1). L'accès SSH au serveur était protégé uniquement par mot de passe, exposant le serveur à des risques de sécurité : attaque par force brute, divulgation accidentelle du mot de passe. De plus, aucune supervision du service n'était en place. L'objectif est double : sécuriser l'accès administrateur et mettre en place une supervision professionnelle du service.

2. Schéma d'architecture sécurisée

MacBook Air (client SSH) :

```
~/.ssh/id_handball      → clé privée ed25519 (CONFIDENTIELLE, jamais partagée)
~/.ssh/id_handball.pub  → clé publique (déposée sur le Pi)
~/.ssh/config           → alias : arbitre-handball / pi-admin
```

Raspberry Pi 4 (serveur SSH + serveur Express) :

```
/home/arbitre/.ssh/authorized_keys  → clé publique ed25519 autorisée
/home/pi/.ssh/authorized_keys       → clé publique ed25519 autorisée
/etc/ssh/sshd_config                → PasswordAuthentication no
                                      PubkeyAuthentication yes
```

3. Étapes de réalisation

Étape 1 -- Création du compte utilisateur limité (principe du moindre privilège)

```
sudo adduser arbitre
```

Vérification : groups arbitre → arbitre : arbitre users (pas de groupe sudo)

Étape 2 -- Génération de la paire de clés asymétriques ed25519 (sur le Mac)

```
ssh-keygen -t ed25519 -C "cle_arbitre"
```

Fichiers générés : ~/.ssh/id_handball (privée) + ~/.ssh/id_handball.pub (publique)
Algorithme ed25519 : courbe elliptique, plus sécurisé que RSA à taille de clé équivalente.
Vérification empreinte + randomart : ssh-keygen -lv -f ~/.ssh/id_handball.pub

Étape 3 -- Dépôt de la clé publique sur le Pi (compte arbitre)

```
ssh-copy-id -i ~/.ssh/id_handball.pub arbitre@192.168.10.34
```

→ Clé déposée automatiquement dans /home/arbitre/.ssh/authorized_keys
→ Permissions configurées automatiquement : chmod 700 .ssh / chmod 600 authorized_keys

Étape 4 -- Désactivation de l'authentification par mot de passe SSH

```
sudo nano /etc/ssh/sshd_config
PasswordAuthentication no    ← ligne décommentée et modifiée
PubkeyAuthentication yes     ← ligne décommentée
sudo systemctl restart ssh
```

⚠ Note : PasswordAuthentication commentée (#) = valeur par défaut yes. Il faut décommenter ET changer.

Étape 5 -- Dépôt de la clé pour le compte pi (admin)

Suite à la désactivation du mot de passe, connexion physique au Pi nécessaire pour :

```
Réactivation temporaire : PasswordAuthentication yes → sudo systemctl restart ssh
Depuis Mac : ssh-copy-id -i ~/.ssh/id_handball.pub pi@192.168.10.34
Re-désactivation : PasswordAuthentication no → sudo systemctl restart ssh
```

Bonne pratique retenue : déposer les clés sur TOUS les comptes AVANT de désactiver le mot de passe.

Étape 6 -- Configuration des accès simplifiés (~/.ssh/config sur le Mac)

```
Host arbitre-handball
    HostName 192.168.10.34
    User arbitre
    IdentityFile ~/.ssh/id_handball
```

```
  Host pi-admin
      HostName 192.168.10.34
      User pi
      IdentityFile ~/.ssh/id_handball
```

Connexion simplifiée : ssh arbitre-handball / ssh pi-admin

Étape 7 -- Supervision du service Express avec pm2

```
  pm2 monit                          → Dashboard temps réel (CPU: 0%, RAM: 62MB,
uptime: 22h)
  pm2 logs scoreboard --lines 20     → Consultation des logs d'activité
  pm2 describe scoreboard            → Détails complets (chemins logs, restarts: 0,
node 20.19.2)
```

Étape 8 – Authentification HTTP Basic sur control.html (Projet1)

Ajout d'une protection sur l'accès à control.html via le serveur Express :
Dans server.js, une fonction checkAuth() vérifie l'en-tête Authorization
de chaque requête avant d'autoriser l'accès à control.html et aux requêtes
POST /api/state (écriture).

Identifiants/mdp : arbitre / handball86
Encodage : Base64 (schéma HTTP Basic — RFC 2617)
index.html reste accessible sans authentification (lecture seule, vue spectateurs)

En effet, le fichier server.js (Projet 1) a été modifié pour intégrer cette protection : la route GET /control.html et la route POST /api/state passent désormais par la fonction checkAuth() avant d'être traitées. La route GET /api/state (lecture) et les fichiers statiques (CSS, JS, logos) restent accessibles sans authentification.

