

Syllabus Admin Infra

BINV3180-1

olivier.choquet@vinci.be

10 novembre, 2023 10 :50

Table des matières

1	Organisation du cours	5
1.1	Objectifs du cours	5
1.2	Supports du cours	5
1.3	Organisation	6
1.4	Evaluation	6
1.4.1	Partie Théorique	6
1.4.2	Partie Pratique	6
2	Réseaux et Infrastructure	7
2.1	Objectifs de ce chapitre	7
2.2	Introduction	7
2.3	Couche 1 : Matériel	9
2.4	Couche 2 : Liaison de données	9
2.4.1	VLAN	9
2.4.2	STP	11
2.5	Couche 3 : Réseau	12
2.5.1	Adresses spéciales	12
2.5.2	Mode d'envoi	13
2.5.3	Commandes réseau	13
2.5.4	ARP	13
2.5.5	Découpage en sous-réseaux	14
2.5.6	Table de routage	15
2.5.7	Passerelle par défaut	15
2.5.8	Routage IP	16
2.6	Couche 4 : Transport	16
3	Services réseaux (DHCP-DNS-NAT)	17
3.1	Objectifs de ce chapitre	17
3.2	Introduction	17
3.3	DNS	17
3.3.1	Configuration DNS	18
3.3.2	Résolution d'un nom	18
3.3.3	Point de vue client	19
3.4	DHCP	20
3.5	NAT	21
3.5.1	Source NAT (SNAT)	22
3.5.2	Port forwarding	22
3.5.3	Configuration NAT - Résumé	23

4	Installation Serveurs Linux et Windows	24
4.1	Objectifs de ce chapitre	24
4.2	Introduction	24
4.3	Distributions	24
4.4	Licences	24
4.4.1	Logiciel Libre – GPL	24
4.4.2	LGPL	25
4.4.3	Licence BSD	25
4.4.4	Licence MIT	25
4.4.5	Remarques	25
4.5	RAID	25
4.6	Installation Linux	26
4.6.1	Partitionnement	26
4.6.1.1	LVM	26
4.6.1.2	Systèmes de fichiers	27
4.6.1.3	Montage des partitions	27
4.6.1.4	Chiffrement des partitions	28
4.6.2	Amorçage	28
4.7	Installation Windows (Windows Server 2016)	28
4.7.1	Configuration de base	29
5	Administration Linux (Debian)	29
5.1	Objectifs de ce chapitre	29
5.2	APT	29
5.2.1	Commandes APT	29
5.3	SSH	30
5.3.1	Fonctionnement	30
5.3.2	Installation	30
5.3.3	Configuration	30
5.3.4	Utilisation	30
5.3.5	Sécurité	31
5.3.6	Copie de fichiers	32
5.3.7	Authentification par clé sur un serveur Linux	32
5.3.8	Tunnel SSH	33
5.3.9	Concrètement cela sert à quoi?	34
5.4	Gestion des utilisateurs	35
5.4.1	adduser-deluser-addgroup-delgroup	35
5.4.2	SU	35
5.4.3	SUDO	36
5.4.3.1	Fonctionnement	36
5.4.3.2	Installation	36
5.4.3.3	Utilisation	36
5.4.3.4	Avantages de SUDO	36
5.5	Passwd	36
5.6	SystemD	36
6	Serveurs Web (Apache)	38
6.1	Objectifs du chapitre	38
6.2	Introduction	38
6.3	Étapes de déploiement d'un site Web	38
6.4	Installer Apache	39
6.5	Caractéristiques d'Apache	39
6.6	VirtualHost	39
6.6.1	Directives	40

6.7	Reverse proxy	40
6.8	Site PHP	41
6.9	Apache et HTTPS	41
6.9.1	Installation	41
6.9.2	Création d'un certificat autosigné	41
6.9.3	Let's Encrypt	42
6.10	Load balancing & Apache	42
6.10.1	Installation	42
7	Partage et accès réseau	43
7.1	Objectifs du chapitre	43
7.2	NFS	43
7.2.1	Installation du serveur NFS	43
7.2.2	Configuration des partages NFS	43
7.2.3	Fonctionnement NFS	43
7.2.4	Exemples exports NFS	43
7.2.5	Exemples client NFS	44
7.2.6	Vérification	44
7.3	SAMBA	44
7.3.1	Installation du serveur SAMBA	44
7.3.2	Configuration du serveur SAMBA	44
7.3.3	Partage public	44
7.3.4	Partage en écriture	45
7.3.5	Partage des "homedirs"	45
7.3.6	Lancer/redémarrer SAMBA	45
7.3.7	Outils / Infos utiles	45
7.4	VPN	46
7.4.1	Classification VPN	46
7.4.2	VPN en images	46
7.4.3	Protocoles VPN	47
7.4.4	Exemple OpenVPN	48
7.5	FTP	48
7.5.1	Modes	49
7.6	Terminal Server	49
7.7	Autres solutions d'accès réseau	49
8	Annuaire et Authentification	49
8.1	Objectifs du chapitre	49
8.2	Modèle de nommage	49
8.3	Modèle fonctionnel	50
8.3.1	Base, Portée et Filtres	50
8.4	Modèle d'informations	51
8.5	Modèle de sécurité	51
8.6	Modèle de replication	51
8.7	LDAP vs SGBD	51
8.8	Active Directory	52
8.8.1	Unité d'organisation	53
8.8.2	Groupes de sécurité	53
8.8.3	Permissions	54
8.8.3.1	Permissions NTFS	54
8.8.3.2	Permissions Partage réseau	54
8.9	GPO	55
9	Gestion des données	55

9.1	Objectifs du chapitre	55
9.2	Base de données	55
9.2.1	DBA	56
9.2.2	Exemple PostgreSQL	56
9.2.2.1	Installation de PostgreSQL	56
9.2.2.2	Configuration de PostgreSQL	56
9.2.2.3	Sauvegarde PostgreSQL	56
9.2.2.4	Performance PostgreSQL	57
9.2.2.5	Réplication PostgreSQL	57
9.3	Sauvegardes	57
9.3.1	Que faut-il sauvegarder ?	57
9.3.2	Sur quel support sauvegarder ?	58
9.3.3	Quel type de sauvegarde, quelle fréquence ?	58
9.3.4	Conservation des données ?	58
9.3.5	Outils de sauvegarde	58
9.3.6	La règle du 3-2-1	58
9.4	Quotas	59
10	Virtualisation	59
10.1	Objectifs du chapitre	59
10.2	Avantages de la virtualisation	59
10.3	Types de virtualisation	60
10.4	Inconvénients de la virtualisation	60
10.5	Hyperviseurs	60
10.6	Virtualisation du stockage	61
10.7	Virtualisation du réseau	61
10.8	Le cas VirtualBox	61
10.8.1	Types de réseau sous VBOX	63
10.8.1.1	NAT	63
10.8.1.2	Pont / Bridge	64
10.8.1.3	Host-only networking	64
11	Conteneurs (Docker)	64
11.1	Objectifs du chapitre	64
11.2	Docker	65
11.2.1	Introduction	65
11.2.2	Docker vs Virtualisation	65
11.2.3	Installation de Docker	66
11.2.4	Utilisation de Docker	67
11.2.4.1	Dockerfile et instructions principales	68
11.2.4.2	Les commandes de gestion des images Docker	69
11.2.4.3	Les commandes de gestion des conteneurs Docker	69
11.2.4.4	Registry et DockerHub	70
11.2.4.5	Bonnes pratiques Dockerfile	70
11.2.5	Couches	70
11.3	docker compose	70
11.3.1	Installation de docker-compose	71
11.3.2	Utilisation de docker-compose	71
11.3.3	docker-compose.yml	71
11.3.4	Directives intéressantes	72
11.3.5	Réseau	73
11.3.6	Commandes intéressantes	73
11.4	The Twelve Factor App	73
11.5	Kubernetes (K8s)	74

11.5.1	Introduction	74
11.5.2	Utilité	74
11.5.3	Vue générale et concepts k8s	74
12	DevOps (Ansible notamment)	76
12.1	Objectifs du chapitre	76
12.2	Introduction	77
12.3	Pipeline de développement	78
12.3.1	DevOps et les développeurs	79
12.3.2	DevOps et les sysadmins	79
12.4	Ansible	79
12.4.1	Introduction	79
12.4.2	Ansible	79
12.4.3	Installation Ansible	81
12.4.4	Utilisation des playbooks	81
12.4.5	Rôle Ansible	82
12.4.6	Variables Ansible	82
12.4.7	Collection Ansible	83
13	Cloud (Terraform notamment)	83
13.1	Objectifs du chapitre	83
13.2	Introduction Cloud	83
13.3	Caractéristiques essentielles du Cloud	83
13.4	Modèles de service	83
13.4.1	Infrastructure as a Service (IaaS)	83
13.4.2	Platform as a Service (PaaS)	84
13.4.3	Software as a Service (SaaS)	84
13.5	Architecture Multi-tenant et Cloud	85
13.5.1	Introduction	85
13.6	Terraform	86
13.6.1	Introduction	86
13.6.2	Installation Terraform	86
13.6.3	Utilisation Terraform	86
13.6.3.1	Documentation Terraform	86

1 Organisation du cours

1.1 Objectifs du cours

[Fiche UE](#)

1.2 Supports du cours

Ce syllabus est l'unique support de ce cours. Il contient la théorie à connaître pour la partie théorique de l'examen. Il contient également des les explications et commandes pour les exercices des séances pratiques. Il vous permettra donc de vous aider à résoudre les exercices proposés durant les séances théoriques et pratiques. L'objectif est donc principalement la compréhension.

Conventions du syllabus :

- *Information importante, qui mérite attention*
- **Information capitale**

Ce syllabus est disponible au format PDF et au format HTML. *Au format HTML, vous disposez d'un bouton pour copier/coller les commandes. Ceci peut s'avérer extrêmement utile pour réaliser les exercices. Je vous*

signale déjà que pour les exercices avec VirtualBox, il est préférable de se connecter en SSH (avec Putty) pour profiter de l'avantage de ce copier/coller.

Vous constituerez vos propres fiches/notes papiers tout au long du cours. Vous pourrez utiliser ces notes papiers durant la partie pratique de l'examen.

1.3 Organisation

Les séances théoriques mixeront des présentations théoriques, des exercices papiers, des préparations/exercices pour les séances pratiques ainsi que des QCM Wooclap. Il y aura des (petits) chapitres à lire dans le syllabus pour cette partie.

Les séances pratiques consisteront en des exercices sur machine. Les exercices renverront également vers ce syllabus.

1.4 Evaluation

L'évaluation de cette UE est composée de :

- 10% en évaluation continue
- 90% lors de l'examen

L'examen est intégré (partie théorique et pratique) au travers d'un examen de 3h durant les sessions de janvier et septembre.

- **Partie théorique (40%)**
 - Questions de théorie
 - Max 1h
 - Questionnaire Papier
 - Aucun support
- **Partie pratique (60%)**
 - Exercices sur machine
 - 2h
 - Syllabus disponible
 - Vos fiches papiers disponibles
 - Internet disponible
 - ChatGPT disponible si aucun blocage global au niveau département/HE Vinci

1.4.1 Partie Théorique

Cet examen comporte une partie théorique d'une durée maximum d'une 1h. Cette partie consistera en des questions de connaissance.

Les éléments en rouge dans ce syllabus, les questions Wooclap présentées durant le cours théorique ainsi que les exercices papiers constituent la matière pour cette partie d'examen.

Pour cette partie, l'objectif est de pouvoir expliquer un concept ou élément présent dans ce syllabus ainsi que son rôle. Les exercices papiers vous aideront à étayer vos explications.

Exemple de question :

Quel est le rôle et l'utilité d'ARP ?

1.4.2 Partie Pratique

Cet examen comporte également une partie pratique d'une durée maximum de 2h. Cette partie se déroulera sur machine et se composera d'exercices similaires à ceux vus durant les séances pratiques. Pour cette partie, vous pouvez disposer de vos propres notes/supports et également d'Internet.

2 Réseaux et Infrastructure

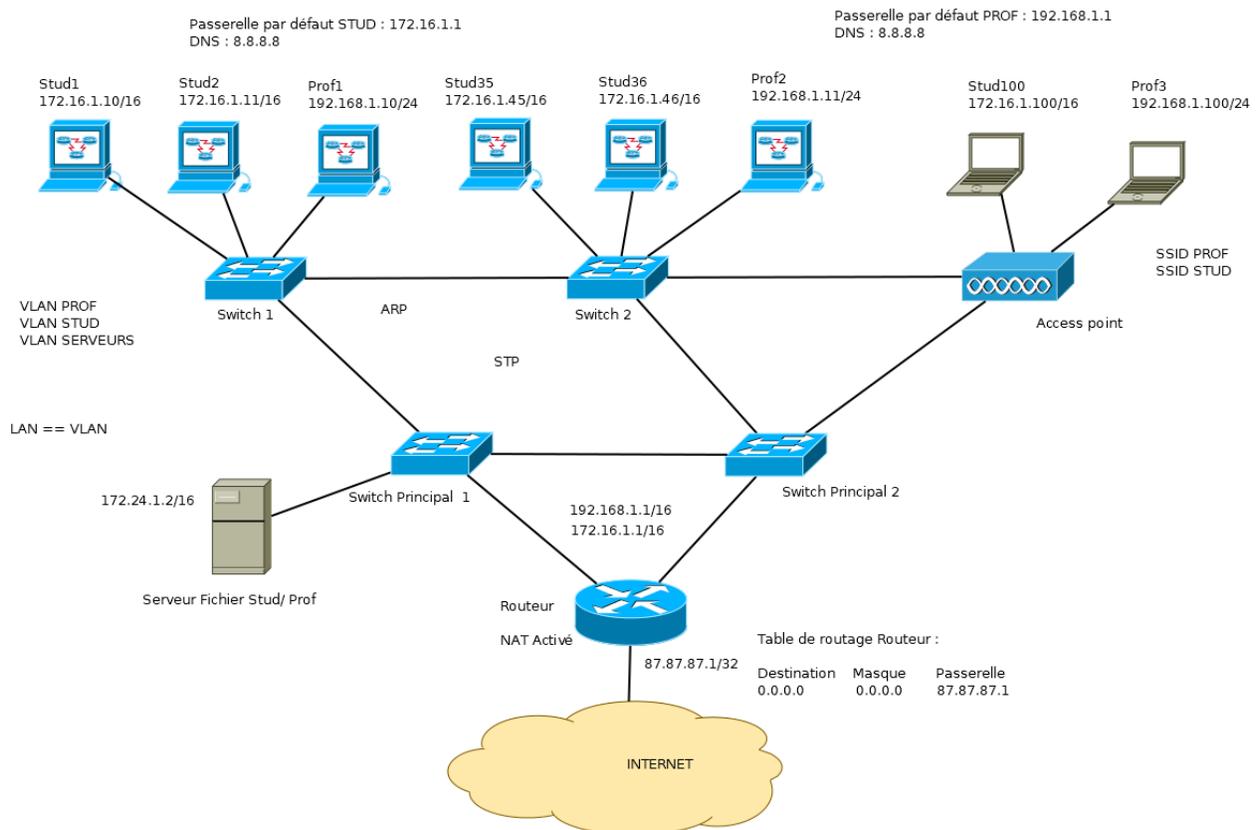
2.1 Objectifs de ce chapitre

- Connaître les différents composants matériels d'un réseau et leur rôle
- Comprendre quels éléments sont utilisés lors de l'envoi d'une requête réseau HTTP/HTTPS
- Savoir manipuler les différentes commandes de base liées au réseau
- Découper un réseau d'entreprise en sous-réseaux

2.2 Introduction

L'infrastructure d'une entreprise nécessitera le déploiement d'un réseau afin de faire communiquer les ordinateurs de l'entreprise entre eux. Typiquement, les PC des employés devront communiquer avec différents serveurs (authentification, partage fichiers ...) déployés par l'entreprise. Pour déployer un réseau d'entreprise, aussi appelé LAN, l'administrateur réseau devra implémenter les 3 premières couches du modèle OSI. Il devra également déployer des services réseaux tel que le DNS, le DHCP,

Voici un exemple de réseau d'entreprise (très fortement inspiré de l'IPL). **Vous retrouverez dans ce schéma tous les composants nécessaires à un réseau d'entreprise aussi appelé LAN.** Certains sont sans doute déjà connu pour vous (IP,DNS, ...), d'autres non (VLAN, STP, ..). Les 2 premiers chapitres de ce syllabus vont les détailler et surtout expliquer leur rôle dans ce schéma.



Voici également une capture réseau réalisé avec le logiciel open source Wireshark. Celui-ci vous permet de mieux comprendre ce qui se passe dans un réseau par exemple lorsque vous faites une requête HTTP. *Attention les écoutes réseau sont encadrées par des textes de lois. Faire des écoutes réseau sans l'accord de l'entreprise, les personnes concernées est un délit.*

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000...	192.168.1.57	52.35.83.137	TCP	55	50149 → 443 [ACK] Seq=1 Ack=1 Win=65019 Len=1 [TCP segment of a reassembled PDU]
2	0.172...	52.35.83.137	192.168.1.57	TCP	60	443 → 50149 [ACK] Seq=1 Ack=2 Win=30693 Len=0
3	3.421...	Clevo_38:92:4c	Broadcast	ARP	42	Who has 192.168.1.6? Tell 192.168.1.57
4	4.053...	Clevo_38:92:4c	Broadcast	ARP	42	Who has 192.168.1.6? Tell 192.168.1.57
5	5.056...	Clevo_38:92:4c	Broadcast	ARP	42	Who has 192.168.1.6? Tell 192.168.1.57
6	5.717...	fe80::2dca:19...	fe80::9e97:26...	DNS	91	Standard query 0xd4b2 A ochoquet.be
7	5.728...	fe80::9e97:26...	fe80::2dca:19...	DNS	2..	Standard query response 0xd4b2 A ochoquet.be A 213.186.33.4 NS dns100.ovh.net NS ns1...
8	5.730...	192.168.1.57	213.186.33.4	TCP	66	50160 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
9	5.731...	fe80::2dca:19...	fe80::9e97:26...	DNS	91	Standard query 0xf333 A ochoquet.be
...	5.739...	213.186.33.4	192.168.1.57	TCP	66	80 → 50160 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1452 SACK_PERM=1 WS=4096
...	5.740...	192.168.1.57	213.186.33.4	TCP	54	50160 → 80 [ACK] Seq=1 Ack=1 Win=262656 Len=0
...	5.740...	fe80::9e97:26...	fe80::2dca:19...	DNS	2..	Standard query response 0xf333 A ochoquet.be A 213.186.33.4 NS ns100.ovh.net NS dns1...
...	5.743...	fe80::2dca:19...	fe80::9e97:26...	DNS	91	Standard query 0xab7b AAAA ochoquet.be
...	5.751...	fe80::9e97:26...	fe80::2dca:19...	DNS	1..	Standard query response 0xab7b AAAA ochoquet.be SOA dns100.ovh.net
✓	7.071...	192.168.1.57	213.186.33.4	HT...	4..	GET /syllabusHTML/ HTTP/1.1
...	7.087...	213.186.33.4	192.168.1.57	TCP	1..	80 → 50160 [ACK] Seq=1 Ack=375 Win=2097152 Len=1452 [TCP segment of a reassembled PD...
...	7.087...	213.186.33.4	192.168.1.57	TCP	1..	80 → 50160 [ACK] Seq=1453 Ack=375 Win=2097152 Len=1452 [TCP segment of a reassembled...
...	7.087...	213.186.33.4	192.168.1.57	TCP	1..	80 → 50160 [ACK] Seq=2905 Ack=375 Win=2097152 Len=1452 [TCP segment of a reassembled...
...	7.087...	213.186.33.4	192.168.1.57	HT...	1..	HTTP/1.1 200 OK (text/html)

Frame 16: 1506 bytes on wire (12048 bits), 1506 bytes captured (12048 bits) on interface \Device\NPF_{EE7F8F6F-CEAF-47D0-872B-9E55AD2D} Ethernet II, Src: Technico_34:e9:64 (9c:97:26:34:e9:64), Dst: Clevo_38:92:4c (80:fa:5b:38:92:4c)
 Internet Protocol Version 4, Src: 213.186.33.4, Dst: 192.168.1.57
 Transmission Control Protocol, Src Port: 80, Dst Port: 50160, Seq: 1, Ack: 375, Len: 1452

```

0000  80 fa 5b 38 92 4c 9c 97 26 34 e9 64 08 00 45 00  ..[.L.. &4.d..E.
0010  05 d4 4b b6 40 00 36 06 3a ce d5 ba 21 04 c0 a8  ..K.@.6. ....1...
0020  01 39 00 50 c3 f0 c4 21 37 de 90 43 fd 69 50 10  .9.P...! 7..c.iP.
  
```

Cette capture réseau a été réalisée chez moi (petit réseau de particulier). J'ai simplement effectué une requête HTTP via un navigateur vers le site <http://ochoquet.be/syllabusHTML>. Le logiciel Wireshark propose 3 fenêtres. La première en haut avec toute la capture réseau, la deuxième en dessous où pour chaque ligne de la capture vous allez retrouver le détail par couche du modèle OSI et enfin les données brutes.

Que remarque-t-on dans cette capture ?

- un protocole du nom d'ARP fait du broadcast, il sera expliqué dans ce syllabus !
- il y a une requête DNS vers ochoquet.be -> le navigateur a besoin de l'adresse IP de ce domaine pour faire la requête HTTP vers <http://ochoquet.be/syllabusHTML>. Vous voyez des enregistrements DNS tel que A, AAAA, SOA et NS !
- on voit ensuite plusieurs requêtes TCP avec comme destination le port 80. La requête HTTP va nécessiter plusieurs segments TCP pour ramener la page HTML. Le protocole HTTP travaille sur le port 80 par défaut.

Je rappelle également le principe d'encapsulation des couches du modèle OSI. Ici la requête HTTP va être encapsulée dans de segments TCP, eux-mêmes encapsulés dans des paquets IP, eux-mêmes encapsulés dans des trames (couche 2).

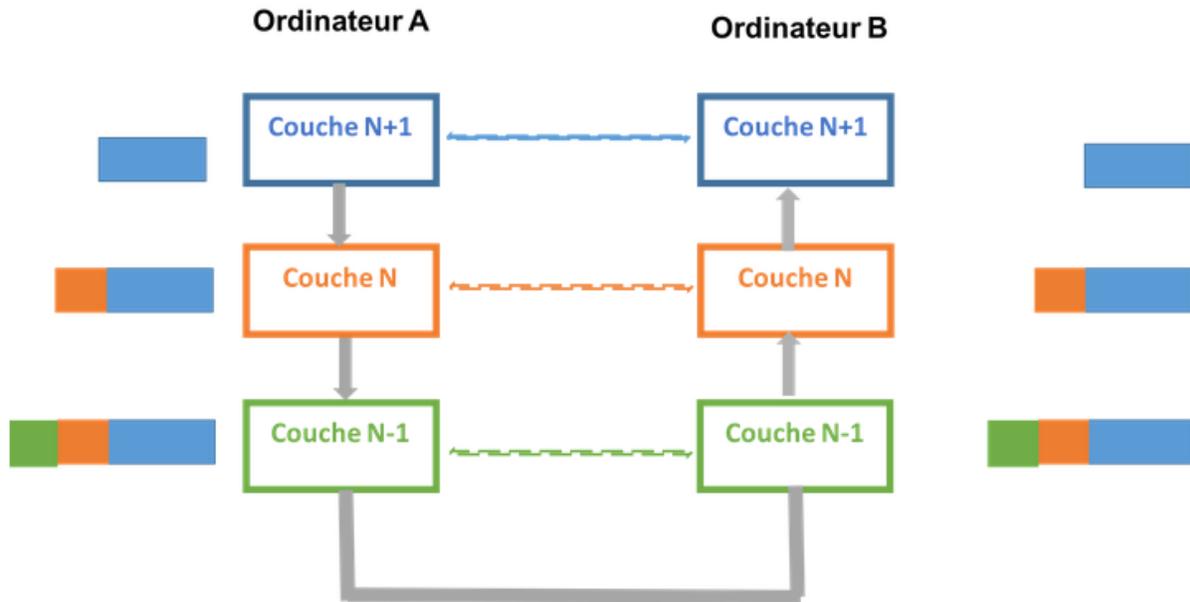


Image issue de <http://openclassroom.com>

2.3 Couche 1 : Matériel

Au niveau de la couche matérielle, différentes solutions existent pour implémenter son réseau physique. Cependant, une implémentation s'impose : **l'utilisation d'un réseau avec une topologie en étoile et des câbles Ethernet**. Ce réseau peut être renforcé avec de la fibre optique pour les liaisons les plus sollicitées. Un réseau sans fil accompagne la plupart du temps également cette architecture.

2.4 Couche 2 : Liaison de données

La couche 2 est généralement implémentée dans une infrastructure par une série de switches (commutateur en français) qui concentrent les câbles Ethernet. Ces switches, aussi appelés concentrateur, disposent d'un grand nombre de ports (24, 48...) afin d'y connecter plusieurs machines. Au niveau de la couche 2, on parle de frames (trame en français). Celles-ci permettent à des PC présents sur le même sous-réseau de dialoguer. **L'identifiant d'une machine est la MAC address**. Une trame sera donc composée d'une MAC address source, d'une MAC address destination et du message. **Le composant principal de cette couche (switch) conserve une table des MAC address permettant d'associer chaque port du switch à une MAC address**. Cette table se met à jour continuellement en fonction de l'activité réseau. Si une machine ne communique plus pendant un laps de temps, sa MAC address est retirée de la table. Cette table permet au switch d'aiguiller les trames directement vers la machine de destination. Si le switch n'a pas connaissance dans sa table de la MAC address de destination, la trame est envoyée sur tous les ports.

2.4.1 VLAN

Les VLAN (Virtual Local Area Network) permettent de séparer certains ports d'un switch par rapport à d'autres. Chaque switch disposera alors d'une table des MAC address par VLAN. L'intérêt des VLAN est de pouvoir utiliser un switch pour connecter des machines appartenant à des réseaux différents tout en garantissant la sécurité. **Les ports se verront attribués un numéro de VLAN et les ports ayant un même numéro de VLAN pourront uniquement communiquer entre eux.**

Exemple : Imaginons une entreprise avec des employés et une direction. Nous décidons de créer un sous-réseau pour chacune des populations. Sans la technique des VLAN, je devrai utiliser 2 switches (un switch pour les machines "direction" et un autre switch pour les machines "employés") tandis qu'avec les VLAN je pourrai utiliser un seul switch.

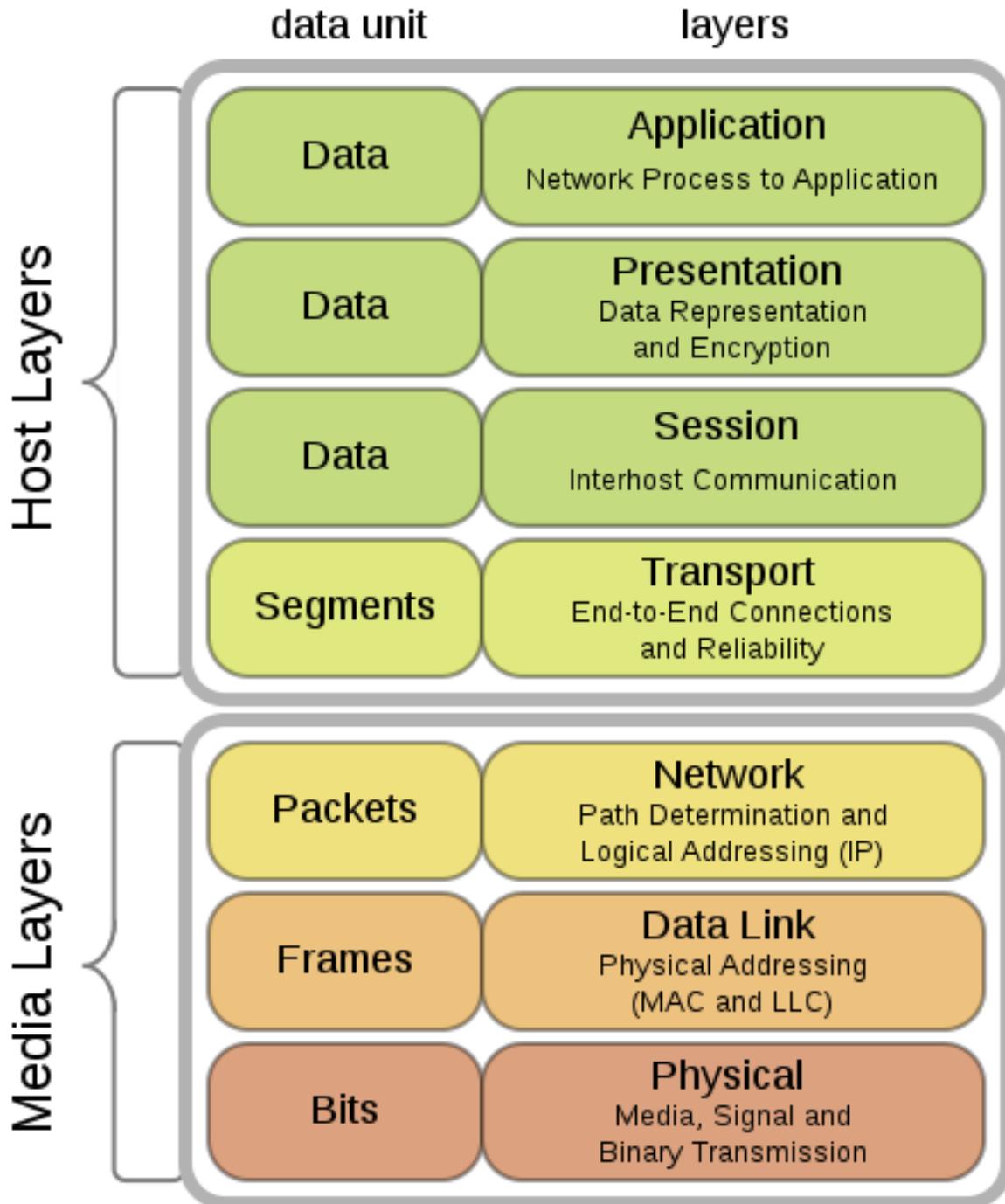


FIGURE 1 – Rappel du Modèle OSI

MAC Address	Port	Time remaining
00-50-51-6A-3E-0E	1	60
00-50-51-7A-BF-0F	2	40
00-50-51-7E-BE-EE	3	10

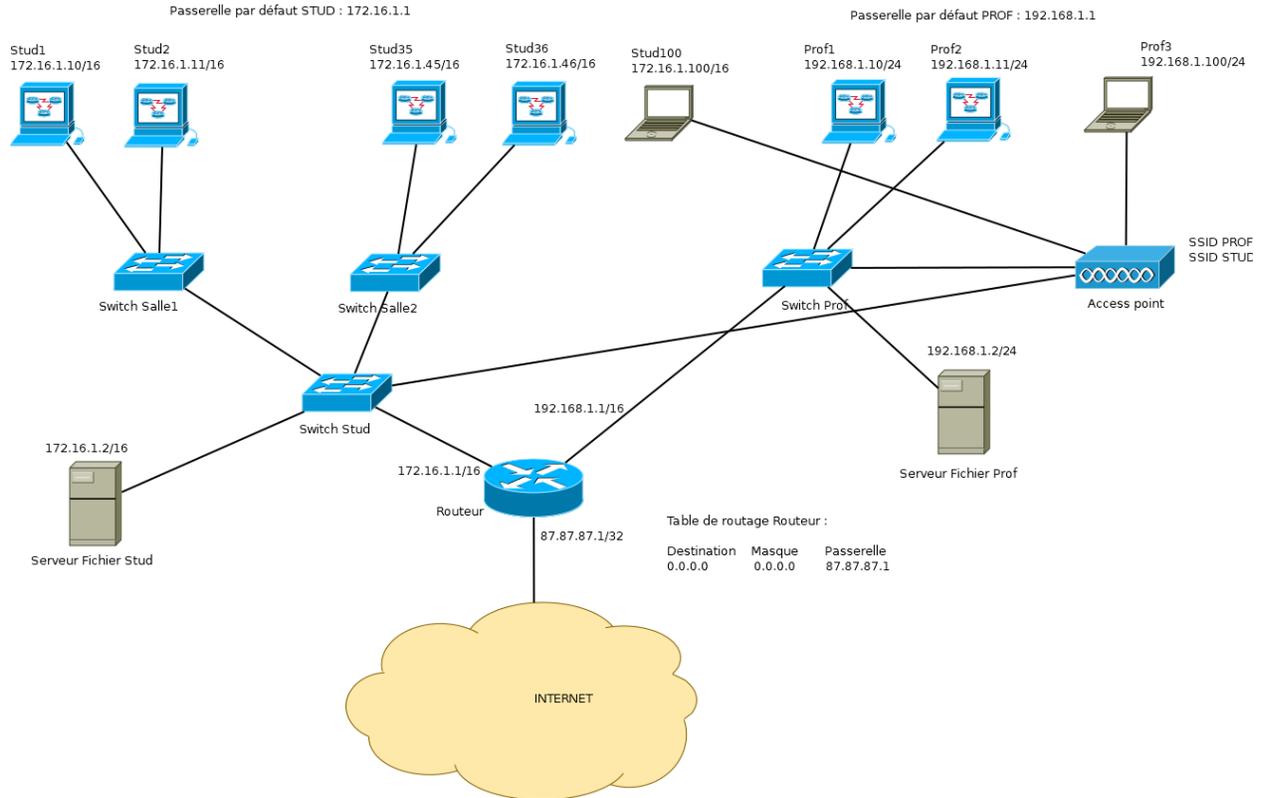
FIGURE 2 – Exemple de Table des MAC address présente dans un switch

L'intérêt des VLAN est multiple :

1. *Sécurité : séparation des sous-réseaux*
2. *Économique : on peut utiliser au maximum les ports disponibles sur les switches (on achètera moins de switch et des plus gros)*
3. *Bande passante : optimisation de la bande passante en réduisant la taille des domaines de diffusion (broadcast)*
4. *Facilité de gestion : on peut facilement changer le numéro de VLAN d'un port et donc faire basculer une machine cliente d'un sous-réseau à un autre*

2.4.2 STP

L'architecture en étoile a un point faible. Le switch, au cœur de cette étoile, peut tomber en panne et dès lors toutes les machines connectées à celui-ci deviennent inaccessibles. Mais il y a plus grave, imaginer que le switch "Stud" tombe en panne dans la [figure STP](#). Les machines des salles 1 et 2 ne peuvent plus communiquer entre elles. Pour éviter cela, on ajoutera un câble (redondance) entre les switches de la salle 1 et 2. Le problème est que vous allez créer une boucle dans le réseau et ceci n'est pas permis. C'est ici qu'intervient le protocole STP (*Spanning Tree Protocol*), il va autoriser à créer des boucles en désactivant logiquement certains liens et en les réactivant en cas de panne.



