PROJET SYSTEMES ET RESEAUX II

Nos premiers pas dans l'administration systeme

07 AVRIL 2023

DJERA Feriel
KHENTICHE Wissam
STRAINCHAMPS Clothilde

Introduction

Pour le cours de système et réseau, nous avons réalisé l'installation et la configuration complète d'un serveur et des infrastructures réseau et services nécessaires à son fonctionnement.

Le but principal de ce serveur est de gérer les services informatiques nécessaires à une petite entreprise comme, la base qui est d'avoir un petit parc de machine présente sur un réseau privée. Ces machines communiqueront entre elles et avec l'extérieur en sécurité. Elles composeront le domaine informatique de notre agence.

L'agence aura en quelque sorte son propre intranet.

La base sera de définir nos réseaux et les services nécessaires pour le gérer puis les chemins de communication nécessaire à notre agence vers le monde extérieur. Nous créerons et implanterons notre domaine sur notre réseau d'agence.

Un serveur web sera mis en place, et les pages héberger par ce dernier seront accessible depuis les postes présents dans notre domaine.

Un service d'authentification et de partage des ressources au sein de notre agence sera également présent.

Un système de sauvegarde de notre serveur sera également installé.

Au cours de ce projet, nous avons rencontré plusieurs difficultés et nous en avons surmonté la plupart. C'était pour chacune la première fois que nous allions administrer un système. Ce fut une expérience très intéressante, cela nous à permis de comprendre comment un système fonctionne à sa base et de mieux appréhender le monde informatique qui nous entoure au quotidien.

Plan du rapport

I. Adressage et routage

A. Installation d'une distribution GNU / Linux Debian 11.

Afin de pouvoir installer la distribution GNU/Linux Debian 11 Bullseye sans interface graphique via le réseau IEM, nous avons préparé deux machines Dell qui seront utilisées comme matériel informatique pendant toute la durée du travail. La première machine, appelée « serveur », sera utilisée comme routeur, tandis que la seconde machine, appelée « client », sera utilisée pour effectuer des tests et des recherches sur Internet.

Pour assurer l'installation, il nous a fallu utiliser un câble Ethernet pour avoir accès à Internet. Après le démarrage de notre machine «serveur» et lui ajouter un câble Ethernet afin de nous connecter à Internet,nous avons lancé le processus d'installation en commençant par attribuer un nom à notre machine et de renseigner le domaine. Nous avons également attribué un mot de passe pour le compte «root». Ensuite, nous avons partitionné les disques et choisi la méthode de partitionnement «Assisté - Utiliser un disque entier», ainsi que modifier le schéma de partitions et défini la taille de la mémoire et la création d'un disque dur.

Pour assurer une bonne installation de GNU/Linux Debian, nous avons dû créer des partitions avec des fonctions et des capacités spécifiques. Nous avons eu besoin d'au moins une partition pour le système d'exploitation (la racine), une pour les données des utilisateurs (home), une pour la gestion des logins (var/log) et une pour l'espace de travail du système d'exploitation (swap). Il était important de séparer ces partitions pour clarifier leur utilisation. Chacune de ces partitions a dû être formatée avec précision afin que le système d'exploitation sache où stocker les données. Nous avons généralement utilisé le format ext4 pour les partitions, sauf pour la partition SWAP qui a été formatée en tant qu'espace d'échange.

Après avoir installé et partitionné notre système, nous avons connecté notre routeur au réseau de l'IEM en utilisant le nom de domaine «sarte.iem». Ensuite, nous avons configuré l'adresse IP de notre poste 1 à 172.31.20.111, avec un netmask de 24 bits et une passerelle par défaut avec une à IP 172.31.20.1. Pour effectuer cette configuration, nous avons édité le fichier «/etc/network/interfaces», qui permet de configurer les interfaces réseau, y compris les adresses IP et les masques de sous-réseau, les passerelles par défaut et les paramètres DNS.

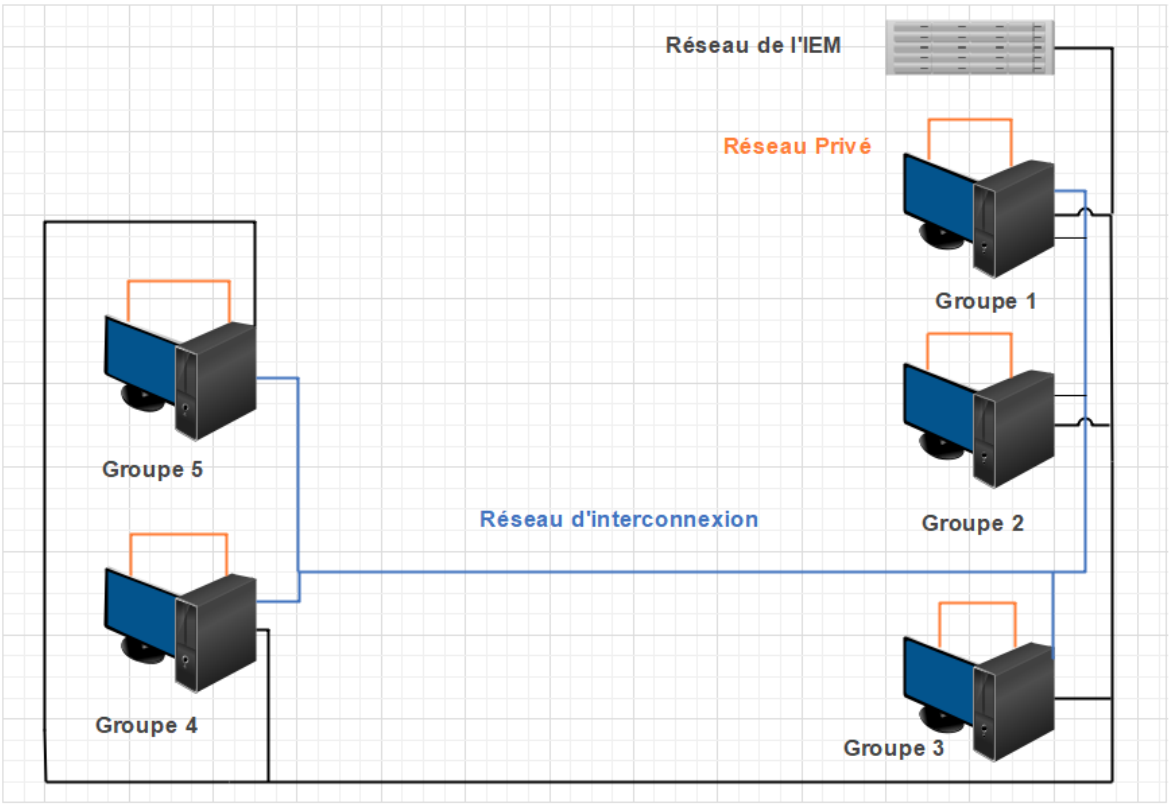
```
# The primary network interface
allow-hotplug eno1
iface eno1 inet static
  address 172.31.20.111
  netmask 255.255.255.0
  gateway 172.31.20.1
```

B. Plan d'adressage et de routage

1) Plan d'adressage

Lors de la première séance de travaux pratiques, nous avons mis en place un schéma d'adressage qui permettra à nos différentes agences de communiquer entre elles.En collaboration avec les autres groupes, nous avons décidé d'utiliser un sous-réseau de la classe C pour nos réseaux d'interconnexion, tandis que pour les réseaux privés, nous avons opté pour une classe A.
Ci-dessous le plan d'adressage de chaque groupe ainsi qu'une architecture du réseau:

	Adresse IEM	Adresse d'interconnexion	Adresse Privée
Groupe N°1	172.31.20.111	192.168.30.1	10.128.1.1
Groupe N°2	172.31.20.112	192.168.30.2	10.128.2.1
Groupe N°3	172.31.20.113	192.168.30.3	10.128.3.1
Groupe N°4	172.31.20.114	192.168.30.4	10.128.4.1
Groupe N°5	172.31.20.115	192.168.30.5	10.128.5.1



2) Implémentation de réseau d'interconnexion et du réseau privé

Pour configurer notre système, nous nous sommes rendus au fichier /etc/network/interfaces afin de configurer les interfaces de notre réseau. D'après ce dernier, notre système dispose de trois cartes réseau qui vont nous permettre de configurer nos différents réseaux. Nous avons connecté les cartes réseau Enp1s0 et Enp3s0 respectivement au réseau d'interconnexion et au réseau privé. Quant à la carte eno1, nous lui avons attribué le réseau IEM juste après l'installation de la distribution GNU/Linux.

```
# The primary network interface
allow-hotplug eno1
iface eno1 inet static
  address 172.31.20.111
  netmask 255.255.255.0
  gateway 172.31.20.1

allow-hotplug enp1s0
iface enp1s0 inet static
  address 192.168.30.1
  netmask 255.255.255.0

allow-hotplug enp3s0
iface enp3s0 inet static
  address 10.128.1.1
  netmask 255.255.255.0
```

3) Règles de routage

Pour permettre à une machine du réseau privé d'une agence de joindre une machine du réseau privé d'une autre agence, nous avons dû activer la fonction de routage sur les routeurs des deux agences. Ensuite, nous avons configuré les règles de routage appropriées pour assurer l'acheminement des paquets de données entre les deux réseaux.

La configuration des règles de routage et l'activation de la fonction de routage sont considérées comme une étape très importante afin de garantir la sécurité et la fiabilité de notre réseau .

Pour cela, nous avons commencé par activer la fonction de routage, pour assurer l'activation, on s'est rendu au fichier «/proc/sys/net/ipv4/ip_forward» qui est un fichier qui contient une valeur numérique (0 ou 1) qui indique si la fonction de routage IP est activée ou désactivée.

Pour l'activer il nous a fallu taper cette commande :

```
echo 1 > /proc/sys/net/ipv4/ip_forword
```

Cela activera la fonction de routage IP sur le système. Cependant cette modification ne sera pas persistante après un démarrage du système. Pour rendre cette modification persistante, on a également ajouté la ligne suivante au «etc/sysctl.conf» :

```
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
```

Cela garantira que la fonction IP est activée automatiquement lors du démarrage du système.