⚠️ Note : HTTP Basic transmet les identifiants en Base64 (non chiffré).
Acceptable en réseau LAN privé ; en production internet → HTTPS obligatoire.

```
pi@raspberrypi:~ $ nano /home/pi/Simple-Scoreboard-with-Timer/server.js
```

```
  GNU nano 8.4                                  /home/pi/Simple-Scoreboard-with-Timer/server.js
const express = require('express');
const fs = require('fs');
const path = require('path');

const app = express();
const PORT = 3000;

// ============ AUTHENTIFICATION control.html ============
const ARBITRE_USER = 'arbitre';
const ARBITRE_PASS = 'handball86';

function checkAuth(req, res, next) {
    const auth = req.headers.authorization;
    if (!auth || !auth.startsWith('Basic ')) {
        res.set('WWW-Authenticate', 'Basic realm="Arbitre - Tableau de score"');
        return res.status(401).send('Accès réservé à l\'arbitre.');
    }
    const credentials = Buffer.from(auth.split(' ')[1], 'base64').toString('utf8');
    const [user, pass] = credentials.split(':');
    if (user !== ARBITRE_USER || pass !== ARBITRE_PASS) {
        res.set('WWW-Authenticate', 'Basic realm="Arbitre - Tableau de score"');
        return res.status(401).send('Identifiants incorrects.');
    }
    next();
}

app.use(express.json());

// control.html protégé
app.get('/control.html', checkAuth, (req, res) => {
    res.sendFile(path.join(__dirname, 'control.html'));
});

// API state protégée (POST uniquement - écriture)
```

```
 GNU nano 8.4                                  /home/pi/Simple-Scoreboard-with-Timer/server.js

app.use(express.json());

// control.html protégé
app.get('/control.html', checkAuth, (req, res) => {
    res.sendFile(path.join(__dirname, 'control.html'));
});

// API state protégée (POST uniquement – écriture)
app.post('/api/state', checkAuth, (req, res) => {
    fs.writeFileSync('data.json', JSON.stringify(req.body, null, 2));
    res.json({ success: true });
});

// Routes publiques
app.use(express.static(__dirname));

app.get('/api/state', (req, res) => {
    const data = JSON.parse(fs.readFileSync('data.json', 'utf8'));
    res.json(data);
});

app.get('/api/logos', (req, res) => {
    const logosDir = path.join(__dirname, 'logos');
    const files = fs.readdirSync(logosDir).filter(f =>
        ['.png', '.jpg', '.jpeg', '.svg'].includes(path.extname(f).toLowerCase())
    );
    res.json(files);
});

app.listen(PORT, '0.0.0.0', () => {
    console.log(`Serveur lancé sur http://0.0.0.0:${PORT}`);
});
```

```
pi@raspberrypi:~ $ nano /home/pi/Simple-Scoreboard-with-Timer/server.js
pi@raspberrypi:~ $ pm2 restart scoreboard
Use --update-env to update environment variables
[PM2] Applying action restartProcessId on app [scoreboard](ids: [ 0 ])
[PM2] [scoreboard](0) ✓
```

id	name	namespace	version	mode	pid	uptime	↺	status	cpu	mem	user	watching
0	scoreboard	default	N/A	fork	1690	0s	1	online	0%	14.0mb	pi	disabled

Commande pm2 restart scoreboard permet de redémarrer le serveur.
Toute modification du fichier server.js sur le Raspberry Pi nécessite un redémarrage du service pour être prise en compte.

4. Tests réalisés et résultats

- Connexion par mot de passe → REFUSÉE ✓
  `ssh -o PubkeyAuthentication=no arbitre@192.168.10.34`
  `→ Permission denied (publickey)`

- Connexion par clé → ACCEPTÉE ✓
  `ssh -i ~/.ssh/id_handball arbitre@192.168.10.34  → arbitre@raspberrypi:~ $`

- Connexion via alias simplifiés → ACCEPTÉE ✓
  `ssh arbitre-handball  → arbitre@raspberrypi:~ $`
  `ssh pi-admin  → pi@raspberrypi:~ $`

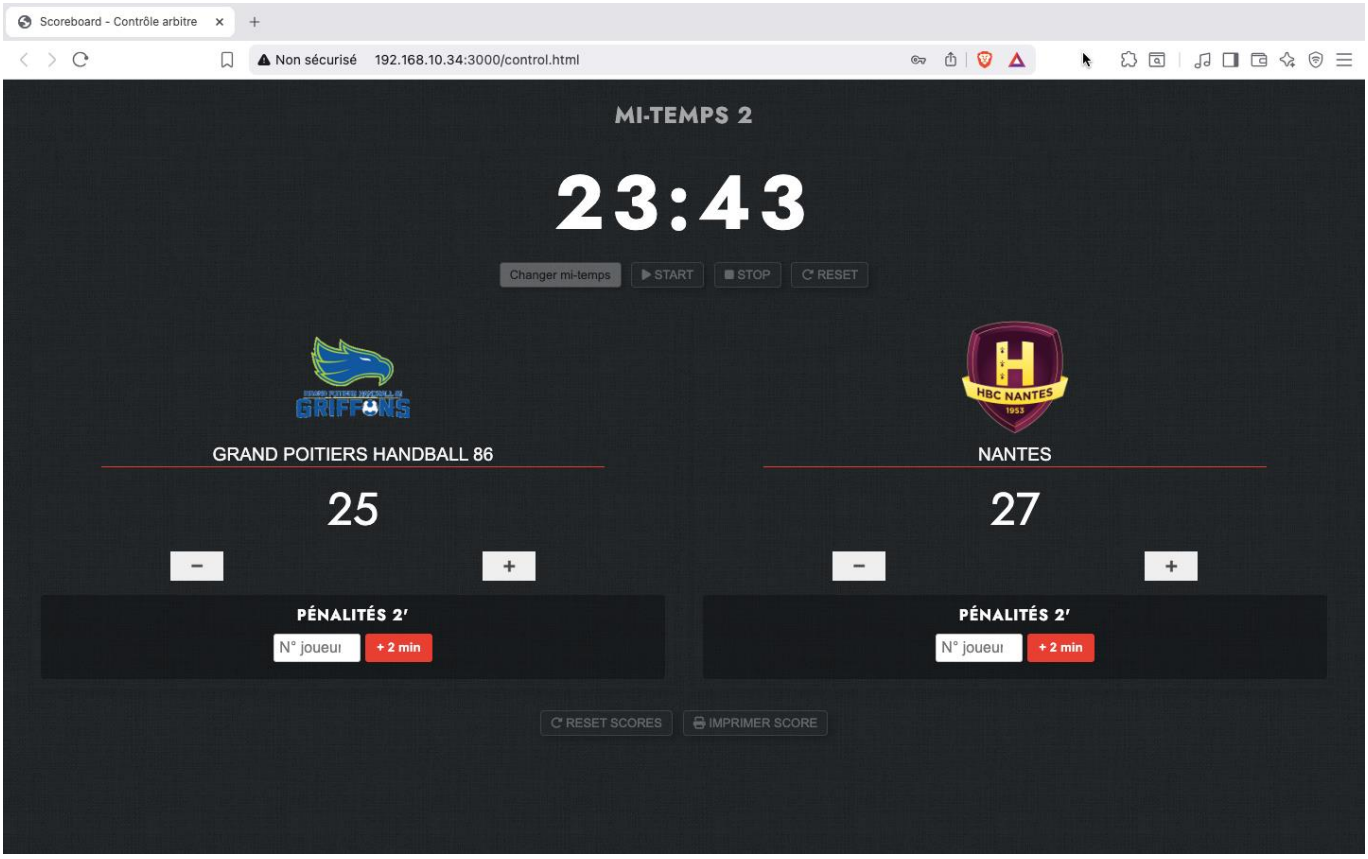
- Droits limités du compte arbitre → VÉRIFIÉS ✓
  `groups arbitre → arbitre : arbitre users  (pas de sudo)`