2.5 Couche 3 : Réseau

La couche réseau est implémentée dans une infrastructure par des routeurs. Ceux-ci feront transiter des paquets IP d'un réseau à un autre. L'identifiant d'une machine pour cette couche est une adresse IP. Une adresse IP est composée d'une adresse réseau et d'un masque. Le masque d'une adresse IP est noté soit sous le format A.B.C.D (Ex : 255.255.0.0) ou sous le format CIDR (Ex : /16). Actuellement l'adressage IP version 4 (IPv4) est encore largement utilisé même si l'adressage IPv6 commence à faire son apparition.

Une adresse IPv4 étant codée sur 32 bits, nous sommes limités à 4 milliards d'adresses IP pour la terre entière, ce qui est insuffisant pour couvrir tous les besoins actuels. Cette limitation impose l'utilisation d'astuces (IP privée/IP publique notamment) pour contenter tout le monde. Une adresse IPv6 est codée sur 128 bits ce qui permet d'avoir une réserve d'adresses IP suffisante. Exemple Adresse IPv6 : 8000 :0000 :0000 :0000 :0123 :4567 :89AB :CDEF ou 8000 ::123 :4567 :89AB :CDEF (suppression des 0 inutiles). Dans la suite de ce document, on s'intéressera à l'adresse IPv4 uniquement.

2.5.1 Adresses spéciales

Certaines adresses IP sont particulières. Vous connaissez peut-être déjà l'adresse 127.0.0.1 qui souvent utilisé pour la boucle locale(localhost). Il existe en fait des plages d'adresse réservées pour certains usages.

Plage	Usage
127.0.0.0/8	Boucle locale
10.0.0.0/8	Adresses privées
172.16.0.0/12	Adresses privées
192.168.0.0/16	Adresses privées
224.0.0.0/4	Multicast

Tableau à connaître

La boucle locale permet de créer un mini-réseau sur la machine permettant à l'utilisateur de tester/utiliser un service réseau sans être connecté (à un véritable réseau). Ceci permet notamment à un développeur d'utiliser un serveur Web sur sa machine de développement en l'absence de connexion réseau. **Les adresses privées sont réservées pour être utilisées dans les réseaux d'entreprises (LAN) et ne sont pas routables sur Internet** (Voir NAT). Vous devez utiliser ces adresses lorsque vous créez votre réseau d'entreprise. Les adresses multicast sont utilisées pour l'envoi de paquet en multicast (voir mode d'envoi ci-dessous).

2.5.2 Mode d'envoi

- Unicast : un à un
- Broadcast : un à tout son sous-réseau.
 - Attention les broadcasts ne passent pas les routeurs, on reste dans son sous-réseau.
- Multicast : un à un groupe
 - le protocole IGMP sera employé pour l'abonnement à un groupe
 - le matériel réseau s'occupera de dupliquer les paquets à destination

2.5.3 Commandes réseau

Linux	Windows	Explication
ifconfig / ip addr	ipconfig	Configuration / Consultation des informations IP d'une machine
route	route	Configuration / Consultation des informations de routage d'une machine
ping	ping	Tester la connectivité IP d'une machine
tracert	tracert	Voir le chemin parcouru par un paquet IP

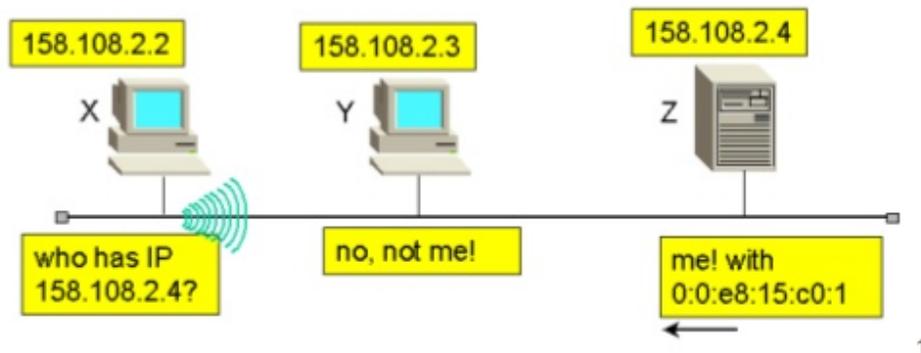
Tableau à connaître

2.5.4 ARP

Lors de l'envoi d'un message, les couches du modèle OSI vont être traversées. Dès qu'un paquet IP est arrivé au sous-réseau de destination, la couche réseau(3) passe la main à la liaison des données(2). Celle-ci, pour envoyer la trame à la machine de destination, a besoin de connaître la MAC address de la machine de destination. Or elle ne connaît ici que l'adresse IP. Le protocole ARP (Address Resolution Protocol) a été mis en place pour résoudre ce problème. **Il permet à une machine de demander l'adresse MAC d'une autre machine sur base de son adresse IP.** Une machine peut dès lors envoyer une requête ARP sur son sous-réseau avec la question suivante : "Quelle est la MAC address de la machine dont l'IP est ... ?" .La machine possédant l'IP en question répondra directement à la machine ayant posé la question. Une requête ARP sera envoyée via l'adresse broadcast et la réponse sera en unicast. À noter que chaque machine dispose d'un cache ARP. Il s'agit d'une mémoire de toutes les requêtes ARP ayant déjà été faites. Ces requêtes ne sont conservées dans le cache ARP que pendant un certain temps. L'utilité de ce cache est d'éviter d'effectuer trop de requêtes ARP (qui sont en broadcast).

Address Resolution Protocol - ARP

IP over Ethernet



D'un point de vue sécurité, le protocole ARP est sensible aux attaques de type ARP cache poisoning/spoofing. Il s'agit pour l'attaquant de répondre à toutes les requêtes ARP en vue de détourner le trafic du sous-réseau vers sa propre machine et d'effectuer des écoutes.

2.5.5 Découpage en sous-réseaux

Vous devez être capable de découper un réseau en sous-réseau. Ceci peut être demandé dans la partie pratique de l'examen. La tâche d'un administrateur système sera de découper de manière intelligente son réseau. Ce découpage permet d'isoler les réseaux pour des raisons de sécurité et d'efficacité. Il devra calculer une adresse réseau et un masque de sous-réseau. Celui-ci permettra de donner le nombre de machines maximales que le sous-réseau pourra accueillir.

Exemple : un réseau employés composé de 25 machines

1. 25 machines → recherche de la puissance de 2 supérieure à 25 → 2
2. Choix d'une adresse réseau parmi les plages d'adresses privées : 192.168.5.0 par ex.
3. Calcul du masque : 32 bits - 5 bits (puissance de 2 calculée précédemment) → /27
4. Résultat : 192.168.5.0/27

Une plage d'adresses pour un réseau débutera par l'adresse du réseau et se terminera par l'adresse de diffusion (broadcast). Entre ces 2 adresses, les autres adresses peuvent être utilisées par des machines.

Exemple : un réseau employés composé de 25 machines

1. Adresse réseau : 192.168.5.0/27
2. Adresse machine 1 : 192.168.5.1/27

3. Adresse machine 2 : 192.168.5.2/27
4. Adresse machine ... : ...
5. Adresse broadcast : 192.168.5.31/27

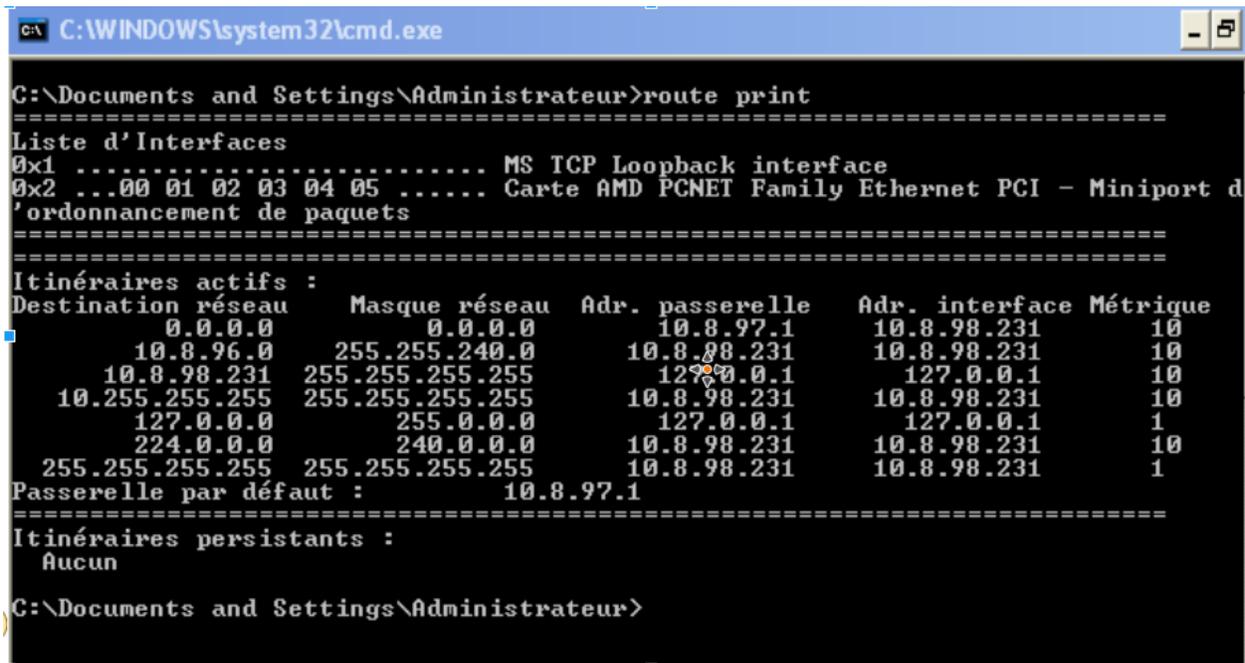
L'adresse de diffusion(broadcast) est la dernière adresse de la plage calculée. Tous les bits machines de cette adresse sont donc à 1.

Adresse réseau : 192.168.5.0/27 → 192.168.5.00011111 → 192.168.5.31/27 Astuce : une adresse réseau est toujours paire et une adresse de diffusion toujours impaire!

2.5.6 Table de routage

Le composant principal de cette couche 3, le routeur, maintient une table de routage. Vous pouvez imaginer par un routeur comme étant un carrefour routier et une table de routage comme étant les panneaux indicateurs présents au carrefour. Une table de routage est un tableau qui précise pour une destination d'un sous-réseau (adresse réseau + masque) une passerelle (adresse IP d'une machine/routeur).

Un exemple ci-dessous :



```

C:\Documents and Settings\Administrateur>route print
=====
Liste d'Interfaces
0x1 ..... MS TCP Loopback interface
0x2 ...00 01 02 03 04 05 ..... Carte AMD PCNET Family Ethernet PCI - Miniport d
'ordonnancement de paquets
=====
Itinéraires actifs :
Destination réseau      Masque réseau  Adr. passerelle  Adr. interface  Métrique
-----
0.0.0.0                 0.0.0.0       10.8.97.1        10.8.98.231     10
10.8.96.0               255.255.240.0 10.8.98.231     10.8.98.231     10
10.8.98.231             255.255.255.255 127.0.0.1       127.0.0.1       10
10.255.255.255          255.255.255.255 10.8.98.231     10.8.98.231     10
127.0.0.0               255.0.0.0     127.0.0.1       127.0.0.1       1
224.0.0.0               240.0.0.0     10.8.98.231     10.8.98.231     10
255.255.255.255        255.255.255.255 10.8.98.231     10.8.98.231     1
Passerelle par défaut :      10.8.97.1
=====
Itinéraires persistants :
Aucun

C:\Documents and Settings\Administrateur>

```

Pour chaque destination, la table de routage indique une passerelle et une métrique. À noter que les destinations sont indiquées par réseau (adresse IP + masque). Les entrées de cette table peuvent être ajoutées à la main ou de manière automatique. On parle dans le premier cas de routage statique et dans le second cas de routage dynamique. Quelques exemples de protocoles de routage dynamique : RIP, OSPF (utilisant l'algorithme de Dijkstra).

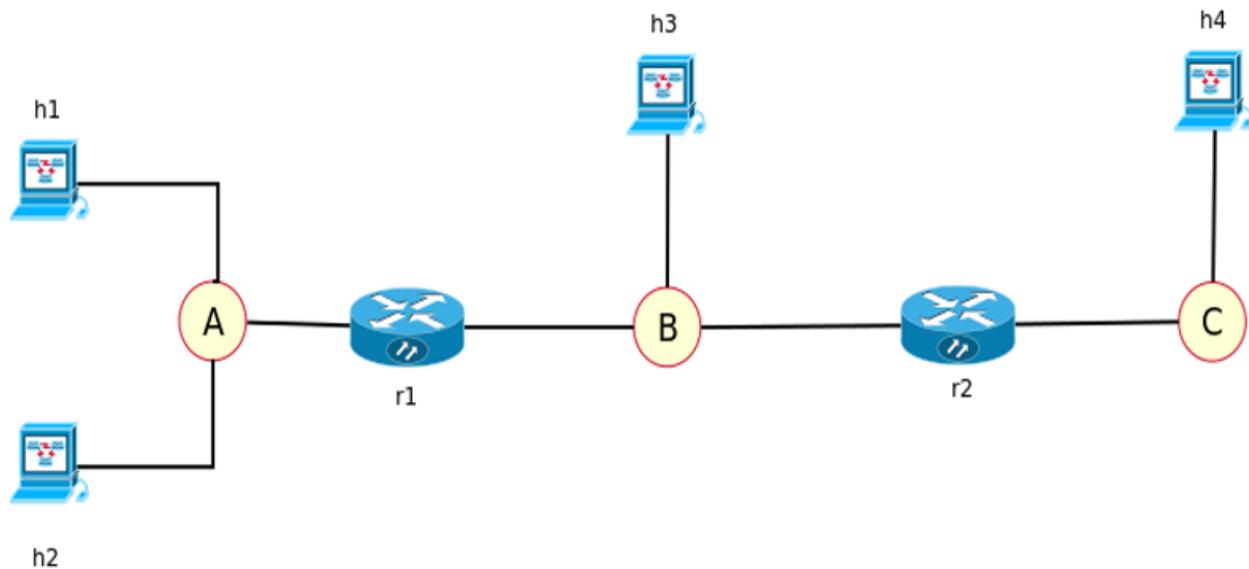
2.5.7 Passerelle par défaut

Vu qu'il est impossible pour chaque machine de constituer une table de routage avec tous les chemins possibles, chaque machine possédera une passerelle par défaut (0.0.0.0/.0.0.0.0). Celle-ci sera utilisée lorsque aucune règle plus précise ne pourra être utilisée dans la table de routage. Grâce à cette passerelle par défaut, il est également plus simple d'automatiser la connexion de machines clientes à son réseau. En effet, il suffira de fournir à chaque machine cliente une

adresse IP, un masque et une passerelle par défaut et celle-ci pourra utiliser notre réseau. Cette automatisation pourra se faire via un serveur DHCP (Voir DHCP).

2.5.8 Routage IP

Une machine a toujours accès au(x) réseau(x) auquel(s) elle est connectée directement. En définissant une adresse IP sur une machine, vous connectez directement cette machine à un sous-réseau. Il n'y a pas besoin de règle de routage supplémentaire pour que cette machine communique avec toutes les autres machines du sous-réseau. Par contre, si la machine essaie de joindre une machine ne se trouvant pas dans le même sous-réseau, celle-ci devra contacter sa passerelle par défaut (souvent un routeur) qui contient une table de routage permettant de transférer les paquets jusqu'à destination.



Dans le cas ci-dessus, h1 et h2 pourront communiquer ensemble simplement en leur définissant une adresse IP. Pour que h1 et h3 discutent, il faut au minimum :

1. une passerelle par défaut (r1) sur h1
2. une passerelle par défaut (r1) sur h3

Pour que h1 et h4 discutent, il faut au minimum :

1. une passerelle par défaut (r1) sur h1
2. une passerelle par défaut (r2) sur h4
3. une règle de routage sur r1 : pour atteindre le réseau C → passer par r2
4. une règle de routage sur r2 : pour atteindre le réseau A → passer par r1

2.6 Couche 4 : Transport

La couche 4 (Transport) va permettre une communication entre applications. Les applications respecteront le modèle d'architecture client-serveur. Cette couche 4 utilise la notion de port. **Une application sera donc identifiée via son port.** Typiquement, une application de type serveur se mettra en écoute sur un port tandis qu'une application cliente se connectera à ce serveur en précisant son adresse IP et port. Des ports par défaut ont été définis pour les applications les plus courantes :

Application	Port réservé
Web(HTTP)	80
SSH	22

Application	Port réservé
HTTPS	443
DNS	53

Tableau à connaître

Chaque machine dispose de 65536 ports. Pour ses besoins personnels, on utilisera les ports au-dessus de 1024 qui ne sont pas réservés. Deux protocoles existent au niveau de la couche 4 : TCP et UDP le premier est un transport jugé fiable car chaque envoi est acquitté (confirmation que l'envoi a été bien reçu). Par contre, UDP se veut plus simple et ne demande aucune confirmation que l'envoi a été correctement reçu. *Une politique de sécurité de base pour les administrateurs système est donc de bloquer sur le serveur tous les ports hormis ceux des applications utilisées.*

3 Services réseaux (DHCP-DNS-NAT)

3.1 Objectifs de ce chapitre

- Savoir expliquer le rôle des 3 services réseaux de base

3.2 Introduction

Pour qu'un réseau d'entreprise fonctionne correctement, il est nécessaire d'avoir au minimum 3 services à savoir :

1. **DNS** (Domain Name System)
2. **DHCP** (Dynamic Host Configuration Protocol)
3. **NAT** (Network Address Translation)

Un administrateur système doit donc veiller à ce que son réseau mette à disposition de ces clients ces 3 services.

3.3 DNS

Pour rappel, le système DNS permet de traduire un nom de domaine en une adresse IP et inversement. Ce système a été longuement décrit dans le cours d'IPP, nous ne détaillerons donc pas son fonctionnement ici. Par contre, nous allons nous attarder à l'usage d'un serveur DNS dans un réseau d'entreprise.

Pour traduire les noms de domaine Internet, il n'est pas obligatoire d'avoir son propre serveur DNS. On peut très bien utiliser le serveur DNS de son ISP (Internet Service Provider) ou encore le très célèbre DNS de Google (8.8.8.8). Dès lors, pourquoi est-il nécessaire pour un administrateur système d'installer un serveur DNS ? La raison est la suivante : *dans un réseau d'entreprise, les machines clientes porteront des noms et ces noms devront être traduits en une adresse IP.*

Une autre raison d'avoir un serveur DNS est la gestion du nom de domaine de l'entreprise. Dans ce cas, l'installation d'un serveur DNS n'est pas obligatoire. On peut très bien utiliser un serveur DNS fourni par son fournisseur de nom de domaine. *En tout cas, l'administrateur système de l'entreprise devra gérer les enregistrements DNS de l'entreprise notamment les enregistrements DNS pointant vers les serveurs de l'entreprise et qui devront être accessibles depuis l'extérieur.*

Un administrateur système installera généralement 2 serveurs DNS. Cette redondance permettra de gérer plus facilement la défaillance d'un serveur.

3.3.1 Configuration DNS

Un serveur DNS gère les enregistrements pour un nom de domaine aussi appelé zone DNS. Une zone DNS contient différents types d'enregistrements :

- **SOA** : Start of Authority (informations générales sur la zone DNS)
- **A** : Enregistrement d'un hôte (correspondance Nom → IP)
- **CNAME** : Alias
- **PTR** : Enregistrement d'une IP (correspondance (IP→ Nom)
- **MX** : Mail server (adresse IP du serveur mail de la zone)
- **NS** : Name Server (serveur DNS pour la zone)

Exemple d'une configuration d'un serveur DNS sous Linux :

```
$TTL      604800
@         IN      SOA      ns.monbeaurezo.be. emailadmin.monbeaurezo.be. (
                        2          ; Serial
                        604800     ; Refresh
                        86400      ; Retry
                        2419200    ; Expire
                        604800 )   ; Negative Cache TTL

@         IN      NS       ns.monbeaurezo.be.
@         IN      A        127.0.0.1
@         IN      AAAA     ::1
ns        IN      A        192.168.1.1
h1        IN      A        192.168.1.5
aliasH1   IN      CNAME    h1
```

3.3.2 Résolution d'un nom

Il est très important de ne pas oublier que toute résolution de noms sur une machine commence par l'inspection du fichier `hosts`. Ce fichier est présent sous Linux à cet endroit `/etc/hosts` et sous Windows à cet endroit `c:\Windows\System32\Drivers\hosts`.

Exemple de fichier `hosts` sous Windows :

```
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#       102.54.94.97       rhino.acme.com           # source server
#       38.25.63.10      x.acme.com              # x client host

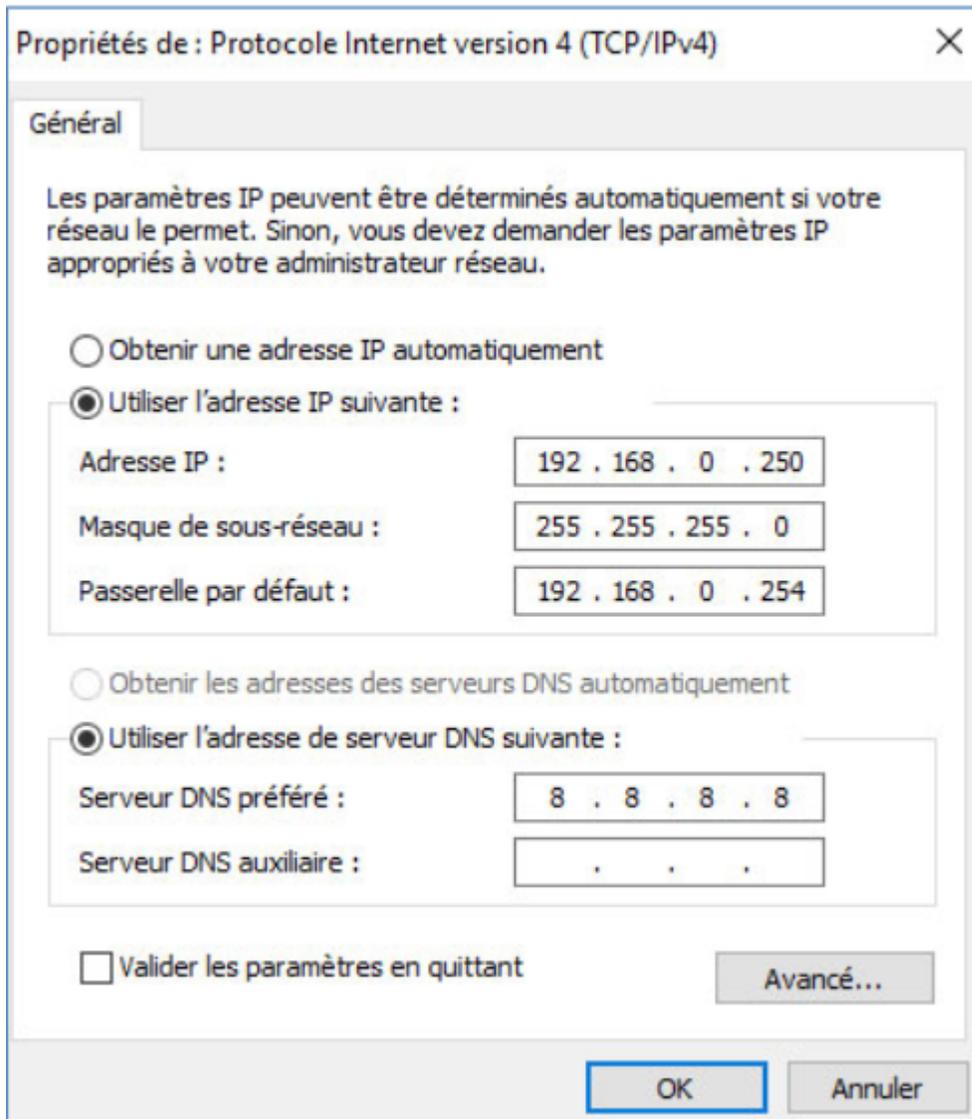
# localhost name resolution is handled within DNS itself.
# 127.0.0.1      localhost
# ::1           localhost

127.0.0.1      localhost.vdsnb.com
127.0.0.1      siteHTML
127.0.0.1      sitePHP
127.0.0.1      siteJetty
127.0.0.1      contsitephpdb
```

Les administrateurs système utilisent abondamment ce fichier pour tester des services car cela évite l'installation d'un serveur DNS. En production, un serveur DNS sera bien évidemment employé.

3.3.3 Point de vue client

Sur une machine Linux, les serveurs DNS sont enregistrés dans le fichier `/etc/resolv.conf`. Sous Windows les serveurs DNS sont renseignés dans les propriétés des paramètres IP.



Sous Windows comme sous Linux, on peut tester les réponses des serveurs DNS avec la commande nslookup.

3.4 DHCP

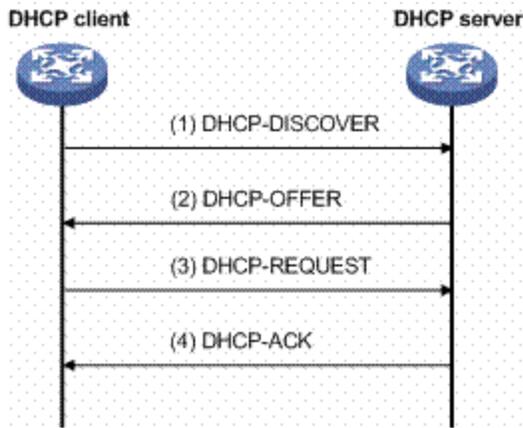
Pour qu'une machine cliente puisse se connecter à un réseau et surfer sur Internet, cette machine a besoin au minimum des informations suivantes :

- se voir attribuer une adresse IP et un masque au sein de ce réseau
- obtenir une passerelle par défaut (pour aller sur Internet notamment)
- obtenir des serveurs DNS (pour traduire les noms en adresse IP)

Un serveur DHCP permettra de répondre à ces besoins de manière automatique. Le protocole DHCP fonctionne comme ceci :

- 1) Un client envoie une requête DHCP_DISCOVER en broadcast à la recherche de serveurs DHCP
- 2) Un ou plusieurs serveurs DHCP répondent avec un DHCP_OFFER contenant au minimum une adresse IP et un masque. Le serveur DHCP peut également envoyer une passerelle par défaut, des serveurs DNS,

- 3) Le client fait la demande auprès d'un serveur DHCP parmi toutes les offres reçues.
- 4) Le serveur DHCP lui confirme qu'il peut utiliser les informations de l'offre.



Les informations reçues par le serveur DHCP ne seront valables qu'un certain temps. On parle de bail. Le client pourra évidemment renouveler son bail.

La requête DHCP_DISCOVER se fait en broadcast. Les broadcast ne passent pas les routeurs. Donc un serveur DHCP devra être présent dans chaque sous-réseau ou celui-ci devra avoir une interface réseau dans chaque sous-réseau.

Un administrateur système installera plusieurs serveurs DHCP pour se prémunir de la panne d'un serveur. Deux serveurs DHCP ne pouvant pas distribuer les mêmes adresses IP, l'administrateur système devra répartir ces adresses de manière intelligente entre les 2 serveurs.

