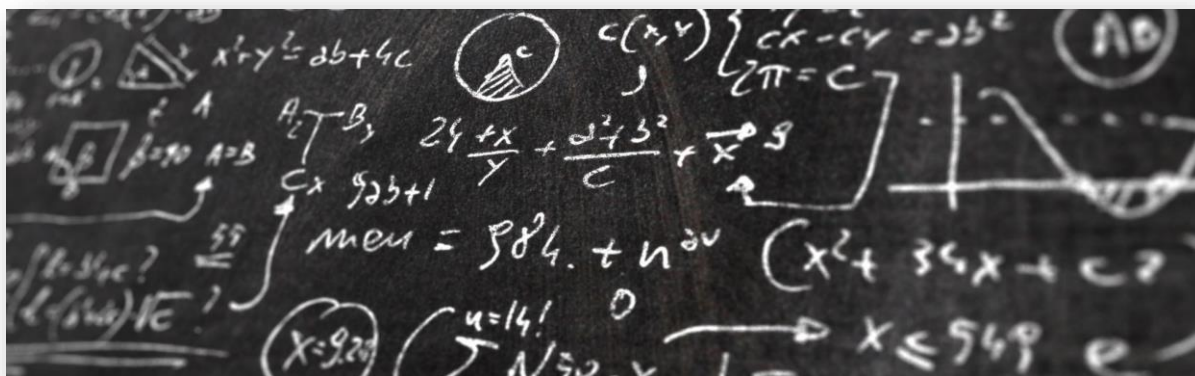


2024  
2025

# Chiffrement - Déchiffrement

RSX112 – SECURITE DES RESEAUX  
STEPHANE LARCHER



# Atelier PKI

## Sécurité en Mode Survie

Guide d'Installation PKI - Atelier RSX112.....	3
Document Étudiant - Groupe ____ .....	3
📄 Informations de Votre Groupe.....	3
🚀 Phase 1 : Connexion et Configuration Initiale .....	3
Étape 1.1 : Première Connexion .....	3
Étape 1.2 : Vérification de l'Infrastructure.....	3
Étape 1.3 : Configuration de l'Environnement de Travail .....	4
Étape 1.4 : Installation des Outils Manquants.....	4
🔧 Phase 2 : Construction de la PKI (40 min).....	4
Étape 2.1 : Connexion à la CA et Préparation .....	4
Étape 2.2 : Configuration OpenSSL.....	5
Étape 2.3 : Création de la CA Racine .....	7
Étape 2.4 : Script d'Automatisation .....	8
Atelier PKI Multi-Groupes : Infrastructure Légère et Scalable.....	11
Durée : 2 heures   10 groupes de 3 étudiants   Proxmox OVH .....	2
Vue d'Ensemble de l'Architecture .....	11
Ressources par groupe (optimisées).....	11
Architecture globale.....	11

Tableau de répartition des ressources .....	11
Préparation de l'Infrastructure (À faire avant l'atelier) .....	2
Script de déploiement automatisé pour l'enseignant .....	2
Configuration pfSense partagé.....	2
Partie 1 : Découverte de l'Infrastructure (20 minutes).....	13
Pour chaque groupe d'étudiants .....	13
Activité 1.1 : Prise en main (10 min).....	13
Activité 1.2 : Comprendre les contraintes (10 min) .....	14
Partie 2 : Construction de la PKI Minimaliste (40 minutes) .....	15
Étape 2.1 : CA Ultra-Légère (15 min).....	15
Étape 2.2 : Automatisation pour Économiser les Ressources (10 min).....	16
Étape 2.3 : Déploiement sur le Serveur Web (15 min) .....	17
Partie 3 : Scénarios Adaptés aux Contraintes (40 minutes).....	18
Scénario 1 : Gestion d'Incident avec Ressources Limitées (20 min) .....	18
Scénario 2 : Cross-Certification entre Groupes (20 min) .....	20
Partie 4 : Optimisations et Monitoring (15 minutes) .....	21
Monitoring des Ressources.....	21
Optimisations Finales .....	22
Débriefing et Évaluation (5 minutes) .....	23
Questions de Réflexion.....	23
Livrables par Groupe .....	23
Barème d'Évaluation .....	24
Annexe : Commandes de Supervision pour l'Enseignant.....	24
Monitoring Global des Groupes .....	24
Reset d'un Groupe .....	24
Sauvegarde des Travaux.....	25
Notes pour l'Enseignant.....	25
Timing Optimisé.....	25
Points d'Attention .....	26
Adaptations Possibles .....	26

