

(Secure SHell)

Maîtriser SSH...

pour se connecter à des serveurs (CLI),  
transférer des fichiers  
et accéder à des dépôts GIT

de manière **simple ET sécurisée !**

<https://www.math.u-bordeaux.fr/~lfacq/FormationSSH.pdf>

# de manière «Simple ET Sécurisée»

- Simple = sans avoir à taper un mot de passe à chaque connexion
- Sécurisé = sans laisser un mot de passe ou une clé en clair dans un fichier
- Grâce à :
  - des clés de chiffrement
  - « l'agent SSH » (ssh-agent) démon dont le rôle est de conserver les clés pendant toute la durée d'une session

# SSH – Contenu / Plan

- Utilisations classiques: ssh, git, scp, sftp, rsync, sshfs
- Principes de fonctionnement
  - Notion des (bi-)clés, utilisation de la clé, agent ssh
- En pratique :
  - Générer sa clef,
  - mise en place
  - l'agent ssh
- Notions de clé serveur :
  - principe & problèmes
- Fichier de configuration
- **TP / Prise en main**
- En option
  - Cas pratiques : connexion @IMB, @PlaFRIM
  - Utilisations avancées :
    - **GIT** depuis PlaFRIM,
    - Redirections de ports (tunnels TCP/IP)

# Utilisations Classiques de SSH

# ssh

connexion en mode terminal  
(ligne de commande) à un serveur

- **ssh serveur.domaine.fr**
- **ssh serveur**
- **ssh user@serveur**
  - dans le cas où votre nom d'utilisateur distant est différent, il faut le préciser (cf `.ssh/config`)

# git (via ssh)

GIT utilise SSH avec la syntaxe *user@server:chemin/du/projet* :

- `git clone git@plmlab.math.cnrs.fr:user/projet`
- `git fetch`
- `git pull`
- `git push`

# scp

## transfert de fichiers (secure copy)

- ***scp chemin-source chemin-destination***
- les « : » servent à spécifier les ressources distantes (noms des serveurs). 3 schémas :
  - « **scp chemin-source serveur:** » (copie vers)
  - « **scp serveur:... chemin-destination** » (copie depuis)
  - « **scp -3 serveur1:... serveur2:** » (de serveur à serveur - avec SSH récent)
- **scp -r ...** : copie récursive de répertoire(s)
- Exemples :
  - **scp f1.pdf f2.pdf acces:docs/** (recopie dans sous répertoire \$HOME/docs/)
  - scp fichier.pdf acces:monfichier.pdf** (recopier+renommage)
  - scp -r docs/ acces:/tmp** (copie récursive du répertoire **docs** dans /tmp distant)
  - scp acces:new.pdf .** (récupère \$HOME/new.pdf dans rep. Courant)

Note : **scp** écrase – sans demander confirmation - la destination si elle existe déjà

# sftp

## ftp sur ssh

(**secure file transfer protocol**)

- Comme ftp, mais sécurisé par SSH  
(en gras : opérations en plus de scp)
  - sftp [login@]serveur
  - **dir** ...
  - get, mget ...
  - put, mput ...
  - **del** ...
  - **mkdir** ...
  - **rmdir** ...



# **rsync** (via ssh)

synchronisation très efficace de fichiers & répertoires

*Copie de source vers destination – ajout/mise à jour, mais pas de re-copie inutiles*

**rsync source destination**

les « : » servent à spécifier les ressources (nom des serveurs)

**rsync -r mon\_repertoire serveur:mon\_rep**

**rsync -a serveur:travaux travaux\_portable**

-r : récursif

-a : « mode archive » (récursif + conserve le maximum de métadonnées)

peut-synchroniser en effacement également ! (--delete)

# sshfs

## « montage » d'un répertoire distant

- permet d'accéder à un répertoire distant comme s'il était présent localement
- Chaque accès à ce répertoire va provoquer une connexion SSH
- ex. d'usage : visualisation de données très volumineuses dont seule une infime fraction est utile. On évite de tout recopier.

```
mkdir mountpoint           #(création d'un nouveau répertoire vide « mountpoint »  
                             en local sur l'ordinateur courant, qui va servir à  
                             « accueillir » le répertoire distant)
```

```
sshfs [user@]host:[dir]    mountpoint
```

```
ls -al mountpoint
```

- Pour démonter : **fusermount -u mountpoint**

ou

```
killall sshfs
```

Principes de fonctionnements

Notions de (bi-)clefs

Rôle de l'agent SSH

# Principe de fonctionnement


- Rôle de SSH :
  - Obtenir une connexion réseau sécurisée entre deux machines
  - Sécurisée = **authentifiée** et **chiffrée** de bout en bout
- 2 moyens classiques d'authentification :
  - Par mot de passe
    - *non recommandé - hors 1ère fois - car risque de vol du mot de passe si compromission du serveur*
  - Par (bi)clés cryptographiques

# Notion de bi-clé cryptographique

partie privée

**Bi-clef** : deux clefs de chiffrement/déchiffrement



- **Partie privée (fichier *id\_rsa*\*)** 
  - Stockée chiffrée par un « mot de passe long » ou « ***passphrase*** » sur tous vos postes de travail
  - *à ne pas divulguer !  
elle sert à prouver votre identité*

\* : le nom change selon le type de clef « *id\_type* »

# Notion de bi-clef cryptographique

partie publique

**Bi-clef** : deux clefs de chiffrement/déchiffrement



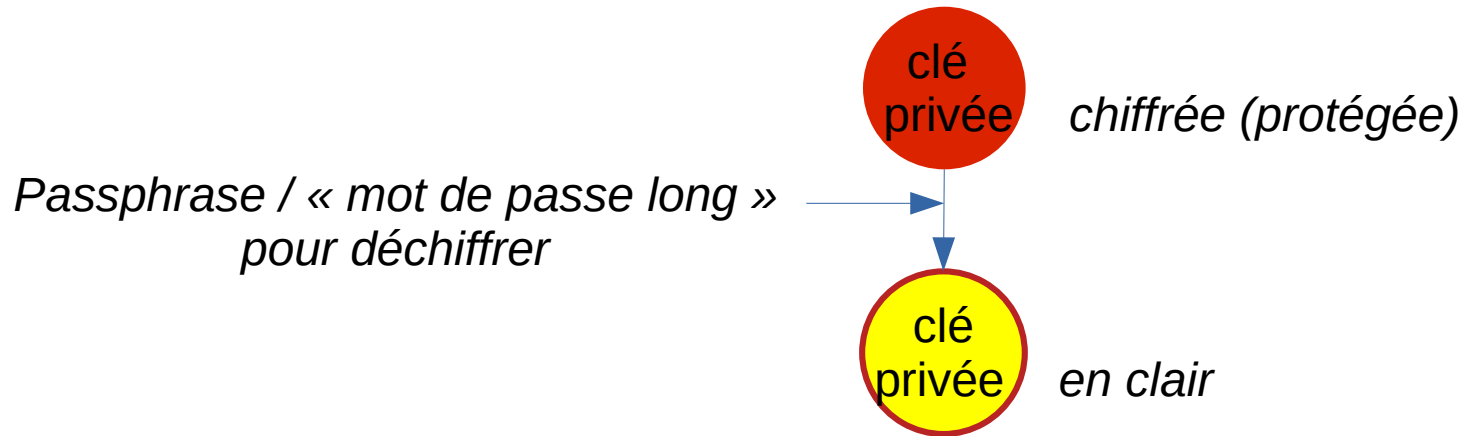
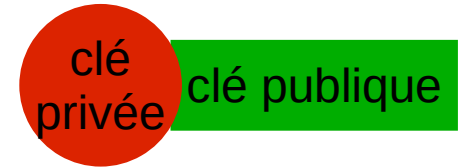
- **Partie publique (fichier *id\_rsa.pub*\*)** clé publique
  - Stockée en clair sur les serveurs distants (et sur le poste de travail à la génération) auprès desquels vous souhaitez être reconnu

\* : le nom change selon le type de clef « *id\_type.pub* »

# Notion de bi-clef cryptographique

## clé chiffrée vs en clair

**Bi-clef** : deux clefs de chiffrement/déchiffrement

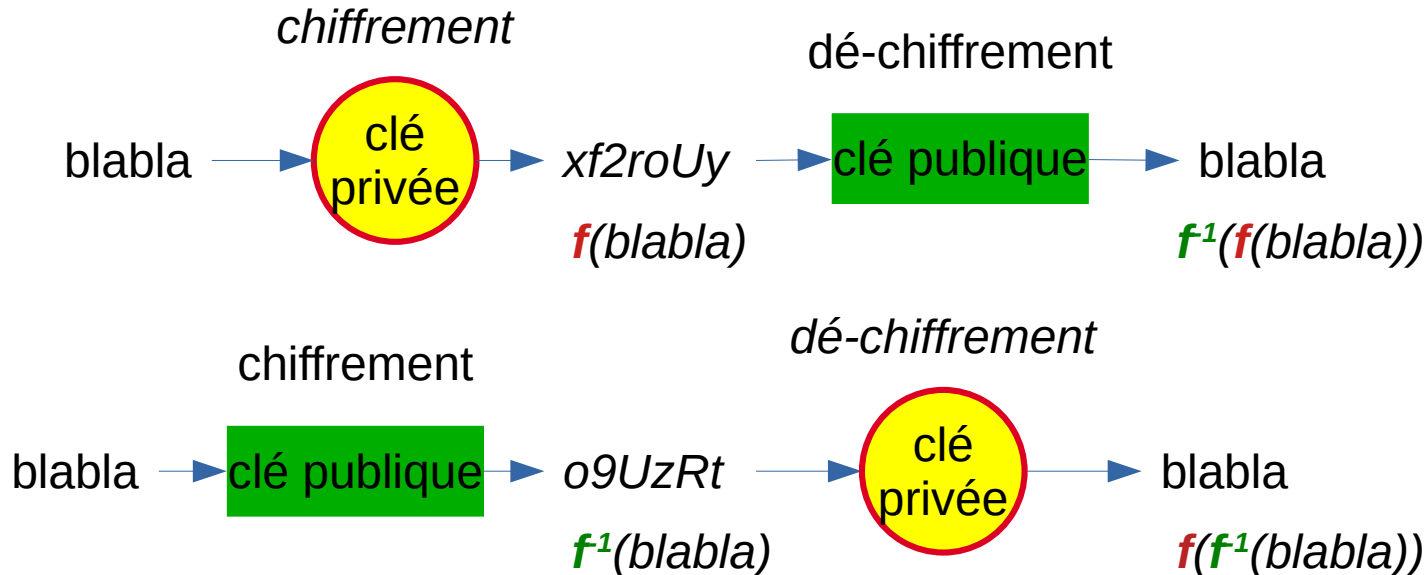
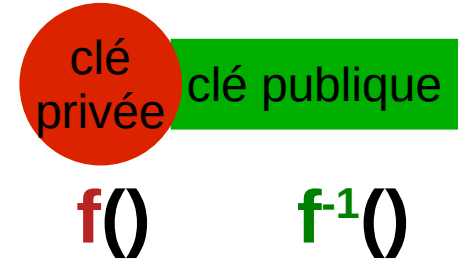