Il est également possible de configurer les serveurs DHCP en failover pour se prémunir des pannes. Dans ce cas, les 2 serveurs se partagent la même étendue et ils peuvent travailler en ACTIF/PASSIF ou en ACTIF/ACTIF. Le mode ACTIF/PASSIF indique simplement qu'un serveur DHCP est le maître et l'autre l'esclave. C'est donc le serveur maître qui répond à toutes les demandes. Si celui-ci tombe en panne, l'esclave prend le relais. Le mode ACTIF/ACTIF indique que les 2 serveurs travaillent sur un même pied d'égalité. Ils distribuent donc tous les 2 des adresses IP et ceci sur la même étendue. Cela permet donc un équilibrage de charge. À noter que ces modes ACTIF/PASSIF ou ACTIF/ACTIF, nécessite une communication entre les serveurs DHCP et ne fonctionne donc correctement qu'avec des serveurs DHCP similaires (non hétérogènes).

Exemple de configuration d'un serveur DHCP sous Linux :

```

# fichier /etc/dhcp3/dhcpd.conf
# distribution d'adresses IP 10.0.0.0/8
subnet 10.0.0.0 netmask 255.0.0.0 {
# restriction de la distribution au range défini
range 10.0.0.10 10.0.0.250 ;
# envoi de la passerelle par défaut (gateway)
option routers 10.0.0.1 ;
# envoi serveur DNS
option domain-name-servers 10.0.0.1 ;
}
  
```

3.5 NAT

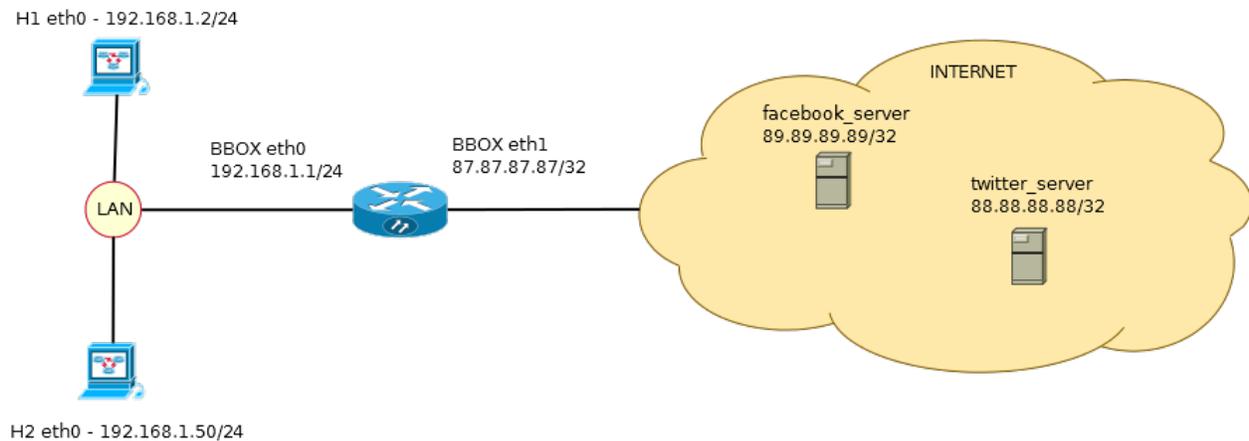
Le NAT est utilisé abondamment par les réseaux d'entreprise pour traduire les adresses IP privées de clients en adresses publiques routables sur Internet. Pour rappel, les adresses IPv4 sont limitées à 4 milliards ce qui ne permet pas de couvrir la totalité du globe terrestre. Donc certaines

plages d'adresses ont été réservées(adresses privées) pour pouvoir être attribuées dans les réseaux locaux des entreprises. Ceci est très bien, mais pour aller sur Internet, j'ai besoin d'une adresse publique autrement dit d'une adresse routable sur Internet. C'est ici qu'intervient le NAT. Vous allez comprendre comment la terre entière peut surfer sur Internet avec seulement 4 milliards d'adresses disponibles.

3.5.1 Source NAT (SNAT)

Dans ce cas, l'adresse IP source du paquet IP est traduite, "natée".

Chaque machine cliente sera connectée au réseau de l'entreprise via une adresse privée et recevra une passerelle par défaut qui sera généralement le routeur qui vous permettra d'aller sur Internet. Ce routeur disposera d'une adresse IP publique reçue par votre fournisseur d'accès à Internet et il disposera également d'une adresse privée dans le réseau de l'entreprise. Nous activerons le NAT sur le routeur ce qui aura pour effet de traduire l'adresse IP source de tout paquet IP sortant sur Internet par l'adresse IP publique du routeur.



Dans cet exemple, les machines H1 et H2 surferont sur Internet grâce à l'adresse IP publique du routeur BBOX. Il reste un petit point à éclaircir : une fois la réponse reçue à une requête envoyée sur Internet, comment le routeur va-t-il savoir que cette réponse est pour la machine H1, H2, ... ?

Le routeur va en fait identifier les machines clientes via un port choisi aléatoirement.

Exemple (requête HTTP) :

Avant NAT	Après NAT
IP Source : 192.168.1.2	IP Source : 87.87.87.1 (adresse du routeur)
IP Destination : 89.89.89.1	IP Destination : 89.89.89.1
Port Source : /	Port Source : 10527 (port aléatoire, le routeur note l'association de ce port à la machine natée à savoir 192.168.1.2)
Port Destination : 80	Port Destination : 80

Lors de la réception de la réponse, le routeur pourra alors savoir en regardant le port la machine à qui est destiné le paquet.

3.5.2 Port forwarding

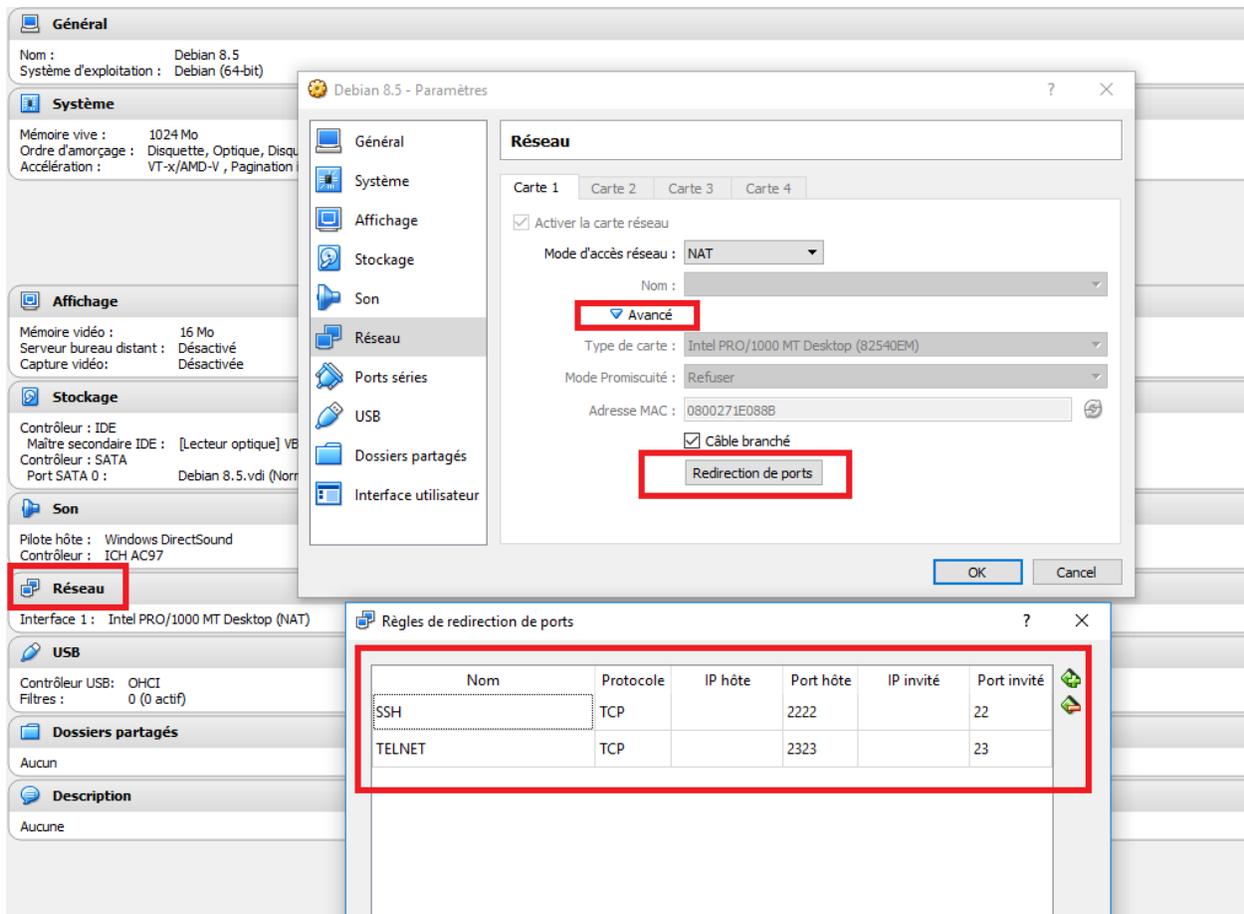
Le NAT introduit une isolation du réseau de l'entreprise. En effet, il est impossible d'atteindre une machine du réseau de l'entreprise depuis l'extérieur. La seule machine que l'on sait atteindre est le routeur. Ceci est intéressant en termes de sécurité, mais nous devons quelquefois

avoir accès à un serveur présent dans le réseau de l'entreprise. Pour ce faire, nous pouvons configurer le NAT pour qu'il fasse du port forwarding.

Le principe est le suivant : pour accéder à un serveur de l'entreprise, on attribue un port sur le routeur dédié à ce serveur. Dès que le routeur reçoit une connexion de l'extérieur sur ce port, il transfère les paquets vers l'adresse privée du serveur.

Le port forwarding est utilisé abondamment dans les réseaux surtout depuis la virtualisation. Par exemple, VirtualBox définit par défaut un réseau NAT entre la machine hôte et la machine virtuelle invitée. Si vous installez un serveur dans la machine virtuelle invitée et que vous voulez y accéder depuis l'extérieur (votre machine hôte ici), vous devrez faire du port forwarding. Ceci n'est pas compliqué, il suffit de faire une association entre un port de la machine hôte et un port la machine virtuelle.

Exemple dans VBOX :



Dans cet exemple, nous relierons le port hôte 2222 au port invité 22. Cela signifie que si vous vous connectez sur le port 2222 de votre machine hôte, vous allez être redirigé vers le port 22 de la machine virtuelle invitée. Autrement dit, si vous faites un Putty sur le port 2222 depuis votre machine hôte, vous vous connecterez en SSH(22 == SSH) sur votre machine virtuelle. Bien sûr il faut avoir installé SSH sur votre machine virtuelle au préalable.

3.5.3 Configuration NAT - Résumé

- SNAT (source NAT)
 - Usage le + fréquent : LAN → WAN

- DNAT (destination NAT → port forwarding)
 - Usage le + fréquent : WAN → DMZ, LAN

4 Installation Serveurs Linux et Windows

4.1 Objectifs de ce chapitre

- Comprendre les différents éléments qui méritent attention lors de l'installation d'un serveur

4.2 Introduction

Le travail d'un administrateur système consistera en l'installation de différents serveurs en vue de satisfaire les différents besoins des utilisateurs. Nous verrons surtout ici comment installer et configurer un serveur Linux (Distribution Debian) en vue d'y installer des services par après. L'installation d'un serveur Windows reste relativement simple. Celle-ci sera donc vue brièvement.

4.3 Distributions

Contrairement aux environnements propriétaires (Windows), le fait que le noyau Linux soit libre a permis à différentes communautés de proposer sa version de Linux. On parle de distribution Linux. On distingue différents types de distributions :

Type	Définition	Exemple
Communautaire	Maintenance et évolution du système assuré par une communauté sans lien avec une entreprise	Debian
Commerciale	Maintenance et évolution du système assuré par une entreprise	Ubuntu, Red Hat
Généraliste	Distribution multi-usage	Ubuntu, Debian
Spécialisée	Distribution à usage précis	Gparted, Raspbian
Dérivée	Distribution ayant comme base une autre distribution	Ubuntu

Chaque distribution peut évidemment cumuler différents types. L'administrateur système choisira une distribution suivant différents critères :

1. L'usage (type ci-dessus)
2. La stabilité
3. La documentation disponible / le support
4. Besoin d'une interface graphique ou pas
5. La licence (voir ci-dessous)

4.4 Licences

Faisons un petit point autour des licences. Ceci est aussi bien utile pour les développeurs que pour les administrateurs système. Il existe une variété impressionnante de licences, nous allons ici simplement nommer et expliquer brièvement les plus importantes.

4.4.1 Logiciel Libre – GPL

Le logiciel libre se définit comme un logiciel qui peut être étudié, modifié et diffusé. Cela garantit donc l'accès au code source du programme et par cette occasion de vérifier la transparence et la sécurité du programme. Le logiciel libre a été initié par Richard Stallman qui créa en 1985 la FSF (Free Software Foundation). À partir

de ce moment, de nombreux développeurs ont distribué leurs logiciels sous licence **GPL (GNU Public Licence)**.

Cette licence se caractérise par 4 libertés à respecter :

1. **La liberté d'exécuter le logiciel, pour n'importe quel usage**
2. **La liberté d'étudier le fonctionnement d'un programme et de l'adapter à ses besoins, ce qui passe par l'accès aux codes sources**
3. **La liberté de redistribuer des copies**
4. **L'obligation de faire bénéficier la communauté des versions modifiées (copyleft)**

La dernière liberté est la plus contraignante, car elle nécessite qu'un logiciel utilisant une licence GPL doit être distribué sous licence GPL. C'est pourquoi d'autres licences ont vu le jour pour mieux s'adapter aux réalités des entreprises.

4.4.2 LGPL

La licence LGPL (Lesser GNU Public Licence) reprend les fondements de la licence GPL en supprimant la restriction sur l'hérité de la licence GPL. Cette licence permet également la cohabitation de plusieurs licences (libres et propriétaires) au sein d'un logiciel. Il s'agit pour ces raisons de la licence préférée des développeurs de librairies.

4.4.3 Licence BSD

La licence BSD (Berkeley System Distribution) reprend les fondements de la licence GPL en supprimant la restriction sur l'hérité de la licence, la restriction sur la redistribution des copies. Cette licence permet également la cohabitation de plusieurs licences (libres et propriétaires) au sein d'un logiciel.

4.4.4 Licence MIT

La licence MIT (Massachusetts Institute of Technology) permet de copier, modifier, redistribuer un logiciel, mais en incluant les copyright des auteurs. Microsoft utilise cette licence dans certains de ces produits.

4.4.5 Remarques

Gratuit ne veut pas dire libre ! La gratuité implique simplement que l'on ne paye pas pour un produit. Open source ne veut pas dire libre ! L'Open source implique simplement que nous avons la possibilité d'avoir accès au code source. La notion de logiciel libre fait référence aux 4 libertés citées ci-dessus.

4.5 RAID

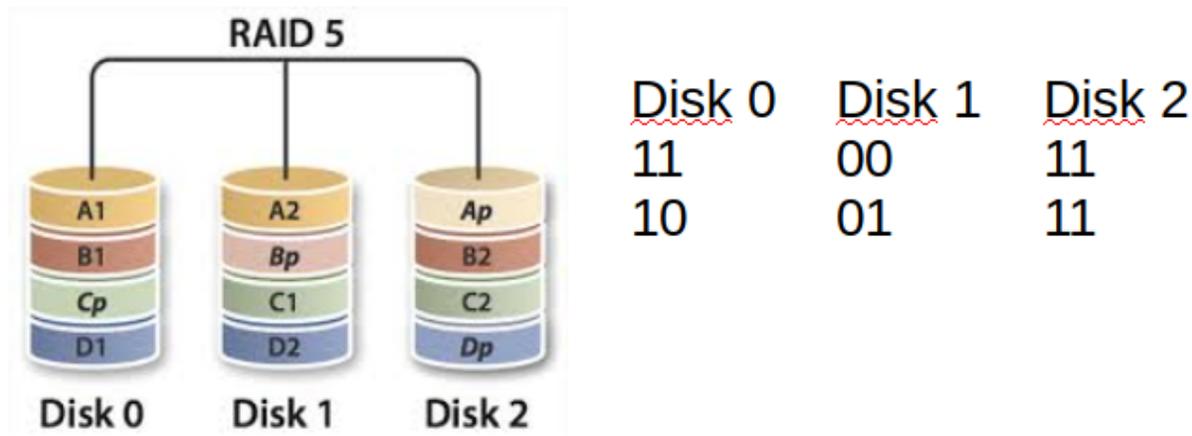
Il est important pour un serveur d'éviter toute panne de disque dur qui pourrait rendre le serveur indisponible.

Le RAID (Redundant Arrays of Inexpensive Disks) est un mécanisme de redondance de disques. Celui-ci a plusieurs objectifs combinables à savoir se prémunir contre la panne d'un disque et améliorer les performances en écriture sur les disques. Le RAID peut s'effectuer directement par le système d'exploitation, on parle alors de RAID logiciel. Le RAID peut également être effectué par un composant physique (contrôleur RAID) à part entière, on parle alors de RAID matériel. Le RAID logiciel a l'avantage d'être moins coûteux tandis que le RAID matériel présente l'avantage du remplacement de disque à chaud (c'est-à-dire sans devoir arrêter le serveur). Il existe différents niveaux de RAID. Voici les plus importants :

Niveau	Explication	Avantages	Inconvénients
RAID 0	Les données sont écrites en parallèle sur plusieurs disques	Gain en performance	Perte d'un disque == perte des données
RAID 1	Les données sont écrites sur des disques en miroir	Tolérance aux pannes disques	Coût -> disque en double
RAID 5	Les données sont écrites en parallèle sur au minimum 3 disques (2 disques de données + 1 disque de parité)	Tolérance aux pannes disques + performance	Minimum de 3 disques requis

Tableau à connaître

Le RAID 5 s'appuie sur l'opérateur logique XOR. La parité de chaque écriture est calculée via cet opérateur. En cas de perte d'un disque, l'information peut être reconstituée à partir des 2 autres disques toujours via cet opérateur XOR.



4.6 Installation Linux

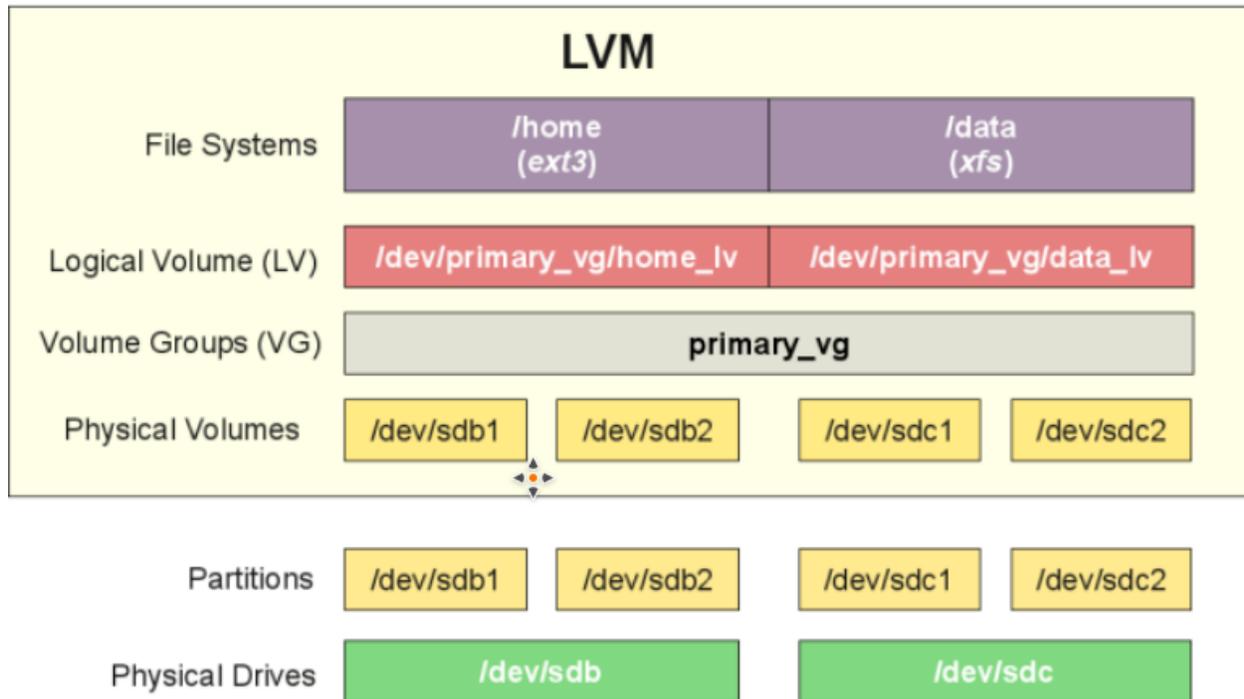
```
## [1] "Lien PDF pour INSTALLATION DEBIAN ci-dessous"
## [1] "http://ochoquet.be/syllabusAdminInfra/InstallDebian12VBOX_01.pdf"
```

4.6.1 Partitionnement

Le partitionnement désigne l'opération de diviser un disque en partitions. Un partitionnement bien réfléchi facilitera la maintenance des serveurs. Une partition système et une partition "données utilisateur" faciliteront par exemple la mise en place de sauvegardes.

4.6.1.1 LVM À l'installation, l'administrateur système devra réaliser un partitionnement. Cependant il parfois difficile d'imaginer comment vont évoluer les données (la taille et la quantité des données) au cours du temps. Afin de répondre à ce problème, les systèmes d'exploitation proposent une gestion dynamique des partitions. Ces partitions pourront grandir par l'ajout de nouveaux disques.

LVM(Logical Volume Manager) est une solution permettant une gestion dynamique du partitionnement disponible sous Linux. Les partitions peuvent ainsi s'étendre sur plusieurs disques. LVM demande de créer des groupes de volumes physiques (VG). On peut facilement ajouter de nouveaux disques à ces groupes ce qui peut faciliter l'agrandissement de partitions logiques (Logical Volume) . Voir ci-dessous.



4.6.1.2 Systèmes de fichiers Lors du choix du partitionnement, il sera nécessaire de préciser le système de fichiers de chaque partition. Voici les principaux :

1. **Ext2,Ext3,Ext4** : système de fichiers Linux incluant le concept de permissions (Ext4 est actuellement la version la plus utilisée sous Linux)
2. **NTFS** : système de fichiers Windows incluant le concept de permissions (NTFS est actuellement la version la plus utilisée sous Windows)
3. **FAT32** : ancien système de fichiers limité à des fichiers de maximum 4GB, pas de systèmes de permissions
4. **Swap** : partition d'échange (extension de la mémoire RAM sous Linux)
5. **Btrfs** : nouveau système de fichier permettant la prise d'instantanés (snapshot d'une partition) et le redimensionnement de partitions à chaud. Ce système utilise en interne les arbres B-tree.

4.6.1.3 Montage des partitions Sous Linux, le montage des partitions est documenté/configuré dans le fichier `/etc/fstab`. Voir ci-dessous.

# <file system>	<dir>	<type>	<options>	<dump>	<pass>
/dev/sda1	/	ext4	defaults	1	1
/dev/hdxx	/usr	ext4	defaults	1	1
/dev/sda5	swap	swap	defaults	0	0

Les disques sont nommés sous Linux suivant leur contrôleur physique qui les gère.

— s -> contrôleur SATA

- h -> contrôleur IDE (vieux disque, cdrom)

Ensuite vient un d pour disk suivi d'une lettre incrémentée automatiquement suivant le nombre de disques.

- a -> premier disque
- b -> second disque et ainsi de suite.

Il s'ensuit un chiffre également incrémenté qui représente le numéro de la partition.

Chaque ligne du fstab indique donc une partition, son point de montage, le système de fichiers utilisé, les options, si une sauvegarde doit être faite avec l'utilitaire dump (peu utilisé), l'ordre de vérification des disques lors d'une demande de vérification(fsck).

À noter qu'à la place d'une partition pour le premier point, on retrouve maintenant un UUID (un identifiant unique de partition) ou un label. L'avantage de ces 2 nouvelles méthodes est d'être moins sensible à un changement de disque. Un disque gardera toujours son label ou UUID et ceci peut donc être utilisé dans des scripts en toute sûreté.

4.6.1.4 Chiffrement des partitions Une partition non chiffrée peut être lue facilement sans autorisations particulières via un live-cd si on a un accès physique à une machine. Si on veut protéger ses données, il est donc nécessaire de crypter ses partitions pour éviter toute fuite de données. Ceci est particulièrement vrai pour les portables d'entreprises. Imaginez que le PDG d'une entreprise se fasse voler son portable dans le train, les voleurs pourront récupérer toutes les données du portable si les partitions ne sont pas cryptées.

Linux propose LUKS(Linux Unified Key Setup) qui permet un chiffrement des partitions (à l'installation ou plus tard). **Windows propose, quant à lui, BitLocker** pour crypter ses partitions.

4.6.2 Amorçage

Lors du démarrage d'une machine, un chargeur d'amorçage(bootloader) est lancé. Celui s'occupera de lancer le système d'exploitation ou de présenter les différents systèmes d'exploitation dans le cas d'un multi-boot. Windows propose winload comme chargeur d'amorçage tandis que Linux propose essentiellement GRUB (GRand Unified Bootloader).

4.7 Installation Windows (Windows Server 2016)

L'installation d'un serveur Windows est assez simple. C'est l'ajout de service, appelé rôle sous Windows, qui reste plus complexe. Les rôles permettent d'installer un Active Directory, un serveur DNS, DHCP,

Pour l'installation, Windows propose différents types d'installation :

- Standard : Support virtualisation limitée
- Datacenter : Support Cloud et virtualisation illimitée
- Essentials : Petite entreprise (25 utilisateurs)

Un aspect important sous Windows est la gestion des licences. Les licences serveur doivent être comptabilisées suivant le nombre de cœurs physiques du processeur. Il faut également comptabiliser les licences d'accès client (CAL).

Windows propose également différentes options pour l'installation d'un serveur.

- Avec expérience utilisateur : Environnement graphique / Recommandé
- Sans expérience utilisateur (core) / gestion du serveur par Sconfig ou Outils d'administration distants
- Nano server : Optimisé pour l'administration à distance de serveur cloud/datacenter

4.7.1 Configuration de base

Après l'installation d'un Windows, il est utile de changer son nom et d'ajouter la machine à un groupe de travail (workgroup) ou domaine. Les groupes de travail et les domaines permettent de regrouper un ensemble de machines dans le but de centraliser la sécurité, les accès. Les groupes de travail sont destinés à des petites structures tandis que les domaines visent des structures plus grandes. Les domaines seront gérés par un Active Directory (Voir Active Directory) .

5 Administration Linux (Debian)

5.1 Objectifs de ce chapitre

- Savoir administrer un serveur Linux (Debian)

5.2 APT

Toutes les distributions Linux disposent d'un système de gestion des packages permettant l'installation facile de logiciels et services. Ce système de gestion de packages résout en outre les problèmes de dépendances. Sous Debian ce programme est APT. Nous ne donnerons ici qu'un résumé du fonctionnement de l'outil APT. Différents dépôts contenant des paquets Debian (.deb) c'est-à-dire des logiciels prêts à être installés sont disponibles sur Internet. **L'outil APT dispose d'un fichier de configuration (/etc/apt/sources.list) permettant de renseigner les dépôts à utiliser.** Il suffit ensuite de mettre à jour depuis les dépôts (mise à jour du cache local) et de demander l'installation du logiciel à APT. L'outil installera automatiquement les dépendances nécessaires pour le logiciel demandé.

/etc/apt/sources.list

```
# deb cdrom:[Debian GNU/Linux 8.5.0 _Jessie_ - Official amd64 NETINST B
inary-1 20160604-15:31]/ jessie main

#deb cdrom:[Debian GNU/Linux 8.5.0 _Jessie_ - Official amd64 NETINST Bi
nary-1 20160604-15:31]/ jessie main

deb http://ftp.be.debian.org/debian/ jessie main
deb-src http://ftp.be.debian.org/debian/ jessie main

deb http://security.debian.org/ jessie/updates main
deb-src http://security.debian.org/ jessie/updates main

# jessie-updates, previously known as 'volatile'
deb http://ftp.be.debian.org/debian/ jessie-updates main
deb-src http://ftp.be.debian.org/debian/ jessie-updates main

deb http://ftp.debian.org/debian jessie-backports main
~
~
~
```

5.2.1 Commandes APT

Mise à jour du dépôt local :

```
apt-get update
```

Mise à jour des logiciels installés :

```
apt-get upgrade
```

Installer un logiciel :

```
apt-get install <paquet1> <paquet2> ...
```

Supprimer un logiciel :

```
apt-get remove <paquet1> <paquet2> ...
```

Rechercher un logiciel/paquet :

```
apt-cache search <word>
```

5.3 SSH

Les systèmes Linux actuels sont le plus souvent gérés en ligne de commande (pas d'interface graphique) et à distance. Pour ce faire, on utilisait telnet mais ce protocole a le gros inconvénient de ne rien crypter. Une simple écoute réseau permet alors de récupérer le mot de passe root. SSH est venu remplacer telnet.

5.3.1 Fonctionnement

Nous ne donnerons ici qu'un résumé du fonctionnement du protocole SSH. Le protocole SSH effectue un échange de clés de chiffrement avant d'utiliser ces dernières pour crypter toutes les communications entre le client et le serveur. Le port 22 est le port par défaut utilisé par SSH.

SSH est un service qui est initialisé/démarré par systemd.

5.3.2 Installation

```
apt-get install ssh
```

5.3.3 Configuration

Le fichier de configuration client est : `/etc/ssh/ssh_config` Le fichier de configuration serveur est : `/etc/ssh/sshd_config` Par défaut, SSH est installé pour permettre une authentification par login et mot de passe pour tous les utilisateurs présents sur le serveur (**hormis root**). **Après avoir effectué une modification dans un fichier de configuration, il faut redémarrer le service pour que les modifications soient prises en compte.**

```
systemctl restart ssh
```

5.3.4 Utilisation

Le client SSH a besoin des informations suivantes : un nom de machine ou une adresse IP, un login et un mot de passe. On peut remplacer l'authentification par login/mdp par une clé.

Comme client SSH, vous connaissez sans doute déjà le client SSH Windows par excellence : Putty.

Voici quelques astuces de configuration dans Putty pour + de confort et facilité :

- Keyboard -> CTRL-H
- Features -> disable keypad (avoir le pavé numérique)
- Connexion -> Data -> Autologin -> root/compte sudoer
- SSH -> Auth -> Clé privée

Mémoriser cette configuration dans une session PuTTY (Save). Vous pourrez ensuite directement vous connecter en cliquant sur cette session.

