PARIS PANTHÉON-ASSAS UNIVERSITÉ

TP03 Infrastructures Cloud

Par David TEJEDA, Vincent LAGOGUE, Tom THIOULOUSE, Thomas PEUGNET, Alexis PLESSIAS

I3 – 2024-2025

Mise en place d'une IGC

Pour ce TP nous choisissons la machine 1 avec l'adresse *100.127.217.69* comme serveur PKI.

Installation des paquets et clés

Pour chaque poste de travail des membres de l'équipe on génère une clé SSH et on la copie sur les postes de travail.

Sous MAC :

On crée la paire de clé :

```
ssh-keygen -t rsa -b 4096 -C "studentlab@machine"
```

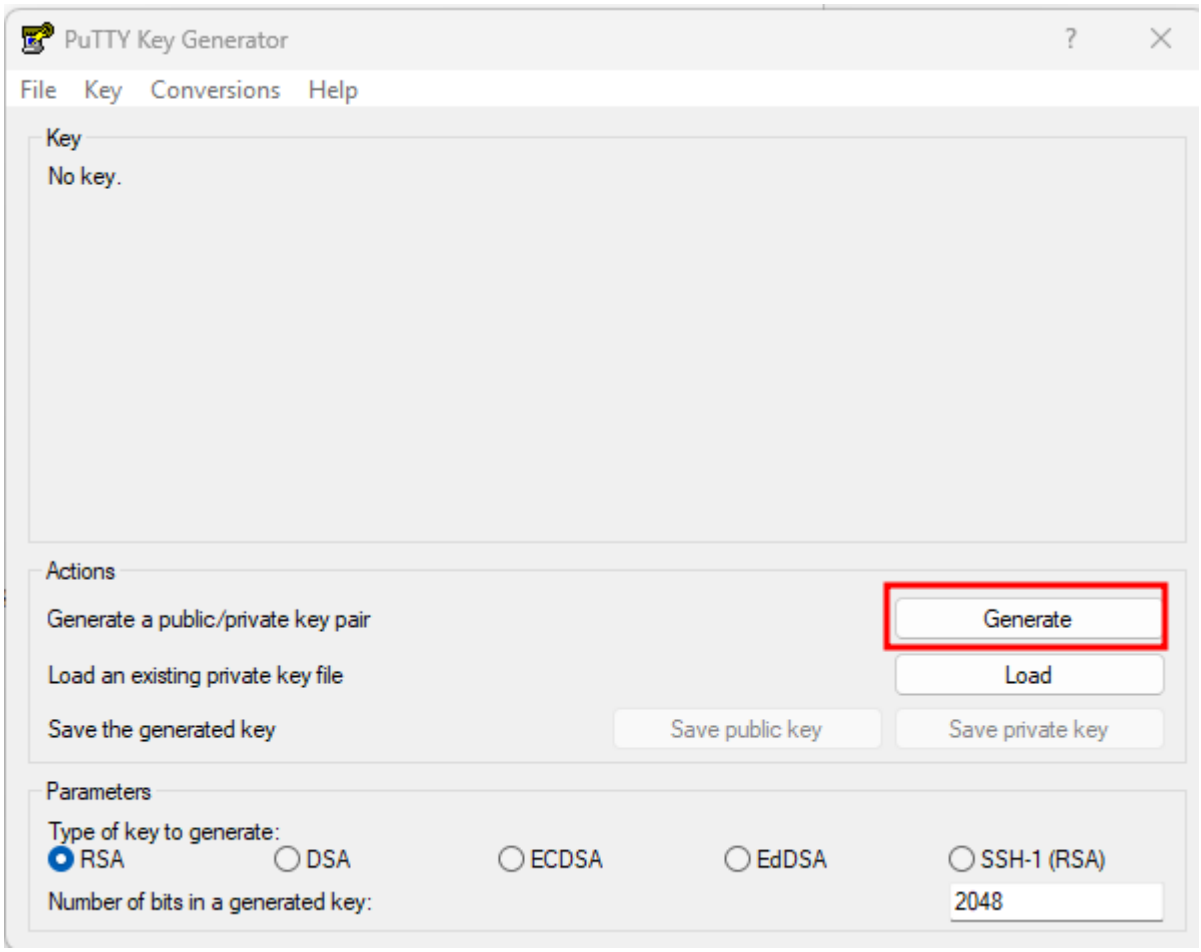
On déploie la clé publique sur les serveurs cibles :

```
ssh-copy-id studentlab@100.127.217.69
ssh-copy-id studentlab@100.98.204.70
ssh-copy-id studentlab@100.79.188.4
```