*On a besoin de la clé privée en clair pour l'utiliser*

# Notion de bi-clef cryptographique

## principe de fonctionnement

**Bi-clef** : deux clefs de chiffrement/déchiffrement  
(*chiffrement asymétrique*)



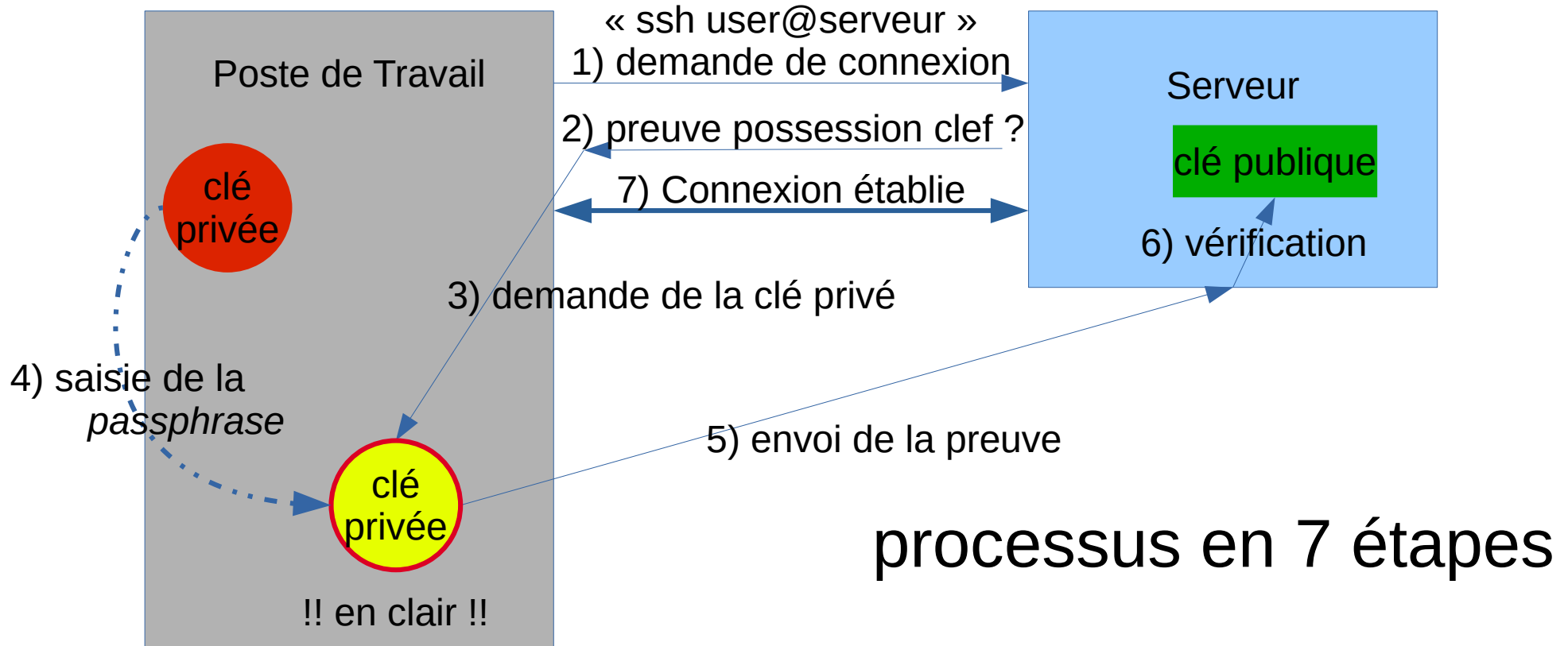


# Utilisation de la clef lors d'une connexion

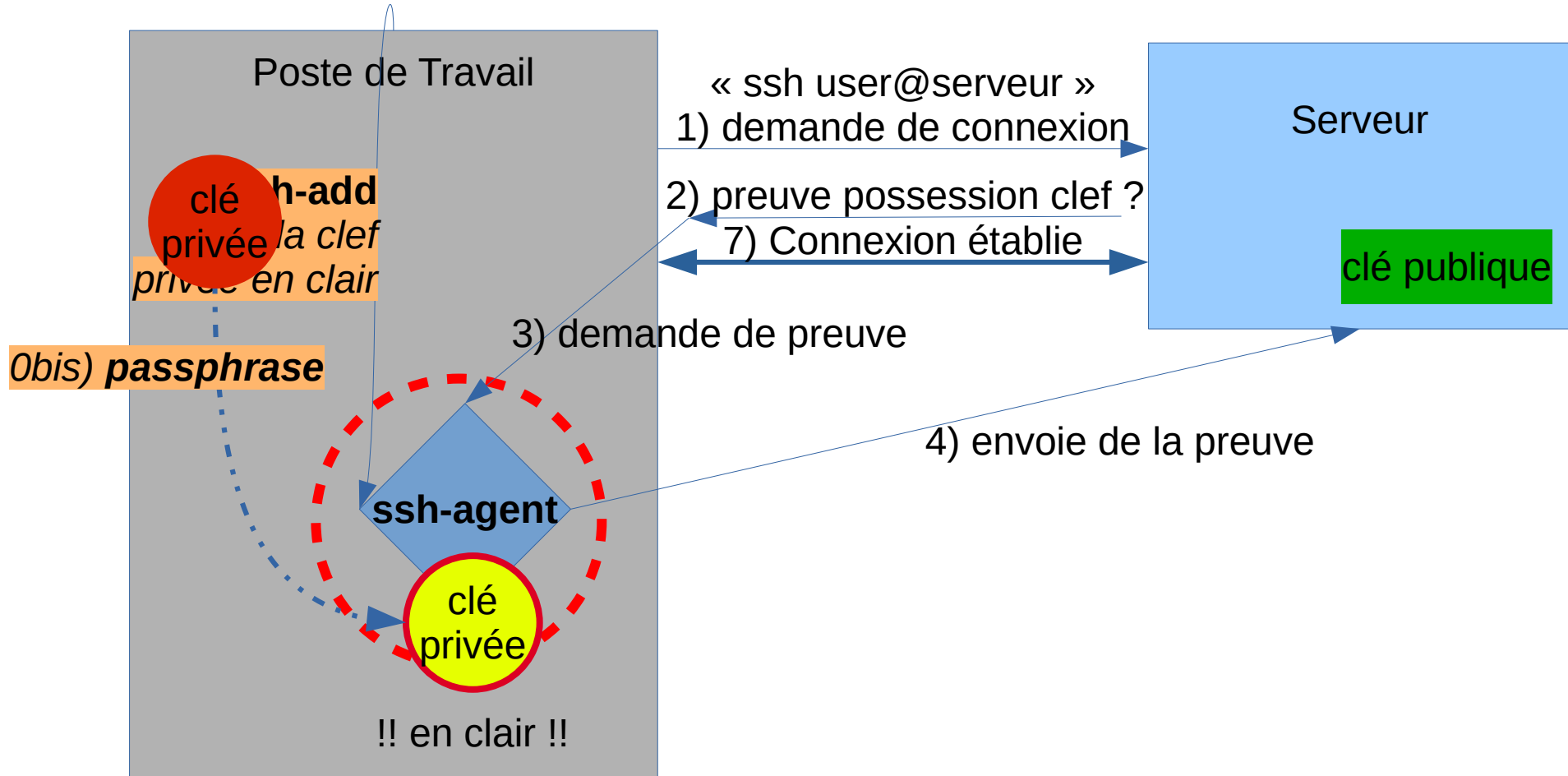


- Pré-positionnement des clefs
  - Privée sur le client / poste de travail
  - Publique sur le serveur