- Simulation d'incident — arrêt du service ✓
  `pm2 stop scoreboard  → status : stopped`
- Résolution d'incident — redémarrage du service ✓
  `pm2 restart scoreboard → status : online  (PID 3365, uptime 0s → remonte)`

- Authentification HTTP Basic control.html → Opérationnelle ✓
  Accès http://192.168.10.34:3000/control.html → Popup navigateur demandant un identifiant + mot de passe → Accès accordé avec arbitre / handball86 sinon Accès refusé avec mauvais identifiants.


192.168.10.34:3000/control.ht
192.168.10.34:3000/control.html
Se connecter
http://192.168.10.34:3000
Votre connexion à ce site n'est pas privée
Nom d'utilisateur
Mot de passe
Annuler
Connexion


192.168.10.34:3000/control.ht
192.168.10.34:3000/control.html
Se connecter
http://192.168.10.34:3000
Votre connexion à ce site n'est pas privée
Nom d'utilisateur
arbitre
Mot de passe
Annuler
Connexion



(Accès à l'interface de contrôle http://192.168.10.34:3000/control.html --> Autorisé)

5. Bilan et apports professionnels

Cette réalisation met en œuvre plusieurs bonnes pratiques d'administration système :

- Principe du moindre privilège : compte arbitre sans droits sudo
- Authentification forte : cryptographie asymétrique ed25519 (résistance aux attaques force brute)
- Suppression de l'authentification par mot de passe (vecteur d'attaque éliminé)
- Supervision proactive : pm2 monit permet de détecter une anomalie avant qu'elle impacte le service
- Traçabilité : logs pm2 conservés dans /home/pi/.pm2/logs/ pour audit
- Résilience : pm2 + systemd garantissent le redémarrage automatique du service après incident ou reboot

Le service scoreboard affiche 0 restart non planifié depuis sa mise en service (22h d'uptime continu lors de la supervision), démontrant la stabilité de la solution déployée en projet 1.

BTS Services Informatiques aux Organisations
Option SISR — Administration des systèmes et des réseaux
SESSION 2026

DOCUMENTATION TECHNIQUE RÉALISATION PROFESSIONNELLE N°2

Administration sécurisée d'un serveur Linux
Gestion des utilisateurs et authentification par clé SSH
Supervision du service avec pm2

Candidat	EMILIEN Noam
N° candidat	02428526184
Période	Mai 2026
Lieu	Domicile
Serveur	Raspberry Pi 4 Model B — Raspberry Pi OS 64-bit
Client SSH	MacBook Air — macOS Big Sur 11

Compétence évaluée : Exploiter, dépanner et superviser une solution d'infrastructure réseau

Sommaire

1. Contexte et problématique

Le Raspberry Pi 4 héberge le serveur web Express du tableau d'affichage handball (Réalisation n°1). L'accès SSH au serveur était protégé uniquement par mot de passe, exposant le serveur à des risques de sécurité : attaques par force brute, divulgation accidentelle du mot de passe. De plus, aucune supervision du service n'était en place. Cette réalisation vise à sécuriser l'accès administrateur et à mettre en place une supervision professionnelle du service.