Nous avons choisi de définir nos règles de routage dans un fichier *routing-rules.conf* qui est dans le dossier de configuration */etc/systemd/system*, car c'est le système standard dans de nombreuses distributions Linux modernes. En utilisant systemd, nous pouvons créer des unités de service personnalisées qui peuvent être démarrées au démarrage du système. Cela nous permet de gérer nos règles de routage de manière cohérente et fiable, tout en utilisant des outils modernes et robustes pour la gestion des services et des processus système.

```
#route sur le reseau d'interco
ip route add 10.128.2.0/24 via 192.168.30.2 dev enp1s0
ip route add 10.128.3.0/24 via 192.168.30.3 dev enp1s0
ip route add 10.128.4.0/24 via 192.168.30.4 dev enp1s0
ip route add 10.128.5.0/24 via 192.168.30.5 dev enp1s0
```

C. Service DHCP

DHCP (Dynamic Host Configuration Protocol) est un protocole réseau utilisé pour distribuer automatiquement des adresses IP et d'autres informations de configuration réseau (tels que les serveurs DNS et les passerelles) à des appareils connectés à un réseau. DHCP permet d'éviter les conflits d'adresses IP en affectant dynamiquement une adresse unique à chaque appareil. Il facilite également la gestion du réseau en permettant une configuration centralisée et automatisée des paramètres de réseau pour les appareils qui se connectent au réseau.

On doit d'abord installer le serveur DHCP avec:
`apt-get install isc-dhcp-server`

Dans le premier, il faut déterminer à quelle interface le serveur va être lié dans notre cas, il s'agit enp3s0, soit la carte de réseau privé. C'est logique puisque l'on veut que notre routeur attribue des adresses automatiquement aux clients connectés.

Ensuite configurer l'interface réseau sur laquelle le serveur DHCP écoutera :
Modifier le fichier */etc/default/isc-dhcp-server*, on décommente la ligne :
`DHCPDv4_CONF=/etc/dhcp/dhcpd.conf`

Et on spécifie l'interface d'écoute en bas remplaçant la ligne et rajouter l'interface réseau :
INTERFACESv4="enp3s0"

Puis éditer le fichier de configuration */etc/dhcp/dhcpd.conf* pour définir les adresses IP à distribuer et associer aux adresses hardware et le DNS à prendre en compte :

```
#Etendue LAN
subnet 10.128.1.0 netmask 255.255.255.0 {
option routers 10.128.1.1;
range 10.128.1.2 10.128.1.7;
}
```

On redémarre le service DHCP pour appliquer les modifications:
`sudo service isc-dhcp-server restart`
Une fois la config terminée, on lance notre DHCP :
`systemctl start isc-dhcp-server`
Et on ajoute le service au démarrage :
`systemctl enable isc-dhcp-serve`

Le service DHCP est alors opérationnel pour notre réseau privé.

D. Synthèse des commande de gestions de paquets

Pour administrer efficacement un système GNU/Linux Debian,il est nécessaire d'installer des paquets supplémentaires, car les commandes apt-get ne sont pas les seuls permettant d'administrer un système de gestion de packages, pour cela nous avons ajoutés les paquets dpkg et dselect qui sont des outils de gestion de paquets . En plus de la commande apt-get, il existe deux autres commandes couramment utilisées pour gérer les paquets sur un système Debian :

apt est une commande plus récente qui fournit une interface simplifiée pour la gestion des paquets. Elle regroupe plusieurs fonctionnalités apt-get et ajoute également quelques fonctionnalités supplémentaires, telles que la prise en charge des transactions apt, la prise en charge de la progression de la commande et la prise en charge de l'installation en mode interactif.

aptitude : est une commande de gestion de paquets qui fournit une interface en mode texte pour gérer les paquets sur un système Debian. Elle peut résoudre les dépendances entre les paquets de manières plus sophistiquée que apt-get et offre également des fonctionnalités avancées pour la recherche et la sélection de paquets.

dpkg est une commande qui permet d'installer et de désinstaller des paquets sur un système Debian. Elle est utile pour installer des paquets qui ne sont pas disponibles dans la distribution standard.

dselect est une alternative à apt-get pour la gestion des paquets sur un système Debian. Elle offre des fonctionnalités supplémentaires, telles que la possibilité de marquer un paquet, de voir l'état des paquets et de prendre en compte les paquets recommandés ou suggérés en plus des dépendances lors de l'installation. Cependant, dselect est considéré comme plus complexe à utiliser que apt-get .

II. Installation et configuration d'un DNS

A. Prérequis

Le service de nom de domaine (DNS – Domain Name System) est la spécification des services de résolution de nom. Ce service est constitué de deux activités distinctes : le processus de traduction des noms d'hôtes en adresse IP et les mécanisme de distribution des données de traduction sous-jacente. Le DNS est une base de données distribuée dont le contenu est réparti sur tout l'internet, avec des serveur DNS stockant en permanance uniquement le sous-ensemble de données dont ils sont responsablent. Les requêtes sur cette base de données massivement distribuée fonctionnent car un DNS à la possibilté de renvoyer automatiquement les requêtes vers le bon serveur, selon un principe (« de bouche à oreilles ») qui s'adapte parfaitement à l'accroissement de la taille du système global. L'ensemble des données des DNS s'appelle aussi *l'espace de noms* du DNS.

La structure du DNS définit une arborescence de noms de domaines (par ex : les .com pour les entités commerciales, .gov pour les entités gouvernementale ...). Les noms de domaines sont organisés au sein d'une arborescence dont la racine est le *domaine racine* auquel ont fait référence par un point « . ». Sous le domaine racine, on trouve un ensemble de domaine de haut niveau (TLD- top level domain) dont les noms sont sous une des deux formes suivantes : des suffixes génériques désignant un type d'organisme (gTLD, par ex : .biz, .com) ou des codes de pays sur deux caractères (ccTLD par ex : .fr, .us, ...).
Certains de ces TLD sont sous-divisés avant que des noms de domaines spécifique à un organisme soient attribués (par ex : .gouv.fr)
Pour obtenir notre propre nom de domaine si nous sommes une entreprise il faudrait s'inscrire au près de l'autorité gérant le TLD dans lequel on veut que notre domaine se trouve.

Les implémentations du DNS, y compris BIND (Berkeley Internet Name Domain) d'Unx comportent les composantes suivantes : un *résolveur*, qui est une bilbiothèque utilisée par les commandes et les programmes utilisateur. Puis un *serveur de nom* qui pour les systèmes Unix est le démon named, qui est configuré par un ensemble de fichier de configuration.

Les serveurs de noms fournissent des services de résolutions de noms pour une certaine zone de DNS. Une zone représente un ensemble d'hôtes au sein d'un domaine.
Certains fichiers de zones contiennent des enregistrements qui associent des noms d'hôtes à des adresses IP qui sont utilisés pour les requêtes du DNS. D'autres définissent des zones de recherche inversée et sont utilisés pour le traitement de requête opposée, donc associé une adresse IP à un nom d'hôte.
Chaque zone de renvoi a au moins une zone de recherche inversée correspondante.

Il existe plusieurs types de serveurs de noms. Ils peuvent fonctionner de différentes manières :

- Ils peuvent effectuer des recherches récursives ou non récursivesen réponse aux requêtes.
- Ils peuvent renvoyer des réponses autorisées ou non autorisées, avec différentes sortent de serveurs associés.
- Des serveurs de noms en cache qui détiennent des enregistrement DNS non officiel pour n'importe qu'elle zone, ils conservent toutes les informations qu'ils obtiennent dans leur cache pendant un certain temps.
- Les retransmetteurs (forwarders) qui sont des serveurs de noms désignés comme la cibles des requêtes extérieures au domaine local. Quand un serveur de nom est configuré de manière à employé un retransmetteur, il envoie toujours des requêtes pour les hôtes qu'il ne cônnait pas au retransmetteur. Et si celui-ci ne peut fournir la réponse, il tente de la déterminer lui-même en contactant d'autres serveurs de noms qu'il connaît.

B. Quelques tests de compréhensions

Après avoir vu les bases théoriques pour concevoir un DNS, on peut maintenant réaliser quelques test pour comprendre comment interroger un serveur de nom et ce qu'il nous renvoie en réponse.

Pour interroger un DNS nous avons plusieurs commandes à notre disposition :

- host + au minimum le hostname de la machine rechercher ou son IP, d'autres options peuvent-être ajouter...

Par exemple la commande

```
host www.bourgogne.fr
```

nous renvoie

```
www.u-bourgogne.fr is an alias for tokyo.dmz.u-
bourgogne.fr.
tokyo.dmz.u-bourgogne.fr has address 193.52.234.14
```

On peut en déduire que la page internet recherché est hébergé sur le serveur tokyo qui se situe dans le domaine de l'universite de bourgogne qui à pour adresse IP 193.52.234.14.

On peut également faire la recherche inverse avec la commande

```
host 193.52.234.14
```

qui nous renvoie

```
14.234.52.193.in-addr.arpa domain name pointer
tokyo.dmz.u-bourgogne.fr.
```

On peut en déduire que la machine ayant pour adresse IP 193.52.234.14 a pour nom tokyo et se situe dans le domaine de l'universite de bourgogne.

- dig simplement ou dig + le hostname de la machine rechercher ou son IP

Par exemple la commande

```
dig www.bourgogne.fr
```

nous renvoie

une partie récapitulative de la requête DNS

```
; <<>> DiG 9.16.37-Debian <<>> www.u-bourgogne.fr
;; global options: +cmd
```

```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 7283
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0,
ADDITIONAL: 1
```

une partie options de la requête DNS

```
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE:
d2a837d09e3112da010000006429eb486fbf54bbea1a8b3d (good)
;; QUESTION SECTION:
;www.u-bourgogne.fr.            IN    A
```

une partie réponse de la requête DNS

```
;; ANSWER SECTION:
www.u-bourgogne.fr. 3600 IN    CNAME     tokyo.dmz.u-
bourgogne.fr.
tokyo.dmz.u-bourgogne.fr. 3600     IN    A     193.52.234.14
```

une partie informations sur la requête DNS

```
;; Query time: 7 msec
;; SERVER: 10.128.1.1#53(10.128.1.1)
;; WHEN: Sun Apr 02 22:53:28 CEST 2023
;; MSG SIZE  rcvd: 129
```

La partie réponse de la requête DNS est la plus importante, on peut voir le nom du domaine rechercher, la classe de requête, le type de la requête puis l'adresse IP associé au nom du domaine rechercher.

