Version : **2024.01**

Dernière mise-à-jour : 2024/10/10 10:39

RH12413 - Gestion du Réseau

Contenu du Module

- **RH12413 - Gestion du Réseau**
  - Contenu du Module
  - Comprendre les Réseaux
    - Présentation des Réseaux
      - Classification des Réseaux
        - Classification par Mode de Transmission
        - Classification par Topologie
        - Classification par Étendue
        - Les Types de LAN
      - Le Modèle Client/Serveur
      - Modèles de Communication
        - Le modèle OSI
        - Spécification NDIS et le Modèle ODI
        - Le modèle TCP/IP
      - Les Raccordements
        - Les Modes de Transmission
        - Les Câbles
        - Les Réseaux sans Fils
        - Le Courant Porteur en Ligne
      - Technologies
        - Ethernet
        - Token-Ring
      - Périphériques Réseaux Spéciaux

        - Les Concentrateurs
        - Les Répéteurs
        - Les Ponts
        - Les Commutateurs
        - Les Routeurs
        - Les Passerelles
    - Comprendre le Chiffrement
        - Introduction à la cryptologie
            - Définitions
                - La Cryptographie
                - Le Chiffrement par Substitution
        - Algorithmes à clé secrète
            - Le Chiffrement Symétrique
        - Algorithmes à clef publique
            - Le Chiffrement Asymétrique
            - La Clef de Session
        - Fonctions de Hachage
        - Signature Numérique
        - Utilisation de GnuPG
            - Présentation
            - Installation
            - Configuration
            - Signer un message
            - Chiffrer un message
        - PKI
            - Certificats X509
- Comprendre IPv4
    - En-tête TCP
    - En-tête UDP
    - Fragmentation et Ré-encapsulation
    - Adressage
    - Masques de sous-réseaux
    - VLSM
    - Ports et sockets

        - /etc/services
        - Résolution d'adresses Ethernet
    - Comprendre IPv6
        - Présentation
        - Adresses IPv6
        - Masque de Sous-réseau
        - Adresses IPv6 Réservées
        - L'Adresse Link-local
        - DHCPv6
    - Configurer le Réseau
        - La Commande nmcli
    - LAB #1 - Configuration du Réseau
        - 1.1 - Connections et Profils
        - 1.2 - Résolution des Noms
        - 1.3 - Ajouter une Deuxième Adresse IP à un Profil
        - 1.4 - La Commande hostname
        - 1.5 - La Commande ip
        - 1.6 - Activer/Désactiver une Interface Manuellement
        - 1.7 - Routage Statique
            - La commande ip
            - Activer/désactiver le routage sur le serveur
    - LAB #2 - Diagnostique du Réseau
        - 2.1 - ping
        - 2.2 - netstat -i
        - 2.3 - traceroute
        - 2.4 - tracepath
    - LAB #3 - Connexions à Distance
        - 3.1 - Telnet
        - 3.2 - wget
        - 3.3 - ftp
        - 3.4 - SSH
            - Présentation
                - SSH-1
                - SSH-2

        - Authentification par mot de passe
        - Authentification par clef asymétrique
        - Configuration du Serveur
        - Configuration du Client
        - Tunnels SSH
    - 3.5 - SCP
        - Présentation
        - Utilisation
    - 3.6 - Mise en Place des Clefs Asymétriques

Comprendre les Réseaux

Présentation des Réseaux

La définition d'un réseau peut être résumé ainsi :

- un ensemble d'**Equipements** (systèmes et périphériques) communiquant entre eux,
- une entité destinée au transport de données dans différents environnements.

Pour que la communication soit efficace, elle doit respecter les critères suivants :

- présenter des informations compréhensibles par tous les participants,
- être compatible avec un maximum d'interlocuteurs différents (dans le cas d'un réseau, les interlocuteurs sont des équipements : imprimantes, ordinateurs, clients, serveurs, téléphones...),
- si l'interlocuteur n'est pas disponible, les informations ne doivent pas se perdre,
- permettre une réduction des coûts (par ex. interconnexion à bas coût),
- permettre une productivité accrue (par ex. interconnexion à haut débit),
- être sécurisée si les informations à transmettre sont dites sensibles,
- garantir l'**unicité** et de l'**universalité** de l'**accès à l'information**.

On peut distinguer deux familles d'**Equipements** - les **Eléments Passifs** et les **Eléments Actifs**.

Les **Eléments Passifs** transmettent le signal d'un point à un autre :

- **Les Infrastructures ou Supports** - des câbles, de l'atmosphère ou des fibres optiques permettant de relier **physiquement** des équipements,
- **La Topologie** - l'architecture d'un réseau définissant les connexions entre les **Equipements** et, éventuellement, la hiérarchie entre eux.

Les **Eléments Actifs** sont des équipements qui consomment de l'énergie en traitant ou en interprétant le signal. Les **Equipements** sont classés selon leurs fonctions :

- **Equipement de Distribution Interne au Réseau** - Répartiteur (Hub, Switch, Commutateur etc.), Borne d'accès (Hotspot), Convertisseur de signal (Transciever), Amplificateur (Répéteur) ...,
- **Equipement d'Interconnexion de Réseaux** - Routeurs, Ponts ...,
- **Nœuds** et **Interfaces Réseaux** - postes informatiques, équipements en réseau ....

Un **Nœud** est une extrémité de connexion qui peut être une intersection de plusieurs connexions ou de plusieurs **Equipements**.

Une **Interface Réseau** est une prise ou élément d'un **Equipement Actif** faisant la connexion vers d'autres **Equipements** réseaux et qui reçoit et émet des données.

Important - Dans le cas d'un mélange d'**Equipements** non-homogènes en termes de performances au sein du même réseau, c'est la loi du plus faible qui emporte.

Tous les **Equipements** connectés au même support doivent respecter un ensemble de règles appelé une **Protocole de Communication**.

Les **Protocoles de Communication** définissent de façon formelle et interopérable la manière dont les informations sont échangées entre les **Equipements**.

Des **Logiciels**, dédiés à la gestion de ces **Protocoles de Communication**, sont installés sur des **Equipements d'Interconnexion** afin de fournir des fonctions de contrôle permettant une communication entre les **Equipements**.

Se basant sur des **Protocoles de Communication**, des **Services** fournissent des fonctionnalités accessibles aux utilisateurs ou d'autres programmes.

L'ensemble des **Equipements**, **Logiciels** et **Protocoles de Communication** constitue l'**Architecture Réseau**.

Classification des Réseaux

Les réseaux peuvent être classifiés de trois façon différentes :

- par **Mode de Transmission**,
- par **Topologie**,
- par **Étendue**.

Classification par Mode de Transmission

Il existe deux **Classes** de réseaux dans cette classification :

- les **Réseaux en Mode de Diffusion**,
  - utilise un seul support de transmission,
  - le message est envoyé sur tout le réseau à l'adresse d'**un** destinataire,
- les **Réseaux en Mode Point à Point**,
  - une seule liaison entre deux équipements,
  - les nœuds permettent de choisir la route en fonction de l'adresse du destinataire,
  - quand deux nœuds non directement connectés entre eux veulent communiquer ils le font par l'intermédiaire des autres noeuds du réseau.

Classification par Topologie

Important - La **Topologie Physique** d'un réseau décrit l'organisation de ce dernier en termes de câblage. La **Topologie Logique** d'un réseau décrit comment les données circulent sur le réseau. En effet c'est le choix des concentrateurs ainsi que les connections des câbles qui déterminent la topologie logique.

La Topologie Physique

Il existe 6 topologies physiques de réseau :

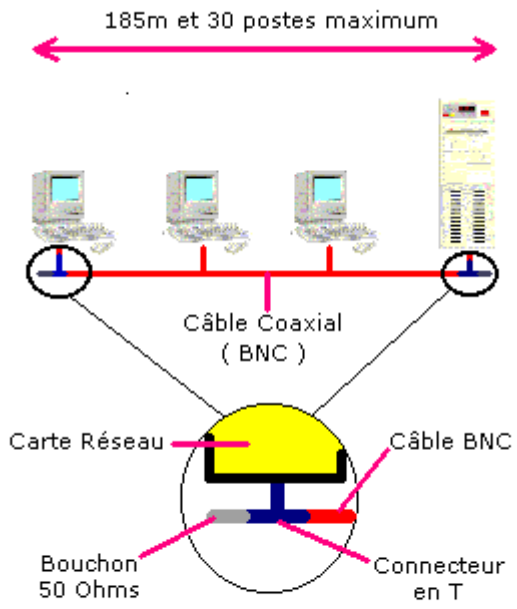
- La Topologie en Ligne,
- La Topologie en Bus,
- La Topologie en Etoile,
- La Topologie en Anneau,
- La Topologie en Arbre,
- La Topologie Maillée.

La Topologie en Ligne

Tous les nœuds sont connectés à un seul support. L'inconvénient de cette topologie est que dans le cas d'une défaillance d'une station, le réseau se trouve coupé en deux sous-réseaux.

La Topologie en Bus

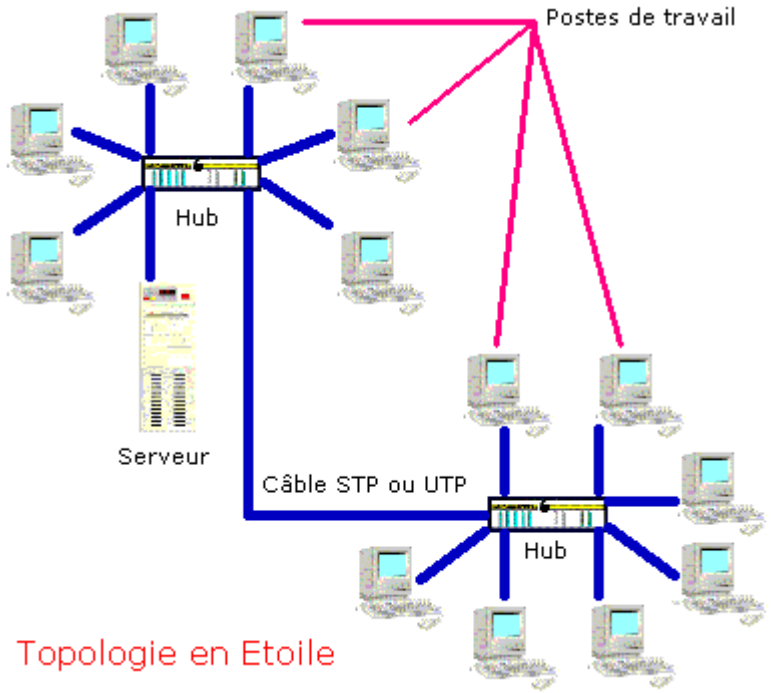
Tous les nœuds sont connectés à un seul support (un câble BNC en T) avec des bouchons à chaque extrémité. La longueur du bus est limitée à **185m**. Le nombre de stations de travail est limité à **30**. Les Stations sont reliées au Bus par des 'T'. Les bouchons sont des terminateurs qui sont des résistances de **50 Ohms**. Quand le support tombe en panne, le réseau ne fonctionne plus. Quand une station tombe en panne, elle ne perturbe pas le fonctionnement de l'ensemble du réseau. Les Stations étant reliés à un suel support, ce type de topologie necessite un **Protocole d'Accès** pour gérer le tour de parole des Stations afin d'éviter des conflits.



Topologie en Bus

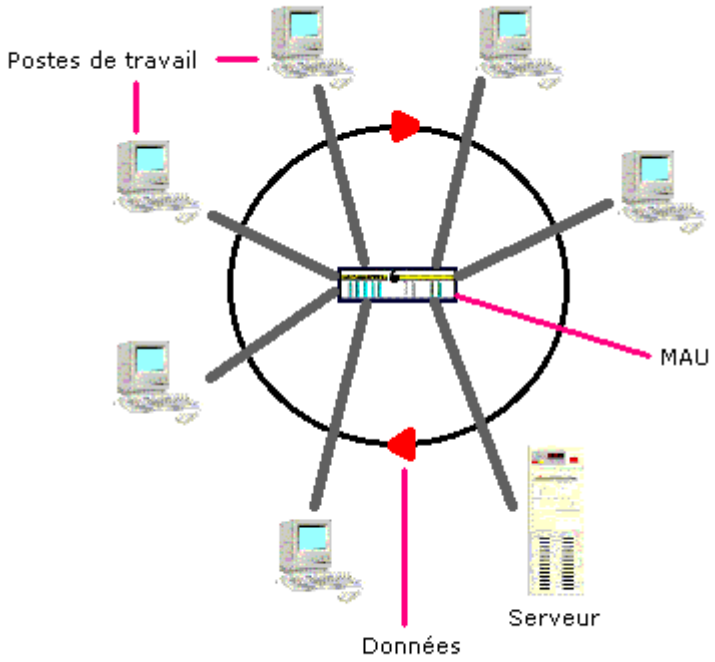
La Topologie en Étoile

Chaque nœud est connecté à un périphérique central appelé un **Hub** (**Concentrateur**) ou un **Switch** (**Commutateur**). Un Hub ou un Switch est prévu pour 4, 8, 16, 32 ... stations. En cas d'un réseau d'un plus grand nombre de stations, plusieurs Hubs ou Switches sont connectés ensemble. Quand une station tombe en panne, elle ne perturbe pas le fonctionnement de l'ensemble du réseau. Le point faible de cette topologie est l'équipement central.



La Topologie en Anneau

Chaque nœud est relié directement à ses deux voisins dans une topologie logique de cercle ininterrompu et une topologie physique en étoile car les stations sont reliées à un type de hub spécial, appelé un **Multistation Access Unit** (MAU).



Topologie en Anneau

Les stations sont reliées à la MAU par un câble 'IBM' munie d'une prise **AUI** du côté de la carte et une prise **Hermaphrodite** du coté de la MAU. Les données sont échangées dans un sens unidirectionnel. Une trame, appelée un **jeton**, circule en permanence. Si l'anneau est brisé, l'ensemble du réseau s'arrête. Pour cette raison, il est courant de voir deux anneaux contre-rotatifs.

La Topologie en Arbre

La Topologie en Arbre est utilisée dans un réseau hierarchique où le sommet, aussi appelé la **racine**, est connecté à plusieurs noeuds de niveau inférieur. Ces neouds peuvent à leur tour être connectés à d'autres noeuds inférieurs. L'ensemble forme une arborescence. Le point faible de cette topologie est sa racine. En cas de défaillance, le réseau est coupé en deux.

La Topologie Maillée

Cette Topologie est utilisée pour des grands réseaux de distribution tels Internet ou le WIFI. Chaque noeud à tous les autres via des liaisons point à

point. Le nombre de liaisons devient très rapidement important en cas d'un grand nombre de noeuds. Par exemple dans le cas de 100 Stations (N), le nombre de liaisons est obtenu par la formule suivante :

```
N(N-1)/2 = 100(100-1)/2 = 4 950
```

Important - La **Topologie Physique** la plus répandue est la **Topologie en Etoile**.

Classification par Etendue

La classification par étendue nous fournit 4 réseaux principaux :

Nom	Description	Traduction	Taille Approximative (M)
PAN	Personal Area Network	Réseau Personnel	1 -10
LAN	Local Area Network	Réseau Local Entreprise (RLE)	5 - 1 200
MAN	Métropolitain Area Network	Réseau Urbain	900 - 100 000
WAN	Wide Area Network	Réseau Long Distance (RLD)	50 000 et au delà

Cependant, d'autres classification existent :

CAN	Campus Area Network	Réseau de Campus
GAN	Global Area Network	Réseau Global
TAN	Tiny Area Network	Réseau Minuscule
FAN	Family Area Network	Réseau Familial
SAN	Storage Area Network	Réseau de Stockage

Important - Etant donné que les WANs sont gérés par des opérateurs de télécommunications qui doivent demander une licence à l'état mais que les LANs ont été historiquement mis en oeuvre dans les entreprises, ces derniers sont en majorité issus du

monde informatique.

Les Types de LAN

Il existe deux types de LAN :

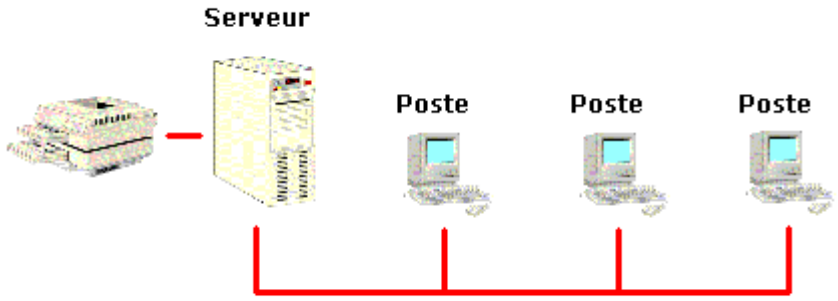
- le réseau à serveur dédié,
- le réseau poste à poste.

Réseau à Serveur Dédié

Le réseau à serveur dédié est caractérisé par le fait que toutes les ressources (imprimantes, applications, lecteurs etc.) sont gérées par le serveur. Les autres micro-ordinateurs ne jouent le rôle de client.

Des exemples des systèmes d'exploitation du réseau à serveur dédié sont :

- Windows NT Server,
- Windows 2000 Server,
- Windows 2003 Server,
- Windows 2008 Server,
- Linux,
- Unix.

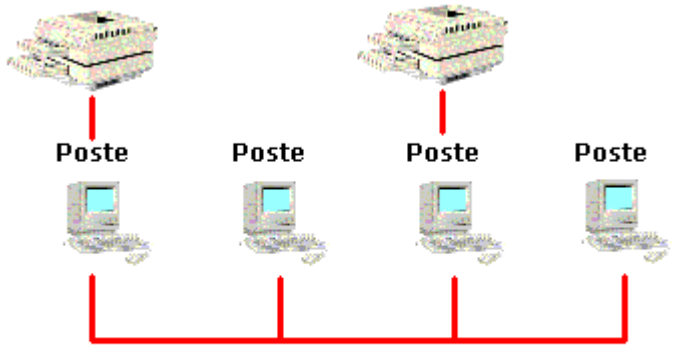


Réseau à Serveur Dédié

Réseau Poste-à-Poste

Le réseau poste à poste est caractérisé par le fait que tous les ordinateurs peuvent jouer le rôle de client et de serveur :

- Windows 95,
- Windows 98,
- Windows NT Workstation.



Réseau Poste à Poste

Le Modèle Client/Serveur

Le modèle Client/Serveur est une des modalités des architectures informatiques distribuées. Dans ce modèle un serveur est tout **Logiciel** fournissant un **Service**.

Le serveur est aussi :

- passif, c'est-à-dire en attente permenante d'une demande, appelée une requête d'un client,
- capable de traiter plusieurs requêtes simultanément en utilisant le **multi-threading**,
- garant de l'intégrité globale.

Le client est, par contre **actif**, étant à l'origine des requêtes.

Il existe trois types de modèle client/serveur :

- **Plat** - tous les clients communiques avec un seul serveur,
- **Hiérarchique** - les clients n'ont de contact qu'avec les serveurs de plus haut niveau qu'eux,
- **Peer-to-Peer** - les équipements sont à la fois client **et** serveur en même temps.

Modèles de Communication

Les réseaux sont bâtis sur des technologies et des modèles. Le modèle **théorique** le plus important est le modèle **O**pen **S**ystem **Interconnection** créé par l'**I**nternational **Organization** for **S**tandardization tandis que le modèle pratique le plus important est le modèle **TCP/IP**.

Le modèle OSI

Le modèle OSI qui a été proposé par l'ISO est devenu le standard en termes de modèle pour décrire l'échange de données entre ordinateurs. Cette norme se repose sur sept couches, de la une - la Couche Physique, à la sept - la Couche d'Application, appelés des services. La communication entre les différentes couches est synchronisée entre le poste émetteur et le poste récepteur grâce à ce que l'on appelle un protocole.

Ce modèle repose sur trois termes :

- Les **Couches**,
- Les **Protocoles**,
- Les **Interfaces**.

Les Couches

Des sept couches :

- Les couches 1 à 3 sont les **Couches Basses** orientées **Transmission**,
- La couche 4 est la **Couche Charnière** entre les **Couches Basses** et les **Couches Hautes**,
- Les couches 5 à 7 sont les **Couches Hautes** orientées **Traitement**.

La couche du même niveau du système **A** parle avec son homologue du système **B**.

- **La Couche Physique** (Couche 1) est responsable :
  - du transfert de données binaires sur le câble physique ou virtuel
  - de la définition de tout aspect physique allant du connecteur jusqu'au câble en passant par la carte réseau, y compris l'organisation même du réseau
  - de la définition des tensions électriques sur le câble pour obtenir le 0 et le 1 binaires
- **La Couche de Liaison** (Couche 2) est responsable :
  - de la réception des données de la couche physique
  - de l'organisation des données en fragments, appelés des trames qui ont un format différent selon s'il s'agit d'un réseau basé sur la technologie Ethernet ou la technologie Token-Ring
  - de la préparation, émission et réception des trames
  - de la gestion de l'accès au réseau
  - de la communication nœud à nœud
  - de la gestion des erreurs
    - avant la transmission, le nœud émetteur calcule un code appelé un CRC et l'incorpore dans les données envoyées
    - le nœud récepteur recalcule un CRC en fonction du contenu de la trame reçue et le compare à celui incorporé avec l'envoi
    - en cas de deux CRC identique, le nœud récepteur envoie un accusé de réception au nœud émetteur
  - de la réception de l'accusé de réception
  - éventuellement de le ré-émission des données
  - En prenant ce modèle, l'IEEE (Institute of Electrical and Eletronics Engineers) l'a étendu avec le Modèle IEEE (802).
    - Dans ce modèle la Couche de Liaison est divisée en deux sous-couches importantes :

    - La **Sous-Couche LLC** (Logical Link Control) qui :
        - gère les accusés de réception
        - gère le flux de trames
    - La **Sous-Couche MAC** (Media Access Control) qui :
        - gère la méthode d'accès au réseau
        - le CSMA/CD dans un réseau basé sur la technologie Ethernet
        - l'accès au jeton dans un réseau basé sur la technologie Token-Ring
        - gère les erreurs

- **La Couche de Réseau** (Couche 3) est responsable de la gestion de la bonne distribution des différentes informations aux bonnes adresses en :
    - identifiant le chemin à emprunter d'un nœud donné à un autre
    - appliquant une conversion des adresses logiques (des noms) en adresses physiques
    - ajoutant des information adressage aux envois
    - détectant des paquets trop volumineux avant l'envoi et en les divisant en trames de données de tailles autorisées

- **La Couche de Transport** (Couche 4) est responsable de veiller à ce que les données soient envoyées correctement en :
    - constituant des paquets de données corrects
    - les envoyant dans le bon ordre
    - vérifiant que les données sont traités dans le même ordre que l'ordre d'émission
    - permettant à un processus sur un nœud de communiquer avec un autre nœud et d'échanger des messages avec lui

- **La Couche de Session** (Couche 5) est responsable :
    - de l'établissement, du maintien, et de la mise à fin de la communication entre deux noeuds distants, c'est-à-dire, de la session
    - de la conversation entre deux processus de vérification de la réception des messages envoyés en séquences, c'est-à-dire, le point de contrôle

- de la sécurité lors de l'ouverture de la session, c'est-à-dire, les droits d'utilisateurs etc.

- **La Couche de Présentation** (Couche 6) est responsable :
    - du formatage et de la mise en forme des données
    - des conversions de données telles le cryptage/décryptage

- **La Couche d'Application** (Couche 7) est responsable :
    - du dialogue homme/machine via des messages affichés
    - du partage des ressources
    - de la messagerie

Les Protocoles

Un **protocole** est un langage commun utilisé par dexu entités en communication pour pouvoir se comprendre. La nature du Protocole dépends directement de la nature de la communication. Cette bature dépend du **paradigme** de communication que l'application nécessite. Le paradigme est un modèle abstrait d'un problème ou d'une situation. Dans le paradigme de la diffusion, l'émetteur envoie dans informations au récepteur sans se soucier de ce que le récepteur va en faire. C'est la responsabilité du récepteur de comprendre et d'utiliser les informations.

Les Interfaces

Chaque couche rend des **services** à la couche immédiatement supérieure et utilise les services de la couche immédiatement inférieure. L'ensemble des services s'appelle une **Interface**. Les services sont composés de **S**ervice **D**ata **U**nits et sont disponibles par un **S**service **A**ccess **P**oint.

Protocol Data Units

L'**Unité de Données** ou *Protocol Data Unit* pour chaque couche comporte un nom spécifique :

- **Application Protocol Data Units** pour la couche **Application**,
- **Présentation Protocol Data Units** pour la couche **Présentation**,
- **Session Protocol Data Units** pour la couche **Session**,
- **Transport Protocol Data Units** pour la couche **Transport**.

Or, pour les **Couches Basses** on parle de :

- **Paquets** pour la couche **Réseau**,
- **Trames** pour la couche **Liaison**,
- **Bits** pouyr la couche **Physique**.

Encapsulation et Désencapsulation

Lorque les données sont communiqueés par le système A au système B, celles-ci commencent au niveau de la couche d'Application. Le couche d'Application ajoute une en-tête à l'unité de données qui contient des **informations de contrôle du protocole**. Au passage de chaque couche, celle-ci ajoute sa propre en-tête. De cette façon, lors de sa descente vers la couche physique, les données et l'entête de la couche supérieure sont encapulsulées :

Couche Système A	Encapsulation
Application	Application Header (AH) + Unité de Données (UD)

Couche Système A	Encapsulation
Présentation	Présentation Header (PH) + AH + UD
Session	Session Header (SH) + PH + AH + UD
Transport	Transport Header (TH) + SH + PH + AH + UD
Réseau	Network Header (NH) + TH + SH + PH + AH + UD
Liaison	Liaison Header (DH) + NH + TH + SH + PH + AH + UD

Lors de son voyage de la couche Physique vers la couche Application dans le système B, les en-têtes sont supprimées par chaque couche correspondante. On parle alors de **désencapsulation** :

Couche Système B	Encapsulation
Liaison	Liaison Header (DH) + NH + TH + SH + PH + AH + UD
Réseau	Network Header (NH) + TH + SH + PH + AH + UD
Transport	Transport Header (TH) + SH + PH + AH + UD
Session	Session Header (SH) + PH + AH + UD
Présentation	Présentation Header (PH) + AH + UD
Application	Application Header (AH) + Unité de Données (UD)

Spécification NDIS et le Modèle ODI

La spécification NDIS (Network Driver Interface Specification) a été introduite conjointement par les sociétés Microsoft et 3Com. Cette spécification ainsi que son homologue, le modèle ODI (Open Datalink Interface) introduit conjointement par les sociétés Novell et Apple à la même époque, définit des standards pour les pilotes de cartes réseau afin qu'ils puissent être indépendants des protocoles utilisées et les systèmes d'exploitation sur les machines. Des deux 'standards', la spécification NDIS est le plus répandu, intervenant a niveau de la sous-couche MAC et l a couche de liaison. Elle spécifie :

- l'interface pilote-matériel
- l'interface pilote-protocole
- l'interface pilote - système d'exploitation

Le modèle TCP/IP

La suite des protocoles TCP/IP (Transmission Control Protocol / Internet Protocol) est issu de la DOD (Dept. Américain de la Défense) et le travail de l'ARPA (Advanced Research Project Agency).