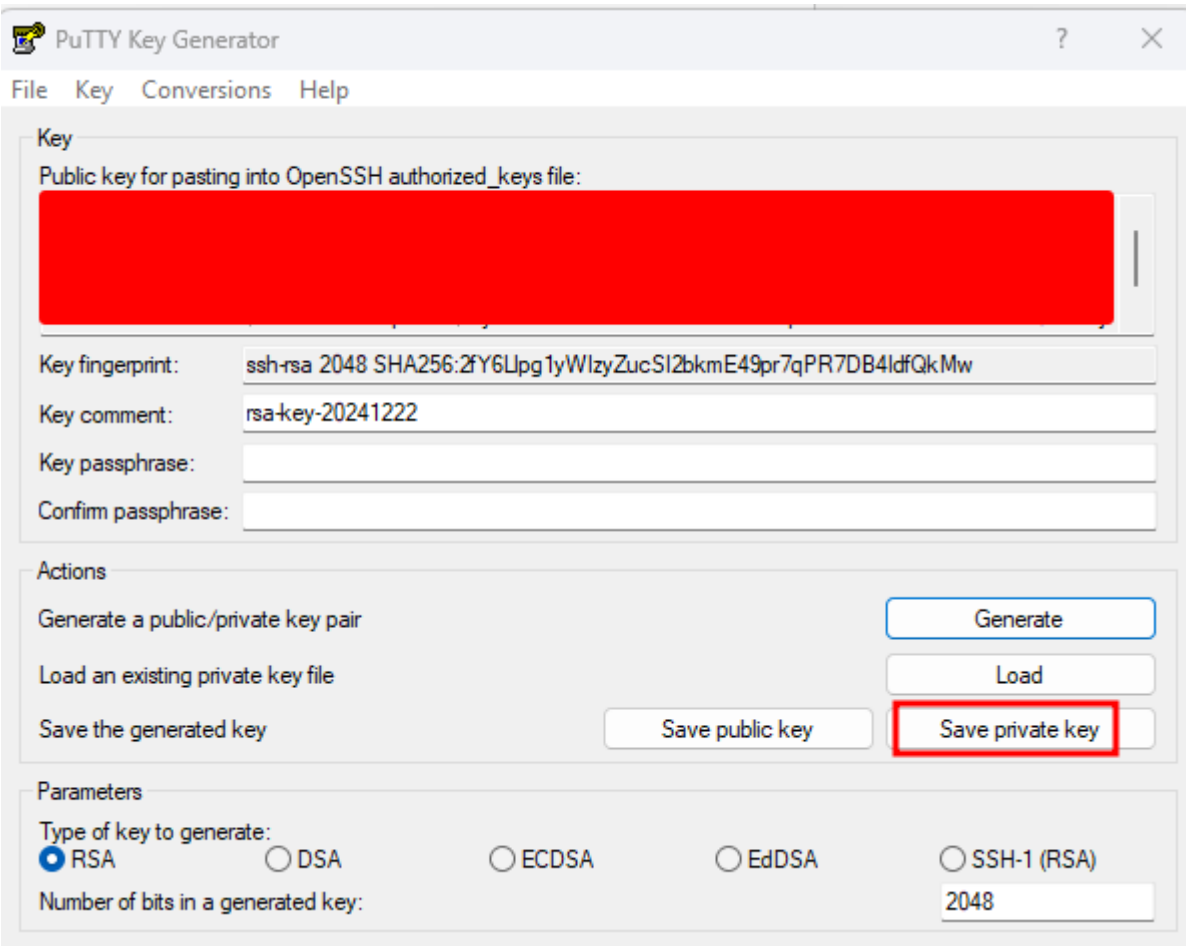
On teste ensuite la connexion avec l'utilitaire *ssh*.

Sous Windows :

On ouvre **PuTTYgen** :



On enregistre la clé privée et la clé publique sur le poste :



