Solution de kiosque

Cette solution peut être installée sur n'importe quel PC, pourvu qu'il dispose d'une connexion réseau. Elle repose sur une Debian minimale, OpenBox et OpenKiosk.

Elle permet de créer des kiosques dédiés à l'affichage d'informations municipales, des PC en libre accès (avec accès via une liste blanche ou noire), des PC de consultation de catalogues de bibliothèque, etc.

Installation du kiosque

Objectif

→ Créer un environnement kiosque sécurisé, minimaliste et clonable, basé sur Debian + Openbox + OpenKiosk.
→ Utilisation visée : bornes de visioconférence, affichage municipal, postes en accès public.
→ À la fin : une image système prête à être clonée pour un déploiement rapide sur d'autres machines.

Prérequis :

→ La solution de kiosque repose sur un système centralisé de fichiers de configuration. Ces fichiers doivent être hébergés sur un serveur web accessible par tous les kiosques déployés.
→ Un serveur web accessible en HTTPS est donc nécessaire, avec un accès SSH permettant de créer et modifier les fichiers de configuration à distance.

> Nous n'installons pas chaque kiosque en suivant cette procédure. Après avoir réalisé une installation fonctionnelle, nous en avons fait une image que nous installons en quelques minutes grâce à Fog Project.

Procédure d'installation

Debian

- Télécharger Debian
- Choisir la catégorie « Petits CD ou clefs USB ».

> Conseil pour la création du compte utilisateur :
> **Utilisez le login : kiosk** pour simplifier les étapes.
> (Si vous choisissez un autre nom, pensez à adapter toutes les commandes et configurations de la documentation en conséquence.)

Pendant l'installation choisir uniquement :

- Serveur SSH
- Utilitaires usuels du système

> **Attention : ne pas installer d'environnement graphique !**

Connexion Réseau

> Dans les exemples ci-dessous, pensez à adapter le nom de l'interface réseau (ici les interfaces sont enp... et wlp...)

Connexion filaire

Dans le cas d'une connexion filaire ajouter dans le fichier `/etc/network/interfaces` :

```
auto enp0s31f6
allow-hotplug enp0s31f6
iface enp0s31f6 inet dhcp
    metric 100
```

Puis pour activer l'interface :

```
ifup enp0s31f6
```

WiFi (WPA Personal)

Créer un ficher `/etc/wpa_supplicant/wpa_supplicant.conf`

```
network={
    ssid="NOM_DU_WIFI"
    psk="MOT_DE_PASSE_DU_WIFI"
}
```

Puis ajouter dans le fichier `/etc/network/interfaces` :

```
auto wlp0s20f3
iface wlp0s20f3 inet dhcp
   wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
   metric 200
```

Puis pour activer l'interface :

```
ifup wlp0s20f3
```

WiFi (WPA Enterprise)

> **Adaptez cette configuration en fonction de vos paramètres de chiffrement et d'authentification**

Créer un ficher `/etc/wpa_supplicant/wpa_supplicant.conf`

```
network={
    ssid="mon-ssid"
    key_mgmt=WPA-EAP
    eap=PEAP
    identity="LOGIN"
    password="PASSWORD"
    phase2="auth=MSCHAPV2"
}
```

Puis ajouter dans le fichier `/etc/network/interfaces` :

```
auto wlp0s20f3
iface wlp0s20f3 inet dhcp
   wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
   metric 200
```

Puis pour activer l’interface :

```
ifup wlp0s20f3
```

Openbox

Installer **xorg** et **openbox**:

```
apt install -y xorg openbox jq
```

OpenKiosk

→ Site d'OpenKiosk

Comme mentionné sur le site, installer OpenKiosk avec ces commandes (avec wget, car curl n’est pas installé par défaut sur Debian).

```
wget https://www.mozdevgroup.com/dropbox/okcd/115/release/OpenKiosk115.20.0-2025-02-16-x86_64.deb
apt install -y ./OpenKiosk115.20.0-2025-02-16-x86_64.deb
```

> Pensez à vérifier sur le site qu'OpenKiosk n’a pas été mis à jour. Si une nouvelle version est disponible, adaptez les commandes avec le nom du nouveau fichier.