Objectifs

- Créer un compte utilisateur limité (principe du moindre privilège)
- Mettre en place une authentification SSH par clé asymétrique ed25519
- Désactiver l'authentification par mot de passe (résistance aux attaques)
- Configurer des accès simplifiés via fichier ~/.ssh/config
- Superviser le service Express via pm2 (monitoring, logs, gestion d'incidents)

2. Architecture de sécurité mise en place

MacBook Air (client SSH)	Raspberry Pi 4 (serveur)
`~/.ssh/id_handball` `(clé privée — CONFIDENTIELLE)`	`/home/arbitre/.ssh/authorized_keys` `(clé publique autorisée)`
`~/.ssh/id_handball.pub` `(clé publique)`	`/home/pi/.ssh/authorized_keys` `(clé publique autorisée)`

`~/.ssh/config`
`(alias arbitre-handball / pi-admin)`
	`/etc/ssh/sshd_config`
	`PasswordAuthentication no`
	`PubkeyAuthentication yes`

Principe de fonctionnement : La clé privée reste sur le Mac et ne quitte jamais le poste client. La clé publique est déposée sur le serveur. Lors de la connexion, le serveur vérifie que le client possède la clé privée correspondante via un mécanisme de défi cryptographique — sans jamais transmettre la clé privée sur le réseau.

3. Création du compte utilisateur limité

Le compte **arbitre** est créé avec des droits volontairement limités, conformément au **principe du moindre privilège** : un utilisateur ne dispose que des droits strictement nécessaires à ses fonctions. Il n'a pas accès à **sudo** et ne peut donc pas administrer le serveur.

```
$ sudo adduser arbitre
```

```
pi@raspberrypi:~ $ sudo adduser arbitre
[sudo] Mot de passe de pi :
Nouveau mot de passe :
Retapez le nouveau mot de passe :
passwd : mot de passe mis à jour avec succès
Modifier les informations associées à un utilisateur pour arbitre
Entrer la nouvelle valeur, ou appuyer sur ENTER pour la valeur par défaut
        NOM []:
        Numéro de chambre []:
        Téléphone professionnel []:
        Téléphone personnel []:
        Autre []:
Is the information correct? [Y/n] y
pi@raspberrypi:~ $ groups arbitre
arbitre : arbitre users
```

Figure 1 — Création du compte arbitre et vérification des groupes (groups arbitre → pas de sudo)

La commande **groups arbitre** confirme que le compte appartient uniquement aux groupes *arbitre* et *users* — sans le groupe **sudo**. Le compte est également visible dans **/etc/passwd** :

```
$ cat /etc/passwd | grep arbitre
```

```
pi@raspberrypi:~ $ cat /etc/passwd | grep arbitre
arbitre:x:1001:1001:,,,:/home/arbitre:/bin/bash
```

Figure 2 — Vérification du compte dans /etc/passwd (UID 1001, shell /bin/bash)

4. Génération de la paire de clés SSH ed25519

La paire de clés est générée sur le **Mac (client)**. L'algorithme **ed25519** est choisi car il repose sur les courbes elliptiques et offre une sécurité supérieure à RSA à taille de clé équivalente, avec des performances meilleures.

```
$ ssh-keygen -t ed25519 -C "cle_arbitre"
```

Fichiers générés :

- **~/.ssh/id_handball** — clé privée (ne quitte JAMAIS le Mac)
- **~/.ssh/id_handball.pub** — clé publique (sera déposée sur le Pi)

```
noam@MacBook-Air-de-EMILIEN ~ % ssh-keygen -t ed25519 -C "cle_arbitre"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/Users/noam/.ssh/id_ed25519): /Users/noam/.ssh/id_handball
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/noam/.ssh/id_handball.
Your public key has been saved in /Users/noam/.ssh/id_handball.pub.
The key fingerprint is:
SHA256:AGTbD/sAxn5U6TzGI1W7VmoCRqYUonTcN/4zjrNGqv0 cle_arbitre
The key's randomart image is:
+--[ED25519 256]--+
| .ooB.o .o.      |
|...* O +o  .     |
|.   * X=. . .    |
|   o +.BB  +     |
|    . +oSo=      |
|     . + B       |
|      o + o      |
|    .. + .       |
|   ...oEo        |
+----[SHA256]-----+
noam@MacBook-Air-de-EMILIEN ~ %
```

Figure 3 — Génération de la paire de clés ed25519 avec randomart

Le **randomart** est une représentation visuelle de l'empreinte de la clé. Il permet une vérification rapide que la clé utilisée est bien la bonne (détection de substitution de clé).

```
$ ls ~/.ssh/
```

```
noam@MacBook-Air-de-EMILIEN ~ % ls ~/.ssh
id_handball      id_handball.pub known_hosts
```

Figure 4 — Les deux fichiers générés : clé privée et clé publique (.pub)

Vérification de l'empreinte et affichage du randomart :

```
$ ssh-keygen -lv -f ~/.ssh/id_handball.pub
```

```
noam@MacBook-Air-de-EMILIEN ~ % ssh-keygen -lv -f ~/.ssh/id_handball.pub
256 SHA256:AGTbD/sAxn5U6TzGI1W7VmoCRqYUonTcN/4zjrNGqv0 cle_arbitre (ED25519)
+--[ED25519 256]--+
| .ooB.o .o.      |
|...* O +o  .     |
|.   * X=. . .    |
|   o +.BB  +     |
|    . +oSo=      |
|     . + B       |
|      o + o      |
|    .. + .       |
|   ...oEo        |
+----[SHA256]-----+
```

