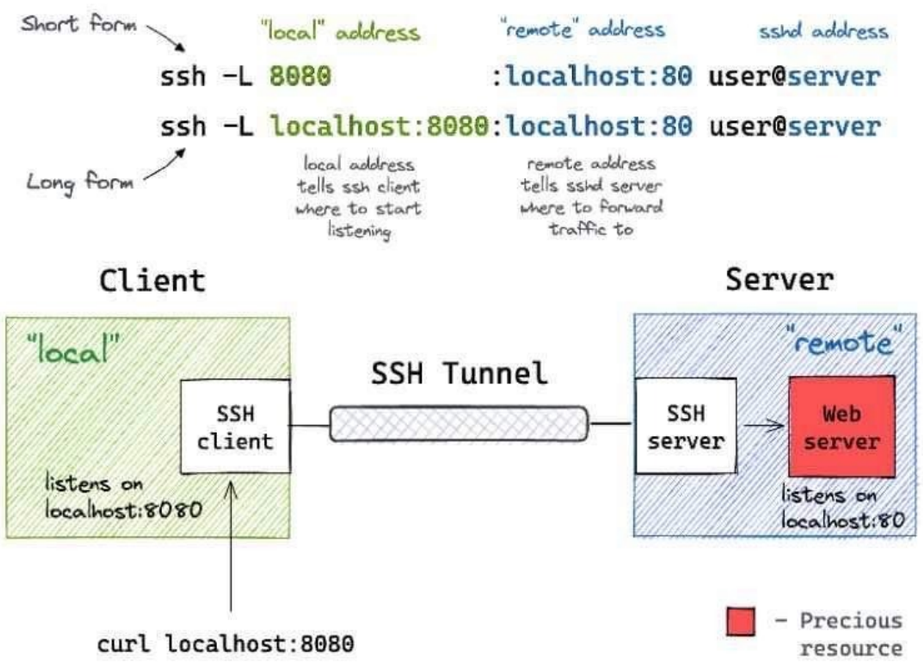
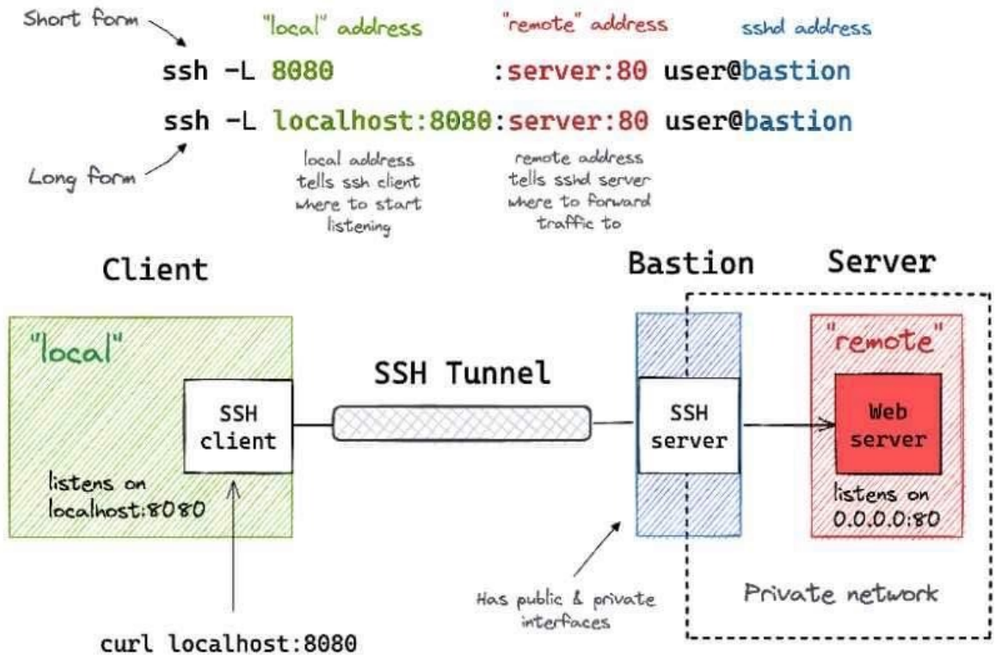
