



Virtualisation et Cloud Computing

PR. KAMAL EL GUEMMAT

Prérequis et objectifs

Prérequis

- M11 : Mathématiques appliquées 1 en S1
- M13 : Techniques de Programmation en S1
- M15 : Technologie des ordinateurs et réseaux en S1

Objectifs

Ce module a pour objectif de maîtriser les concepts de base de la virtualisation et du cloud computing et d'approfondir les compétences techniques en réseaux informatiques. A la fin de ce module, l'élève ingénieur doit être capable de :

- Etudier les domaines de virtualisation (serveurs/desktops, réseaux, applications, stockage, données).
- Mettre en œuvre la virtualisation des architectures en utilisant les différentes méthodes.
- Créer des machines virtuelles.
- Mettre en cluster des machines virtuelles.
- Maîtriser le principe des Data center.
- Déployer et administrer des serveurs virtuels à distance.
- Mettre en œuvre les permissions, la surveillance, la haute disponibilité et de l'optimisation en utilisation la virtualisation.
- Créer et administrer des conteneurs virtuels d'applications.
- Mettre en œuvre et déployer des applications dans des conteneurs.
- Évaluer les apports du Cloud pour l'entreprise.
- Identifier les principales offres Cloud du marché.
- Explorer et l'exploiter des environnements Cloud.
- Etudier les technologies de virtualisation utilisés dans les réseaux locaux et étendus.
- Maîtriser les protocoles et technologies avancés des niveaux 1,2,3.
- Concevoir des réseaux virtuels avancés.
- Découvrir les principales applications actuelles des réseaux ainsi que des évolutions technologiques.