Figure 5 — Empreinte SHA256 et randomart de la clé ed25519

5. Dépôt de la clé publique sur le serveur

La commande **ssh-copy-id** dépose automatiquement la clé publique dans le fichier **authorized_keys** du compte cible, et configure les permissions correctes (chmod 700 .ssh / chmod 600 authorized_keys).

```
$ ssh-copy-id -i ~/.ssh/id_handball.pub arbitre@192.168.10.34
```

```
noam@MacBook-Air-de-EMILIEN ~ % ssh-copy-id -i ~/.ssh/id_handball.pub arbitre@192.168.10.34
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/Users/noam/.ssh/id_handball.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
arbitre@192.168.10.34's password:

Number of key(s) added:        1

Now try logging into the machine, with:   "ssh 'arbitre@192.168.10.34'"
and check to make sure that only the key(s) you wanted were added.
```

Figure 6 — Dépôt de la clé publique sur le compte arbitre du Pi

Vérification côté serveur que la clé est bien présente :

```
$ cat ~/.ssh/authorized_keys
```

```
arbitre@raspberrypi:~ $ ls ~/.ssh
authorized_keys
arbitre@raspberrypi:~ $ cat ~/.ssh/authorized_keys
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIIWYoQ0v7yvOGTz8NicEhXLnYrRoH1c3vOFtkpHdA/xn cle_arbitre
arbitre@raspberrypi:~ $
```

Figure 7 — Clé publique ed25519 présente dans authorized_keys du compte arbitre

Note importante : La même clé publique a ensuite été déposée pour le compte **pi** (administrateur). La bonne pratique retenue : déposer les clés sur TOUS les comptes nécessaires *avant* de désactiver l'authentification par mot de passe.

6. Désactivation de l'authentification par mot de passe

La désactivation de l'authentification par mot de passe est la mesure de sécurité la plus importante : elle rend les attaques par force brute impossibles, car aucun mot de passe n'est accepté, quelle que soit sa longueur ou complexité.

```
$ sudo nano /etc/ssh/sshd_config
```

AVANT (commenté = valeur par défaut yes)	APRÈS (décommenté et modifié)
`#PubkeyAuthentication yes` `#PasswordAuthentication yes`	`PubkeyAuthentication yes` `PasswordAuthentication no`

■■ **Point important :** Une ligne commentée (#PasswordAuthentication yes) applique la valeur par défaut = yes. Il faut obligatoirement *décommenter ET changer* la valeur.

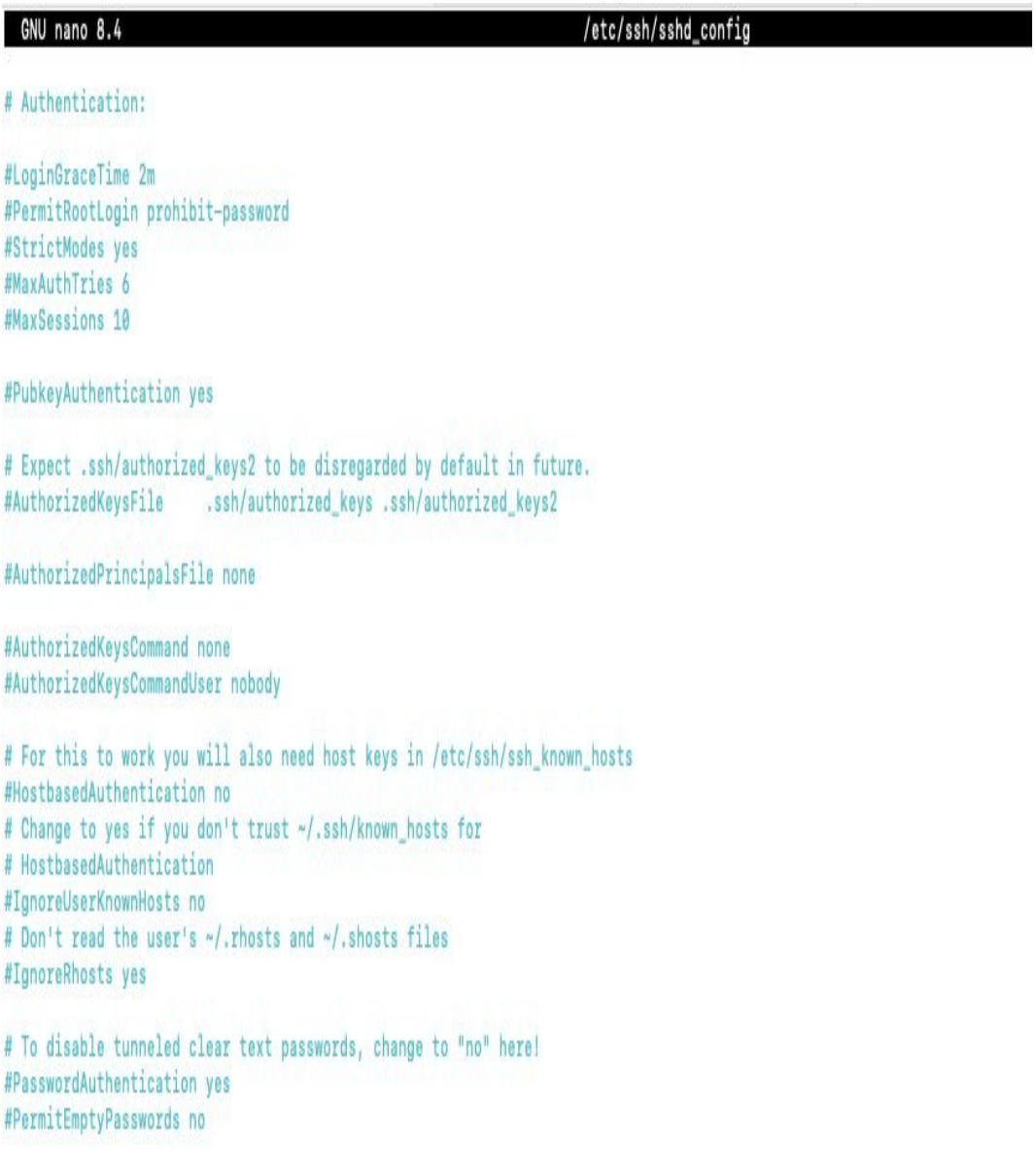


Figure 8 — sshd_config AVANT modification (tout commenté = valeurs par défaut)