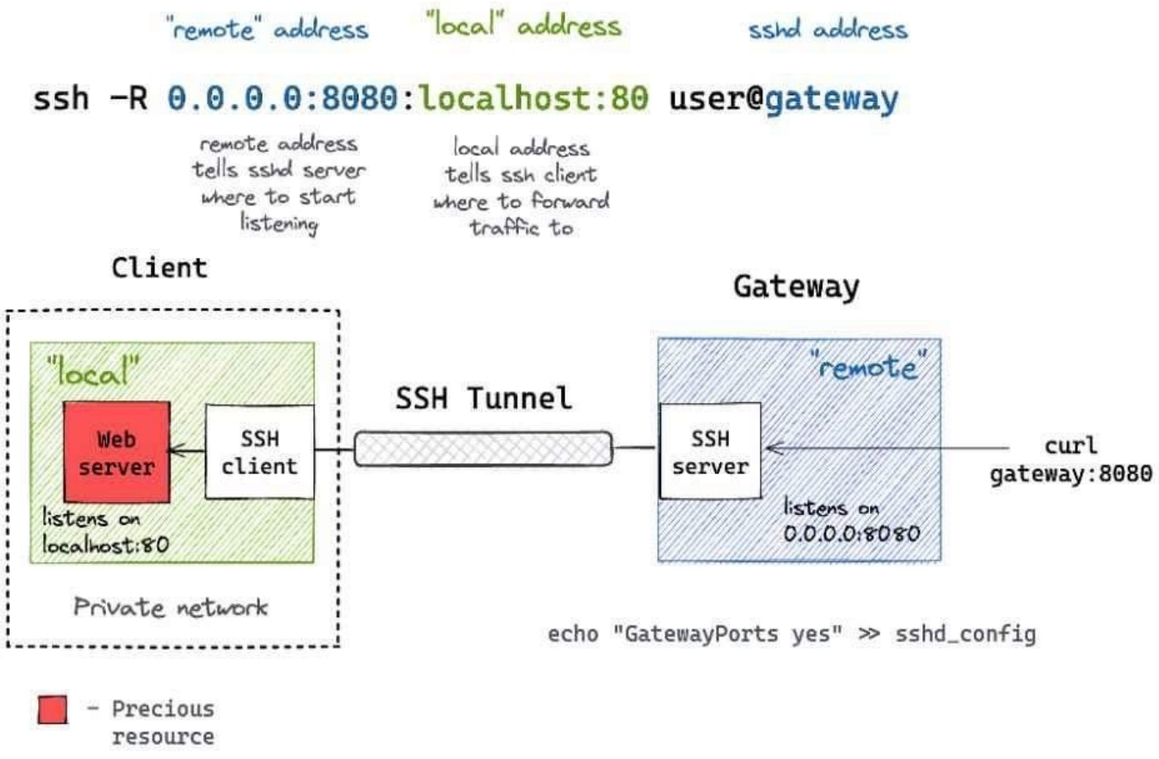
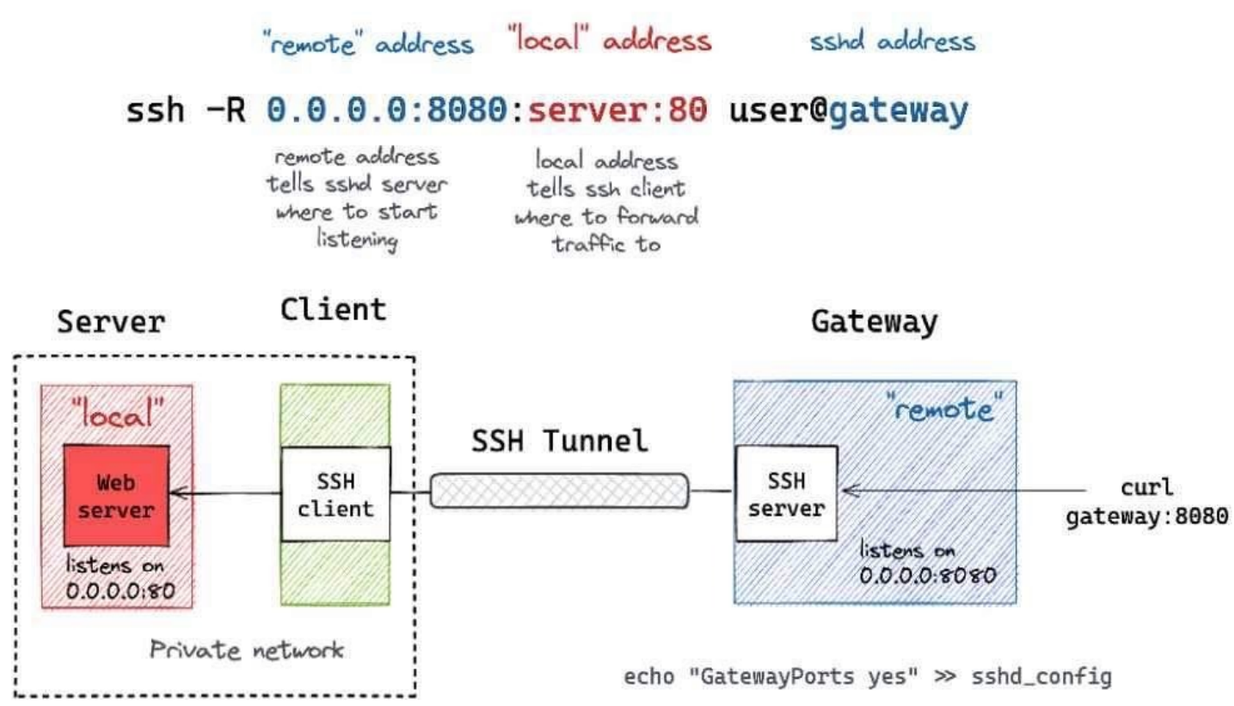