Exemple :

**1. key in the URL
cs.uco.edu**

2. select ssh

**3. give it a name
cs**

4. click on save

Sous Linux/MacOS :

```
ssh nomutilisateur@nommachineOUadresseIP
```

5.3.5 Sécurité

Par défaut, l'option « PermitRootLogin » dans `/etc/ssh/sshd_config` est positionnée à « without-password » ce qui indique que les connexions avec l'utilisateur root ne sont possibles qu'avec une clé. Il est également possible de restreindre l'utilisation que depuis certaines machines et qu'avec certains utilisateurs.

```
AllowUsers olivier@192.168.1.*
AllowUsers admin bob
```

Ici les utilisateurs admin et bob sont uniquement autorisés et l'utilisateur olivier depuis le sous-réseau 192.168.1.0/24

5.3.6 Copie de fichiers

Il est à noter que dès que vous avez un accès SSH, vous pouvez copier des fichiers entre votre machine hôte et invitée via SCP/SFTP. Ceci peut se faire en ligne de commande (Linux), avec le logiciel WinSCP (Windows) ou Cyberduck(Mac).

Exemple d'une copie de fichier depuis une machine physique vers une VM Azure en ligne de commande :

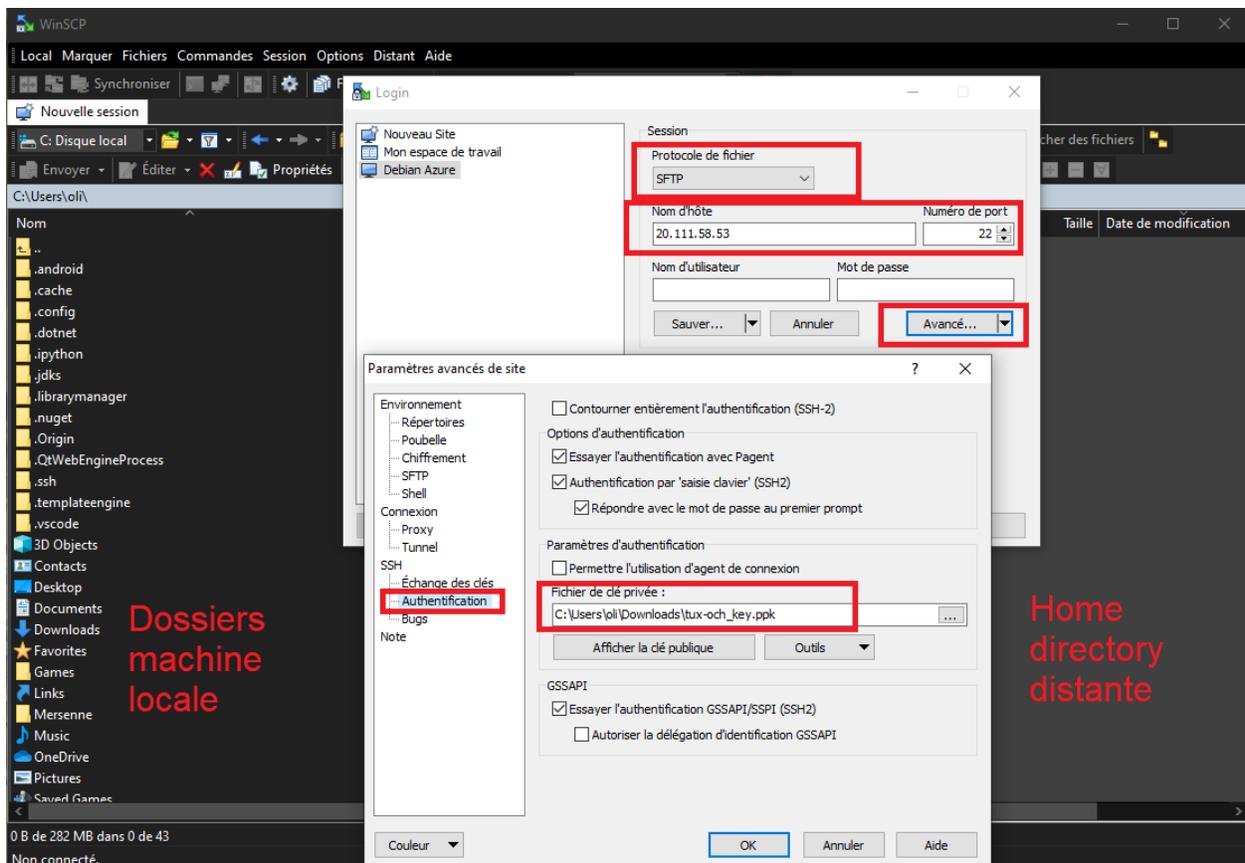
```
scp -i debian-och.pem monfichier.txt azureuser@20.15.56.87:/home/azureuser
```

Ceci copie le fichier monfichier.txt dans la home directory de l'utilisateur azureuser.

```
scp -i debian-och.pem -r mon dossier azureuser@20.15.56.87:/home/azureuser
```

Avec -r on peut copier récursivement tout un dossier.

Sous Windows, copie de fichiers avec le logiciel WinSCP (logiciel gratuit à télécharger) :



5.3.7 Authentification par clé sur un serveur Linux

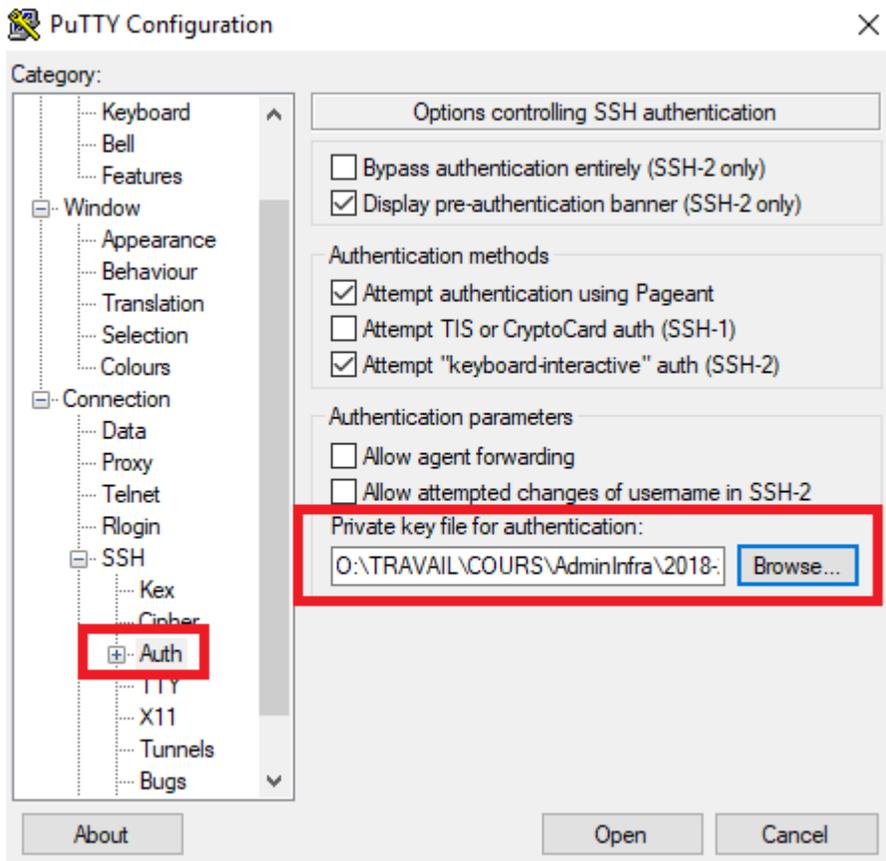
Pour réaliser une authentification SSH par clé, les étapes suivantes sont nécessaires :

1. Générer une paire de clés (privée/publique) sur le client via par exemple Puttygen (Windows) ou ssh-keygen (Linux)

Attention si vous utiliser Puttygen → copier le contenu de la clé publique généré dans un fichier texte. N'utiliser pas le bouton « Save public key » car il enregistre la clé sous un format non reconnu sous Linux.

2. Copier le contenu de la clé publique dans la home directory de l'utilisateur visé sur le serveur dans
~/.ssh/authorized_keys
3. Se connecter en précisant la clé privée

Exemple de connexion par clé via Putty :



5.3.8 Tunnel SSH

La création d'un tunnel SSH permet de connecter 2 machines en encapsulant le trafic de la première et en le redirigeant vers la seconde. Cette technique est souvent appelée le VPN du pauvre car elle permet notamment de donner accès à une machine du réseau local de l'entreprise à des ordinateurs distants et à moindres frais (de configuration).

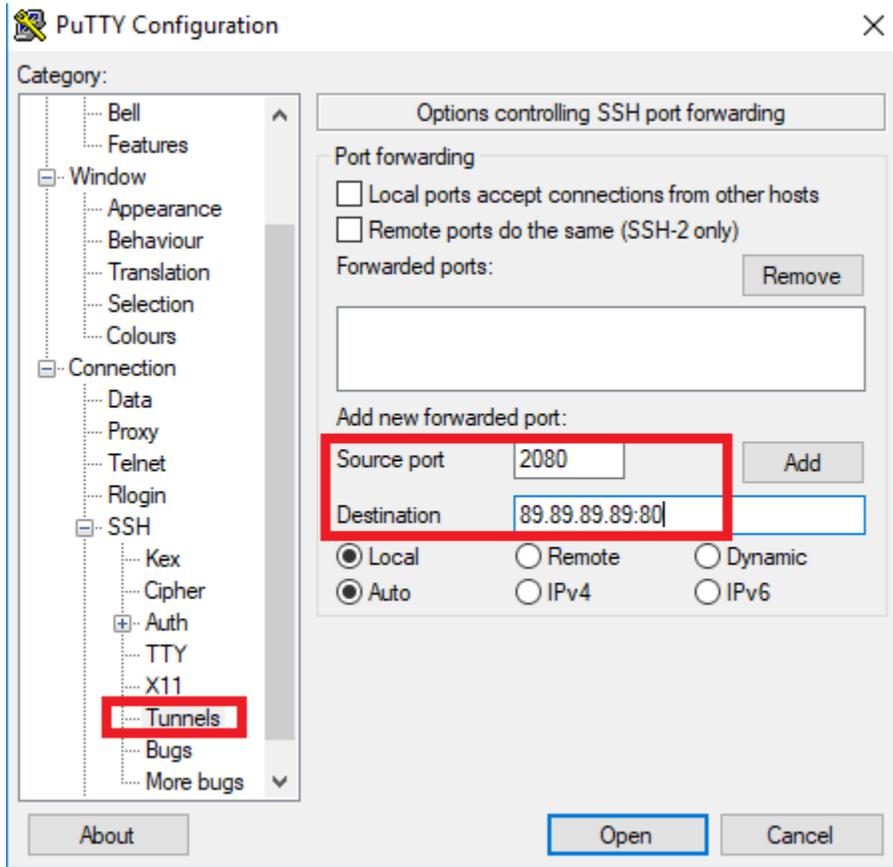
L' idée est de connecter via SSH un port local d'une machine cliente à un port distant d'un serveur. Cela fait drôlement penser à du port forwarding comme on peut le faire avec du NAT (Voir NAT) et c'est le cas.

Exemple Linux :

```
ssh -L 2080:localhost:80 olivier@89.89.89.89
```

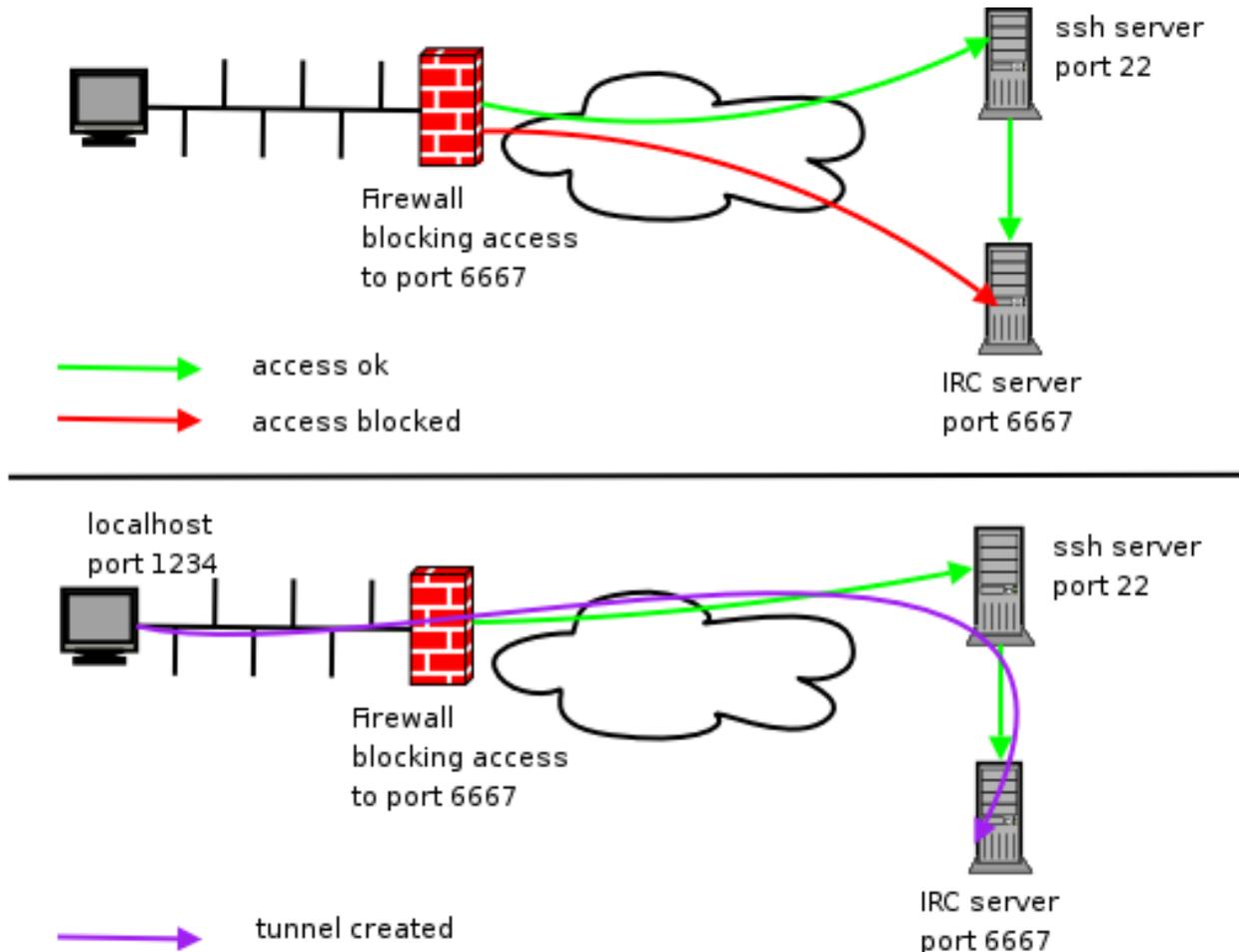
Dans cet exemple, on crée un tunnel sur le port 2080 de la machine locale et le port 80 du serveur 89.89.89.89. Le trafic de la machine locale sur le port 2080 sera donc envoyé vers le port 80 du serveur.

Exemple Windows (Putty) :



5.3.9 Concrètement cela sert à quoi ?

À accéder à un serveur de votre entreprise depuis la maison alors qu'il n'est pas directement accessible depuis Internet.



Un tunnel utilisant le SSH (OpenClassRoom) Attribution ©Wikipedia

5.4 Gestion des utilisateurs

5.4.1 adduser-deluser-addgroup-delgroup

Ces commandes sont suffisamment explicites. Consulter la documentation à ce sujet pour connaître les options intéressantes.

Il est à noter que Adduser crée un profil pour l'utilisateur basé sur un répertoire squelette situé dans `/etc/skel`. Tout fichier placé par l'administrateur dans ce répertoire squelette sera copié par défaut dans le répertoire de l'utilisateur lors de l'appel à adduser.

Par défaut, la home directory créée par adduser est accessible en lecture à tout le monde (voir `/etc/adduser.conf`). Attention, ceci peut ne pas correspondre à votre politique de confidentialité. Ceci peut être changé dans `/etc/adduser.conf`.

5.4.2 SU

Cette commande permet de changer d'utilisateur. Sans argument, cela permet de devenir root.

```
su admin
```

5.4.3 SUDO

La commande sudo a pour objectif de permettre à des utilisateurs d'exécuter des commandes en tant que superutilisateur.

5.4.3.1 Fonctionnement Pour qu'un utilisateur puisse exécuter une commande avec « sudo », il doit faire partie du groupe sudo.

```
apt-get install sudo
```

5.4.3.2 Installation

5.4.3.3 Utilisation Pour ajouter un utilisateur au groupe sudo :

```
adduser toto sudo
```

5.4.3.4 Avantages de SUDO Les avantages du SUDO sont les suivants :

1. Permettre à des utilisateurs d'exécuter une commande en tant que superutilisateur sans devoir le mot de passe de root.
2. Travailler en mode non privilégié et n'utiliser le mode privilégié que quand cela est nécessaire. Ceci réduit le risque de commettre des dommages pour le système.
3. Contrôler et enregistrer qui fait quoi (SUDO enregistre toutes les commandes sudo effectuées dans /var/log/auth.log).
4. Renforcer la sécurité. En désactivant le compte root et en le remplaçant par un compte « sudo », un attaquant ne connaîtra pas le mot de passe, ni le nom du compte !

5.5 Passwd

Passwd permet de changer le mot de passe de son compte et de tous les comptes (pour le root). Il est également possible de verrouiller ou de désactiver le compte root. Verrouiller le compte root empêche simplement de pouvoir se connecter directement avec le compte root tandis que la désactivation le rend totalement inutilisable.

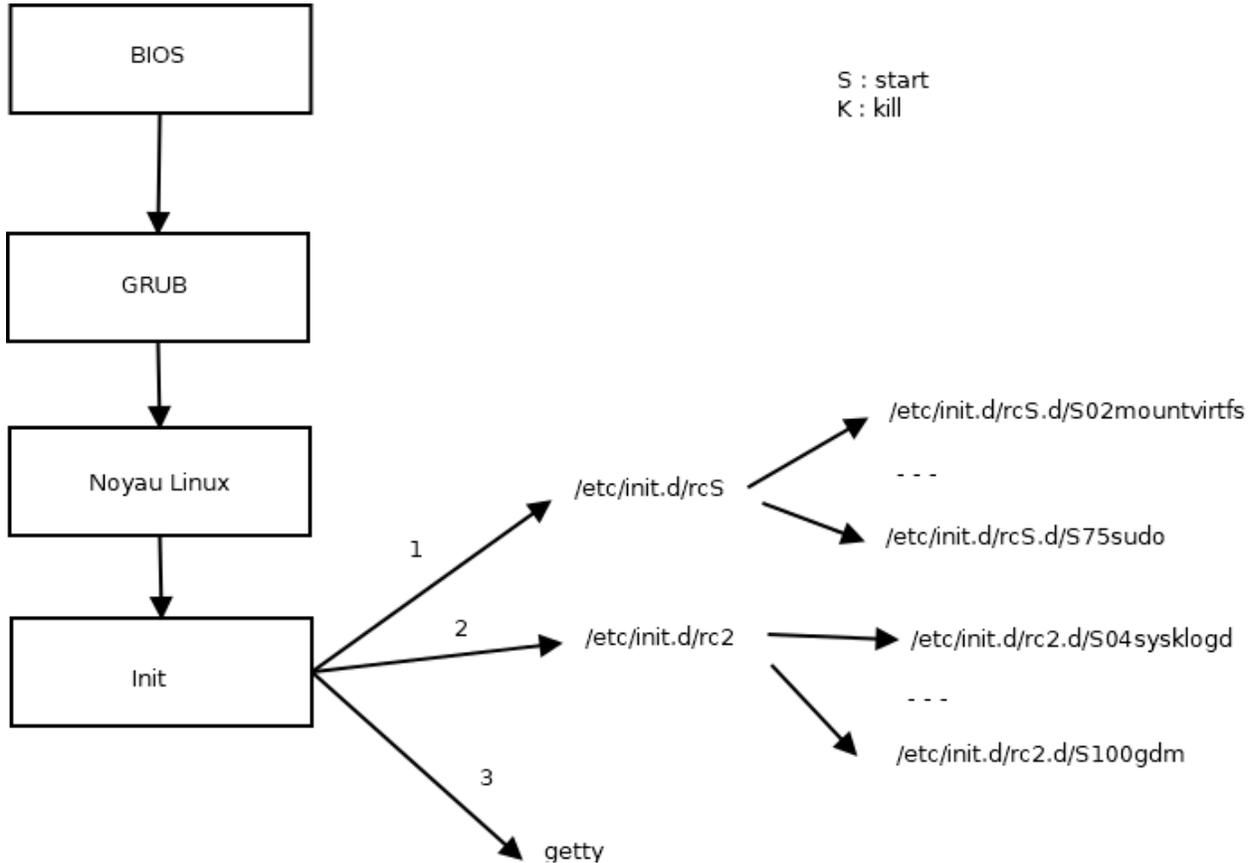
5.6 SystemD

Le système d'exploitation Linux est né sur base du système d'exploitation Unix. Il a donc récupéré énormément de caractéristiques de ce système notamment son système d'initialisation. Les systèmes Unix utilisent une architecture System V. Cette architecture possédait à l'origine de nombreux avantages tel que la mémoire partagée, les sémaphores (cfr. Cours Pgm Sys) qui sont d'ailleurs toujours utilisé aujourd'hui. Ce système d'initialisation était donc le premier processus (init) que lançaient les systèmes Unix et Linux. L'architecture System V divisait l'environnement d'exécution en une série de runlevel et disposait d'un fichier inittab qui précisait quelles applications étaient lancées suivant le **runlevel**.

Les niveaux d'exécutions :

- **0** : arrêt (la commande `init 0` arrête le système)
- **1** : mono-utilisateur (utiliser par exemple pour la maintenance)
- **3** : multi-utilisateur sans environnement graphique
- **2-4** : idem que 3, mais peut être défini par l'utilisateur (peu utilisé en pratique)
- **5** : multi-utilisateur avec environnement graphique
- **6** : redémarrage (la commande `init 6` redémarre le système)

Le système d'initialisation en System V :



Encore actuellement, vous verrez des commandes de démarrage de service tel que `/etc/init.d/ssh restart` qui constitue une trace de cette architecture. Cependant cette architecture date de 1983 et certains manquements ont commencé à apparaître par rapport à de nouveaux besoins. C'est ainsi que SystemD est apparu. SystemD reprend l'architecture SystemV en y ajoutant ceci :

- Fournit des capacités poussées de parallélisation
- Utilise les sockets et l'activation D-Bus pour démarrer les services
- Offre un démarrage à la demande des daemons
- Implémente une logique de contrôle transactionnelle des dépendances entre services
- Piste les processus en utilisant les cgroups Linux
- Supporte les snapshots et leurs restaurations
- Maintiens les points de montage (montage/démontage)

L'objectif principal de SystemD (tout comme SystemV) est de démarrer des services, appelé daemons dans le monde Linux. Il est donc normal que celui-ci propose différentes manières d'implémenter son service.

Voici les types de service :

1. Simple : démarre le service immédiatement comme processus principal
2. Forking : démarre le service dans un processus fils
3. OneShot : idem que simple, mais attend la fin du processus
4. Dbus : idem que simple, mais attend d'un nom Dbus
5. Notify : idem que simple, mais le processus avertira SystemD quand il peut traiter les unités suivantes

Exemple de définition d'un service :

```
[Unit]
Description=add-client-identifrier
Before=network-pre.target
Wants=network-pre.target
[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/home/ipl/add-client-identifrier.pl
ExecStop=
[Install]
WantedBy=multi-user.target
```

Les niveaux d'exécution en SystemD (assez similaire à System V) :

- 0 : poweroff.target
- 1 : rescue.target
- 3 : multi-user.target
- 2,4 : multi-user.target
- 5 : graphical.target
- 6 : reboot.target
- emergency.target

6 Serveurs Web (Apache)

6.1 Objectifs du chapitre

— Déployer un site Web via Apache

Ce chapitre a surtout pour but de vous aider pour les exercices.

6.2 Introduction

Il est évident qu'un administrateur système devra installer tôt ou tard un serveur web dans son entreprise vu le nombre croissant d'applications Web. Différents Serveurs Web existent. Voici un tableau des principaux avec leurs caractéristiques.

Serveur Web	Caractéristiques
Apache	Leader historique, hautement configurable, modules chargés dynamiquement, présent sous Windows et Linux
Nginx	Challenger, optimisé pour la performance (résolution problème CK10 - servir 1000 clients simultanément), modules compilés, présent sous Linux
IIS	Serveur Web Windows, support ASP.NET

Après ce bref tour d'horizon des serveurs Web, nous allons nous intéresser plus particulièrement à Apache.

6.3 Etapes de déploiement d'un site Web

1. Installer le paquet du serveur Web (Apache)
2. Transférer/Installer le code du site Web sur le serveur (/var/www)
3. Créer un VirtualHost
4. Activer le site
5. Faire correspondre la Directive ServerName et /etc/hosts
6. Tester le site Web en local

6.4 Installer Apache

```
apt-get install apache2
```

L'installation crée un compte et un groupe `www-data`. Apache 2 fonctionne par défaut sur ce compte et groupe pour des raisons de sécurité et tourne sur le port 80. Un site de base (page HTML) est placé dans `/var/www` ce qui permet de tester directement Apache2 après son installation : <http://adresseip>. Il est à noter que si vous voulez tester apache sur un serveur ne disposant pas d'interface graphique (et donc pas de navigateur classique), vous pouvez installer **lynx** qui est un navigateur en mode texte (c'est moche, mais cela permet de tester!).

6.5 Caractéristiques d'Apache

Apache étant hautement configurable, il se caractérise par une configuration morcelée. Le fichier de configuration de base est `/etc/apache2/apache2.conf`. Ce fichier inclut tout simplement d'autres fichiers et répertoires à savoir :

- `/etc/apache2/sites-available` : définitions de site Web (VirtualHost)
- `/etc/apache2/sites-enabled` : définitions de site Web (VirtualHost) activés
- `/etc/apache2/mods-available` : liste des modules (SSL, proxy, ..) installés
- `/etc/apache2/mods-enabled` : liste des modules (SSL, proxy, ..) activés
- `/etc/apache2/conf-available` : liste des configurations (charset, ..) disponibles
- `/etc/apache2/conf-enabled` : liste des configurations (charset, ...) activés
- `/etc/init.d/apache2` : un service qui sera démarré/arrêté par SystemD
- `/etc/apache2/ports.conf` : la configuration des ports pour apache (80 et 443 par défaut)

Un administrateur système écrira/préparera ces configurations dans les répertoires «available» et il activera ensuite ces configurations via les commandes suivantes :

```
#activer/ désactiver un site
a2ensite/a2dissite <filenamesite>
#activer/ désactiver une configuration
a2enconf/a2disconf <filenameconf>
#activer/ désactiver un module
a2enmod/a2dismod <filenamemod>
```

6.6 VirtualHost

Les virtualhosts permettent de déployer plusieurs sites Web sur un même serveur (même adresse IP). La distinction se fait en général sur le nom du site, apache doit en effet savoir suivant l'URL quel site il doit présenter.

L'ajouter d'un vhost se fait en créant un fichier dans `/etc/apache2/sites-available/«monsite.conf»`.

Exemple Vhost :

```
<VirtualHost *:80>
    ServerName monsite

    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/htdocs/monsite
    ErrorLog ${APACHE_LOG_DIR}/monsite_error.log
    CustomLog ${APACHE_LOG_DIR}/monsite_access.log combined

    <Directory /var/www/htdocs/monsite>
        Require all granted
```

```
    AllowOverride All
  </Directory>
</VirtualHost>
```

Il est possible de déboguer et de vérifier la syntaxe du vhost avec la commande suivante :

```
apache2ctl configtest
```

Ensuite, il faut activer le site comme suit :

```
#activer un site
#commande à entrer dans /etc/apache2/sites-available
a2ensite monsite.conf
```

6.6.1 Directives

La directive «ServerName» est nécessaire pour qu'Apache fasse une distinction sur le nom du site. Toute URL comportant «monsite.be» utilisera ce vhost.

La directive «ServerAdmin» permet de préciser le responsable du site.

La directive «DocumentRoot» permet de préciser l'endroit où se trouve l'arborescence du site.

La directive «ErrorLog» et «CustomLog» permettent de préciser où les logs seront stockés.

On peut ensuite appliquer des règles/restrictions sur le site via la directive «Directory».

Ici on autorise tout le monde à voir le site «Require all granted» et on autorise les utilisateurs à redéfinir ces règles «AllowOverride All». Ceci permet par exemple de définir des .htaccess.

La directive «Require» peut autoriser ou interdire des adresses IPs, des utilisateurs...

Exemples :

```
# n'autoriser l'accès au site que depuis localhost
require ip localhost
# accès uniquement au site pour l'utilisateur admin
require user admin
# pas de redéfinition possible (pas de .htaccess)
AllowOverride None
```

6.7 Reverse proxy

Un proxy inverse est un serveur frontal c'est-à-dire un serveur exposé sur Internet et par lequel toutes les requêtes passeront. Ce serveur ne traitera pas les requêtes, mais se contentera de les rediriger vers d'autres serveurs internes à l'entreprise.

Les intérêts de ce mécanisme sont multiples. Vu qu'il n'y a qu'un seul point d'accès, la sécurité est plus facile à gérer. Cela permet également de mettre en œuvre du «load balancing» entre des serveurs internes. C'est également un moyen simple de rendre disponible un serveur interne sur le Web (pas besoin de configuration réseau).

Pour mettre en place un reverse proxy, il faut activer le module apache « proxy_http » et « proxy ».

```
a2enmod proxy proxy_http && systemctl restart apache2
```

Ensuite dans le fichier VirtualHost :

```
<VirtualHost *:80>
    ServerName siteReverseProxy
    ServerAdmin webmaster@localhost

    # attention / final !!!
    ProxyPass / http://www.example.com/
    ProxyPassReverse / http://www.example.com/

    ErrorLog ${APACHE_LOG_DIR}/siteReverse_error.log
    CustomLog ${APACHE_LOG_DIR}/siteReverse_access.log combined
</VirtualHost>
```

6.8 Site PHP

Apache peut être configuré pour servir des pages PHP. Il suffit d'installer PHP ainsi que le module PHP pour apache et de redémarrer le service apache2.

```
apt-get install php php-mysql libapache2-mod-php
```

Un site PHP possède généralement une base de données MySQL ou MariaDB. Voici comment installer MySQL et injecter un fichier SQL permettant la création de la base de données.

Installer MySQL :

```
apt-get install default-mysql-server
```

Injecter un fichier de création DB :

```
mysql -u username -p < file.sql
```

6.9 Apache et HTTPS

Un serveur Web doit être sécurisé en particulier les échanges entre le client et le serveur doivent être cryptés. Ceci se fait aisément grâce au paquet Openssl. **Le port par défaut pour les communications https est le 443.**

6.9.1 Installation

```
apt-get install openssl && a2enmod ssl && systemctl restart apache2
```

6.9.2 Création d'un certificat autosigné

La commande openssl permet de créer un certificat ainsi qu'une clé associée à ce certificat.

```
mkdir /etc/apache2/ssl &&
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/apache2/ssl/apache.key
↪ -out /etc/apache2/ssl/apache.crt
```

Ici la clé et le certificat seront déposés dans le répertoire /etc/apache2/ssl créé au préalable.