- La suite des protocoles TCP/IP
  - a été introduite en 1974
  - a été utilisée dans l'ARPAnet en 1975
  - permet la communication entre des réseaux à base de systèmes d'exploitation, architectures et technologies différents
  - est très proche du modèle OSI en termes d'architecture et se place au niveau de la couche d'Application jusqu'à la couche Réseau.
  - est, en réalité, une suite de protocoles et de services :
    - **IP** (Internet Protocol)
      - le protocole IP s'intègre dans la couche Réseau du modèle OSI en assurant la communication entre les systèmes. Bien qu'il puisse découper des messages en fragments ou datagrammes et les reconstituer dans le bon ordre à l'arrivée, il ne garantit pas la réception.
    - **ICMP** (Internet Control Message Protocol)
      - le protocole ICMP produit des messages de contrôle aidant à synchroniser le réseau. Un exemple de ceci est la commande ping.
    - **TCP** (Transmission Control Protocol)
      - le protocole TCP se trouve au niveau de la couche de Transport du modèle OSI et s'occupe de la transmission des données entre noeuds.
    - **UDP** (User Datagram Protocol)
      - le protocole UDP n'est pas orienté connexion. Il est utilisé pour la transmission rapide de messages entre nœuds sans garantir leur acheminement.
    - **Telnet**
      - le protocole Telnet est utilisé pour établir une connexion de terminal à distance. Il se trouve dans la couche d'Application du modèle OSI.
    - **Ftp** (File Transfer Protocol)
      - le protocole ftp est utilisé pour le transfert de fichiers. Il se trouve dans la couche d'Application du modèle OSI.
    - **SMTP** (Simple Message Transfer Protocol)
      - le service SMTP est utilisé pour le transfert de courrier électronique. Il se trouve dans la couche d'Application du modèle OSI.
    - **DNS** (Domain Name Service)
      - le service DNS est utilisé pour le résolution de noms en adresses IP. Il se trouve dans la couche d'Application du modèle OSI.
    - **SNMP** (Simple Network Management Protocol)

        - le protocole SNMP est composé d'un agent et un gestionnaire. L'agent SNMP collecte des informations sur les périphériques, les configurations et les performances tandis que le gestionnaire SNMP reçois ses informations et réagit en conséquence.
    - **NFS** (Network File System)
        - le NFS a été mis au point par Sun Microsystems
        - le NFS génère un lien virtuel entre les lecteurs et les disques durs permettant de monter dans un disque virtuel local un disque distant
    - et aussi POP3, NNTP, IMAP etc ...

Le modèle TCP/IP est composé de 4 couches :

- La couche d'Accès Réseau
    - Cette couche spécifie la forme sous laquelle les données doivent être acheminées, quelque soit le type de réseau utilisé.
- La couche Internet
    - Cette couche est chargée de fournir le paquet de données.
- La couche de Transport
    - Cette couche assure l'acheminement des données et se charge des mécanismes permettant de connaître l'état de la transmission.
- La couche d'Application
    - Cette couche englobe les applications standards de réseau telles ftp, telnet, ssh, etc..

Les noms des Unités de Données sont différents selon le protocole utilisé et la couche du modèle TCP/IP :

Couche	**TCP**	**UDP**
Application	Stream	Message
Transport	Segment	Packet
Internet	Datagram	Datagram
Réseau	Frame	Frame

Les Raccordements

Les Modes de Transmission

On peut distinguer 3 modes de transmission :

- La **Liaison Simplex**,
  - Les données ne circulent que dans un **seul** sens de l'émetteur ver le récepteur,
  - La liaison nécessite deux canaux de transmissions,
- La **Liaison Half-Duplex** aussi appelée la **Liaison à l'Alternat** ou encore la **Liaison Semi-Duplex**,
  - Les données circulent dans un sens ou l'autre mais jamais dans les deux sens en même temps. Chaque extrémité émet donc à son tour,
  - La liaison permet d'avoir une liaison bi-directionnelle qui utilise la totalité de la banse passante,
- La **Liaison Full-Duplex** dans les deux sens en **même** temps. Chaque extrémité peut émettre et recevoir simultanément,
  - La liaison est caractérisée par une bande passante divisée par deux pour chaque sens des émissions.

Les Câbles

Le Câble Coaxial

En partant de l'extérieur, le câble coaxial est composé :

- d'une **Gaine** en caoutchouc, PVC ou Téflon pour protéger le câble,
- d'un **Blindage** en métal pour diminuer le bruit du aux parasites,
- d'un **Isolant** (diélectrique) pour éviter le contact entre le blindage et l'âme et ainsi éviter des courts-circuits,
- d'un **Âme** en cuivre ou torsadés pour transporter les données.

Avantages :

- **Peux coûteux**,
- Facilement **manipulable**,
- Peut être utilisé pour de **longues distances**,
- A un débit de 10 Mbit/s dans un LAN et 100 Mbit/s dans un WAN.

Inconvénients :

- Fragile,
- Instable,
- Vulnérable aux interférences,
- Half-Duplex.

Le Câble Paire Torsadée

Ce câble existe sous deux formes selon son utilisation :

- **Monobrin** pour du câblage **horizontal** (**Capillaire**),
  - chaque fil est composé d'un seul conducteur en cuivre,
  - la distance ne doit pas dépassée 90m.
- **Multibrin** pour des **cordons de brassage** :
  - chaque fil est composé de plusieurs brins en cuivre,
  - câble souple.

Avantages :

- Un débit de 10 Mbit/s à 10 GBit/s,
- A une bande passante plus large,
- Pas d'interruption par coupure du câble,
- Permet le **câblage universel** (téléphonie, fax, données ...),
- Full-Duplex.

Inconvénients :

- Nombre de câbles > câble coaxial,
- Plus cher,
- Plus encombrant dans les gaines techniques.

Catagories de Blindage

Il existe trois catagories de blindage :

- **Twisted** ou Torsadé,
- **Foiled** ou Entouré,
- **Shielded** ou Avec Ecran.

De ce fait, il existe 5 catagories de câbles Paire Torsadée :

Nom anglais ^ Appelation Ancienne ^ Nouvelle Appelation ^

Unshielded Twisted Pair	UTP	U/UTP
Foiled Twisted Pair	FTP	F/UTP
Shield Twisted Pair	STP	S/UTP
Shield Foiled Twisted Pair	SFTP	SF/UTP
Shield Shield Twisted Pair	S/STP	SS/STP3

Ces catégories donnent lieu à des **Classes** :

Classe	**Débit**	**Nombre de Paires Torsadées**	**Connecteur**	**Commentaires**
3	10 Mbit/s	4	RJ11	
4	16 Mbit/s	4	S/O	Non-utilisée de nos jours
5	100 Mbit/s	4	RJ45	Obsolète
5e/D	1 Gbit/s sur 100m	4	RJ45	S/O
6/E	2.5 Gbit/s sur 100m ou 10 Gbit/s sur 25m à 55m	4	Idéal pour PoE	
7/F	10 Gbit/s sur 100m	4	GG45 ou Tera	Paires individuellement et collectivement blindées. Problème de compatibilité avec les classes precédentes due au connecteur.

La Prise RJ45

Une prise RJ45 comporte 8 broches. Un câble peut être **droit** quand la broche 1 d'une extremité est connectée à la broche 1 de la prise RJ45 à l'autre extrémité, la broche 2 d'une extremité est connectée à la broche 2 de la prise RJ45 à l'autre extrémité et ainsi de suite ou bien **croisé** quand le brochage est inversé.

Les câbles croisés sont utilisés lors du branchement de deux équipements identiques (PC à PC, Hub à Hub, Routeur à Routeur).

Channel Link et Basic Link

Le **Channel Link** ou **Canal** est l'ensemble du **Basic Link** ou **Lien** de base et les cordons de brassage et de raccordement des équipements qui sont limités en distance à 10m.

Le **Basic Link** est le lien entre la prise RJ45 murale et la baie de brassage. Il est limité à 90m en classe 5D.

La Fibre Optique

La **Fibre Optique** est un fil de **Silice** permettant le transfert de la lumière. De ce fait elle est caractérisée par :

- des meilleures performances que le cuivre,
- de plus de communications simultanément,
- de la capacité de relier de plus grandes distances,
- une insensibilité aux perturbations,
- une résistance à la corrosion.

Qui plus est, elle ne produit aucune perturbation.

Elle est composée :

- d'un coeur de 10, de 50/125 ou de 62.50 micron,
- d'une gaine de 125 micron,
- d'une protection de 230 micron.

Il existe deux types de fibres, la **Fibre Monomode** et la **Fibre Multimodes**.

La Fibre Monomode :

- a un coeur de 8 à 10 Microns,
- est divisée en sous-catégories de distance,
    - 10 Km,
    - 15 Km,
    - 20 Km,
    - 50 Km,
    - 80 Km,
    - 100 Km.

La Fibre Multimode :

- a un coeur de 62,50 micron ou de 50/125 micron avec une gaine orange,
- permet plusieurs trajets lumineux appelés **modes** en même temps en Full Duplex,
- est utilisée pour de bas débits ou de courtes distances,
    - 2 Km pour 100 Mbit/s,
    - 500 m pour 1 Gbit/s.

Les Réseaux sans Fils

Les réseaux sans fils sans basés sur une liaison qui utilise des ondes radio-électriques (radio et infra-rouges).

Il existe des technologies différentes en fonction de la fréquence utilisée et de la portée des transmissions :

- Réseaux Personnels sans Fils - Bluetooth, HomeRF,
- Réseaux Locaux sans Fils - LiFI, WiFI,
- Réseaux Métropolitains sans Fil - wImax,
- Réseaux Etendus sans Fils - GSM, GPRS, UMTS.

Les principales ondes utilisées pour la transmission des données sont :

- Ondes GSM - Ondes Hertziennes repeosant sur des micro-ondes à basse fréquence avec une portée d'une dizaine de kilomètres,
- Ondes Wi-Fi - Ondes Hertziennes reposant sur des micro-ondes à haute fréquence avec une portée de 20 à 50 mètres,
- Ondes Satellitaires - Ondes Hertziennes longues portées.

Le Courant Porteur en Ligne

Le CPL utilise le réseau électrique domestique, le réseau moyenne et basse tension pour transmettre des informations numériques.

Le CPL superpose un signal à plus haute fréquence au signal électrique.

Seuls donc, les fils conducteurs transportent les signaux CPL.

Le coupleur intégré en entrée des boîtiers CPL élimine les composants basses fréquences pour isoler le signal CPL.

Le CPL utilise la phase électrique et le neutre. De ce fait, une installation triphasée fournit 3 réseaux CPL différents.

Le signal CPL ne s'arrête pas necéssairement aux limites de l'installation électrique. En effet en cas de compteurs non-numériques le signal les traversent.

Les normes CPL sont :

Norme	Débit Théorique	Débit Pratique	Temps pour copier 1 Go
Homeplug 1.01	14 Mbps	5.4 Mbps	25m 20s
Homeplug 1.1	85 Mbps	12 Mbps	11m 20s
PréUPA 200	200 Mbps	30 Mbps	4m 30s

Technologies

Il existe plusieurs technologies de réseau :

- Ethernet,
- Token-Ring,
- ARCnet,
- etc..

Nous détaillerons ici les deux technologies les plus répandues, à savoir Ethernet et Token-Ring.

Ethernet

La technologie Ethernet se repose sur :

- une topologie logique de bus,
- une topologie physique de bus ou étoile.

L'accès au bus utilise le **CSMA/CD**, Carrier Sense Multiple Access / Collision Detection (Accès Multiple à Détection de Porteuse / Détection de Collisions).

Il faut noter que :

- les données sont transmises à chaque nœud - c'est la méthode d'**accès multiple**,
- chaque nœud qui veut émettre écoute le réseau - c'est la **détection de porteuse**,
- quand le réseau est silencieux une trame est émise dans laquelle se trouvent les données ainsi que l'adresse du destinataire,
- le système est dit donc **aléatoire** ou **non-déterministe**,
- quand deux nœuds émettent en même temps, il y a **collision de données**,
- les deux nœuds vont donc cesser d'émettre, se mettant en attente jusqu'à ce qu'ils commencent à émettre de nouveau.

Token-Ring

La technologie Token-Ring se repose sur :

- une topologie logique en anneau,
- une topologie physique en étoile.

Token-Ring se traduit par **Anneau à Jeton**. Il n'est pas aussi répandu que l'Ethernet pour des raisons de coûts. En effet le rajout d'un nœud en Token-Ring peut coûter jusqu'à **4 fois plus cher qu'en Ethernet**.

Il faut noter que :

- les données sont transmises dans le réseau par un système appelé **méthode de passage de jeton**,
- le jeton est une **trame numérique vide** de données qui tourne en permanence dans l'anneau,
- quand un nœud souhaite émettre, il saisit le jeton, y dépose des données avec l'adresse du destinataire et ensuite laisse poursuivre son chemin jusqu'à sa destination,
- pendant son voyage, aucun autre nœud ne peut émettre,
- une fois arrivé à sa destination, le jeton dépose ses données et retourne à l'émetteur pour confirmer la livraison,
- ce système est appelé **déterministe**.

L'intérêt de la technologie Token-Ring se trouve dans le fait :

- qu'il **évite des collisions**,
- qu'il est **possible de déterminer avec exactitude le temps que prenne l'acheminement des données**.

La technologie Token-Ring est donc idéale, voire obligatoire, dans des installations où chaque nœud doit disposer d'une opportunité à intervalle fixe d'émettre des données.

Périphériques Réseaux Spéciaux

En plus du câblage, les périphériques de réseau spéciaux sont des éléments primordiaux tant au niveau de la topologie physique que la topologie logique.

Les périphériques de réseau spéciaux sont :

- les Concentrateurs ou *Hubs*,
- les Répéteurs ou *Repeaters*,
- les Ponts ou *Bridges*,
- les Commutateurs ou *Switches*,
- les Routeurs ou *Routers*,
- les Passerelles ou *Gateways*.

L'objectif ici est de vous permettre de comprendre le rôle de chaque périphérique.

Les Concentrateurs

Les Concentrateurs permettent une connectivité entre les nœuds en topologie en étoile. Selon leur configuration, la topologie logique peut être en étoile, en bus ou en anneau. Il existe de multiples types de Concentrateurs allant du plus simple au Concentrateur intelligent.

- **Le Concentrateur Simple**
  - est une boîte de raccordement centrale,
  - joue le rôle de récepteur et du réémetteur des signaux sans accélération ni gestion de ceux-ci,
  - est un périphérique utilisé pour des groupes de travail.
- **Le Concentrateur Évolué**
  - est un Concentrateur simple qui offre en plus l'amplification des signaux, la gestion du type de topologie logique grâce à des capacités d'être configurés à l'aide d'un logiciel ainsi que l'homogénéisation du réseau en offrant des ports pour un câblage différent. Par exemple, 8 ports en paire torsadée non-blindée et un port BNC.
- **Le Concentrateur Intelligent**
  - est un Concentrateur évolué qui offre en plus la détection automatique des pannes, la connectique avec un Pont ou un Routeur ainsi que le

diagnostic et la génération de rapports.

Les Répéteurs

Un Répéteur est un périphérique réseau simple. Il est utilisé pour amplifier le signal quand :

- la longueur du câble dépasse la limite autorisée,
- le câble passe par une zone ou les interférences sont importantes.

Éventuellement, et uniquement dans le cas où le Répéteur serait muni d'une telle fonction, celui-ci peut être utiliser pour connecter deux réseaux ayant un câblage différent.

Les Ponts

Un Pont est **Répéteur intelligent**. Outre sa capacité d'amplifier les signaux, le Pont analyse le trafic qui passe par lui et met à jour une liste d'adresses des cartes réseau, appelée **une table de routage**, n'autorisant que les transmissions destinées à d'autres segments du réseau.

Les **diffusions** sont néanmoins autorisées.

Comme un Pont doit être intelligent, on utilise souvent un micro-ordinateur comme Pont. Forcément équipé de 2 cartes réseau, le Pont peut également jouer le rôle de serveur de fichiers.

Le Pont sert donc à isoler des segments du réseau pour des raisons de :

- **sécurité** afin d'éviter à ce que des données sensibles soient propagées sur tout le réseau,
- **performance** afin qu'une partie du réseau trop chargée ralentisse le réseau entier,
- **fiabilité** afin par exemple qu'une carte en panne ne gène pas le reste du réseau avec une diffusion.

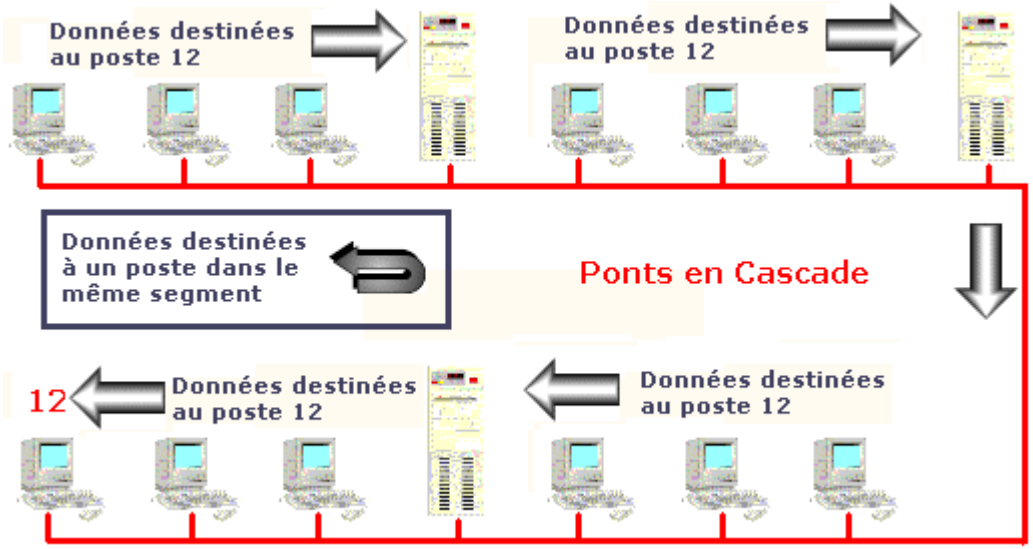
Il existe trois types de configuration de Ponts

Le Pont de Base

Le Pont de Base est utilisé très rarement pour isoler deux segments.

Le Pont en Cascade

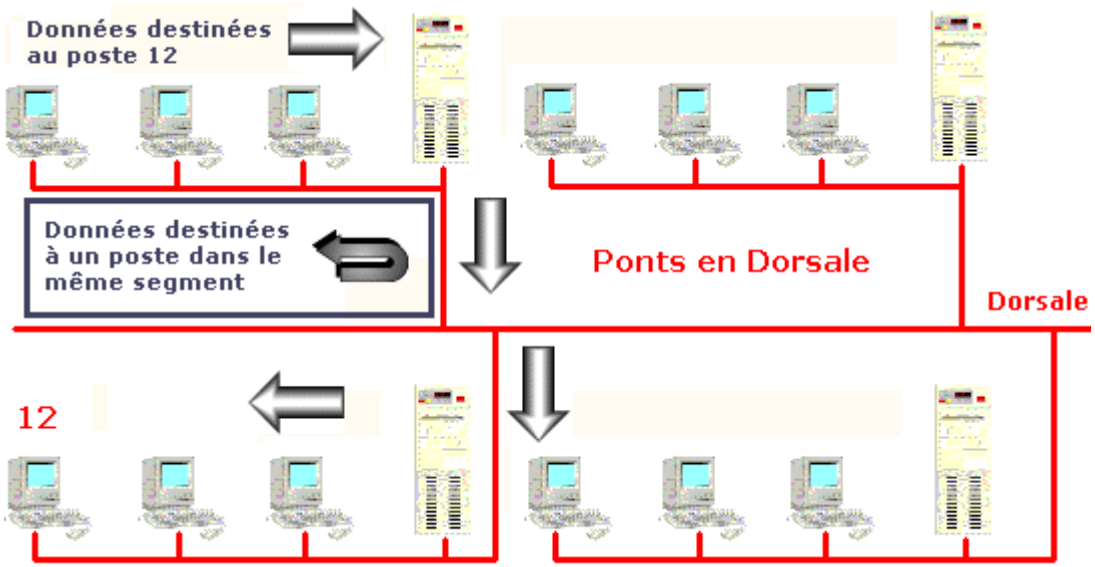
Le Pont en Cascade est à éviter car les données en provenance d'un segment doivent passer par plusieurs Ponts. Ceci a pour conséquence de ralentir la transmission des données, voire même de créer un trafic superflu en cas de rémission par le nœud



Le Pont en Dorsale

Le Pont en Dorsale coûte plus chère que la configuration précédente car il faut un nombre de Ponts équivalent au nombre de segments + 1. Par contre

elle réduit les problèmes précédemment cités puisque les données ne transitent que par deux Ponts.



Les Commutateurs

Un Commutateur peut être considéré comme un Concentrateur intelligent et un Pont. Ils sont gérés souvent par des logiciels. La topologie physique d'un réseau commuté est en étoile. Par contre la topologie logique est spéciale, elle s'appelle une topologie commutée.

Lors de la communication de données entre deux nœuds, le Commutateur ouvre une connexion temporaire virtuelle en fermant les autres ports. De cette façon la bande passante totale est disponible pour cette transmission et les risques de collision sont minimisés.

Certains Commutateurs haut de gamme sont équipés d'un système anti-catastrophe qui leur permet d'isoler une partie d'un réseau en panne afin que les autres parties puissent continuer à fonctionner sans problème.

Les Routeurs

Un Routeur est un Pont sophistiqué capable :

- d'assurer l'interconnexion entre des segments,
- de filtrer le trafic,
- d'isoler une partie du réseau,
- d' explorer les informations d'adressage pour trouver le chemin le plus approprié et le plus rentable pour la transmission des données.

Les Routeurs utilisent une table de routage pour stocker les informations sur :

- les adresses du réseau,
- les solutions de connexion vers d'autres réseaux,
- l'efficacité des différentes routes.

Il existe deux types de Routeur :

- le **Routeur Statique**
  - la table de routage est éditer manuellement,
  - les routes empruntées pour la transmission des données sont toujours les mêmes,
  - il n'y a pas de recherche d'efficacité.
- le **Routeur Dynamique**
  - découvre automatiquement les routes à emprunter dans un réseau.

Les Passerelles

Ce périphérique, souvent un logiciel, sert à faire une conversion de données :

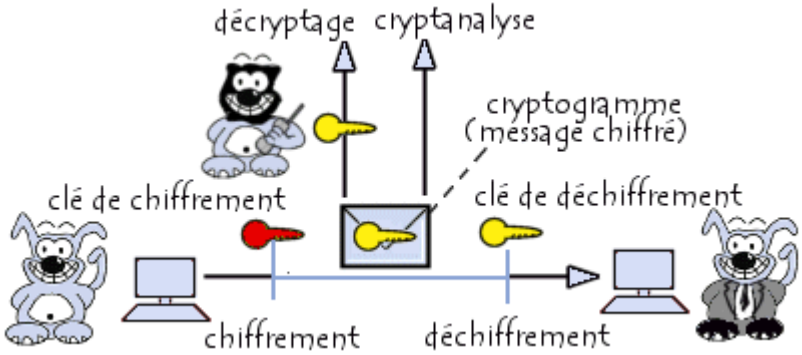
- entre deux technologies différentes (Ethernet - Token-Ring),
- entre deux protocoles différents,
- entre des formats de données différents.

Comprendre le Chiffrement

Introduction à la cryptologie

Définitions

- **La Cryptologie**
  - La science qui étudie les aspects scientifiques de ces techniques, c'est-à-dire qu'elle englobe la cryptographie et la cryptanalyse.
- **La Cryptanalyse**
  - Lorsque la clef de déchiffrement n'est pas connue de l'attaquant on parle alors de cryptanalyse ou cryptoanalyse (on entend souvent aussi le terme plus familier de cassage).
- **La Cryptographie**
  - Un terme générique désignant l'ensemble des techniques permettant de chiffrer des messages, c'est-à-dire permettant de les rendre inintelligibles sans une action spécifique. Les verbes crypter et chiffrer sont utilisés.
- **Le Décryptement ou Décryptage**
  - Est le fait d'essayer de déchiffrer illégitimement le message (que la clé de déchiffrement soit connue ou non de l'attaquant).



La Cryptographie

La cryptographie apporte quatre points clefs:

- La confidentialité
  - consiste à rendre l'information inintelligible à d'autres personnes que les acteurs de la transaction.

- L'intégrité
  - consiste à déterminer si les données n'ont pas été altérées durant la communication (de manière fortuite ou intentionnelle).
- L'authentification
  - consiste à assurer l'identité d'un utilisateur.
- La non-répudiation
  - est la garantie qu'aucun des correspondants ne pourra nier la transaction.

La cryptographie est basée sur l'arithmétique. Il s'agit, dans le cas d'un texte, de transformer les lettres qui composent le message en une succession de chiffres (sous forme de bits dans le cas de l'informatique), puis ensuite de faire des calculs sur ces chiffres pour:

- Procéder au chiffrement
  - Le résultat de cette modification (le message chiffré) est appelé cryptogramme (Ciphertext) par opposition au message initial, appelé message en clair (Plaintext)
- Procéder au déchiffrement

Le chiffrement se fait à l'aide d'une clef de chiffrement. Le déchiffrement nécessite une clef de déchiffrement.

On distingue deux types de clefs:

- Les clés symétriques:
  - des clés utilisées pour le chiffrement ainsi que pour le déchiffrement. On parle alors de chiffrement symétrique ou de chiffrement à clé secrète.
- Les clés asymétriques:
  - des clés utilisées dans le cas du chiffrement asymétrique (aussi appelé chiffrement à clé publique). Dans ce cas, une clé différente est utilisée pour le chiffrement et pour le déchiffrement.