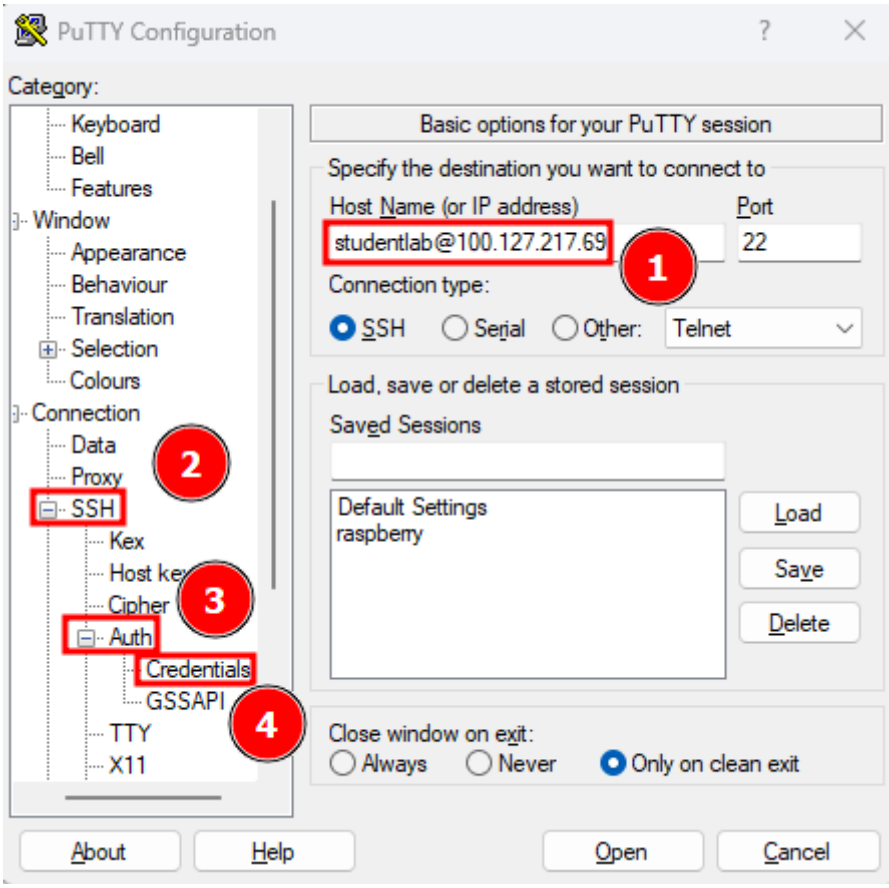
Dans la zone **Public key for pasting into OpenSSH authorized_keys file**, on copie la clé publique.

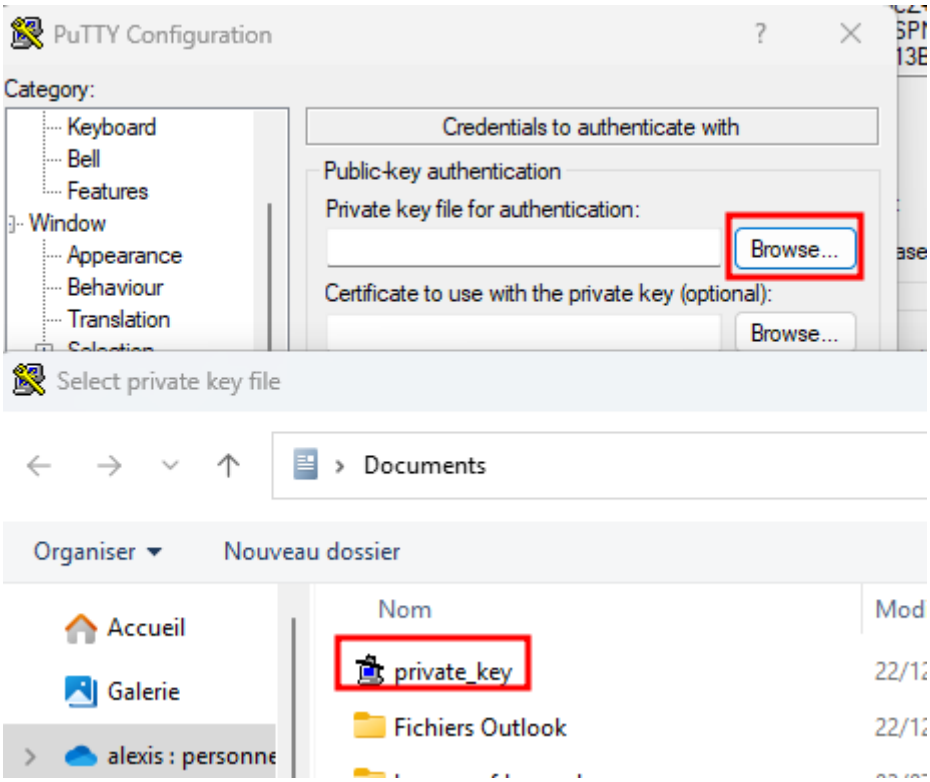
Ensuite dans **PuTTY** on se connecte aux différentes VMs. On ajoute notre clé au fichier *.ssh/authorized_keys* :