Configuration

Désactiver le menu grub au démarrage du PC & les messages kernel au boot

Éditer le fichier `/etc/default/grub` et changez la valeur de `GRUB_TIMEOUT` à `0` puis `GRUB_CMDLINE_LINUX_DEFAULT` à `"quiet loglevel=0"`

Pour appliquer ce changement, exécuter la commande suivante :

```
update-grub
```

Configuration d’un auto-login

Éditer le fichier `/etc/systemd/system/getty@tty1.service.d/override.conf` avec :

```
systemctl edit getty@tty1.service
```

Rajouter ces trois lignes :

> Attention il faut les rajouter à la 3ème ligne du fichier (juste en dessous de la ligne n°2, donc) !

```
[Service]
ExecStart=
ExecStart=-/sbin/agetty --autologin root --noclear %I 38400 linux
```

Puis, pour activer ce changement au démarrage du PC :

```
systemctl enable getty@tty1.service
```

Création du script de configuration de premier lancement

> Vérifiez que Python3 est installé sur la machine. Si ce n'est pas le cas, installez-le.

Dans `/root/.profile` ajouter à la fin :

```
if [ ! -e /home/kiosk/firstTime.ok ]; then
    python3 /home/kiosk/firstTime.py
    sleep 2
    reboot
fi
```