ssh offre :
- accès à une machine (shell, execution de commande)
- authentification forte
- chiffrement des échanges
- encapsulation de flux

) remplace rlogin, rsh, telnet, rcp, ftp

configuration du serveur : /etc/ssh/sshd_config
↳ auth (keys, passwd), protocol (SSHv1, v2 ...), services (sftp), adresse d'écoute (+port), users autorisés, algo de chiffrement.

configuration du client : /etc/ssh/ssh_config ⇝ surchargeable avec ~/.ssh/config
↳ paramètres de connection par défaut (host, user, clés, protocole, compression), forward (yes/no)
↳ connection par hôte ↘ évite de taper la commande ssh à chaque fois.

```
Host devserver
    HostName 192.168.42.42
    Port 2222
    User tibonan
```
⇝ ssh devserver

Authentification

Trust On First Use ⇝ TOFU

clés du serveur : /etc/ssh/ssh_host_rsa_key.pub - /etc/ssh/ssh_host_dsa_key.pub
fingerprints serveurs stockés chez le client : /etc/ssh/ssh_known_hosts ou ~/.ssh/known_hosts

- pas dans les known_hosts : first login donc on demande si c'est trustable ✓ (on conserve le fingerprint)
- known host mais clé différente : danger possible, MITM ? → refus
- known host et clé correcte ✓

ssh_config
StrictHostChecking ask

sshd_config

```
PasswordAuthentication yes
PubkeyAuthentication yes
```

→ auth par mdp
→ auth par clé
) on peut mettre les 2

→ se faire une clé SSH : ssh-keygen -t [dsa|ecdsa|ed25519|rsa] -b bits
↳ ~/.ssh/id_dsa[.pub] (ou id_rsa, id_ecdsa...)