De plus la commande dig peut nous permettre de spécifier quel DNS on veut utiliser pour faire notre recherche. On peut alors entrer la commande :

```
dig @8.8.8.8 www.u-bourgogne.fr
```

qui nous renvoie la même réponse que précedement avec des changements dans la partie information sur la requête DNS.

```
;; Query time: 31 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Sun Apr 02 23:12:22 CEST 2023
;; MSG SIZE  rcvd: 87
```

Et qui nous montre que la rêquete est partie du DNS de Google.

Pour finir l'option + trace de dig qui peut être interessante pour savoir le parcours de notre requête.

- nslookup seul fera tourner en boucle la commande et on pourra entrer l'hostname de la machine rechercher ou son IP et nous renverra une réponse puis exit pour sortir de nslookup, ou bien nslookup + hostname de la machine rechercher ou son IP. Nslookup à approximativement les mêmes fonctions que host.

C. Mise en pratique

Le but de cette partie est d'expliquer la mise en œuvre de notre serveur de noms et les choix faits pour la configuration de celui-ci. L'objectif est de répertorier les machines présentes sur notre réseau privé pour qu'elles sachent entre-elles qui est la machine client, qui est la machine serveur... Sur notre réseau nous avons une capacité d'acceuil de 6 clients, donc 6 adresses IP (nom) différentes à répertorier dans notre serveur de noms, plus l'adresse IP de notre serveur. Ceci créer alors notre domaine.

A11filles est le hostname de notre machine serveur.

Dans le cadre de notre projet notre domaine est local donc il ne sera accessible que sur le réseau privé de nôtre agence, donc aucun besoin de ce déclarer au autorité.

1) Installation de BIND9

   La première chose a faire est de passer en mode root pour avoir tous les droits. Toute installation et configuration de paquet sur une machine doivent se faire en mode 'super-utilisateur'. On execute alors la commande
   `su -`
   et on entre notre mot de passe root.

   On installe ensuite le paquet BIND9 avec le gestionnaire d'application de notre choix avec la commande :
   `apt-get install bind9`

   Ensuite on se rend dans le dossier etc puis dans le dossier bind, où se situe tous nos fichiers de configuration nécessaire à la mise en place de notre serveur de noms avec la commande :
   `cd /etc/bind`

2) Déclaration de notre zone

   Dans le dossier bind nous avons un fichier de configuration *named.conf.local* et c'est dans ce fichier que se trouveront nos déclaration de zones.
   Nous allons maintenant ajouter nos zones dns, la première sera la zone de resolution de notre domaine principale qui se situe sur notre réseau d'agence privé. Le nom choisi pour notre domaine est alors : *agence-filles.dijon* et la zone est déclarer comme suit :

```
zone "agence-filles.dijon" IN {//définition du nom de notre zone
        type master; //type de DNS
        file "/etc/bind/db.agence-filles.dijon";
          //où se situe le fichier de configuration de la zone
        notify no; //notification au serveur esclave quand une zone est
        mise à jour, inutile dans notre cas, car pas de serveur esclave.
};
```

Puis comme nous l'avons vu dans les préresquis ci-dessus cette zone est suivi de sa zone de résolution inverse déclarer comme suit :

```
zone "1.128.10.in-addr.arpa"{//adresse IP noté à l'envers
                                                    sans la partie hostID
     type master;
     file "/etc/bind/db.10";
     notify no;
};
```

Si on le souhaite on peut rajouter un attribut de plus à notre zone qui est `Allow-transfert{IP d'un autre serveur de noms}` qui autorise le transfert vers cette autre seveur de nom dans le cas ou il y aurait un serveur de type esclave.

3) Fichier de résolution de zone

Pour cette partie nous devons créer un fichier de résolution de zone, on peut donc copier le fichier *db.127* se situant dans le dossier bind. Ce fichier nous servira de base, il nous faudra alors modifier quelques lignes pour l'adapter à notre cas.

On commence par créer le fichier de résolution de zone de notre domaine. En se plaçant dans */etc/bind* on copier le fichier *db.127* et on renomme la copie avec la commande :
`cp db.127 db.agence-filles.dijon`

On modifie le fichier db.agence-filles.dijon et on obtient ceci :

```
;
; BIND data file for agence-filles.dijon
;
$TTL    604800
@   IN  SOA A11filles.agence-filles.dijon. root.A11filles.agence-filles.dijon. (
                  4     ; Serial
             604800     ; Refresh
              86400     ; Retry
             2419200    ; Expire
             604800 )   ; Negative Cache TTL

@            IN      NS      A11filles.agence-filles.dijon.
A11filles    IN      A       10.128.1.1
client1      IN      A       10.128.1.2
client2      IN      A       10.128.1.3
client3      IN      A       10.128.1.4
client4      IN      A       10.128.1.5
client5      IN      A       10.128.1.6
client6      IN      A       10.128.1.7
informatique IN      CNAME   A11filles.agence-filles.dijon.
```

On a donc pour le domaine *agence-filles.dijon* défini notre DNS principale qui est A11filles.agence-filles.dijon et associé chaque noms à une adresse IP.

On fait de même pour le fichier de résolution inverse. On copie *db.127* et on renomme la copie avec la commande :
`cp db.127 db.10`

On modifie le fichier db.10 et on obtient ceci :

```
;
; BIND reverse data file for local 10.128.1.XXX net
;
$TTL    604800
@       IN      SOA     A11filles.agence-filles.dijon. root.A11filles.agence-filles.dijon. (
                              2         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL

@       IN      NS      A11filles.agence-filles.dijon.
1       IN      PTR     A11filles.agence-filles.dijon.
2       IN      PTR     client1.agence-filles.dijon.
3       IN      PTR     client2.agence-filles.dijon.
4       IN      PTR     client3.agence-filles.dijon.
5       IN      PTR     client4.agence-filles.dijon.
6       IN      PTR     client5.agence-filles.dijon.
7       IN      PTR     client6.agence-filles.dijon.
informatique       IN      CNAME   A11filles.agence-filles.dijon.
```

On a donc pour le domaine *agence-filles.dijon* défini notre DNS principale qui est A11filles.agence-filles.dijon et associé chaque adresse IP un nom.

4) Indication d'un retransmetteur

On va également indiquer le serveur de noms d'un autre domaine qui est plus grand qui est celui de l'iem. On fait ceci car notre serveur ne connait pas beaucoup de machines, donc si une de nos machines veut en joindre une que notre DNS ne connaît pas il demandera à cet autre DNS si lui il la connaît et ainsi de suite...

Pour ceci on se rend dans le *fichier named.conf.options* et dans la partie *forwarders* on renseigne l'adresse IP du DNS de l'iem comme ceci :

```
forwarders {
          172.31.21.35;
     };
```

5) Redémarrage et vérification de la configuration de bind9

Cette étape à pour but de redémarer le service bind9 et de vérifier si les modifications que nous avons apporter précedement son juste. Chaque petite erreur de résonement ou même de faute de frappe peut faire que la mise en service échoue.

Pour redémarrer le service on entre la commande :
`systemctl restart bind9`
Nous n'avons aucun retour commande ce qui est bon signe, puis pour vérifier que le service est redémarré on entre la commande :
`systemctl status bind9`
en retour console on à bien que le service est activé. Il est essentiel de rentrer ces commandes après toute modifications des fichier de configuration pour que le service prend en compte les dernières modifications.

On vérifie nos fichiers de configuration avec la commande :
`named-checkconf /etc/bind/named.conf.local`
Nous n'avons aucun retour console ce qui signifie que notre fichier est correct.

On vérifie nos fichiers de zone avec la commande :
`named-checkzone agence-filles.dijon /etc/bind/db.agence-filles.dijon`
et
`named-checkzone agence-filles.dijon /etc/bind/db.10`
Nous avons un retour console qui est OK ce qui signifie que notre fichier est correct.

On modifie également dans le fichier*/etc/dhcp/dhcpd.conf* pour que notre DNS soit pris en compte.

6) Tests de vérifications

Il nous reste maintenant à vérifier le bon fonctionnement de notre serveur de noms. On interroge donc notre serveur avec la commande nslookup suivi du nom du client ou du serveur puis nslookup suivi de l'IP du client ou du serveur. On obtient alors le résultat suivant.

Test numéro 1 :

```
root@A11filles:/# nslookup A11filles
Server:         10.128.1.1 // on recherche bien à partir du DNS que
                                                    l'on vient de créer
Address:        10.128.1.1#53

Name:   A11filles.agence-filles.dijon
Address: 10.128.1.1

root@A11filles:/# nslookup 10.128.1.1
```

```
1.1.128.10.in-addr.arpa name = A11filles.agence-
filles.dijon.
```
On obtient les bonnes informations, la machine A11filles à l'adresse IP 10.128.1.1 ce qui est correcte sur notre réseau et vice et versa.

Test numéro 2 :

```
root@A11filles:/# nslookup client1
Server:         10.128.1.1
Address:        10.128.1.1#53

Name:   client1.agence-filles.dijon
Address: 10.128.1.2

root@A11filles:/# nslookup 10.128.1.2
2.1.128.10.in-addr.arpa name = client1.agence-
filles.dijon.
```
On obtient les bonnes informations, la machine client1 à l'adresse IP 10.128.1.2 ce qui est correcte sur notre réseau et vice et versa.

L'installation et la configuration de notre DNS est alors terminé, et le DNS est fonctionnel pour le domaine *agence-filles.dijon*.

III. Installation d'un serveur LAMP

A. Prérequis

Dans cette partie un peu plus théorique nous verrons la définition d'un serveur web puis son fonctionnement.