Puis créer un fichier `/home/kiosk/firstTime.py` avec ce contenu :

```
import os
import subprocess
from time import sleep
import requests
import ipaddress
import json
import uuid
import signal
import time


def command(command):
    """Executes a command and returns the output"""
    result = subprocess.run(command, shell=True, capture_output=True, text=True)
    return result.stdout, result.stderr


# Prevent ctrl-c
def handler(signum, frame):
    pass
```

```
signal.signal(signal.SIGINT, handler)



def list_network_interfaces():
    """Lists available network interfaces"""
    stdout, stderr = command("ip link show")
    interfaces = []
    if stderr:
        print(f"Error retrieving interfaces: {stderr}")
    else:
        for line in stdout.splitlines():
            if line[0].isdigit():
                interface_name = line.split(":")[1].split()[0]
                interfaces.append(interface_name)
            if "lo" in interfaces:
                interfaces.remove("lo")
    return interfaces



def configure_wired(interface):
    """Configures the wired connection"""
    print(f"Configuring wired connection on {interface}...")
    dhcp = input("Do you want to use DHCP? (yes/no): ").strip().lower()

    if dhcp == 'yes':
        # Configure DHCP for a wired interface
        with open("/etc/network/interfaces", "w") as myfile:
            myfile.write("source /etc/network/interfaces.d/*\n")
            myfile.write("auto lo\niface lo inet loopback\n")
            myfile.write(f"auto {interface}\nallow-hotplug {interface}\niface {interface} inet
dhcp")

        command(f"ifup {interface}")
        print(f"{interface} configured with DHCP.")
    else:
        # Manual configuration (static IP)
        ip = input("Enter the static IP address: ").strip()
        netmask = input("Enter the subnet mask (e.g., 255.255.255.0): ").strip()
        cidr = ipaddress.IPv4Network(f"0.0.0.0/{netmask}").prefixlen
        gateway = input("Enter the default gateway: ").strip()
```

```
        dns = input("Enter the DNS address: ").strip()

        # Static IP configuration
        with open("/etc/network/interfaces", "w") as myfile:
            myfile.write("source /etc/network/interfaces.d/*\n")
            myfile.write("auto lo\niface lo inet loopback\n")
            myfile.write(f"auto {interface}\nallow-hotplug {interface}\niface {interface} inet static\n\taddress {ip}/{cidr})\n\tgateway {gateway}")

        command(f"echo nameserver {dns} > /etc/resolv.conf")
        command(f"ifup {interface}")
        print(f"{interface} configured with static IP {ip}.")



def configure_wifi(interface):
    """Configures the Wi-Fi connection"""
    print(f"Configuring Wi-Fi on {interface}...")
    ssid = input("Enter the Wi-Fi network name (SSID): ").strip()
    password = input("Enter the Wi-Fi network password: ").strip()

    # Configure Wi-Fi via `nmcli` (NetworkManager)
    with open("/etc/network/interfaces", "w") as myfile:
            myfile.write("source /etc/network/interfaces.d/*\n")
            myfile.write("auto lo\niface lo inet loopback\n")
            myfile.write(f"auto {interface}\nallow-hotplug {interface}\niface {interface} inet dhcp\n\twpa-ssid {ssid}\n\twpa-psk {password}")

    command(f"ifup {interface}")
    print(f"Wi-Fi connection established on {interface}.")



def configure_network():
    """Main function to configure the network interface"""
    interfaces = list_network_interfaces()

    if not interfaces:
        print("No network interfaces found.")
        return

    print("Available network interfaces:")
```

```
    print("0. Skip this step")
    for i, iface in enumerate(interfaces, 1):
        print(f"{i}. {iface}")

    choice = int(input("Choose the network interface to configure (number): ").strip()) - 1

    if choice not in range(len(interfaces)):
        print("Invalid choice.")
        return



    selected_interface = interfaces[choice]
    print(f"Chosen interface: {selected_interface}")

    connection_type = input("Do you want to configure a wired connection or Wi-Fi (only WPA with password)? (wired/wifi): ").strip().lower()

    if connection_type == "wired":
        configure_wired(selected_interface)
    elif connection_type == "wifi":
        configure_wifi(selected_interface)
    else:
        print("Invalid choice.")



if not os.path.isfile("firstTime.ok"):
    print('''

    ██   ██ ██  ██████  ███████ ██   ██
    ██  ██  ██ ██    ██ ██      ██  ██
    █████   ██ ██    ██ ███████ █████
    ██  ██  ██ ██    ██      ██ ██  ██
    ██   ██ ██  ██████  ███████ ██   ██

    ''')

    sleep(2)
    print("[*] First-time configuration script [*]")
    print("[*] - Version: 1.0                  [*]")
    print("[*] - By: Alexandre Salmetoz        [*]")
```

```
    sleep(2)

    try:
        print("\n1. Hostname configuration")
        hostname = input("Hostname: ")
        command(f"hostnamectl set-hostname {hostname}")
        print("Hostname: OK")
    except Exception as e:
        print(f"Error: {e}")

    try:
        print("\n2. Network connection configuration")
        configure_network()
    except Exception as e:
        print(f"Error: {e}")

    try:
        command("touch /home/kiosk/firstTime.ok")

        with open("/etc/systemd/system/getty@tty1.service.d/override.conf", 'r') as file:
            content = file.read()
        content = content.replace('root', 'kiosk')
        with open("/etc/systemd/system/getty@tty1.service.d/override.conf", 'w') as file:
            file.write(content)

        command("systemctl mask getty@tty1.service")

    except Exception as e:
        print(f"Error: {e}")



    print("\n[ Configuration complete! ]\n")

    print("The device will restart...")
```

Ce script est lancé au premier lancement du pc seulement, pour permettre de configurer le hostname, le réseau et redémarrer

Création d’un service système pour récupérer les fichiers de configuration au démarrage du PC

Pour ce faire, créer un fichier `/etc/systemd/system/kiosk.service` avec ce contenu :

```
[Unit]
Description=Execute start.sh at startup

[Service]
ExecStart=/home/kiosk/start.sh
Restart=on-failure
User=root
StandardInput=tty
StandardOutput=journal
StandardError=journal
TTYPath=/dev/tty3

[Install]
WantedBy=multi-user.target
```