# Guide d'Installation PKI - Atelier RSX112

Document Étudiant - Groupe \_\_\_\_

## Informations de Votre Groupe

**Complétez ces informations :**

- Numéro de groupe : \_\_\_\_
- VLAN : 11\_\_
- Subnet : 10.1\_\_\_.0.0/24
- Membres :
  - PKI Admin : \_\_\_\_\_
  - Web Admin : \_\_\_\_\_
  - Security Officer : \_\_\_\_\_

## Phase 1 : Connexion et Configuration Initiale

### Étape 1.1 : Première Connexion

# Depuis votre poste de travail, connectez-vous au conteneur Admin

# Remplacez X par votre numéro de groupe

ssh [pkilab@10.1X.0.4](mailto:pkilab@10.1X.0.4)

# Mot de passe : pkilab2024

✓ **Checkpoint** : Vous devez voir le prompt pkilab@GX-Client-Admin:~\$

### Étape 1.2 : Vérification de l'Infrastructure

# Test de connectivité vers vos autres conteneurs

ping -c 2 10.1X.0.2 # PKI-CA

ping -c 2 10.1X.0.3 # Web-Server

# Vérification des ressources disponibles

free -h        # Mémoire disponible

df -h        # Espace disque

nproc        # Nombre de CPU (devrait être 1)

✓ **Checkpoint** : Les 2 pings doivent répondre

## Étape 1.3 : Configuration de l'Environnement de Travail

```
# Création de la structure de dossiers
mkdir -p ~/pki-lab/{scripts,certs,docs,backup}
cd ~/pki-lab
```

```
# Création d'un fichier de notes
cat > docs/README.md << EOF
# PKI Lab - Groupe $USER
Date: $(date)
Membres: [À compléter]
```

```
## Journal de bord
- $(date "+%H:%M") : Début de l'atelier
EOF
```

```
# Configuration Git pour le suivi (optionnel mais recommandé)
git config --global user.name "Groupe X"
git config --global user.email "groupeX@pkilab.local"
cd ~/pki-lab && git init
```

## Étape 1.4 : Installation des Outils Manquants

```
# Vérification des outils installés
which openssl nginx curl wget
```

```
# Si des outils manquent (ne devrait pas arriver)
sudo apk update && sudo apk add openssl nginx curl wget
```

## Phase 2 : Construction de la PKI (40 min)

### Étape 2.1 : Connexion à la CA et Préparation

➤ **Personne responsable** : PKI Admin

```
# Depuis Admin, connexion à la CA  
ssh pkilab@10.1X.0.2
```

```
# Création de la structure PKI  
mkdir -p ~/mini-ca/{private,certs,crl,csr,newcerts}  
cd ~/mini-ca  
chmod 700 private
```

```
# Initialisation des fichiers de base  
touch index.txt  
echo 01 > serial  
echo 01 > crlnumber
```