→ rajouter les clés (pub) autorisées dans ~/.ssh/authorized_keys

```
command="dump /home",no-pty
↪ ssh-dss AAAAC3...51R==
↪ foobarteam.net
                    ~/.ssh/authorized_keys
```

[options] keytype base64-encoded key comment

- from = "pattern-list"
- command = "cat yo.txt"
- no-pty

→ restriction par @dresse
→ permet de whitelister des commandes
→ interdit shell / tty

```
Match User user Address 1.2.3.4
  ForceCommand  /path/to/script.sh
                    sshd_config
```

→ problèmes : beaucoup de clés (beaucoup d'utilisateurs), trust dans le fingerprint
→ solutions : signer les clés des machines et des utilisateurs, certificats SSH

* Signature des clés machine

- ssh-keygen -f server_ca ← créer la clé de signature
- ssh-keygen -s server_ca -I host_auth_server -h -n auth.example.com -V +52w ssh_host_rsa_key.pub
  (-s : clé pour signer ; -I : clé à signer ; -h : machine (pas user) ; -n : nom de domaine machine ; -V : validité)
- modifier /etc/ssh/sshd_config du server :

```
HostCertificate /etc/ssh/ssh_host_rsa_key-cert.pub
                    sshd_config
```

- modifier ~/.ssh/known_hosts des clients

```
>> cat ~/.ssh/known_hosts
@cert-authority *.example.com ecdsa-sha2-nistp521 AAAAA...
```

↑ signature de la clé machine

* Signature des clés utilisateurs

- ssh-keygen -f user_ca
- ssh-keygen -s user_ca -I user_str -n user -V +52w id_ecdsa.pub
  (-s : clé pour signer ; -I : clé à signe ; -n : identifiant utilisateur sur le serveur ; -V : validité)
- modifier le fichier /etc/ssh/sshd_config

```
TrustedUserCAKeys /etc/ssh/user_ca.pub
                    sshd_config
```

- on delete la clé de /home/user/.ssh/authorized_keys sur le serveur

TOOLS

- scp -r (répertoire) ./rip login@server:/home/less
- sftp login@server → il faut l'activer dans sshd_config :

```
Subsystem sftp /usr/lib/openssh/sftp-server
                    sshd_config
```

↓ pour finir (pas de scp/sftp)

- ssh login@server cat file > /tmp/file ← téléchargement
- ssh login@server < file "cat > /tmp/file" ← envoi d'un fichier sur le serv
- ssh server "tar cz /etc" | tar xz ← téléchargement compressé (- de bande passante)
- tar cz | ssh server "cat > tutu.tgz" ← envoi le dossier courant (le décompresse pas)
- ssh server cat /path/to/remote/file | diff /path/to/local/file - → diff remote/local

* ssh-agent :

- flemme de taper le mdp non ? ⇝ ssh-add → créer un socket localement → eval "ssh-agent"
  → socket stocké côté serveur : SSH_AUTH_SOCK

si machine distante compromise, le socket est détournable pour nous usurper
↳ export SSH_AUTH_SOCK = /tmp/ssh-DEAF/agent.1337

→ ssh-add -c ~/.ssh/ma_cle_prive ⇝ si le serveur demande l'accès à cette clé ça demande une interaction côté client donc c'est ⊕ sécu

* Multiplexage TCP :

Objectif : Amélioration des performances

- Principe : faire passer tous les flux TCP dans la même connexion
- Utilise la notion de Master/Slave

```
Host GetinMeForFree
    ControlMaster auto
    ControlPath ~/.ssh/currents/\%r@\%h:\%p
```

ssh_config

droits nécessaire sur le socket UNIX.

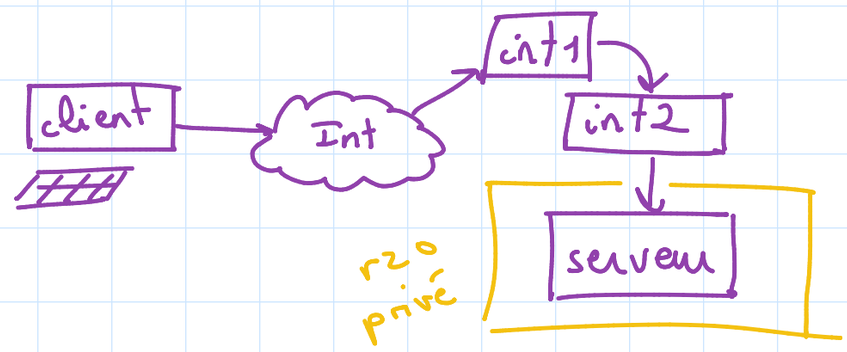
Problème

- Dès que la connexion est établie, *tous le le monde*[2] peut s'en servir sans remettre de mdp / passphrase