Puis, pour activer le service :

```
systemctl enable kiosk.service
```

Voici le script qui permet de récupérer les fichiers de configuration auprès du serveur à chaque démarrage, à placer dans `/home/kiosk/start.sh` :

```
#!/bin/bash

start_time=$(date +%s)
timeout=10  # Délai d'attente en secondes avant que le script saute cette étape du ping
while ! ping -c 1 -W 1 "8.8.8.8" &> /dev/null; do
    elapsed_time=$(($(date +%s) - start_time))
    if [ "$elapsed_time" -ge "$timeout" ]; then
        log_message "!! Aucune connexion à Internet !!"
        break
    fi
    sleep 0.1
done

if [ -e /home/kiosk/firstTime.ok ]; then
```

```
        echo "Démarrage..."
        python3 /home/kiosk/getConfig.py
        source /home/kiosk/boot_script.sh
fi
```

> Ce script va récupérer avec « python3 /home/kiosk/getConfig.py » le fichier de configuration pour OpenKiosk, une Whitelist, un ficher de configuration système et un script qui sera exécuté à chaque démarrage du pc sur un serveur (dont l'url est contenue dans la variable $DOMAIN_NAME), puis placer ces fichiers dans /usr/lib/OpenKiosk/

Rendre ce script exécutable :

```
chmod +x /home/kiosk/start.sh
```

Puis créer le script python `/home/kiosk/getConfig.py` :

```
import requests
import json
import subprocess
import os



DOMAIN_NAME = "kiosk-config.mondomaine.fr"



def command(command):
    """Exécute une commande et retourne la sortie"""
    result = subprocess.run(command, shell=True, capture_output=True, text=True)
    return result.stdout

hostname = command("hostname")[:-1]

# --- Config OpenKiosk ---

res = requests.get("https://" + DOMAIN_NAME + "/" + hostname + "/config.cfg")

with open("/usr/lib/OpenKiosk/openkiosk.cfg", 'w') as f:
    f.write(res.text)
```

```
# --- Config System ---

res = requests.get("https://" + DOMAIN_NAME + "/" + hostname + "/config_system.json")

with open("/home/kiosk/config_system.json", 'w') as f:
    f.write(res.text)



# --- Boot Script ---

res = requests.get("https://" + DOMAIN_NAME + "/" + hostname + "/boot_script.sh")

with open("/home/kiosk/boot_script.sh", 'w') as f:
    f.write(res.text)
command("chmod +x /home/kiosk/boot_script.sh")



# --- Config Filters ---

res = requests.get("https://" + DOMAIN_NAME + "/" + hostname + "/filters.txt")

with open("/usr/lib/OpenKiosk/filters.txt", 'w') as f:
    f.write(res.text)
```

> Changez la variable DOMAIN_NAME si nécessaire, les fichiers seront récupérés à la racine.
> Par exemple, pour un kiosk, les fichiers de configurations seront :
>
> → https://$DOMAIN_NAME/$HOSTNAME/config.cfg
> → https://$DOMAIN_NAME/$HOSTNAME/filters.txt
> → https://$DOMAIN_NAME/$HOSTNAME/config_system.json
> → https://$DOMAIN_NAME/$HOSTNAME/boot_script.sh

Création d’un service système pour lancer Openbox au démarrage du PC

Pour ce faire, créer un fichier `/etc/systemd/system/kiosk-openbox.service` avec ce contenu :

```
[Unit]
Description=Execute start-openbox.sh at startup
After=kiosk.service
```

```
[Service]
ExecStart=/home/kiosk/start-openbox.sh
Restart=on-failure
User=kiosk
StandardInput=tty
StandardError=journal
StandardOutput=journal
TTYPath=/dev/tty3

[Install]
WantedBy=multi-user.target
```

Puis activer le service :

```
systemctl enable kiosk-openbox.service
```

Ce script permet de lancer Openbox au démarrage du PC, à placer dans `/home/kiosk/start-openbox.sh` :

```
#!/bin/bash

if [ -e /home/kiosk/firstTime.ok ]; then
    if [ "$(jq -r '.cursor' /home/kiosk/config_system.json)" == "false" ]; then
        startx -- -nocursor
    else
        startx
    fi
fi
```

Pour rendre le script exécutable :

```
chmod +x /home/kiosk/start-openbox.sh
```