## Étape 2.2 : Configuration OpenSSL

```
# Création du fichier de configuration  
cat > openssl.cnf << 'EOF'  
# Configuration OpenSSL Minimale pour Groupe X  
[ ca ]  
default_ca = CA_default
```

```
[ CA_default ]  
# Répertoires  
dir = .  
certs = $dir/certs  
crl_dir = $dir/crl  
new_certs_dir = $dir/newcerts  
database = $dir/index.txt  
serial = $dir/serial  
RANDFILE = $dir/private/.rand
```

```
# Certificat et clé de la CA  
private_key = $dir/private/ca.key  
certificate = $dir/certs/ca.crt
```

```
# Politique et paramètres  
name_opt = ca_default  
cert_opt = ca_default  
default_days = 365  
preserve = no
```

```
policy      = policy_loose
default_md   = sha256
```

```
[ policy_loose ]
# Politique permissive pour l'atelier
countryName      = optional
stateOrProvinceName = optional
localityName     = optional
organizationName  = optional
organizationalUnitName = optional
commonName       = supplied
emailAddress      = optional
```

```
[ req ]
# Paramètres pour les requêtes
default_bits      = 2048
distinguished_name = req_distinguished_name
string_mask       = utf8only
default_md         = sha256
```

```
[ req_distinguished_name ]
# Champs du DN
countryName      = Pays (2 lettres)
stateOrProvinceName = Région
localityName     = Ville
0.organizationName = Organisation
organizationalUnitName = Département
commonName       = Common Name
emailAddress      = Email
```

```
# Valeurs par défaut
countryName_default      = FR
stateOrProvinceName_default = IDF
localityName_default     = Paris
0.organizationName_default = PKI Lab Groupe X
```

```
[ v3_ca ]
# Extensions pour certificat CA
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true
```

```
keyUsage = critical, digitalSignature, cRLSign, keyCertSign
```

```
[ v3_intermediate_ca ]
```

```
# Extensions pour CA intermédiaire
```

```
subjectKeyIdentifier = hash
```

```
authorityKeyIdentifier = keyid:always,issuer
```

```
basicConstraints = critical, CA:true, pathlen:0
```

```
keyUsage = critical, digitalSignature, cRLSign, keyCertSign
```

```
[ usr_cert ]
```

```
# Extensions pour certificats utilisateur
```

```
basicConstraints = CA:FALSE
```

```
nsCertType = client, email
```

```
nsComment = "Certificat Utilisateur Groupe X"
```

```
subjectKeyIdentifier = hash
```

```
authorityKeyIdentifier = keyid,issuer
```

```
keyUsage = critical, nonRepudiation, digitalSignature, keyEncipherment
```

```
extendedKeyUsage = clientAuth, emailProtection
```

```
[ server_cert ]
```

```
# Extensions pour certificats serveur
```

```
basicConstraints = CA:FALSE
```

```
nsCertType = server
```

```
nsComment = "Certificat Serveur Groupe X"
```

```
subjectKeyIdentifier = hash
```

```
authorityKeyIdentifier = keyid,issuer:always
```

```
keyUsage = critical, digitalSignature, keyEncipherment
```

```
extendedKeyUsage = serverAuth
```

```
EOF
```

## Étape 2.3 : Création de la CA Racine

```
# Génération de la clé privée CA (2048 bits pour économiser les ressources)
```

```
openssl genrsa -aes256 -out private/ca.key 2048
```

```
# Mot de passe suggéré : GroupeX_CA_2024!
```

```
# Protection de la clé
```

```
chmod 400 private/ca.key
```

```
# Génération du certificat auto-signé
```



```
openssl req -config openssl.cnf \  
-key private/ca.key \  
-new -x509 -days 1825 -sha256 -extensions v3_ca \  
-out certs/ca.crt \  
-subj "/C=FR/ST=IDF/L=Paris/O=PKI Lab/OU=Groupe X/CN=Groupe X Root CA"
```

# Vérification

```
openssl x509 -noout -text -in certs/ca.crt | grep -A2 "Subject:"
```

✓ **Checkpoint** : Vous devez voir "CN = Groupe X Root CA"