Qu'est ce qu'un serveur web ? Un serveur web est un serveur informatique permettant de stocker et de publier des pages web (écrit en html, php, ...) sur internet ou sur l'intranet. Le client qui est en général un navigateur web envoie une demande de page web au serveur web et lui renvoie en réponse la page demander. Tout ça par le protocole de communication http ou https, on parle alors de requête et de réponse http ou https. Un serveur web peu faire réference à des composants logiciel, matériel ou bien les deux fonctionnant ensemble.

Qu'est-ce qu'Apache ? Le logiciel libre Apache HTTP Server est un serveur http créer et maintenu au sein de la fondation Apache.
C'est donc Apache qui gère la connexion fluide et sécurisé entre le client et le serveur. De plus Apache est hautement personnalisable car il est basé sur plusieurs modules ce qui permet à l'administrateur du serveur d'activer ou de désactiver les modules. Apache à beaucoup d'avantages : il est gratuit, fiable, flexible, facile à configurer et il y a beaucoup de documentation sur internet, mais aussi quelques inconvénients comme : un problème de performance sur les sites web avec un énorme trafique et trop d'options de configuration peuvent mener à la vulnérabilité de la sécurité.

Les fichiers permettant de configurer le serveur web Apache se situe dans /etc/apache2, on y trouve alors un fichier avec tous les modes d'Apache disponibles et dans un autre ceux qui sont déjà activé. De même pour les configurations et les sites (hôte virtuel).
En général on créer et édite les fichiers dans le dossier pas activé, puis par le biais d'une commande on active ou désactive le fichier de configuration, il passe alors dans le dossier des configurations activé.

Le plus souvent un serveur web à besoins d'autres options pour gérer par exemple PHP, les bases de données... il est donc nécessaire de les installer en plus.

B. Installation et configuration de Apache 2

Le but de cette partie est d'expliquer la mise en place de notre serveur de web et les choix faits pour la configuration de celui-ci.

1) Instalation d'Apache2

   La première chose à faire est de passer en mode root pour avoir tous les droits. Toute installation et configurations de paquets sur une machine doivent se faire en mode 'super-utilisateur'.

On exécute alors la commande
`su -`
et on entre notre mot de passe root.

On installe ensuite le paquet Apache2 avec le gestionnaire d'application de notre choix avec la commande :
`apt-get install apache2`

Pour tester notre installation on peut entrer la commande :
`links2 http://agence-filles.dijon`
L'utilitaire links2 s'ouvre alors dans notre terminale et on voit apparaître une page web préconçue par Apache, par défault cette page se trouve dans /var/www/html. Par la suite cette redirection sera changer.

2) Configuration de l'espace de travail de l'utilisateur Web1

Tout d'abord web1 est un utilisateur dévelopant des applications web, il doit donc avoir un accès plus libre au serveur web Apache plus complet que les autres utilisateurs.

Nous allons donc créer un compte unix pour cette utilisateur avec la commande :
`adduser web1`
avec comme retour console

```
Ajout de l'utilisateur « web1 » ...
Ajout du nouveau groupe « web1 » (1008) ...
Ajout du nouvel utilisateur « web1 » (1007) avec le
groupe « web1 » ...
Création du répertoire personnel « /home/web1 »...
Copie des fichiers depuis « /etc/skel »...
Nouveau mot de passe :
Retapez le nouveau mot de passe :
passwd: password updated successfully
Changing the user information for test1
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Cette information est-elle correcte ? [O/n]o
```

Notre utilisateur web1 est alors créer, il a alors son propre dossier /home/web1.

On veut par la suite que les fichiers web que va développer web1 se situe dans /srv/web1, car /srv est un dossier qui peut contenir des données pour les services fournis par le système, le dossier /web1 est alors à sa place.

On créer le dossier web1 dans /srv avec les commandes :
```
cd /srv
mkdir web1
```

On rend l'utilisateur web1 propriétaire du dossier web1 avec la commande :
```
chown -R web1 /srv/web1
```
car sinon il n'aurait eu aucune action possible dessus.
On se positionne comme étant l'utilisateur web1 avec la commande :
```
su web1
```
puis on créer le dossier /www ou se siturons les pages web développer par web1 plutard. On créer une page d'acceuil test se situant dans le fichier *index.html,* cette page nous indiquera lors de nos tests suivant que nous sommes bien sur la page web de web1.

Par la suite on veut que la page définit dans *index.html* soit accessible depuis un naviguateur via l'adresse : http://agence-filles.dijon/web1. Pour cela il nous faut définir un hôte virtuel (*virtualHost)* dans un fichier de configuration d'Apache.

On se positionne dans le dossier /etc/apache2 pour la suite de la configuration.

Premièrement, dans le fichier apache2.conf on décoche les lignes suivantes pour que le serveur web ai accès au dossier /srv.

```
<Directory /srv/>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
```

Deuxièmement, on se rend dans le dossier *sites-available*. Ce dossier contient les fichiers de configuration des hôtes virtuels et le fichier *000-default.conf* est la configuration adopter par défault, on se servira de ce fichier comme base et on le modifira à notre convenance.
On copie donc *000-default.conf* et on nomme la copie *web1.conf* avec la commande :
```
cp 000-default.conf web1.conf
```
On modifie web1.conf jusqu'à obtenir ceci :

```
<VirtualHost *:80>
```
//on ouvre la balise de déclaration d'un virtualhost sur le port 80, donc n'est configuré que pour des requêtes http

```
  ServerAdmin web1@A11filles
  ServerName  agence-filles.dijon
  DocumentRoot /srv
```
//On l'emplacement de la page d'acceuil web de notre serveur

```
  ErrorLog ${APACHE_LOG_DIR}/error.log
  CustomLog ${APACHE_LOG_DIR}/access.log combined

  Alias "/web1" "/srv/web1/www"
```
//on créer un alias web1 qui pointe vers

les fichiers qui seront éditer par web1

```
  <Directory "/srv"> //on redonne accès à Apache sur /srv
    AllowOverride All
    Options Indexes FollowSymLinks
    Require all granted
  </Directory>
</VirtualHost>
```

Il nous reste maintenant à rendre active cette configuration, pour cela on entre la commande :

```
a2ensite web1.conf
```

a2ensite est un script qui configure apache2 pour activer le site fourni (qui contient un bloc). Il accomplit cela en générant des liens symboliques dans /etc/apache2/sites-enabled.
Puis on désactive la configuration par défault avec la commande :

```
a2dissite 000-default.conf
```

a2dissite désactive un site en supprimant ces connexions symboliques.
On relance le service apache avec les commandes :

```
systemctl reload apache2
```

et

```
systemctl restart apache2
```

Donc maintenant depuis un poste client de notre réseau privé, on peut dans un naviguateur web entrer l'adresse suivante : http://agence-filles.dijon qui nous amènera sur la nouvelle page d'acceuil de notre serveur ou on parra y lire « Bienvenue sur le serveur des filles ». Cette page est éditée dans le dossier /srv sous le nom de *index.html.*
Puis faire une nouvelle requête en entrant l'adresse suivante : http://agence-filles.dijon/web1, qui nous amènera sur la page web de web1 que l'on a édité précedement.

On peu maintenant conclure que l'espace de travail de web1 est totalement configuré et opérationnel.

3) Configuration de l'espace de travail de l'utilisateur Web2

Pour l'utilisateur web2, qui est un utilisateur faisant partie du département de la maintenance informatique de l'agence on doit également lui créer un espace spécifique sur le serveur web, avec comme pour web1 avec un peu plus de liberté qu'un utilisateur lamda. La différence avec web1 est que web2 à une adresse web différente de web1, qui sera http://informatique.agence-filles.dijon.

Comme pour web1 on créer un utilisateur unix web2 avec la commande :

```
adduser web2
```

Notre utilisateur web2 est alors créer, il a alors son propre dossier /home/web2. On veut par la suite que les fichiers web de web2 se situe dans /srv/web2, car /srv est un dossier qui peut contenir des données pour les services fournis par le système, le dossier /web2 est alors à sa place.

On créer le dossier web2 dans /srv avec les commandes :

```
cd /srv
mkdir web2
```

On rend l'utilisateur web2 propriétaire du dossier web2 avec la commande :

```
chown -R web2 /srv/web1
```

car sinon il n'aurait eu aucune action possible dessus.
On se positionne comme étant l'utilisateur web1 avec la commande :

```
su web2
```

puis on créer le dossier /www ou se siturons les pages web créer par web2 plutard. On créer une page d'acceuil test se situant dans le fichier index.html, cette page nous indiquera lors de nos tests suivant que nous sommes bien sur la page web de web2.

Deuxièmement, on se rend dans le dossier *sites-available,* et on ajoute un hôte virtuel au fichier web1.conf pour l'utilisateur web2.

```
<VirtualHost *:80>

  ServerAdmin web2@A11filles
        ServerName  informatique.agence-filles.dijon //on
                              définie le nom du serveur qui héberge la page
        DocumentRoot /srv/web2/www // son emplaçement

  ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        <Directory "/srv/web2/www">// on donne les droits à apache
                AllowOverride All
                Options Indexes FollowSymLinks
                Require all granted
        </Directory>
</VirtualHost>
```

Ensuite, on remarque que le nom du serveur mis n'est pas exactement celui de notre serveur physique qui héberge notre page, pourtant se sont les mêmes, ils ont juste le nom qui diffère.
On a vu précedement que l'on définit le nom d'une machine dans notre domaine grâce à un serveur de noms. Il nous faut donc donner deux noms à notre machine serveur. On revient donc dans le fichier *db.agence-filles.dijon* et on ajoute la ligne suivante à la fin :

```
informatique   IN CNAME   A11filles.agence-filles.dijon.
```

de même dans db.10.

Il nous reste maintenant à rendre active cette configuration comme précedement et à relancer le service apache.

Donc maintenant depuis un poste client de notre réseau privé, on peut dans un naviguateur web entrer l'adresse suivante : http://informatique.agence-filles.dijon qui nous amènera sur la page web de web2 que l'on a édité précedement.

4) Configuration de l'espace de travail de l'utilisateur Web3

Pour finir on va créer un dernier utilisateur web3 destiné à être utiliser par des utilisateurs occasionnels. Web3 ne doit donc pas avoir de droit sur une autre partie de notre serveur, il aura juste accès à son /home et ce qui l'en compose. La page de l'utilisateur web3 sera disponible depuis un poste client à l'adresse : http://agence-filles.dijon/~web3.