Figure 9 — sshd_config APRÈS modification (PasswordAuthentication no actif)

```
$ sudo systemctl restart ssh
```

7. Configuration des accès

a) Accès simplifiés

Le fichier **~/.ssh/config** sur le Mac permet de définir des alias de connexion, évitant de saisir à chaque fois l'IP, le nom d'utilisateur et le chemin de la clé.

Alias	Compte	Usage
`arbitre-handball`	`arbitre (droits limités)`	`ssh arbitre-handball`
`pi-admin`	`pi (administrateur)`	`ssh pi-admin`

Figure 10 — Fichier ~/.ssh/config avec les deux alias (arbitre-handball et pi-admin)

b) Authentification HTTP Basic

L'accès à control.html est protégé par authentification HTTP Basic. Le navigateur affiche une popup demandant un identifiant et un mot de passe avant tout accès.

Le POST /api/state est également protégé — la lecture (GET) reste publique pour la TV.

Pop up d'authentification apparaissant lors d'une tentative de connexion

à la page d'accès de contrôle (control.html)

8. Tests de sécurité

Test 1 — Connexion par mot de passe refusée

Tentative de connexion sans utiliser la clé privée :

```
$ ssh -o PubkeyAuthentication=no arbitre@192.168.10.34
→ arbitre@192.168.10.34: Permission denied (publickey).
```

```
noam@MacBook-Air-de-EMILIEN ~ % ssh  arbitre@192.168.10.34
arbitre@192.168.10.34: Permission denied (publickey).
noam@MacBook-Air-de-EMILIEN ~ %
```

Figure 11 — Connexion refusée : le mot de passe est désactivé ✓

Test 2 — Connexion par clé acceptée

Connexion en spécifiant la clé privée :

```
$ ssh -i ~/.ssh/id_handball arbitre@192.168.10.34
```

```
noam@MacBook-Air-de-EMILIEN ~ % ssh -i ~/.ssh/id_handball arbitre@192.168.10.34
Linux raspberrypi 6.18.29+rpt-rpi-v8 #1 SMP PREEMPT Debian 1:6.18.29-1+rpt1 (2026-05-12) aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun May 24 18:52:27 2026 from 192.168.10.35
arbitre@raspberrypi:~ $
```

Figure 12 — Connexion acceptée par clé sans mot de passe ✓

Test 3 — Connexion via alias simplifiés

```
$ ssh arbitre-handball → arbitre@raspberrypi:~ $
```

```
noam@MacBook-Air-de-EMILIEN ~ % ssh arbitre-handball
Linux raspberrypi 6.18.29+rpt-rpi-v8 #1 SMP PREEMPT Debian 1:6.18.29-1+rpt1 (2026-05-12) aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon May 25 17:21:28 2026 from 192.168.10.35
arbitre@raspberrypi:~ $
```

Figure 13 — Connexion via alias arbitre-handball ✓

```
$ ssh pi-admin → pi@raspberrypi:~ $
```

```
noam@MacBook-Air-de-EMILIEN ~ % ssh pi-admin
Linux raspberrypi 6.18.29+rpt-rpi-v8 #1 SMP PREEMPT Debian 1:6.18.29-1+rpt1 (2026-05-12) aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon May 25 17:43:15 2026 from 192.168.10.35
pi@raspberrypi:~ $
```

Figure 14 — Connexion via alias pi-admin ✓

9. Supervision du service avec pm2

pm2 (Process Manager 2) est le gestionnaire de processus Node.js utilisé pour démarrer et superviser le serveur Express du tableau d'affichage handball. Il garantit le redémarrage automatique du service après incident ou reboot.

Dashboard de monitoring temps réel

```
$ pm2 monit
```

Figure 15 — Dashboard pm2 monit : CPU 0%, RAM 62MB, uptime 67s, 0 restarts

Le dashboard affiche en temps réel : consommation CPU (0%), mémoire RAM (62 MB), uptime du service, nombre de redémarrages et métriques Node.js (heap size, event loop latency).