Le VirtualHost sera modifié de la sorte :

```
<VirtualHost *:443>
    ServerName monsite.be

    ServerAdmin webmaster@localhost
```

```
DocumentRoot /var/www/htdocs/monsite
ErrorLog ${APACHE_LOG_DIR}/monsite_error.log
CustomLog ${APACHE_LOG_DIR}/monsite_access.log combined

SSLEngine on
SSLCertificateFile /etc/apache2/ssl/apache.crt
SSLCertificateKeyFile /etc/apache2/ssl/apache.key

<Directory /var/www/htdocs/monsite>
Require all granted
AllowOverride All
</Directory>
</VirtualHost>
```

6.9.3 Let's Encrypt

Let's encrypt est une autorité de certification libre, gratuite et automatisée. Ceci permet d'obtenir un certificat valide pour son site Web sans trop d'effort. Cependant, la machine servant le site Web doit être «publiquement» accessible ainsi que le nom du domaine. Cela veut dire qu'en test ce procédé n'est pas applicable.

[Let's Encrypt](#)

6.10 Load balancing & Apache

En infrastructure, il est courant de répartir la charge sur plusieurs serveurs. Cela permet d'accélérer le temps de réponse aux clients. L'ensemble des requêtes sont alors balancées entre plusieurs serveurs Web.

6.10.1 Installation

```
# installation du module proxy si cela n'a pas encore été fait
a2enmod proxy
# installation du module proxy_http si cela n'a pas encore été fait
a2enmod proxy_http
# installation du module proxy_balancer
a2enmod proxy_balancer
# installation du module contenant la méthode de répartition de la charge
a2enmod lbmethod_byrequests
# redémarrer apache pour prendre en compte les modules
systemctl restart apache2
```

Ensuite dans le fichier VirtualHost :

```
<VirtualHost *:80>
    ServerName monsite.be

    <Proxy balancer://mycluster>
        BalancerMember http://127.0.0.1:8080
        BalancerMember http://127.0.0.1:8081
    </Proxy>
    ProxyPreserveHost On
    ProxyPass / balancer://mycluster/
    ProxyPassReverse / balancer://mycluster/
</VirtualHost>
```

7 Partage et accès réseau

7.1 Objectifs du chapitre

- Connaître les différents moyens de partage et d'accès à un serveur et leur rôle

7.2 NFS

NFS est un protocole réseau couramment employé pour partager des fichiers sur un réseau. NFS fonctionne en mode client-serveur. Les versions les plus utilisées sont la 2 et 3. Nous ne parlerons pas ici de la version 4. **NFS est un protocole performant, mais sans sécurité accrue. Ce protocole est le protocole de partage réseau par excellence sous Linux/MacOS.**

7.2.1 Installation du serveur NFS

```
apt-get install nfs-kernel-server nfs-common
```

7.2.2 Configuration des partages NFS

Les partages sont à définir dans `/etc/exports` et ont la forme suivante :

```
<share> <host1>(<options>) <hostN>(<options>)
```

Les hôtes sont des adresses IP ou des noms DNS de machines qui pourront accéder au partage. On peut utiliser le joker (*) pour spécifier plus facilement plusieurs machines ou un sous-réseau. Les options les plus courantes sont :

1. ro : read only
2. rw : read write
3. root_squash : réduction des droits du root distant à un simple utilisateur
4. all_squash : réduction des droits de tout utilisateur distant à un simple utilisateur
5. sync : les opérations sont synchrones

7.2.3 Fonctionnement NFS

NFS vérifie l'identité des utilisateurs via les UID, GID. Il faut donc que les UID, GID de la machine distante et locale corresponde. L'UID de root est toujours le même (0).

Il est donc important de comprendre que NFS (v2,V3) ne demande pas aux utilisateurs de s'authentifier. En effet, on autorise via le fichier « exports » une série d'hôtes à se connecter et ensuite c'est via les permissions classiques UNIX/LINUX que les droits sont positionnés sur le répertoire du serveur NFS. NFS va ensuite simplement vérifier les UID, GID locaux(serveur NFS) et distants (client NFS).

Pour ces raisons, il est alors très facile de casser la sécurité NFS. C'est pourquoi il est presque exclusivement utilisé pour des partages en lecture seule généraliste ou pour faire des backups à distance.

7.2.4 Exemples exports NFS

```
/home/olivier 192.168.1.1(ro)
```

→ la machine ayant l'IP 192.168.1.1 peut monter le répertoire `/home/olivier`. Ce répertoire sera accessible en lecture seule.

```
/home/public *.ipl.be(rw, sync)
```

→ la machine ayant pour nom `*.ipl.be` peuvent monter le répertoire `/home/bob`. Ce répertoire sera accessible en lecture/écriture et toute opération d'écriture sera effectuée directement(sync).

```
/home/net 192.168.1.0/24 (rw,async,no_root_squash)
```

→ les machines du réseau 192.168.1.0/24 peuvent monter le répertoire /home/net en lecture/écriture.

7.2.5 Exemples client NFS

```
mount -t nfs 192.168.1.1:/home/public /datas/public
# -o → options
mount -t nfs -o soft,intr,rsize=8192,wsiz=8192 192.168.1.65:/volume1/BACKUP /home/nfs
```

7.2.6 Vérification

```
showmount -e
```

7.3 SAMBA

SMB/CIFS est un protocole propriétaire utilisé sous Windows pour les partages réseau.

Vous connaissez certainement ces partages réseaux qui ont des paths UNC de la forme : \\machineserveur\partage.

Samba est un logiciel né sous Linux qui implémente ce protocole propriétaire SMB/CIFS. Il permet donc une interopérabilité entre les 2 mondes(Linux/Windows) notamment :

- Créer des partages sous Linux accessible sous Windows
- Transformer un serveur Linux en Domain Controller
- Authentifier des clients Linux sur un Active Directory

Samba fonctionne tout comme NFS en mode client-serveur. Samba possède une gestion plus élaborée au niveau de l'authentification et des droits des utilisateurs par rapport à NFS.

7.3.1 Installation du serveur SAMBA

```
apt-get install samba samba-common-bin
```

7.3.2 Configuration du serveur SAMBA

La configuration s'effectue dans /etc/samba/smb.conf. Nous n'allons pas ici décrire tous les paramètres, mais voici les principaux. Les partages se définissent entre [...] (section). Certaines sections sont prédéfinies comme :

- global : paramètres globaux de serveur SAMBA
- homes : partager automatiquement les homedirs des utilisateurs du serveur
- printers : partager les imprimantes du serveur.

À l'intérieur de ces sections, les paramètres se définissent de la sorte : paramètre = valeur Voici quelques exemples de configuration classique ci-dessous.

7.3.3 Partage public

Le but est ici de donner accès à tout le monde en lecture seule à un répertoire situé sur le serveur.

```
[global]
server string = monserveurLinux
security = user
```

```
# les utilisateurs qui se connecte au partage seront mappés sur le compte anonyme
map to guest = Bad user
# compte anonyme pour les « Bad user »
guest account = nobody
# compatibilité protocole SMB et Windows 10
min protocol = LANMAN2
max protocol = NT1
[public]
# chemin local du partage
path = /home/samba/allusers

# accès possible en anonyme à ce partage
public = yes
readonly = yes
# répertoire visible dans l'explorateur de fichiers
browseable = yes
```

7.3.4 Partage en écriture

```
[partageEcriture]
# chemin local du partage
path = /home/samba/partageEcriture
browseable = yes
write list = user1, user2
```

Ici plus question d'accès anonyme, il est nécessaire d'ajouter un compte SAMBA avec mot de passe.

```
smbpasswd -a user1
```

7.3.5 Partage des “homedirs”

Le but ici est permettre l'accès en écriture à leur home directory (depuis Windows par ex.) aux utilisateurs enregistrés sur le serveur Linux.

```
[global]
server string = monserveurLinux
security = user
[homes]
# inutile de rendre visibles à tout le monde ces partages
browseable = no
writable = yes
# les utilisateurs pouvant accéder à ce partage sont ceux
# correspondant à %S c'est-à-dire au nom du partage ...
# c'est-à-dire le nom de l'utilisateur ici
valid users = %S
```

7.3.6 Lancer/redémarrer SAMBA

```
systemctl restart samba
```

7.3.7 Outils / Infos utiles

La commande «testparm» permet de vérifier la configuration de votre smb.conf.

Sous Windows, la commande «NET USE» (à taper dans une console DOS) permet de voir les connexions SMB et de les supprimer. NET USE / DELETE.

Voici quelques paramètres intéressants :

- write list = user1 user2 ..
- create mask / directory mask
- comment (une description du partage)
- log file
- host allow / host deny

On peut donner un groupe via @ : valid users = @admins

7.4 VPN

Un VPN (Virtual Private Network) est un système permettant de relier 2 réseaux via un réseau non sûr tout en garantissant un trafic sécurisé (crypté) et de manière transparente. On parle de tunnel.

Les VPN sont de plus en plus utilisés, car ils permettent notamment de se protéger des écoutes sur un réseau public (NSA, ...) et d'anonymiser sa connexion.

7.4.1 Classification VPN

Il existe différents types de VPN à savoir :

1. LAN to LAN / Site to Site permettant de relier 2 réseaux (Ex : relier des succursales d'une entreprise éparpillées dans le monde)
2. RoadWarrior permettant à un PC externe de se connecter à l'entreprise (Nomadisme des employés)

Il est important de comprendre que le VPN crée un réseau et le configure pour que le réseau distant (LAN to LAN) ou PC distant (Road Warrior) soit considéré comme s'il était dans l'entreprise. L'utilisateur pourra donc utiliser tout ce qui accessible dans le LAN de l'entreprise (imprimantes ...).

7.4.2 VPN en images

Road Warrior

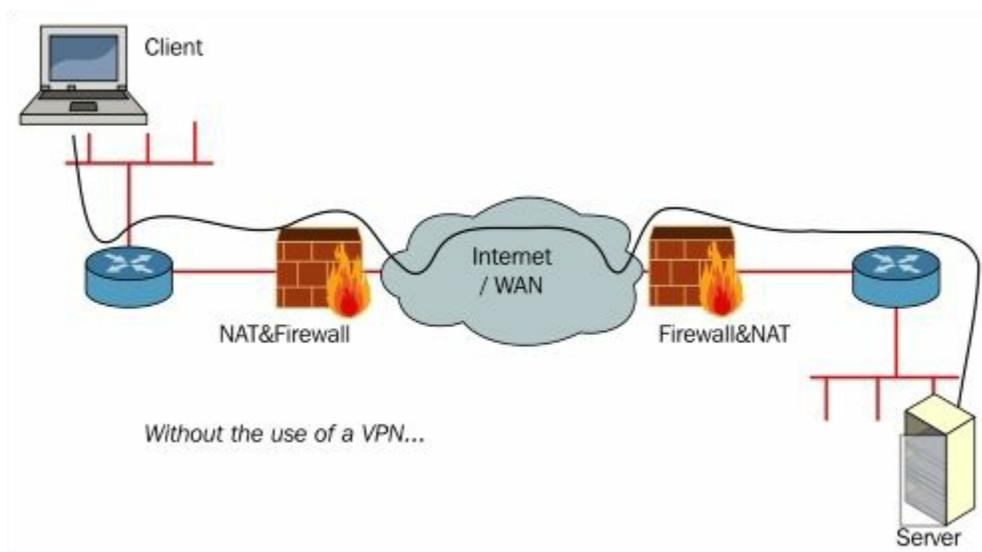


Image issue de : https://www.junmajinlong.com/files/Mastering_OpenVPN.pdf

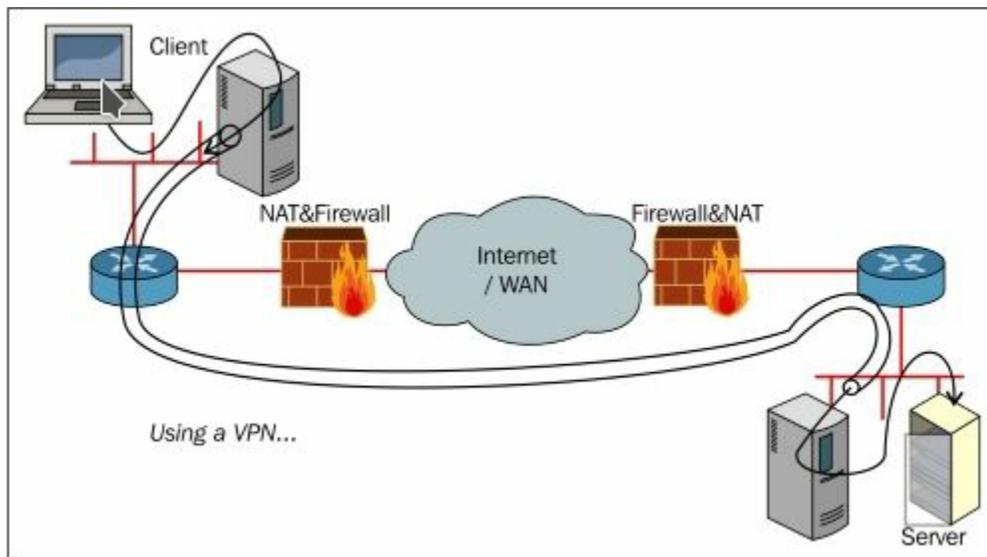


Image issue de : https://www.junmajinlong.com/files/Mastering_OpenVPN.pdf

LAN-to-LAN / site-to-site

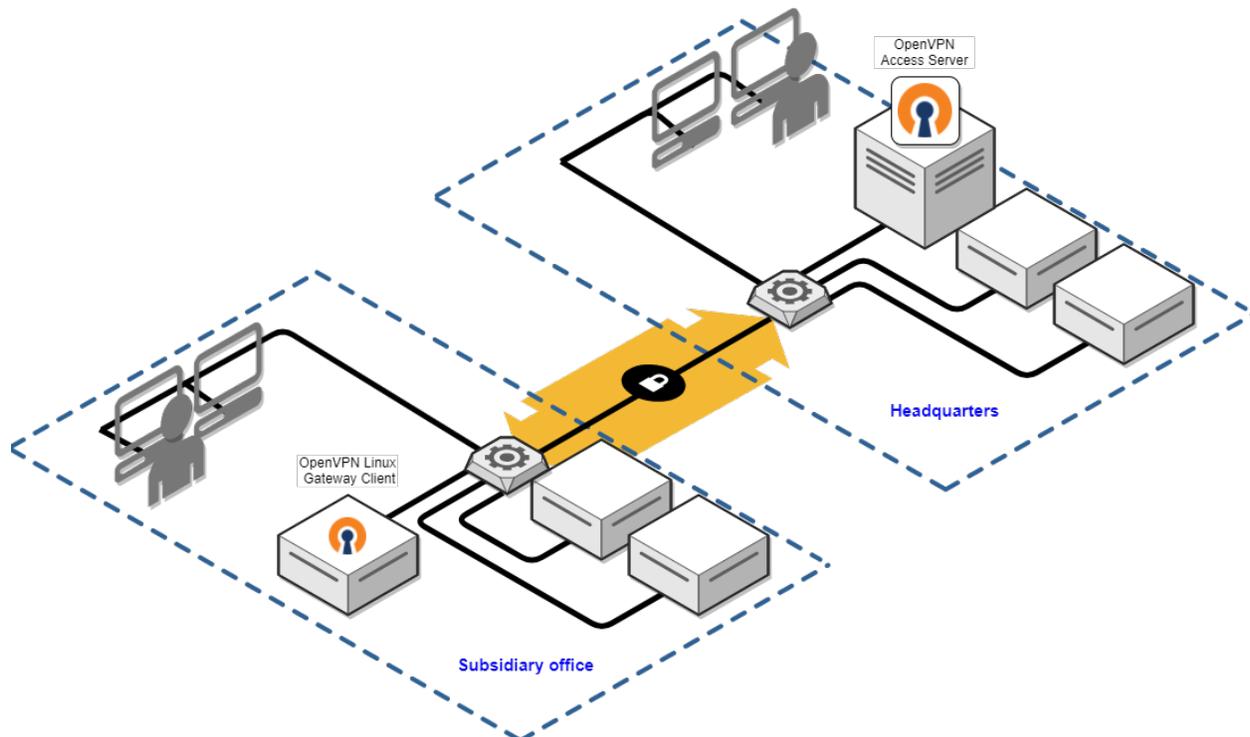


Image issue de : <https://openvpn.net/vpn-server-resources/site-to-site-routing-explained-in-detail/>

7.4.3 Protocoles VPN

Protocole	Couche Réseau	Remarques
PPTP (Point to Point Tunneling Protocol)	Couche Liaison des données	Road Warrior

Protocole	Couche Réseau	Remarques
L2TP (Layer 2 Tunneling Protocol)	Couche Liaison des données	Road Warrior / Remplaçant PPTP
IPSec	Couche Réseau	Site To Site / intégré à IPV6 / se configure essentiellement sur les routeurs-firewalls des entreprises
OpenVPN	Couche Application	OpenVPN est un logiciel créant un VPN en se basant sur SSL/TLS – RoadWarrior ou Site to Site

7.4.4 Exemple OpenVPN

```
# Serveur VPN (server.conf)
mode server
proto udp
dev tun
topology subnet
# clé et certificat SSL
ca keys/ca.crt
cert keys/cert.crt
dh keys/dh2048.pem
# réseau créé
server 10.50.0.0 255.255.255.0
keepalive 10 120
# compression des échanges
comp-lzo
```

```
#client VPN (client.conf)
client
proto udp
dev tun
remote 89.89.89.89 1194
nobind
ca /etc/keys/ca.crt
cert /etc/keys/roadwarrior.crt
key /etc/keyroadwarrior.key
comp-lzo
```

7.5 FTP

Le FTP (File Transfer Protocol) est un protocole réseau standard. On le retrouve donc très facilement sous n'importe quel environnement (Windows/Linux). De nombreux clients (graphiques ou non) existent. On l'utilise encore fréquemment pour transférer du code source sur un serveur hébergé.

Comme son nom l'indique, ce protocole est fait pour transférer des fichiers entre ordinateurs. C'est le protocole le plus efficace pour cette tâche. IL est dès lors très utilisé DNS les tâches de backups.

Le protocole FTP utilise un canal pour le transfert des données et un autre pour le contrôle. C'est pourquoi il utilise par défaut 2 ports (20 → données, 21 → contrôle). Le canal de contrôle permet d'envoyer les commandes FTP (put, get, open, close, ls, ...).

Il peut être sécurisé via SSL/TLS (FTPS) ou par SSH (SFTP). Regardez dans WinSCP, vous verrez que vous transférez par défaut vos fichiers par SFTP !

7.5.1 Modes

Le protocole FTP peut s'utiliser en mode actif ou passif.

Dans le mode actif, le client peut choisir son port de connexion pour la réception des données. Le serveur initialisera une connexion de son port 20 vers le port choisi par le client. Le client doit donc accepter les connexions entrantes sur le port choisi. Ceci pose souvent problème car les clients se trouvent généralement dans un LAN qui effectue du NAT vers l'extérieur. C'est pourquoi le mode actif est le moins utilisé.

Dans le mode passif, le serveur impose le port de connexion pour le transfert des données. Le port choisi par le serveur est envoyé au client qui initialise alors la connexion.

7.6 Terminal Server

L'administration d'un serveur Linux se fait assez facilement via SSH. Le pendant dans le monde Windows est le Terminal Server aussi appelé Bureau à distance. Le Terminal Server repose sur le protocole RDP (Remote Desktop Protocol) développé par Microsoft.

Terminal Server permet non seulement la prise à distance d'un serveur avec son interface graphique, mais aussi de monter des lecteurs locaux sur le serveur distant.

7.7 Autres solutions d'accès réseau

SCP, Teamviewer, VNC, Nomachine, Citrix,

8 Annares et Authentification

8.1 Objectifs du chapitre

— Connaître les différents composants d'un Active Directory et leurs rôles

Dans un réseau d'entreprise, il devient rapidement nécessaire de centraliser les authentifications afin de faciliter la maintenance de la politique des accès, des droits, des stratégies de sécurité.

Le protocole LDAP (Lightweight Directory Access Protocol) a été défini pour permettre d'interroger et de modifier un annuaire. Il est depuis devenu une référence et un standard pour l'authentification.

LDAP est en fait devenu bien plus qu'un protocole, c'est une norme qui définit :

1. un protocole : comment sont échangées les données
2. un modèle de nommage : comment sont nommées les entrées dans l'annuaire
3. un modèle fonctionnel : quelles sont les méthodes pour accéder aux données
4. un modèle d'information : nature et description des données
5. un modèle de sécurité : description de la sécurité des données (quel chiffrement ...)
6. Réplication : comment répliquer des données entre serveurs LDAP pour se prémunir des pannes ? Ce point n'est pas encore standardisé.

Différentes implémentations de la norme LDAP existent. OpenLDAP est la solution reconnue du monde libre (paquet slapd dans Debian). La solution Active Directory de Microsoft est sans doute la plus connue et répandue. Nous en parlerons plus longuement (Voir Active Directory).

8.2 Modèle de nommage

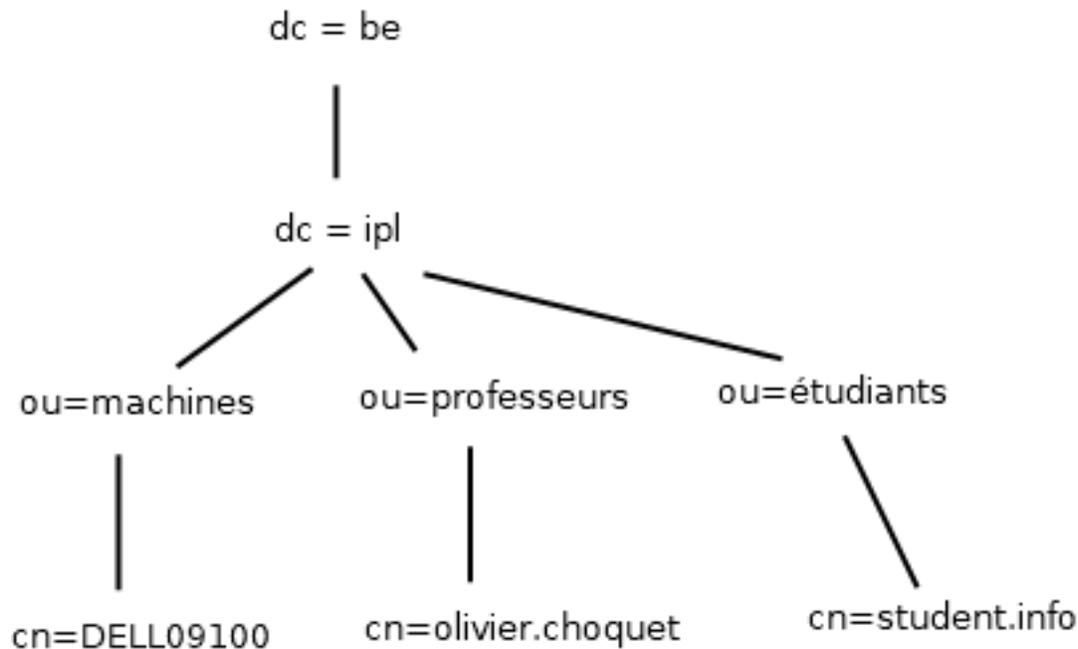
Un annuaire sera élaboré à partir d'une structure de données de type arbre hiérarchique représentant l'organisation d'une entreprise. La racine de cet arbre sera un nom DNS (le nom DNS de l'entreprise généralement),

les nœuds seront les divisions de l'entreprise (départements, sections, année d'étude ...) et les feuilles seront les objets (machines ou utilisateurs principalement).

Voici le vocabulaire de base utilisé dans un annuaire LDAP :

- **DC : Domain Component. Racine de l'arbre**
- **DN : Distinguished Name. Chemin complet vers un élément (Les DN sont uniques)**
- **OU : Organizational Unit. Division de l'entreprise rassemblant des CN.**
- **CN : Common Name. Nom d'un élément**

Exemple :



DN : «cn=olivier.choquet,ou=professeurs, dc=ipl,dc=be» CN : «olivier.choquet»

LDAP s'impose comme moyen d'authentification car il est bien standardisé et documenté. De plus sa structure de données en arbre est bien adaptée en termes de performance à l'authentification (beaucoup de lectures et peu d'écriture).

8.3 Modèle fonctionnel

LDAP a défini différentes méthodes permettant de modifier et consulter l'annuaire. Voici la liste des principales méthodes :

- **Bind : s'authentifier auprès du serveur LDAP. Ceci est nécessaire avant de demander au serveur une opération au serveur**
- **Add/Modify/Delete : mise à jour de l'annuaire.**
- **Search : «search» permettra de rechercher un élément ou plusieurs éléments dans l'annuaire en précisant une base, une portée et éventuellement des filtres (voir ci-dessous).**
- **Compare : vérifie qu'un élément contient ou non un attribut**
- **Unbind : se déconnecter du serveur**

8.3.1 Base, Portée et Filtres

Une recherche dans l'annuaire devra préciser ces 3 éléments. La base indique l'endroit où la recherche commence dans l'annuaire (nœud dans l'arbre). La portée indique où l'on s'arrête et peut prendre comme

valeur :

- Base : uniquement la base
- One : fils directs
- Sub : récursif sur toute l'arborescence

Les filtres permettent de ne prendre en compte dans la recherche que des éléments respectant certaines règles (élément ayant l'attribut,). LDAP fournit différents opérateurs pour construire les filtres :

- = (égalité)
- ~= (approximation)
- >= (supérieur / inférieur ou égal)
- : (intersection)
- (OU logique)
- & (ET logique)
- ! (négation)

Exemple : `&(&(objectclass=prof)(cn=A*)(!(sexe=M)))` Une recherche LDAP est également appelée Query LDAP. Un annuaire LDAP peut également être interrogé via une URL. Exemple : <ldap://localhost:389/ou=professeurs,dc=ipl,dc=be?uid?sub>

8.4 Modèle d'informations

LDAP définit le modèle d'information suivant :

- **Entrée** : composé d'attributs, possède un type (classe d'objets)
- **Schéma** : définition des attributs possibles et classes d'objets
- **DN (Distinguished Name)**

Exemple :

```
dn: cn=olivier.choquet, ou=profs, dc=ipl,dc=be\  
objectClass: user\  
cn: olivier.choquet\  
mail: olivier.choquet@vinci.be\  
bureau: A050\  

```

8.5 Modèle de sécurité

Le transport des messages LDAP sera chiffré via SSL/TLS. LDAP présente différentes méthodes d'authentification. **L'utilisateur, une fois authentifié (via un bind), aura accès aux données suivant les règles établies dans les ACL (Access Control List).**

8.6 Modèle de réplication

Un modèle de réplication est prévu dans la norme LDAP, mais il n'est pas encore standardisé. La réplication est un élément important pour assurer une redondance qui reste le moyen privilégié par les administrateurs système pour se prémunir des pannes. Les différentes implémentations LDAP (Active Directory, OpenLDAP, ..) ont développé leurs systèmes de réplication.

À noter également que LDAP fournit un format d'échange standard nommé LDIF (LDAP Data Interchange Format) qui permet d'échanger/sauvegarder de l'information entre serveurs LDAP. Cependant celui-ci ne permet pas une réplication aisée.

8.7 LDAP vs SGBD

LDAP	SGBD
Optimiser pour la lecture, faible en écriture	Optimiser pour les lectures et écritures
Isolation pas garantie	Support Transaction (ACID)
Réplication aisée	Réplication plus complexe
Attribut multi-valeurs	/
Query LDAP	Query SQL
Modèle de données défini, mais extensible facilement	Définition du modèle par le développeur, extensible mais difficile

A noter qu'une base de données peut être utilisée comme implémentation d'une solution serveur LDAP.

8.8 Active Directory

L'Active Directory est une implémentation Microsoft d'un annuaire LDAP. Cette implémentation est évidemment adaptée aux environnements Microsoft. **L'Active Directory permettra de gérer l'authentification des utilisateurs d'un ou plusieurs domaines, de gérer les droits des utilisateurs via des groupes de sécurité.** L'Active Directory introduit son propre vocabulaire en plus du vocabulaire LDAP.

Un Active Directory définira une forêt qui sera composée d'arbres (domaine parent avec des domaines enfants) et/ou de domaines. Les domaines seront, quant à eux, constitués d'unités d'organisation et enfin d'ordinateurs, de groupes et utilisateurs.

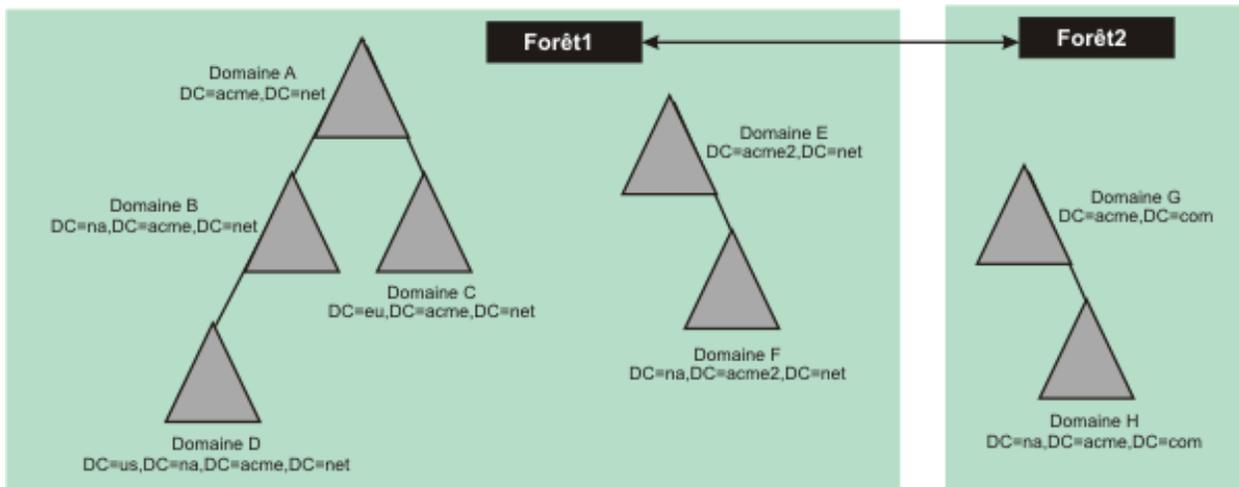


Image issue de ibm.com

Chaque domaine connaît ses objets, mais est incapable de localiser les objets d'un autre domaine. C'est pourquoi la forêt maintient un catalogue global qui sera consulté en cas de besoin par les domaines. Entre différentes forêts, on peut établir des relations d'approbations ce qui permet de s'authentifier dans la forêt 1 avec les identifiants de la forêt 2 et vice-versa.

Dans un Active Directory, nous allons retrouver différents objets. Les plus importants sont les utilisateurs, les ordinateurs du domaine, les groupes de sécurité, les GPO (Voir GPO), les unités d'organisation.