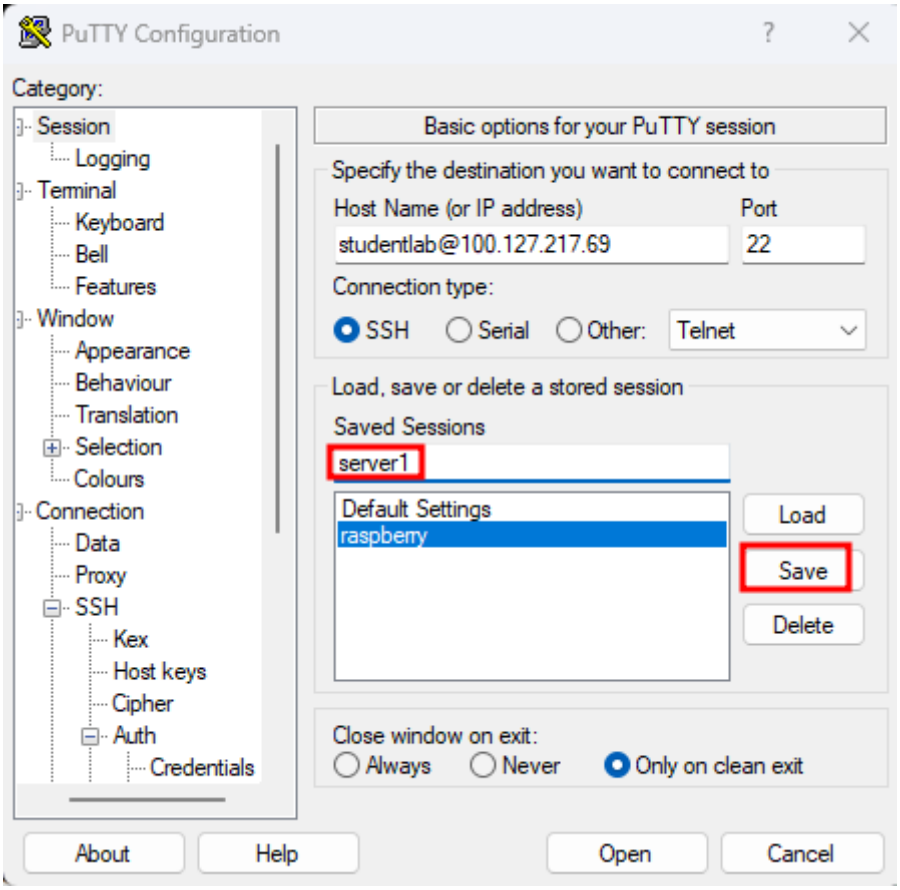
```
studentlab@1124BUBUSTD25:~$ sudo mkdir /.ssh
studentlab@1124BUBUSTD25:~$ nano .ssh/authorized_keys
```

On crée d'abord le fichier sur toutes les VMs

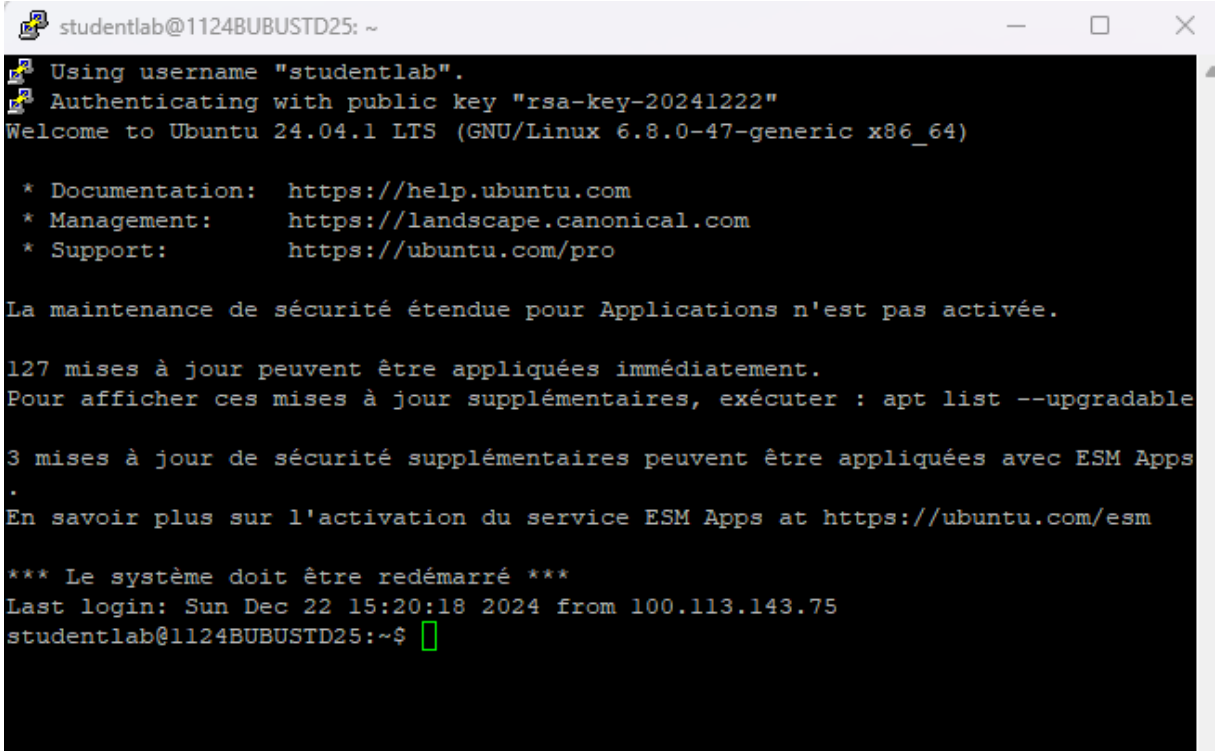
Ensuite on paramètre la session pour la connexion sans mot de passe dans **PuTTY** :