Ajouter l'utilisateur kiosk au groupe `input` :

```
usermod -a -G input kiosk
```

Configuration d'un autostart pour OpenKiosk

Créer un fichier `autostart` dans `/home/kiosk/.config/openbox/` :

```
if [ "$(jq -r '.screen_saver' /home/kiosk/config_system.json)" = "false" ]; then
    xset s off
    xset s noblank
    xset -dpms
else
    xset dpms 0 0 $(jq -r '.screen_saver_minutes' /home/kiosk/config_system.json | awk '{print
$1*60}')
fi

if [ "$(jq -r '.sound' /home/kiosk/config_system.json)" = "true" ]; then
    pactl set-sink-mute @DEFAULT_SINK@ 0
else
    pactl set-sink-mute @DEFAULT_SINK@ 1
fi

xmodmap -e "keycode 64 ="
xmodmap -e "keycode 68 ="
xmodmap -e "keycode 69 ="
xmodmap -e "keycode 70 ="
xmodmap -e "keycode 71 ="
xmodmap -e "keycode 72 ="
xmodmap -e "keycode 73 ="
xmodmap -e "keycode 74 ="
xmodmap -e "keycode 75 ="
xmodmap -e "keycode 76 ="
xmodmap -e "keycode 95 ="
xmodmap -e "keycode 96 ="

xset led named "Num Lock"
OpenKiosk
```

> Les commandes *xset* permettent de désactiver l’écran de veille (si besoin de laisser l’écran de veille, commenter ces 3 premières lignes)

> La commande *xmodmap -e "keycode 64 ="* permet de désactiver la touche ALT du clavier pour éviter toute tentative de raccourcis clavier

Mise en place d’une page par défaut pour OpenKiosk :

Créer un fichier `/home/kiosk/default.html` :

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Kiosque - Page par défaut</title>
    <style>
        /* Base */
        body {
            font-family: 'Arial', sans-serif;
            background-color: #f9f9f9;
            margin: 0;
            padding: 0;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            flex-direction: column;
        }

        /* Header */
        .header {
            background: linear-gradient(135deg, #fc00ff, #00dbde);
            color: white;
            padding: 50px 30px;
            border-radius: 12px;
            text-align: center;
            box-shadow: 0 10px 30px rgba(0, 0, 0, 0.15);
            margin-bottom: 40px;
            max-width: 90%;
            width: 500px;
        }

        .header p {
            font-size: 36px;
            font-weight: 700;
            margin: 0;
            text-shadow: 2px 2px 10px rgba(0, 0, 0, 0.4);
```

```
}

/* Conteneur principal */
.container {
    background-color: #ffffff;
    border-radius: 12px;
    box-shadow: 0 10px 20px rgba(0, 0, 0, 0.1);
    padding: 40px 30px;
    max-width: 90%;
    width: 500px;
    text-align: left;
    line-height: 1.6;
    color: #333;
}

/* Titre de la section */
h1 {
    font-size: 28px;
    color: #333;
    margin-bottom: 20px;
    text-align: center;
    font-weight: bold;
}

/* Liste et paragraphes */
p {
    font-size: 16px;
    margin-bottom: 20px;
}

ul {
    list-style-type: none;
    padding: 0;
    font-size: 16px;
}

ul li {
    background-color: #f5f5f5;
    color: #f44336;
    padding: 12px;
```

```
            margin: 8px 0;
            border-radius: 8px;
            box-shadow: 0 1px 3px rgba(0, 0, 0, 0.1);
            display: flex;
            align-items: center;
        }

        ul li::before {
            content: '⚠️';
            margin-right: 10px;
            color: #f44336;
        }

        /* Footer */
        .footer {
            margin-top: 20px;
            font-size: 12px;
            color: #888;
            text-align: center;
        }

        .footer small {
            font-style: italic;
        }
    </style>
</head>
<body>