Un serveur hébergeant un Active Directory est appelé contrôleur de domaine. Il est conseillé d'avoir au minimum 2 contrôleurs de domaine par Active Directory pour se prémunir des pannes.

8.8.1 Unité d'organisation

Une unité d'organisation est un regroupement d'objets de l'Active Directory. C'est un nœud dans l'arbre LDAP. Les unités d'organisation trouvent essentiellement leur utilité par le fait que des GPO puissent être définies à ce niveau.

Ne pas confondre les unités d'organisation et les groupes de sécurité. Des GPO ne peuvent pas être définies sur des groupes et des permissions ne peuvent pas être définies sur des unités d'organisations. Un objet(utilisateur) peut appartenir à plusieurs groupes, il ne pourra pas contre être placé que dans une seule unité d'organisation.

8.8.2 Groupes de sécurité

Il est évidemment bien plus aisé de gérer les droits des utilisateurs via des groupes. Il existe différentes étendues de groupe :

- Domain Local : groupe utilisé uniquement dans son domaine de création
- Global : groupe utilisé dans tous les domaines approuvés
- Universel : groupe utilisé dans les domaines de la forêt

Microsoft propose une recommandation pour la gestion des droits et des groupes. Il s'agit de la recommandation AG(U)DLP.

Account → Global → (Universel) → Domain Local → Permission

L'idée est donc de placer les utilisateurs dans des groupes globaux, de placer ceux-ci dans groupe locaux et ensuite d'appliquer les permissions sur ses groupes. Éventuellement dans de grandes structures avec plusieurs domaines, un niveau de groupe universel peut être introduit.

Exemple :

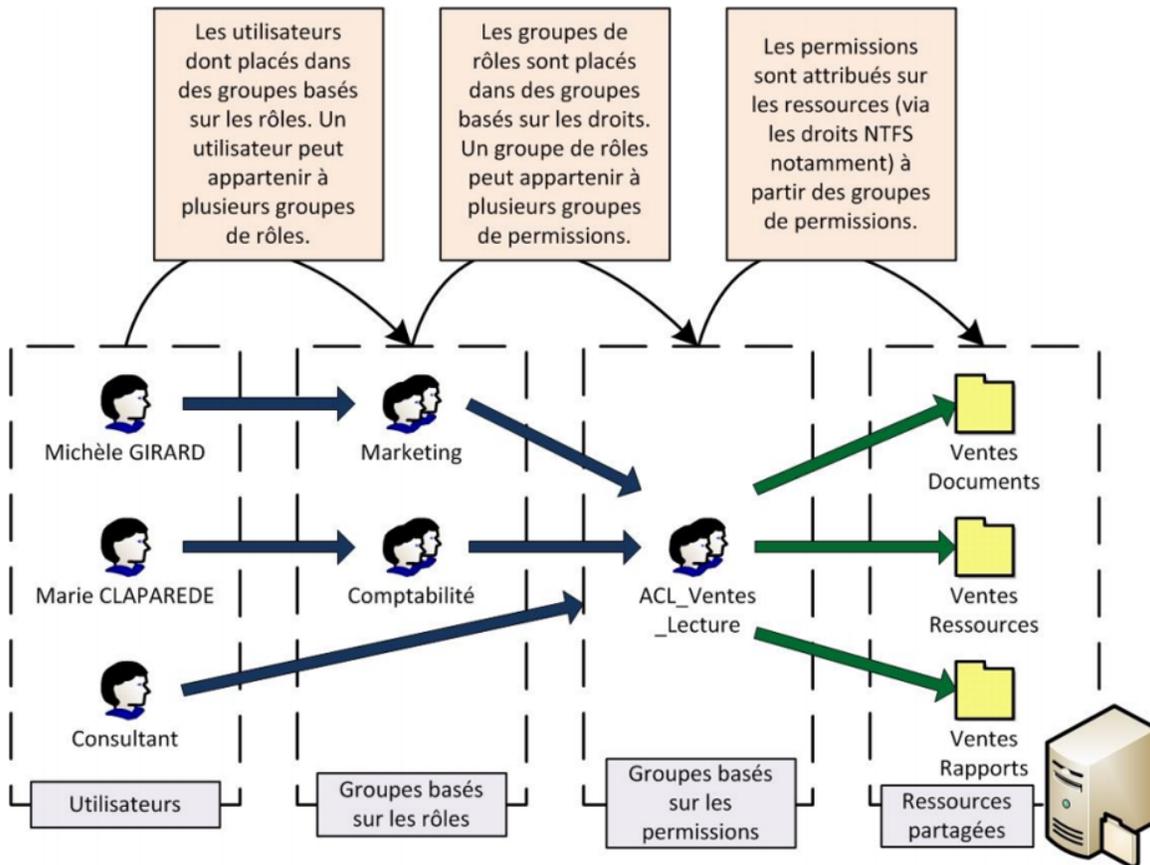


Image labo-microsoft.supinfo.com / schéma utile à connaître pour expliquer AG(U)DLP

8.8.3 Permissions

8.8.3.1 Permissions NTFS Microsoft utilise le système de fichiers NTFS et dès lors les permissions seront des permissions NTFS. Les permissions NTFS sont très riches (beaucoup de possibilités).

Voici les permissions les plus communes :

- R (Read)
- W (Write)
- RE (Read and Execution)
- M (Modify)
- FC (Full Control)

Les permissions sont héritées. Le répertoire C:\test héritera des permissions mises sur C:\. Il existe la possibilité de bloquer l'héritage. Les permissions NTFS définissent différentes étendues (ce dossier, ce dossier et sous-dossier, uniquement fichiers ...).

8.8.3.2 Permissions Partage réseau Les partages réseau sont des dossiers (présents généralement sur des serveurs) qui sont partagés et donc accessibles sur le réseau depuis n'importe quelle machine. Ces partages réseau utilisent le protocole SMB (Voir SAMBA).

Attention à ne pas confondre les permissions NTFS qui s'appliquent directement sur des fichiers et dossiers locaux (présents sur le système de fichiers) et les permissions de partage qui sont définies sur les partages réseau.

Les permissions de partage réseau sont plus limitées (R, M, FC). Lors de l'accès à un partage réseau, le système évalue tout d'abord les permissions du partage réseau et ensuite les permissions NTFS (puisque tout partage réseau se retrouve toujours physiquement sur un système de fichiers).

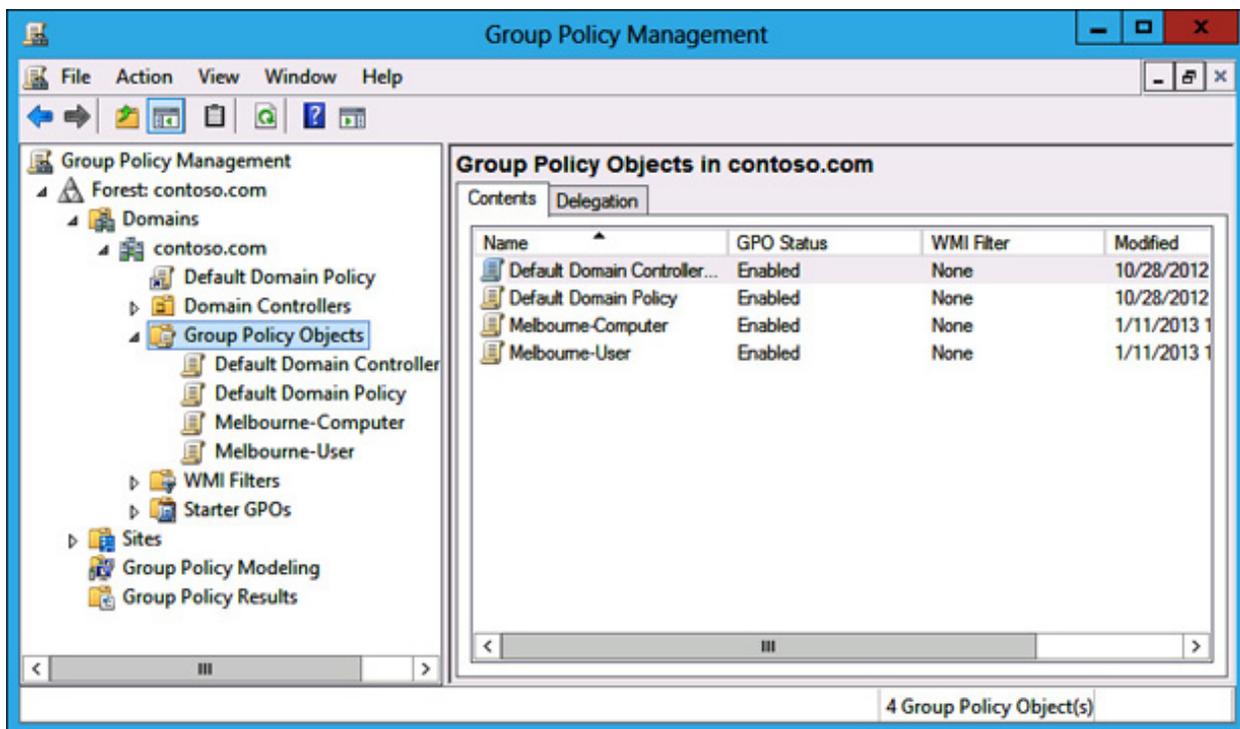
8.9 GPO

Les GPO (Group Policy Object) permettent de définir des stratégies de sécurité et/ou de configurer des paramètres de manière centralisée pour un domaine ou une forêt. Par exemple, on peut imposer un proxy pour les clients, définir un fond d'écran, définir une politique de mot de passe (longueur, complexité ...).

Les GPO, aussi appelées stratégie de groupe en français, permettent de déployer des stratégies/paramètres sur 2 types objets de l'Active Directory : les utilisateurs et les ordinateurs. Les GPO se créent le plus souvent sur des Organizational Unit.

Les GPO machines (configuration ordinateur) s'appliquent au démarrage de la machine. Les GPO utilisateurs (configuration utilisateur) s'appliquent à l'ouverture de session d'un utilisateur.

Console de gestion des GPO :



9 Gestion des données

9.1 Objectifs du chapitre

- Etablir un plan de sauvegarde des données

9.2 Base de données

En tant que développeur, vous savez que la majorité des données sont stockées dans des bases de données. Nous allons donc nous intéresser à ce composant essentiel de l'entreprise du point de vue administration

système.

Nous allons particulièrement au type de base de données le plus répandu à savoir les bases de données relationnelles. Nous prendrons PostgreSQL comme exemple.

9.2.1 DBA

Dans les grandes entreprises, il existe souvent des DBA(Database Administrator) qui sont en charge de gérer et documenter les bases de données. Dans les entreprises plus petites, c'est généralement l'administrateur système qui effectue ce travail.

Les rôles du DBA sont les suivants :

1. Documenter la base de données (DSD, accès à la base de données ...)
2. Gérer le stockage (croissance de la DB, défragmentation ...)
3. Vérifier la structuration de la base de données (normes, intégrité)
4. Sécuriser l'accès aux données (droits)
5. Optimiser la base de données (création d'index ...)
6. Fournir aux développeurs des jeux de données (DB test, anonymisation des données ...)
7. Migration/ mise à jour de la base de données
8. Sauvegarder la base de données

9.2.2 Exemple PostgreSQL

Un moteur de base de données est constitué de plusieurs programmes. PostgreSQL est composé d'un programme superviseur (postmaster), du serveur exécutant les requêtes (postgres) et d'un client interactif (psql).

```
apt-get install postgresql
```

9.2.2.1 Installation de PostgreSQL

9.2.2.2 Configuration de PostgreSQL PostgreSQL se configure via le fichier postgresql.conf et pg_hba.conf. Le premier fichier permet de gérer les paramètres globaux (nombre maximum de connexions, SSL ...) tandis que le deuxième permet de gérer les accès.

#	TYPE	DATABASE	USER	ADDRESS	METHOD
host	all		all	192.168.54.1/32	reject
host	all		all	0.0.0.0/0	krb5

9.2.2.3 Sauvegarde PostgreSQL Il est important de sauvegarder les bases de données. Voici les erreurs à ne pas commettre :

- Pas de sauvegarde
- Sauvegarde jamais testée
- Ne pas documenter sa sauvegarde (qui l'a fait/ quand, comment ...)
- Sauvegarder sur le même disque
- Sauvegarder au même endroit (incendie ?)
- N'avoir qu'une sauvegarde très récente car elle contiendra également l'erreur

Les sauvegardes de type fichiers ou instantanés ne conviennent pas aux bases de données. Il est préférable d'utiliser l'outil fourni par le moteur de base de données. Dans le cas de PostgreSQL, nous avons pg_dump.

9.2.2.4 Performance PostgreSQL Il est important de savoir que les limitations des bases de données se situent essentiellement du côté des composants sous-jacents (CPU, RAM, vitesse disque, capacité disque ...).

Par exemple, la limite au niveau des tables pour PostgreSQL est actuellement de 64 To. Ce n'est donc pas ceci qui va poser problème. PostgreSQL reste stable et performant même avec des DB 30 GB.

Sachant ceci, il faut commencer par surveiller ces éléments via les commandes/outils du système d'exploitation. Par exemple : uptime pour la charge CPU, free pour la charge mémoire.

La performance passe également par l'ajout d'index. Un index peut accélérer par 1000 000 une requête ! PostgreSQL dispose de la commande EXPLAIN qui décrit le plan d'exécution d'une requête. Ceci s'avère être un outil puissant pour accélérer une requête problématique.

L'utilisation de LIMIT est également à envisager. Doit-on vraiment charger l'entièreté d'une table, les premiers résultats ne sont-ils pas suffisants ?

L'utilisation régulière de VACUUM pour supprimer définitivement les données expirées est à envisager, particulièrement sur les bases de données des développeurs qui créent et détruisent souvent leurs bases de données.

L'utilisation d'ANALYSE permet de collecter des statistiques en vue d'optimiser la base de données.

Un petit mot sur les ORM pour finir. Ceux-ci sont très utiles, mais l'optimisation s'avère souvent plus complexe à gérer vu que vous laissez l'ORM créer les requêtes SQL pour vous. En tant que développeur, posez-vous (ou à votre chef de projet) donc la question de la performance avant de vous décider.

9.2.2.5 Réplication PostgreSQL Comme d'habitude, les administrateurs systèmes ou DBA vont vouloir se prémunir des pannes en installant de la redondance. Avec les bases de données, ceci peut se faire en activant la réplication. Ce mécanisme permet à 2 moteurs de base de données de partager des bases de données. Les données de ces bases de données se retrouvent alors répliquées/dupliquées au minimum sur 2 serveurs de bases de données.

La réplication la plus utilisée est celle dite du maître – esclave. Toutes les opérations sont faites sur le serveur maître et celui-ci envoie régulièrement au serveur esclave ses journaux de transactions. Ce dernier rejoue alors le journal de transactions sur sa base de données.

9.3 Sauvegardes

Un administrateur système sera en charge de la pérennité des données. Pour cette tâche, des sauvegardes ou backups seront nécessaires. Les sauvegardes permettent de réduire les risques liés à des pannes, virus (ransomware), erreurs humaines. **Bien sûr il est souvent impossible de tout sauvegarder, c'est pourquoi il faut définir une politique de sauvegarde qui répond aux questions suivantes.**

9.3.1 Que faut-il sauvegarder ?

Différents types de données se retrouvent dans une entreprise à savoir :

1. Fichiers utilisateurs
2. Système d'exploitation
3. Programmes
4. Base de données
5. Configuration des utilisateurs (profil des utilisateurs)
6. Configuration de programmes
7. ...

Comme il est souvent impossible de tout sauvegarder (pour des raisons monétaires), il faut établir de priorités. Cette priorisation trouve souvent son origine dans la gestion des risques liés à l'entreprise.

Exemple : À l'IPL, les fichiers utilisateurs (U) ne sont pas sauvegardés. Ceci n'est pas prioritaire. Par contre, sa comptabilité et son fichier étudiant le sont.

9.3.2 Sur quel support sauvegarder ?

Pour une entreprise, contrairement aux particuliers l'emploi de disques durs externes n'est pas efficace. Il est également vital de stocker les sauvegardes à distance. Stocker les sauvegardes au sein même de l'entreprise est une très mauvaise idée. Imaginez un incendie et vos données et backups disparaîtront en même temps.

Les entreprises auront donc souvent recours à des NAS (Network Area Storage) ou SAN distant ou encore à un stockage dans le Cloud. Dans ce dernier cas, la confidentialité des données doit être évoquée.

9.3.3 Quel type de sauvegarde, quelle fréquence ?

Il existe différents types de sauvegarde à savoir :

1. **complète** : une copie complète des données est faite
2. **incrémentielle** : une copie des modifications depuis la dernière sauvegarde (complète ou incrémentielle) est faite
3. **différentielle** : une copie des modifications depuis la dernière sauvegarde complète est faite
4. **miroir** : une seule sauvegarde complète.
5. **instantané/snapshot** : pas vraiment une sauvegarde car difficile de restaurer un seul fichier

Souvent la fréquence de sauvegarde va influencer le type de sauvegarde.

9.3.4 Conservation des données ?

Les sauvegardes peuvent occuper beaucoup de place. Il est donc nécessaire à réfléchir au stockage et à comment gagner de la place. La compression des données est un moyen souvent utilisé. **La déduplication est un autre moyen mis en place par les entreprises. Ce mécanisme est souvent intégré au SAN.** Il s'agit de découper les données en bloc et toute nouvelle occurrence d'un bloc est remplacée par un pointeur. Ce mécanisme fonctionne très bien avec les backups car beaucoup de données se répètent. Imaginez une fréquence de backups complets tous les mois. Dans ces backups, beaucoup de données seront identiques.

Il est nécessaire de crypter ses backups pour éviter à une personne mal intentionnée de récupérer facilement les données de l'entreprise. L'utilisation grandissante du Cloud renforce également ce besoin.

9.3.5 Outils de sauvegarde

Il existe bien évidemment différents logiciels sur le marché offrant la possibilité de réaliser les types de sauvegardes cités ci-dessus. Souvent ceux-ci se basent sur des outils/protocoles déjà fort connus comme le SSH ou FTP pour transférer les backups à distance, des outils zip pour la compression.

Sous Linux, l'outil rsync est fortement utilisé dans des scripts maison ou par le biais de logiciels.

9.3.6 La règle du 3-2-1

Il existe une règle fortement conseillée pour les sauvegardes de données. Il s'agit de la fameuse règle 3-2-1.

- **3** : conserver 3 copies des données (originale + 2 sauvegardes)

- **2 : conserver les données sur 2 supports différents**
- **1 : conserver 1 copie de sauvegarde dans un lieu différent (des 2 autres copies)**

9.4 Quotas

Un administrateur système doit veiller à ce que les ressources soient utilisées de manière raisonnée. Il est également impératif que ces serveurs restent opérationnels et ne soient pas bloqués à cause d'un manque d'espace disque provoqué par un utilisateur.

Les quotas permettent donc de régler ce problème en imposant une limite aux utilisateurs. Le mécanisme de quotas propose une limite soft et une limite hard. Quand un utilisateur dépasse sa limite soft, il reçoit un avertissement lui indiquant qu'il peut continuer à travailler pour une période de grâce définie par l'administrateur. Une fois cette période grâce écoulée, il est bloqué s'il n'est pas redescendu en dessous de la limite soft.

Si l'utilisateur dépasse la limite hard, il est bloqué directement.

Exemple : si on définit que les utilisateurs ont droit à un espace de 100MB, on fixera une limite soft à 75MB et la limite hard à 100MB.

10 Virtualisation

10.1 Objectifs du chapitre

- **Comprendre la virtualisation, les différents types de virtualisation**

Malgré le fait que le terme virtualisation s'est popularisé depuis le début des années 2000 avec les solutions de virtualisation PC, cela reste un concept assez ancien qui date du début des années 80. On peut en fait virtualiser bien plus qu'un PC.

10.2 Avantages de la virtualisation

La virtualisation apporte des avantages certains dans un environnement informatique d'où l'engouement généré par ce concept actuellement. La définition de la virtualisation est la suivante :

La virtualisation consiste en l'abstraction d'un élément du monde réel en le rendant virtuel. Ceci dans l'objectif de rendre l'élément plus facilement :

- **Configurable**
- **Transportable**
- **Optimisé (meilleure allocation des ressources)**
- **Disponible (facilité de déploiement)**
- **Sécurisé (isolation)**

À noter que chaque solution de virtualisation n'a pas forcément pour but de rencontrer l'ensemble de ces avantages.

Voici quelques exemples concrets :

1. **OPTIMISATION** : La virtualisation des serveurs permet d'optimiser l'allocation des ressources (mémoire, CPU) à la demande et donc d'éviter un gaspillage de ressources serveur. Au total, il faudra moins de serveurs physiques, cela va se répercuter positivement sur la facture électrique de l'entreprise.
2. **TRANSPORT** : Les solutions de virtualisation de systèmes d'exploitation (VirtualBox) permettent un transport facile de tout un système. Il s'agit de simples fichiers et VirtualBox peut s'installer sur n'importe quel système d'exploitation hôte.
3. **CONFIGURABLE** : Les VLAN (voir VLAN) permettent une configuration plus aisée des réseaux via une console Web. Sans les VLAN, on doit physiquement brancher/débrancher des câbles si on veut changer une machine de sous-réseau.

4. **SÉCURITÉ** : Les solutions de virtualisation de systèmes d'exploitation ou serveurs permettent de prendre des instantanés (snapshot) de l'état du système. Cela s'avère très utile et pratique avant toute mise à jour périlleuse.
5. **DISPONIBILITÉ** : Déploiement d'images de pc/serveurs à la demande

10.3 Types de virtualisation

En informatique, on a essayé depuis très longtemps de virtualiser un maximum de composants/concepts afin de tirer profit des avantages de la virtualisation cités ci-dessus.

Type de virtualisation	But recherché	Exemple de solution
Serveurs	Regrouper/Diminuer le nombre de serveurs physiques	VMWare ESXi, HyperV
Applications	Faciliter le déploiement	ThinApp, XenApp, Docker
Poste de Travail	Faciliter le déploiement	Virtual Desktop Infrastructure (VDI)
Stockage	Mutualiser le stockage, diminuer le nombre de disques	Storage Area Network (SAN)
Réseau	Faciliter la configuration	Virtual LAN (vLAN)

À noter que même si Docker n'est pas un logiciel de virtualisation à proprement parlé (Voir Docker), son succès dépend également du fait qu'il permet de créer une image pour une application et donc de faciliter son déploiement. C'est pourquoi si on devait classer Docker dans ce tableau il serait dans la virtualisation d'applications.

10.4 Inconvénients de la virtualisation

L'inconvénient majeur de la virtualisation est l'ajout d'une couche de virtualisation entre le système physique et le composant virtualisé. Ceci permet d'obtenir les avantages cités ci-dessus, mais dégrade les performances.

Il existe également des dangers liés à la virtualisation. Un système virtualisé est plus complexe qu'un système purement physique. Cette complexité doit être maîtrisée par l'équipe informatique. Un autre danger est de croire que tout peut se virtualiser. La virtualisation progresse de plus en plus, mais il reste difficile de l'utiliser dans certaines situations comme par exemple :

- les machines des développeurs. Ceux-ci utilisent des technologies à la pointe où l'on a peu de recul sur leur interaction/adéquation avec la virtualisation
- toute connectique externe est difficile à virtualiser (port, composants spécifiques utilisés en industrie).

10.5 Hyperviseurs

La virtualisation des machines se fait via des hyperviseurs. **Un hyperviseur est un logiciel de virtualisation permettant à plusieurs machines virtuelles de fonctionner simultanément sur un même système physique.**

On distingue 2 types d'hyperviseurs : 1. **Hyperviseur type 1 / natif / bare-metal** 2. **Hyperviseur type 2 / hosted**

Un hyperviseur de type 1 sera utilisé dans le cadre de la virtualisation de serveurs. Il s'agit d'un système d'exploitation spécialement dédié à virtualisation qui s'exécute directement sur le matériel.

Un exemple : À l'IPL, nous avons 3 serveurs physiques sur lesquels est installé VMWare ESX(hyperviseur de type 1). Celui-ci permet ensuite de créer des serveurs virtuels qui seront répartis sur les 3 serveurs physiques. L'avantage est que l'administrateur système peut via cet hyperviseur contrôler assez finement l'allocation des

ressources. Il peut notamment facilement reprendre des ressources d'un serveur (car moins utilisé à certains moments) et les attribuer à un autre serveur. Ceci est impossible à faire avec une architecture purement physique.

A noter que depuis l'apparition du Cloud, de nombreuses offres proposent des **serveurs privés virtuel (VPS)**. Il s'agit d'une illustration de l'utilisation d'hyperviseur bare-metal par des hébergeurs Cloud.

Ce que vous connaissez le plus est certainement **les hyperviseurs de type 2. Ces hyperviseurs s'installent dans un système d'exploitation comme un logiciel classique**. On peut citer **VirtualBox**, **VMWare Workstation** dans cette catégorie. Cette catégorie d'hyperviseur est la moins performante, mais elle est très utile pour effectuer facilement des tests sans perturber l'existant. En effet, les machines virtuelles créées via ces hyperviseurs sont par défaut isolées du réseau physique (NAT) et on peut évidemment installer ce que l'on veut dans ces machines virtuelles sans perturbation pour la machine physique (hôte).

10.6 Virtualisation du stockage

Lorsqu'on virtualise les serveurs, il devient rapidement intéressant et nécessaire que les ressources disques des serveurs virtuels soient également virtuelles. En effet, si on attribue des ressources disques physiques à un serveur virtuel, on ne pourra pas facilement augmenter ou diminuer les ressources disques consommées par ce serveur virtuel.

La virtualisation du stockage consiste donc en la création d'une baie de stockage (ensemble de disques physiques) qui sera présentée en un ou plusieurs ensembles logiques et dynamiques. On parle de LUN (Logical Unit Number). Une LUN est donc un espace de stockage logique et dynamique.

Ces baies de stockage sont accessibles via le réseau essentiellement via 2 protocoles (SMB, Fiber Channel). On parle de SAN (Storage Area Network). **Un SAN est donc une baie de stockage avec des LUN accessibles via le réseau.**

10.7 Virtualisation du réseau

Dans une infrastructure, les administrateurs système virtualiseront le réseau. Ils auront recours essentiellement aux VLAN (Voir VLAN).

Les hyperviseurs vont également présenter aux serveurs virtuels des carte réseaux virtuelles que l'on pourra configurer (NAT, Bridge, Adresse IP, vlan, ...).

Les hébergeurs Cloud ont également la nécessité de virtualiser leur réseau pour permettre à leurs clients de configurer l'environnement acheté/loué.

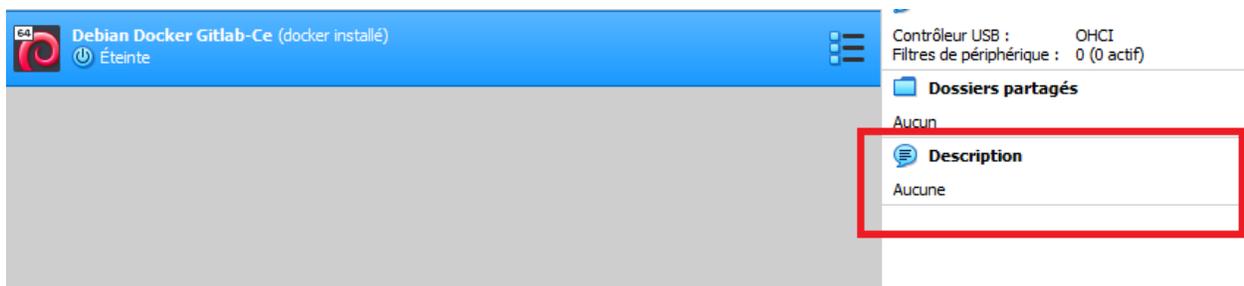
10.8 Le cas VirtualBox

Virtualbox est une solution de virtualisation bien connue utile pour installer/tester d'autres systèmes sur sa machine sans la nécessité d'altérer celle-ci (pas de repartitionement, dual boot, ...).

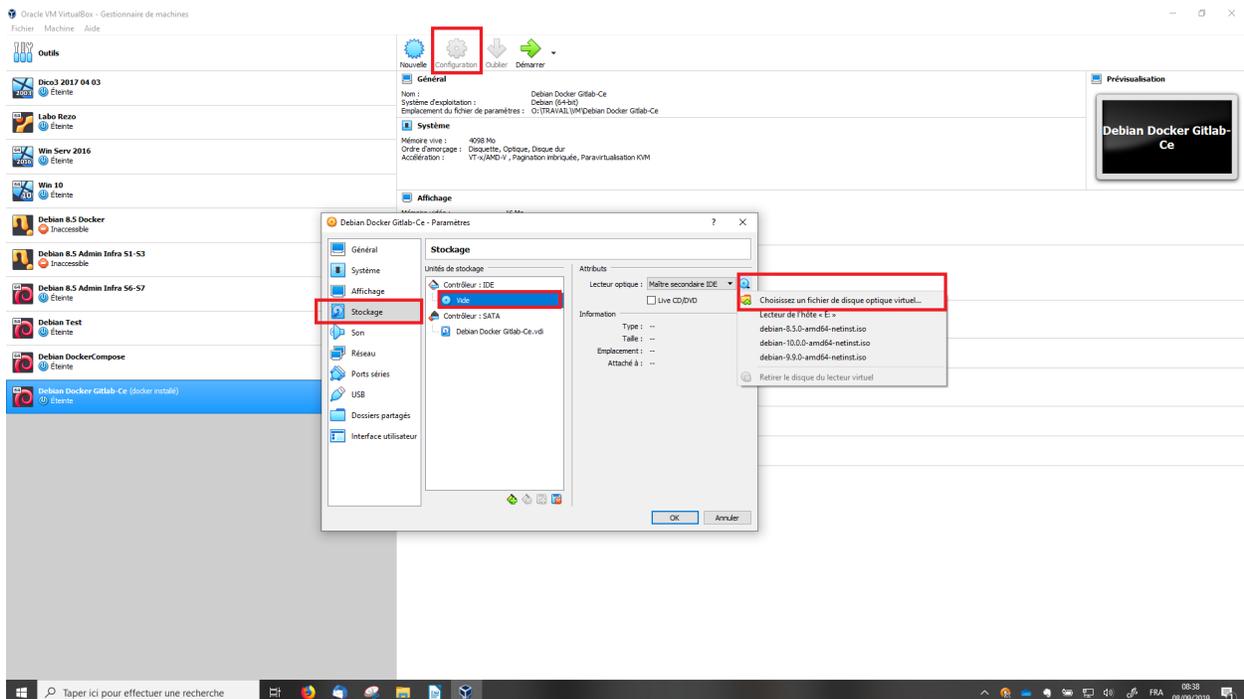
La possibilité dans le solutions de virtualisation de réaliser un instantané/snapshot est un énorme avantage. Ceci permet de revenir facilement à un état antérieur en cas de souci. **Il est donc vivement conseillé de réaliser un instantané avant toute modification importante d'un système.**



On peut facilement laisser des instructions/commentaires dans la zone description prévue par Virtualbox. Utile par exemple pour laisser les comptes/mot de passes de test, ils seront ainsi toujours au même endroit que vos fichiers du système virtualisé.