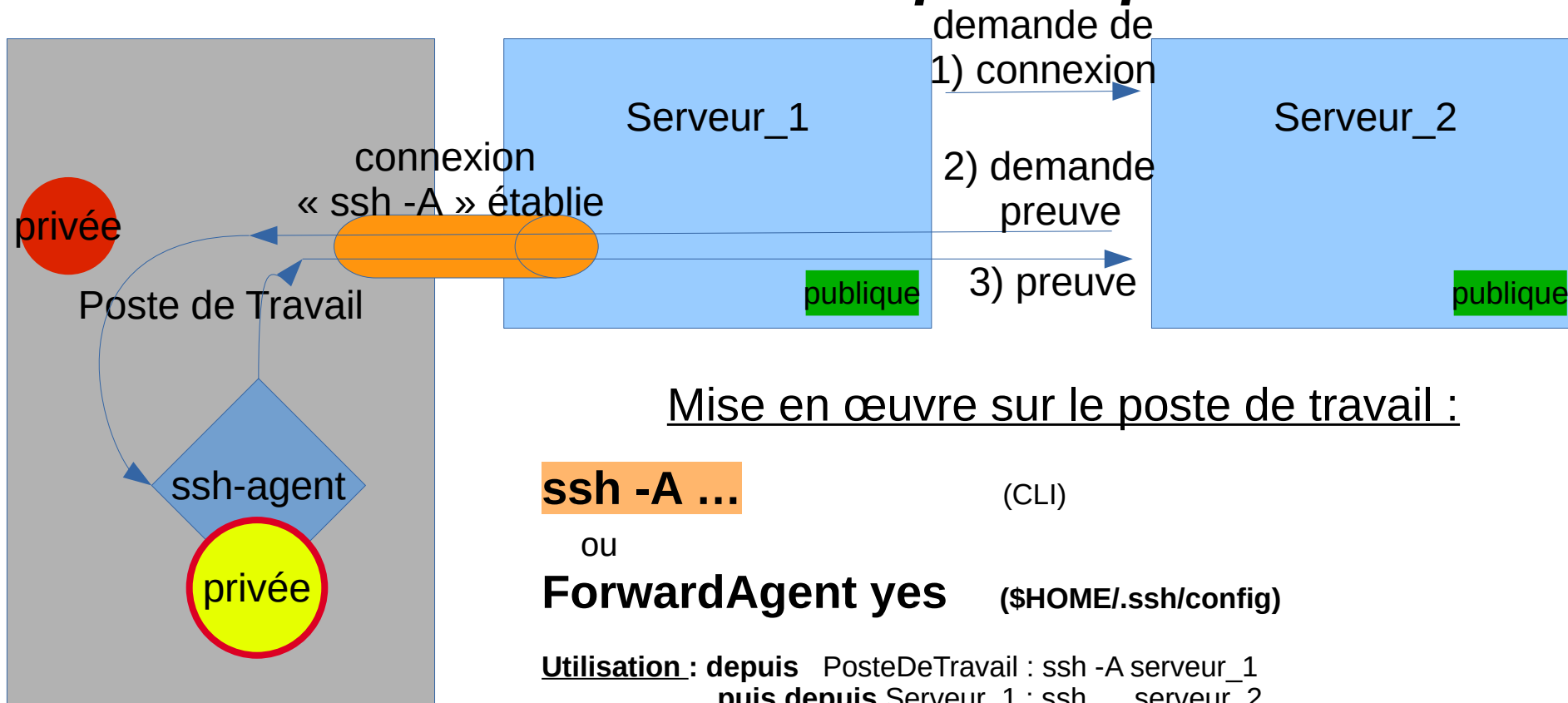
# Utilisation de la clef lors d'une connexion



# Saisir une seule fois sa *passphrase* par session grâce au **SSH Agent** (sur poste de travail)



# ... et rebondir de serveur en serveur *sans ressaisir sa passphrase*



# Les types de clé pour SSH

- **DSA, ECDSA** : non sûres ou déconseillées
- **Ed25519** :
  - Recommandé pour le futur...
  - ...mais possibles problèmes de compatibilité (récent)
- **RSA** : ***généré par défaut par ssh-keygen (en 3072 bits)***
  - OK avec au minimum 3072 bits
    - insuffisant en dessous / danger si moins de 1024 bits
  - Compatibilité maximale

Lister ses clés avec leur longueur :

```
for key in ~/.ssh/{id*,*.pub}; do ssh-keygen -l -f "${key}"; done | sort | uniq
```

En pratique,  
les commandes

# ssh-keygen

- **ssh-keygen** : pour générer votre bi-clé
  - Accepter le nom de fichier par défaut (`$HOME/.ssh/id_rsa`)
    - tapez « entrée / return »
  - Choisir un mot de passe (*passphrase*) qui peut être très long, servant à chiffrer la clé privée
    - **Mot de passe vide = trou de sécurité ! => pas la bonne solution**
  - clé par défaut dans :
    - privée\* : `$HOME/.ssh/id_rsa`
    - publique\* : `$HOME/.ssh/id_rsa.pub`

privée

publique

privée

publique

\* :le nom change selon le type de clé

# *Passphrase* perdue ?

- Si vous avez définitivement perdu votre *passphrase*... vous êtes bon pour « jeter » le bi-clé correspondant, relancer **ssh-keygen** et installer votre nouvelle clé partout où nécessaire
- Il est bon de nettoyer/effacer les anciennes clés « perdues » car vous risquez ne plus trop savoir où vous en êtes par la suite



# A quoi ressemble une clef publique

*id\_rsa.pub*

publique

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDF8eiOPv9+t71Y9J
JTT5kvK+hmB2K6l85TSOzRNVvcZo7VjyHF4xEddFLAsnCIVGe1Qo4HfwJv
eCpAsRbEcfGZ9qA2BJzdJrHca+o8bjLZo7ht9o3ybH6CqUMbcWleT12xDEn
Eljv+i58x6HF7nqYvELqVN80zrhx8An5O9cvxcUYEIW/
63BzwJBUvlGQCqKCAD8/5mVLkBT74wwq4w7yXZvk/NULKXmxge/
Y1QiN6/b0q1PIHQRJSVeeAC7jhCmw+QKM/
xwaZLeYdY5j2cHO2K9Dk7gEvVKgpsJDLU05YMEi57GuC+iVAMFMYCmA
P1jBJZqAP3QQglrLo/RybzD lfacq@monordi
```

- Format :
  - **TypeDeLaClef** **Clef** **Commentaire**
  - le tout sur une seule ligne

# A quoi ressemble une clef privée

## *id\_rsa (format PEM)*



-----BEGIN RSA **PRIVATE KEY**-----

Proc-Type: 4,ENCRYPTED

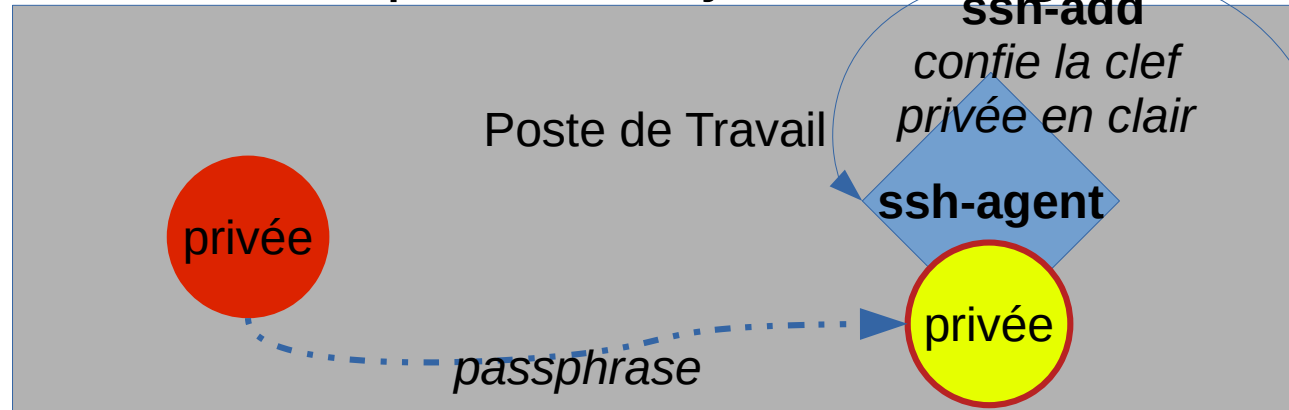
DEK-Info: AES-128-CBC,6230E5C2DC9D407ED22FDC7E450A765A

P4kulRYvzAse7Nq0u3NOV5LORPRoZ1n9qOy/y8oEfagvtaELsU6imgTJ91Hj07cj  
kHpEYKf5SpUbok8dShgQP64DGi+IBK0qlkytfJVQNXylZ6bjDiwiulUOkmDCJSs7  
fskKxCQpLqeLAisDX/9fT/UivIV84tR5y9APpfJfKb0MohDlaobDt4VjPDzkqcu1  
nS9DM1lzD6iVlpUWRK1TDQI31EZvR89NHffpy47WhU4ezeNqlawcQ3zWtJDKLKU0  
....cGYOAoAsEBMXYNCCBhq79+3TMpZoAaL1r1R2qdRwhYYibJjRU0uLDBmaNGxReH2n  
KPoi3gSgq/E2uA0mkIG6jt4GQbV9leUWOY5y1fqZM/HWTznHAJ2G5+Jb7WCeYO0U  
6RY2taK7LnEBLXBPqiLe9PNWiix9nLxjcoUneglvi+1SUEaK/fsYpTb37OUw9xT  
-----END RSA **PRIVATE KEY**-----

# ssh-add

privée

- **ssh-add** : pour ajouter sa clef dans l'agent SSH
  - prends la clef par défaut dans : `$HOME/.ssh/id_rsa`
  - pour préciser un autre fichier : **ssh-add fichier\_clef**
  - il faut taper sa *passphrase* pour déchiffrer la clé
- **ssh-add -L** : lister les clefs connues par l'agent
- **ssh-add -l** : idem pour lister juste les signatures des clés



# notions avancées ?

- Fonctionnalités un peu complexes...



- => juste retenir que cela existe !
- ... et venir nous voir quand vous avez besoin

# ssh-agent

- programme qui garde les clefs privées déchiffrées
- généralement déjà lancé automatiquement par votre environnement graphique pour la durée de votre session
- Si cela ne se fait pas tout seul, il peut-être nécessaire de le lancer à la main :
  - **ssh-agent /bin/bash** : démarrage d'un nouveau shell (interpréteur de commande) « relié » à un nouveau **ssh-agent**
  - **eval \$(ssh-agent)** : démarrage d'un nouveau **ssh-agent** dans le shell courant (ex: sous *windows/git bash*)

SSH  
Avancé



# ssh-agent

## variables d'environnement

- `$ env | grep SSH`  
`SSH_AGENT_PID=5477`  
`SSH_AUTH_SOCK=/tmp/ssh-mLVmITqHg521/agent.5475`
- `$ ssh-agent`  
`SSH_AUTH_SOCK=/tmp/ssh-2gxay2uHa746/agent.1054; export SSH_AUTH_SOCK;`  
`SSH_AGENT_PID=1055; export SSH_AGENT_PID;`  
`echo Agent pid 1055;`
- `$ eval $(ssh-agent)` : exécute dans le shell courant  
les commandes renvoyées par `ssh-agent` (cf au dessus)

# Recopier sa clé publique

sur les serveur distants

## ssh-copy-id

publique

- **ssh-copy-id** **serveur**  
**ssh-copy-id user@serveur**  
pour recopier sa clé publique sur un serveur afin d'autoriser les connexions par clé (1ère fois => par mot de passe)
  - clé source : par défaut dans : **\$HOME/.ssh/id\_rsa.pub**
    - pour préciser un autre fichier :  
**ssh-copy-id -i FichierClefPub serveur**
  - fichier destination sur **serveur** : **\$HOME/.ssh/authorized\_keys**  
qui liste toutes les clé autorisées à se connecter par SSH au nom de l'utilisateur du compte
- à faire aussi sur serveurs GIT type Gitlab/Hub, mais via interface web

publique

# Exemple de \$HOME/.ssh/authorized\_keys

**# clés publiques autorisant la connexion à ce compte**

# 3 clefs => 3 lignes

# format : [options] Type\_de\_clef Clef Commentaire