## Étape 2.4 : Script d'Automatisation

# Création d'un script pour faciliter l'émission de certificats

```
cat > issue-cert.sh << 'EOF'
```

```
#!/bin/bash
```

```
# Script d'émission de certificat - Groupe X
```

```
set -e # Arrêt en cas d'erreur
```

```
# Vérification des arguments
```

```
if [ $# -lt 2 ]; then
```

```
    echo "Usage: $0 <type> <nom>"
```

```
    echo " type: server ou user"
```

```
    echo " nom: nom du certificat (sans espaces)"
```

```
    exit 1
```

```
fi
```

```
TYPE=$1
```

```
NAME=$2
```

```
echo "=== Émission d'un certificat $TYPE pour $NAME ==="
```

```
# Génération de la clé privée
```

```
echo "[1/4] Génération de la clé privée..."
```

```
openssl genrsa -out private/${NAME}.key 2048
```

```
chmod 400 private/${NAME}.key
```

```
# Création de la CSR
```

```
echo "[2/4] Création de la demande de certificat..."
```

```

if [ "$TYPE" = "server" ]; then
    # Pour un serveur, on ajoute des SAN
    cat > /tmp/${NAME}.cnf << END
[req]
distinguished_name = req_distinguished_name
req_extensions = v3_req

[req_distinguished_name]

[v3_req]
subjectAltName = @alt_names

[alt_names]
DNS.1 = ${NAME}
DNS.2 = ${NAME}.groupeX.local
DNS.3 = localhost
IP.1 = 10.1X.0.3
IP.2 = 127.0.0.1
END

    openssl req -config /tmp/${NAME}.cnf -new -key private/${NAME}.key \
        -out csr/${NAME}.csr \
        -subj "/C=FR/ST=IDF/L=Paris/O=PKI Lab/OU=Groupe
X/CN=${NAME}.groupeX.local"

    rm -f /tmp/${NAME}.cnf
else
    # Pour un utilisateur
    openssl req -new -key private/${NAME}.key \
        -out csr/${NAME}.csr \
        -subj "/C=FR/ST=IDF/L=Paris/O=PKI Lab/OU=Groupe
X/CN=${NAME}/emailAddress=${NAME}@groupeX.local"
fi

# Signature du certificat
echo "[3/4] Signature par la CA..."
openssl ca -batch -config openssl.cnf \
    -extensions ${TYPE}_cert \
    -days 365 -notext -md sha256 \
    -in csr/${NAME}.csr \
    -out certs/${NAME}.crt

```

```
# Création d'un bundle PEM  
echo
```



2	111	10.11.0.0/24	121	122	123	1024 MB
3	112	10.12.0.0/24	131	132	133	1024 MB
...	...	...	...	...	...	...
10	119	10.19.0.0/24	201	202	203	1024 MB

**Total : ~12 GB RAM + 2 GB pfSense = 14 GB RAM utilisés**

# Partie 1 : Découverte de l'Infrastructure

## Pour chaque groupe d'étudiants

### Distribution des accès :

# Fiche de connexion pour le Groupe X

=====

GROUPE X - PKI LAB

=====

VLAN: 11X

Réseau: 10.1X.0.0/24

Conteneurs:

- PKI-CA: 10.1X.0.2 (CT1X1)

- Web-Server: 10.1X.0.3 (CT1X2)

- Client: 10.1X.0.4 (CT1X3)

Credentials:

- User: pkilab

- Pass: pkilab2024

Première connexion:

ssh pkilab@[IP\_PROXMOX] -p [PORT\_FORWARD\_GROUPE\_X]

=====

## Activité 1.1 : Prise en main

### Sur le conteneur Client-Admin :

# Connexion au client admin

ssh [pkilab@10.1X.0.4](ssh://pkilab@10.1X.0.4)

# Vérifier la connectivité

ping -c 2 10.1X.0.2 # PKI-CA

ping -c 2 10.1X.0.3 # Web-Server

ping -c 2 8.8.8.8 # Internet

# Explorer l'environnement minimal

df -h # Espace disque limité

free -h # RAM limitée

nproc # 1 CPU

```
# Créer l'environnement de travail
mkdir -p ~/pki-lab/{scripts,certs,docs}
cd ~/pki-lab
```

## Activité 1.2 : Comprendre les contraintes (10 min)

### Discussion en groupe :

- "Avec seulement 256 MB de RAM pour la CA, quelles optimisations sont nécessaires ?"
- "Comment gérer efficacement l'espace disque limité ?"
- "Quels services sont vraiment essentiels ?"