Le Chiffrement par Substitution

Le chiffrement par substitution consiste à remplacer dans un message une ou plusieurs entités (généralement des lettres) par une ou plusieurs autres entités. On distingue généralement plusieurs types de cryptosystèmes par substitution :

- La substitution **monoalphabétique**
  - consiste à remplacer chaque lettre du message par une autre lettre de l'alphabet
- La substitution **polyalphabétique**

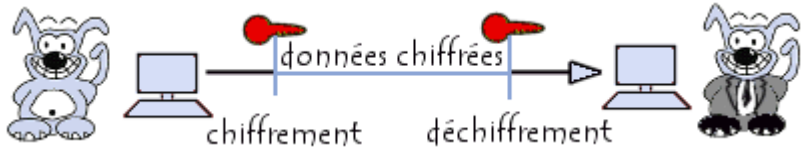
    - consiste à utiliser une suite de chiffres monoalphabétique réutilisée périodiquement
- La substitution **homophonique**
    - permet de faire correspondre à chaque lettre du message en clair un ensemble possible d'autres caractères
- La substitution de **polygrammes**
    - consiste à substituer un groupe de caractères (polygramme) dans le message par un autre groupe de caractères

Algorithmes à clé secrète

Le Chiffrement Symétrique

Ce système est aussi appelé le système à **Clef Secrète** ou à **clef privée**.

Ce système consiste à effectuer une opération de chiffrement par algorithme mais comporte un inconvénient, à savoir qu'il nécessite un canal sécurisé pour la transmission de la clef de chiffrement/déchiffrement.



Important - Le système de Méthode du Masque Jetable (One Time Pad) fût mis au point dans les années 1920. Il utilisait une clef générée aléatoirement à usage unique.

Les algorithmes de chiffrement symétrique couramment utilisés en informatique sont:

- **Data Encryption Standard** (DES),
- **Triple DES** (3DES),
- **RC2**,
- **Blowfish**,
- **International Data Encryption Algorithm** (IDEA),

- **Advanced Encryption Standard** (AES).

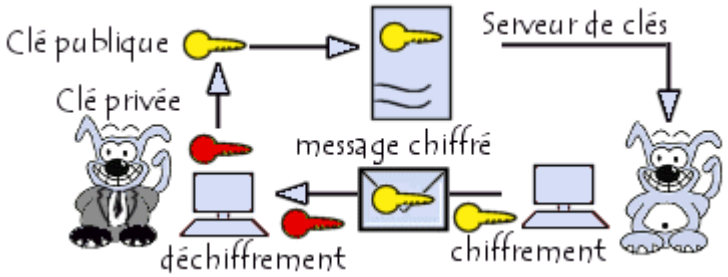
Algorithmes à clef publique

Le Chiffrement Asymétrique

Ce système est aussi appelé **Système à Clef Publique**.

Ce système consiste à avoir deux clefs appelées des **bi-clefs**:

- Une clef **publique** pour le chiffrement
- Une clef **secrète** ou **privée** pour le déchiffrement



- L'utilisateur A (celui qui déchiffre) choisit une clef privée.
- A partir de cette clef il génère plusieurs clefs publiques grâce à un algorithme.
- L'utilisateur B (celui qui chiffre) choisit une des clefs publiques à travers un canal non-sécurisé pour chiffrer les données à l'attention de l'utilisateur A.

Ce système est basé sur ce que l'on appelle une **fonction à trappe à sens unique** ou **one-way trap door**.

Il existe toutefois un problème – s'assurer que la clef publique récupérée est bien celle qui correspond au destinataire !

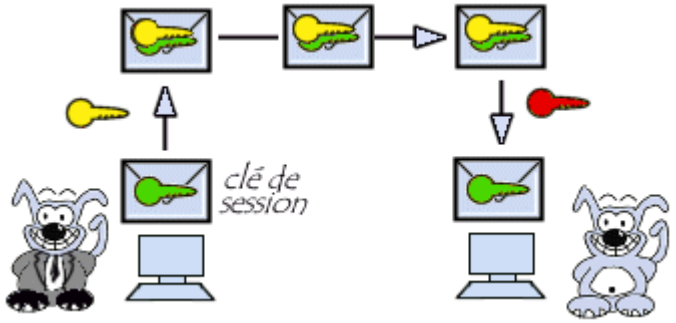
Les algorithmes de chiffrement asymétrique couramment utilisés en informatique sont:

- **Digital Signature Algorithm** (DSA)

- **Rivest, Shamir, Adleman** (RSA)

La Clef de Session

Ce système est un compromis entre le système symétrique et le système asymétrique. Il permet l'envoie de données chiffrées à l'aide d'un algorithme de chiffrement symétrique par un canal non-sécurisé et a été mis au point pour palier au problème de lenteur de déchiffrement du système asymétrique.

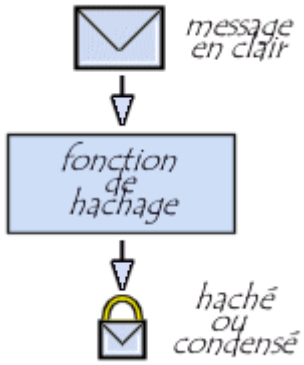


Ce système fonctionne de la façon suivante :

- L'utilisateur A chiffre une clef privée générée aléatoirement, appelée une « clef de session », en utilisant une des clefs publiques de l'utilisateur B.
- L'utilisateur A chiffre les données avec la clef de session.
- L'utilisateur B déchiffre la clef de session en utilisant sa propre clef privée.
- L'utilisateur B déchiffre les données en utilisant la clef de session.

Fonctions de Hachage

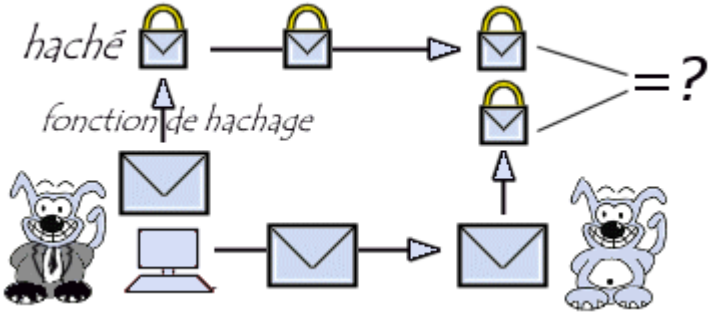
La fonction de **hachage**, aussi appelée une fonction de **condensation**, est à **sens unique** (one way function). Il « condense » un message en clair et produit un haché unique.



Les deux algorithmes de hachage utilisés sont:

- **Message Digest 5** (MD5)
- **Secure Hash Algorithm** (SHA)

Lors de son envoie, le message est accompagné de son haché et il est donc possible de garantir son intégrité:

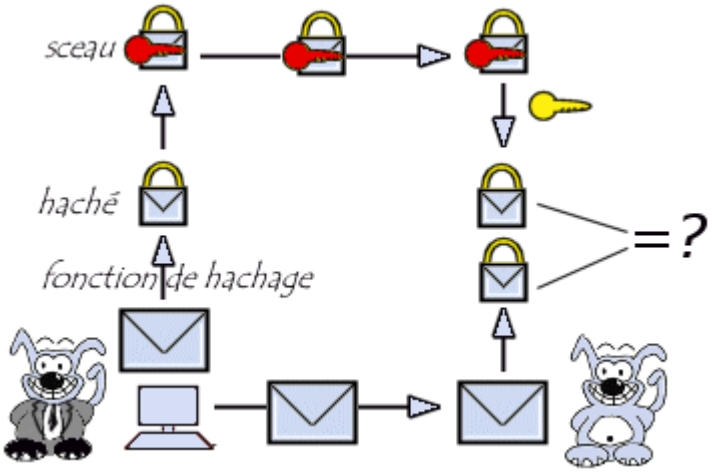


- A la réception du message, le destinataire ou l'utilisateur B calcule le haché du message reçu et le compare avec le haché accompagnant le document.
- Si le message ou le haché a été falsifié durant la communication, les deux empreintes ne correspondront pas.

Important - Ce système permet de vérifier que l'empreinte correspond bien au message reçu, mais ne permet pas de prouver que le message a bien été envoyé par l'utilisateur A.

Signature Numérique

Pour garantir l'authentification du message l'utilisateur A va chiffrer ou **signer** le haché à l'aide de sa clé privée. Le haché signé est appelé un **sceau**.



- L'utilisateur A envoie le sceau au destinataire.
- A la réception du message L'utilisateur B déchiffre le sceau avec la clé publique de l'utilisateur A.
- Il compare le haché obtenu au haché reçu en pièce jointe.

Ce mécanisme de création de sceau est appelé **scellement**.

Ce mécanisme est identique au procédé utilisé par SSH lors d'une connexion

Utilisation de GnuPG

Présentation

GNU Privacy Guard permet aux utilisateurs de transférer des messages chiffrés et/ou signés.

Installation

Sous RHEL 9, le paquet gnupg est installé par défaut :

```
[root@redhat9 ~]# whereis gpg
gpg: /usr/bin/gpg /usr/share/man/man1/gpg.1.gz
```

Configuration

Pour initialiser GnuPG, saisissez la commande suivante :

```
[root@redhat9 ~]# gpg
gpg: directory '/root/.gnupg' created
gpg: keybox '/root/.gnupg/pubring.kbx' created
gpg: WARNING: no command supplied.  Trying to guess what you mean ...
gpg: Go ahead and type your message ...
^C
gpg: signal Interrupt caught ... exiting
```

Pour générer les clefs, saisissez la commande suivante :

```
[root@redhat9 ~]# gpg --full-generate-key
gpg (GnuPG) 2.2.20; Copyright (C) 2020 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
  (14) Existing key from card
```

```
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048)
Requested keysize is 2048 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0)
Key does not expire at all
Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.

Real name: I2TCH
Email address: infos@i2tch.co.uk
Comment: Test Key
You selected this USER-ID:
    "I2TCH (Test Key) <infos@i2tch.co.uk>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: /root/.gnupg/trustdb.gpg: trustdb created
gpg: key 8B4DEC5CC2B2AC5A marked as ultimately trusted
gpg: directory '/root/.gnupg/openpgp-revocs.d' created
```

```
gpg: revocation certificate stored as '/root/.gnupg/openpgp-
revocs.d/9666229B8B43D80C1832BE0D8B4DEC5CC2B2AC5A.rev'
public and secret key created and signed.

pub   rsa2048 2021-08-24 [SC]
      9666229B8B43D80C1832BE0D8B4DEC5CC2B2AC5A
uid                      I2TCH (Test Key) <infos@i2tch.co.uk>
sub   rsa2048 2021-08-24 [E]
```

Important - Lorsque le système vous la demande, entrez la passphrase **fenestros**.

La liste de clefs peut être visualisée avec la commande suivante :

```
[root@redhat9 ~]# gpg --list-keys
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
/root/.gnupg/pubring.kbx
------------------------
pub   rsa2048 2021-08-24 [SC]
      9666229B8B43D80C1832BE0D8B4DEC5CC2B2AC5A
uid           [ultimate] I2TCH (Test Key) <infos@i2tch.co.uk>
sub   rsa2048 2021-08-24 [E]
```

Important - Pour importer la clef d'un correspondant dans sa trousse de clefs il convient d'utiliser la commande suivante :

```
# gpg --import la-clef.asc
```

Pour exporter sa clef publique, il convient d'utiliser la commande suivante :

```
[root@redhat9 ~]# gpg --export --armor I2TCH > ~/I2TCH.asc
[root@redhat9 ~]# cat I2TCH.asc
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQENBGElDSgBCACih8Jfs1nlSPiK/wGCygz2WSljsiXdXlnSHaklznxNldpY4Xrj
TPl145L95XJkHsMf++74MVMdGBn1TnG6m+J1iXkV2EbZzxw9rExA5u9W6rtzWIzP
a/90kuQNAfc/sCUoAM10MqOvpiuc+vSHoJNuqdh4Vv1K3wSg+yQKBXacStZ/7ZS3
0PFXXFCjP6IW4a7h761EcyCXPWhuDfc7qXqLiRjNJS9xKWj0/Hd/O+UYi20XgGB8
VnjMoHodvNvmmsLCvBM8bsxUxT4izFKRHk4xM2AaQurmiU9i1J8nOC51a2IinOtD
QT1WCryY1pnnNzO14BY8VjN2eFWIFh9R9UZhABEBAAG0JEkyVENIIChUZXN0IEtl
eSkgPGluZm9zQGkydGNoLmNvLnVrPokBTgQTAQgAOBYhBJZmIpuLQ9gMGDK+DYtN
7FzCsqxaBQJhJQ0oAhsDBQsJCAcCBhUKCQgLAgQWAgMBAh4BAheAAAoJEItN7FzC
sqxaFAkH/1ZQrtW6oNsATiG0i+X6obmWfMcRaKZiGcT5TNYdjEvXzDM/ND43nVzy
wBHJR6jZ45M4e+OeQAe01VrqBJGirrgZDOg0m8gXdXr0mygAFmUwQ6E+qYlawx7j
29p2al54zpaaRSy2r/y5+hD0KV/OQxzb9xUSm0qhqMFryh+hBBvJXqNVdBH0lk+j
ENK/8BvD5FtjgU6r3pvICWiA+hwSQ2bCT+l2O83twP5o19oRE3dTd+pX5/RI5Kgj
+YuD6jtVzCnA2hbjCJ4xVErEBubg/1f9D4IgnZp5QTaznpH6US2rZ1Xhz2P6Jo95
61kuoR4K4H7zvdyEOgbtZf3iDfrAc/i5AQ0EYSUNKAEIALidAGF/Ev18YfokQy5z
Xssxj2UuKRYwRO6xr731aBaYKgOym0/56Aj944WhWmJ0/RyIMpRz51p/yFLtHy1H
nWg0a3WnwGssQbL4UErEe1wUrNb3hLsvFXYDehZTWcr2adfl94Yv4yaOa9vYmb5p
Qu5tAoDQ1PUqZYsR83IjIQinF2ZgQh6+cK+MfojtwwarmwhHJnYAhbOux3WB0FVy
h6SbGxA4Sps/ANqpgR/TPFlXzXI1vVFN9x9QMhMNGjyO1oIs8dcYLYoixb970shx
9IucE6Yw7SBfVlJ5ezI+Q+CNEzCJgJ/kUXNST/QWdq/h7lSE2CNnhrcYAoOdEAaB
pNUAEQEAAYkBNgQYAQgAIBYhBJZmIpuLQ9gMGDK+DYtN7FzCsqxaBQJhJQ0oAhsM
AAoJEItN7FzCsqxadFgH/R3ncPLtfjlRE0bZM6MUbutnQxq4RbBp9JrbqYhFy97o
lWbhMrca8Ts9pCZE3/kFbsNhg3uoe7rbECYMvmCJ2Gi8RtM45SAyzezYyR45fa2W
825P+DaUdZ4ahX1jzaNEWgzMjKRt2P84ih1St7oW9OcOT/04kCYhmsGfLZPch9+R
W+S8kIoiBJ8ucL5KNy9TAOTTvk4fC7w9plovpU9fJRs7CMg0kKEnTrgkH06bVK65
+4aNWr0LPPNzJaalBMLAghbzcMzRVwsB79AuKciUP/6ZTjyEGXtH/cF5Xxup5qHT
WEhhheTEBxVhlpK40Gs0B6TMSkBGq8LjQ98V3hghYa4=
=0TAN
-----END PGP PUBLIC KEY BLOCK-----
```

Cette clef peut ensuite être jointe à des messages électroniques ou bien être déposée sur un serveur de clefs tel que http://www.keyserver.net.

Signer un message

Créez maintenant un message à signer :

```
[root@redhat9 ~]# vi ~/message.txt
[root@redhat9 ~]# cat ~/message.txt
This is a test message for gpg
```

Pour signer ce message en format binaire, il convient d'utiliser la commande suivante :

```
[root@redhat9 ~]# gpg --default-key I2TCH --detach-sign message.txt
gpg: using "I2TCH" as default secret key for signing
[root@redhat9 ~]# ls -l | grep message
-rw-r--r--. 1 root root   31 Aug 24 11:22 message.txt
-rw-r--r--. 1 root root  329 Aug 24 11:23 message.txt.sig
[root@redhat9 ~]# cat message.txt.sig
0!f"C
M\²Za%infos@i2tch.co.uk
        M\²ZT2oh@<E=n)\jED$kFvʌ`@ʔL/4XYO?49U*cje?sh
-p&Ƶa2i?qUuQ慫                                  ¿<![l
9ʌB|RA?Rk#b2V65mt"vC,:n
/H4&                    krZ
a+ 6%6O%<z+(qsv[root@redhat9 ~]#
```

Pour signer ce message en format ascii, il convient d'utiliser la commande suivante :

```
[root@redhat9 ~]# gpg --default-key I2TCH --armor --detach-sign message.txt
gpg: using "I2TCH" as default secret key for signing
[root@redhat9 ~]# ls -l | grep message
-rw-r--r--. 1 root root   31 Aug 24 11:22 message.txt
```

```
-rw-r--r--. 1 root root  512 Aug 24 11:24 message.txt.asc
-rw-r--r--. 1 root root  329 Aug 24 11:23 message.txt.sig
[root@redhat9 ~]# cat message.txt.asc
-----BEGIN PGP SIGNATURE-----

iQFGBAABCAAwFiEElmYim4tD2AwYMr4Ni03sXMKyrFoFAmElDywSHGluZm9zQGky
dGNoLmNvLnVrAAoJEItN7FzCsqxac1YIAIohAPQ8x2G60HW8yhJKIJxCLrM+gvKz
GsTB/l+vPDEP6fToBnvMkvQwJqqQ7C0m7WkE4M2VWte6RxcpnUVcdwSlkpTKT4ww
Dbwlt7kgwX0MNPr4qOQfAG8azJB40UCRd9aq3nwstdZWmLiQ48zraR/h50WOFN/H
0muyB4khwk2lonE/z7T09BNb8kMajK0CC+ZTSb2eOHb4U2C1jfzUybfR2v2+ApmC
Dmj4vu2jM5YnElP5Kbz4me/JY5zZbYIFhTb8TMq7kVIuibaB4keERVdd+fk0FY1Z
WFggEvw1tSuoC3rZ0y1c0Rj59HoZ9QxaKX8n+wq5+A4k8slt6WzuAu8=
=//z2
-----END PGP SIGNATURE-----
```

Pour vérifier la signature d'un message signé en mode ascii, il convient d'utiliser la commande :

```
[root@redhat9 ~]# gpg --verify message.txt.asc
gpg: assuming signed data in 'message.txt'
gpg: Signature made Tue 24 Aug 2021 11:24:28 EDT
gpg:                using RSA key 9666229B8B43D80C1832BE0D8B4DEC5CC2B2AC5A
gpg:                issuer "infos@i2tch.co.uk"
gpg: Good signature from "I2TCH (Test Key) <infos@i2tch.co.uk>" [ultimate]
```

Pour vérifier la signature d'un message signé en mode ascii et produit en dehors du message lui-même, il convient d'utiliser la commande :

```
[root@redhat9 ~]# gpg --verify message.txt.asc message.txt
gpg: Signature made Tue 24 Aug 2021 11:24:28 EDT
gpg:                using RSA key 9666229B8B43D80C1832BE0D8B4DEC5CC2B2AC5A
gpg:                issuer "infos@i2tch.co.uk"
gpg: Good signature from "I2TCH (Test Key) <infos@i2tch.co.uk>" [ultimate]
```

Pour signer ce message **dans le message lui-même** en format ascii, il convient d'utiliser la commande suivante :

```
[root@redhat9 ~]# gpg --default-key I2TCH --clearsign message.txt
gpg: using "I2TCH" as default secret key for signing
File 'message.txt.asc' exists. Overwrite? (y/N) y
[root@redhat9 ~]# ls -l | grep message
-rw-r--r--. 1 root root   31 Aug 24 11:22 message.txt
-rw-r--r--. 1 root root  592 Aug 24 11:28 message.txt.asc
-rw-r--r--. 1 root root  329 Aug 24 11:23 message.txt.sig
[root@redhat9 ~]# cat message.txt.asc
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA256

This is a test message for gpg
-----BEGIN PGP SIGNATURE-----

iQFGBAEBCAAwFiEElmYim4tD2AwYMr4Ni03sXMKyrFoFAmElEBMSHGluZm9zQGky
dGNoLmNvLnVrAAoJEItN7FzCsqxaQa0H+gLxI8PTEJtbg6q+PmhlsQq2PkITRDFB
bC5vW8CQzXUNA08aqkBEOgA1OvX9gJG0Q/aJO7fPrQFWP9g7IYPax/GvmgHCmS7B
Hc5uULOawGvulctflk7xCmhgtaFndwCUN685xCPDOdhUMs0rX9Zqj8pKhbwh4Xpz
Q7vY5gPJTn2aj4PL5GkXN/ZzGclFTVN9o5BQuxYnTCB694WzZepf48dMPaIdlDxJ
l2yHf/jZGt2ZE2hoVllvjMN81LhjaqMxIoSTLwUAn+WBtrwNreQdERxtQv0waIA7
NNFzGPdi0HGdJhjYJ/v4eFbi5X4gvHVVazzOpY5p48yVgCRAwZHJh/0=
=C3OQ
-----END PGP SIGNATURE-----
```

Chiffrer un message

Pour chiffrer un message, il faut disposer de la clef publique du destinataire du message. Ce dernier utilisera ensuite sa clef privée pour déchiffrer le message. Il convient de préciser le destinataire du message, ou plus précisément la clef publique à utiliser, lors d'un chiffrement :

```
gpg --recipient <destinataire> --encrypt <message>
```

- <*destinataire*> représente toute information permettant de distinguer sans ambigüité une clef publique dans votre trousseau. Cette information peut-être le nom ou l'adresse email associé à la clef publique que vous voulez utiliser,

- <*message*> représente le message à chiffrer.

Par exemple pour chiffrer un message en mode binaire, il convient de saisir la commande suivante :

```
[root@redhat9 ~]# gpg --recipient I2TCH --encrypt message.txt
[root@redhat9 ~]# ls -l | grep message
-rw-r--r--. 1 root root   31 Aug 24 11:22 message.txt
-rw-r--r--. 1 root root  592 Aug 24 11:28 message.txt.asc
-rw-r--r--. 1 root root  367 Aug 24 11:30 message.txt.gpg
-rw-r--r--. 1 root root  329 Aug 24 11:23 message.txt.sig
[root@redhat9 ~]# cat message.txt.gpg

EeJ ц

     pqa=w_wZI)0,G@"s"+i:(AVG;@GX)[□ba9hh%7
                                          Wg7X
                                              o#U>g□HEre8K\R*4u0n@"{SIlgt6gy]Z{t0'p@k{%I~}puO-
#făt^S)[Ŝ)Xq=#94t;fMнгC|UVoз,H|+.!4:DmZlO]bI{H[root@redhat9 ~]#
```

Et pour chiffrer un message en mode ascii, il convient de saisir la commande suivante :

```
[root@redhat9 ~]# gpg --recipient I2TCH --armor --encrypt message.txt
File 'message.txt.asc' exists. Overwrite? (y/N) y
[root@redhat9 ~]# ls -l | grep message
-rw-r--r--. 1 root root   31 Aug 24 11:22 message.txt
-rw-r--r--. 1 root root  561 Aug 24 11:32 message.txt.asc
-rw-r--r--. 1 root root  367 Aug 24 11:30 message.txt.gpg
-rw-r--r--. 1 root root  329 Aug 24 11:23 message.txt.sig
[root@redhat9 ~]# cat message.txt.asc
-----BEGIN PGP MESSAGE-----

hQEMA0XsZUog1b4LAQf7BgGL8LMcMbLdD4nSOwc45FLNyj9MXkr0ru01jBRb3UP/
MW6VxWekLrWOXRBvFo/dS1Y/KIAYiZ9kDVSYwbbrQxOql/F4sWBagWAOs/gzeWt6
MrKuOK6pgPdgO57AcImOeUjPL42RHh6enGRdud+GWiZNQKAvPiCNikfhJUza+o1Z
```

```
GyAcq5RMSuohOp2weai5CwcVqZddrTvKzjkoUrMCwnMxGKjdbNRC3+DKEI9B4L3j
7Dno9DseQcebD3NYEICSt2oJr+xazejiLj4X8nerBrCqV7nK9v7mvxTKCIL5iOBR
duBPFvgJuSVnSJZ+XzBeEQ8q24L3FLV9B5yJnF+e8tJeASweIXfqWaeWNObfAHC3
dkMtvNUNs6jkmFUGdONYosNlHW9jFWllpe2Q5Ra13kdZob3o1eevU2iGBAx0Gi0Z
yEB3HjqYFKxFj+lCj4KP59O55sEpePgAo2qhPhfeMw==
=UDxQ
-----END PGP MESSAGE-----
```

Pour décrypter un message il convient d'utiliser la commande suivante :

```
[root@redhat9 ~]# gpg --decrypt message.txt.asc
gpg: encrypted with 2048-bit RSA key, ID 45EC654A20D5BE0B, created 2021-08-24
      "I2TCH (Test Key) <infos@i2tch.co.uk>"
This is a test message for gpg
```

PKI

On appelle **PKI** (Public Key Infrastucture, ou en français **infrastructure à clé publique (ICP)**, parfois **infrastructure de gestion de clés (IGC)**) l'ensemble des solutions techniques basées sur la cryptographie à clé publique.

Les cryptosystèmes à clés publiques permettent de s'affranchir de la nécessité d'avoir recours systématiquement à un canal sécurisé pour s'échanger les clés. En revanche, la publication de la clé publique à grande échelle doit se faire en toute confiance pour assurer que :

- La clé publique est bien celle de son propriétaire ;
- Le propriétaire de la clé est digne de confiance ;
- La clé est toujours valide.

Ainsi, il est nécessaire d'associer au bi-clé (ensemble clé publique / clé privée) un certificat délivré par un **tiers de confiance** : l'infrastructure de gestion de clés.

Le tiers de confiance est une entité appelée communément autorité de certification (ou en anglais Certification authority, abrégé CA) chargée d'assurer la véracité des informations contenues dans le certificat de clé publique et de sa validité.

Pour ce faire, l'autorité signe le certificat de clé publique à l'aide de sa propre clé en utilisant le principe de signature numérique.

Le rôle de l'infrastructure de clés publiques est multiple et couvre notamment les champs suivants :

- enregistrer des demandes de clés en vérifiant l'identité des demandeurs ;
- générer les paires de clés (clé privée / clé publique) ;
- garantir la confidentialité des clés privées correspondant aux clés publiques ;
- certifier l'association entre chaque utilisateurs et sa clé publique ;
- révoquer des clés (en cas de perte par son propriétaire, d'expiration de sa date de validité ou de compromission).