```
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA6 publique JWCjmkDToGnTlztG2nf8gCALjbqq
XXAvGWVQE+3Fifls2ey207f6z1v9kFICbUCFwVACQy4nsmEug5Q== facq@scratchy
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDRDDqqOhtzo publique c1jlhpfR5VswE
o8J3YRohX1Kpd6B4H60pUrUu1Hv1Ts08qoCkTbgvM5G6JX8/rstiC 4V9/sNJDKPzP
lfacq@o780-33.math.u-bordeaux1.fr
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDR publique EQPWtGeMpPyUY1+hEeR3D
YPpBuMSYCTKBi8hMa9fUys8t4T7iamMqGDNGOfPWIMRNGeo30Av6vEV3brFazq6Sxdkbsa37pLw
mSttj9uLbF+wbUbniN8Tvm6x8n7ELMWRSnvGPMLtOLW+nlf8sUTh/HjviZDbsC+TvAf lfacq@toto
```

**# autres possibilités (options) :**

# \* forcer la commande à exécuter pour une certaine clef : command= « commande »

# \* imposer des adresses IP source des clients : from= « pattern »

# ...





# Recopier sa clé privée sur vos portables & postes de travail

privée

- À recopier « à la main »
  - Par clé USB,
  - Avec **scp** (en saisissant son mot de passe...)
  - Par copier coller du contenu du fichier
    - (copier - lire la clef) `cat $HOME/.ssh/id_rsa`
    - (coller - écrire la clef) `cat > $HOME/.ssh/id_rsa` (!! écrase le fichier!!)  
(terminer avec Control-d)
- besoin de rectifier les droits du fichier :
  - `chmod go= $HOME/.ssh/id_rsa`  
(les droits « **g**roup » et « **o**ther » sont positionnés à « rien »)

*SSH refuse que la clé privée soit lisible par d'autres utilisateurs*

# Un ou plusieurs bi-clé ?

- Généralement un seul bi-clé suffit
  - Recopier la partie privée (à minima) sur tous vos postes de travail (connexion physique) :
    - Ordinateur portable
    - Ordinateur de bureau
- Plusieurs clé, pourquoi faire ?
  - Niveaux de sécurité différents
  - Segmenter les risques
  - Période de migration de clefs (si besoin de générer une nouvelle clé plus sécurisée)



*divulguer la partie publique d'une clé facilite son « cassage »*

# Notion de clé serveur, et problèmes associés

# Clés de Serveurs / Principe

- Les clés « utilisateurs » servent à prouver au serveur l'identité des utilisateurs
- Réciproquement, chaque serveur a aussi une clé individuelle pour prouver son identité aux utilisateurs
- **=> Première connexion : il faut accepter la clé du serveur**
- Les clés serveurs sont collectées dans votre fichier :  
`$HOME/.ssh/known_hosts`

# Clés de Serveurs

## ex : première connexion

```
facq@x7400:~$ ssh acceskey.math.u-bordeaux.fr
```

```
The authenticity of host 'acceskey.math.u-bordeaux.fr  
(147.210.16.1)' can't be established.
```

```
RSA key fingerprint is
```

```
SHA256:sQMmHigzyisfjRT+7iw8lvpQUUvp+k3kE8ojDFL2oJE.
```

```
Are you sure you want to continue connecting (yes/no)?
```

Cela se produit aussi avec un serveur git inconnu !

faire dans ce cas : **ssh nom.du.serveur.git** et accepter l'ajout de ce serveur

# Clés de Serveurs

## Problèmes

- Problème quand la clé du serveur change : refus de connexion
  - Pourquoi ? Pour éviter les piratages
    - si connexion par mot de passe => vol de mot de passe **risque**
  - Situation « Normale » si le serveur a été réinstallé avec génération d'une nouvelle clé
- Si besoin & situation maîtrisée : supprimer l'ancienne clé
  - Supprimer la ligne (n° indiqué) concernant le serveur
  - Utiliser la commande toute prête suggérée par la commande **ssh** :  
**\$ ssh-keygen -R serveur**

# Clés de Serveurs / Problèmes

## exemple de refus : mauvaise clé

```
[facq@login02 ~]$ ssh login01
```

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
@      WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!      @
```

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
```

```
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
```

```
It is also possible that a host key has just been changed.
```

```
The fingerprint for the RSA key sent by the remote host is
```

```
SHA256:l0WhYi9nRpgP0vUazZF8UPDq2A3W85NzwVg/x8wsIAY.
```

```
Please contact your system administrator.
```

```
Add correct host key in /gpfs/home/facq/.ssh/known_hosts to get rid of this message.
```

```
Offending RSA key in /gpfs/home/facq/.ssh/known_hosts:10
```

```
Password authentication is disabled to avoid man-in-the-middle attacks.
```

```
Keyboard-interactive authentication is disabled to avoid man-in-the-middle attacks.
```

```
Permission denied (publickey,password,keyboard-interactive).
```

# Clefs de Serveurs / Problèmes

## exemple de « Warning » - 2 clefs (bad/good)

```
facq@x7400:~$ ssh acceskey.math.u-bordeaux.fr
```

```
Warning: the RSA host key for 'acceskey.math.u-bordeaux.fr'  
differs from the key for the IP address '147.210.16.1'
```

```
Offending key for IP in /home/facq/.ssh/known_hosts:19
```

```
Matching host key in /home/facq/.ssh/known_hosts:217
```

```
Are you sure you want to continue connecting (yes/no)?
```



# Clés de Serveurs / Refus

Solution : supprimer la clé

```
$ ssh-keygen -R serveur
```

```
ssh-keygen -R login01
```

```
# Host login01 found: line 10
```

```
/home/facq/.ssh/known_hosts updated.
```

```
Original contents retained as
```

```
/home/facq/.ssh/known_hosts.old
```

Fichier de configuration  
**\$HOME/.ssh/config**

# Fichier de config : `$HOME/.ssh/config`

- contient :
  - Des options globales
  - Des paragraphes : un paragraphe par serveur distant
    - Configuration fine pour chaque serveur
    - Ex : User      (*identité « login » distant*)
  - Des options par défaut type **Host \***      (**Host *pattern***)
- les paragraphes commencent par une ligne « **Host ...** »
- les indentations ne sont là que pour faciliter la lecture

# Exemple de fichier \$HOME/.ssh/config

**Host acces**

**HostName** *acceskey.math.u-bordeaux.fr*

**User** *facq*

# utilisation : **ssh acces**

# au lieu de : **ssh facq@acceskey.math.u-bordeaux.fr**

# gain : abréviation du nom du serveur & précision du User

# **HostName** : précise, si besoin, le nom réel « complet » du serveur

**Host accesbis**

**HostName** *acceskey.math.univ-bordeaux.fr*

**User** *superfacq*

# si besoin de se connecter avec un autre nom d'utilisateur

# usage : **ssh accesbis**

# au lieu de : **ssh superfacq@acces**

# Exemple de paragraphe

## \$HOME/.ssh/config plus complet

**Host acces**

**HostName acceskey.math.u-bordeaux.fr**

**User facq**

*# propage l'accès à l'agent (-A)*

**ForwardAgent yes**

*# garde les clé automatiquement à la première saisie*

**AddKeysToAgent yes**

*# propage l'accès au display X11 (-X)*

**ForwardX11 yes**

# Sécurité : Propager l'accès à l'Agent SSH et à votre terminal X11 ?

- Ne propager l'accès à l'agent et/ou au terminal X11 **que** vers des serveurs « fiables »
  - Sinon, donne accès au super utilisateur de ce serveur à
    - l'utilisation de votre clef
    - votre écran graphique et votre clavier
- Si besoin : (recommandé / plus sûr)
  - « -X » : déport graphique en mode limité: accès partiel à votre écran+clavier
    - ForwardX11 yes
    - **ForwardX11Trusted no**
- Uniquement si nécessaire :
  - « -Y » : déport graphique en mode total (accès à tout)
    - ForwardX11 yes
    - **ForwardX11Trusted yes**

# Exemple de fichier \$HOME/.ssh/config