PuTTY Configuration
Category:
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
SSH
Kex
Host key
Cipher
Auth
Credentials
GSSAPI
TTY
X11
Basic options for your PuTTY session
Specify the destination you want to connect to
Host Name (or IP address)
Port
studentlab@100.127.217.69
22
Connection type:
SSH
Serial
Other:
Telnet
Load, save or delete a stored session
Saved Sessions
Default Settings
raspberry
Load
Save
Delete
Close window on exit:
Always
Never
Only on clean exit
About
Help
Open
Cancel
1
2
3
4


PuTTY Configuration
Category:
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Credentials to authenticate with
Public-key authentication
Private key file for authentication:
Browse...
Certificate to use with the private key (optional):
Browse...
Select private key file
Documents
Organiser
Nouveau dossier
Accueil
Galerie
alexis : personne
Nom
Mod
private_key
Fichiers Outlook



On sauvegarde pour des utilisations futures



La connexion est directe sans demande de mot de passe

On fait de même pour enregistrer les autres sessions des serveurs.

On installe les paquets suivants :

```
sudo apt install jq nload tcpdump tmux golang -y
```

Déploiement du service cfssl

Préparation de l'environnement

On installe les binaires dans *$HOME* :

```
go install github.com/cloudflare/cfssl/cmd/cfssl@latest
go install github.com/cloudflare/cfssl/cmd/cfssljson@latest
go install github.com/cloudflare/cfssl/cmd/multirootca@latest
```

On copie ces binaires dans */usr/local/bin*.

```
sudo cp $HOME/go/bin/cfssl /usr/local/bin/
sudo cp $HOME/go/bin/cfssljson /usr/local/bin/
sudo cp $HOME/go/bin/multirootca /usr/local/bin/
```

On crée les répertoires */var/lib/pki/{root,intermediate,config,certificates}*.

Création de la CA racine

On passe en **root** et on lance les commandes suivantes pour créer le certificat racine.

```
export PKI=/var/lib/pki
cat << EOF > $PKI/root/root-csr.json
{
"CN": "EFREI RS Root Certificate Authority",
"key": {
"algo": "ecdsa",
"size": 256
},
"names": [
{
"C": "FR",
"L": "Ivry-sur-Seine",
"O": "EFREI",
"OU": "CA Services",
"ST": "IDF"
}
],
"ca": {
"expiry": "87600h"
}
}
EOF
cfssl gencert -initca $PKI/root/root-csr.json \
| cfssljson -bare $PKI/root/root-ca
```

Création de la CA intermédiaire

```
cat << EOF > $PKI/intermediate/intermediate-csr.json
{
"CN": "EFREI RS Root Certificate Authority",
"key": {
"algo": "ecdsa",
"size": 256
},
"names": [
{
"C": "FR",
"L": "Ivry-sur-Seine",
"O": "EFREI",
"OU": "CA Services",
"ST": "IDF"
}
]
}
EOF
cfssl genkey $PKI/intermediate/intermediate-csr.json \
| cfssljson -bare $PKI/intermediate/intermediate-ca
cat << EOF > $PKI/config/config.json
{
"signing": {
"default": {
"expiry": "8760h"
},
"profiles": {
"intermediate": {
"usages": ["cert sign", "crl sign"],
"expiry": "70080h",
"ca_constraint": {
"is_ca": true,
"max_path_len": 1
}
```

```
}
}
}
}
EOF
cfssl sign \
-ca $PKI/root/root-ca.pem \
-ca-key $PKI/root/root-ca-key.pem \
-config $PKI/config/config.json \
-profile intermediate $PKI/intermediate/intermediate-ca.csr \
| cfssljson -bare $PKI/intermediate/intermediate-ca
```