Comme précedement on créer un utilisateur unix web3.

Notre utilisateur web3 est alors créer, il a alors son propre dossier */home/web3*. On veut par la suite que les fichiers web de web3 se situe dans */home/web3/public_html*. On se positionne comme étant l'utilisateur web3, on créer le dossier *public_html* et on créer in fichier html qui nous indique que nous sommes sur la page de web3.

Il nous reste maintenant à relancer le service apache.

Maintenant depuis un poste client de notre réseau privé, on peut dans un naviguateur web entrer l'adresse suivante : http://agence-filles.dijon/~web3 qui nous amènera sur la page web de web3 que l'on a édité précedement.

C. Installation de PHP

PHP (Hypertext Preprocessor) est un langage de programmation open source très populaire utilisé pour le développement d'applications web. PHP est principalement utilisé pour les sites web dynamiques et les applications web, mais il peut également être utilisé pour les scripts en ligne de commande. Il est est capable de générer du contenu dynamique, de collecter des données de formulaire, de gérer des cookies, de se connecter à des bases de données, de créer des sessions utilisateur, de manipuler des fichiers, de communiquer avec des serveurs distants, et bien plus encore. PHP est également capable de se connecter à de nombreux systèmes de gestion de bases de données, tels que MySQL, PostgreSQL et Oracle. Une des principales caractéristiques de PHP est sa compatibilité avec de nombreux serveurs web, tels qu'Apache, Nginx et Microsoft IIS. En utilisant PHP, vous pouvez facilement créer des sites web dynamiques, des blogs, des forums, des applications e-commerce, des systèmes de gestion de contenu, et bien plus encore.

Commande d'installation :
```
apt-get install php
```
Cela nous installe directement la dernière version disponible de PHP.

Dans le dossier */etc/php* on peut trouver les dossiers des différentes versions de PHP que nous avons intallé. PHP à plusieur modules qui peuvent être activé par le biais du fichier de configuration php.ini se trouvant par exemple dans */etc/php/8.2/apache2.* Mais nous n'avons pas besoin d'y modifier quelquechose pour le moment.

Pour tester que php est bien installer, on créer une page index.php avec une partie php, par le biais de web1. On voit que notre code php est bien traiter par notre serveur web.

L'installation de php est alors terminée et opérationnelle.

D. Installation de MariaDB et création d'une BD

Mariadb est Système de Gestion de Base de Données (SGBD) communautaire de Mysql. Dans cette procédure, nous allons voir comment l'installer sur notre machine

1) Installation de MariaDB

   Nous allons installer les paquets serveurs et clients de Mariadb avec la commande :
   ```
   apt install mariadb-server mariadb-client
   ```

   Ensuite nous allons activer le service Mariadb et le démarrer
   ```
   systemctl enable mariadb
   systemctl start mariadb
   ```

   Et on vérifié que le service en cours d'exécution avec la commande :
   ```
   systemctl status mariadb
   ```

2) Configuration de MariaDB

   Dans cette partie on va ajouter une couche de sécurité à notre SGBD avec mySQL. MySQL est un système de gestion de bases de données relationnelles (SGBDR) open source. Il est utilisé pour stocker, organiser et récupérer des données pour les applications web. Il utilise le langage SQL pour communiquer avec la base de données et est connu pour sa vitesse, sa fiabilité et sa compatibilité avec de nombreux systèmes d'exploitation.

   On installe alors mySQL :
   ```
   mysql_secure_installation
   ```

On à alors le retour console suivant, et on répond au question qui nous son posé :

```
Enter current password for root (enter for none):
Change the root password? [Y/n] Y
New password:  votre_mdp
Re-enter new password: votre_mdp
Remove anonymous users? [Y/n] Y
Disallow root login remotely? [Y/n] Y
Remove test database and access to it? [Y/n] Y
Reload privilege tables now? [Y/n] Y
```

Nous pouvons nous connecter à MariaDB à l'aide de l'utilisateur root et du mot de passe choisis précédemment avec :

```
mysql -u root -p
```

Ensuite notre mot de passe nous sera demandé puis on va avoir la possibilité d'exécuter vos requêtes SQL. La première chose à faire est de créer un autre utilisateur qui aura moins de permissions que l'utilisateur root.

Donc on a créé un utilisateur Db1, et une base de données nommée *BDD*

```
create data base BDD;
```

Puis on a donné à cet utilisateur tous les privilèges nécessaires pour pouvoir lire et ajouter des donnés à cette base de données.
On a ensuite créé une table *Fille*s qui a comme attributs (Nom, Prenom, Age, Adressemail), avec l'insertion des tuples, avec larequêtes sql suivante :

```
create table if not exists Filles(
                        Nom varchar(30) not null,
                        Prenom varchar(30),
                        Age int,
                        Adressemail varchar(30),
                        primary key(Nom));
```

Nous avons ensuite inséré des informations dans notre table Filles :

```
insert into Filles(Nom,Prenom,Age,Adressemail)
          values('kh','wissam','21','ghft');
```

Voici la table *Filles*:

```
MariaDB [(none)]> use BDD;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [BDD]> select * from Filles;
+-----+-----------+------+-------------+
| Nom | Prenom    | Age  | Adressemail |
+-----+-----------+------+-------------+
| Dj  | Feriel    |   22 | .fr         |
| kh  | wissam    |   21 | ghft        |
| st  | Clothilde |   22 | @gmail      |
+-----+-----------+------+-------------+
3 rows in set (0,000 sec)
```

L'installation et la configuration de MariaDB est alors terminée et opérationnelle.

E. Installation de Posgresql et création d'une BD

PostgreSQL est un système de gestion de base de données relationnelles open-source, qui offre des fonctionnalités avancées de stockage et de traitement de données. PostgreSQL prend en charge les requêtes SQL, les transactions ACID (Atomicité, Cohérence, Isolation, Durabilité), les vues, les déclencheurs, les fonctions stockées et les procédures stockées. Il offre également une extensibilité élevée avec la possibilité de créer des types de données personnalisés, des fonctions d'agrégation, des opérateurs personnalisés, etc.

1) Intallation de Posgres :

   Nous allons installer les paquets serveurs et clients de Mariadb avec la commande :
   apt install postgresql

   Et on vérifié que le service en cours d'exécution.

```
root@A11filles:~# systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
     Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor preset: enabled)
     Active: active (exited) since Tue 2023-04-04 17:37:25 CEST; 14min ago
    Process: 821 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
   Main PID: 821 (code=exited, status=0/SUCCESS)
        CPU: 1ms

avril 04 17:37:25 A11filles systemd[1]: Starting PostgreSQL RDBMS...
avril 04 17:37:25 A11filles systemd[1]: Finished PostgreSQL RDBMS.
```

2) Configuration de Postges

```
root@A11filles:~# su - postgres
postgres@A11filles:~$ psql -l
                                  Liste des bases de données
   Nom    | Propriétaire | Encodage | Collationnement | Type caract. |    Droits d'accès
----------+--------------+----------+-----------------+--------------+----------------------
 BDD      | user2        | UTF8     | fr_FR.UTF-8     | fr_FR.UTF-8  |
 BDD2     | user2        | UTF8     | fr_FR.UTF-8     | fr_FR.UTF-8  |
 postgres | postgres     | UTF8     | fr_FR.UTF-8     | fr_FR.UTF-8  |
 template0 | postgres    | UTF8     | fr_FR.UTF-8     | fr_FR.UTF-8  | =c/postgres          +
          |              |          |                 |              | postgres=CTc/postgres
 template1 | postgres    | UTF8     | fr_FR.UTF-8     | fr_FR.UTF-8  | =c/postgres          +
          |              |          |                 |              | postgres=CTc/postgres
(5 lignes)

postgres@A11filles:~$ psql BDD
psql (13.9 (Debian 13.9-0+deb11u1))
Saisissez « help » pour l'aide.

BDD=# select * from Filles;
 nom  | prenom | age |   adressemail
------+--------+-----+-----------------
 Jean | Luc    |  40 | ghft@hotmil.com
(1 ligne)
```

F. Utilisation du trio PHP, Apache ET MySQL

Le trio PHP, Apache et MySQL est un ensemble de technologies utilisées pour développer des applications web dynamiques. PHP est utilisé pour générer du contenu web dynamique, Apache sert de serveur web pour héberger les fichiers PHP, et MySQL est utilisé pour stocker, organiser et récupérer des données pour les applications web.

1) Exemple (MariaDB) avec des données MySQL sans PDO

Une fois que les paquets sont installés, nous avons crée un script *index.php* dans */srv/web1/www* pour se connecter à la base de données mysql *BDD* afin d'afficher son contenudans la page web1, le voici le script:

```php
<html>
    <head>
        <title>Bienvenu dans ma page</title>
    </head>
    <body>
        <h1>Donnees mysql sans PDO</h1>
       <p>
       <?php
        $servername = 'localhost';
        $username = 'user1';
        $password = 'user1';
        $dbname= 'BDD';
        $conn=mysqli_connect($servername, $username, $password, $dbname);
        if(!$conn){
                echo "encore moi!!";
                die("La connexion a échouée:".mysqli_connect_error());
        }
         echo "connexion réussi";
         echo "<br>";
        $query="select * from Filles";
        $result=mysqli_query($conn,$query);
        echo "Donnees de ma table<br>";
        while($row=mysqli_fetch_assoc($result)){
                echo "Nom:".$row["Nom"]." Prenom:".$row["Prenom"]." Age:".$row["Age"]." mail:".$row["Adressemail"]."<br>";
        }
       ?>
       </p>
    </body>
</html>
```

on a cet affichage :

```
                                                    Donnees mysql sans PDO

connexion réussi
Donnees de ma table
Nom:Dj Prenom:Feriel Age:22 mail:.fr
Nom:kh Prenom:wissam Age:21 mail:ghft
Nom:st Prenom:Clothilde Age:22 mail:@gmail
```

2) Exemple (Postgresql) avec des données MySQL sans PDO

Pour afficher la BDD de postgres, dans web1/www, nous avons crée un script index2.php qui se connecte à la BDD et affiche les tuples de la table Filles, mais sans la PDO. Voici le script :

```
<html>
    <body>
       <h1>Donnees postgres sans PDO</h1>
       <p>
        <?php
        $host = "localhost";
        $user = "user2";
        $password = "user2";
        $bdname = "BDD";
        $cn = pg_connect("host=$host dbname=$bdname user=$user password=$password");
        if (!$cn) {
            die("Erreur " . pg_last_error());
        }
        $sql = 'select * from Filles';
        $result = pg_query($cn, $sql);
        // Traitement des données récupérées
        echo "Donnees de ma table<br>";
        while($row=pg_fetch_assoc($result)){
            echo "Nom:".$row["Nom"]." Prenom:".$row["Prenom"]." Age:".$row["Age"]." mail:".$row["Adressemail"]."<br>";
        }
        // Fermeture de la connexion à la base de données PostgreSQL
        pg_close($cn);
        ?>
       </p>
    </body>
</html>
```

Pour conclure cette partie, l'installation du serveur LAMP est terminer et totalement fontionnel.