```
# Ignore des directives si inconnues de votre version de ssh
IgnoreUnknown    AddKeysToAgent, ProxyJump, UseKeyChain

# abréviation « ssh acces »
Host acces
    HostName acceskey.math.u-bordeaux.fr
    User facq
    ForwardAgent    yes
    IdentityFile    ~/.ssh/myspecialkey    #(option) utiliser cette clef spécifique
    IdentitiesOnly    yes                #(option) n'envoyer que la/les clefs précisées

# accès directe à deux machines internes
# par rebond sur une machine intermédiaire (acces)
# (SSH version ≥ 7.3)
Host bureau, servisu
    User facq
    ProxyJump acces

# entrées par défaut (attributs ajoutés à toutes les entrées qui 'matchent')
Host *
    AddKeysToAgent yes
    # sur macOS, stocke la clé dans le trousseau de clés (keychain)
    UseKeychain    yes
```

# Debug de la config

- L'option **-v** permet d'afficher des informations de debug et en particulier l'utilisation des directives du fichier de configuration :

```
$ ssh -v monserveur |& grep config
```

```
debug1: Reading configuration data /facq/.ssh/config
```

```
debug1: /facq/.ssh/config line 17: Applying options for monserveur
```

```
debug1: /facq/.ssh/config line 230: Applying options for *
```

```
debug1: /facq/.ssh/config line 232: Ignored unknown option "usekeychain"
```

```
debug1: Reading configuration data /etc/ssh/ssh_config
```

```
debug1: /etc/ssh/ssh_config line 19: include /etc/ssh/ssh_config.d/*.conf matched no files
```

```
debug1: /etc/ssh/ssh_config line 21: Applying options for *
```



# Astuce

- Pour quitter une session SSH plantée / qui ne répond plus :
  - taper « *[entrée]*~. »

séquence de 3 touches: [*return/retour*] [tilde] [point]

# Se connecter à l'IMB depuis l'extérieur

- `ssh acceskey.math.u-bordeaux.fr`

- /!\ nécessite d'ouvrir l'accès au préalable

- <https://www.math.u-bordeaux.fr/intranet/imb/acces>

note :

`ssh acces.math.u-bordeaux.fr` : accès par mot de passe

`ssh acceskey.math.u-bordeaux.fr` : accès par clé

- Alternative graphique (avec accélération) :

- X2GO

- <https://www.math.u-bordeaux.fr/imb/cellule/connexion-a-l-imb-depuis-l-internet>

Sur certains serveurs : accès par clé **mais** avec la nécessité, toutes les 24h,  
de rentrer votre mot de passe pour une nouvelle connexion

# TP : générer sa clé et préparer son environnement SSH

- Générez votre bi-clé : **ssh-keygen** *(sauf si vous en avez déjà une !)*
- Chargez la dans votre agent (à chaque nouvelle session) : **ssh-add**
- Copier votre clef public ... : **\$HOME/.ssh/id\_rsa.pub**  publique
  - Prerequis pour se connecter à l'IMB (si nécessaire) - Ouvrir l'accès ici :  
<https://www.math.u-bordeaux.fr/intranet/imb/acces>
  - ... sur des serveurs (IMB, curta) : **ssh-copy-id acces.math.u-bordeaux.fr**
  - ... sur des serveurs GIT
    - Ex : plmlab.math.cnrs.fr / gitlab.inria.fr ( / github... )
      - Menu «Vous»(en haut à droite) => Settings
      - SSH Keys (barre gauche)
- Copie votre clef privée : **\$HOME/.ssh/id\_rsa**  privée
  - sur portable(s), sur poste de travail IMB
- Créer son fichier de config (**\$HOME/.ssh/config**) pour simplifier les connexions

Questions ?

Cas d'utilisation particuliers



# Github/Gitlab depuis PlaFRIM

- Problème :
  - Filtrage réseau en sortie de PlaFRIM
  - Sites avec beaucoup d'IP différentes et qui changent souvent
- Solution :
  - Accès sur serveur fixe définit dans le fichier .ssh/config

\$HOME/.ssh/config :