### Mini-exercice : Optimisation mémoire

```
# Surveiller l'utilisation mémoire
while true; do
  clear
  echo "=== Utilisation Mémoire ==="
  free -h
  echo ""
  echo "=== Top 5 Processus ==="
  ps aux --sort=-%mem | head -6
  sleep 2
done
```

## Partie 2 : Construction de la PKI Minimaliste

### Étape 2.1 : CA Ultra-Légère

Sur le conteneur PKI-CA (CT1X1) :

```
# Connexion à la CA
ssh pkilab@10.1X.0.2

# Configuration OpenSSL minimale
mkdir -p ~/mini-ca/{private,certs,crl,csr}
cd ~/mini-ca

# Configuration simplifiée mais fonctionnelle
cat > openssl-minimal.cnf << 'EOF'
[ ca ]
default_ca = CA_default

[ CA_default ]
dir          = .
certs        = $dir/certs
new_certs_dir = $dir/certs
database     = $dir/index.txt
serial       = $dir/serial
private_key   = $dir/private/ca.key
certificate   = $dir/certs/ca.crt
default_md    = sha256
policy        = policy_anything
default_days  = 365

[ policy_anything ]
countryName    = optional
stateOrProvinceName = optional
localityName    = optional
organizationName = optional
organizationalUnitName = optional
commonName      = supplied
emailAddress     = optional

[ req ]
default_bits    = 2048
```



```
distinguished_name = req_distinguished_name
x509_extensions    = v3_ca
```

```
[ req_distinguished_name ]
```

```
[ v3_ca ]
basicConstraints = critical,CA:TRUE
keyUsage = critical,keyCertSign,cRLSign
```

```
[ server_cert ]
basicConstraints = CA:FALSE
keyUsage = critical,digitalSignature,keyEncipherment
extendedKeyUsage = serverAuth
subjectAltName = @alt_names
```

```
[ alt_names ]
DNS.1 = localhost
DNS.2 = web.groupeX.local
IP.1 = 10.1X.0.3
EOF
```

```
# Initialisation rapide
touch index.txt
echo 01 > serial
```

```
# Génération de la CA (clé plus petite pour économiser les ressources)
openssl genrsa -out private/ca.key 2048
chmod 400 private/ca.key
```

```
openssl req -new -x509 -days 1825 -key private/ca.key \
-out certs/ca.crt -config openssl-minimal.cnf \
-subj "/CN=Groupe$X CA/O=PKI Lab/C=FR"
```

```
echo "CA créée avec succès!"
ls -la certs/ca.crt
```

## Étape 2.2 : Automatisation pour Économiser les Ressources

```
# Script d'émission de certificats optimisé
cat > issue-cert.sh << 'EOF'
```

```
#!/bin/bash
# Usage: ./issue-cert.sh <hostname>

HOSTNAME=$1
if [ -z "$HOSTNAME" ]; then
    echo "Usage: $0 <hostname>"
    exit 1
fi

# Génération de clé (RSA 2048 au lieu de 4096)
openssl genrsa -out private/$HOSTNAME.key 2048

# CSR
openssl req -new -key private/$HOSTNAME.key \
    -out csr/$HOSTNAME.csr \
    -subj "/CN=$HOSTNAME/O=Groupe X/C=FR"

# Signature
openssl ca -batch -config openssl-minimal.cnf \
    -extensions server_cert \
    -out certs/$HOSTNAME.crt \
    -infiles csr/$HOSTNAME.csr

# Nettoyage pour économiser l'espace
rm -f csr/$HOSTNAME.csr
gzip certs/$HOSTNAME.crt
gzip private/$HOSTNAME.key

echo "Certificat émis: certs/$HOSTNAME.crt.gz"
EOF

chmod +x issue-cert.sh

# Test
./issue-cert.sh web-server
```