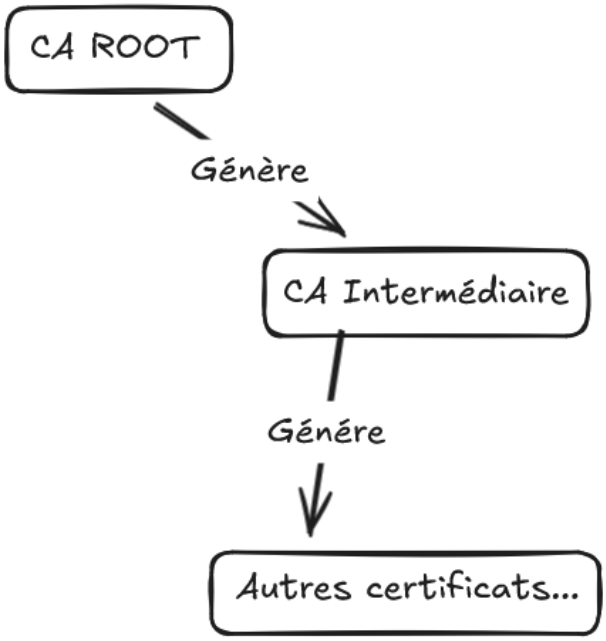


Schéma résumé de l'infrastructure PKI

Il est judicieux de créer un CA intermédiaire afin de limiter les risques de compromission de la CA root. En réduisant l'exposition du CA root et en le réservant uniquement à la signature de CA intermédiaires, on limite les risques de compromission. Ainsi, même si un CA intermédiaire est compromis, le CA root reste intact.

On pourrait mettre en place d'autres mesures de sécurité :

- Mettre des mots de passe pour les certificats
- Créer d'autres CA intermédiaires pour les catégories de certificat
- Activer l'OCSP pour permettre la révocation en temps réel des certificats compromis

Création des certificats intermédiaires

On vérifie la création d'un certificat test :

```
# Configuration du fichier config.json pour le CA
cat << EOF > $PKI/config/config.json
{
  "signing": {
    "default": {
      "expiry": "8760h"
    },
    "profiles": {
      "intermediate": {
        "usages": ["cert sign", "crl sign"],
        "expiry": "70080h",
        "ca_constraint": {
          "is_ca": true,
          "max_path_len": 1
        }
      },
      "host": {
        "usages": ["signing", "digital signing", "key encipherment", "server auth"],
        "expiry": "8760h"
      }
    }
  }
}
EOF

# Configuration d'un certificat de test
cat << EOF > $PKI/certificates/lab-efrei-csr.json
{
  "CN": "lab.efrei.io",
  "hosts": ["lab.efrei.io", "192.168.122.120"],
  "names": [
    {
      "C": "FR",
```

```
      "L": "Ivry-sur-Seine",
      "O": "EFREI",
      "OU": "CA Services"
    }
  ]
}
EOF

# Génération du certificat pour le test
cfssl gencert \
  -ca $PKI/intermediate/intermediate-ca.pem \
  -ca-key $PKI/intermediate/intermediate-ca-key.pem \
  -config $PKI/config/config.json \
  -profile host $PKI/certificates/lab-efrei-csr.json \
| cfssljson -bare $PKI/certificates/lab-efrei

# Lecture du certificat généré
openssl x509 \
  -in $PKI/certificates/lab-efrei.pem \
  -noout -subject -issuer -startdate -enddate

# Création d'un bundle complet (fullchain)
cat \
  $PKI/certificates/lab-efrei.pem \
  $PKI/intermediate/intermediate-ca.pem \
> $PKI/certificates/lab-efrei-fullchain.pem

# Vérification du certificat généré
openssl verify \
  -CAfile <(cat $PKI/root/root-ca.pem $PKI/intermediate/intermediate-ca.pem) \
  $PKI/certificates/lab-efrei.pem
```

```
2024/12/23 09:02:44 [INFO] generate received request
2024/12/23 09:02:44 [INFO] received CSR
2024/12/23 09:02:44 [INFO] generating key: ecdsa-256
2024/12/23 09:02:44 [INFO] encoded CSR
2024/12/23 09:02:44 [INFO] signed certificate with serial number 273951110833123150052091104666372858959759498599
subject=C = FR, L = Ivry-sur-Seine, O = EFREI, OU = CA Services, CN = lab.efrei.io
issuer=C = FR, ST = IDF, L = Ivry-sur-Seine, O = EFREI, OU = CA Services, CN = EFREI RS Root Certificate Authority
notBefore=Dec 23 07:58:00 2024 GMT
notAfter=Dec 23 07:58:00 2025 GMT
/var/lib/pki/certificates/lab-efrei.pem: OK
```

Lancement du webservice

Tout d'abord, on crée un certificat pour le webservice :

```
# CSR
cat << EOF > $PKI/certificates/multirootca-server-csr.json
{
  "CN": "pki.efrei.io",
  "hosts": ["pki.efrei.io", "$IP_ADDR"],
  "names": [
    {
      "C": "FR",
      "L": "Ivry-sur-Seine",
      "O": "EFREI",
      "OU": "CA Services"
    }
  ]
}
EOF

# Certificat
cfssl gencert \
  -ca $PKI/intermediate/intermediate-ca.pem \
  -ca-key $PKI/intermediate/intermediate-ca-key.pem \
  -config $PKI/config/config.json \
  -profile host $PKI/certificates/multirootca-server-csr.json \
| cfssljson -bare $PKI/certificates/multirootca-server

# Création du bundle
cat \
  $PKI/certificates/multirootca-server.pem \
  $PKI/intermediate/intermediate-ca.pem \
  > $PKI/certificates/multirootca.pem

# Sécurisation des profils avec une clé
KEY=$(openssl rand -hex 16)
echo $KEY > $PKI/auth.key
```