* Proxyjump : ssh -J intermediaire1, intermediaire2 serveur

ou dans ~/.ssh/config :

```
Host server1
    ProxyJump intermediaire1, intermediaire2
    Hostname ip_server
```



* Meta-commandes SSH :

<enter>~ donne accès à des meta-commandes

```
~.   - terminate connection (and any multiplexed sessions)
~B   - send a BREAK to the remote system
~C   - open a command line
~R   - request rekey
~V/v - decrease/increase verbosity (LogLevel)
~^Z  - suspend ssh
~#   - list forwarded connections
~&   - background ssh (when waiting for connections to terminate)
~?   - this message
~~   - send the escape character by typing it twice
```

⇝ si répond plus très pratique

* ssh -ND 1080 login@server ⇝ tunnel SOCKS → faut que l'app soit prévu pour
pas de shell ↵ ↳ proxy socks ↳ permet d'accéder à n'importe quel service sur le serveur ↳ sinon tsocks

* VPN SSH :
  - sudo ssh -N -w 0:0 root@machine2 → créer une interface tun sur le serveur et le client
  - ip addr 10.0.0.1 peer 10.0.0.2 dev tun0 → mode point à point
  - ip link set up dev tun0 → configuration du routage

⊖ il faut être root sur les 2 machines.
⊖ peu optimal niveau charge utile sur le r20.

* ssh -A popeye@server ⇝ creer le socket donc ñ probleme que ssh-agent
⇝ abréviation de ForwardAgent yes
⇝ permet d'utiliser sa clé locale sur "server"

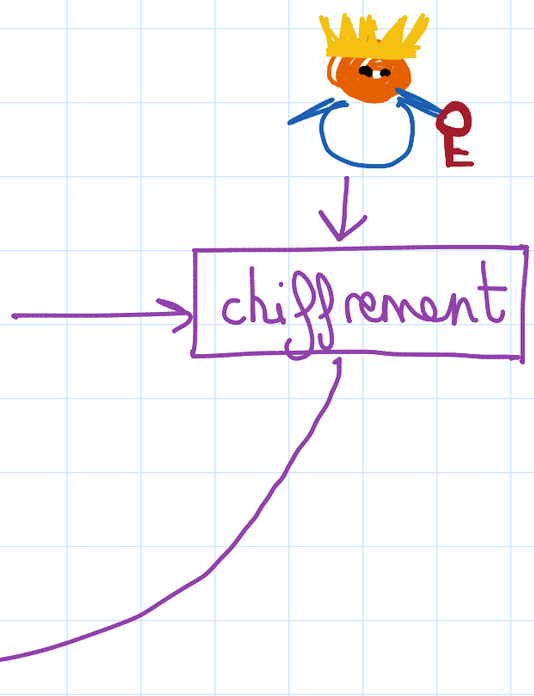
X509 ?

enregistrer auprès de l'AC (Autorité de certification) :
- ↳ créer le dossier
- ↳ envoi des infos + clé publiques (CSR - Certificate Signing Request)
- ↳ Enquête de l'AC ...
- ↳ Envoi du certificat liant la clé à l'identité de la personne (signé par l'AC)

cette signature est trusté par tout le monde (/etc/ssl/certs)

=> HIERARCHIE DES AC ~> il faut vérifier le chemin complet !!!

Certificats X509 (RFC 5280) :
- Version
- Numéro de série
- Issuer *
- Subject *
- Dates de validité (notBefore, notAfter)
- Clé publique
- Extensions
- Signature de l'emetteur
- ...