L'installation d'un système d'exploitation dans VirtualBox se fait la plupart du temps via un ISO (Debian, Windows) que l'on place dans le lecteur de cdrom virtuel.

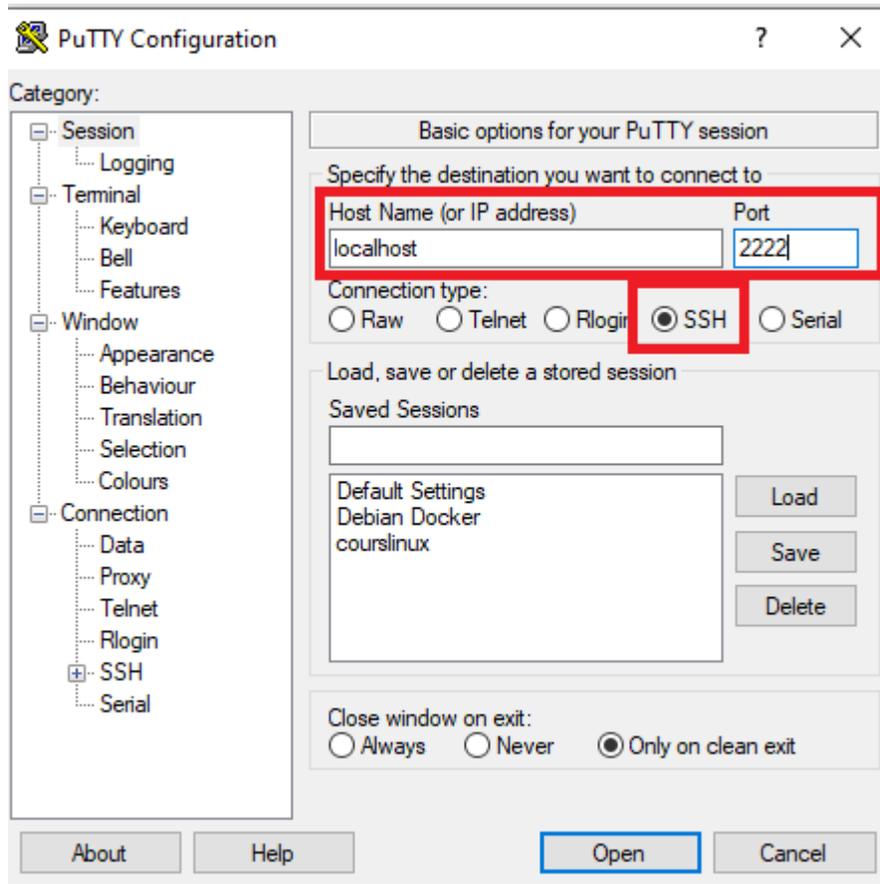


10.8.1 Types de réseau sous VBOX

10.8.1.1 NAT Au niveau réseau, votre machine virtuelle Debian est configurée en NAT par défaut. Via cette configuration, vous ne pouvez pas par défaut accéder depuis votre machine (machine hôte) à la machine virtuelle Debian (machine invitée). Il est cependant utile de pouvoir accéder à votre machine invitée via le réseau afin de tester les différents services que vous installerez (Serveur Web, SSH, ...). Pour cela il faut simplement configurer la redirection de port du NAT. L'idée est la suivante : faire correspondre un numéro de port de la machine hôte à un port de la machine invitée. Toutes les requêtes adressées alors sur le port de la machine hôte sont redirigées vers le port de la machine invitée. Un exemple :

The screenshot shows the configuration of network settings for a Debian 8.5 virtual machine in VirtualBox. The main window is 'Debian 8.5 - Paramètres' with the 'Réseau' tab selected. Under 'Mode d'accès réseau', 'NAT' is chosen. The 'Avancé' section is expanded, and the 'Redirection de ports' checkbox is checked. Below this, the 'Règles de redirection de ports' window is open, displaying a table of port forwarding rules.

Nom	Protocole	IP hôte	Port hôte	IP invité	Port invité
SSH	TCP		2222		22
TELNET	TCP		2323		23



10.8.1.2 Pont / Bridge Une autre manière de configurer sa carte réseau sous VirtualBox est de créer un pont. Votre machine virtuelle sera alors connectée directement au même réseau que votre machine hôte. La machine virtuelle pourra communiquer avec toutes les machines du réseau.

10.8.1.3 Host-only networking Il est souvent utile de reproduire en virtuel un environnement proche de la réalité. Celui-ci implique souvent le recours à plusieurs machines virtuelles. Exemple : pour tester un partage Samba présent sur un serveur Linux, il est utile d'avoir une machine virtuelle « client Windows ».

Comment faire alors pour que ces 2 machines virtuelles invitées communiquent entre elles ?

VirtualBox propose une solution simple : le host-only networking ou réseau privé d'hôte. En ajoutant une carte réseau configurée sur ce mode réseau, les machines virtuelles invitées pourront communiquer entre elles sans perturbation pour le « véritable » réseau physique connecté à la machine hôte. Il faudra cependant définir un réseau et des adresses IPs pour les machines virtuelles hôtes.

11 Conteneurs (Docker)

11.1 Objectifs du chapitre

- Comprendre les architectures à base de conteneurs
- Savoir créer un Dockerfile
- Déployer des applications via Docker, docker-compose

11.2 Docker

11.2.1 Introduction

Docker est une solution d'architecture à base de conteneurs. Les architectures à base de conteneurs sont une évolution avantageuse de la virtualisation. La virtualisation permet une abstraction vis-à-vis de la couche matérielle et donc de déployer du code, des environnements plus facilement. Cependant la virtualisation au niveau matériel reste lourde en termes de performance et de stockage contrairement aux architectures à base de conteneurs.

On distingue donc actuellement la virtualisation matérielle par opposition à la virtualisation au niveau système d'exploitation aussi appelée architecture à base de conteneurs. C'est à cette catégorie qu'appartient Docker. **Docker n'est donc pas un logiciel de virtualisation, mais un isolateur.**

Pour réaliser cette isolation, Docker s'appuie sur 2 éléments introduits dans le noyau Linux : les cgroups et les namespaces.

Les cgroups (Control Groups) permettent de fixer/limiter les ressources (CPU, Réseau, disque, nombre de processus) allouées à un conteneur ou un ensemble de conteneurs.

Les namespaces permettent d'isoler des ressources. Ainsi un conteneur ou ensemble de conteneurs ne voient que les ressources de son namespace.

Les conteneurs sont proches des machines virtuelles, mais présentent un avantage important. Alors que la virtualisation consiste à exécuter de nombreux systèmes d'exploitation sur un seul et même système, les containers se partagent le même noyau de système d'exploitation et isolent les processus de l'application du reste du système.

Pour faire simple, plutôt que de virtualiser le hardware comme l'hyperviseur, le conteneur virtualise le système d'exploitation. Il est donc nettement plus efficace qu'un hyperviseur en termes de consommation des ressources système. Concrètement, il est possible d'exécuter près de 4 à 6 fois plus d'instances d'applications avec un container qu'avec des machines virtuelles sur le même hardware.

Les architectures à base de conteneurs sont très utilisées car elles permettent facilement des mises à l'échelle (scaling).

En effet, on peut facilement augmenter le nombre de conteneurs lorsqu'une application est fortement demandée. On appelle cela la mise à échelle horizontale (**horizontal scaling**). On peut également augmenter les ressources des conteneurs via les cgroups. On appelle cela la mise à échelle verticale (**vertical scaling**). On peut bien évidemment combiner les 2 mises à l'échelle. Quand la demande est moins forte, on peut réduire le nombre de conteneurs et/ou les ressources et par conséquent épargner de l'argent.

11.2.2 Docker vs Virtualisation

La grosse différence entre Docker et les systèmes de virtualisation classiques se situe au niveau de l'interaction avec le système d'exploitation hôte.

- Docker partagera/utilisera le même noyau(Linux) pour tous les conteneurs.
- Chaque VM géré par un hyperviseur aura son propre OS

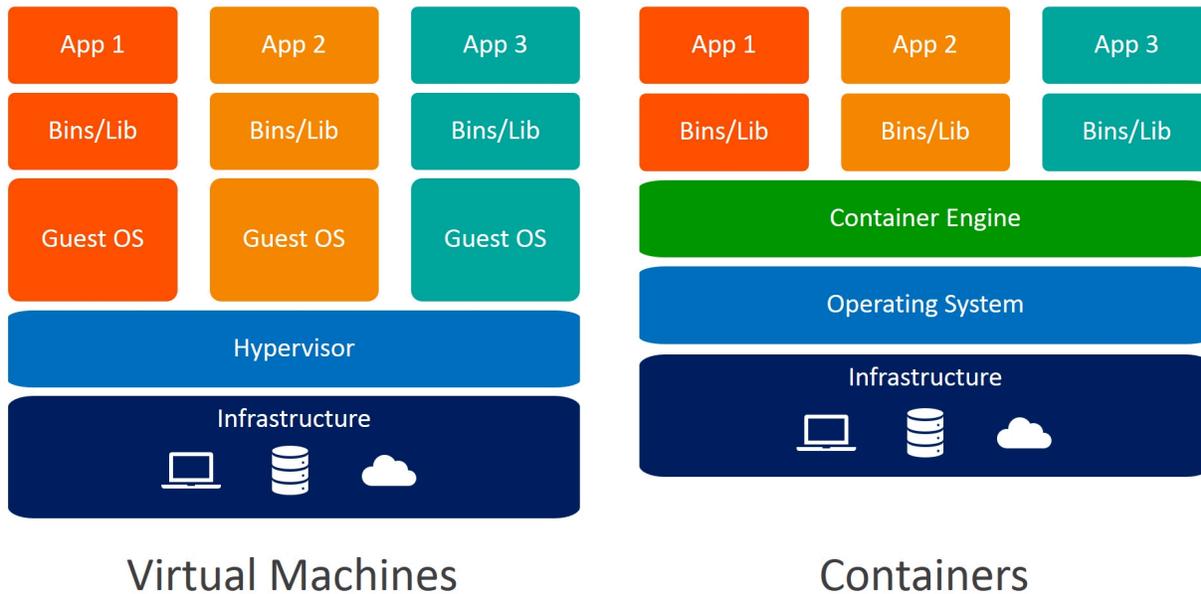


Image issue de : <https://www.weave.works/blog/a-practical-guide-to-choosing-between-docker-containers-and-vms>

Si on reprend les caractéristiques de la virtualisation pour comparaison :

Différences	Docker	Machine Virtuelle
Optimisation	léger (taille et empreinte mémoire)	+ lourd (taille et empreinte mémoire)
Transport	Transport facile (taille légère)	Taille élevée
Disponibilité	Dockerhub et registry privée	Images sur Azure, ...
Configurable	moins d'options configurables	+ options configurables
Sécurité	moins de sécurité (partage du noyau par les conteneurs)	isolation peut être totale

11.2.3 Installation de Docker

Il existe une version de Docker avec interface graphique nommée Docker-Desktop disponible pour Windows, Linux et Mac. Celle-ci est très utile pour les développeurs pour tester la conteneurisation de leurs applications.

Dans ce cours, nous installerons Docker Engine CE. Il s'agit simplement du moteur Docker version Community Edition sans interface graphique.

Sous Debian :

Supprimer éventuellement une ancienne version de docker :

```
for pkg in docker.io docker-doc docker-compose podman-docker containerd runc; do apt-get
→ remove $pkg; done
```

Installer les packages pour permettre à apt de travailler en HTTPS :

```
apt-get update
apt-get install ca-certificates curl gnupg
```

Ajouter la clé GPG officielle du dépôt docker :

```
install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o
  ↪ /etc/apt/keyrings/docker.gpg
chmod a+r /etc/apt/keyrings/docker.gpg
```

Ajouter le dépôt docker au sources.list :

```
echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg]
  ↪ https://download.docker.com/linux/debian \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Installer de Docker :

```
apt-get update
apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
  ↪ docker-compose-plugin
```

Vérification de l'installation :

```
docker run hello-world
```

11.2.4 Utilisation de Docker

Voyons comment commencer avec Docker c'est-à-dire créer un conteneur pour une application.

Pour déployer une application via Docker, il y a donc 3 étapes essentielles :

- Création d'un Dockerfile (recette de cuisine de l'application à déployer)
- Créer l'image Docker (`docker build -t <imagename> <pathToDockerfile>`)
- Créer un conteneur à partir de l'image créée (`docker run -d -p 9000 :80 --name <containername> <imagename>`)

Le schéma ci-dessous représente toutes les commandes Docker.

Remarquez l'enchaînement des 3 étapes essentielles décrites ci-dessus.

Les points suivants de ce syllabus détailleront les différents éléments de ce schéma à savoir :

- *Le Dockerfile et ses principales instructions*
- *Les commandes de gestion des images Docker*
- *Les commandes de gestion des conteneurs Docker*
- *Les Registry et DockerHub*

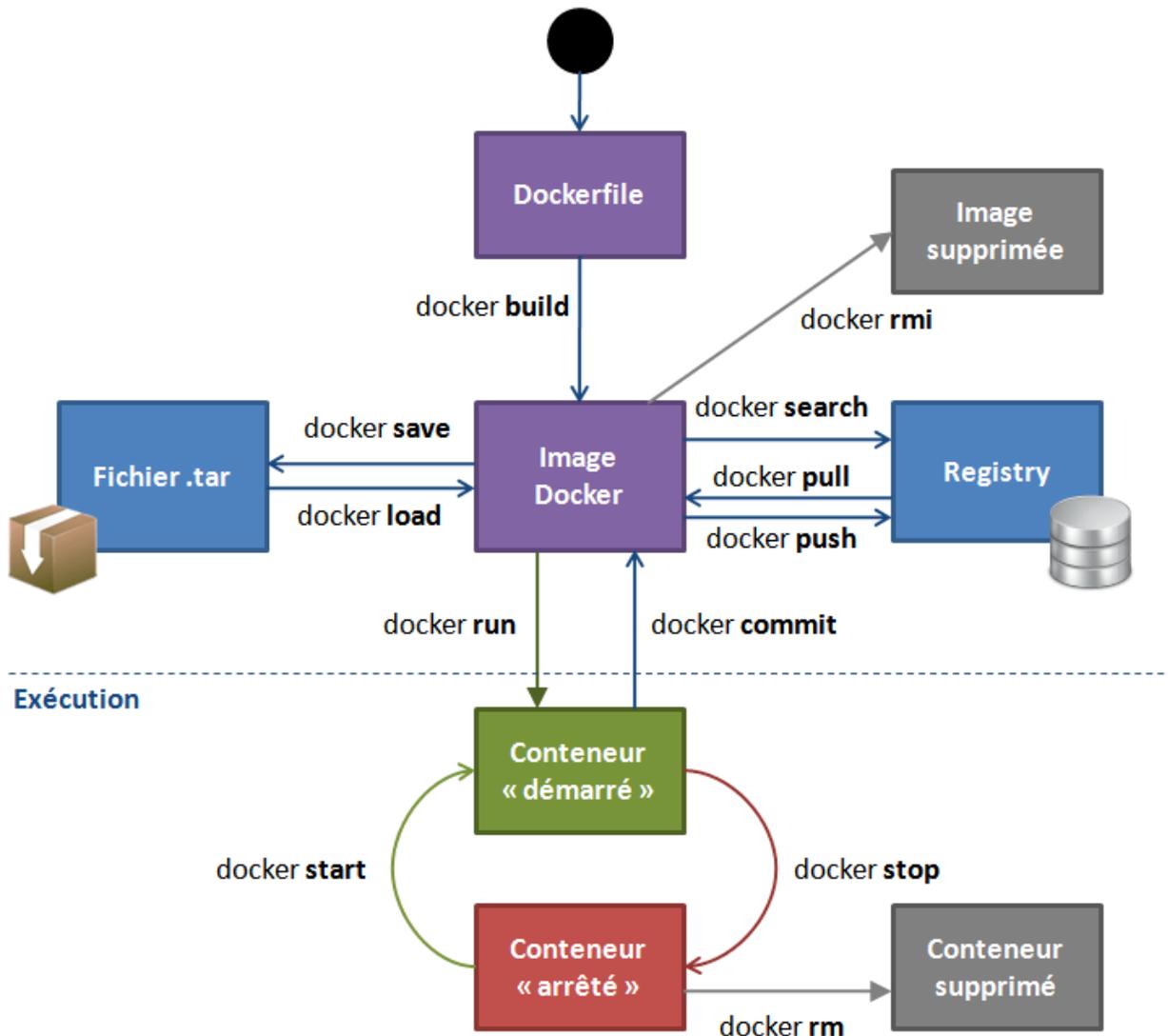


Image issue de <https://www.atomit.fr/2016/10/les-bases-sur-docker/>

11.2.4.1 Dockerfile et instructions principales La conteneurisation d'une application commence par la création d'un dossier contenant tout ce qui est nécessaire à l'application. Il est nécessaire d'avoir dans ce dossier au minimum :

- le code de l'application
- un fichier Dockerfile

Voici les principales instructions à utiliser dans le Dockerfile :

On crée une image Docker pour notre application en se basant sur une image déjà existante (Debian, Apache, Nodejs, ...). Ces images proviennent d'une registry. La registry par défaut est le DockerHub. Un Dockerfile commence donc toujours par l'instruction FROM.

```
FROM debian
```

La commande LABEL permet d'ajouter des métadonnées. Ceci n'est pas nécessaire au fonctionnement de l'image/conteneur. Cela sert plus de documentation. La syntaxe est LABEL key=value. Key et value sont libres.

```
LABEL maintainer="Olivier Choquet"  
LABEL description="image custom SSH"  
LABEL version="1.0"
```

La commande RUN permet d'ajouter à l'image de base des éléments en exécutant une/des commandes.

```
RUN apt-get install ssh -q
```

La commande COPY est indispensable. Elle permet de copier des fichiers/répertoires depuis la machine hôte dans l'image/conteneur. La syntaxe est COPY SRC (machine hôte) DST (image/conteneur). Les deux arguments doivent être soit des fichiers, soit des répertoires. Dans le cas d'un répertoire, COPY effectue une copie récursive du répertoire.

```
COPY maconfig.conf /etc/maconfig.conf
```

La commande CMD permet d'exécuter une commande une fois le conteneur démarré.

```
CMD npm start
```

Vous retrouverez plus de détails et plus de commandes dans la documentation Docker si nécessaire :

[Documentation Instructions Dockerfile](#)

11.2.4.2 Les commandes de gestion des images Docker Quand votre Dockerfile est prêt, il est nécessaire de créer une image afin de pouvoir ensuite créer des conteneurs à partir de cette image.

La commande pour créer une image est build. Il est intéressant de tagger/donner un nom à son image. Cela se fait avec l'option -t. Il faut donner à la commande build le chemin vers le Dockerfile. Dans l'exemple ci-dessous, le Dockerfile se trouve dans le répertoire courant d'où le . en fin de commande.

Créer une image :

```
docker build -t "monimagessh" .
```

Lister les images :

```
docker images
```

Supprimer une image :

```
docker rmi monimagessh
```

11.2.4.3 Les commandes de gestion des conteneurs Docker La commande pour créer un conteneur est run. Celle-ci prend plusieurs arguments :

- -p :port forwarding
- -d : exécution en tant que démon, le prompt est rendu
- --name : donner un nom au conteneur (ce nom peut être utilisé ensuite à la place du container_id)

Créer un conteneur à partir d'une image :

```
docker run -d -p 2222:22 --name contssh monimagessh
```

Lister tous les conteneurs :

```
docker ps -a
```

Démarrer, arrêter un conteneur :

```
docker start/stop/contssh
```

Supprimer un conteneur :

```
docker rm contssh
```

Se connecter à l'intérieur d'un conteneur :

```
docker exec -it contssh bash
```

Déboguer un conteneur :

```
docker logs contssh
```

Arrêter tous les conteneurs :

```
docker stop $(docker ps -aq)
```

Supprimer les conteneurs et images non utilisées pour plus de clarté :

```
docker system prune -a
```

11.2.4.4 Registry et DockerHub Une registry Docker est simplement un dépôt avec des images Docker. La registry la plus connue et utilisée par défaut par Docker est le DockerHub. Celle-ci est publique et contient des images officielles pour la majorité des briques de base utilisées par les applications faites par les développeurs.

On peut par exemple y retrouver : Debian, Apache, NodeJS, MongoDB, MySQL, ...

Le DockerHub est l'endroit où vous devez rechercher votre image de base !

Les images présentes sur le DockerHub ont des tags. Ces tags représentent des versions précises. Vous pouvez utiliser ces tags lorsque vous faites un FROM.

Exemple :

```
FROM apache:2.4
```

11.2.4.5 Bonnes pratiques Dockerfile (https://docs.docker.com/develop/develop-images/dockerfile_best-practices/)

11.2.5 Couches

A noter que Docker utilise un système de couche (layer). Chaque ligne du Dockerfile correspond à une couche et possède son empreinte (hash).

Ainsi nul besoin pour Docker de télécharger (pull) à chaque build la même image sur le DockerHub. Il va réutiliser les couches qui n'ont pas changé.

(<https://docs.docker.com/build/guide/layers/>)

11.3 docker compose

Docker est prévu pour faire fonctionner des architectures microservices. Dans ce type d'architecture, on crée un conteneur par service.

Ceci a pour but de pouvoir multiplier le nombre de conteneurs suivant la demande. Imaginez un site Web PHP MVC, nous créerons un Dockerfile avec le code et le serveur Web et un autre Dockerfile avec la base de

données. Le Dockerfile avec le code et le serveur Web peut facilement être multiplié en plusieurs conteneurs pour augmenter les performances. C'est ce que l'on appelle de la mise à échelle horizontale (horizontal scaling).

Ceci est bien beau mais :

- **Comment gère-t-on la communication entre les différents conteneurs ? Le conteneur application devra vraisemblablement communiquer avec le conteneur db.**
- **Comment rend-t-on des données persistantes ? Un conteneur est stateless mais les données d'un conteneur db devront être persistées.**

C'est ici qu'intervient docker compose qui est un script python permettant de simplifier la création de ces architectures microservices. Avec un seul fichier YAML, on pourra créer notre architecture. docker compose ajoutera tous les services (conteneurs) dans un même réseau ce qui permettra une communication entre les conteneurs.

docker compose permet également facilement la création de volumes. Ceci permet d'effectuer la persistance de données en dehors d'un conteneur. Ainsi lorsqu'un conteneur est détruit, les données ne sont pas perdues. Les données (DB, code source ...) sont généralement stockées sur la machine hôte et un volume est créé dans le conteneur pour qu'il puisse utiliser ces données. Il s'agit tout simplement d'un point de montage entre le conteneur et la machine hôte.

11.3.1 Installation de docker-compose

docker compose a normalement déjà été installé avec docker.

Vérifier l'installation :

```
docker compose version
```

11.3.2 Utilisation de docker-compose

L'utilisation de docker compose se fait en 3 étapes :

1. Création éventuelle des Dockerfile pour les différents conteneurs si on n'emploie pas directement une image du DockerHub
2. Créer un fichier YAML nommé docker-compose.yml
3. Lancer «docker-compose up»

11.3.3 docker-compose.yml

Les fichiers YAML sont de simples fichiers textes dont le but est de fournir une description structurée et lisible. L'arborescence est créée par une indentation d'espaces.

Voici un exemple du déploiement d'un site WordPress :

```
version: '3'
#liste des services
services:
  db:
    image: mysql:5.7 # utilisation d'une image du DockerHub
    volumes:
      - db_data:/var/lib/mysql #persistance données MySQL
    restart: always # politique redémarrage conteneur
    environment:
      MYSQL_ROOT_PASSWORD: somewordpress
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress
```

```
wordpress:
  depends_on:
    - db
  image: wordpress:latest
  ports:
    - "8000:80"
  restart: always
  environment:
    WORDPRESS_DB_HOST: db:3306
    WORDPRESS_DB_USER: wordpress
    WORDPRESS_DB_PASSWORD: wordpress
    WORDPRESS_DB_NAME: wordpress

volumes:
  db_data: {}
```

11.3.4 Directives intéressantes

La directive «build» permet de construire une image Docker à partir d'un Dockerfile.

La directive «ports» permet d'effectuer la redirection de ports (port forwarding) entre la machine hôte et le conteneur. Cette directive attend un tableau, c'est pourquoi chaque élément commence par un «-».

La directive volumes permet de lier un répertoire de la machine hôte au conteneur. Les volumes ne sont pas détruits par défaut lors de la destruction du conteneur. C'est donc via ce biais que les données persistantes des conteneurs sont gérées. Deux manières sont utilisées pour la création des volumes :

1. /machinehote/path :/container/path -> lien entre l'arborescence de la machine hôte et le conteneur
2. mysql-data:/container/path -> création d'un volume logique mysql-data lié à l'arborescence du conteneur. Il faut dans ce cas préciser ce volume dans la section volumes du docker-compose.yml (voir exemple ci-dessous).

Exemple :

```
version : '3'
services :
  myapp :
    build :
      context : ./dirWithDockerFile/
    ports:
      - 8000:80
  mysql :
    image: mariadb:10.4
    volumes:
      - mysql-data:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: rootpass
      MYSQL_DATABASE: mydb
volumes :
  mysql-data: {}
```

11.3.5 Réseau

Les noms des services(conteneurs) deviennent des noms réseaux. Imaginez que mon application ci-dessus doit accéder dans son code à la base de données MySQL. Cette base donnée mysql est située dans un autre conteneur nommé mysql, je peux y accéder au niveau réseau en donnant le nom mysql.

11.3.6 Commandes intéressantes

Créer et lancer tous les services présents dans le docker-compose.yml :

```
docker compose up -d
```

Supprimer tous les services (conteneurs) présents dans le docker-compose.yml :

```
docker compose down
```

Supprimer tous les services (conteneurs) présents dans le docker-compose.yml ainsi que les volumes (données persistantes) :

```
docker-compose down -v
```

Vérifier la syntaxe du fichier yaml :

```
docker compose config
```

Voir les conteneurs et leur état :

```
docker compose ps -a
```

Voir les logs (utile pour comprendre une erreur de déploiement) :

```
docker compose logs
```

Pour supprimer les images, conteneurs, les réseaux et volumes : *Ceci peut être utile pour effectuer un nettoyage complet*