    <div class="header">
        <p>Page par défaut du Kiosque</p>
    </div>

    <div class="container">
        <h1>Raisons pour lesquelles cette page s'affiche :</h1>
        <ul>
            <li>Le kiosque n'est pas connecté à Internet.</li>
            <li>Il n'y a pas de configuration pour le hostname du kiosque sur le serveur.</li>
            <li>La récupération de la configuration auprès du serveur a échoué.</li>
        </ul>
```

```
        <div class="footer">
            <p>Merci de vérifier la connexion et la configuration du kiosque.</p>
            <p><small>Solution de kiosque créée par Alexandre Salmetoz</small></p>
        </div>
    </div>

</body>
</html>
```

Dans le fichier `/usr/lib/OpenKiosk/openkiosk.cfg` ajouter à la fin :

```
pref("browser.startup.homepage", "file:///home/kiosk/default.html");
```

Mise en place du système audio :

Installer pulseaudio et pavucontrol (interface graphique pour gérer l’audio) :

```
apt install -y pulseaudio pavucontrol alsa-utils
```

Créer un dossier `/home/kiosk/.config/pulse` :

```
mkdir /home/kiosk/.config/pulse
```

Changer le propriétaire du dossier sur l’utilisateur kiosk :

```
chown kiosk: /home/kiosk/.config/pulse
```

Supprimer le service pulseaudio qui se lance par défaut :

```
mv /etc/systemd/user/default.target.wants/pulseaudio.service
/etc/systemd/user/default.target.wants/pulseaudio.service.old
mv /etc/systemd/user/sockets.target.wants/pulseaudio.socket
/etc/systemd/user/sockets.target.wants/pulseaudio.socket.old
```

Création d’un service système dans `/etc/systemd/system/pulseaudio.service` :

```
[Unit]
Description=PulseAudio system server

[Service]
Type=notify
```

```
ExecStart=pulseaudio --daemonize=no --system --realtime --log-target=journal

[Install]
WantedBy=multi-user.target
```

Activation du service :

```
systemctl enable pulseaudio.service
```

Ajouter l'utilisateur kiosk et pulse aux groupes suivants :

```
usermod -a -G pulse-access kiosk
usermod -a -G pulse kiosk
usermod -a -G audio kiosk
usermod -a -G pulse-access pulse
usermod -a -G pulse pulse
usermod -a -G audio pulse
```

Désactivation des terminaux virtuels

En faisant la combinaison de touches CTRL+ALT+F1 ~ F6, on peut accéder à plusieurs terminaux virtuels. Il faut les **désactiver** pour empêcher la sortie d'OpenKiosk

Créer un fichier `/etc/X11/xorg.conf` :

```
Section "ServerFlags"
    Option "DontVTSwitch" "true"
    Option "DontZap" "yes"
EndSection
```

Désactivation des raccourcis clavier sur openbox

Il y a de nombreux raccourcis clavier sur **openbox** qui pourraient permettre à l'utilisateur de sortir du navigateur, nous allons donc **tous** les désactiver.

Pour cela, dans le fichier `/etc/xdg/openbox/rc.xml`, repérer les balises `<keyboard>` et `</keyboard>`, puis **supprimer tous les commentaires** se trouvant **entre** ces balises.

> Un commentaire commence par <!-- et finit par -->

Puis commenter toutes les lignes entre les deux balise `<keyboard>` et `</keyboard>`, pour cela rajouter editer les balises comme ceci : `<!-- <keyboard>` et `</keyboard> -->`.

Configuration d'une imprimante

Afin d'ajouter une nouvelle imprimante sur le kiosque, nous utilisons `cups.` Pour l'installer :

```
apt install cups
```

> Installez les pilotes (fichier PPD) de votre imprimante (consultez le site fabricant pour récupérer les pilotes)

Ensuite, pour ajouter l'imprimante :

```
lpadmin -p Nom_Imprimante -E -v URI_Imprimante -m PPD_File_Name
```

> L'option -m à pour répertoire courant : **/usr/share/cups/model/**
> Et donc l'option : -m **something/test.ppd**
> ...fera référence au fichier : **/usr/share/cups/model/something/test.ppd**

Ensuite désactiver la auto-détection d'imprimante réseau :

Dans le fichier `/etc/cups/cupsd.conf` changez :

- `Browsing yes` en `Browsing No`
- `BrowseLocalProtocols dnssd` en `BrowseLocalProtocols none`

Puis faites :