## Étape 2.3 : Déploiement sur le Serveur Web

Sur le conteneur Web-Server (CT1X2) :

```
# Configuration Nginx ultra-légère
ssh pkilab@10.1X.0.3

# Récupération des certificats
mkdir -p /home/pkilab/ssl
scp pkilab@10.1X.0.2:~/mini-ca/certs/web-server.crt.gz ~/ssl/
scp pkilab@10.1X.0.2:~/mini-ca/private/web-server.key.gz ~/ssl/
cd ~/ssl && gunzip *.gz

# Configuration Nginx minimale
sudo tee /etc/nginx/conf.d/ssl.conf << 'EOF'
server {
    listen 443 ssl;
    server_name web.groupeX.local;

    ssl_certificate /home/pkilab/ssl/web-server.crt;
    ssl_certificate_key /home/pkilab/ssl/web-server.key;

    # Optimisations pour environnement contraint
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout 5m;

    location / {
        return 200 "Groupe X - PKI Lab - Serveur Sécurisé\n";
        add_header Content-Type text/plain;
    }
}
EOF

# Vérification syntaxe et démarrage
sudo nginx -t
sudo rc-service nginx start
```

## Partie 3 : Scénarios Adaptés aux Contraintes

### Scénario 1 : Gestion d'Incident avec Ressources Limitées

**Simulation** : Un certificat a été compromis !

```

# Sur Client-Admin (CT1X3)
cd ~/pki-lab/scripts

# Outil de réponse d'incident léger
cat > incident-response-lite.sh << 'EOF'
#!/bin/bash

echo "=== INCIDENT RESPONSE TOOLKIT (Lite) ==="
echo "Optimisé pour environnements contraints"
echo ""

# 1. Isolation réseau (simulée)
echo "[1] Isolation réseau du serveur compromis"
echo "  # iptables -A INPUT -s 10.1X.0.3 -j DROP"
echo "  (Non exécuté en lab)"

# 2. Collecte d'informations minimale
echo -e "\n[2] Collecte d'informations"
mkdir -p ../incident-$(date +%Y%m%d)
cd ../incident-$(date +%Y%m%d)

# Inventaire des certificats (version légère)
echo -e "\n[*] Inventaire des certificats actifs:"
ssh pkilab@10.1X.0.2 "cd ~/mini-ca && cat index.txt" > cert-inventory.txt
cat cert-inventory.txt

# 3. Révocation
echo -e "\n[3] Révocation du certificat compromis"
read -p "Entrer le numéro de série à révoquer: " SERIAL

ssh pkilab@10.1X.0.2 << REVOKE
cd ~/mini-ca
openssl ca -config openssl-minimal.cnf -revoke certs/$SERIAL.pem
openssl ca -config openssl-minimal.cnf -gencrl -out crl/ca.crl
echo "CRL générée: $(wc -c < crl/ca.crl) bytes"
REVOKE

# 4. Notification (version simple)
echo -e "\n[4] Notifications"
cat > notification.txt << MSG
ALERTE SÉCURITÉ - Groupe X

```

=====

Date: \$(date)  
Certificat révoqué: \$SERIAL  
Action: Mettre à jour la CRL  
Contact: pkilab@groupeX  
MSG

echo "Notification créée: notification.txt"