PLAN DU COURS

CHAPITRE I : Datacenter

CHAPITRE II : Cloud Computing

CHAPITRE III : OCI

CHAPITRE IV : Cloud native

Datacenter

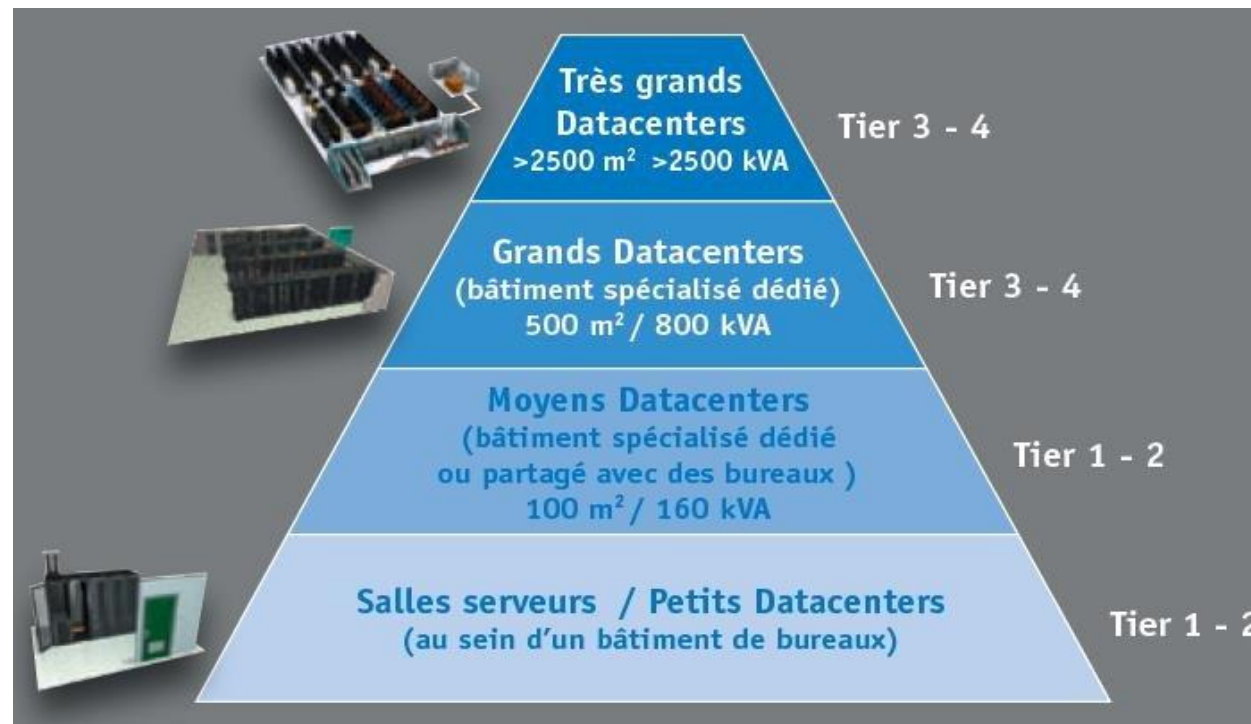
CHAPITRE I

Objectifs du chapitre I

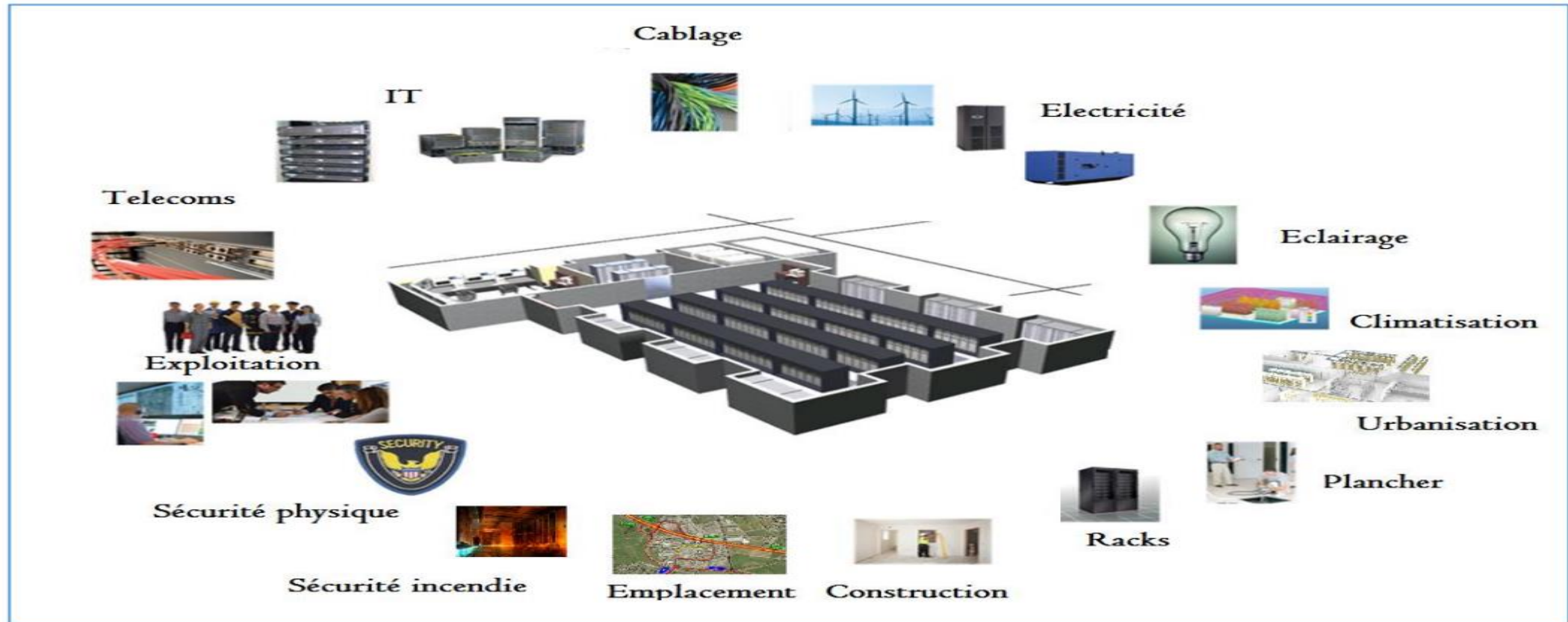
1. Généralités
2. Virtualisation du centre de données
3. Conteneurisation
4. Multi conteneurisation
5. Virtualisation des réseaux

I.1. Généralités

- Infrastructure Informatique.
- Organiser, traiter, stocker et entreposer de grandes quantités de données.