```
##### git clone gitlab.com/....
```

```
Host gitlab.com
```

```
HostName altssh.gitlab.com
```

```
User git
```

```
Port 443
```

```
##### git clone github.com/....
```

```
Host github.com
```

```
HostName ssh.github.com
```

```
User git
```

```
Port 443
```

# Connexion à PlaFRIM

**\$HOME/.ssh/config :**

...

Host plaf

    User facq

    ForwardAgent    yes

    ForwardX11      yes

    AddKeysToAgent  yes

    ProxyCommand ssh -l facq ssh.plafrim.fr -W plafrim:22

...



# Utilisations Avancées

## redirection de ports / tunnels





# Redirection de ports réseau via SSH (tunnels tcp)

- But :
  - Traverser les pare-feu (filtrage réseau)
  - Sécuriser les échanges (chiffrement) pour des applications non protégées
- Exemples :
  - Déport graphique
    - X11 (basique) : inclus dans SSH (-Y)
    - (Turbo)VNC (accéléré/optimisé)
    - Exemple : **matlab**
  - Déport traitement
    - Exemple : Jupyter Notebook
      - Kernel qui tourne sur serveur de calcul
  - Accès distant à des ports réseau locaux (*localhost*, *IP=127.0.0.1*)
  - Proxy HTTP (web) sur machine très fermée (ex : pour installer des packages)

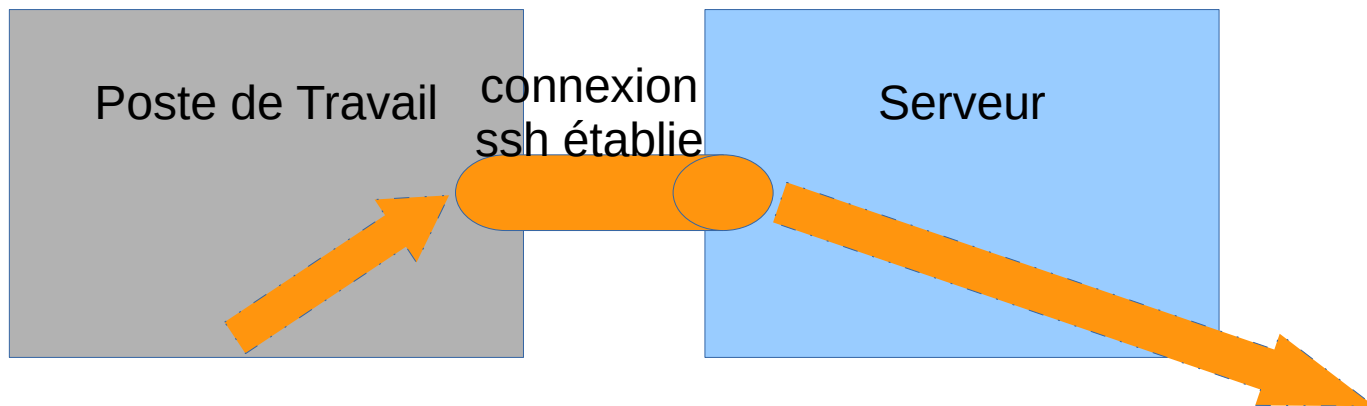