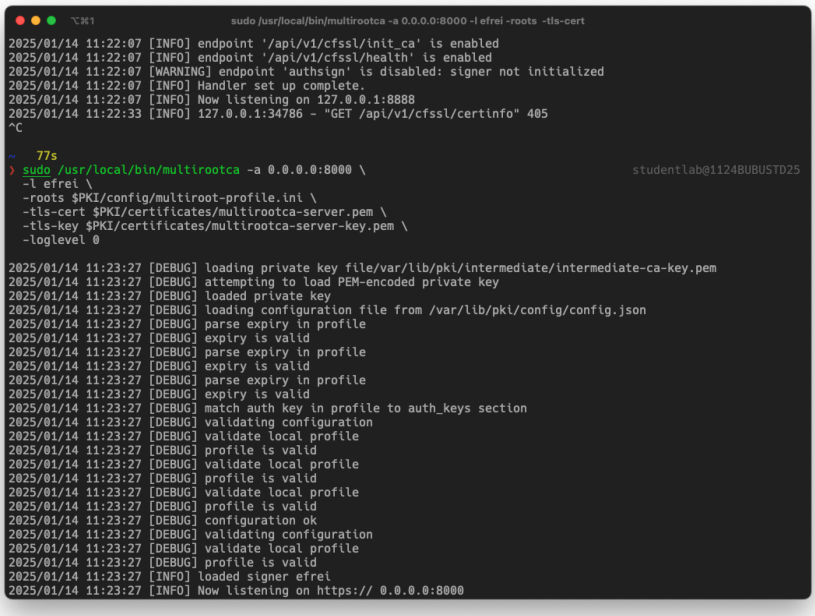
```
cat << EOF > $PKI/config/config.json
{
  "signing": {
    "default": {
      "expiry": "8760h"
    },
    "profiles": {
      "intermediate": {
        "usages": ["cert sign", "crl sign"],
        "expiry": "70080h",
        "ca_constraint": {
          "is_ca": true,
          "max_path_len": 1
        }
      },
      "host": {
        "usages": ["signing", "digital signing", "key encipherment", "server auth",
"client auth"],
        "expiry": "8760h",
        "auth_key": "default"
      }
    }
  },
  "auth_keys": {
    "default": {
      "key": "$KEY",
      "type": "standard"
    }
  }
}
EOF
```

On crée un fichier de configuration :

```
- name: PKI | Webservice | Création du fichier de configuration pour le webservice
  notify:
  - Redémarrage de la machine
  - Test du webservice par localhost
  copy:
    dest: "$PKI/config/multiroot-profile.ini"
    content: |
      [efrei]
      private = file://$PKI/intermediate/intermediate-ca-key.pem
      certificate = $PKI/intermediate/intermediate-ca.pem
      config = $PKI/config/config.json
```

On lance le service :

```
sudo /usr/local/bin/multirootca -a 0.0.0.0:8000 \
  -l efrei \
  -roots $PKI/config/multiroot-profile.ini \
  -tls-cert $PKI/certificates/multirootca-server.pem \
  -tls-key $PKI/certificates/multirootca-server-key.pem \
  -loglevel 0
```



On crée un **systemd** pour notre service :

```
[Unit]
Description=MultiRootCA
Service After=network.target

[Service]
Type=simple
ExecStart=sudo /usr/local/bin/multirootca \
    -a 0.0.0.0:8000 \
    -l efrei \ -roots /var/lib/pki/config/multiroot-profile.ini \
    -tls-cert /var/lib/pki/certificates/multirootca-server.pem \
    -tls-key /var/lib/pki/certificates/multirootca-server-key.pem \
    -loglevel 0
User=root
Restart=on-failure

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

On rend notre service disponible :

```
sudo systemctl daemon-reload
sudo systemctl enable multirootca
```

On crée la demande de certificat pour tester le webservice :

```
cd $HOME
export IP_ADDR=100.127.217.69 # IP d'écoute du webservice
export KEY="0b2b7cda34425ec452e71d8ea0fd4676" # Secret généré plus haut pour s'authentifier sur le webservice
# CSR
cat << "EOF" > my-cert-request-csr.json
{
"CN": "compute.efrei.io",
"hosts": ["$HOST_NAME", "$HOST_IP"],
8
"names": [
{
"C": "FR",
```