```
systemctl disable cups-browsed
```

Puis mettre l'imprimante ajoutée précédemment en imprimante par défaut avec :

```
lpadmin -d Nom_Imprimante
```

Redémarrage automatique du kiosque chaque nuit à 2h

La mise en place d'un redémarrage automatique des kiosques peut être utile pour :

- qu'ils récupèrent automatiquement des fichiers de configuration à jour
- que l'adresse IP soit régulièrement présente dans les logs du serveur, ce qui peut être utile pour les identifier facilement en cas de besoin.

Création d'un cron dédié :

```
crontab -e
```

...puis copier-coller cette ligne :

```
0 2 * * * /sbin/reboot
```

Mises à jour automatiques

Pour mettre en place des mises à jour automatiques nous allons utiliser `unattended-upgrades`

```
apt install -y unattended-upgrades
```

Éditer le fichier `/etc/apt/apt.conf.d/50unattended-upgrades` et dé-commenter ces deux lignes :

```
"origin=Debian,codename=${distro_codename}-updates";
"origin=Debian,codename=${distro_codename}-proposed-updates";
```

Dans le même fichier dé-commenter ces deux lignes et changer les valeurs comme ceci :

```
Unattended-Upgrade::Automatic-Reboot "true";
Unattended-Upgrade::Automatic-Reboot-Time "01:00";
```

Et pour finir dé-commenter cette ligne et mettre la valeur sur `true` :

```
Unattended-Upgrade::SyslogEnable "true";
```

> En cas de besoin, les logs se trouvent dans **/var/log/unattended-upgrades**

Activer unattended-upgrades :

```
cp /usr/share/unattended-upgrades/20auto-upgrades /etc/apt/apt.conf.d/20auto-upgrades
```

Éditer le fichier `/etc/apt/apt.conf.d/20auto-upgrades` :

```
APT::Periodic::Update-Package-Lists "1";
APT::Periodic::Unattended-Upgrade "1";
APT::Periodic::Download-Upgradeable-Packages "1";
APT::Periodic::AutocleanInterval "30";
```

Par défaut les mise à jour se déclenchent vers 6h du matin. Pour modifier l'heure :

```
systemctl edit apt-daily-upgrade.timer
```

Pour déclencher les mises à jour à 1h30 le matin, ajouter **à la troisième ligne** :

```
[Timer]
OnCalendar=
OnCalendar=01:30
RandomizedDelaySec=0
```

Appliquer les changements :

```
systemctl restart apt-daily-upgrade.timer
```

Pour faire de même avec le téléchargement des paquets (qui se fait par défaut à 6h) :

```
systemctl edit apt-daily.timer
```

Pour déclencher les mises à jour à 1h le matin, ajouter **à la troisième ligne** :

```
[Timer]
OnCalendar=
OnCalendar=01:00
RandomizedDelaySec=0
```

Appliquer les changements :

```
systemctl restart apt-daily.timer
```

Changer la langue d’affichage d’OpenKiosk

Lancer Openbox (à partir de l'utilisateur kiosk) :