Une infrastructure à clé publique est en règle générale composée de trois entités distinctes :

- L'autorité d'enregistrement (AE ou RA pour Recording authority), chargée des formalité administratives telles que la vérification de l'identité des demandeurs, le suivi et la gestion des demandes, etc.) ;
- L'autorité de certification (AC ou CA pour Certification Authority), chargée des tâches techniques de création de certificats. L'autorité de certification est ainsi chargée de la signature des demandes de certificat (CSR pour Certificate Signing Request, parfois appelées PKCS#10, nom du format correspondant). L'autorité de certification a également pour mission la signature des listes de révocations (CRL pour Certificate Revocation List) ;
- L'Autorité de dépôt (Repository) dont la mission est de conserver en sécurité les certificats.

Certificats X509

Pour palier aux problèmes liés à des clefs publiques piratées, un système de certificats a été mis en place.

Le certificat permet d’associer la clef publique à une entité ou une personne. Les certificats sont délivrés par des Organismes de Certification.

Les certificats sont des fichiers divisés en deux parties :

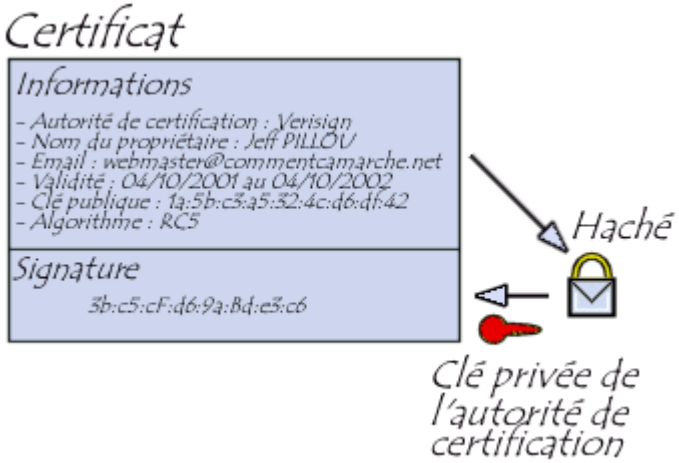
- La partie contenant les informations
- La partie contenant la signature de l'autorité de certification

La structure des certificats est normalisée par le standard **X.509** de l’**Union internationale des télécommunications**.

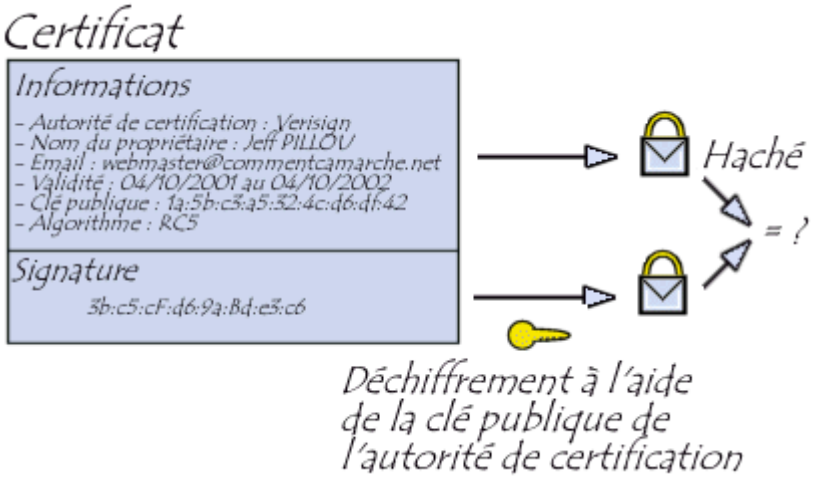
Elle contient :

- Le nom de l'autorité de certification
- Le nom du propriétaire du certificat
- La date de validité du certificat
- L'algorithme de chiffrement utilisé
- La clé publique du propriétaire

Le Certificat est signé par l'autorité de certification:



La vérification se passe ainsi:



Comprendre IPv4

En-tête TCP

L'en-tête TCP est codée sur 4 octets soit 32 bits :

1er octet	2ème octet	3ème octet	4 ème octet
Port source		Port destination	
Numéro de séquence			
Numéro d'acquittement			
Offset	Flags	Fenêtre	
Checksum		Pointeur Urgent	
Options			Padding
Données			

Vous noterez que les numéros de ports sont codés sur 16 bits. Cette information nous permet de calculer le nombres de ports maximum en IPv4, soit 2^{16} ports ou 65 535.

L'**Offset** contient la taille de l'en-tête.

Les **Flags** sont :

- URG - Si la valeur est 1 le pointeur urgent est utilisé. Le numéro de séquence et le pointeur urgent indique un octet spécifique.
- ACK - Si la valeur est 1, le paquet est un accusé de réception
- PSH - Si la valeur est 1, les données sont immédiatement présentées à l'application
- RST - Si la valeur est 1, la communication comporte un problème et la connexion est réinitialisée
- SYN - Si la valeur est 1, le paquet est un paquet de synchronisation
- FIN - Si la valeur est 1, le paquet indique la fin de la connexion

La **Fenêtre** est codée sur 16 bits. La Fenêtre est une donnée liée au fonctionnement d'expédition de données appelé le **sliding window** ou la **fenêtre glissante**. Puisque il serait impossible, pour des raisons de performance, d'attendre l'accusé de réception de chaque paquet envoyé, l'expéditeur envoie des paquets par groupe. La taille de cette groupe s'appelle la Fenêtre. Dans le cas d'un problème de réception d'une partie de la Fenêtre, toute la Fenêtre est ré-expédiée.

Le **Checksum** est une façon de calculer si le paquet est complet.

Le **Padding** est un champ pouvant être rempli de valeurs nulles de façon à ce que la taille de l'en-tête soit un multiple de 32

En-tête UDP

L'en-tête UDP est codée sur 4 octets soit 32 bits :

1er octet	**2ème octet**	**3ème octet**	**4 ème octet**
Port source		Port destination	
longueur		Checksum	
Données			

L'en-tête UDP a une longueur de 8 octets.

Fragmentation et Ré-encapsulation

La taille limite d'un paquet TCP, l'en-tête comprise, ne peut pas dépasser **65 535 octets**. Cependant chaque réseau est qualifié par son MTU (Maximum Tranfer Unit). Cette valeur est la taille maximum d'un paquet autorisée. L'unité est en **octets**. Pour un réseau Ethernet sa valeur est de 1 500. Quand un paquet doit être expédié sur un réseau ayant un MTU inférieur à sa propre taille, le paquet doit être **fractionné**. A la sortie du réseau, le paquet est reconstitué. Cette reconstitution s'appelle **ré-encapsulation**.

Adressage

L'adressage IP requière que chaque périphérique sur le réseau possède une adresse IP unique de 4 octets, soit 32 bits au format XXX.XXX.XXX.XXX De cette façon le nombre total d'adresses est de 2^{32} = 4.3 Milliards.

Les adresses IP sont divisées en 5 classes, de A à E. Les 4 octets des classes A à C sont divisés en deux, une partie qui s'appelle le **Net ID** qui identifie le réseau et une partie qui s'appelle le **Host ID** qui identifie le hôte :

	1er octet	2ème octet	3ème octet	4 ème octet
A	Net ID	Host ID		
B	Net ID		Host ID	
C	Net ID			Host ID
D	Multicast			
E	Réservé			

L'attribution d'une classe dépend du nombre de hôtes à connecter. Chaque classe est identifié par un **Class ID** composé de 1 à 3 bits :

Classe	Bits ID Classe	Valeur ID Classe	Bits ID Réseau	Nb. de Réseaux	Bits ID hôtes	Nb. d'adresses	Octet de Départ
A	1	0	7	2^7=128	24	2^{24}=16 777 216	1 - 126
B	2	10	14	2^{14}=16 834	16	2^{16}=65 535	128 - 191
C	3	110	21	2^{21}=2 097 152	8	2^8=256	192 - 223

Dans chaque classe, certaines adresses sont réservées pour un usage privé :

Classe	IP de Départ	IP de Fin
A	10.0.0.0	10.255.255.255
B	172.16.0.0	172.31.255.255
C	192.168.0.0	192.168.255.255

Il existe des adresses particulières ne pouvant pas être utilisées pour identifier un hôte :

Adresse Particulière	Description
169.254.0.0 à 169.254.255.255	Automatic Private IP Addressing de Microsoft
Hôte du réseau courant	Tous les bits du Net ID sont à 0
Adresse de réseau	Tous les bits du Host ID sont à 0
Adresse de diffusion	Tous les bits du Host ID sont à 1

L'adresse de réseau identifie le **segment** du réseau entier tandis que l'adresse de diffusion identifie tous les hôtes sur le segment de réseau.

Afin de mieux comprendre l'adresse de réseau et l'adresse de diffusion, prenons le cas de l'adresse 192.168.10.1 en classe C :

	1er octet	2ème octet	3ème octet	4 ème octet
	Net ID			Host ID
Adresse IP	192	168	10	1
Binaire	11000000	10101000	000001010	00000001
Calcul de l'adresse de réseau				
Binaire	11000000	10101000	000001010	**00000000**
Adresse réseau	192	168	10	0
Calcul de l'adresse de diffusion				
Binaire	11000000	10101000	000001010	**11111111**
Adresse de diffusion	192	168	10	255

Masques de sous-réseaux

Tout comme l'adresse IP, le masque de sous-réseau compte 4 octets ou 32 bits. Les masques de sous-réseaux permettent d'identifer le Net ID et le Host ID :

Classe	Masque	Notation CIDR
A	255.0.0.0	/8
B	255.255.0.0	/16
C	255.255.255.0	/24

Le terme **CIDR** veut dire **Classless InterDomain Routing**. Le terme Notation CIDR correspond au nombre de bits d'une valeur de 1 dans le masque de sous-réseau.

Quand un hôte souhaite émettre il procède d'abord à l'identification de sa propre adresse réseau par un calcul AND (ET) appliqué à sa propre adresse et son masque de sous-réseau qui stipule :

- 1 x 1 = 1
- 0 x 1 = 0
- 1 x 0 = 0
- 0 x 0 = 0

Prenons le cas de l'adresse IP 192.168.10.1 ayant un masque de 255.255.255.0 :

	1er octet	2ème octet	3ème octet	4 ème octet
Adresse IP	192	168	10	1
Binaire	11000000	10101000	00001010	00000001
Masque de sous-réseau				
Binaire	11111111	11111111	11111111	00000000
Calcul AND	11000000	10101000	00001010	00000000
Adresse réseau	192	168	10	0

Cet hôte essaie de communiquer avec un hôte ayant une adresse IP de 192.168.10.10. Il procède donc au même calcul en appliquant **son propre masque de sous-réseau** à l'adresse IP de l'hôte destinataire :

	1er octet	2ème octet	3ème octet	4 ème octet
Adresse IP	192	168	10	10
Binaire	11000000	10101000	00001010	00001010
Masque de sous-réseau				

	1er octet	2ème octet	3ème octet	4 ème octet
Binaire	11111111	11111111	11111111	00000000
Calcul AND	11000000	10101000	00001010	00000000
Adresse réseau	192	168	10	0

Puisque l'adresse réseau est identique dans les deux cas, l'hôte émetteur présume que l'hôte de destination se trouve sur son réseau et envoie les paquets directement sur le réseau sans s'adresser à sa passerelle par défaut.

L'hôte émetteur essaie maintenant de communiquer avec avec un hôte ayant une adresse IP de 192.168.2.1. Il procède donc au même calcul en appliquant **son propre masque de sous-réseau** à l'adresse IP de l'hôte destinataire :

	1er octet	2ème octet	3ème octet	4 ème octet
Adresse IP	192	168	2	1
Binaire	11000000	10101000	00000010	00000001
Masque de sous-réseau				
Binaire	11111111	11111111	11111111	00000000
Calcul AND	11000000	10101000	00000010	00000000
Adresse réseau	192	168	2	0

Dans ce cas, l'hôte émetteur constate que le réseau de destination 192.168.2.0 n'est pas identique à son propre réseau 192.168.10.0. Il adresse donc les paquets à la passerelle par défaut.

VLSM

Puisque le stock de réseaux disponibles sous IPv4 est presque épuisé, une solution a du être trouvée pour créer des sous-réseaux en attendant l'introduction de l'IPv6. Cette solution s'appelle le VLSM ou Variable Length Subnet Masks. Le VLSM exprime les masques de sous-réseaux au format CIDR.

Son principe est simple. Afin de créer des réseaux différents à partir d'une adresse réseau d'une classe donnée, il convient de réduire le nombre d'hôtes. De cette façon les bits 'libérés' du Host ID peuvent être utilisés pour identifier les sous-réseaux.

Pour illustrer ceci, prenons l'exemple d'un réseau 192.168.1.0. Sur ce réseau, nous pouvons mettre 2^8-2 soit 254 hôtes entre 192.168.1.1 au

192.168.1.254.

Supposons que nous souhaiterions diviser notre réseau en 2 sous-réseaux. Pour coder 2 sous-réseaux, il faut que l'on libère 2 bits du Host ID. Les deux bits libérés auront les valeurs binaires suivantes :

- 00
- 01
- 10
- 11

Les valeurs binaires du quatrième octet de nos adresses de sous-réseaux seront donc :

- 192.168.1.00XXXXXX
- 192.168.1.01XXXXXX
- 192.168.1.10XXXXXX
- 192.168.1.11XXXXXX

où les XXXXXX représentent les bits que nous réservons pour décrire les hôtes dans chacun des sous-réseaux.

Nous ne pouvons pas utiliser les deux sous-réseaux suivants :

- 192.168.1.00XXXXXX
- 192.168.1.11XXXXXX

car ceux-ci correspondent aux débuts de l'adresse réseau 192.168.1.0 et de l'adresse de diffusion 192.168.1.255.

Nous pouvons utiliser les deux sous-réseaux suivants :

- 192.168.1.01XXXXXX
- 192.168.1.10XXXXXX

Pour le premier sous-réseau l'adresse réseau et l'adresse de diffusion sont :

Sous-réseau #1	192	168	1	01XXXXXX
Calcul de l'adresse de réseau				
Binaire	11000000	10101000	00000001	01**000000**

Adresse réseau	192	168	1	**64**
Calcul de l'adresse de diffusion				
Binaire	11000000	10101000	00000001	01**111111**
Adresse de diffusion	192	168	1	**127**

- L'adresse CIDR du réseau est donc 192.168.1.64/26 car le Net ID est codé sur 24+2 bits.
- Le masque de sous-réseau est donc le 11111111.11111111.11111111.11000000 ou le 255.255.255.192
- Nous pouvons avoir 2^6-2 soit 62 hôtes.
- La plage valide d'adresses IP est de 192.168.1.65 à 192.168.1.126

Pour le deuxième sous-réseau l'adresse réseau et l'adresse de diffusion sont :

Sous-réseau #2	192	168	1	10XXXXXX
Calcul de l'adresse de réseau				
Binaire	11000000	10101000	00000001	10**000000**
Adresse réseau	192	168	1	**128**
Calcul de l'adresse de diffusion				
Binaire	11000000	10101000	00000001	10**111111**
Adresse de diffusion	192	168	1	**191**

- L'adresse CIDR du réseau est donc 192.168.1.128/26 car le Net ID est codé sur 24+2 bits.
- Le masque de sous-réseau est donc le 11111111.11111111.11111111.11000000 ou le 255.255.255.192
- Nous pouvons avoir 2^6-2 soit 62 hôtes.
- La plage valide d'adresses IP est de 192.168.1.129 à 192.168.1.190

La valeur qui sépare les sous-réseaux est 64. Cette valeur comporte le nom **incrément**.

Ports et sockets

Afin que les données arrivent aux applications que les attendent, TCP utilise des numéros de ports sur la couche transport. Le numéros de ports sont divisés en trois groupes :

- **Well Known Ports**
  - De 1 à 1023
- **Registered Ports**
  - De 1024 à 49151
- **Dynamic** et/ou **Private Ports**
  - De 49152 à 65535

Le couple **numéro IP:numéro de port** s'appelle un **socket**.

/etc/services

Les ports les plus utilisés sont détaillés dans le fichier **/etc/services** :

```
[root@redhat9 ~]# more /etc/services
# /etc/services:
# $Id: services,v 1.49 2017/08/18 12:43:23 ovasik Exp $
#
# Network services, Internet style
# IANA services version: last updated 2016-07-08
#
# Note that it is presently the policy of IANA to assign a single well-known
# port number for both TCP and UDP; hence, most entries here have two entries
# even if the protocol doesn't support UDP operations.
# Updated from RFC 1700, ``Assigned Numbers'' (October 1994).  Not all ports
# are included, only the more common ones.
#
# The latest IANA port assignments can be gotten from
#       http://www.iana.org/assignments/port-numbers
# The Well Known Ports are those from 0 through 1023.
# The Registered Ports are those from 1024 through 49151
```

```
# The Dynamic and/or Private Ports are those from 49152 through 65535
#
# Each line describes one service, and is of the form:
#
# service-name  port/protocol  [aliases ...]   [# comment]

tcpmux          1/tcp                           # TCP port service multiplexer
tcpmux          1/udp                           # TCP port service multiplexer
rje             5/tcp                           # Remote Job Entry
rje             5/udp                           # Remote Job Entry
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
systat          11/udp          users
daytime         13/tcp
daytime         13/udp
qotd            17/tcp          quote
qotd            17/udp          quote
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp-data        20/udp
# 21 is registered to ftp, but also used by fsp
ftp             21/tcp
ftp             21/udp          fsp fspd
ssh             22/tcp                          # The Secure Shell (SSH) Protocol
ssh             22/udp                          # The Secure Shell (SSH) Protocol
telnet          23/tcp
telnet          23/udp
# 24 - private mail system
lmtp            24/tcp                          # LMTP Mail Delivery
lmtp            24/udp                          # LMTP Mail Delivery
```

```
smtp            25/tcp          mail
smtp            25/udp          mail
time            37/tcp          timserver
time            37/udp          timserver
rlp             39/tcp          resource        # resource location
--More--(0%)
[q]
```

Notez que les ports sont listés par deux :

- le port TCP
- le port UDP

La liste la plus complète peut être consultée à l'adresse suivante **https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml**.

Pour connaitre la liste des sockets ouverts sur l'ordinateur, saisissez la commande suivante :

```
[root@redhat9 ~]# netstat -an | more
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:15023         0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.102:22           10.0.2.1:59570          ESTABLISHED
tcp6       0      0 ::1:15023               :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::23                   :::*                    LISTEN
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 ::1:631                 :::*                    LISTEN
tcp6       0      0 ::1:55794               ::1:22                  ESTABLISHED
tcp6       0      0 ::1:22                  ::1:55794               ESTABLISHED
udp        0      0 127.0.0.1:323           0.0.0.0:*
udp        0      0 0.0.0.0:42140           0.0.0.0:*
```

```
udp        0      0 0.0.0.0:5353            0.0.0.0:*
udp6       0      0 ::1:323                 :::*
udp6       0      0 :::58479                :::*
udp6       0      0 :::5353                 :::*
raw6       0      0 :::58                   :::*                    7
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ACC ]     STREAM     LISTENING     22852    /var/run/lsm/ipc/sim
unix  2      [ ACC ]     STREAM     LISTENING     22853    /var/run/lsm/ipc/simc
unix  2      [ ACC ]     STREAM     LISTENING     23975    @/tmp/.ICE-unix/1794
unix  2      [ ACC ]     STREAM     LISTENING     23732    @/tmp/dbus-40kNqTSK
unix  2      [ ]         DGRAM                    40557    /run/user/1000/systemd/notify
unix  2      [ ]         DGRAM                    23906    /run/user/42/systemd/notify
unix  2      [ ACC ]     STREAM     LISTENING     40560    /run/user/1000/systemd/private
unix  2      [ ACC ]     STREAM     LISTENING     23909    /run/user/42/systemd/private
unix  2      [ ACC ]     STREAM     LISTENING     40570    /run/user/1000/bus
unix  2      [ ACC ]     STREAM     LISTENING     23930    /run/user/42/bus
unix  2      [ ACC ]     STREAM     LISTENING     40572    /run/user/1000/pulse/native
unix  2      [ ACC ]     STREAM     LISTENING     23932    /run/user/42/pulse/native
unix  2      [ ACC ]     STREAM     LISTENING     40574    /run/user/1000/pipewire-0
unix  2      [ ACC ]     STREAM     LISTENING     23934    /run/user/42/pipewire-0
unix  3      [ ]         DGRAM      CONNECTED     2446     /run/systemd/notify
unix  2      [ ACC ]     STREAM     LISTENING     40576    /run/user/1000/pipewire-0-manager
unix  2      [ ACC ]     STREAM     LISTENING     23936    /run/user/42/pipewire-0-manager
unix  2      [ ACC ]     STREAM     LISTENING     2451     /run/systemd/userdb/io.systemd.DynamicUser
unix  2      [ ACC ]     STREAM     LISTENING     2452     /run/systemd/io.system.ManagedOOM
unix  2      [ ]         DGRAM      CONNECTED     25598    /run/chrony/chronyd.sock
unix  18     [ ]         DGRAM      CONNECTED     2459     /run/systemd/journal/dev-log
unix  9      [ ]         DGRAM      CONNECTED     2461     /run/systemd/journal/socket
unix  2      [ ACC ]     STREAM     LISTENING     22376    /run/user/42/wayland-0
unix  2      [ ACC ]     STREAM     LISTENING     2463     /run/systemd/journal/stdout
unix  2      [ ACC ]     STREAM     LISTENING     23731    @/tmp/dbus-peR2NXOg
unix  2      [ ACC ]     STREAM     LISTENING     22391    /tmp/dbus-ApzcsH4y3k
unix  2      [ ACC ]     STREAM     LISTENING     23976    /tmp/.ICE-unix/1794
```

```
unix  2      [ ACC ]     STREAM     LISTENING     22756    @ISCSID_UIP_ABSTRACT_NAMESPACE
unix  2      [ ACC ]     STREAM     LISTENING     31411    /etc/httpd/run/cgisock.1083
unix  2      [ ACC ]     STREAM     LISTENING     25894    @/var/lib/gdm/.cache/ibus/dbus-hSex1tg6
unix  2      [ ACC ]     STREAM     LISTENING     22373    /tmp/.X11-unix/X1024
unix  2      [ ACC ]     STREAM     LISTENING     22375    /tmp/.X11-unix/X1025
unix  2      [ ACC ]     STREAM     LISTENING     23734    @/tmp/dbus-TkF9Vnen
--More--
[q]
```

Pour connaitre la liste des applications ayant ouvert un port sur l'ordinateur, saisissez la commande suivante :

```
[root@redhat9 ~]# netstat -anp | more
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      5583/sshd: /usr/sbi
tcp        0      0 127.0.0.1:15023         0.0.0.0:*               LISTEN      5603/ssh
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      878/cupsd
tcp        0      0 10.0.2.102:22           10.0.2.1:59570          ESTABLISHED 4306/sshd: trainee
tcp6       0      0 ::1:15023               :::*                    LISTEN      5603/ssh
tcp6       0      0 :::22                   :::*                    LISTEN      5583/sshd: /usr/sbi
tcp6       0      0 :::23                   :::*                    LISTEN      1/systemd
tcp6       0      0 :::80                   :::*                    LISTEN      1083/httpd
tcp6       0      0 ::1:631                 :::*                    LISTEN      878/cupsd
tcp6       0      0 ::1:55794               ::1:22                  ESTABLISHED 5603/ssh
tcp6       0      0 ::1:22                  ::1:55794               ESTABLISHED 5596/sshd: trainee
udp        0      0 127.0.0.1:323           0.0.0.0:*                           2675/chronyd
udp        0      0 0.0.0.0:42140           0.0.0.0:*                           752/avahi-daemon: r
udp        0      0 0.0.0.0:5353            0.0.0.0:*                           752/avahi-daemon: r
udp6       0      0 ::1:323                 :::*                                2675/chronyd
udp6       0      0 :::58479                :::*                                752/avahi-daemon: r
udp6       0      0 :::5353                 :::*                                752/avahi-daemon: r
raw6       0      0 :::58                   :::*                    7           4456/NetworkManager
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   PID/Program name     Path
```

```
unix  2      [ ACC ]     STREAM     LISTENING     22852    755/lsmd             /var/run/lsm/ipc/sim
unix  2      [ ACC ]     STREAM     LISTENING     22853    755/lsmd             /var/run/lsm/ipc/simc
unix  2      [ ACC ]     STREAM     LISTENING     23975    1794/gnome-session-  @/tmp/.ICE-unix/1794
unix  2      [ ACC ]     STREAM     LISTENING     23732    1140/gdm             @/tmp/dbus-40kNqTSK
unix  2      [ ]         DGRAM                    40557    4129/systemd         /run/user/1000/systemd/notify
unix  2      [ ]         DGRAM                    23906    1340/systemd         /run/user/42/systemd/notify
unix  2      [ ACC ]     STREAM     LISTENING     40560    4129/systemd         /run/user/1000/systemd/private
unix  2      [ ACC ]     STREAM     LISTENING     23909    1340/systemd         /run/user/42/systemd/private
unix  2      [ ACC ]     STREAM     LISTENING     40570    4129/systemd         /run/user/1000/bus
unix  2      [ ACC ]     STREAM     LISTENING     23930    1340/systemd         /run/user/42/bus
unix  2      [ ACC ]     STREAM     LISTENING     40572    4129/systemd         /run/user/1000/pulse/native
unix  2      [ ACC ]     STREAM     LISTENING     23932    1340/systemd         /run/user/42/pulse/native
unix  2      [ ACC ]     STREAM     LISTENING     40574    4129/systemd         /run/user/1000/pipewire-0
unix  2      [ ACC ]     STREAM     LISTENING     23934    1340/systemd         /run/user/42/pipewire-0
unix  3      [ ]         DGRAM      CONNECTED     2446     1/systemd            /run/systemd/notify
unix  2      [ ACC ]     STREAM     LISTENING     40576    4129/systemd         /run/user/1000/pipewire-0-manager
unix  2      [ ACC ]     STREAM     LISTENING     23936    1340/systemd         /run/user/42/pipewire-0-manager
unix  2      [ ACC ]     STREAM     LISTENING     2451     1/systemd
/run/systemd/userdb/io.systemd.DynamicUser
unix  2      [ ACC ]     STREAM     LISTENING     2452     1/systemd            /run/systemd/io.system.ManagedOOM
unix  2      [ ]         DGRAM      CONNECTED     25598    2675/chronyd         /run/chrony/chronyd.sock
unix  18     [ ]         DGRAM      CONNECTED     2459     1/systemd            /run/systemd/journal/dev-log
unix  9      [ ]         DGRAM      CONNECTED     2461     1/systemd            /run/systemd/journal/socket
unix  2      [ ACC ]     STREAM     LISTENING     22376    1802/gnome-shell     /run/user/42/wayland-0
unix  2      [ ACC ]     STREAM     LISTENING     2463     1/systemd            /run/systemd/journal/stdout
unix  2      [ ACC ]     STREAM     LISTENING     23731    1140/gdm             @/tmp/dbus-peR2NXOg
unix  2      [ ACC ]     STREAM     LISTENING     22391    1825/dbus-daemon     /tmp/dbus-ApzcsH4y3k
unix  2      [ ACC ]     STREAM     LISTENING     23976    1794/gnome-session-  /tmp/.ICE-unix/1794
unix  2      [ ACC ]     STREAM     LISTENING     22756    1/systemd            @ISCSID_UIP_ABSTRACT_NAMESPACE
unix  2      [ ACC ]     STREAM     LISTENING     31411    3121/httpd           /etc/httpd/run/cgisock.1083
unix  2      [ ACC ]     STREAM     LISTENING     25894    2130/ibus-daemon     @/var/lib/gdm/.cache/ibus/dbus-
hSex1tg6
unix  2      [ ACC ]     STREAM     LISTENING     22373    1831/Xwayland        /tmp/.X11-unix/X1024
unix  2      [ ACC ]     STREAM     LISTENING     22375    1802/gnome-shell     /tmp/.X11-unix/X1025
```

```
unix  2      [ ACC ]     STREAM     LISTENING     23734    1140/gdm            @/tmp/dbus-TkF9Vnen
--More--
[q]
```

La commande **ss** peut aussi être utilisée pour connaître la liste des sockets ouverts :

```
[root@redhat9 ~]# ss -ta
State                  Recv-Q                 Send-Q                                  Local Address:Port
Peer Address:Port                  Process
LISTEN                 0                      128                                           0.0.0.0:ssh
0.0.0.0:*
LISTEN                 0                      128                                       127.0.0.1:15023
0.0.0.0:*
LISTEN                 0                      4096                                      127.0.0.1:ipp
0.0.0.0:*
ESTAB                  0                      0                                      10.0.2.102:ssh
10.0.2.1:59570
LISTEN                 0                      128                                           [::1]:15023
[::]:*
LISTEN                 0                      128                                            [::]:ssh
[::]:*
LISTEN                 0                      4096                                             *:telnet
*:*
LISTEN                 0                      511                                              *:http
*:*
LISTEN                 0                      4096                                          [::1]:ipp
[::]:*
ESTAB                  0                      0                                             [::1]:55794
[::1]:ssh
ESTAB                  0                      0                                             [::1]:ssh
[::1]:55794
```

Résolution d'adresses Ethernet

Chaque protocole peut être encapsulé dans une **trame** Ethernet. Lorsque la trame doit être transportée de l'expéditeur au destinataire, ce premier doit connaitre l'adresse Ethernet du dernier. L'adresse Ethernet est aussi appelée l'adresse **Physique** ou l'adresse **MAC**.