Consultation des logs d'activité

```
$ pm2 logs scoreboard --lines 20
```

```
pi@raspberrypi:~ $ pm2 logs scoreboard --lines 20
[TAILING] Tailing last 20 lines for [scoreboard] process (change the value with --lines option)
/home/pi/.pm2/logs/scoreboard-error.log last 20 lines:
/home/pi/.pm2/logs/scoreboard-out.log last 20 lines:
0|scoreboa | Serveur lancé sur http://0.0.0.0:3000
0|scoreboa | Serveur lancé sur http://0.0.0.0:3000
0|scoreboa | Serveur lancé sur http://0.0.0.0:3000
0|scoreboa | Serveur lancé sur http://0.0.0.0:3000
0|scoreboa | Serveur lancé sur http://0.0.0.0:3000
0|scoreboa | Serveur lancé sur http://0.0.0.0:3000
0|scoreboa | Serveur lancé sur http://0.0.0.0:3000
0|scoreboa | Serveur lancé sur http://0.0.0.0:3000
0|scoreboa | Serveur lancé sur http://0.0.0.0:3000
0|scoreboa | Serveur lancé sur http://0.0.0.0:3000
0|scoreboa | Serveur lancé sur http://0.0.0.0:3000
0|scoreboa | Serveur lancé sur http://0.0.0.0:3000
0|scoreboa | Serveur lancé sur http://0.0.0.0:3000
0|scoreboa | Serveur lancé sur http://0.0.0.0:3000
0|scoreboa | Serveur lancé sur http://0.0.0.0:3000
0|scoreboa | Serveur lancé sur http://0.0.0.0:3000
0|scoreboa | Serveur lancé sur http://0.0.0.0:3000
0|scoreboa | Serveur lancé sur http://0.0.0.0:3000
0|scoreboa | Serveur lancé sur http://0.0.0.0:3000
0|scoreboa | Serveur lancé sur http://0.0.0.0:3000

^C
```

Figure 16 — Logs du service scoreboard (20 dernières lignes)

Les logs montrent les démarrages successifs du serveur Express sur le port 3000. Ils sont stockés dans **/home/pi/.pm2/logs/** pour audit et traçabilité.

Description détaillée du service

```
$ pm2 describe scoreboard
```

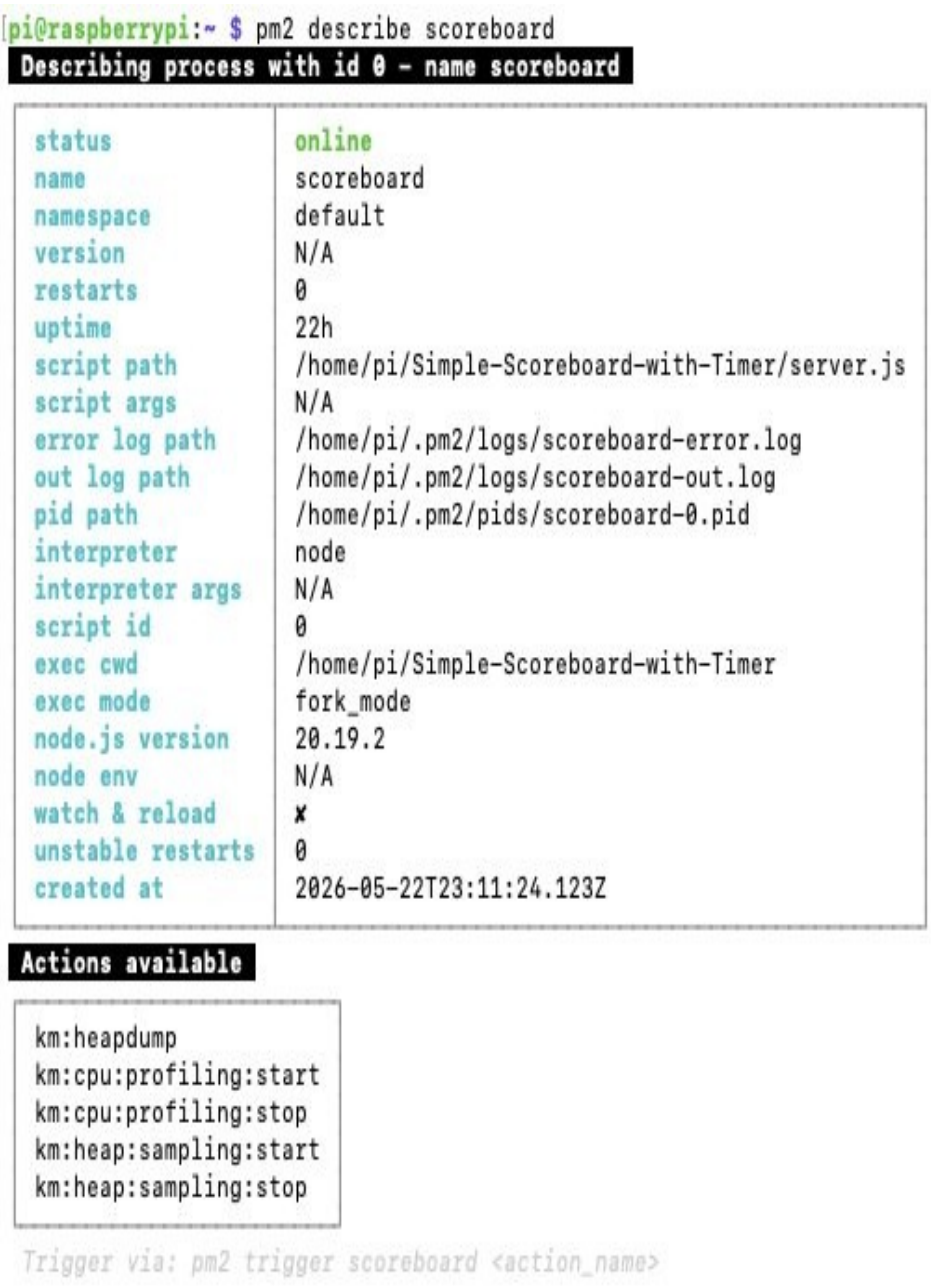


Figure 17 — Description pm2 : status online, 0 restarts non planifiés, uptime 22h