IV. Sécurité

A. Filtrage

Maintenant que tous les services communiquant avec l'extérieur, sont configurés on peut commencer à s'interroger sur la sécuriter globale de notre installation. A part les mots de passe rien n'est garantit. Plutôt nous avons définitt des règles de routage mais que pour le réseau d'interconnexion. L'un des compléments au routage est le filtrage, il nous permettra d'accepter ou de refuser certaine connexion, et de donner internet à notre réseau privé.

Nous allos donc devoir écrire des règles de filtrage qui sont des iptables. Les iptables seront rendue persistantes dans notre système grâce à un script qui s'exécutera après le démarrage de notre serveur.

Ce script sera écrit en shell et sera dans le fichier *rc.local* et se situra dans le dossier */etc*.

On défini notre matrice de filtrage :

	Internet	Réseau IEM	Réseau privé	Serveur
Internet			http/https dns	http/https dns
Réseau IEM				ssh
Réseau privé	http/https dns			http/https dns, base de données
Serveur	http/https dns	ssh	http/https dns, dhcp base de données	

Mainteneant que la liste des services et leurs sens de direction vers une architecture ou une autre est définie on peut éditer nos règles de filtrage dans le fichier *rc.local.*

```
#iptables

#on supprime les règles préexistante
iptables -F

#on défini notre boucle par défaut
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP

#on autorise la boucle interne
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

#le serveur peuvent utiliser internet avec les protocoles http/https
iptables -A FORWARD -p tcp --dport http -s 172.31.20.111/24 -j ACCEPT
iptables -A FORWARD -p tcp --dport https -s 172.31.20.111/24 -j ACCEPT

#les postes du réseau privé peuvent utiliser internet avec les protocoles http/https
iptables -A FORWARD -p tcp --dport http -s 10.128.1.0/24 -j ACCEPT
iptables -A FORWARD -p tcp --dport https -s 10.128.1.0/24 -j ACCEPT

#on laisse normalement passer DNS pour tous
iptables -A FORWARD -p tcp --dport domain -j ACCEPT
iptables -A FORWARD -p udp --dport domain -j ACCEPT

#on accepte les paquets en retour
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

#iptables pour l'internet sur le seveur (masquerading)
iptables -t nat -A POSTROUTING -s 172.31.20.111/24 -j MASQUERADE

#iptables pour l'internet sur le reseau privee (masquerading)
iptables -t nat -A POSTROUTING -s 10.128.1.0/24 -j MASQUERADE

#autoriser les échanges sur le port 22 en entrée
iptables -t filter -A INPUT -p tcp --dport 22 -j ACCEPT
#autoriser les échanges sur le port 22 en sortie
iptables -t filter -A OUTPUT -p tcp --sport 22 -j ACCEPT
```

B. Droits et accès spéciaux

Afin de permettre à certains utilisateurs, tels que le webmaster, de modifier la configuration d'Apache, nous avons remarqué que les trois utilisateurs que nous avons créés lors des questions précédentes, à savoir web1, web2 et web3, peuvent avoir les compétences nécessaires pour gérer les serveurs web.

Ainsi, nous avons décidé de créer un groupe nommé ModifApache qui inclura nos trois utilisateurs.

Cependant, nous avons constaté que la commande « groupadd » qui permet la création d'un groupe sur Linux n'est pas disponible sur notre machine car en consultant la variable $PATH qui est une variable d'environnement de type Unix qui permet de déterminer les chemins de recherche des commandes.

En exécutant cette variable avec la commande echo $PATH nous avons constater que le chemin /usr/sbin manque dans le résultat de la commande

Afin de l'ajouter nous avons exécuter la commande export.

```
root@A11filles:/# echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
root@A11filles:/# export PATH=$PATH:/usr/sbin
root@A11filles:/# echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/usr/sbin
root@A11filles:/#
```

Une fois que le groupe a été créé, nous avons édité le fichier de configuration d'Apache pour changer la directive «/var/www» en remplaçant «all» par «group» dans la directive «require». Cela permet de définir les droits d'accès des utilisateurs et des groupes pour accéder aux fichiers dans le répertoire de destination.

```
<Directory />
        Options FollowSymLinks
        AllowOverride None
        Require all denied
</Directory>

<Directory /usr/share>
        AllowOverride None
        Require all granted
</Directory>

<Directory /var/www/>
        Options Indexes FollowSymLinks
        AllowOverride None
        Require group modifAppache
</Directory>
```

V. Sauvegarde

A. Sauvegarde de /etc

Pour mettre en place une sauvegarde régulière du fichier /etc du serveur vers le client, nous avons utilisé la commande rsync. Cette commande permet de synchroniser les fichiers nécessaires entre le serveur et le client, tout en conservant les attributs et les permissions de chaque fichier.

Pour cela, nous avons créé un dossier nommé «sauvegarde» dans le répertoire /var/backups, qui est utilisé pour stocker des sauvegardes périodiques des fichiers système importants tels que les configurations des paquets et des alternatives du système de gestion de paquets dpkg. Nous avons également créé un script shell à l'intérieur de ce dossier pour assurer la sauvegarde régulière du fichier /etc du serveur vers le client en utilisant la commande rsync.
Voici le contenu du script:

```
  GNU nano 5.4                  script.sh
#!/bin/bash
rsync -a /etc /etudiant@10.128.1.2:Documents
```

Cette commande va copier tous les fichiers et dossiers du répertoire «/etc» de notre serveur vers le répertoire «Documents» de l'utilisateur «etudiant» de la machine cliente ayant l'adresse IP 10.128.1.2 .

Ensuite, pour assurer cette sauvegarde de manière régulière, nous avons édité le fichier «crontab file» en utilisant la commande crontab -e, puis ajouté la ligne suivante :

```
# For more information see the manual pages of crontab(5) and c
#
# m h  dom mon dow   command

0 15 * * * /var/backups/sauvegarde/script.sh
```

B. Sauvegarde des bases de données

Pour assurer la tolérance aux pannes et la sauvegarde de nos données critiques, nous avons mis en place une stratégie de sauvegarde régulière de nos bases de données MySQL à l'aide d'un script. En automatisant ce processus, nous pouvons garantir que toutes nos bases de données sont sauvegardées de manière cohérente et régulière. Nous avons créé un utilisateur spécifique nommé « backup » qui est chargé de cette tâche.

Pour réaliser cette sauvegarde on s'est connecté à notre serveur de base de données, ensuite on a créé l'utilisateur '**backup'@'localhost**' ce dernier agira uniquement depuis le serveur local (depuis le script qui sera situé sur le serveur).

```
MariaDB [(none)]> CREATE USER 'backup'@'localhost' IDENTIFIED BY
 '';
```

On donne uniquement les droit de lecture à l'utilisateur «backup» sur toutes les bases de données :

```
MariaDB [(none)]> GRANT SHOW DATABASES, SELECT, LOCK TABLES, REL
OAD ON *.* to 'backup'@'localhost' ;
```

On demande au serveur de relire la table des droits pour s'assurer que les commandes prennent bien effet :

```
MariaDB [(none)]> FLUSH PRIVILEGES ;
```

Pour notre script de sauvegarde on a fait en sorte qu'il automatise le listage, la lecture et la sauvegarde de chaque base de données,il nous a fallut d'abord créer un répertoire temporaire afin de stocker les données dans /save_BD.
Pour cette sauvegarde nous utiliserons *mysqldump*. C'est un utilitaire qui permet d'exporter une base ou un groupe de bases vers un fichier texte .
Ensuite, nous avons créé notre script en y incluant le contenu suivant :

```
  GNU nano 5.4                backup_script.sh
#!/bin/bash
# On liste nos bases de données
LISTEBDD=$( echo 'show databases' | mysql -u backup --password=>
for BDD in $LISTEBDD do
# Exclusion des BDD information_schema , mysql et Database
if [[ $BDD != "information_schema" ]] && [[ $BDD != "mysql" ]] >
# Emplacement du dossier ou nous allons stocker les bases de do>
  CHEMIN=/home/user/save_BD/$BDD
# On backup notre base de donnees
  mysqldump -u backup --single-transaction --password= $BDD > ">
  echo "|Sauvegarde de la base de donnees $BDD.sql ";
fi
done
```

Ensuite, pour assurer cette sauvegarde de manière régulière, nous avons édité le fichier « crontab file » en utilisant la commande crontab -e, puis ajouté la ligne suivante :

```
# m h  dom mon dow   command
00 4 * * * /save_BD/backup_script.sh
```

VI. Authentification et Partage de ressources

A. Prérequis sur LDAP et SAMBA

1) LDAP

LDAP comme son nom complet l'indique (Lightway Directory Access Protocol) est un protocole supportant un service d'annuaire. Il s'agit d'une base de données et de moyens d'acceder aux informations qu'elle contient.