```
docker compose down --rmi all --volumes
```

11.4 The Twelve Factor App

La conception de Docker s'est inspirée de la méthodologie des 12 facteurs, une méthodologie pour créer des applications SaaS [**Modèles de service**]. Voici un lien vers cette méthodologie [<https://12factor.net/fr/>]. Voyons comment Docker applique celle-ci.

Retenez cinq facteurs avec leurs exemples Docker

- 1) Code de base : La notion d'image en Docker permet clairement de déployer plusieurs fois un même code ou des versions différentes de ce même code.
- 2) Dépendances : Le Dockerfile rend explicite les dépendances
- 3) Configuration : Il est possible de passer des variables d'environnement à un conteneur. Cependant un orchestrateur(k8s) pourra le faire de manière plus élégante.
- 4) Services externes : Les ressources peuvent être découvertes via leur nom. Cependant un orchestrateur(k8s) pourra le faire de manière plus élégante.
- 5) Build, Release, Run : docker build, docker run
- 6) Processus : Docker est principalement stateless. Les volumes existent mais il sera plus élégant de gérer cela avec un orchestrateur. Il n'est pas rassurant de savoir ces volumes stockés sur une machine hôte qui peut faillir.
- 7) Association de ports : docker run -p 8080 :80

- 8) Concurrence : Le côté stateless des conteneurs permet facilement de créer plusieurs conteneurs pour faire face à la montée en charge.
- 9) Jetable : Avec Docker, on crée, on déploie et supprime facilement une application.
- 10) Parité Dev / Prod : la même image peut être lancée en prod ou en dev.
- 11) Logs : Par défaut tous les logs Docker sont envoyés vers stdout (d'où le fait que Docker oblige les applications à tourner en avant-plan).
- 12) Processus d'administration : Il est très facile de se connecter et d'interagir avec un conteneur. `docker exec -it`

11.5 Kubernetes (K8s)

11.5.1 Introduction

Vu le succès des conteneurs, la majorité des applications sont maintenant déployées de la sorte. Il devient donc nécessaire d'avoir un outil pour gérer ces conteneurs. Kubernetes répond à cette demande. **Kubernetes est un orchestrateur de conteneurs.**

11.5.2 Utilité

Kubernetes permet d'appréhender les défis liés à la gestion des conteneurs notamment :

- L'équilibrage de charge entre plusieurs conteneurs (load balancing)
- La gestion des différents stockage pour les conteneurs (volumes). Ceux-ci peuvent être locaux, dans le Cloud, ...
- La gestion et l'allocation des ressources au conteneurs de manière dynamique
- La gestion de l'état de santé des conteneurs
- La gestion des informations de configurations et des secrets (clé SSH, password, ..)

11.5.3 Vue générale et concepts k8s

Kubernetes est composé d'un nœud maître permettant la gestion du cluster k8s. Un cluster est un ensemble de machines physiques ou virtuelles. Chaque machine du cluster est appelée un "worker node" et contiennent une solution de déploiement de conteneurs (Docker par ex.).

Chaque "worker node" héberge un pod. Un pod est l'unité de déploiement d'une application dans k8s. Il s'agit d'un ou plusieurs conteneurs. Dans la pratique, le plus souvent un pod == un conteneur/un processus

Un label est un couple "clé :valeur" que l'on peut attacher à un objet notamment un pod. Les selectors permettent de sélectionner un objet k8s suivant son label

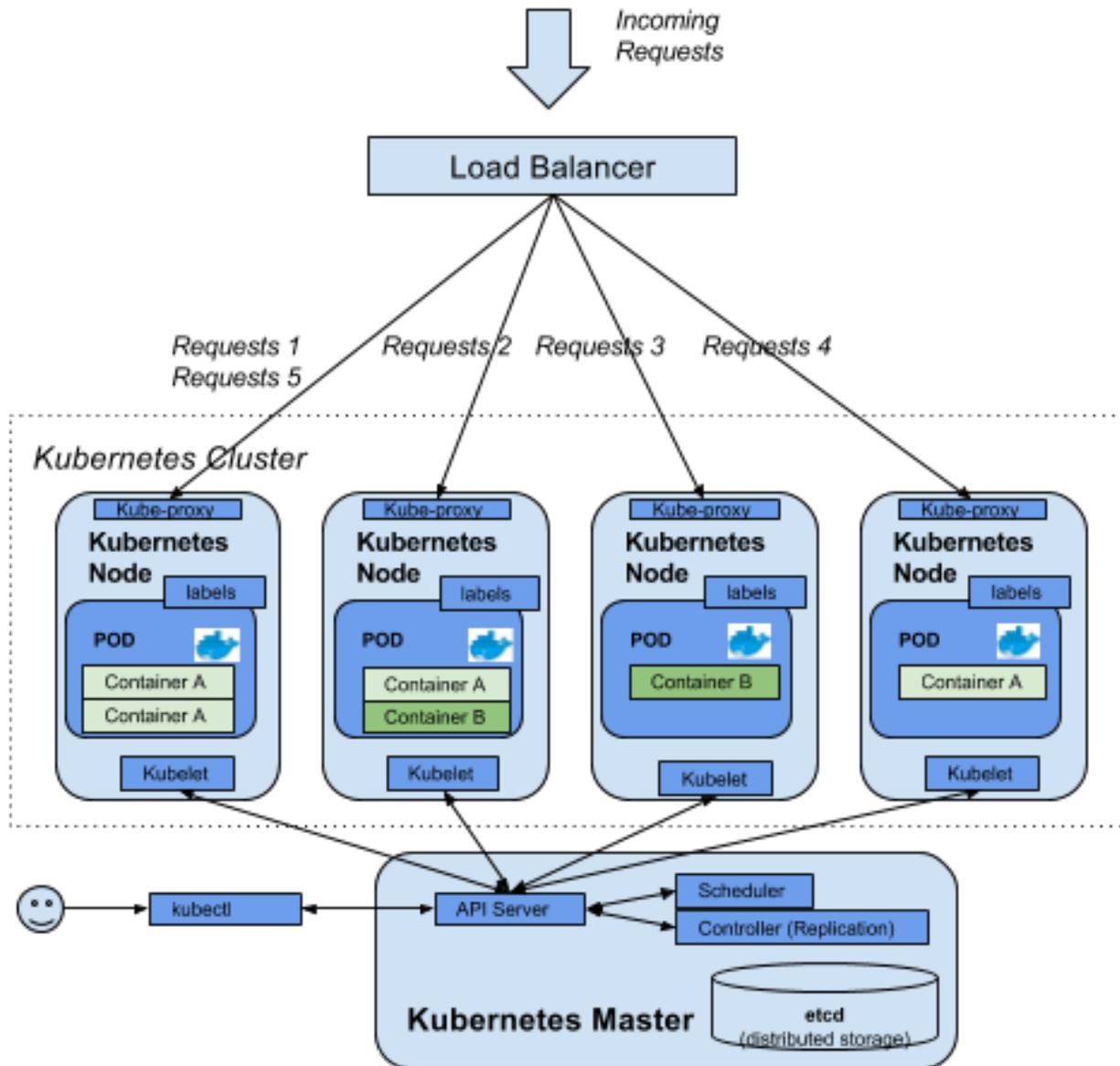


Image issue de : <https://hub.alfresco.com/t5/alfresco-process-services/activiti-7-deep-dive-series-deploying-and-running-a-business/ba-p/288347>

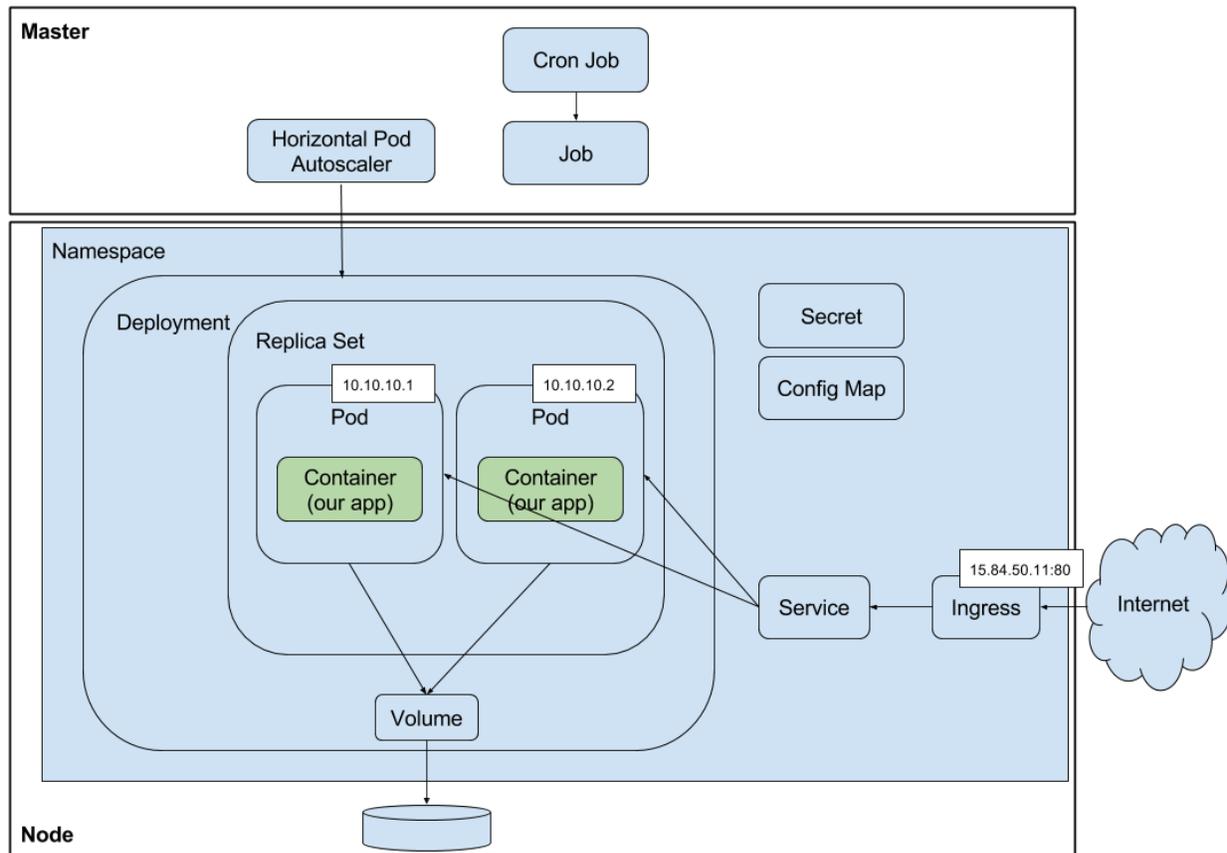


Image issue de : <https://hub.alfresco.com/t5/alfresco-process-services/activiti-7-deep-dive-series-deploying-and-running-a-business/ba-p/288347>

Un fichier `deployment.yml` permet de définir comment un Pod sera déployé. Ce fichier contient une instruction “`replica`” qui indiquera à K8s combien d’instances de ce Pod il devra lancer/maintenir.

Un fichier `services.yml` permet de définir et gérer le réseau à l’intérieur du cluster. Vu que les pods peuvent être créés, détruits et recréés il est nécessaire qu’un service puisse cibler les pods encore actifs à un moment t.

Ingress est un load balancer qui permet de faire communiquer le monde extérieur avec le cluster k8s. Nginx peut fournir ce service.

Les Persistent Volume Claim(PVC) permettent de persister des données. Cela va beaucoup plus loin que l’utilisation de simples volumes (PV). Il s’agit d’une demande d’espace adressée à k8s. Celui-ci attribuera alors au pod un stockage suivant sa demande. A noter aussi que l’intérêt de k8s par rapport au stockage est de pouvoir utiliser des stockages variés (Amazon, Azure, Local, ...).

12 DevOps (Ansible notamment)

12.1 Objectifs du chapitre

- Consolider les notions DevOps
- Déployer des configurations via Ansible

12.2 Introduction

Dans un développement de logiciel actuel, on ne livre plus seulement un exécutable au client. La livraison du logiciel inclut de plus en plus le déploiement de l'application et la mise à jour en continu. Ces 2 derniers points impliquent les administrateurs système.

Dans cette approche, il est donc nécessaire d'unifier 2 entités souvent séparées en entreprise : les développeurs et les administrateurs système aussi appelés opérationnels.

SysAdmin vs Developer

Il est important de savoir que les développeurs et les administrateurs système ont des objectifs différents et que par conséquent leurs réponses par défaut face à une demande seront différentes. Les développeurs vivent des besoins de leurs clients et sont donc toujours enclins à développer un nouveau logiciel. Ils répondent souvent par l'affirmative à une demande. Les administrateurs système ont pour objectif de préserver l'environnement informatique. À toute demande, ils vont donc privilégier une réponse négative si vous ne leur donnez pas un minimum de garantie.

DeVops est donc une approche visant à faire travailler ensemble les administrateurs système et les développeurs. DevOps mise sur la confiance entre les acteurs, incite les acteurs à expérimenter et s'inspire de la philosophie japonaise Kai Zen bien connue dans le monde l'industrie. *Il s'agit en fait d'un état d'esprit et d'un processus d'amélioration continue.*

Le déploiement d'une application faisant maintenant partie du produit ainsi que les mises à jour en continu, il est donc nécessaire d'avoir une infrastructure avec le concours des opérationnels pour mener à bien ce nouveau mode de développement logiciel.

DevOps en image :



Image issue de (https://res.cloudinary.com/practicaldev/image/fetch/s---dbI8WY9--/c_limit%2Cf_auto%2Cfl_progressive%2Cq_auto%2Cw_880/http://aisaac.io/content/images/2018/11/DevOps.jpg)

Schéma à connaître

On voit sur cette image le cycle DevOps depuis le cahier des charges(Plan) jusqu'à la surveillance de l'application en production. Ce cycle se répète à l'infini.

12.3 Pipeline de développement

L'approche DevOps mise également sur l'automatisation via des outils. On parlera souvent de pipeline de développement. Il s'agit d'une chaîne de production logicielle la plus automatisée possible jusqu'au client.

Sans surprise, les pipelines utiliseront les éléments vus dans les chapitres précédents surtout les conteneurs, les orchestrateurs, le Cloud et les outils de configurations.

Exemple pipeline :

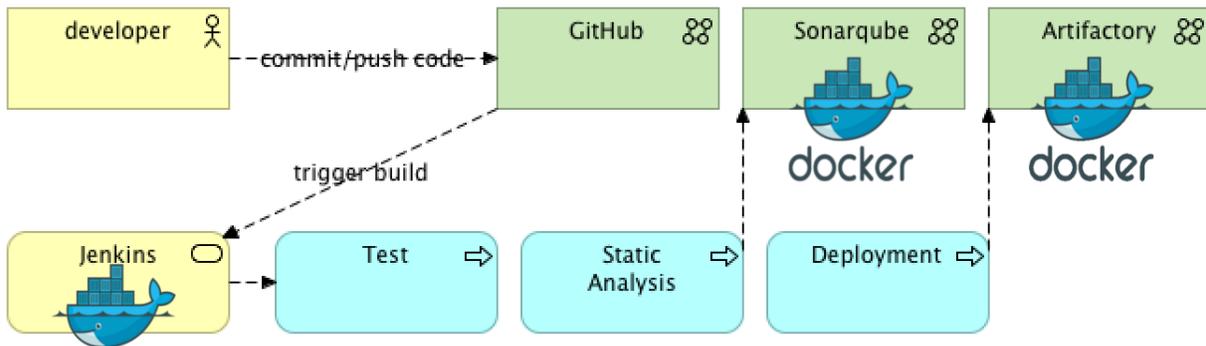


Image issue de : (<https://dzone.com/articles/dockerizing-jenkins-part-2-deployment-with-maven-a>)

On peut remarquer que Docker reste un composant essentiel dans ces diagrammes. Il est donc nécessaire qu'il soit connu par les développeurs et les opérationnels.

12.3.1 DevOps et les développeurs

DevOps va privilégier une architecture microservice, car celle-ci permet un meilleur contrôle et une montée en charge facilitée. Dans ce cadre, les développeurs doivent faciliter un déploiement impliquant une montée en charge (cfr Docker par ex.). Ils doivent être également conscients que leur code va certainement appeler d'autres services. Il faudra donc prévoir un code résilient par rapport à cet état des choses (quid si un service dont je dépends ne répond pas?). Il existe des patterns spécifiques aux architectures microservices (Ex : retry pattern).

Le dépôt de code sera également structuré de manière à faciliter l'approche DevOps qui demande au minimum un environnement de développement, de tests et de production. Il existe plusieurs techniques de structuration du dépôt visant à atteindre cet objectif.

12.3.2 DevOps et les sysadmins

Le rôle des sysadmins dans le DeVops sera notamment de mettre à disposition des développeurs de l'IaaS ou du PaaS (voir Déploiement Cloud ci-dessous). Ils seront souvent également en charge de la partie CD (Continuous Deployment)dans la chaine DeVops.Les outils utilisés par les sysadmins dans ce cadre sont des outils de gestion de configuration (Chef, Puppet, Ansible, ..), des outils de gestion et utilisation des conteneurs (Docker, K8s). Nous verrons Ansible en labos car il peut également être utile à un développeur.

12.4 Ansible

12.4.1 Introduction

Pour les administrateurs systèmes, il est nécessaire de disposer d'outils pour automatiser les tâches d'installations et de configurations. En effet, les administrateurs systèmes doivent s'occuper d'un parc de machines et il est donc insensé de configurer toutes ses machines à la main.

Pour ce faire, il existe différents outils permettant d'automatiser les installations et configurations. Nous pouvons citer Puppet, Chef et Ansible. Nous verrons Ansible en labos car il peut également se révéler utile pour un développeur.

12.4.2 Ansible

Ansible a défini son propre vocabulaire. On parle de *contrôleur* qui est est la machine qui exécute Ansible en tant que tel et de *cibles* qui sont les machines qu'Ansible configure. Les cibles sont définies dans l'inventaire d'Ansible, simple fichier reprenant des noms de machines ou IP de machines. *A noter que le contrôleur et la cible peuvent être la même machine.* Dans ce cas, cela signifie que nous automatisons alors simplement

un ensemble de tâches pour une seule machine. Nous procéderons de cette manière durant les labos. *Vous verrez que l'on précisera toujours dans notre fichier Ansible localhost comme cible !*

Ansible se base sur SSH pour pouvoir se connecter et déployer des configurations sur les cibles. Ansible utilise des fichiers YAML. **Je vous conseille d'utiliser VS Code pour faire votre fichier YAML, vous aurez ainsi déjà une vérification syntaxique. Vous pouvez utiliser l'extension Remote SSH de VSCode pour éditer un fichier sur votre VM à travers SSH !**

Ansible parle de playbooks. Un playbook est simplement un fichier YAML décrivant un ensemble de tâches à effectuer sur un ensemble de cibles.

Ansible dispose de nombreux modules permettant d'effectuer des tâches. Voici les plus importants :

- apt : pour installer un package (Debian/Ubuntu)
- git : cloner un repo
- command : exécuter une command shell
- template : copier un fichier de configuration modèle
- lineinfile : modifier des lignes dans un fichier

Ces modules sont disponibles dans la liste des modules builtin : (<https://docs.ansible.com/ansible/latest/collections/ansible/builtin/index.html#plugins-in-ansible-builtin>)

Voici la liste complète des modules et plugins : (https://docs.ansible.com/ansible/latest/collections/index_module.html) **Utilisez la recherche par catégorie de modules ou par nom de module**

Ansible dispose d'attributs spéciaux. Ceux peuvent être considéré comme des éléments du langage Ansible.

- gather_facts : collecter des infos sur la cible (IP, OS information, ...). Ces informations sont ensuite disponibles en tant que variables.
- register : stocker le résultat d'un module dans une variable
- vars : permet de définir des variables
- loop/item : répéter une opération
- become : devenir root ou utilisateur privilégié
- notify/handlers : appeler une tâche
- async/poll : tâche asynchrone (en arrière plan)

Exemples :

```
---
# gather facts, notify/handlers
- hosts: localhost
  gather_facts: true
  tasks:
  - name: Install Nginx
    package:
      name: nginx
      state: latest
    when: ansible_distribution == 'CentOS'
    notify: restart nginx

  handlers:
  - name: restart nginx
    service:
      name: nginx
      state: restarted

---
# loop ands item
- hosts: localhost
```

```
- name: Create new users
  user:
    name: '{{ item }}'
    state: present
  loop:
    - john
    - mike
    - andrew
```

```
---
# become, vars , loop, gather facts
# gather facts true by default
- name: Install software
  become: yes
  hosts: all
  vars:
    packages:
      - name: neofetch
        required: True

      - name: cpu-checker
        required: True

      - name: screenfetch
        required: False
  tasks:
    - name: Install "{{ item.name }}" on Ubuntu
      apt:
        name: "{{ item.name }}"
        state: present

      when:
        - item.required == True
        - ansible_facts['distribution'] == "Ubuntu"

    loop: "{{ packages }}"
```

12.4.3 Installation Ansible

```
apt-get install ansible
```

12.4.4 Utilisation des playbooks

```
---
# hosts : liste des hosts où le playbook s'appliquera
- hosts: Web
  tasks:
# name : description de la tâche
  - name: "Install Apache Web Server"
# module ansible appelé (voir ansible modules)
  apt:
    name: apache2
```

```
state: latest
```

Voici comment lancer le playbook :

```
ansible-playbook playbook.yml
```

Vous devez être root pour lancer cette commande. D'autres options et subtilités existent dans Ansible mais nous nous limiterons pour ce cours à des playbook simples et lancés en tant que root.

Pour créer vos tâches, il est utile de consultez la liste des [Modules Ansible disponibles](#)

Il est possible de vérifier la syntaxe de votre fichier yml avec “ansible-playbook playbook.yml - -syntax-check”.

Il est possible également de lancer ansible-playbook en mode verbeux avec “ansible-playbook playbook.yml -v”. Vous pouvez mettre jusqu'à 4 “v” pour plus de détails (-vvvv).

12.4.5 Rôle Ansible

Ansible permet de séparer un playbook dans plusieurs fichiers. On appelle cela un **rôle** Ansible. Un rôle Ansible est un répertoire contenant l'arborescence suivante :

- defaults : valeurs par défaut des variables. celles-ci peuvent être surchargées par définition explicite ou vars
- files : fichiers statiques à déployer
- handlers : tâches pouvant être appelées par notification
- tasks : tâches à effectuer
- templates : modèles de fichier de configuration
- vars : variables
- README.md : description générale du rôle

L'avantage de ces rôles est la réutilisation.

```
# Créer un rôle
ansible-galaxy init <roledir>
```

Pour utiliser un rôle, il suffit de l'appeler dans un playbook.

```
---
- name: "test role"
  hosts: localhost
  gather_facts: false
  roles:
    - monrole
```

12.4.6 Variables Ansible

L'utilisation de variables dans un playbook ou un rôle permet de faciliter la réutilisation.

Dans un playbook, on peut définir des variables directement après la directive hosts de la manière suivante :

```
---
- name: "test"
  hosts: localhost
  vars:
    myvar: myvarvalue
  tasks:
    name: "show myvar value"
    ansible.builtin.command: echo {{ myvar }}
```

Dans un rôle ansible, on peut définir des variables dans les dossiers suivants :

- default : pour définir une valeur à une variable si elle n'est pas précisée lors de l'appel au rôle
- vars : pour imposer une valeur à une variable (moins utile)

Lors de l'appel à un rôle ansible, on précise les valeurs des variables de la même manière que pour les playbooks.

12.4.7 Collection Ansible

Ansible propose également un mécanisme d'extension et de partage des rôles sous un format standardisé. Ceci s'appelle une collection Ansible. La commande `ansible-galaxy` permet de gérer ces collections.

Ceci ne fera pas l'objet de ce cours.

13 Cloud (Terraform notamment)

13.1 Objectifs du chapitre

- Connaître les modèles de service Cloud
- Architecture Multi-tenant et Cloud
- Déployer des services PaaS avec Terraform

13.2 Introduction Cloud

A l'origine, le terme Cloud (nuage) était la représentation d'Internet dans des schémas réseaux. Actuellement le Cloud désigne une infrastructure où tout est virtualisé et abstrait. On ne sait plus exactement où sont stockées les données, ni comment par.

Le Cloud est une abstraction qui s'est déroulée en plusieurs étapes du réseau, du Web, de l'infrastructure. Le cloud est devenu très populaire car il permet d'accélérer la transition numérique pour de nombreuses entreprises. Par ces caractéristiques (ci-dessous), le Cloud permet aux entreprises d'être plus agiles et compétitives.

13.3 Caractéristiques essentielles du Cloud

- Accès aux services à la demande
- Accès réseau large bande
- Réservoir de ressources
- Elasticité (redimensionnement rapide) - Scaling
- Facturation à l'usage - Réduction des coûts (enfin si bien étudié)
- Automatisation (éviter les tâches répétitives, ...)
- Résilience (reprise rapide des activités en cas de panne, catastrophe)
- Sécurité (les grands acteurs du Cloud ont une équipe sécurité)

13.4 Modèles de service

13.4.1 Infrastructure as a Service (IaaS)

IaaS : Déploiement d'une infrastructure via les outils du fournisseur Cloud. L'utilisateur peut paramétrer le réseau, les serveurs, Ex : Amazon EC2, Azure VM, ...

Terraform est certainement l'outil le plus utilisé pour déployer de l'IaaS. Il permet de créer des VM et d'autres éléments d'infrastructure sur AWS, Azure, Cet outil est décrit dans la section suivante.

13.4.2 Platform as a Service (PaaS)

PaaS : Déploiement d'une plateforme via les outils du fournisseur Cloud. L'utilisateur peut paramétrer la plateforme mais celui ci n'a aucun accès, vue sur l'infrastructure. Ceci est sans le doute le modèle de service Cloud le plus employé par les développeurs. Ex : Heroku, AWS Elastic Beanstalk, Cloud Foundry, ...

13.4.3 Software as a Service (SaaS)

SaaS : Accès à une application via Internet. Aucun accès, vue sur l'infrastructure, paramétrage par l'utilisateur limité. Ex : Gmail, Office365, ...

Comparaison en schéma :

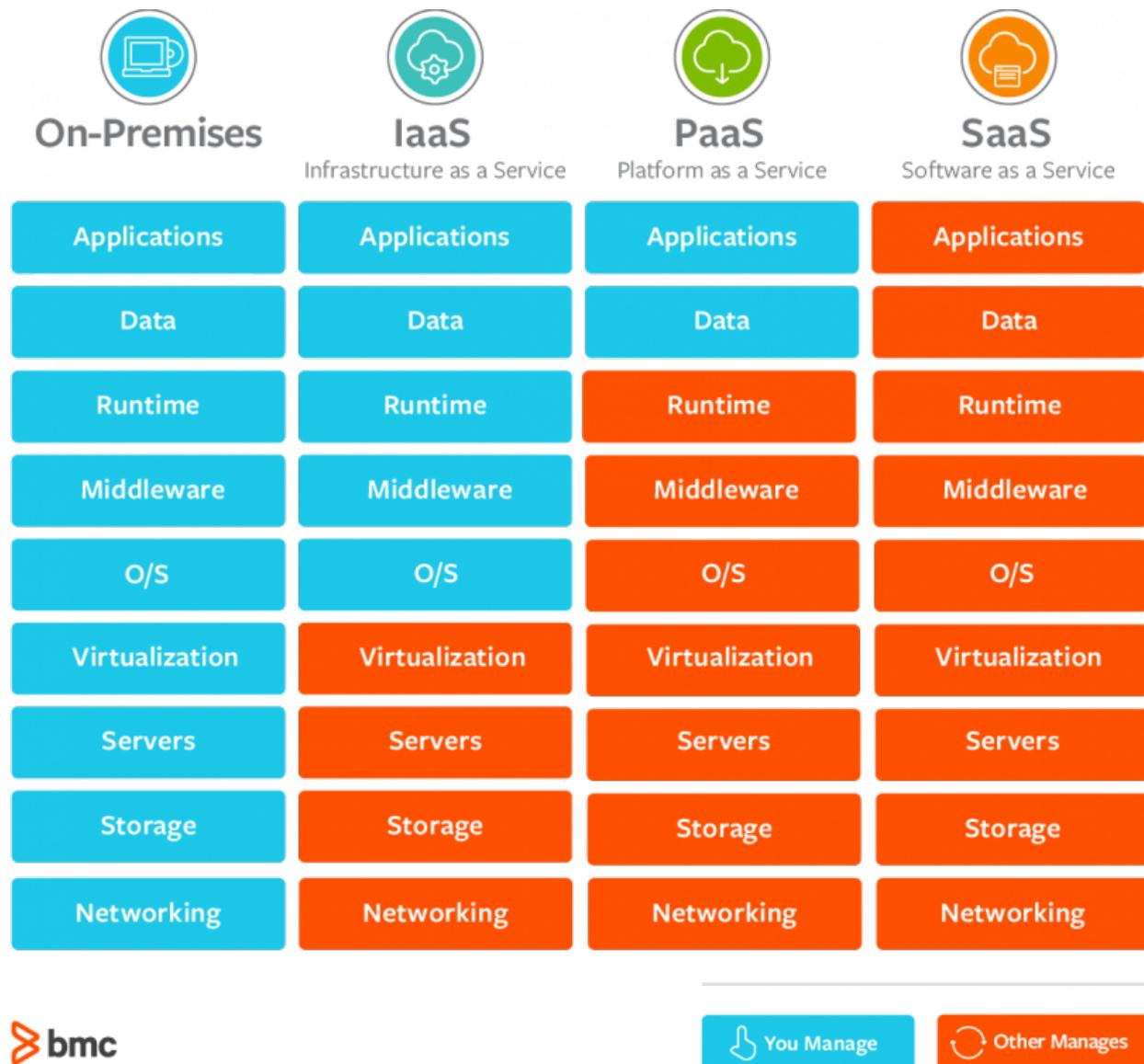


Image issue de : (<https://cloudmeb.com/resource/saas-vs-paas-vs-iaas-quelle-est-la-difference-et-comment-choisir/?lang=fr>)

13.5 Architecture Multi-tenant et Cloud

13.5.1 Introduction

Qu'est qu'une architecture multi-tenant ?

Dans une architecture multi-tenant, une même instance d'une application logicielle est utilisée par plusieurs clients, ces derniers étant des « tenants ».

Le modèle multi-tenant peut s'avérer économique, étant donné que les coûts liés au développement et à la maintenance des logiciels sont partagés.

Par opposition, une architecture single-tenant est une architecture dans laquelle chaque client possède sa propre instance de logiciel et peut avoir accès au code. Dans une architecture multitenant, le fournisseur n'exécute les mises à jour qu'une seule fois, alors que dans une architecture single-tenant, il doit gérer plusieurs instances du logiciel pour appliquer les mises à jour.

3 approches sont possibles :

Approches	Description	Avantages	Inconvénients
Multi-tenant	Les clients se partagent la même application et la même DB	Coût de location et maintenance faible, mise à l'échelle aisée, déploiement facile des mises à jour	Effets de bords liés aux colocataires (sécurité, performance)
Multi-DB	Les clients se partagent la même application mais ont chacun leur propre base de données	Isolation totale des données (sécurité)	Coût location et maintenance plus élevé. Mise à l'échelle compliquée
Multi-instances / Single tenant	Chaque client a sa propre instance d'application et sa propre DB	Isolation totale, mise à l'échelle aisée, personnalisations possibles	Coût très élevé

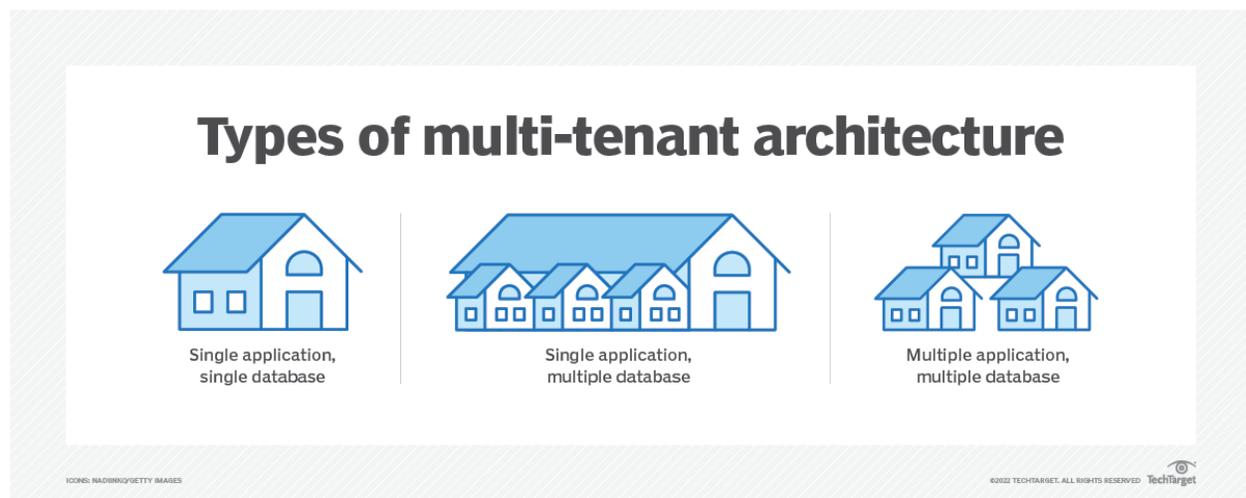


Image issue de : <https://www.techtargget.com/whatis/definition/multi-tenancy>

Les modèles de service SaaS présentes dans le Cloud propose souvent les approches multi-tenant et single-tenant. Ainsi suivant ses priorités et son budget, on peut effectuer un choix.

13.6 Terraform

13.6.1 Introduction

Terraform est un outil permettant de déployer des ressources dans le Cloud. En clair, il permet d'automatiser la création de ressources (VM, image docker, ...) en local ou dans le Cloud. Il dispose de nombreux connecteurs (providers) permettant de créer des ressources aussi bien dans AWS (Amazon), Azure, en local, Ce dernier point le rend particulièrement intéressant au sein des entreprises.

13.6.2 Installation Terraform

Installation des prérequis :

```
apt-get update && apt-get install -y gnupg software-properties-common wget curl
```

Récupération de la clé GPG :

```
wget -O- https://apt.releases.hashicorp.com/gpg | gpg --dearmor -o  
→ /usr/share/keyrings/hashicorp-archive-keyring.gpg
```

Ajout de repository APT pour Terraform :

```
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg]  
→ https://apt.releases.hashicorp.com $(lsb_release -cs) main" | tee  
→ /etc/apt/sources.list.d/hashicorp.list
```

Installation de Terraform :

```
apt-get update && apt-get install terraform
```

13.6.3 Utilisation Terraform

Terraform utilise un fichier YAML nommé main.tf. Dans ce fichier est défini un ou plusieurs providers (endroit où se feront les créations) et comment y accéder.

Pour utiliser Azure comme provider, il faut installer la console Azure CLI :

```
curl -sL https://aka.ms/InstallAzureCLIDeb | bash
```

```
az login --use-device-code
```

Ceci vous donnera en retour un affichage JSON où vous verrez la jonction avec votre abonnement Azure.

Ensuite il faut créer le fichier main.tf qui contiendra les instructions/modules terraform qui effectueront les actions dans le Cloud. Pour se faire il faut se référer à la documentation :

13.6.3.1 Documentation Terraform Dans le cadre de ce cours, nous utiliserons uniquement Terraform pour déployer des App Service dans le Cloud Azure. Voici 2 liens vers la documentation la plus utile :

- https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/linux_web_app
- <https://github.com/hashicorp/terraform-provider-azurerm/tree/main/examples/app-service>

Une fois le fichier main.tf réalisé, l'utilisation de Terraform est simple.

Installation des providers requis :

```
terraform init
```

Voir ce que Terraform prévoit de réaliser :

```
terraform plan
```

Lancer la création des ressources Terraform :

```
terraform apply
```

Détruire les ressources Terraform :

```
terraform destroy
```