Code metrics value

Heap Size	17.07 MiB
Heap Usage	75.81 %
Used Heap Size	12.94 MiB
Active requests	0
Active handles	4
Event Loop Latency	0.31 ms
Event Loop Latency p95	1.13 ms
HTTP Mean Latency	4 ms
HTTP P95 Latency	5 ms
HTTP	0.01 req/min

Divergent env variables from local env

PWD	/home/pi/Simple-Scoreboard-with-Timer
SSH_CONNECTION	192.168.10.35 54756 192.168.10.34 22
XDG_SESSION_ID	5
SSH_CLIENT	192.168.10.35 54756 22
SSH_TTY	/dev/pts/1

```
Add your own code metrics: http://bit.ly/code-metrics
Use `pm2 logs scoreboard [--lines 1000]` to display logs
Use `pm2 env 0` to display environment variables
Use `pm2 monit` to monitor CPU and Memory usage scoreboard
```

Figure 18 — Métriques détaillées : heap, latence event loop, connexion SSH active

10. Simulation et résolution d'incident

Simulation d'une panne du service Express pour valider la procédure de redémarrage. Cette procédure est documentée et applicable en cas d'incident réel.

Simulation de la panne

```
$ pm2 stop scoreboard
```

Le service passe en status **stopped** — le tableau d'affichage n'est plus accessible.

Résolution de l'incident

```
$ pm2 restart scoreboard
```

Le service repasse en status **online** — le tableau d'affichage est rétabli.

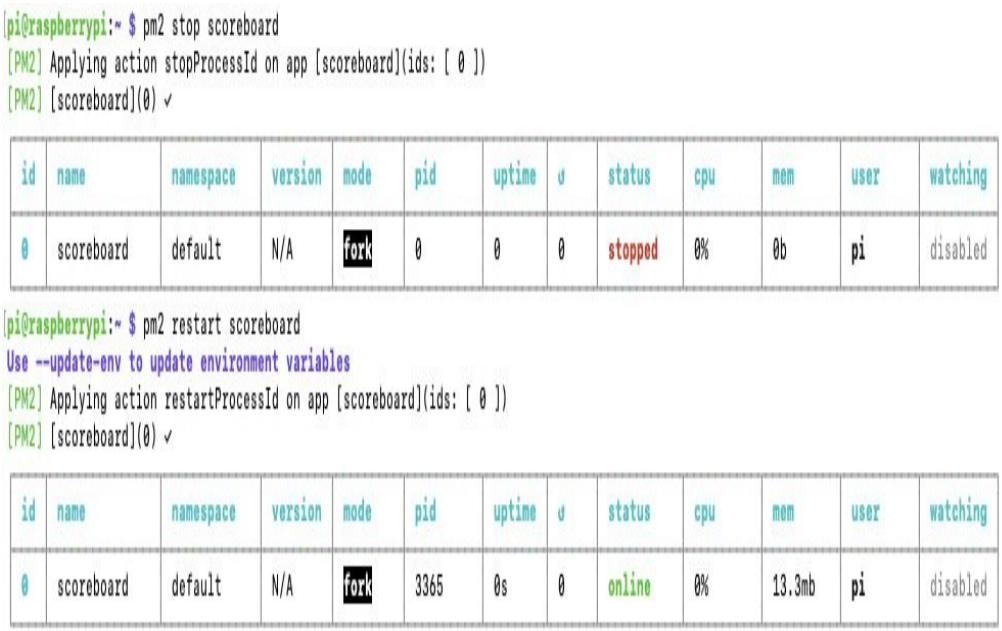


Figure 19 — Simulation incident (stopped) et résolution (online, PID 3365) ✓

Procédure d'administration à distance

Toutes les opérations de supervision et de résolution d'incident s'effectuent via SSH depuis le Mac :

```
$ ssh pi-admin # Connexion admin sécurisée
$ pm2 list # État de tous les services
$ pm2 logs scoreboard --lines 50 # Analyse des logs
$ pm2 restart scoreboard # Redémarrage du service
```

11. Bilan et apports professionnels

Bonne pratique	Mise en œuvre
Principe du moindre privilège	Compte arbitre sans sudo — accès limité au strict nécessaire
Authentification forte	Clé ed25519 (cryptographie asymétrique) — résistance aux attaques par force brute
Suppression vecteur d'attaque	PasswordAuthentication no — connexion par mot de passe impossible
Accès simplifié et sécurisé	Fichier ~/.ssh/config — alias pour chaque compte sans compromettre la sécurité
Supervision proactive	pm2 monit — détection d'anomalie avant impact sur le service
Traçabilité	Logs pm2 dans /home/pi/.pm2/logs/ — audit des activités du service
Résilience	pm2 + systemd — redémarrage automatique après incident ou reboot

Résultat obtenu : Le service scoreboard affichait **0 restart non planifié** lors de la supervision (22h d'uptime continu). La solution déployée en projet 1 est stable, sécurisée et supervisée selon les standards professionnels.

Lien avec le projet 1 : Cette réalisation sécurise et supervise directement l'infrastructure du tableau d'affichage handball. Les deux réalisations forment un ensemble cohérent : conception + déploiement (Projet 1) / exploitation + supervision sécurisée (Projet 2).