# Notion de Port réseau / IP

- Sur internet, le protocole le plus utilisé s'appelle TCP. Il permet à chaque ordinateur de disposer de 65536 ports réseau (1 à 65535) permettant de créer des canaux de communication vers d'autres ordinateurs
- Ex de ports célèbres :
  - Port 80 = HTTP
  - Port 443 = HTTPS
  - Port 22 = SSH
  - X11 = 6000...
- Une connexion (TCP) est identifiée par le quadruplet :
  - { ( IP source, Port source ) → ( IP destination, Port Destination ) }



# Principe des redirections / tunnels

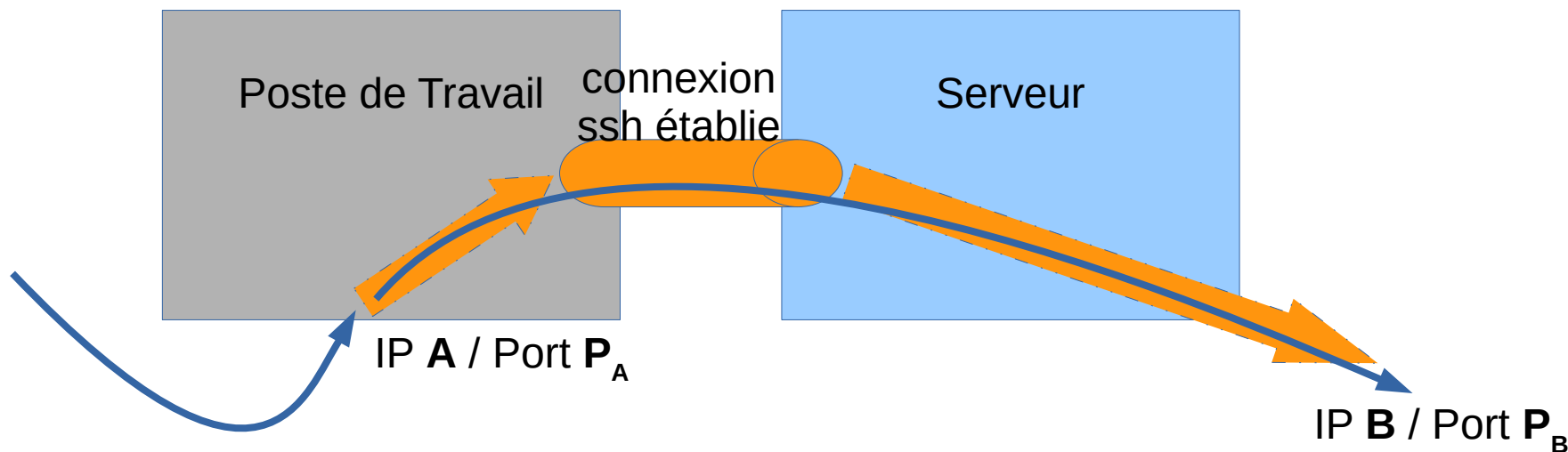
- Les connexions arrivant d'un côté de la connexion SSH vont déclencher une nouvelle connexion depuis l'autre côté du SSH, vers la destination définie





# Principe des redirections / tunnels

- les connexions arrivant sur le port  $P_A$  (de l'IP  $A$ )
- seront redirigées sur le port  $P_B$  (de l'IP  $B$ )
- les paquets circulent ensuite dans les 2 sens

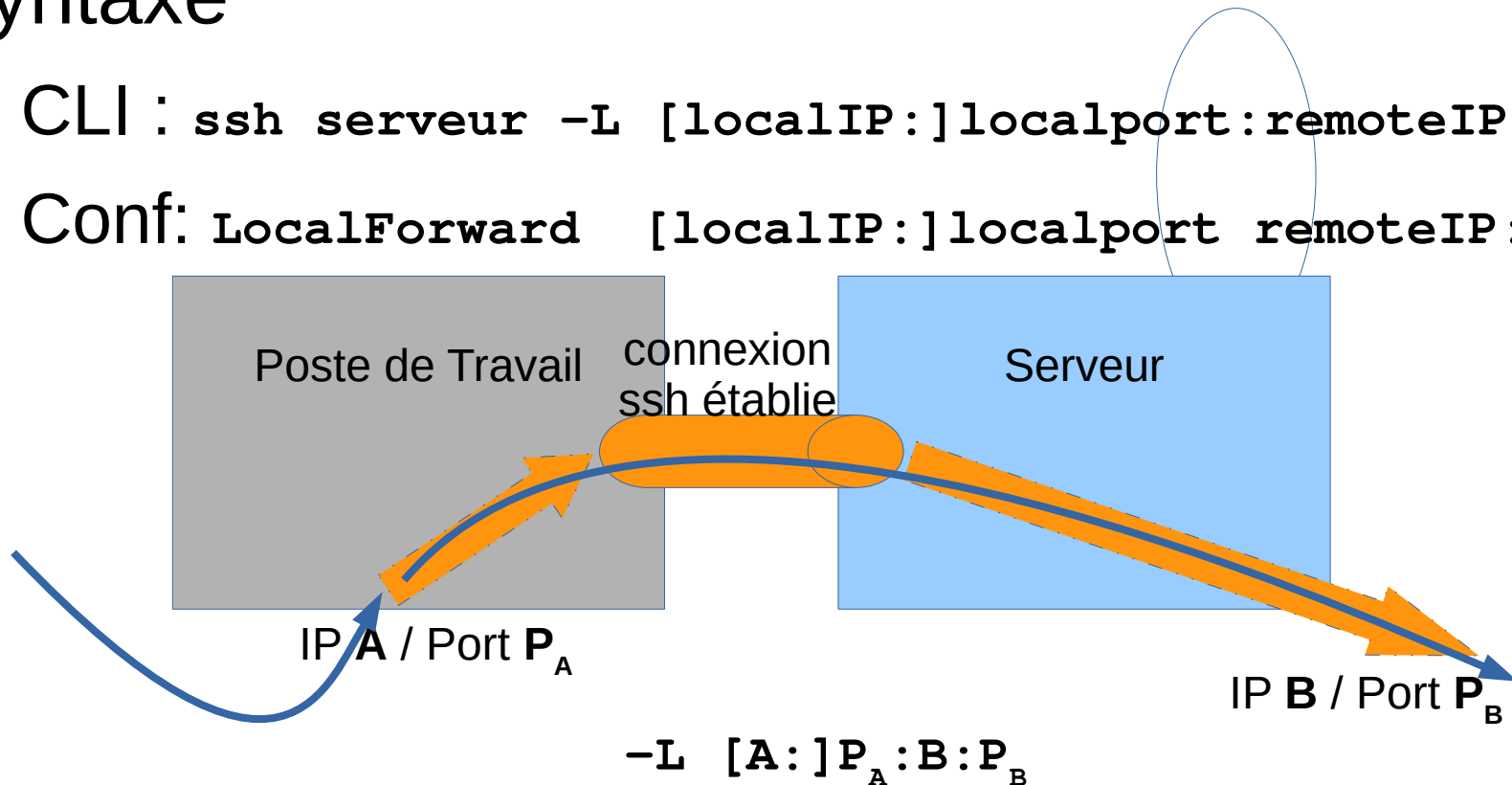




# Principe des redirections / tunnels

- Syntaxe

- CLI : `ssh serveur -L [localIP:]localport:remoteIP:remotePort`
- Conf: `LocalForward [localIP:]localport remoteIP:remotePort`

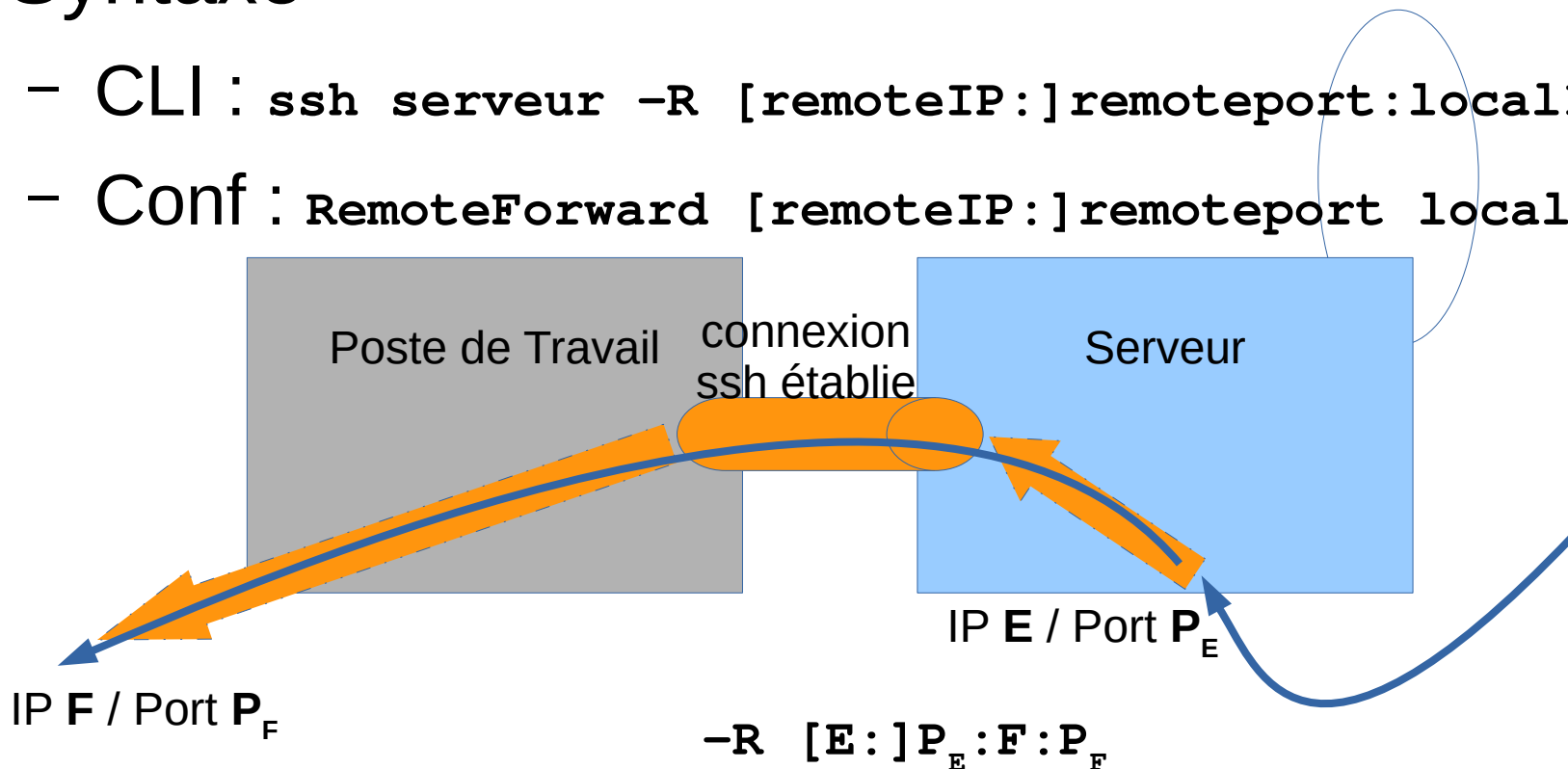




# Redirections / Tunnels : cela fonctionne aussi dans l'autre sens

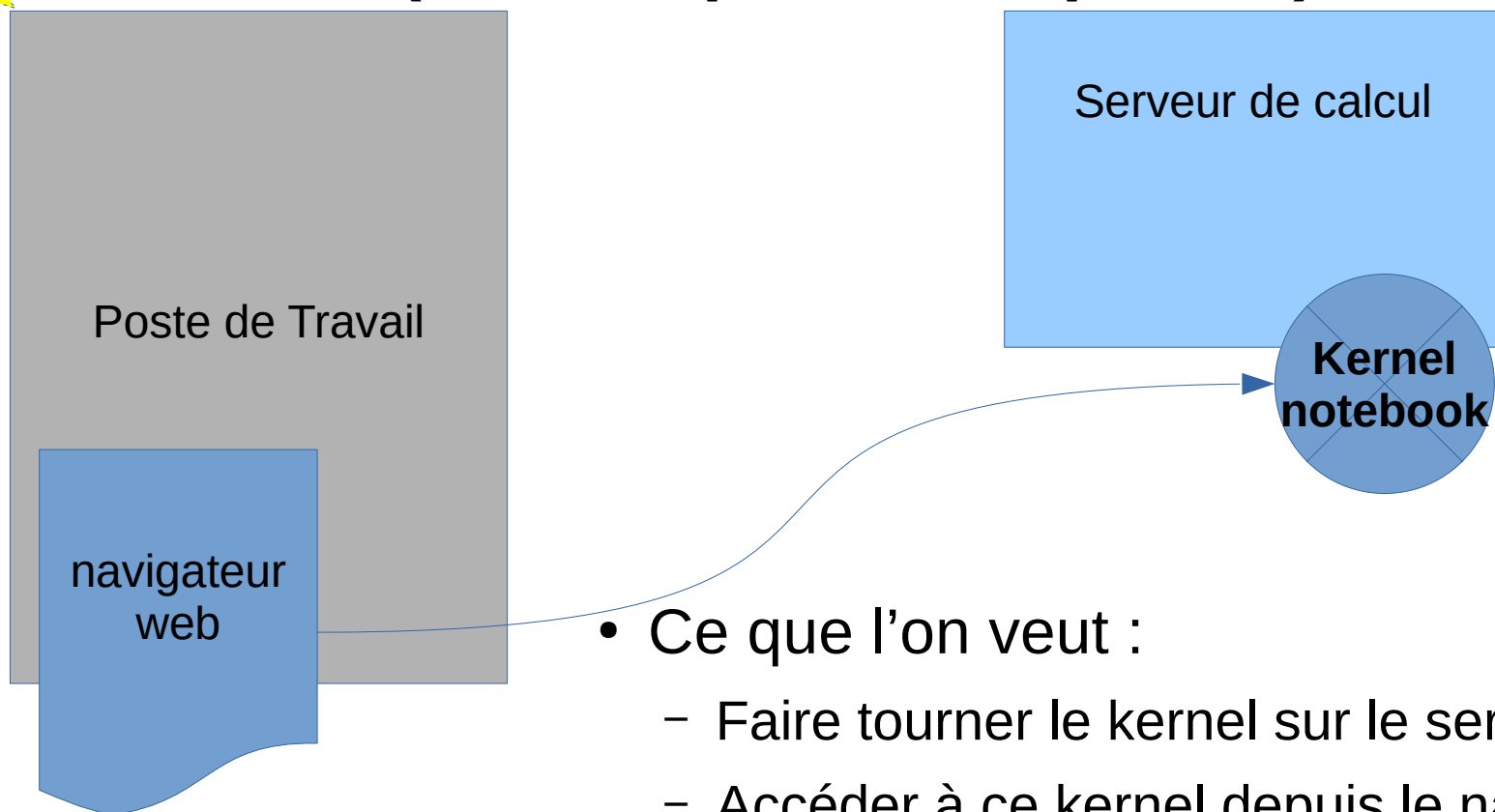