# 5. Surveillance mémoire pendant l'incident  
echo -e "\n[5] Impact sur les ressources:"  
free -h  
df -h ~/  
EOF

chmod +x incident-response-lite.sh  
./incident-response-lite.sh

## Scénario 2 : Cross-Certification entre Groupes (20 min)

**Exercice collaboratif :** Les groupes X et Y doivent établir une confiance mutuelle

# Préparation sur le groupe X  
cd ~/pki-lab

# 1. Exporter le certificat CA  
scp [pkilab@10.1X.0.2:~/mini-ca/certs/ca.crt](mailto:pkilab@10.1X.0.2:~/mini-ca/certs/ca.crt) ~/groupeX-ca.crt

# 2. Script d'échange sécurisé  
cat > exchange-ca.sh << 'EOF'  
#!/bin/bash  
# Échange sécurisé de CA entre groupes

PARTNER\_GROUP=\$1  
PARTNER\_IP="10.1\${PARTNER\_GROUP}.0.4"

echo "Établissement de confiance avec Groupe \$PARTNER\_GROUP"

# Envoi de notre CA  
echo "[>] Envoi de notre CA..."  
scp ~/groupeX-ca.crt [pkilab@\\$PARTNER\\_IP:~/partner-ca.crt](mailto:pkilab@$PARTNER_IP:~/partner-ca.crt)

```

# Réception de leur CA
echo "[<] Réception de la CA partenaire..."
scp pkilab@\$PARTNER IP:~/groupe\${PARTNER\_GROUP}-ca.crt ~/

# Création du bundle
echo "[+] Création du bundle de confiance..."
cat ~/groupeX-ca.crt ~/groupe${PARTNER_GROUP}-ca.crt > ~/trust-bundle.crt

# Test de validation croisée
echo "[✓] Test de validation..."
openssl verify -CAfile ~/trust-bundle.crt -partial_chain ~/test-cert.crt

echo "Confiance établie avec Groupe $PARTNER_GROUP!"
EOF

chmod +x exchange-ca.sh

```

#### Test de communication sécurisée inter-groupes :

```

# Après établissement de la confiance
curl --cacert ~/trust-bundle.crt https://10.1Y.0.3/

```

## Partie 4 : Optimisations et Monitoring

### Monitoring des Ressources

```

# Script de monitoring léger
cat > monitor-pki.sh << 'EOF'
#!/bin/bash

# Dashboard simple
while true; do
    clear
    echo "=== PKI Monitor - Groupe X ==="
    echo "Time: $(date)"
    echo ""

```

```

# État des conteneurs
echo "[Conteneurs]"
for ip in 2 3 4; do
    if ping -c 1 -W 1 10.1X.0.$ip >/dev/null 2>&1; then
        echo " 10.1X.0.$ip: ✓ UP"
    else
        echo " 10.1X.0.$ip: ✗ DOWN"
    fi
done

# Ressources CA
echo -e "\n[CA Resources]"
ssh pkilab@10.1X.0.2 "free -h | grep Mem" 2>/dev/null || echo " CA unreachable"

# Certificats
echo -e "\n[Certificates]"
ssh pkilab@10.1X.0.2 "cd ~/mini-ca && ls certs/*.crt 2>/dev/null | wc -l" 2>/dev/null |
\
    xargs -l {} echo " Active certificates: {}"

# Espace disque
echo -e "\n[Disk Usage]"
df -h / | tail -1 | awk '{print " Used: "$3" / "$2 (" "$5")"}'

sleep 5
done
EOF

chmod +x monitor-pki.sh

```

## Optimisations Finales

```

# Nettoyage et optimisation
cat > optimize-pki.sh << 'EOF'
#!/bin/bash

echo "=== Optimisation PKI ==="

# 1. Compression des logs

```

```
find ~/mini-ca -name "*.log" -exec gzip {} \;
```

```
# 2. Nettoyage des fichiers temporaires
```

```
rm -f ~/mini-ca/csr/*
```

```
rm -f ~/mini-ca/*.old
```

```
# 3. Rotation des certificats expirés
```

```
# (script de nettoyage)
```

```
# 4. Optimisation mémoire Nginx
```

```
sudo killall -HUP nginx
```

```
echo "Optimisation terminée!"
```

```
EOF
```

## Débriefing et Évaluation

### Questions de Réflexion

#### 1. Contraintes et Solutions

- "Comment les limitations de ressources ont-elles influencé vos choix ?"
- "Quelles optimisations supplémentaires pourriez-vous implémenter ?"