Les composantes de LDAP les plus importantes :

- *slapd* qui est le démond de OpenLDAP
- OpenLDAP qui supporte les moteurs de base de données
- Des utilitaires comme : ldapadd et ldap modify qui permettent d'ajouter ou modifier des entrées de l'annuaire, ldapsearch qui permet d'effectuer une recherche dans l'anuaire, ldappasswd pour modifier le mot de passe d'une entrée.
- Ces fichiers de configuration se situe dans */etc/ldap*

Les annuaires LDAP sont des arborescences logique et leur racine est généralement une construction correspondant au nom de domaine sur lequel on veut l'implanter, exprimé dans un format : dc=exemple,dc=com.

Chaque composante du nom de domaine devient la valeur d'un attribut dc (domain component) et ils sont tous regroupes dans une liste séparée par des virgules. Cela s'appelle la base de l'annuaire, correspondant dans ce cas exemple.com. Les noms de domaines ayant plus de deux composantes auront des attributs dc supplémentaires dans la liste (par exemple dc=research,dc=exemple, dc=com).

Les listes de paires attribut=valeur représentent la méthode employée pour faire référence a tout entrée de l'annuaire. Les espaces entre les éléments ne sont pas significatifs.

L'enregistrement provenant d'une base de données d'annuaire est sous format LDIF (LDAP Data Interchange Format). Il est organisé comme une suite de paires d'attributs et de valeurs (séparées par des deux-points).

Le schéma est le nom donné à l'ensemble de définitions des objets et des attributs qui définit la structure des entrées (enregistrements) d'une base de données LDAP. Les objets LDAP sont normalisés afin de garantir l'interopérabilité entre les différents serveurs d'annuaires. Les définitions de schémas sont stockées dans des fichiers du répertoire /etc/openldap/schema. Le paquetage OpenLDAP fournit la quasitotalité du schéma standard et vous pouvez ajoutez de nouvelles définitions si nécessaire. Pour spécifier les fichiers utilisés, on se sert d'entrées dans le fichier slapd.conf.

2) SAMBA
3)

Samba est un serveur de fichiers pour Linux gratuit compatible avec les réseaux Microsoft Windows. C'est-à-dire qu'il permet de partager les fichiers et les imprimantes d'un serveur linux avec les ordinateurs d'un réseau Microsoft Windows ou bien Linux.

Le protocole de communication permettant cette communication s'appelle SMB (*Server Message Block*).

Samba est constitué d'un serveur et d'un client, ainsi que de quelques outils permettant de réaliser des services pratiques ou bien de tester la configuration.

- Le serveur est constitué de deux applications (appelées *démons*) :
  - *smbd,* noyau du serveur, fournissant les services d'authentification et d'accès aux ressources
  - *nmbd*, permettant de montrer les services offerts par Samba (affichage des serveurs Samba dans le voisinage réseau, ...)
- le client: *smbclient* est un client pour linux fournissant une interface permettant de transférer des fichiers, accèder à des imprimantes
- *smbtar*: permettant d'effectuer un transfert de ou vers un fichier TAR sous linux
- *testparm* vérifiant la syntaxe du fichier *smb.conf*, le fichier de configuration de Samba

B. Installation et Configuration de LDAP

1) Instalation de LDAP

La première chose a faire est de passer en mode root pour avoir tous les droits. Toutes installations et configurations de paquets sur une machine doivent se faire en mode 'super-utilisateur'. On execute alors la commande

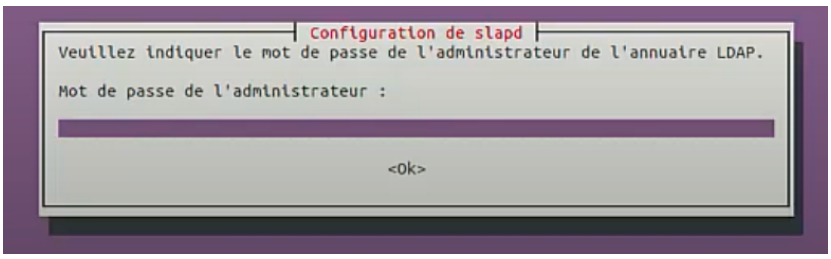
```
su -
```

et on entre notre mot de passe root.

On installe ensuite slapd et ldap-utils avec le gestionnaire d'application de notre choix avec la commande :

```
apt-get install slapd ldap-utils
```

Après cela une question nous sera posé et l'on y répondera.



2) Configuration de LDAP

Pour la suite de la configuration on se place dans le fichier */etc/ldap*.

On veut ensuite supprimé le dossier slapd.d car il est déjà préconfiguré et on ne peut pas le modifier à notre guise. Pour cela on arrête le démon slapd avec la commande :
`/etc/init.d/slapd stop`
Puis on supprime le dossier slapd.d avec la commande :
`rm -r /etc/ldap/slapd.d`

On va ensuite éditer le fichier principale de slapd qui se trouve dans */etc/ldap/slapd.* Il nous suffit de rajouter la ligne suivante :
`SLAPD_CONF=/etc/ldap/slapd.conf`
On ajoute ensuite le schema de samba pour LDAP en le copiant dans les schema de LDAP avec la commande :
`cp /usr/share/doc/samba/examples/LDAP/samba.schema /etc/ldap/schema/samba.schema`

Pour la suite de la configuration nous avons besoin d'un mot de passe hasher pour ldap pour plus de sécurité. On va donc en générer un nouveau qui nous sera donner sous forme de clé SSHA. On réalise ceci avec la commande :

```
root@A11filles:/# slappasswd
New password:
Re-enter new password:
{SSHA}zJW0RnjNNCLX3jxr4ITYFv0hLSFxWIlz
```

Nous allons ensuite configurer la base de notre annuaire avec un fichier LDIF. On créer le fichier *init.ldif* dans */etc/ldap* et on l'édit comme suit :

```
1  #définition du domaine
2  dn: dc=agence-filles,dc=dijon
3  objectclass: dcObject
4  objectclass: organization
5  o: agencefilles
6  dc: agence-filles
7
8  dn: cn=root,dc=agence-filles,dc=dijon
9  objectclass: organizationalRole
10 cn: root
11
12 #définition de l'admin
13 dn: cn=admin,dc=agence-filles,dc=dijon
14 objectClass: simpleSecurityObject
15 objectClass: organizationalRole
16 cn: admin
17 description: LDAP administrator
18 userPassword: {SSHA}zJW0RnjNNCLX3jxr4ITYFv0hLSFxWIlz
19
```

On à donc défini notre racine qui est notre domaine *agence-filles.dijon* puis l'administrateur LDAP auquel on rajoute la clé SSHA comme attribut.
Cet attribut est nécessaire car il faut le mot de passe de l'administrateur LDAP pour pouvoir intaller un contrôleur de domaine avec SAMBA. Donc combiner LDAP et SAMBA.

Ensuite il nous reste à créer et configurer le fichier *slapd.conf* que l'on situra dans */etc/ldap*. Pour cette étape un fichier nous à été fournit au préalable on ne modifira alors que les lignes suivantes :

On inclue le schema de samba.

```
# Schema and objectClass definitions
include         /etc/ldap/schema/core.schema
include         /etc/ldap/schema/cosine.schema
include         /etc/ldap/schema/nis.schema
include         /etc/ldap/schema/inetorgperson.schema
include         /etc/ldap/schema/samba.schema
include         /etc/ldap/schema/misc.schema
```

On redonne la base de notre annuaire et les informations sur son administrateur et on indique où les base de données de LDAP seront stockées sur notre machine.

```
# The base of your directory in database #1
suffix          "dc=agence-filles,dc=dijon"
checkpoint 512 60

# rootdn directive for specifying a superuser on the database. This is needed
# for syncrepl.
# rootpw        a crypter en dessous
rootdn          "dc=agence-filles,dc=dijon"
rootpw          {SSHA}zJW0RnjNNCLX3jxr4ITYFv0hLSFxWIlz

# Where the database file are physically stored for database #1
directory       "/ldap"
```

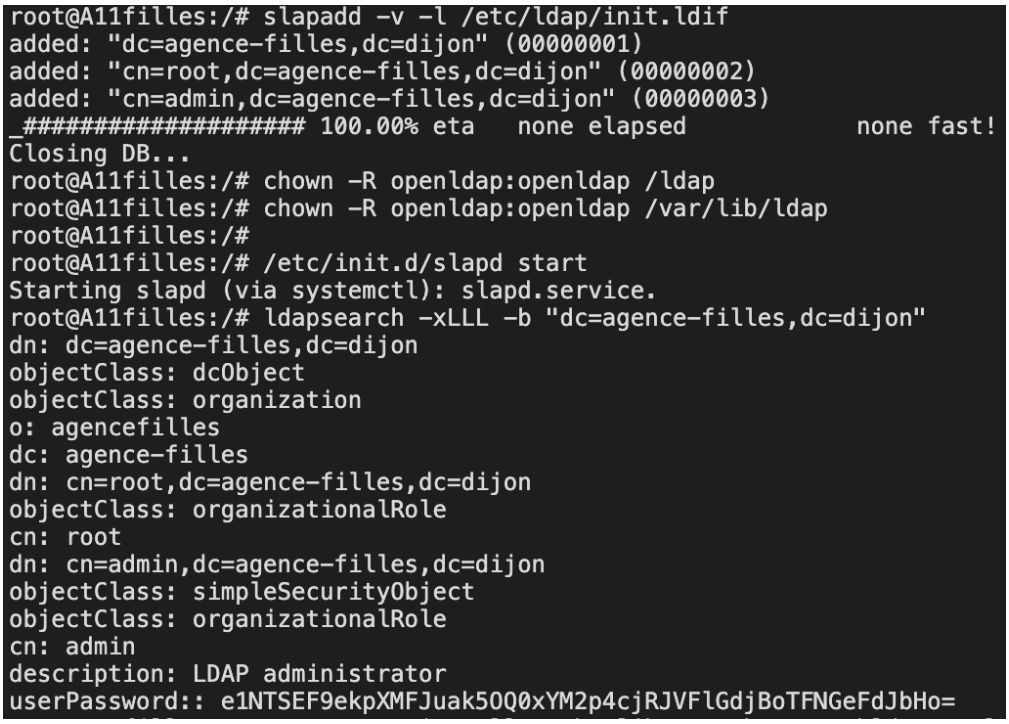
Seule l'adiministrateur peut écrire dans l'annuaire

```
# Securisation du DIT
## Par defaut l admin peux tout faire et on peut lire le LDAP
access to *
        by dn="cn=admin,dc=agence-filles,dc=dijon" write
        by * read
```

On créer ensuite un répertoire /ldap à la racine, c'est là que nis base de données pour LDAP seront stockées. On le met ici car ce fichier peut devenir volumineux avec le temps.
On entre la commande suivante pour le créer :
`mkdir /ldap`

On vérifie qu'il est bien nettoyé avec la commande :
`rm -fr ldap/*`
car si il n'est pas vide et contient de précedente base de données le service ne pourras pas redémarrer.