```
startx
```

Aller dans les paramètre d’OpenKiosk (Shift + F1) (mdp par défaut `admin`)→ General → Language → English (US) → Search for more languages… → Select to add language → Français → Add → Ok

> Une fois les fichiers de configuration mis en place sur le serveur HTTP (voir livre dédié), redémarrer la machine.

Auteur original de cette documentation : Alexandre Salmetoz.

Serveur et fichiers de configuration

Mise en place de la configuration sur le serveur

La configuration des kiosques (qui détermine notamment leur finalité) est réalisée à partir de 4 fichiers téléchargés à chaque démarrage. Ils doivent donc être placés sur un serveur web accessible par les kiosques (intranet, site web de la collectivité...) :

- `config.cfg`
  contient les éléments de configuration d'OpenKiosk
- `filters.txt`
  contient les listes d'URL accessibles (liste blanche) ou interdites (liste noire)
- `config_system.json`
  contient les éléments de configuration du système
- `boot_script.sh`
  contient une liste de commandes exécutées en ***root*** au démarrage du kiosque

Les paramètres sont plutôt parlants et devraient vous permettre d'adapter votre solution à des usages très différents.

Affichage municipal :

Pour de l'affichage d'informations municipales, l'utilisation d'une solution de type [slides.com]() (qui repose sur reveal.js) peut faire l'affaire. Dans ce cas, modifiez le fichier **`config.cfg`** de telle sorte que la page de lecture de la présentation s'ouvre en plein écran au démarrage, désactivez le curseur et la mise en veille de l'écran dans le fichier **`config_system.json`**, etc.

Exemples de fichiers de configuration :

> **Ces exemples permettent de créer un kiosque dédié à de l'accès public sécurisé.**

Fichier **`config.cfg`** :

```
// Ficher de configuration OpenKiosk
pref("browser.startup.homepage", "https://example.com/");
```

```
pref("browser.download.dir", "/tmp");
pref("openkiosk.fullscreen.enabled", false);
pref("openkiosk.filters.enabled", true);
pref("openkiosk.filters.protocol.about.enabled", false);
pref("openkiosk.filters.protocol.blob.enabled", true);
pref("openkiosk.filters.protocol.data.enabled", true);
pref("openkiosk.filters.protocol.file.enabled", false);
pref("openkiosk.filters.protocol.ftp.enabled", false);
pref("openkiosk.filters.protocol.javascript.enabled", false);
pref("openkiosk.filters.protocol.mailto.enabled", false);
pref("openkiosk.filters.protocol.res.enabled", false);
pref("openkiosk.filters.protocol.sms.enabled", false);
pref("openkiosk.filters.protocol.tel.enabled", false);
pref("openkiosk.filters.protocol.viewsource.enabled", false);
pref("openkiosk.session.inactiveTerminal.enabled", true);
pref("openkiosk.session.inactiveTerminal.minutes", 10);
pref("openkiosk.session.inactiveTerminal.warn.enabled", false);
pref("openkiosk.session.inactiveTerminal.warn.secondes", 10);
pref("openkiosk.session.inactiveTerminal.warn.manual.enabled", false);
pref("openkiosk.session.cookies.enabled", false);
pref("openkiosk.tabs.enabled", true);
pref("openkiosk.osk.enabled", false);
pref("openkiosk.ui.urlbar.enabled", false);
pref("openkiosk.ui.context.menu.enabled", false);
pref("openkiosk.ui.context.search.enabled", false);
pref("openkiosk.admin.password",
"e52f15f817710380d47db05179ec4deb,98c34518c3ca6afa539bdcf87ce274b8bc56938f77d4917b02c38e646862
3d43");
pref("openkiosk.print.web.enabled", false);
pref("openkiosk.keys.settings.enabled", true);
pref("openkiosk.file.upload.enabled", false);
pref("security.mixed_content.block_active_content", false);
pref("permissions.default.microphone", 0);
pref("permissions.default.camera", 0);
pref("openkiosk.filters.strictmode.enabled", true);
pref("openkiosk.filters.file", "/usr/lib/OpenKiosk/filters.txt");
pref("print.always_print_silent", false);
```

Fichier `filters.txt` :

```
# Whitelist (* = wilcard pour autoriser tous les sous-domaines)
filter[https://*.ma-mairie.fr, ALL]
filter[https://*.gouv.fr, ALL]
filter[https://*.example.com, ALL]
```

Fichier `config_system.json` :

```
{
   "screen_saver": "true",
   "screen_saver_minutes": 10,
   "cursor": "true",
   "sound": "true"
}
```

Fichier `boot_script.sh` :

```
# Script qui s'éxecute en root au démarrage
```

> Auteur original de cette documentation : Alexandre Salmetoz.