* Identification : CN : Common Name
O : Organisation
OU : Organisation Unit
L : Locality / City
ST : State / Province
C : Country

2 encodages : Distinguished Encoding Rules (DER) ou Privacy Enhanced Mail (PEM)
- ↓ encodage binaire (ASN.1)
- ↓ ASCII : base64 du DER avec
  ---BEGIN CERTIFICATE---
  ---END CERTIFICATE---

Extensions X509 : Basics Constraints, Subject Key Identifier, Subject Alternative Name, Key Usage et Extended Key Usage
- ↳ utilisable pour signer les certificats, profondeur de vérification
- ↳ identifiant de la clé
- ↳ nom associé au certificat
- ↳ usage autorisé de la clé

```
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
        2b:9f:7e:e5:ca:25:a6:25:14:20:47:82:75:3a:9b:b9
  Signature Algorithm: sha1WithRSAEncryption
    Issuer: C=ZA, O=Thawte Consulting (Pty) Ltd.,
            CN=Thawte SGC CA
    Validity
        Not Before: Oct 26 00:00:00 2011 GMT
        Not After : Sep 30 23:59:59 2013 GMT
    Subject: C=US, ST=California, L=Mountain View,
             O=Google Inc, CN=mail.google.com
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
            Public-Key: (1024 bit)
            Modulus:
              00:af:39:15:98:68:e4:92:fe:4f:4f:f1:bb:ff:0d:
              2e:b0:fe:25:aa:bd:68:04:67:27:ea:6c:43:4c:a7:
              6d:cb:c8:8f:7e:81:ee:87:26:25:10:12:54:33:9e:
              aa:3d:9b:8f:8e:92:b3:4b:01:e3:f9:4a:29:c3:0f:
              fd:ac:b7:d3:4c:97:29:3f:69:55:cf:70:83:04:af:
              2e:04:6e:74:d6:0f:17:09:fe:9e:20:24:24:e3:c7:
              68:9c:ac:11:bd:92:e4:b2:1b:09:f2:02:32:bb:55:
              1b:2d:16:5f:30:12:23:e2:4c:4a:8d:c2:da:3f:e1:
              b8:bf:f7:3a:b1:86:be:f0:c5
            Exponent: 65537 (0x10001)
```

```
X509v3 extensions:
      X509v3 Basic Constraints: critical
        CA:FALSE
      X509v3 CRL Distribution Points:

        Full Name:
          URI:http://crl.thawte.com/ThawteSGCCA.crl

      X509v3 Extended Key Usage:
        TLS Web Server Authentication,
        TLS Web Client Authentication,
        Netscape Server Gated Crypto
      Authority Information Access:
        OCSP - URI:http://ocsp.thawte.com
        CA Issuers -
         URI:http://www.thawte.com/repository/Thawte_SGC_CA.crt

    Signature Algorithm: sha1WithRSAEncryption
       35:80:11:cd:52:3e:84:29:fb:c1:28:e1:20:e5:02:8f:5f:71:
       65:58:1d:62:72:57:3c:e6:5e:25:61:d3:cb:ad:22:f8:d8:81:
       a4:e7:f4:ae:7c:d9:c1:6d:aa:93:0d:62:07:9f:f2:67:47:99:
       34:33:4f:3d:02:74:f4:81:d6:38:08:21:e8:e2:a1:fa:05:41:
       9c:9c:c9:f9:f3:c8:a3:ee:0d:a5:d7:50:54:5e:2f:7d:79:b7:
       7e:0a:7c:b6:e2:2c:a8:ae:fe:94:d7:cd:16:30:71:04:aa:9e:
       79:c3:d2:b6:24:a7:25:ab:f0:48:8e:2f:c3:a7:bb:50:dd:0f:
       cf:b0
```

PKI
(Public Key Infrastructure)

- générer les certificats
- distribution
- stockage
- révoque
- ...

Comment mettre un certificat hors service ?

① Base de données de certificats à ne plus utiliser (CRL) — Certificate Revocation List
  ⊖ doit être régulièrement mis à jour
② Interrogation d'un serveur (OCSP) — Online Certificate Status Protocol
  ⊖ besoin d'une grande disponibilité du serveur OCSP

↳① on check l'identifiant du certificat, si il est dans le CRL, c'est
↳② On interroge le serveur OCSP pour check si le certificat est toujours valide.
  ↳ le serveur répond : oui/non.

Certificate Transparency List (CTL)

- bdd (en ajout seulement) de tous les certificats signés des CA.
  ↳ publique / accessible
- Même si quelqu'un arrive à voler la clé privée d'un CA (et qu'il forge un certificat valide)
  ↳ le certificat sera invalide car même s'il est signé il n'est pas dans la CTL