On ajoute une entrée dans notre annuaire qui est notre fichier *init.ldif*. On remarque que tout c'est bien passer.
On s'assure que l'utilisateur openLDAP dispose des privilèges nécessaires pour accéder à son répertoire.
Puis on redémarre et teste le service.
On peut voir les étapes décrites précedement sur cette capture d'écran.



On remarque alors que notre configuration est bonne grâce au test, car nous avons la liste des entrée que nous avons crée dans le fichier *init.ldif*.

Le service LDAP est alors fonctionnel.

C. Installation et configuration de SAMBA pour LDAP

1) Instalation de LDAP

On installe ensuite samba et smbldap-tools avec le gestionnaire d'application de notre choix avec la commande :
`apt-get install samba smbldap-tools`

2) Configuration de SAMBA pour l'utilisation avec LDAP

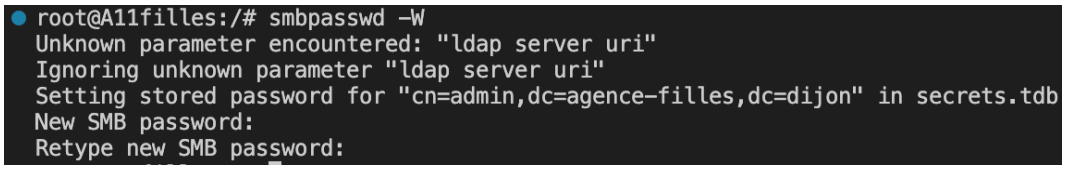
On commence par créer les répertoire SAMBA qui n'ont pas été créer automatiquement :

```
mkdir -v /var/lib/samba/profiles
chmod 777 /var/lib/samba/profiles
mkdir -v -p /var/lib/samba/netlogon
```

Puis on édite le fichier de configuration de SAMBA qui est smb.conf qui nous à été fournis au préalable. Ce fichier se situe dans */etc/samba.*

On modifie et décoche certaine ligne pour être cohérent avec ce que l'on à déclarer dans LDAP. Il y a eu beaucoup de modification dans ce fichier je conseille donc d'aller le regarder.

On enregistre un mot de passe passe pour samba :

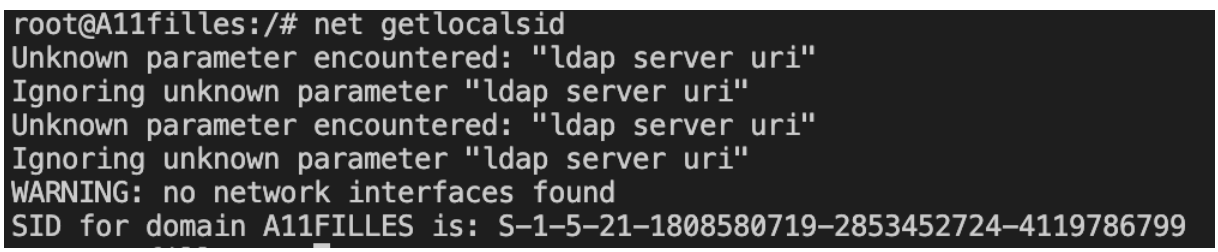


Puis on redémarre le service et nous avons un retour console qui nous dit que le service est bien activé.

Il nous faut à présent configurer smpldap-tools qui nous servira à gérer les diverses entités samba dans le contexte LDAP.

On supprime les anciens et ajoute les nouveaux fichiers de configuration *smbldap.conf* et *smbldap_bind.conf* dans *smbldap-tools.*

On va tout d'abord commencer à chercher à obtenir notre SID (Security Identifier) qui est un identificateur de sécurité unique utilisé pour identifier de manière unique les objets de sécurité tels que les utilisateurs, les groupes et les ordinateurs.



Puis on édite le fichier *smbldap.conf*, et on le modifie en cohérence avec les étapes précédante et c'est là que notre SID nous est demandé.

On édite le fichier *smbldap_bind.conf :*

```
##############################
# Credential Configuration #
##############################
# Notes: you can specify two differents configuration if you use a
# master ldap for writing access and a slave ldap server for reading access
# By default, we will use the same DN (so it will work for standard Samba
# release)
# Donne les pass pour modifier le Ldap avec les smbtools
slaveDN="cn=agence-filles,dc=info,dc=dijon"
slavePw="ldapfilles"
masterDN="cn=admin,dc=agence-filles,dc=dijon"
masterPw="ldapfilles"
```

On va ensuite appliquer les bonnes permissions à ces deux fichiers de configuration avec les commandes :

```
chmod 0644 /etc/smbldap-tools/smbldap.conf
chmod 0600 /etc/smbldap-tools/smbldap_bind.conf
```

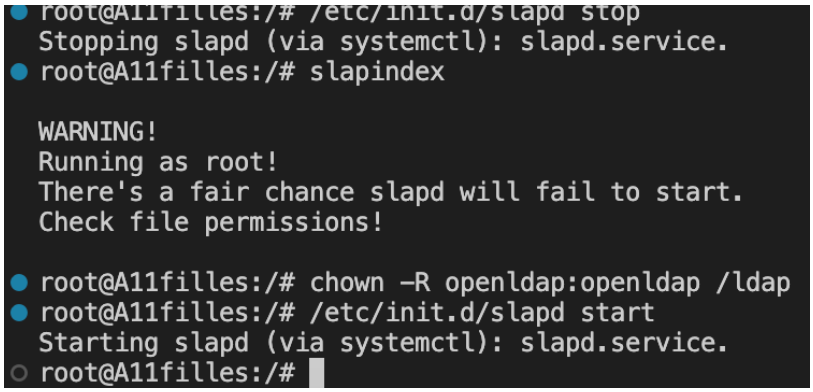
On peuple ensuite notre annuaire avec les entrées essentielles à SAMBA avec `smbldap-populate`

```
root@A11filles:/# smbldap-populate
Populating LDAP directory for domain AGENCEFILLES (S-1-5-21-1808580719-2853452724-4119786799)
(using builtin directory structure)

Use of uninitialized value $prefix in substitution (s///) at /usr/sbin/smbldap-populate line 175.
Use of uninitialized value $prefix in split at /usr/sbin/smbldap-populate line 178.
entry dc=agence-filles,dc=dijon already exist.
adding new entry: ou=Personnels,dc=agence-filles,dc=dijon
adding new entry: ou=Fonctions,dc=agence-filles,dc=dijon
adding new entry: ou=Machines,dc=agence-filles,dc=dijon
entry sambaDomainName=AGENCEFILLES,dc=agence-filles,dc=dijon already exist. Updating it...
adding new entry: uid=root,ou=Personnels,dc=agence-filles,dc=dijon
adding new entry: uid=nobody,ou=Personnels,dc=agence-filles,dc=dijon
adding new entry: cn=Domain Admins,ou=Fonctions,dc=agence-filles,dc=dijon
adding new entry: cn=Domain Users,ou=Fonctions,dc=agence-filles,dc=dijon
adding new entry: cn=Domain Guests,ou=Fonctions,dc=agence-filles,dc=dijon
adding new entry: cn=Domain Computers,ou=Fonctions,dc=agence-filles,dc=dijon
adding new entry: cn=Administrators,ou=Fonctions,dc=agence-filles,dc=dijon
adding new entry: cn=Account Operators,ou=Fonctions,dc=agence-filles,dc=dijon
adding new entry: cn=Print Operators,ou=Fonctions,dc=agence-filles,dc=dijon
adding new entry: cn=Backup Operators,ou=Fonctions,dc=agence-filles,dc=dijon
adding new entry: cn=Replicators,ou=Fonctions,dc=agence-filles,dc=dijon

Please provide a password for the domain root:
Changing UNIX and samba passwords for root
New password:
Retype new password:
```

On stope le serveur LDAP, éxecute slapindex afin de forcer la mise à jour des index par rapport à la configuration, et on redémarre le serveur LDAP:



3) Test d'ajout d'un utilisateur

Le répertoire home va accueillir les répertoires des utilisateurs. Mais pour le moment, le répertoire ne doit contenir que le premier utilisateur créé lors de l'installation de l'OS (filles), nos utilisateur web (web1, web2, web3) et test1. Mais aucun compte n'est présent dans la base de données LDAP. On va prendre l'exemple de Marie Dupont pour la création d'un compte.

On commence par ajouter l'utilisateur :

```
smbldap-useradd -a -m -M mdupont -c "Marie Dupont"
                                                    mdupont
```

-a : définit le compte SAMBA (et UNIX)
-m : créer le répertoire utilisateur (si il n'existe pas déjà)
-M : définit le nom comme partie de l'adresse email
-c : créer le nom complet
mdupont et le login utilisé pour la connexion

Pour que la création du compte soit complète, il faut créer un mot de passe.

```
smbldap-passwd mdupont
```

(qui sera : chocolat)

On peut alors créer les groupes que l'on souhaite avoir avec la commande :

```
smbldap-groupadd nom_du_groupe
```

Par exemple on peut créer un groupe *Administrateur* ou on y mettra root et Marie Dupont.

```
smbldap-groupmod -m root Administrateur
smbldap-groupmod -m mdupont Administrateur
```

L'installation et la configuration de SAMBA pour l'utilisation de LDAP est alors terminer et fontionnelle.

Conclusion

Pour conclure, nous pouvons dire que ce projet est dans l'ensemble réussi car tout ce que nous voulions intaller et configurer fonctionne bien.