Pour connaître l'adresse Ethernet du destinataire, l'expéditeur fait appel au protocol **ARP**. Les informations reçues sont stockées dans une table. Pour visualiser ces informations, il convient d'utiliser la commande suivante :

```
[root@redhat9 ~]# arp -a
_gateway (10.0.2.1) at 92:8f:ca:52:ce:96 [ether] on ens18
```

Les options de cette commande sont :

```
[root@redhat9 ~]# arp --help
Usage:
  arp [-vn]  [<HW>] [-i <if>] [-a] [<hostname>]             <-Display ARP cache
  arp [-v]          [-i <if>] -d  <host> [pub]               <-Delete ARP entry
  arp [-vnD] [<HW>] [-i <if>] -f  [<filename>]            <-Add entry from file
  arp [-v]   [<HW>] [-i <if>] -s  <host> <hwaddr> [temp]             <-Add entry
  arp [-v]   [<HW>] [-i <if>] -Ds <host> <if> [netmask <nm>] pub           <-''-

        -a                      display (all) hosts in alternative (BSD) style
        -e                      display (all) hosts in default (Linux) style
        -s, --set               set a new ARP entry
        -d, --delete            delete a specified entry
        -v, --verbose           be verbose
        -n, --numeric           don't resolve names
        -i, --device            specify network interface (e.g. eth0)
        -D, --use-device        read <hwaddr> from given device
        -A, -p, --protocol      specify protocol family
        -f, --file              read new entries from file or from /etc/ethers

  <HW>=Use '-H <hw>' to specify hardware address type. Default: ether
```

```
  List of possible hardware types (which support ARP):
    ash (Ash) ether (Ethernet) ax25 (AMPR AX.25)
    netrom (AMPR NET/ROM) rose (AMPR ROSE) arcnet (ARCnet)
    dlci (Frame Relay DLCI) fddi (Fiber Distributed Data Interface) hippi (HIPPI)
    irda (IrLAP) x25 (generic X.25) infiniband (InfiniBand)
    eui64 (Generic EUI-64)
```

Comprendre IPv6

Présentation

IPv6 peut être utilisé en parallèle avec IPv4 dans un modèle à double pile. Dans cette configuration, une interface réseau peut avoir une ou plusieurs adresses IPv6 ainsi que des adresses IPv4. RHEL 9 fonctionne par défaut en mode double pile.

Adresses IPv6

Une adresse IPv6 est un nombre de 128 bits, normalement exprimé sous la forme de huit groupes de quatre **nibbles** (demi-octets) hexadécimaux séparés par deux points. Chaque nibble représente quatre bits de l'adresse IPv6, de sorte que chaque groupe représente 16 bits de l'adresse IPv6.

```
2001:0db8:0000:0010:0000:0000:0000:0001
```

Pour faciliter l'écriture des adresses IPv6, il n'est pas nécessaire d'écrire les zéros en tête d'un groupe séparé par deux points. Cependant, au moins un chiffre hexadécimal doit être écrit dans chaque groupe séparé par des deux-points :

```
2001:db8:0:10:0:0:0:1
```

En vertu de ces règles, 2001:db8::0010:0:0:0:1 serait une autre façon moins pratique d'écrire l'adresse de l'exemple, mais il s'agit d'une représentation valide de la même adresse

Le conseils pour rédiger des adresses lisibles de manière cohérente sont :

- Supprimer les zéros initiaux dans un groupe.
- Utiliser : : pour raccourcir autant que possible.
- Si une adresse contient deux groupes de zéros consécutifs de même longueur, il est préférable de raccourcir les groupes de zéros les plus à gauche en : : et les groupes les plus à droite en :0 : pour chaque groupe.
- Bien que cela soit autorisé, n'utiliser pas : : pour raccourcir un groupe de zéros. Utiliser plutôt :0 : et conserver : : pour les groupes de zéros consécutifs.
- Utiliser toujours des lettres minuscules pour les nombres hexadécimaux de a à f

Par exemple :

```
2001:db8:0:10::1
```

Dernièrement, un socket IPv6 doit comporter les caractères [ ] autour de l'adresse IPv6 :

```
[2001:db8:0:10::1]:80
```

Une adresse unicast IPv6 normale est divisée en deux parties :

- Le préfixe de réseau,
  - Le préfixe identifie le sous-réseau.
- L'identifiant d'interface,
  - Deux interfaces réseau sur le même sous-réseau ne peuvent pas avoir le même identifiant,
  - Un identifiant d'interface identifie une interface particulière sur le sous-réseau.

Masque de Sous-réseau

Contrairement à l'IPv4, l'IPv6 dispose d'un masque de sous-réseau standard de /64, utilisé pour presque toutes les adresses normales. Cela signifie qu'un seul sous-réseau peut contenir autant d'hôtes que nécessaire. Généralement, le fournisseur de réseau attribue un préfixe plus court à une organisation, par exemple, /48. Cela laisse au reste du réseau la possibilité d'attribuer des sous-réseaux (toujours de longueur /64) à partir du préfixe attribué. Pour un préfixe /48, il reste 16 bits pour les sous-réseaux, soit 65536 sous-réseaux.

Par exemple, dans le cas de l'adresse **2001:0db8:0000:0001:0000:0000:0000:0001**, exprimée en tant que **2001:0db8:0:1/64**, la partie NetID est **2001:0db8:0000:0001** et la partie HostID est **0000:0000:0000:0001**.

En regardant le NetID, la partie **2001:0db8:0000**, exprimée en tant que **2001:db8::/48** représente l'allocation fournie, tant que **0001/16** représente le sous-réseau.

Adresses IPv6 Réservées

Les adresses IPv6 réservés à une utilisation spécifique sont :

Adresse	Description
::1/128	L'adresse loopback similaire à l'adresse 127.0.0.1/8
::	L'adresse d'écoute global similaire à l'adresse 0.0.0.0
::/0	La route par défaut similaire à l'adresse 0.0.0.0/0
2000::/3	Cet espace d'adressage concerne les adresse réseaux allouées par l'IANA, allons de 2000::/16 à 3fff::/16
fd00::/8	Issue de la RFC 4193, ceci est similaire à la RFC 1918, c'est-à-dire une espace d'adressage privé.
fe80::/10	Adresses Link-local
ff00::/8	Adresses Multicast

L'Adresse Link-local

Chaque interface sur un réseau est configurée automatiquement avec une adresse Link-local :

```
[root@redhat9 ~]# ifconfig
ens18: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.102  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::2da3:cf78:c904:b9b9  prefixlen 64  scopeid 0x20<link>
        ether 92:86:d7:66:e7:5a  txqueuelen 1000  (Ethernet)
        RX packets 21754  bytes 51437196 (49.0 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 14363  bytes 1838520 (1.7 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
```

```
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 944  bytes 110925 (108.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 944  bytes 110925 (108.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Pour tester la connectivité de l'adresse Link-local, il convient d'utiliser la commande **ping6** :

```
[root@redhat9 ~]# ping6 -c4 fe80::2da3:cf78:c904:b9b9%ens18
PING fe80::2da3:cf78:c904:b9b9%ens18(fe80::2da3:cf78:c904:b9b9%ens18) 56 data bytes
64 bytes from fe80::2da3:cf78:c904:b9b9%ens18: icmp_seq=1 ttl=64 time=0.111 ms
64 bytes from fe80::2da3:cf78:c904:b9b9%ens18: icmp_seq=2 ttl=64 time=0.107 ms
64 bytes from fe80::2da3:cf78:c904:b9b9%ens18: icmp_seq=3 ttl=64 time=0.116 ms
64 bytes from fe80::2da3:cf78:c904:b9b9%ens18: icmp_seq=4 ttl=64 time=0.145 ms

--- fe80::2da3:cf78:c904:b9b9%ens18 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3054ms
rtt min/avg/max/mdev = 0.107/0.119/0.145/0.014 ms
```

Important : Notez qu'à la fin de l'adresse, il faut ajouter le **scope** qui est représenté par le caractère **%** suivi par le nom de l'interface.

DHCPv6

DHCPv6 ne fonctionne pas de la même façon que DHCPv4 parce qu'il n'existe pas d'adresses de diffusion sous IPv6.

En résumé, l'hôte envoie une requête DHCPv6 à l'adresse multicast **all-dhcp-servers**, **ff02::1:2** sur le port **547/udp**. En retour, l'hôte reçoit les informations demandées sur le port **546/udp** de son adresse Link-local.

Configurer le Réseau

RHEL 9 utilise **Network Manager** pour gérer le réseau. Network Manager est composé de deux éléments :

- un service qui gère les connexions réseaux et rapporte leurs états,
- des front-ends qui passent par un API de configuration du service.

Important : Notez qu'avec cette version de NetworkManager, IPv6 est activée par défaut.

Le service NetworkManager doit toujours être lancé :

```
[root@redhat9 ~]# systemctl status NetworkManager.service
● NetworkManager.service - Network Manager
     Loaded: loaded (/usr/lib/systemd/system/NetworkManager.service; enabled; preset: enabled)
     Active: active (running) since Sat 2024-09-28 15:37:17 CEST; 20h ago
       Docs: man:NetworkManager(8)
   Main PID: 857 (NetworkManager)
      Tasks: 3 (limit: 48800)
     Memory: 12.0M
        CPU: 1.834s
     CGroup: /system.slice/NetworkManager.service
             └─857 /usr/sbin/NetworkManager --no-daemon

Sep 28 15:37:18 redhat9.ittraining.loc NetworkManager[857]: <info>  [1727530638.6471] device (ens18): state
change: config -> ip-config (reason 'none', sys-iface-state: 'managed')
Sep 28 15:37:18 redhat9.ittraining.loc NetworkManager[857]: <info>  [1727530638.6481] policy: set 'ens18' (ens18)
as default for IPv4 routing and DNS
Sep 28 15:37:18 redhat9.ittraining.loc NetworkManager[857]: <info>  [1727530638.7310] device (ens18): state
change: ip-config -> ip-check (reason 'none', sys-iface-state: 'managed')
Sep 28 15:37:18 redhat9.ittraining.loc NetworkManager[857]: <info>  [1727530638.7327] device (ens18): state
```

```
change: ip-check -> secondaries (reason 'none', sys-iface-state: 'managed')
Sep 28 15:37:18 redhat9.ittraining.loc NetworkManager[857]: <info>  [1727530638.7328] device (ens18): state
change: secondaries -> activated (reason 'none', sys-iface-state: 'managed')
Sep 28 15:37:18 redhat9.ittraining.loc NetworkManager[857]: <info>  [1727530638.7331] manager: NetworkManager
state is now CONNECTED_SITE
Sep 28 15:37:18 redhat9.ittraining.loc NetworkManager[857]: <info>  [1727530638.7333] device (ens18): Activation:
successful, device activated.
Sep 28 15:37:18 redhat9.ittraining.loc NetworkManager[857]: <info>  [1727530638.7338] manager: NetworkManager
state is now CONNECTED_GLOBAL
Sep 28 15:37:18 redhat9.ittraining.loc NetworkManager[857]: <info>  [1727530638.7340] manager: startup complete
Sep 28 15:37:29 redhat9.ittraining.loc NetworkManager[857]: <info>  [1727530649.0717] agent-manager:
agent[3c1f9786a5e709c2,:1.25/org.gnome.Shell.NetworkAgent/42]: agent registered
```

La Commande nmcli

La commande **nmcli** (Network Manager Command Line Interface) est utilisée pour configurer NetworkManager.

Les options et les sous-commandes peuvent être consultées en utilisant les commandes suivantes :

```
[root@redhat9 ~]# nmcli help
Usage: nmcli [OPTIONS] OBJECT { COMMAND | help }

OPTIONS
  -a, --ask                                ask for missing parameters
  -c, --colors auto|yes|no                 whether to use colors in output
  -e, --escape yes|no                      escape columns separators in values
  -f, --fields <field,...>|all|common      specify fields to output
  -g, --get-values <field,...>|all|common  shortcut for -m tabular -t -f
  -h, --help                               print this help
  -m, --mode tabular|multiline             output mode
  -o, --overview                           overview mode
  -p, --pretty                             pretty output
  -s, --show-secrets                       allow displaying passwords
```

```
  -t, --terse                              terse output
  -v, --version                            show program version
  -w, --wait <seconds>                     set timeout waiting for finishing operations

OBJECT
  g[eneral]       NetworkManager's general status and operations
  n[etworking]    overall networking control
  r[adio]         NetworkManager radio switches
  c[onnection]    NetworkManager's connections
  d[evice]        devices managed by NetworkManager
  a[gent]         NetworkManager secret agent or polkit agent
  m[onitor]       monitor NetworkManager changes
```

LAB #1 - Configuration du Réseau

1.1 - Connections et Profils

NetworkManager inclus la notion de **connections** ou **profils** permettant des configurations différentes en fonction de la localisation. Pour voir les connections actuelles, utilisez la commande **nmcli c** avec la sous-commande **show** :

```
[root@redhat9 ~]# nmcli c show
NAME   UUID                                  TYPE      DEVICE
ens18  ea4c8254-6236-3130-8323-8b3f71d807a1  ethernet  ens18
lo     8df82174-1d45-4506-9248-6bfcd2d20240  loopback  lo
```

Créez donc un profil IP fixe rattaché au périphérique **ens18** :

```
[root@redhat9 ~]# nmcli connection add con-name ip_fixe ifname ens18 type ethernet ip4 10.0.2.102/24 gw4 10.0.2.1
Connection 'ip_fixe' (b3d51921-4deb-4975-ad52-f31993b2af0c) successfully added.
```

Constatez sa présence :

```
[root@redhat9 ~]# nmcli c show
NAME     UUID                                  TYPE      DEVICE
ens18    ea4c8254-6236-3130-8323-8b3f71d807a1  ethernet  ens18
lo       8df82174-1d45-4506-9248-6bfcd2d20240  loopback  lo
ip_fixe  b3d51921-4deb-4975-ad52-f31993b2af0c  ethernet  --
```

Notez que la sortie n'indique pas que le profil **ip_fixe** soit associé au periphérique **ens18** car le profil **ip_fixe** n'est pas activé :

```
[root@redhat9 ~]# nmcli d show
GENERAL.DEVICE:                         ens18
GENERAL.TYPE:                           ethernet
GENERAL.HWADDR:                         92:86:D7:66:E7:5A
GENERAL.MTU:                            1500
GENERAL.STATE:                          100 (connected)
GENERAL.CONNECTION:                     ens18
GENERAL.CON-PATH:                       /org/freedesktop/NetworkManager/ActiveConnection/2
WIRED-PROPERTIES.CARRIER:               on
IP4.ADDRESS[1]:                         10.0.2.101/24
IP4.GATEWAY:                            10.0.2.1
IP4.ROUTE[1]:                           dst = 10.0.2.0/24, nh = 0.0.0.0, mt = 100
IP4.ROUTE[2]:                           dst = 0.0.0.0/0, nh = 10.0.2.1, mt = 100
IP4.DNS[1]:                             8.8.8.8
IP6.ADDRESS[1]:                         fe80::9086:d7ff:fe66:e75a/64
IP6.GATEWAY:                            --
IP6.ROUTE[1]:                           dst = fe80::/64, nh = ::, mt = 1024

GENERAL.DEVICE:                         lo
GENERAL.TYPE:                           loopback
GENERAL.HWADDR:                         00:00:00:00:00:00
GENERAL.MTU:                            65536
GENERAL.STATE:                          100 (connected (externally))
GENERAL.CONNECTION:                     lo
GENERAL.CON-PATH:                       /org/freedesktop/NetworkManager/ActiveConnection/1
IP4.ADDRESS[1]:                         127.0.0.1/8
```

```
IP4.GATEWAY:                            --
IP6.ADDRESS[1]:                         ::1/128
IP6.GATEWAY:                            --
IP6.ROUTE[1]:                           dst = ::1/128, nh = ::, mt = 256
```

Pour activer le profil ip_fixe, utilisez la commande suivante :

```
[root@redhat9 ~]# nmcli connection up ip_fixe
```

Notez que votre terminal est bloqué à cause du changement de l'adresse IP.

A faire - Revenez à l'accueil de Guacamole et re-connectez-vous à la VM en tant que trainee en utilisant la connexion **RedHat9_SSH_10.0.2.102**.

Le profil ip_fixe est maintenant activé tandis que le profil ens18 a été désactivé :

```
[root@redhat9 ~]# nmcli c show
NAME     UUID                                  TYPE      DEVICE
ip_fixe  b3d51921-4deb-4975-ad52-f31993b2af0c  ethernet  ens18
lo       8df82174-1d45-4506-9248-6bfcd2d20240  loopback  lo
ens18    ea4c8254-6236-3130-8323-8b3f71d807a1  ethernet  --
[root@redhat9 ~]# nmcli d show
GENERAL.DEVICE:                         ens18
GENERAL.TYPE:                           ethernet
GENERAL.HWADDR:                         92:86:D7:66:E7:5A
GENERAL.MTU:                            1500
GENERAL.STATE:                          100 (connected)
GENERAL.CONNECTION:                     ip_fixe
GENERAL.CON-PATH:                       /org/freedesktop/NetworkManager/ActiveConnection/3
WIRED-PROPERTIES.CARRIER:               on
IP4.ADDRESS[1]:                         10.0.2.102/24
```

```
IP4.GATEWAY:                            10.0.2.1
IP4.ROUTE[1]:                           dst = 10.0.2.0/24, nh = 0.0.0.0, mt = 100
IP4.ROUTE[2]:                           dst = 0.0.0.0/0, nh = 10.0.2.1, mt = 100
IP6.ADDRESS[1]:                         fe80::2da3:cf78:c904:b9b9/64
IP6.GATEWAY:                            --
IP6.ROUTE[1]:                           dst = fe80::/64, nh = ::, mt = 1024

GENERAL.DEVICE:                         lo
GENERAL.TYPE:                           loopback
GENERAL.HWADDR:                         00:00:00:00:00:00
GENERAL.MTU:                            65536
GENERAL.STATE:                          100 (connected (externally))
GENERAL.CONNECTION:                     lo
GENERAL.CON-PATH:                       /org/freedesktop/NetworkManager/ActiveConnection/1
IP4.ADDRESS[1]:                         127.0.0.1/8
IP4.GATEWAY:                            --
IP6.ADDRESS[1]:                         ::1/128
IP6.GATEWAY:                            --
IP6.ROUTE[1]:                           dst = ::1/128, nh = ::, mt = 256
```

Pour consulter les paramètres du profil **ens18**, utilisez la commande suivante :

```
[root@redhat9 ~]# nmcli -p connection show ens18
===============================================================================
                      Connection profile details (ens18)
===============================================================================
connection.id:                          ens18
connection.uuid:                        ea4c8254-6236-3130-8323-8b3f71d807a1
connection.stable-id:                   --
connection.type:                        802-3-ethernet
connection.interface-name:              ens18
connection.autoconnect:                 yes
connection.autoconnect-priority:        -999
connection.autoconnect-retries:         -1 (default)
```

```
connection.multi-connect:               0 (default)
connection.auth-retries:                -1
connection.timestamp:                   1727605468
connection.permissions:                 --
connection.zone:                        --
connection.controller:                  --
connection.master:                      --
connection.slave-type:                  --
connection.port-type:                   --
connection.autoconnect-slaves:          -1 (default)
connection.autoconnect-ports:           -1 (default)
connection.secondaries:                 --
connection.gateway-ping-timeout:        0
connection.metered:                     unknown
connection.lldp:                        default
connection.mdns:                        -1 (default)
connection.llmnr:                       -1 (default)
connection.dns-over-tls:                -1 (default)
connection.mptcp-flags:                 0x0 (default)
connection.wait-device-timeout:         -1
connection.wait-activation-delay:       -1
------------------------------------------------------------------------------
802-3-ethernet.port:                    --
802-3-ethernet.speed:                   0
802-3-ethernet.duplex:                  --
802-3-ethernet.auto-negotiate:          no
802-3-ethernet.mac-address:             --
802-3-ethernet.cloned-mac-address:      --
802-3-ethernet.generate-mac-address-mask:--
802-3-ethernet.mac-address-blacklist:   --
802-3-ethernet.mtu:                     auto
802-3-ethernet.s390-subchannels:        --
802-3-ethernet.s390-nettype:            --
802-3-ethernet.s390-options:            --
```

```
802-3-ethernet.wake-on-lan:             default
802-3-ethernet.wake-on-lan-password:    --
802-3-ethernet.accept-all-mac-addresses:-1 (default)
---------------------------------------------------------------------------
ipv4.method:                            manual
ipv4.dns:                               8.8.8.8
ipv4.dns-search:                        --
ipv4.dns-options:                       --
ipv4.dns-priority:                      0
ipv4.addresses:                         10.0.2.101/24
lines 1-55
[q]
```

De même, pour consulter les paramètres du profil **ip_fixe**, utilisez la commande suivante :

```
[root@redhat9 ~]# nmcli -p connection show ip_fixe
===========================================================================
                    Connection profile details (ip_fixe)
===========================================================================
connection.id:                          ip_fixe
connection.uuid:                        b3d51921-4deb-4975-ad52-f31993b2af0c
connection.stable-id:                   --
connection.type:                        802-3-ethernet
connection.interface-name:              ens18
connection.autoconnect:                 yes
connection.autoconnect-priority:        0
connection.autoconnect-retries:         -1 (default)
connection.multi-connect:               0 (default)
connection.auth-retries:                -1
connection.timestamp:                   1727605469
connection.permissions:                 --
connection.zone:                        --
connection.controller:                  --
connection.master:                      --
```

```
connection.slave-type:                  --
connection.port-type:                   --
connection.autoconnect-slaves:          -1 (default)
connection.autoconnect-ports:           -1 (default)
connection.secondaries:                 --
connection.gateway-ping-timeout:        0
connection.metered:                     unknown
connection.lldp:                        default
connection.mdns:                        -1 (default)
connection.llmnr:                       -1 (default)
connection.dns-over-tls:                -1 (default)
connection.mptcp-flags:                 0x0 (default)
connection.wait-device-timeout:         -1
connection.wait-activation-delay:       -1
------------------------------------------------------------------------------
802-3-ethernet.port:                    --
802-3-ethernet.speed:                   0
802-3-ethernet.duplex:                  --
802-3-ethernet.auto-negotiate:          no
802-3-ethernet.mac-address:             --
802-3-ethernet.cloned-mac-address:      --
802-3-ethernet.generate-mac-address-mask:--
802-3-ethernet.mac-address-blacklist:   --
802-3-ethernet.mtu:                     auto
802-3-ethernet.s390-subchannels:        --
802-3-ethernet.s390-nettype:            --
802-3-ethernet.s390-options:            --
802-3-ethernet.wake-on-lan:             default
802-3-ethernet.wake-on-lan-password:    --
802-3-ethernet.accept-all-mac-addresses:-1 (default)
------------------------------------------------------------------------------
ipv4.method:                            manual
ipv4.dns:                               --
ipv4.dns-search:                        --
```

```
ipv4.dns-options:                       --
ipv4.dns-priority:                      0
ipv4.addresses:                         10.0.2.102/24
lines 1-55
[q]
```

Pour consulter la liste profils associés à un périphérique, utilisez la commande suivante :

```
[root@redhat9 ~]# nmcli -f CONNECTIONS device show ens18
CONNECTIONS.AVAILABLE-CONNECTION-PATHS:
/org/freedesktop/NetworkManager/Settings/1,/org/freedesktop/NetworkManager/Settings/3
CONNECTIONS.AVAILABLE-CONNECTIONS[1]:   ea4c8254-6236-3130-8323-8b3f71d807a1 | ens18
CONNECTIONS.AVAILABLE-CONNECTIONS[2]:   b3d51921-4deb-4975-ad52-f31993b2af0c | ip_fixe
```

Les fichiers de configuration pour le periphérique **ens18** se trouvent dans le répertoire **/etc/NetworkManager/system-connections** :