- Syntaxe

- CLI : `ssh serveur -R [remoteIP:]remoteport:localIP:localPort`
- Conf : `RemoteForward [remoteIP:]remoteport localIP:localPort`



SSH  
Avancé

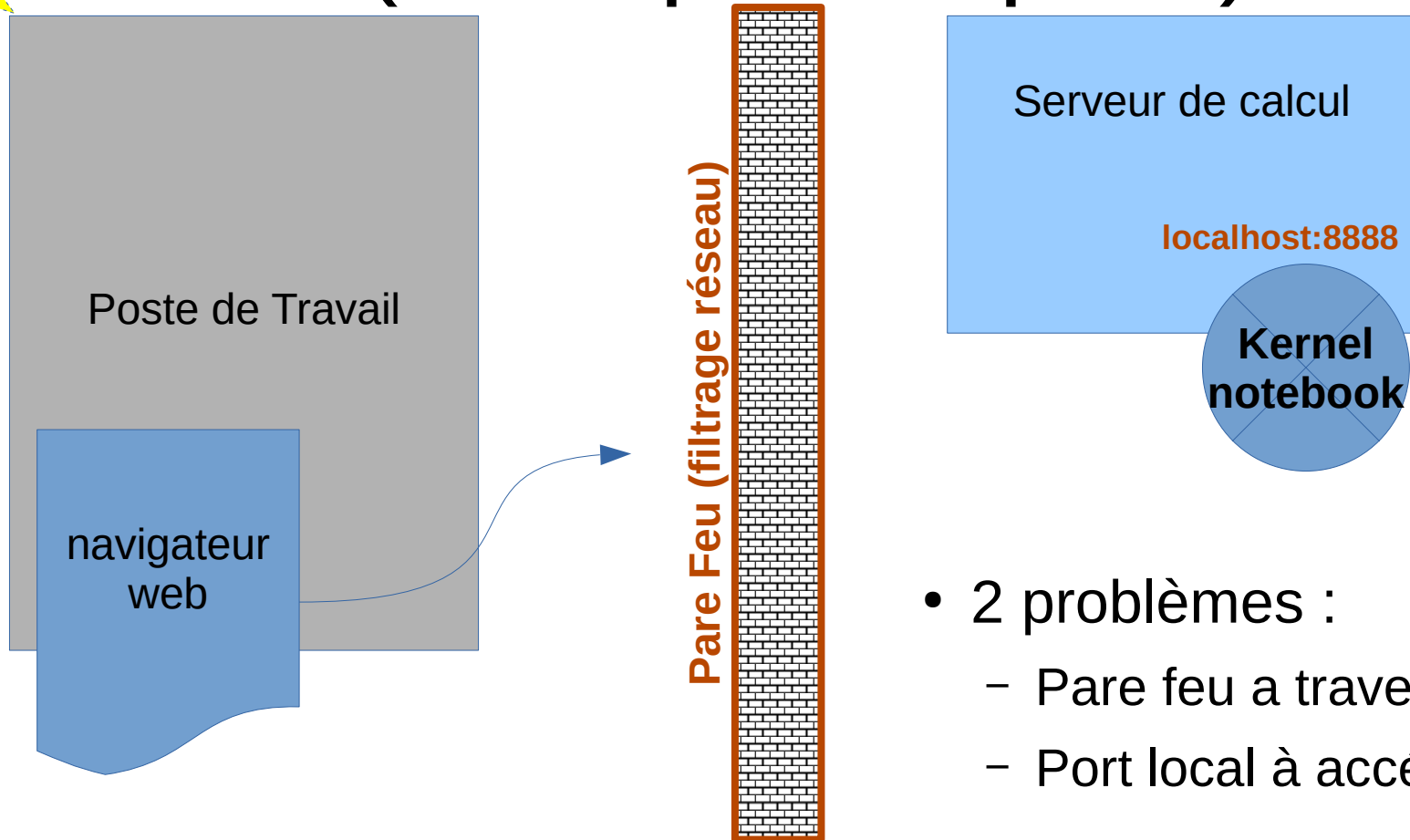
# Tunnel sortant : jupyter notebook (exemple simplifié)



- Ce que l'on veut :
  - Faire tourner le kernel sur le serveur de calcul
  - Accéder à ce kernel depuis le navigateur

SSH  
Avancé

# Tunnel sortant : jupyter notebook (exemple simplifié)

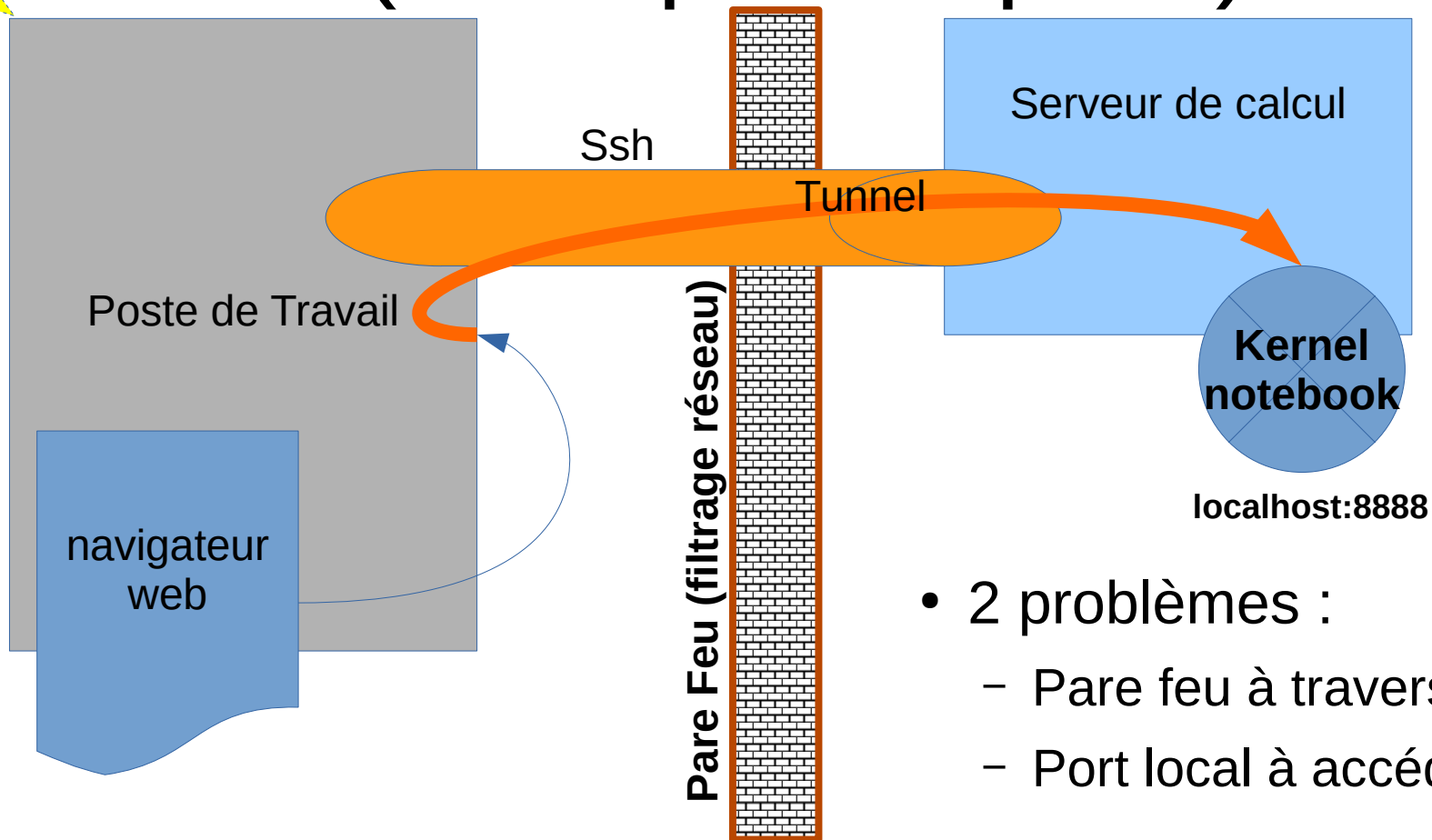


- 2 problèmes :
  - Pare feu a traverser
  - Port local à accéder



SSH  
Avancé

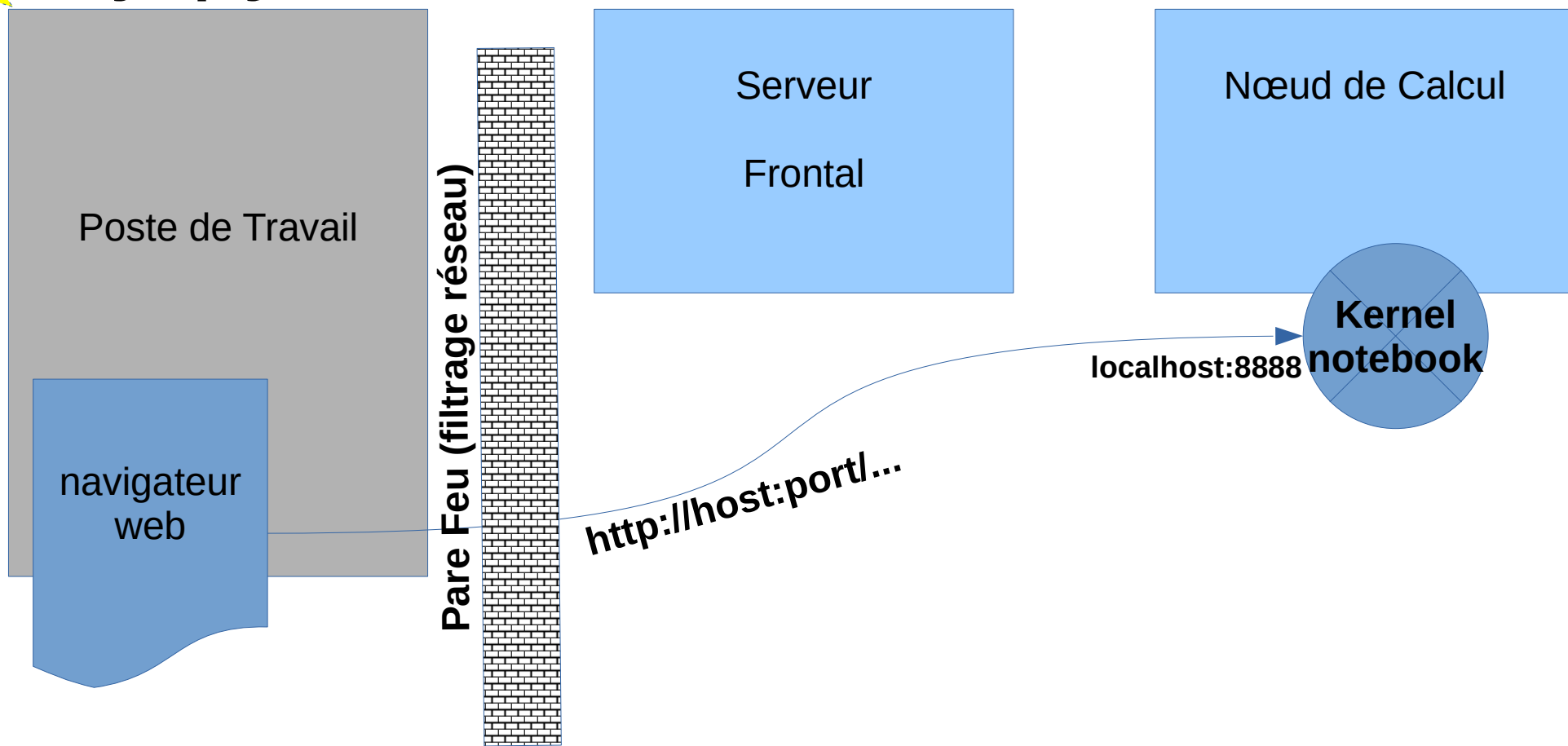
# Tunnel sortant : jupyter notebook (exemple simplifié)



- 2 problèmes :
  - Pare feu à traverser
  - Port local à accéder

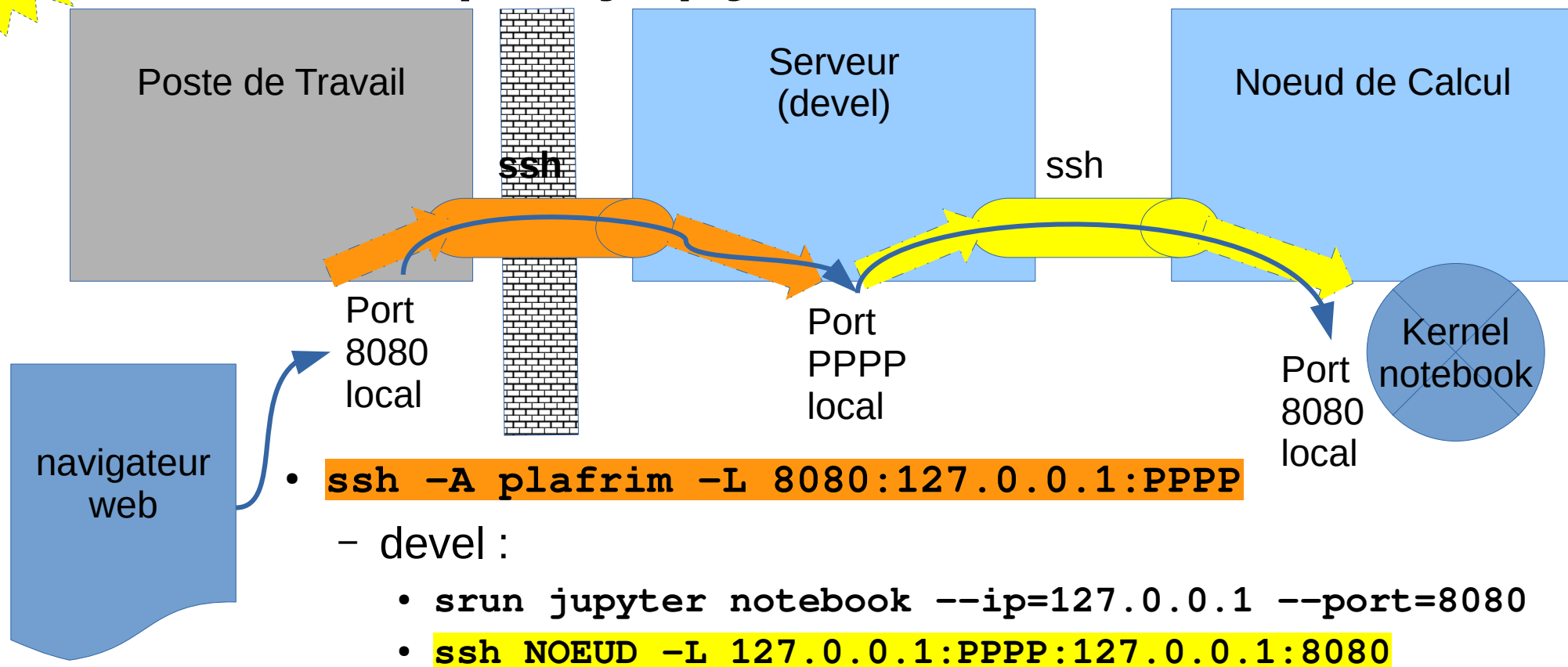


# Tunnel sortant : exemple jupyter notebook / PlaFRIM3





# Tunnel sortant : exemple jupyter notebook



# Fin

Merci pour votre attention !

et surtout merci pour votre feedback à venir :

- Des éléments essentiels manquants ?
- Des éléments pas clairs ou mal expliqués ?
- Des éléments trompeurs qui vous ont **induit en erreur** ?

=> merci de m'en faire part afin d'améliorer cette présentation :-)