#### 2. Scalabilité

- "Cette infrastructure pourrait-elle supporter 100 certificats ?"
- "Comment automatiser davantage sans augmenter les ressources ?"

#### 3. Sécurité vs Performance

- "Quels compromis de sécurité avez-vous dû faire ?"
- "Sont-ils acceptables en production ?"

### Livrables par Groupe

Chaque groupe doit fournir :

- Capture d'écran** du certificat CA
- Test HTTPS** fonctionnel
- Script d'automatisation** créé
- Rapport d'incident** (1 page)



## Barème d'Évaluation

- Infrastructure fonctionnelle : 40%
- Gestion d'incident : 30%
- Optimisations : 20%
- Travail en équipe : 10%

## Annexe : Commandes de Supervision pour l'Enseignant

### Monitoring Global des Groupes

```
#!/bin/bash
# monitor-all-groups.sh - Pour l'enseignant

echo "=== État Global des Groupes ==="
for group in {1..10}; do
    echo -n "Groupe $group: "
    vlan=$((109 + group))

    # Test rapide de chaque conteneur
    for host in 2 3 4; do
        if ping -c 1 -W 1 10.$vlan.0.$host >/dev/null 2>&1; then
            echo -n "✓ "
        else
            echo -n "X "
        fi
    done
    echo ""
done

# Utilisation mémoire totale
echo -e "\nRAM totale utilisée:"
pct list | grep -E "G[0-9]+" | awk '{sum+=$5} END {print sum/1024 " GB"}'
```

### Reset d'un Groupe

```
#!/bin/bash
# reset-group.sh <numéro_groupe>
```

```

GROUP=$1
BASE_ID=$((100 + (GROUP * 10)))

echo "Reset du groupe $GROUP..."
for i in 1 2 3; do
    pct stop $((BASE_ID + i)) 2>/dev/null
    pct rollback $((BASE_ID + i)) initial
    pct start $((BASE_ID + i))
done
echo "Groupe $GROUP réinitialisé!"

```

## Sauvegarde des Travaux

```

#!/bin/bash
# backup-group-work.sh

for group in {1..10}; do
    echo "Backup groupe $group..."
    vlan=$((109 + group))
    mkdir -p /backup/pki-lab/groupe$group

    # Sauvegarde des certificats CA
    scp -r pkilab@10.\$vlan.0.2:~/mini-ca/* \
        /backup/pki-lab/groupe$group/ 2>/dev/null || \
        echo " Groupe $group: Pas de données"
done

```

## Notes pour l'Enseignant

### Timing Optimisé

- **00:00-00:20** : Distribution accès et découverte
- **00:20-01:00** : Construction PKI
- **01:00-01:40** : Scénarios
- **01:40-01:55** : Optimisations
- **01:55-02:00** : Débriefing

## Points d'Attention

1. **Démarrage échelonné** : Ne pas démarrer tous les groupes simultanément
2. **Support** : Circuler activement, les environnements contraints peuvent frustrer
3. **Snapshots** : Faire des snapshots après chaque grande étape
4. **Plan B** : Avoir 2-3 environnements de secours pré-configurés

## Adaptations Possibles

- **Groupes avancés** : Ajouter IPv6, DANE, ou CT logs
- **Groupes en difficulté** : Fournir des scripts pré-écrits
- **Temps supplémentaire** : Ajouter la surveillance ou l'automatisation Ansible

*Infrastructure optimisée pour Proxmox avec ressources limitées*

*10 groupes × 3 conteneurs = 30 conteneurs + 1 VM pfSense*

*RAM totale : ~14 GB | Stockage : ~200 GB*