```
[root@redhat9 ~]# ls -l /etc/NetworkManager/system-connections
total 8
-rw-------. 1 root root 253 Oct 19  2023 ens18.nmconnection
-rw-------. 1 root root 218 Sep 29 12:21 ip_fixe.nmconnection
```

1.2 - Résolution des Noms

L'étude du fichier **/etc/NetworkManager/system-connections/ip_fixe.nmconnection** démontre l'abscence de directives concernant les DNS :

```
[root@redhat9 ~]# cat /etc/NetworkManager/system-connections/ip_fixe.nmconnection
[connection]
id=ip_fixe
uuid=b3d51921-4deb-4975-ad52-f31993b2af0c
type=ethernet
interface-name=ens18
```

```
[ethernet]

[ipv4]
address1=10.0.2.102/24,10.0.2.1
method=manual

[ipv6]
addr-gen-mode=default
method=auto

[proxy]
```

La résolution des noms est donc inactive :

```
[root@redhat9 ~]# ping www.free.fr
ping: www.free.fr: Name or service not known
```

Modifiez donc la configuration du profil **ip_fixe** :

```
[root@redhat9 ~]# nmcli connection mod ip_fixe ipv4.dns 8.8.8.8
```

L'étude du fichier **/etc/NetworkManager/system-connections/ip_fixe.nmconnection** démontre que la directive concernant le serveur DNS a été ajoutée :

```
[root@redhat9 ~]# cat /etc/NetworkManager/system-connections/ip_fixe.nmconnection
[connection]
id=ip_fixe
uuid=b3d51921-4deb-4975-ad52-f31993b2af0c
type=ethernet
interface-name=ens18
timestamp=1727605469

[ethernet]
```

```
[ipv4]
address1=10.0.2.102/24,10.0.2.1
dns=8.8.8.8;
method=manual

[ipv6]
addr-gen-mode=default
method=auto

[proxy]
```

Afin que la modification du serveur DNS soit prise en compte, re-démarrez le service NetworkManager :

```
[root@redhat9 ~]# systemctl restart NetworkManager.service

[root@redhat9 ~]# systemctl status NetworkManager.service
● NetworkManager.service - Network Manager
     Loaded: loaded (/usr/lib/systemd/system/NetworkManager.service; enabled; preset: enabled)
     Active: active (running) since Sun 2024-09-29 12:36:35 CEST; 11s ago
       Docs: man:NetworkManager(8)
   Main PID: 4456 (NetworkManager)
      Tasks: 4 (limit: 48800)
     Memory: 5.5M
        CPU: 82ms
     CGroup: /system.slice/NetworkManager.service
             └─4456 /usr/sbin/NetworkManager --no-daemon

Sep 29 12:36:35 redhat9.ittraining.loc NetworkManager[4456]: <info>  [1727606195.5342] device (lo): state change:
secondaries -> activated (reason 'none', sys-iface-state: 'external')
Sep 29 12:36:35 redhat9.ittraining.loc NetworkManager[4456]: <info>  [1727606195.5347] device (lo): Activation:
successful, device activated.
Sep 29 12:36:35 redhat9.ittraining.loc NetworkManager[4456]: <info>  [1727606195.5360] device (ens18): state
change: ip-config -> ip-check (reason 'none', sys-iface-state: 'assume')
Sep 29 12:36:35 redhat9.ittraining.loc NetworkManager[4456]: <info>  [1727606195.5654] device (ens18): state
```

```
change: ip-check -> secondaries (reason 'none', sys-iface-state: 'assume')
Sep 29 12:36:35 redhat9.ittraining.loc NetworkManager[4456]: <info>  [1727606195.5660] device (ens18): state
change: secondaries -> activated (reason 'none', sys-iface-state: 'assume')
Sep 29 12:36:35 redhat9.ittraining.loc NetworkManager[4456]: <info>  [1727606195.5663] manager: NetworkManager
state is now CONNECTED_SITE
Sep 29 12:36:35 redhat9.ittraining.loc NetworkManager[4456]: <info>  [1727606195.5666] device (ens18):
Activation: successful, device activated.
Sep 29 12:36:35 redhat9.ittraining.loc NetworkManager[4456]: <info>  [1727606195.5671] manager: NetworkManager
state is now CONNECTED_GLOBAL
Sep 29 12:36:35 redhat9.ittraining.loc NetworkManager[4456]: <info>  [1727606195.5673] manager: startup complete
Sep 29 12:36:36 redhat9.ittraining.loc NetworkManager[4456]: <info>  [1727606196.0852] agent-manager:
agent[923443df876692f7,:1.25/org.gnome.Shell.NetworkAgent/42]: agent registered
```

Vérifiez que le fichier **/etc/resolv.conf** ait été modifié par NetworkManager :

```
[root@redhat9 ~]# cat /etc/resolv.conf
# Generated by NetworkManager
search ittraining.loc
nameserver 8.8.8.8
```

Dernièrement vérifiez la resolution des noms :

```
[root@redhat9 ~]# ping www.free.fr
PING www.free.fr (212.27.48.10) 56(84) bytes of data.
64 bytes from www.free.fr (212.27.48.10): icmp_seq=1 ttl=47 time=89.2 ms
64 bytes from www.free.fr (212.27.48.10): icmp_seq=2 ttl=47 time=89.2 ms
64 bytes from www.free.fr (212.27.48.10): icmp_seq=3 ttl=47 time=89.3 ms
^C
--- www.free.fr ping statistics ---
4 packets transmitted, 3 received, 25% packet loss, time 3005ms
rtt min/avg/max/mdev = 89.153/89.186/89.252/0.046 ms
```

Important : Notez qu'il existe un front-end graphique en mode texte, **nmtui**, pour configurer NetworkManager.

1.3 - Ajouter une Deuxième Adresse IP à un Profil

Pour ajouter une deuxième adresse IP à un profil sous RHEL 9, il convient d'utiliser la commande suivante :

```
[root@redhat9 ~]# nmcli connection mod ip_fixe +ipv4.addresses 192.168.1.2/24
```

Rechargez la configuration du profil :

```
[root@redhat9 ~]# nmcli con up ip_fixe
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/3)
```

Saisissez ensuite la commande suivante :

```
[root@redhat9 ~]# nmcli connection show ip_fixe
connection.id:                          ip_fixe
connection.uuid:                        b3d51921-4deb-4975-ad52-f31993b2af0c
connection.stable-id:                   --
connection.type:                        802-3-ethernet
connection.interface-name:              ens18
connection.autoconnect:                 yes
connection.autoconnect-priority:        0
connection.autoconnect-retries:         -1 (default)
connection.multi-connect:               0 (default)
connection.auth-retries:                -1
connection.timestamp:                   1727606325
connection.permissions:                 --
connection.zone:                        --
connection.controller:                  --
```

```
connection.master:                      --
connection.slave-type:                  --
connection.port-type:                   --
connection.autoconnect-slaves:          -1 (default)
connection.autoconnect-ports:           -1 (default)
connection.secondaries:                 --
connection.gateway-ping-timeout:        0
connection.metered:                     unknown
connection.lldp:                        default
connection.mdns:                        -1 (default)
connection.llmnr:                       -1 (default)
connection.dns-over-tls:                -1 (default)
connection.mptcp-flags:                 0x0 (default)
connection.wait-device-timeout:         -1
connection.wait-activation-delay:       -1
802-3-ethernet.port:                    --
802-3-ethernet.speed:                   0
802-3-ethernet.duplex:                  --
802-3-ethernet.auto-negotiate:          no
802-3-ethernet.mac-address:             --
802-3-ethernet.cloned-mac-address:      --
802-3-ethernet.generate-mac-address-mask:--
802-3-ethernet.mac-address-blacklist:   --
802-3-ethernet.mtu:                     auto
802-3-ethernet.s390-subchannels:        --
802-3-ethernet.s390-nettype:            --
802-3-ethernet.s390-options:            --
802-3-ethernet.wake-on-lan:             default
802-3-ethernet.wake-on-lan-password:    --
802-3-ethernet.accept-all-mac-addresses:-1 (default)
ipv4.method:                            manual
ipv4.dns:                               8.8.8.8
ipv4.dns-search:                        --
ipv4.dns-options:                       --
```

```
ipv4.dns-priority:                      0
ipv4.addresses:                         10.0.2.102/24, 192.168.1.2/24
ipv4.gateway:                           10.0.2.1
ipv4.routes:                            --
ipv4.route-metric:                      -1
ipv4.route-table:                       0 (unspec)
ipv4.routing-rules:                     --
ipv4.replace-local-rule:                -1 (default)
ipv4.ignore-auto-routes:                no
ipv4.ignore-auto-dns:                   no
ipv4.dhcp-client-id:                    --
ipv4.dhcp-iaid:                         --
ipv4.dhcp-dscp:                         --
ipv4.dhcp-timeout:                      0 (default)
ipv4.dhcp-send-hostname:                yes
ipv4.dhcp-hostname:                     --
ipv4.dhcp-fqdn:                         --
ipv4.dhcp-hostname-flags:               0x0 (none)
ipv4.never-default:                     no
ipv4.may-fail:                          yes
ipv4.required-timeout:                  -1 (default)
ipv4.dad-timeout:                       -1 (default)
ipv4.dhcp-vendor-class-identifier:      --
ipv4.link-local:                        0 (default)
ipv4.dhcp-reject-servers:               --
ipv4.auto-route-ext-gw:                 -1 (default)
ipv6.method:                            auto
ipv6.dns:                               --
ipv6.dns-search:                        --
ipv6.dns-options:                       --
ipv6.dns-priority:                      0
ipv6.addresses:                         --
ipv6.gateway:                           --
ipv6.routes:                            --
```

```
ipv6.route-metric:                      -1
ipv6.route-table:                       0 (unspec)
ipv6.routing-rules:                     --
ipv6.replace-local-rule:                -1 (default)
ipv6.ignore-auto-routes:                no
ipv6.ignore-auto-dns:                   no
ipv6.never-default:                     no
ipv6.may-fail:                          yes
ipv6.required-timeout:                  -1 (default)
ipv6.ip6-privacy:                       -1 (unknown)
ipv6.addr-gen-mode:                     default
ipv6.ra-timeout:                        0 (default)
ipv6.mtu:                               auto
ipv6.dhcp-pd-hint:                      --
ipv6.dhcp-duid:                         --
ipv6.dhcp-iaid:                         --
ipv6.dhcp-timeout:                      0 (default)
ipv6.dhcp-send-hostname:                yes
ipv6.dhcp-hostname:                     --
ipv6.dhcp-hostname-flags:               0x0 (none)
ipv6.auto-route-ext-gw:                 -1 (default)
ipv6.token:                             --
proxy.method:                           none
proxy.browser-only:                     no
proxy.pac-url:                          --
proxy.pac-script:                       --
GENERAL.NAME:                           ip_fixe
GENERAL.UUID:                           b3d51921-4deb-4975-ad52-f31993b2af0c
GENERAL.DEVICES:                        ens18
GENERAL.IP-IFACE:                       ens18
GENERAL.STATE:                          activated
GENERAL.DEFAULT:                        yes
GENERAL.DEFAULT6:                       no
GENERAL.SPEC-OBJECT:                    --
```

```
GENERAL.VPN:                            no
GENERAL.DBUS-PATH:                      /org/freedesktop/NetworkManager/ActiveConnection/3
GENERAL.CON-PATH:                       /org/freedesktop/NetworkManager/Settings/3
GENERAL.ZONE:                           --
GENERAL.MASTER-PATH:                    --
IP4.ADDRESS[1]:                         192.168.1.2/24
IP4.ADDRESS[2]:                         10.0.2.102/24
IP4.GATEWAY:                            10.0.2.1
IP4.ROUTE[1]:                           dst = 10.0.2.0/24, nh = 0.0.0.0, mt = 100
IP4.ROUTE[2]:                           dst = 192.168.1.0/24, nh = 0.0.0.0, mt = 100
IP4.ROUTE[3]:                           dst = 0.0.0.0/0, nh = 10.0.2.1, mt = 100
IP4.DNS[1]:                             8.8.8.8
IP6.ADDRESS[1]:                         fe80::2da3:cf78:c904:b9b9/64
IP6.GATEWAY:                            --
IP6.ROUTE[1]:                           dst = fe80::/64, nh = ::, mt = 1024
lines 77-131/131 (END)
[q]
```

Important : Notez l'ajout de l'adresse secondaire à la ligne **ipv4.addresses:** ainsi que l'ajout de la ligne **IP4.ADDRESS[2]:**.

Consultez maintenant le contenu du fichier **/etc/NetworkManager/system-connections/ip_fixe.nmconnection** :

```
[root@redhat9 ~]# cat /etc/NetworkManager/system-connections/ip_fixe.nmconnection
[connection]
id=ip_fixe
uuid=b3d51921-4deb-4975-ad52-f31993b2af0c
type=ethernet
interface-name=ens18
timestamp=1727606195
```

```
[ethernet]

[ipv4]
address1=10.0.2.102/24,10.0.2.1
address2=192.168.1.2/24
dns=8.8.8.8;
method=manual

[ipv6]
addr-gen-mode=default
method=auto

[proxy]
```

Important : Notez l'ajout de la ligne **address2=192.168.1.2/24**.

1.4 - La Commande hostname

La procédure de la modification du hostname est simplifiée et sa prise en compte est immédiate :

```
[root@redhat9 ~]# hostname
redhat9.ittraining.loc

[root@redhat9 ~]# cat /etc/hostname
redhat9.ittraining.loc

[root@redhat9 ~]# nmcli general hostname redhat.ittraining.loc

[root@redhat9 ~]# cat /etc/hostname
redhat.ittraining.loc
```

```
[root@redhat9 ~]# nmcli general hostname redhat9.ittraining.loc

[root@redhat9 ~]# cat /etc/hostname
redhat9.ittraining.loc

[root@redhat9 ~]# hostname
redhat9.ittraining.loc
```

1.5 - La Commande ip

Sous RHEL 9 la commande **ip** est préférée par rapport à la commande ifconfig :

```
[root@redhat9 ~]# ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 92:86:d7:66:e7:5a brd ff:ff:ff:ff:ff:ff
    altname enp0s18
    inet 10.0.2.102/24 brd 10.0.2.255 scope global noprefixroute ens18
       valid_lft forever preferred_lft forever
    inet 192.168.1.2/24 brd 192.168.1.255 scope global noprefixroute ens18
       valid_lft forever preferred_lft forever
    inet6 fe80::2da3:cf78:c904:b9b9/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Options de la Commande ip

Les options de cette commande sont :

```
[root@redhat9 ~]# ip --help
Usage: ip [ OPTIONS ] OBJECT { COMMAND | help }
       ip [ -force ] -batch filename
where  OBJECT := { address | addrlabel | amt | fou | help | ila | ioam | l2tp |
                   link | macsec | maddress | monitor | mptcp | mroute | mrule |
                   neighbor | neighbour | netconf | netns | nexthop | ntable |
                   ntbl | route | rule | sr | tap | tcpmetrics |
                   token | tunnel | tuntap | vrf | xfrm }
       OPTIONS := { -V[ersion] | -s[tatistics] | -d[etails] | -r[esolve] |
                    -h[uman-readable] | -iec | -j[son] | -p[retty] |
                    -f[amily] { inet | inet6 | mpls | bridge | link } |
                    -4 | -6 | -M | -B | -0 |
                    -l[oops] { maximum-addr-flush-attempts } | -br[ief] |
                    -o[neline] | -t[imestamp] | -ts[hort] | -b[atch] [filename] |
                    -rc[vbuf] [size] | -n[etns] name | -N[umeric] | -a[ll] |
                    -c[olor]}
```

1.6 - Activer/Désactiver une Interface Manuellement

Deux commandes existent pour désactiver et activer manuellement une interface réseau :

```
# nmcli device disconnect enp0s3
# nmcli device connect enp0s3
```

Important : Veuillez ne **PAS** exécuter ces deux commandes.

1.7 - Routage Statique

La commande ip

Sous RHEL 9, pour supprimer la route vers le réseau 192.168.1.0 il convient d'utiliser la commande ip et non pas la commande route :

```
[root@redhat9 ~]# ip route
default via 10.0.2.1 dev ens18 proto static metric 100
10.0.2.0/24 dev ens18 proto kernel scope link src 10.0.2.102 metric 100
192.168.1.0/24 dev ens18 proto kernel scope link src 192.168.1.2 metric 100

[root@redhat9 ~]# ip route del 192.168.1.0/24 via 0.0.0.0

[root@redhat9 ~]# ip route
default via 10.0.2.1 dev ens18 proto static metric 100
10.0.2.0/24 dev ens18 proto kernel scope link src 10.0.2.102 metric 100
```

Pour ajouter la route vers le réseau 192.168.1.0 :

```
[root@redhat9 ~]# ip route add 192.168.1.0/24 via 10.0.2.1

[root@redhat9 ~]# ip route
default via 10.0.2.1 dev ens18 proto static metric 100
10.0.2.0/24 dev ens18 proto kernel scope link src 10.0.2.102 metric 100
192.168.1.0/24 via 10.0.2.1 dev ens18
```

Important - La commande utilisée pour ajouter une passerelle par défaut prend la forme suivante **ip route add default via *adresse ip***.

Activer le routage sur le serveur

Pour activer le routage IPv4 sur le serveur, il convient d'activer la retransmission des paquets:

```
[root@redhat9 ~]# cat /proc/sys/net/ipv4/ip_forward
0
[root@redhat9 ~]# echo 1 > /proc/sys/net/ipv4/ip_forward

[root@redhat9 ~]# cat /proc/sys/net/ipv4/ip_forward
1
```

Pour activer le routage IPv6 sur le serveur, il convient d'activer la retransmission des paquets:

```
[root@redhat9 ~]# cat /proc/sys/net/ipv6/conf/all/forwarding
0

[root@redhat9 ~]# echo "1" > /proc/sys/net/ipv6/conf/all/forwarding

[root@redhat9 ~]# cat /proc/sys/net/ipv6/conf/all/forwarding
1
```

LAB #2 - Diagnostique du Réseau

2.1 - ping

Pour tester l'accessibilité d'une machine, vous devez utiliser la commande **ping** :

```
[root@redhat9 ~]# ping -c4 10.0.2.1
PING 10.0.2.1 (10.0.2.1) 56(84) bytes of data.
64 bytes from 10.0.2.1: icmp_seq=1 ttl=64 time=0.157 ms
64 bytes from 10.0.2.1: icmp_seq=2 ttl=64 time=0.130 ms
```

```
64 bytes from 10.0.2.1: icmp_seq=3 ttl=64 time=0.212 ms
64 bytes from 10.0.2.1: icmp_seq=4 ttl=64 time=0.222 ms

--- 10.0.2.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3104ms
rtt min/avg/max/mdev = 0.130/0.180/0.222/0.038 ms
```

Options de la commande ping

Les options de cette commande sont :

```
[root@redhat9 ~]# ping --help
ping: invalid option -- '-'

Usage
  ping [options] <destination>

Options:
  <destination>      dns name or ip address
  -a                 use audible ping
  -A                 use adaptive ping
  -B                 sticky source address
  -c <count>         stop after <count> replies
  -D                 print timestamps
  -d                 use SO_DEBUG socket option
  -f                 flood ping
  -h                 print help and exit
  -I <interface>     either interface name or address
  -i <interval>      seconds between sending each packet
  -L                 suppress loopback of multicast packets
  -l <preload>       send <preload> number of packages while waiting replies
  -m <mark>          tag the packets going out
  -M <pmtud opt>     define mtu discovery, can be one of <do|dont|want>
```

```
  -n                  no dns name resolution
  -O                  report outstanding replies
  -p <pattern>        contents of padding byte
  -q                  quiet output
  -Q <tclass>         use quality of service <tclass> bits
  -s <size>           use <size> as number of data bytes to be sent
  -S <size>           use <size> as SO_SNDBUF socket option value
  -t <ttl>            define time to live
  -U                  print user-to-user latency
  -v                  verbose output
  -V                  print version and exit
  -w <deadline>       reply wait <deadline> in seconds
  -W <timeout>        time to wait for response

IPv4 options:
  -4                  use IPv4
  -b                  allow pinging broadcast
  -R                  record route
  -T <timestamp>      define timestamp, can be one of <tsonly|tsandaddr|tsprespec>

IPv6 options:
  -6                  use IPv6
  -F <flowlabel>      define flow label, default is random
  -N <nodeinfo opt>   use icmp6 node info query, try <help> as argument

For more details see ping(8).
```

2.2 - netstat -i

Pour visualiser les statistiques réseaux, vous disposez de la commande **netstat** :

```
[root@redhat9 ~]# netstat -i
Kernel Interface table
```

```
Iface             MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
ens18            1500    18785      0      0 0         12157      0      0      0 BMRU
lo              65536      105      0      0 0           105      0      0      0 LRU
```

Options de la commande netstat

Les options de cette commande sont :

```
[root@redhat9 ~]# netstat --help
usage: netstat [-vWeenNcCF] [<Af>] -r         netstat {-V|--version|-h|--help}
       netstat [-vWnNcaeol] [<Socket> ...]
       netstat { [-vWeenNac] -I[<Iface>] | [-veenNac] -i | [-cnNe] -M | -s [-6tuw] } [delay]

        -r, --route              display routing table
        -I, --interfaces=<Iface> display interface table for <Iface>
        -i, --interfaces         display interface table
        -g, --groups             display multicast group memberships
        -s, --statistics         display networking statistics (like SNMP)
        -M, --masquerade         display masqueraded connections

        -v, --verbose            be verbose
        -W, --wide               don't truncate IP addresses
        -n, --numeric            don't resolve names
        --numeric-hosts          don't resolve host names
        --numeric-ports          don't resolve port names
        --numeric-users          don't resolve user names
        -N, --symbolic           resolve hardware names
        -e, --extend             display other/more information
        -p, --programs           display PID/Program name for sockets
        -o, --timers             display timers
        -c, --continuous         continuous listing

        -l, --listening          display listening server sockets
```

```
       -a, --all                display all sockets (default: connected)
       -F, --fib                display Forwarding Information Base (default)
       -C, --cache              display routing cache instead of FIB
       -Z, --context            display SELinux security context for sockets

  <Socket>={-t|--tcp} {-u|--udp} {-U|--udplite} {-S|--sctp} {-w|--raw}
           {-x|--unix} --ax25 --ipx --netrom
  <AF>=Use '-6|-4' or '-A <af>' or '--<af>'; default: inet
  List of possible address families (which support routing):
    inet (DARPA Internet) inet6 (IPv6) ax25 (AMPR AX.25)
    netrom (AMPR NET/ROM) ipx (Novell IPX) ddp (Appletalk DDP)
    x25 (CCITT X.25)
```

2.3 - traceroute

La commande ping est à la base de la commande **traceroute**. Cette commande sert à découvrir la route empruntée pour accéder à un site donné :

```
[root@redhat9 ~]# traceroute www.ittraining.team
bash: traceroute: command not found...
Install package 'traceroute' to provide command 'traceroute'? [N/y] y


 * Waiting in queue...
 * Loading list of packages....
The following packages have to be installed:
 traceroute-3:2.1.0-18.el9.x86_64       Traces the route taken by packets over an IPv4/IPv6 network
Proceed with changes? [N/y] y


 * Waiting in queue...
 * Waiting for authentication...
 * Waiting in queue...
 * Downloading packages...
```

```
 * Requesting data...
 * Testing changes...
 * Installing packages...
traceroute to www.ittraining.team (136.143.190.199), 30 hops max, 60 byte packets
 1  _gateway (10.0.2.1)  0.437 ms  0.396 ms  0.380 ms
 2  51.79.19.252 (51.79.19.252)  0.554 ms  0.689 ms  0.818 ms
 3  10.161.82.50 (10.161.82.50)  0.372 ms 10.161.82.52 (10.161.82.52)  0.444 ms 10.161.82.50 (10.161.82.50)
0.372 ms
 4  10.34.98.50 (10.34.98.50)  0.483 ms  0.580 ms  0.973 ms
 5  10.74.8.92 (10.74.8.92)  0.232 ms 10.74.8.90 (10.74.8.90)  9.825 ms 10.74.8.94 (10.74.8.94)  0.206 ms
 6  10.95.81.10 (10.95.81.10)  0.712 ms 10.95.81.8 (10.95.81.8)  1.103 ms 10.95.81.10 (10.95.81.10)  1.435 ms
 7  be101.chi-ch2-sbb1-8k.il.us (198.27.73.207)  17.425 ms be101.chi-ch2-sbb2-8k.il.us (192.99.146.141)  17.089
ms be101.chi-ch2-sbb1-8k.il.us (198.27.73.207)  17.055 ms
 8  * * *
 9  10.200.1.1 (10.200.1.1)  66.593 ms  68.592 ms *
10  * 10.200.1.1 (10.200.1.1)  68.518 ms *
11  * * *
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
21  * *^C
```