```
"L": "Ivry-sur-Seine",
"O": "EFREI",
"OU": "CA Services"
}
]
}
EOF
# Configuration de la demande
cat << EOF > request-profile.json
{
"signing": {
"default": {
"auth_remote": {
"remote": "ca_server",
"auth_key": "default"
}
}
},
"auth_keys": {
"default": {
"key": "$KEY",
"type": "standard"
}
},
"remotes": {
"ca_server": "https://$IP_ADDR:8000"
}
}
EOF
# Envoi de la demande
cfssl gencert \
-config ./request-profile.json \
-tls-remote-ca $PKI/intermediate/intermediate-ca.pem \
-profile host ./my-cert-request-csr.json \
| cfssljson -bare my-cert
# Vérification du certificat
```

```
openssl x509 -in ./my-cert.pem -noout -subject -issuer -startdate -enddate

cd $HOME

export IP_ADDR=100.127.217.69 # IP d'écoute du webservice
export KEY="0b2b7cda34425ec452e71d8ea0fd4676"  # Secret généré pour s'authentifier sur le webservice

# CSR
cat << "EOF" > my-cert-request-csr.json
{
  "CN": "compute.efrei.io",
  "hosts": ["$HOST_NAME", "$HOST_IP"],
  "names": [
    {
      "C": "FR",
      "L": "Ivry-sur-Seine",
      "O": "EFREI",
      "OU": "CA Services"
    }
  ]
}
EOF

# Configuration de la demande
cat << EOF > request-profile.json
{
  "signing": {
    "default": {
      "auth_remote": {
        "remote": "ca_server",
        "auth_key": "default"
      }
    }
  },
  "auth_keys": {
```

```
    "default": {
      "key": "$KEY",
      "type": "standard"
    }
  },
  "remotes": {
    "ca_server": "https://$IP_ADDR:8000"
  }
}
EOF

# Envoi de la demande
cfssl gencert \
  -config ./request-profile.json \
  -tls-remote-ca $PKI/intermediate/intermediate-ca.pem \
  -profile host ./my-cert-request-csr.json \
  | cfssljson -bare my-cert

# Vérification du certificat
openssl x509 -in ./my-cert.pem -noout -subject -issuer -startdate -enddate
```

```
⌥⌘1                    sudo /usr/local/bin/multirootca -a 0.0.0.0:8000 -l efrei -roots -tls-cert
2025/01/14 11:22:07 [INFO] Now listening on 127.0.0.1:8888
2025/01/14 11:22:33 [INFO] 127.0.0.1:34786 - "GET /api/v1/cfssl/certinfo" 405
^C

~   77s
❯ sudo /usr/local/bin/multirootca -a 0.0.0.0:8000 \                                  studentlab@1124BUBUSTD25
  -l efrei \
  -roots $PKI/config/multiroot-profile.ini \
  -tls-cert $PKI/certificates/multirootca-server.pem \
  -tls-key $PKI/certificates/multirootca-server-key.pem \
  -loglevel 0

2025/01/14 11:23:27 [DEBUG] loading private key file/var/lib/pki/intermediate/intermediate-ca-key.pem
2025/01/14 11:23:27 [DEBUG] attempting to load PEM-encoded private key
2025/01/14 11:23:27 [DEBUG] loaded private key
2025/01/14 11:23:27 [DEBUG] loading configuration file from /var/lib/pki/config/config.json
2025/01/14 11:23:27 [DEBUG] parse expiry in profile
2025/01/14 11:23:27 [DEBUG] expiry is valid
2025/01/14 11:23:27 [DEBUG] parse expiry in profile
2025/01/14 11:23:27 [DEBUG] expiry is valid
2025/01/14 11:23:27 [DEBUG] parse expiry in profile
2025/01/14 11:23:27 [DEBUG] expiry is valid
2025/01/14 11:23:27 [DEBUG] match auth key in profile to auth_keys section
2025/01/14 11:23:27 [DEBUG] validating configuration
2025/01/14 11:23:27 [DEBUG] validate local profile
2025/01/14 11:23:27 [DEBUG] profile is valid
2025/01/14 11:23:27 [DEBUG] validate local profile
2025/01/14 11:23:27 [DEBUG] profile is valid
2025/01/14 11:23:27 [DEBUG] validate local profile
2025/01/14 11:23:27 [DEBUG] profile is valid
2025/01/14 11:23:27 [DEBUG] configuration ok
2025/01/14 11:23:27 [DEBUG] validating configuration
2025/01/14 11:23:27 [DEBUG] validate local profile
2025/01/14 11:23:27 [DEBUG] profile is valid
2025/01/14 11:23:27 [INFO] loaded signer efrei
2025/01/14 11:23:27 [INFO] Now listening on https:// 0.0.0.0:8000
2025/01/14 11:30:29 [INFO] signed certificate with serial number 272584315971035977153044664655785189123365297888
2025/01/14 11:30:29 [INFO] signature: requester=100.127.217.69:38102, label=efrei, profile=host, serialno=27258431
597103597715304466465578518912336529788 8
```

On voit que le certificat a bien été signé