Options de la commande traceroute

Les options de cette commande sont :

```
[root@redhat9 ~]# traceroute --help
```

```
Usage:
  traceroute [ -46dFITnreAUDV ] [ -f first_ttl ] [ -g gate,... ] [ -i device ] [ -m max_ttl ] [ -N squeries ] [ -
p port ] [ -t tos ] [ -l flow_label ] [ -w MAX,HERE,NEAR ] [ -q nqueries ] [ -s src_addr ] [ -z sendwait ] [ --
fwmark=num ] host [ packetlen ]
Options:
  -4                          Use IPv4
  -6                          Use IPv6
  -d  --debug                 Enable socket level debugging
  -F  --dont-fragment         Do not fragment packets
  -f first_ttl  --first=first_ttl
                              Start from the first_ttl hop (instead from 1)
  -g gate,...  --gateway=gate,...
                              Route packets through the specified gateway
                              (maximum 8 for IPv4 and 127 for IPv6)
  -I  --icmp                  Use ICMP ECHO for tracerouting
  -T  --tcp                   Use TCP SYN for tracerouting (default port is 80)
  -i device  --interface=device
                              Specify a network interface to operate with
  -m max_ttl  --max-hops=max_ttl
                              Set the max number of hops (max TTL to be
                              reached). Default is 30
  -N squeries  --sim-queries=squeries
                              Set the number of probes to be tried
                              simultaneously (default is 16)
  -n                          Do not resolve IP addresses to their domain names
  -p port  --port=port        Set the destination port to use. It is either
                              initial udp port value for "default" method
                              (incremented by each probe, default is 33434), or
                              initial seq for "icmp" (incremented as well,
                              default from 1), or some constant destination
                              port for other methods (with default of 80 for
                              "tcp", 53 for "udp", etc.)
  -t tos  --tos=tos           Set the TOS (IPv4 type of service) or TC (IPv6
                              traffic class) value for outgoing packets
```

```
-l flow_label  --flowlabel=flow_label
                            Use specified flow_label for IPv6 packets
-w MAX,HERE,NEAR  --wait=MAX,HERE,NEAR
                            Wait for a probe no more than HERE (default 3)
                            times longer than a response from the same hop,
                            or no more than NEAR (default 10) times than some
                            next hop, or MAX (default 5.0) seconds (float
                            point values allowed too)
-q nqueries  --queries=nqueries
                            Set the number of probes per each hop. Default is
                            3
-r                          Bypass the normal routing and send directly to a
                            host on an attached network
-s src_addr  --source=src_addr
                            Use source src_addr for outgoing packets
-z sendwait  --sendwait=sendwait
                            Minimal time interval between probes (default 0).
                            If the value is more than 10, then it specifies a
                            number in milliseconds, else it is a number of
                            seconds (float point values allowed too)
-e  --extensions            Show ICMP extensions (if present), including MPLS
-A  --as-path-lookups       Perform AS path lookups in routing registries and
                            print results directly after the corresponding
                            addresses
-M name  --module=name      Use specified module (either builtin or external)
                            for traceroute operations. Most methods have
                            their shortcuts (`-I' means `-M icmp' etc.)
-O OPTS,...  --options=OPTS,...
                            Use module-specific option OPTS for the
                            traceroute module. Several OPTS allowed,
                            separated by comma. If OPTS is "help", print info
                            about available options
--sport=num                 Use source port num for outgoing packets. Implies
                            `-N 1'
```

```
  --fwmark=num                Set firewall mark for outgoing packets
  -U  --udp                   Use UDP to particular port for tracerouting
                              (instead of increasing the port per each probe),
                              default port is 53
  -UL                         Use UDPLITE for tracerouting (default dest port
                              is 53)
  -D  --dccp                  Use DCCP Request for tracerouting (default port
                              is 33434)
  -P prot  --protocol=prot    Use raw packet of protocol prot for tracerouting
  --mtu                       Discover MTU along the path being traced. Implies
                              `-F -N 1'
  --back                      Guess the number of hops in the backward path and
                              print if it differs
  -V  --version               Print version info and exit
  --help                      Read this help and exit

Arguments:
+     host          The host to traceroute to
      packetlen     The full packet length (default is the length of an IP
                    header plus 40). Can be ignored or increased to a minimal
                    allowed value
```

2.4 - tracepath

Une autre commande qui sert à découvrir la route empruntée pour accéder à un site donné est **tracepath** :

```
[root@redhat9 ~]# tracepath www.ittraining.team
 1?: [LOCALHOST]                      pmtu 1500
 1:  _gateway                                              0.199ms
 1:  _gateway                                              0.138ms
 2:  51.79.19.252                                          0.651ms
 3:  10.161.82.52                                          0.704ms
 4:  10.34.98.56                                           0.628ms
```

```
 5:  10.74.8.88                                            0.301ms
 6:  10.95.81.8                                           36.365ms
 7:  be101.chi-ch2-sbb2-8k.il.us                          17.152ms
 8:  be101.chi-ch2-sbb2-8k.il.us                          17.442ms asymm  7
 9:  10.200.1.1                                           69.375ms
10:  10.200.1.1                                           68.618ms asymm  9
11:  no reply
12:  no reply
13:  no reply
14:  no reply
^C
```

Options de la commande tracepath

Les options de cette commande sont :

```
[root@redhat9 ~]# tracepath --help
tracepath: invalid option -- '-'

Usage
  tracepath [options] <destination>

Options:
  -4             use IPv4
  -6             use IPv6
  -b             print both name and ip
  -l <length>    use packet <length>
  -m <hops>      use maximum <hops>
  -n             no dns name resolution
  -p <port>      use destination <port>
  -V             print version and exit
  <destination>  dns name or ip address
```

```
For more details see tracepath(8).
```

LAB #3 - Connexions à Distance

3.1 - Telnet

Important - Si la commande **telnet** n'est pas installée sous RedHat 9, installez-le à l'aide de la commande **dnf install telnet** en tant que root.

La commande **telnet** est utilisée pour établir une connexion à distance avec un serveur telnet :

```
  # telnet numero_ip
```

Important - Le service telnet revient à une redirection des canaux standards d'entrée et de sortie. Notez que la connexion n'est **pas** sécurisée. Pour fermer la connexion, il faut saisir la commande **exit**. La commande telnet n'offre pas de services de transfert de fichiers. Pour cela, il convient d'utiliser la command **ftp**.

Options de la commande telnet

Les options de cette commande sont :

```
[root@redhat9 ~]# telnet --help
telnet: invalid option -- '-'
Usage: telnet [-4] [-6] [-8] [-E] [-L] [-S tos] [-a] [-c] [-d] [-e char] [-l user]
```

```
        [-n tracefile] [-b hostalias ] [-r]
 [host-name [port]]
```

3.2 - wget

La commande **wget** est utilisée pour récupérer un fichier via http, https ou ftp :

```
[root@redhat9 ~]# wget
https://www.dropbox.com/scl/fi/c0cbo91y2i7qwjexeldgt/wget_file.txt?rlkey=g8fgje9z8oeqgb4nd2g7x3wkx
--2024-09-29 13:56:10--
https://www.dropbox.com/scl/fi/c0cbo91y2i7qwjexeldgt/wget_file.txt?rlkey=g8fgje9z8oeqgb4nd2g7x3wkx
Resolving www.dropbox.com (www.dropbox.com)... 162.125.11.18, 2620:100:6050:18::a27d:b12
Connecting to www.dropbox.com (www.dropbox.com)|162.125.11.18|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location:
https://uc22f408c6cacfcc03fb4dbb269d.dl.dropboxusercontent.com/cd/0/inline/CbfNI4-pa7NlBqzKN3_KWHUoXFn8kcMiV99ekp
nDl2iVRanx9yx3YdpDcC5FHk8MJqHFfDnbPFeENko4TWJUAKMwZJ82s18b69iKmgbpMCFyd5oGQHLZs6uB3xy0_nmJl59Ru2MjCGeyEo9ikQ3Uaaq
Y/file# [following]
--2024-09-29 13:56:11--
https://uc22f408c6cacfcc03fb4dbb269d.dl.dropboxusercontent.com/cd/0/inline/CbfNI4-pa7NlBqzKN3_KWHUoXFn8kcMiV99ekp
nDl2iVRanx9yx3YdpDcC5FHk8MJqHFfDnbPFeENko4TWJUAKMwZJ82s18b69iKmgbpMCFyd5oGQHLZs6uB3xy0_nmJl59Ru2MjCGeyEo9ikQ3Uaaq
Y/file
Resolving uc22f408c6cacfcc03fb4dbb269d.dl.dropboxusercontent.com
(uc22f408c6cacfcc03fb4dbb269d.dl.dropboxusercontent.com)... 162.125.11.15, 2620:100:6050:15::a27d:b0f
Connecting to uc22f408c6cacfcc03fb4dbb269d.dl.dropboxusercontent.com
(uc22f408c6cacfcc03fb4dbb269d.dl.dropboxusercontent.com)|162.125.11.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 46 [text/plain]
Saving to: ‘wget_file.txt?rlkey=g8fgje9z8oeqgb4nd2g7x3wkx’

wget_file.txt?rlkey=g8fgje9z8oeqgb4nd2g7x3wkx
100%[=========================================================================================
========>]      46  --.-KB/s    in 0s
```

```
2024-09-29 13:56:11 (40.9 MB/s) - ‘wget_file.txt?rlkey=g8fgje9z8oeqgb4nd2g7x3wkx’ saved [46/46]

[root@redhat9 ~]# cat wget_file.txt?rlkey=g8fgje9z8oeqgb4nd2g7x3wkx
This is a file retrieved by the wget command.
```

Options de la commande wget

Les options de cette commande sont :

```
[root@redhat9 ~]# wget --help
GNU Wget 1.21.1, a non-interactive network retriever.
Usage: wget [OPTION]... [URL]...

Mandatory arguments to long options are mandatory for short options too.

Startup:
  -V,  --version                   display the version of Wget and exit
  -h,  --help                      print this help
  -b,  --background                go to background after startup
  -e,  --execute=COMMAND           execute a `.wgetrc'-style command

Logging and input file:
  -o,  --output-file=FILE          log messages to FILE
  -a,  --append-output=FILE        append messages to FILE
  -d,  --debug                     print lots of debugging information
  -q,  --quiet                     quiet (no output)
  -v,  --verbose                   be verbose (this is the default)
  -nv, --no-verbose                turn off verboseness, without being quiet
       --report-speed=TYPE         output bandwidth as TYPE.  TYPE can be bits
  -i,  --input-file=FILE           download URLs found in local or external FILE
  -F,  --force-html                treat input file as HTML
  -B,  --base=URL                  resolves HTML input-file links (-i -F)
                                     relative to URL
```

```
       --config=FILE                specify config file to use
       --no-config                  do not read any config file
       --rejected-log=FILE          log reasons for URL rejection to FILE

Download:
  -t,  --tries=NUMBER               set number of retries to NUMBER (0 unlimits)
       --retry-connrefused          retry even if connection is refused
       --retry-on-http-error=ERRORS    comma-separated list of HTTP errors to retry
  -O,  --output-document=FILE       write documents to FILE
  -nc, --no-clobber                 skip downloads that would download to
                                      existing files (overwriting them)
       --no-netrc                   don't try to obtain credentials from .netrc
  -c,  --continue                   resume getting a partially-downloaded file
       --start-pos=OFFSET           start downloading from zero-based position OFFSET
       --progress=TYPE              select progress gauge type
       --show-progress              display the progress bar in any verbosity mode
  -N,  --timestamping               don't re-retrieve files unless newer than
                                      local
       --no-if-modified-since       don't use conditional if-modified-since get
                                      requests in timestamping mode
       --no-use-server-timestamps   don't set the local file's timestamp by
                                      the one on the server
  -S,  --server-response            print server response
       --spider                     don't download anything
  -T,  --timeout=SECONDS            set all timeout values to SECONDS
       --dns-timeout=SECS           set the DNS lookup timeout to SECS
       --connect-timeout=SECS       set the connect timeout to SECS
       --read-timeout=SECS          set the read timeout to SECS
  -w,  --wait=SECONDS               wait SECONDS between retrievals
                                      (applies if more then 1 URL is to be retrieved)
       --waitretry=SECONDS          wait 1..SECONDS between retries of a retrieval
                                      (applies if more then 1 URL is to be retrieved)
       --random-wait                wait from 0.5*WAIT...1.5*WAIT secs between retrievals
                                      (applies if more then 1 URL is to be retrieved)
```

```
       --no-proxy                  explicitly turn off proxy
  -Q,  --quota=NUMBER              set retrieval quota to NUMBER
       --bind-address=ADDRESS      bind to ADDRESS (hostname or IP) on local host
       --limit-rate=RATE           limit download rate to RATE
       --no-dns-cache              disable caching DNS lookups
       --restrict-file-names=OS    restrict chars in file names to ones OS allows
       --ignore-case               ignore case when matching files/directories
  -4,  --inet4-only                connect only to IPv4 addresses
  -6,  --inet6-only                connect only to IPv6 addresses
       --prefer-family=FAMILY      connect first to addresses of specified family,
                                     one of IPv6, IPv4, or none
       --user=USER                 set both ftp and http user to USER
       --password=PASS             set both ftp and http password to PASS
       --ask-password              prompt for passwords
       --use-askpass=COMMAND       specify credential handler for requesting
                                     username and password.  If no COMMAND is
                                     specified the WGET_ASKPASS or the SSH_ASKPASS
                                     environment variable is used.
       --no-iri                    turn off IRI support
       --local-encoding=ENC        use ENC as the local encoding for IRIs
       --remote-encoding=ENC       use ENC as the default remote encoding
       --unlink                    remove file before clobber
       --xattr                     turn on storage of metadata in extended file attributes

Directories:
  -nd, --no-directories            don't create directories
  -x,  --force-directories         force creation of directories
  -nH, --no-host-directories       don't create host directories
       --protocol-directories      use protocol name in directories
  -P,  --directory-prefix=PREFIX   save files to PREFIX/..
       --cut-dirs=NUMBER           ignore NUMBER remote directory components

HTTP options:
       --http-user=USER            set http user to USER
```

```
       --http-password=PASS        set http password to PASS
       --no-cache                  disallow server-cached data
       --default-page=NAME         change the default page name (normally
                                     this is 'index.html'.)
  -E,  --adjust-extension          save HTML/CSS documents with proper extensions
       --ignore-length             ignore 'Content-Length' header field
       --header=STRING             insert STRING among the headers
       --compression=TYPE          choose compression, one of auto, gzip and none. (default: none)
       --max-redirect              maximum redirections allowed per page
       --proxy-user=USER           set USER as proxy username
       --proxy-password=PASS       set PASS as proxy password
       --referer=URL               include 'Referer: URL' header in HTTP request
       --save-headers              save the HTTP headers to file
  -U,  --user-agent=AGENT          identify as AGENT instead of Wget/VERSION
       --no-http-keep-alive        disable HTTP keep-alive (persistent connections)
       --no-cookies                don't use cookies
       --load-cookies=FILE         load cookies from FILE before session
       --save-cookies=FILE         save cookies to FILE after session
       --keep-session-cookies      load and save session (non-permanent) cookies
       --post-data=STRING          use the POST method; send STRING as the data
       --post-file=FILE            use the POST method; send contents of FILE
       --method=HTTPMethod         use method "HTTPMethod" in the request
       --body-data=STRING          send STRING as data. --method MUST be set
       --body-file=FILE            send contents of FILE. --method MUST be set
       --content-disposition       honor the Content-Disposition header when
                                     choosing local file names (EXPERIMENTAL)
       --content-on-error          output the received content on server errors
       --auth-no-challenge         send Basic HTTP authentication information
                                     without first waiting for the server's
                                     challenge

HTTPS (SSL/TLS) options:
       --secure-protocol=PR        choose secure protocol, one of auto, SSLv2,
                                     SSLv3, TLSv1, TLSv1_1, TLSv1_2 and PFS
```

```
       --https-only               only follow secure HTTPS links
       --no-check-certificate     don't validate the server's certificate
       --certificate=FILE         client certificate file
       --certificate-type=TYPE    client certificate type, PEM or DER
       --private-key=FILE         private key file
       --private-key-type=TYPE    private key type, PEM or DER
       --ca-certificate=FILE      file with the bundle of CAs
       --ca-directory=DIR         directory where hash list of CAs is stored
       --crl-file=FILE            file with bundle of CRLs
       --pinnedpubkey=FILE/HASHES  Public key (PEM/DER) file, or any number
                                  of base64 encoded sha256 hashes preceded by
                                  'sha256//' and separated by ';', to verify
                                  peer against

       --ciphers=STR           Set the priority string (GnuTLS) or cipher list string (OpenSSL) directly.
                                  Use with care. This option overrides --secure-protocol.
                                  The format and syntax of this string depend on the specific SSL/TLS engine.
HSTS options:
       --no-hsts                  disable HSTS
       --hsts-file                path of HSTS database (will override default)

FTP options:
       --ftp-user=USER            set ftp user to USER
       --ftp-password=PASS        set ftp password to PASS
       --no-remove-listing        don't remove '.listing' files
       --no-glob                  turn off FTP file name globbing
       --no-passive-ftp           disable the "passive" transfer mode
       --preserve-permissions     preserve remote file permissions
       --retr-symlinks            when recursing, get linked-to files (not dir)

FTPS options:
       --ftps-implicit               use implicit FTPS (default port is 990)
       --ftps-resume-ssl             resume the SSL/TLS session started in the control connection when
                                       opening a data connection
```

```
       --ftps-clear-data-connection    cipher the control channel only; all the data will be in plaintext
       --ftps-fallback-to-ftp          fall back to FTP if FTPS is not supported in the target server
WARC options:
       --warc-file=FILENAME        save request/response data to a .warc.gz file
       --warc-header=STRING        insert STRING into the warcinfo record
       --warc-max-size=NUMBER      set maximum size of WARC files to NUMBER
       --warc-cdx                  write CDX index files
       --warc-dedup=FILENAME       do not store records listed in this CDX file
       --no-warc-compression       do not compress WARC files with GZIP
       --no-warc-digests           do not calculate SHA1 digests
       --no-warc-keep-log          do not store the log file in a WARC record
       --warc-tempdir=DIRECTORY    location for temporary files created by the
                                     WARC writer

Recursive download:
  -r,  --recursive                 specify recursive download
  -l,  --level=NUMBER              maximum recursion depth (inf or 0 for infinite)
       --delete-after              delete files locally after downloading them
  -k,  --convert-links             make links in downloaded HTML or CSS point to
                                     local files
       --convert-file-only         convert the file part of the URLs only (usually known as the basename)
       --backups=N                 before writing file X, rotate up to N backup files
  -K,  --backup-converted          before converting file X, back up as X.orig
  -m,  --mirror                    shortcut for -N -r -l inf --no-remove-listing
  -p,  --page-requisites           get all images, etc. needed to display HTML page
       --strict-comments           turn on strict (SGML) handling of HTML comments

Recursive accept/reject:
  -A,  --accept=LIST               comma-separated list of accepted extensions
  -R,  --reject=LIST               comma-separated list of rejected extensions
       --accept-regex=REGEX        regex matching accepted URLs
       --reject-regex=REGEX        regex matching rejected URLs
       --regex-type=TYPE           regex type (posix|pcre)
  -D,  --domains=LIST              comma-separated list of accepted domains
```

```
       --exclude-domains=LIST      comma-separated list of rejected domains
       --follow-ftp                follow FTP links from HTML documents
       --follow-tags=LIST          comma-separated list of followed HTML tags
       --ignore-tags=LIST          comma-separated list of ignored HTML tags
  -H,  --span-hosts                go to foreign hosts when recursive
  -L,  --relative                  follow relative links only
  -I,  --include-directories=LIST  list of allowed directories
       --trust-server-names        use the name specified by the redirection
                                     URL's last component
  -X,  --exclude-directories=LIST  list of excluded directories
  -np, --no-parent                 don't ascend to the parent directory

Email bug reports, questions, discussions to <bug-wget@gnu.org>
and/or open issues at https://savannah.gnu.org/bugs/?func=additem&group=wget.
```

3.3 - ftp

Important - Si la commande **ftp** n'est pas installée sous RedHat 9, installez-le à l'aide de la commande **dnf install ftp** en tant que root.

La commande **ftp** est utilisée pour le transfert de fichiers. Une fois connecté, il convient d'utiliser la commande **help** pour afficher la liste des commandes disponibles :

```
[root@redhat9 ~]# ftp
ftp> help
Commands may be abbreviated.  Commands are:

!               debug           mdir            sendport        site
$               dir             mget            put             size
account         disconnect      mkdir           pwd             status
append          exit            mls             quit            struct
```

```
ascii           form            mode            quote           system
bell            get             modtime         recv            sunique
binary          glob            mput            reget           tenex
bye             hash            newer           rstatus         tick
case            help            nmap            rhelp           trace
cd              idle            nlist           rename          type
cdup            image           ntrans          reset           user
chmod           lcd             open            restart         umask
close           ls              prompt          rmdir           verbose
cr              macdef          passive         runique         ?
delete          mdelete         proxy           send
ftp>
```

Le caractère **!** permet d'exécuter une commande sur la machine cliente

```
ftp> !pwd
/root
```

Pour transférer un fichier vers le serveur, il convient d'utiliser la commande **put** :

```
ftp> put nom_fichier_local nom_fichier_distant
```

Vous pouvez également transférer plusieurs fichiers à la fois grâce à la commande **mput**. Dans ce cas précis, il convient de saisir la commande suivante:

```
ftp> mput nom*.*
```

Pour transférer un fichier du serveur, il convient d'utiliser la commande **get** :

```
ftp> get nom_fichier
```

Vous pouvez également transférer plusieurs fichiers à la fois grâce à la commande **mget** (voir la commande **mput** ci-dessus).

Pour supprimer un fichier sur le serveur, il convient d'utiliser la commande **del** :

```
ftp> del nom_fichier
```

Pour fermer la session, il convient d'utiliser la commande **quit** :

```
ftp> quit
[root@redhat9 ~]#
```

3.4 - SSH

Présentation

La commande **ssh** est le successeur et la remplaçante de la commande **rlogin**. Il permet d'établir des connexions sécurisées avec une machine distante. SSH comporte cinq acteurs :

- Le **serveur SSH**
  - le démon sshd, qui s'occupe des authentifications et autorisations des clients,
- Le **client SSH**
  - ssh ou scp, qui assure la connexion et le dialogue avec le serveur,
- La **session** qui représente la connexion courante et qui commence juste après l'authentification réussie,
- Les **clefs**
  - **Couple de clef utilisateur asymétriques** et persistantes qui assurent l'identité d'un utilisateur et qui sont stockés sur disque dur,
  - **Clef hôte asymétrique et persistante** garantissant l'identité du serveur er qui est conservé sur disque dur
  - **Clef serveur asymétrique et temporaire** utilisée par le protocole SSH1 qui sert au chiffrement de la clé de session,
  - **Clef de session symétrique qui est générée aléatoirement** et qui permet le chiiffrement de la communication entre le client et le serveur. Elle est détruite en fin de session. SSH-1 utilise une seule clef tandis que SSH-2 utilise une clef par direction de la communication,
- La **base de données des hôtes connus** qui stocke les clés des connexions précédentes.

SSH fonctionne de la manière suivante pour la la mise en place d'un canal sécurisé:

- Le client contacte le serveur sur son port 22,
- Les client et le serveur échangent leur version de SSH. En cas de non-compatibilité de versions, l'un des deux met fin au processus,
- Le serveur SSH s'identifie auprès du client en lui fournissant :

    - Sa clé hôte,
    - Sa clé serveur,
    - Une séquence aléatoire de huit octets à inclure dans les futures réponses du client,
    - Une liste de méthodes de chiffrage, compression et authentification,
- Le client et le serveur produisent un identifiant identique, un haché MD5 long de 128 bits contenant la clé hôte, la clé serveur et la séquence aléatoire,
- Le client génère sa clé de session symétrique et la chiffre deux fois de suite, une fois avec la clé hôte du serveur et la deuxième fois avec la clé serveur. Le client envoie cette clé au serveur accompagnée de la séquence aléatoire et un choix d'algorithmes supportés,
- Le serveur déchiffre la clé de session,
- Le client et le serveur mettent en place le canal sécurisé.

SSH-1

SSH-1 utilise une paire de clefs de type RSA1. Il assure l'intégrité des données par une **Contrôle de Redondance Cyclique** (CRC) et est un bloc dit **monolithique**.

Afin de s'identifier, le client essaie chacune des six méthodes suivantes :

- **Kerberos**,
- **Rhosts**,
- **RhostsRSA**,
- Par **clef asymétrique**,
- **TIS**,
- Par **mot de passe**.

SSH-2

SSH-2 utilise **DSA** ou **RSA**. Il assure l'intégrité des données par l'algorithme **HMAC**. SSH-2 est organisé en trois **couches** :

- **SSH-TRANS** – Transport Layer Protocol,
- **SSH-AUTH** – Authentification Protocol,
- **SSH-CONN** – Connection Protocol.

SSH-2 diffère de SSH-1 essentiellement dans la phase authentification.

Trois méthodes d'authentification :

- Par **clef asymétrique**,
  - Identique à SSH-1 sauf avec l'algorithme DSA,
- **RhostsRSA**,
- Par **mot de passe**.

Options de la commande

Les options de cette commande sont :

```
[root@redhat9 ~]# ssh --help
unknown option -- -
usage: ssh [-46AaCfGgKkMNnqsTtVvXxYy] [-B bind_interface]
           [-b bind_address] [-c cipher_spec] [-D [bind_address:]port]
           [-E log_file] [-e escape_char] [-F configfile] [-I pkcs11]
           [-i identity_file] [-J [user@]host[:port]] [-L address]
           [-l login_name] [-m mac_spec] [-O ctl_cmd] [-o option] [-p port]
           [-Q query_option] [-R address] [-S ctl_path] [-W host:port]
           [-w local_tun[:remote_tun]] destination [command]
```

Authentification par mot de passe

L'utilisateur fournit un mot de passe au client ssh. Le client ssh le transmet de façon sécurisée au serveur ssh puis le serveur vérifie le mot de passe et l'accepte ou non.

Avantage:

- Aucune configuration de clef asymétrique n'est nécessaire.

Inconvénients:

- L'utilisateur doit fournir à chaque connexion un identifiant et un mot de passe,
- Moins sécurisé qu'un système par clef asymétrique.

Authentification par clef asymétrique

- Le **client** envoie au serveur une requête d'authentification par clé asymétrique qui contient le module de la clé à utiliser,
- Le **serveur** recherche une correspondance pour ce module dans le fichier des clés autorisés **~/.ssh/authorized_keys**,
  - Dans le cas où une correspondance n'est pas trouvée, le serveur met fin à la communication,
  - Dans le cas contraire le serveur génère une chaîne aléatoire de 256 bits appelée un **challenge** et la chiffre avec la **clé publique du client**,
- Le **client** reçoit le challenge et le décrypte avec la partie privée de sa clé. Il combine le challenge avec l'identifiant de session et chiffre le résultat. Ensuite il envoie le résultat chiffré au serveur.
- Le **serveur** génère le même haché et le compare avec celui reçu du client. Si les deux hachés sont identiques, l'authentification est réussie.

Configuration du Serveur

La configuration du serveur s'effectue dans le fichier **/etc/ssh/sshd_config** :

```
[root@redhat9 ~]# cat /etc/ssh/sshd_config
#       $OpenBSD: sshd_config,v 1.104 2021/07/02 05:11:21 dtucker Exp $

# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.
```

```
# To modify the system-wide sshd configuration, create a  *.conf  file under
#  /etc/ssh/sshd_config.d/  which will be automatically included below
Include /etc/ssh/sshd_config.d/*.conf

# If you want to change the port on a SELinux system, you have to tell
# SELinux about this change.
# semanage port -a -t ssh_port_t -p tcp #PORTNUMBER
#
#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
#PermitRootLogin prohibit-password
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes
```

```
# The default is to check both .ssh/authorized_keys and .ssh/authorized_keys2
# but this is overridden so installations will only check .ssh/authorized_keys
AuthorizedKeysFile      .ssh/authorized_keys

#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
#PermitEmptyPasswords no

# Change to no to disable s/key passwords
#KbdInteractiveAuthentication yes

# Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes
#KerberosGetAFSToken no
#KerberosUseKuserok yes

# GSSAPI options
#GSSAPIAuthentication no
#GSSAPICleanupCredentials yes
```

```
#GSSAPIStrictAcceptorCheck yes
#GSSAPIKeyExchange no
#GSSAPIEnablek5users no

# Set this to 'yes' to enable PAM authentication, account processing,
# and session processing. If this is enabled, PAM authentication will
# be allowed through the KbdInteractiveAuthentication and
# PasswordAuthentication.  Depending on your PAM configuration,
# PAM authentication via KbdInteractiveAuthentication may bypass
# the setting of "PermitRootLogin without-password".
# If you just want the PAM account and session checks to run without
# PAM authentication, then enable this but set PasswordAuthentication
# and KbdInteractiveAuthentication to 'no'.
# WARNING: 'UsePAM no' is not supported in RHEL and may cause several
# problems.
#UsePAM no

#AllowAgentForwarding yes
#AllowTcpForwarding yes
#GatewayPorts no
#X11Forwarding no
#X11DisplayOffset 10
#X11UseLocalhost yes
#PermitTTY yes
#PrintMotd yes
#PrintLastLog yes
#TCPKeepAlive yes
#PermitUserEnvironment no
#Compression delayed
#ClientAliveInterval 0
#ClientAliveCountMax 3
#UseDNS no
#PidFile /var/run/sshd.pid
#MaxStartups 10:30:100
```

```
#PermitTunnel no
#ChrootDirectory none
#VersionAddendum none

# no default banner path
#Banner none

# override default of no subsystems
Subsystem       sftp    /usr/libexec/openssh/sftp-server

# Example of overriding settings on a per-user basis
#Match User anoncvs
#       X11Forwarding no
#       AllowTcpForwarding no
#       PermitTTY no
#       ForceCommand cvs server
```

Pour ôter les lignes de commentaires dans ce fichier, utilisez la commande suivante :

```
[root@redhat9 ~]# cd /tmp ; grep -E -v '^(#|$)'  /etc/ssh/sshd_config > sshd_config

[root@redhat9 tmp]# cat sshd_config
Include /etc/ssh/sshd_config.d/*.conf
AuthorizedKeysFile      .ssh/authorized_keys
Subsystem       sftp    /usr/libexec/openssh/sftp-server
```

Pour sécuriser le serveur ssh, ajoutez ou modifiez les directives suivantes :

```
AllowGroups adm
Banner /etc/issue.net
HostbasedAuthentication no
IgnoreRhosts yes
LoginGraceTime 60
LogLevel INFO
```

```
PermitEmptyPasswords no
PermitRootLogin no
PrintLastLog yes
Protocol 2
StrictModes yes
X11Forwarding no
```

Votre fichier ressemblera à celui-ci :

```
[root@redhat9 tmp]# vi sshd_config

[root@redhat9 tmp]# cat sshd_config
AllowGroups adm
Banner /etc/issue.net
HostbasedAuthentication no
IgnoreRhosts yes
LoginGraceTime 60
LogLevel INFO
PermitEmptyPasswords no
PermitRootLogin no
PrintLastLog yes
Protocol 2
StrictModes yes
X11Forwarding no
Include /etc/ssh/sshd_config.d/*.conf
AuthorizedKeysFile      .ssh/authorized_keys
Subsystem       sftp    /usr/libexec/openssh/sftp-server
```

Renommez le fichier **/etc/ssh/sshd_config** en **/etc/ssh/sshd_config.old** :

```
[root@redhat9 tmp]# cp /etc/ssh/sshd_config /etc/ssh/sshd_config.old
```

Copiez le fichier **/tmp/sshd_config** vers **/etc/ssh/** :

```
[root@redhat9 tmp]# cp /tmp/sshd_config /etc/ssh
cp: overwrite '/etc/ssh/sshd_config'? y
```

Redémarrez le service sshd :

```
[root@redhat9 tmp]# systemctl restart sshd

[root@redhat9 tmp]# systemctl status sshd
● sshd.service - OpenSSH server daemon
     Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; preset: enabled)
     Active: active (running) since Sun 2024-09-29 14:06:49 CEST; 9s ago
       Docs: man:sshd(8)
             man:sshd_config(5)
   Main PID: 5560 (sshd)
      Tasks: 1 (limit: 48800)
     Memory: 1.4M
        CPU: 13ms
     CGroup: /system.slice/sshd.service
             └─5560 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Sep 29 14:06:49 redhat9.ittraining.loc systemd[1]: Starting OpenSSH server daemon...
Sep 29 14:06:49 redhat9.ittraining.loc sshd[5560]: Server listening on 0.0.0.0 port 22.
Sep 29 14:06:49 redhat9.ittraining.loc sshd[5560]: Server listening on :: port 22.
Sep 29 14:06:49 redhat9.ittraining.loc systemd[1]: Started OpenSSH server daemon.
```

Mettez l'utilisateur **trainee** dans le groupe **adm** :

```
[root@redhat9 tmp]# groups trainee
trainee : trainee

[root@redhat9 tmp]# usermod -aG adm trainee

[root@redhat9 tmp]# groups trainee
```

```
trainee : trainee adm
```

Pour générer les clefs du serveur, saisissez la commande suivante en tant que **root**. Notez que la passphrase doit être **vide**.

```
[root@redhat9 tmp]# ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/root/.ssh/id_dsa): /etc/ssh/ssh_host_dsa_key
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /etc/ssh/ssh_host_dsa_key
Your public key has been saved in /etc/ssh/ssh_host_dsa_key.pub
The key fingerprint is:
SHA256:VWOX1JHwvNZHo8kC6OLgKXypZI97W56JBiKwc9YjFpk root@redhat9.ittraining.loc
The key's randomart image is:
+---[DSA 1024]----+
|            +o++o|
|         .  o o+..|
|   o   . ..    +.|
|. E   .  .. . o.+|
|.. + . .S  . + oo|
|*.* B .     . . .|
|oX.* +           |
|o *o+ o          |
| +++.+           |
+----[SHA256]-----+
```

De la même façon, il est possible de générer les clefs au format **RSA**, **ECDSA** et **ED25519** :

```
[root@redhat9 tmp]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): /etc/ssh/ssh_host_rsa_key
/etc/ssh/ssh_host_rsa_key already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
```

```
Enter same passphrase again:
Your identification has been saved in /etc/ssh/ssh_host_rsa_key
Your public key has been saved in /etc/ssh/ssh_host_rsa_key.pub
The key fingerprint is:
SHA256:SjQBVxLP5mtWmsHUWN0j1FvgokkEmKbEftzWc1ZlrAM root@redhat9.ittraining.loc
The key's randomart image is:
+---[RSA 3072]----+
|   ...==o.oo.o.+o|
|    o.+= =  E.=.o|
|   o +o.*.o .+.= |
|    o.o*o.ooo.+  |
|     ...S ++   . |
|     . . *       |
|      . *        |
|       o         |
|                 |
+----[SHA256]-----+

[root@redhat9 tmp]# ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/root/.ssh/id_ecdsa): /etc/ssh/ssh_host_ecdsa_key
/etc/ssh/ssh_host_ecdsa_key already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /etc/ssh/ssh_host_ecdsa_key
Your public key has been saved in /etc/ssh/ssh_host_ecdsa_key.pub
The key fingerprint is:
SHA256:auaDgRcvbqpC5SkGadt7AizXVta2A8w5hYwVgUPXNlc root@redhat9.ittraining.loc
The key's randomart image is:
+---[ECDSA 256]---+
|   ..===   .E    |
|    +.o = .      |
| .   + = o       |
```

```
|o. .. O o        |
|+ =o.= +S.       |
|.Bo+* ..o        |
|+.o+.++  .       |
|.  o+=.          |
|o..oo ..         |
+----[SHA256]-----+

[root@redhat9 tmp]# ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519): /etc/ssh/ssh_host_ed25519_key
/etc/ssh/ssh_host_ed25519_key already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /etc/ssh/ssh_host_ed25519_key
Your public key has been saved in /etc/ssh/ssh_host_ed25519_key.pub
The key fingerprint is:
SHA256:UwUjA5Ln700GWzhNxvIvdNafLOhD1/hdrMigOvARQwI root@redhat9.ittraining.loc
The key's randomart image is:
+--[ED25519 256]--+
|  E.....o.o..    |
|    o.o .o+o     |
|     =   B.  .   |
|      + +.= o .  |
|       +S* +. +o.|
|    . . o.=o.+ ++|
|     o o =o+..o.o|
|      o o .oo ...|
|      .o    .    |
+----[SHA256]-----+
```

Les clefs publiques générées possèdent l'extension **.pub**. Les clefs privées n'ont pas d'extension :

```
[root@redhat9 tmp]# ls /etc/ssh
moduli      ssh_config.d  sshd_config.d    ssh_host_dsa_key      ssh_host_ecdsa_key      ssh_host_ed25519_key
ssh_host_rsa_key
ssh_config  sshd_config   sshd_config.old  ssh_host_dsa_key.pub  ssh_host_ecdsa_key.pub  ssh_host_ed25519_key.pub
ssh_host_rsa_key.pub
```

Re-démarrez ensuite le service sshd :

```
[root@redhat9 tmp]# systemctl restart sshd.service

[root@redhat9 tmp]# systemctl status sshd.service
● sshd.service - OpenSSH server daemon
     Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; preset: enabled)
     Active: active (running) since Sun 2024-09-29 14:14:14 CEST; 13s ago
       Docs: man:sshd(8)
             man:sshd_config(5)
   Main PID: 5583 (sshd)
      Tasks: 1 (limit: 48800)
     Memory: 1.3M
        CPU: 12ms
     CGroup: /system.slice/sshd.service
             └─5583 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Sep 29 14:14:14 redhat9.ittraining.loc systemd[1]: sshd.service: Deactivated successfully.
Sep 29 14:14:14 redhat9.ittraining.loc sshd[5583]: Server listening on 0.0.0.0 port 22.
Sep 29 14:14:14 redhat9.ittraining.loc systemd[1]: Stopped OpenSSH server daemon.
Sep 29 14:14:14 redhat9.ittraining.loc sshd[5583]: Server listening on :: port 22.
Sep 29 14:14:14 redhat9.ittraining.loc systemd[1]: Starting OpenSSH server daemon...
Sep 29 14:14:14 redhat9.ittraining.loc systemd[1]: Started OpenSSH server daemon.
```

Configuration du Client

Saisissez maintenant les commandes suivantes en tant que **trainee** :

Important - Lors de la génération des clefs, la passphrase doit être **vide**.

```
[root@redhat9 tmp]# exit
logout

[trainee@redhat9 ~]$ ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/trainee/.ssh/id_dsa):
Created directory '/home/trainee/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/trainee/.ssh/id_dsa
Your public key has been saved in /home/trainee/.ssh/id_dsa.pub
The key fingerprint is:
SHA256:TT4VSKqep7i/5t6FoPSQ88LmTo3eLRBBFxz3Cz2Kxi4 trainee@redhat9.ittraining.loc
The key's randomart image is:
+---[DSA 1024]----+
|   ...+o.....    |
|    .... +.  .   |
|     .  o = .    |
|    .o o * +     |
|    =.* S =      |
|   o.% o . .     |
|    E.B o .      |
|   = =o= .       |
|   .**Boo        |
+----[SHA256]-----+

[trainee@redhat9 ~]$ ssh-keygen -t rsa
```

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/trainee/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/trainee/.ssh/id_rsa
Your public key has been saved in /home/trainee/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:75jUyGd9zw6fA5Z2KIZAacp+0b5KLeWpOONItt/4Z9k trainee@redhat9.ittraining.loc
The key's randomart image is:
+---[RSA 3072]----+
|        .        |
|       +         |
|    . + .        |
|     o o .       |
|    .   S..   o  |
|     . o+*.+ * . |
|   o  .o++X = =  |
|  o oo=.o@ E . *.|
|   oo=+=* .    o*|
+----[SHA256]-----+

[trainee@redhat9 ~]$ ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/trainee/.ssh/id_ecdsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/trainee/.ssh/id_ecdsa
Your public key has been saved in /home/trainee/.ssh/id_ecdsa.pub
The key fingerprint is:
SHA256:xjl9G3ycaF3s9cuyCqHhPvUWyN1qzBGxJtZwFAFeRvw trainee@redhat9.ittraining.loc
The key's randomart image is:
+---[ECDSA 256]---+
|         .=Bo    |
|        ..o+   . |
```

```
|         .+ +   +|
|       . = * E +o|
|       .S.* O = o|
|      ..o=.* =. .|
|       o..+ =. o |
|      ..  .B  o  |
|       .. o...   |
+----[SHA256]-----+

[trainee@redhat9 ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/trainee/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/trainee/.ssh/id_ed25519
Your public key has been saved in /home/trainee/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:qtFzx7O0swNGsRoP9Cy+cISURCvWYpANMn1WtljZ8WQ trainee@redhat9.ittraining.loc
The key's randomart image is:
+--[ED25519 256]--+
|== ooo+o..E      |
|ooo.==o.o+       |
|  =++o.o o.      |
| o o. = =        |
|     o BS        |
|    ..+.+.       |
|    .o+o..=      |
|     o.o oo+     |
|    .     ++     |
+----[SHA256]-----+
```

Les clés générées seront placées dans le répertoire **~/.ssh/** :

```
[trainee@redhat9 ~]$ ls .ssh
```

```
id_dsa  id_dsa.pub  id_ecdsa  id_ecdsa.pub  id_ed25519  id_ed25519.pub  id_rsa  id_rsa.pub
```

Tunnels SSH

Le protocole SSH peut être utilisé pour sécuriser les protocoles tels telnet, pop3 etc.. En effet, on peut créer un *tunnel* SSH dans lequel passe les communications du protocole non-sécurisé.

La commande pour créer un tunnel ssh prend la forme suivante :

```
ssh -N -f compte@hôte -Lport-local:localhost:port_distant
```

Dans votre cas, vous allez créer un tunnel dans votre propre vm entre le port 15023 et le port 23 :

```
[trainee@redhat9 ~]$ ssh -N -f trainee@localhost -L15023:localhost:23
The authenticity of host 'localhost (::1)' can't be established.
ED25519 key fingerprint is SHA256:UwUjA5Ln700GWzhNxvIvdNafLOhD1/hdrMigOvARQwI.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.
\S
Kernel \r on an \m
trainee@localhost's password: trainee
```

Installez maintenant le serveur telnet :

```
[trainee@redhat9 ~]$ su -
Password:
[root@redhat9 ~]# dnf install telnet-server
Updating Subscription Management repositories.
Last metadata expiration check: 3:26:47 ago on Sun 29 Sep 2024 10:52:14 AM CEST.
Dependencies resolved.
================================================================================
```

```
================================================================================
 Package                    Architecture                Version
Repository                                          Size
================================================================================
================================================================================
Installing:
 telnet-server              x86_64                      1:0.17-85.el9
rhel-9-for-x86_64-appstream-rpms                    41 k

Transaction Summary
================================================================================
================================================================================
Install  1 Package

Total download size: 41 k
Installed size: 58 k
Is this ok [y/N]: y
Downloading Packages:
telnet-server-0.17-85.el9.x86_64.rpm
145 kB/s |  41 kB     00:00
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Total
144 kB/s |  41 kB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :
1/1
  Installing       : telnet-server-1:0.17-85.el9.x86_64
1/1
  Running scriptlet: telnet-server-1:0.17-85.el9.x86_64
```

```
1/1
  Verifying        : telnet-server-1:0.17-85.el9.x86_64
1/1
Installed products updated.

Installed:
  telnet-server-1:0.17-85.el9.x86_64

Complete!
```

Telnet n'est ni démarré ni activé. Il convient donc de le démarrer et de l'activer :

```
[root@redhat9 ~]# systemctl status telnet.socket
○ telnet.socket - Telnet Server Activation Socket
     Loaded: loaded (/usr/lib/systemd/system/telnet.socket; disabled; preset: disabled)
     Active: inactive (dead)
       Docs: man:telnetd(8)
     Listen: [::]:23 (Stream)
   Accepted: 0; Connected: 0;

[root@redhat9 ~]# systemctl start telnet.socket

[root@redhat9 ~]# systemctl status telnet.socket
● telnet.socket - Telnet Server Activation Socket
     Loaded: loaded (/usr/lib/systemd/system/telnet.socket; disabled; preset: disabled)
     Active: active (listening) since Sun 2024-09-29 14:19:51 CEST; 13s ago
      Until: Sun 2024-09-29 14:19:51 CEST; 13s ago
       Docs: man:telnetd(8)
     Listen: [::]:23 (Stream)
   Accepted: 0; Connected: 0;
      Tasks: 0 (limit: 48800)
     Memory: 8.0K
        CPU: 700us
     CGroup: /system.slice/telnet.socket
```

```
Sep 29 14:19:51 redhat9.ittraining.loc systemd[1]: Listening on Telnet Server Activation Socket.
[root@redhat9 ~]# systemctl enable telnet.socket
Created symlink /etc/systemd/system/sockets.target.wants/telnet.socket → /usr/lib/systemd/system/telnet.socket.
```

Connectez-vous ensuite via telnet sur le port 15023, vous constaterez que votre connexion n'aboutit pas :

```
[root@redhat9 ~]# telnet localhost 15023
Trying ::1...
Connected to localhost.
Escape character is '^]'.

Kernel 5.14.0-427.37.1.el9_4.x86_64 on an x86_64
redhat9 login: trainee
Password: trainee
Last login: Sun Sep 29 12:26:00 from 10.0.2.1

[trainee@redhat9 ~]$ whoami
trainee

[trainee@redhat9 ~]$ pwd
/home/trainee
```

Important - Notez bien que votre communication telnet passe par le tunnel SSH.

3.5 - SCP

Présentation

La commande **scp** est le successeur et la remplaçante de la commande **rcp** de la famille des commandes **remote**. Il permet de faire des transferts

sécurisés à partir d'une machine distante :

```
$ scp compte@numero_ip(nom_de_machine):/chemin_distant/fichier_distant /chemin_local/fichier_local
```

ou vers une machine distante :

```
$ scp /chemin_local/fichier_local compte@numero_ip(nom_de_machine):/chemin_distant/fichier_distant
```

Utilisation

Nous allons maintenant utiliser **scp** pour chercher un fichier sur le «serveur» :

Créez le fichier **/home/trainee/scp_test** :

```
[trainee@redhat9 ~]$ touch scp-test

[trainee@redhat9 ~]$ exit
logout
Connection closed by foreign host.

[root@redhat9 ~]#
```

Récupérez le fichier **scp_test** en utilisant scp :

```
[root@redhat9 ~]# scp trainee@127.0.0.1:/home/trainee/scp-test .
The authenticity of host '127.0.0.1 (127.0.0.1)' can't be established.
ED25519 key fingerprint is SHA256:UwUjA5Ln700GWzhNxvIvdNafLOhD1/hdrMigOvARQwI.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '127.0.0.1' (ED25519) to the list of known hosts.
\S
Kernel \r on an \m
```

```
trainee@127.0.0.1's password: trainee

[root@redhat9 ~]# ls -l
total 2042944
-rw-------. 1 root    root          1226 Oct 19  2023  anaconda-ks.cfg
-rw-r--r--. 1 trainee trainee 2091941797 Oct 19  2023  ansible-automation-platform-setup-bundle-2.4-2.2-
x86_64.tar.gz
-rw-r--r--. 1 root    root            64 Sep 27 08:24  device.map
-rw-------. 1 root    root          7118 Sep 27 08:24  grub.cfg
drwxr-xr-x. 3 root    root            21 Oct 19  2023  home
-rw-r--r--. 1 root    root            98 Sep 27 08:23  montages.list
-rw-r--r--. 1 root    root          2109 Sep 25 16:20  passwd
-rw-r--r--. 1 root    root             0 Sep 29 14:24  scp-test
-rw-r--r--. 1 root    root           457 Sep 27 08:22  structure.list
-rw-r--r--. 1 root    root            46 Sep 29 13:56 'wget_file.txt?rlkey=g8fgje9z8oeqgb4nd2g7x3wkx'
```

3.6 - Mise en Place des Clefs Asymétriques

Il convient maintenant de se connecter sur le «serveur» en utilisant ssh et vérifiez la présence du répertoire ~/.ssh :

```
[root@redhat9 ~]# ssh -l trainee 127.0.0.1
\S
Kernel \r on an \m
trainee@127.0.0.1's password: trainee
Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-dashboard
Last login: Sun Sep 29 14:21:21 2024 from localhost

[trainee@redhat9 ~]$ ls -la | grep .ssh
-rw-------.  1 trainee trainee   20 Sep 25 15:18 .lesshst
drwx------.  2 trainee trainee  188 Sep 29 14:18 .ssh
```

Important - Si le dossier distant .ssh n'existe pas dans le répertoire personnel de l'utilisateur connecté, il faut le créer avec des permissions de 700. Dans votre cas, puisque votre machine joue le rôle de serveur **et** du client, le dossier /home/trainee/.ssh existe **déjà**.

Ensuite, il convient de transférer le fichier local **.ssh/id_ecdsa.pub** du «client» vers le «serveur» en le renommant en **authorized_keys** :

```
[trainee@redhat9 ~]$ exit
logout
Connection to 127.0.0.1 closed.

[root@redhat9 ~]# exit
logout

[trainee@redhat9 ~]$ scp .ssh/id_ecdsa.pub trainee@127.0.0.1:/home/trainee/.ssh/authorized_keys
The authenticity of host '127.0.0.1 (127.0.0.1)' can't be established.
ED25519 key fingerprint is SHA256:UwUjA5Ln700GWzhNxvIvdNafLOhD1/hdrMigOvARQwI.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:1: localhost
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '127.0.0.1' (ED25519) to the list of known hosts.
\S
Kernel \r on an \m
trainee@127.0.0.1's password: trainee
id_ecdsa.pub
100%  192   427.3KB/s   00:00
```

Connectez-vous via ssh :

```
[trainee@redhat9 ~]$ ssh -l trainee localhost
\S
```

```
Kernel \r on an \m
Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-dashboard
Last login: Sun Sep 29 14:26:20 2024 from 127.0.0.1
[trainee@redhat9 ~]$
```

Important - Lors de la connexion au serveur, l'authentification utilise le couple de clefs asymétrique au format ecdsa et aucun mot de passe n'est requis.

Insérez maintenant les clefs publiques restantes dans le fichier .ssh/authorized_keys :

```
[trainee@redhat9 ~]$ cd .ssh

[trainee@redhat9 .ssh]$ ls
authorized_keys  id_dsa  id_dsa.pub  id_ecdsa  id_ecdsa.pub  id_ed25519  id_ed25519.pub  id_rsa  id_rsa.pub
known_hosts  known_hosts.old

[trainee@redhat9 .ssh]$ cat authorized_keys
ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBGJIMNJ1m+xIpYzfYwK7VpdCI9inhQx3wptO+z4Xsl3XYcb+WIXsEsJpKSyQn
Ov98HmfZVJWcqXaSBkE5mskFGI= trainee@redhat9.ittraining.loc

[trainee@redhat9 .ssh]$ cat id_rsa.pub >> authorized_keys

[trainee@redhat9 .ssh]$ cat id_dsa.pub >> authorized_keys

[trainee@redhat9 .ssh]$ cat id_ed25519.pub >> authorized_keys

[trainee@redhat9 .ssh]$ cat authorized_keys
ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBGJIMNJ1m+xIpYzfYwK7VpdCI9inhQx3wptO+z4Xsl3XYcb+WIXsEsJpKSyQn
```

```
Ov98HmfZVJWcqXaSBkE5mskFGI= trainee@redhat9.ittraining.loc
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDSBgNBWkHU0UWXablPEQLRXjcdG7WYN0651bSh122wAMpMo5j3Jjlxm+tCKILpU5kjmpLIC+YCVw3nNr2e8
RxIucxfEy1NIXrrCS0Ps3t5zsdsta/Z716BvX0x4tRuKqLPOMmwLWyvjYG4ixnECNSJTnHOmj122wxpdBX6HWJpJ+61NIvOnI+fAGTU6nPmb0jkan
NT/HZ27eDlqo88gNAXrRSt/Uc1bglNue5NGKn2TcuUEIau0tmZpguAcWfrESl4SK7bg0c+IJZHZw8T8Fkpo44T4oWj03/x+uWjpphOfFPux6nAiF4
+bRxFZnoUMvhe/WGX13Yxf0w3mymRyZXeSDyb0KF32Vy9hVh+0nVAIRtzwCc82aFtVdlJi9wW54tUwUlMkVFjVylJRNif37FwRd65BNp2eoN7v2IT
dZ+uWLGs3HJZKp+xLTHMZ3CZMkLyNsbdwyD5VAsD3dFB49W2voSC9DN5xlUWPp4m7TbNifa8b7nuQKP8p2FaVdr3YWc=
trainee@redhat9.ittraining.loc
ssh-dss
AAAAB3NzaC1kc3MAAACBAOlvM/inrDQNvmCMcWH4eTNud6egbiG4XysF++2If9Dx89mW8RWZmlvyiV8wRf71UyHHdiiBc/3SkYrka04l65F3GYH8B
Q0jiaf4WpzTvn1uuEjS2kO3+vrXW1JxYlWiOYyb44eufnrxK+qia6FkhC6Wmn6xnibmbwkBeXTXOQp7AAAAFQCYu7xH3JreaiS4bYuZ1b3MV4IAmQ
AAAIEApEwhXE8jn2SWk/tpQKkX5dATCa1K5T+XMEzunjeEr+w1F1tapptOFaaujmZeelNgBYT4LauNYRu5TuXmoDpOp2q1puQKmGjW3b6bQRN0Pal
o/rcPlI6NN3Efm242vhspWEl4fjYFoVxPfaG0ysTwjOmgM4TamcxgrYDclDc0hNUAAACBAJrqqa12g422N6YRw3CXbyMwSv2xagX09YjwvsbDBMyC
JtqoDg+6YavISLU3VQYJ+FmzBzObS2lkzk0yGMgqKOmnRIPaPi3HmpSPXp7828BU4lTsN4yv6zp4C1MIazvnE2rqBIVy1ZhCt9ADiCHZrRY2M/Czl
jfUhi/LinnznFVs trainee@redhat9.ittraining.loc
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIKzhdX1reo1vStZd/lfAQ1Yjz7eo0qj7ir/f6jyGp4iG trainee@redhat9.ittraining.loc

[trainee@redhat9 .ssh]$ exit
logout
Connection to localhost closed.
[trainee@redhat9 ~]$
```