I.1. Généralités



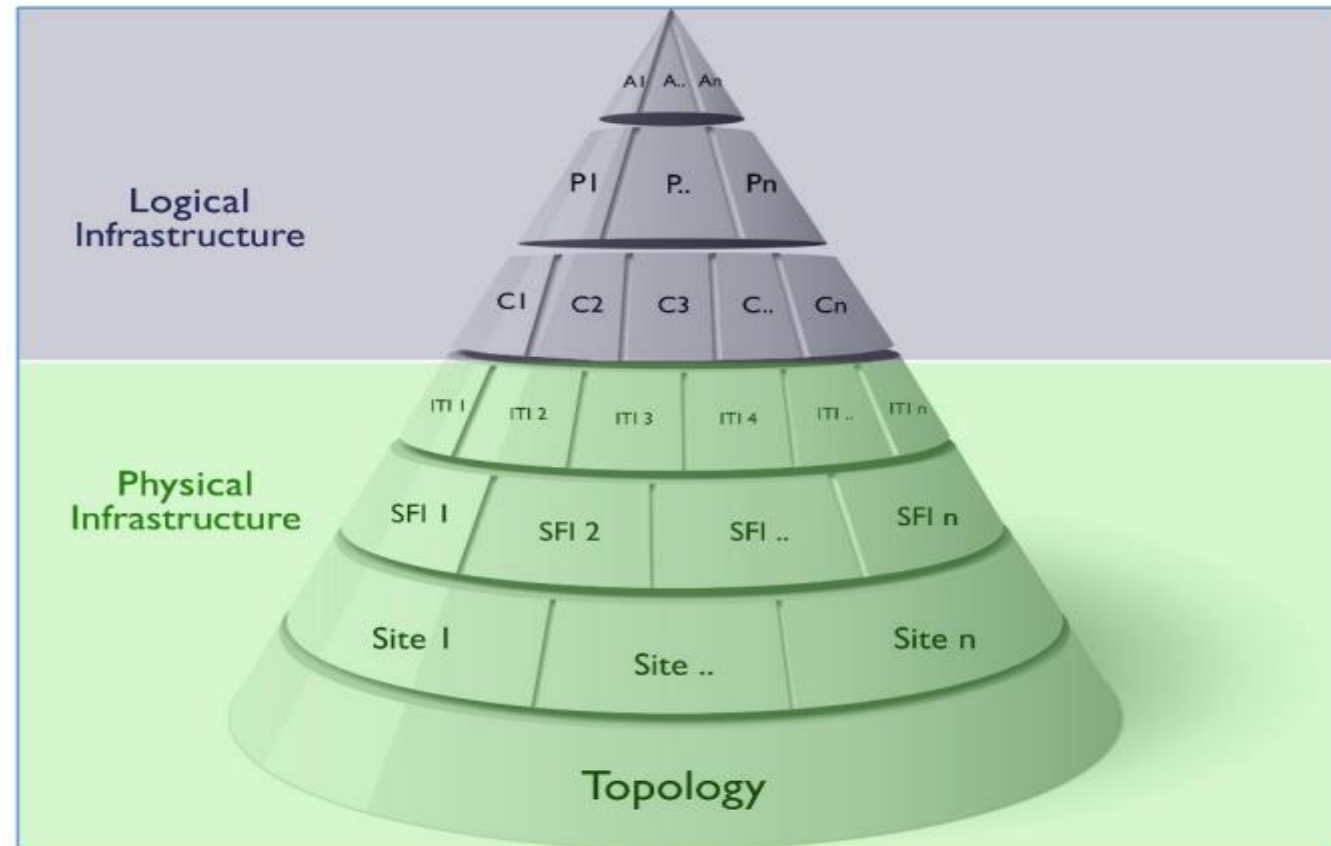
I.1. Généralités

- Les 1960 Datacenter 1.0; Les 1980 Datacenter 2.0; Les 2000 Datacenter 3.0.
 - Virtualisation.
 - Internalisation et externalisation.
 - Cloud.
 - IoT.
 - BigData.
 - Green IT.
 - Etc.
- Normes, Standards et bonnes pratiques appliqués au Datacenter.

I.1.Généralités

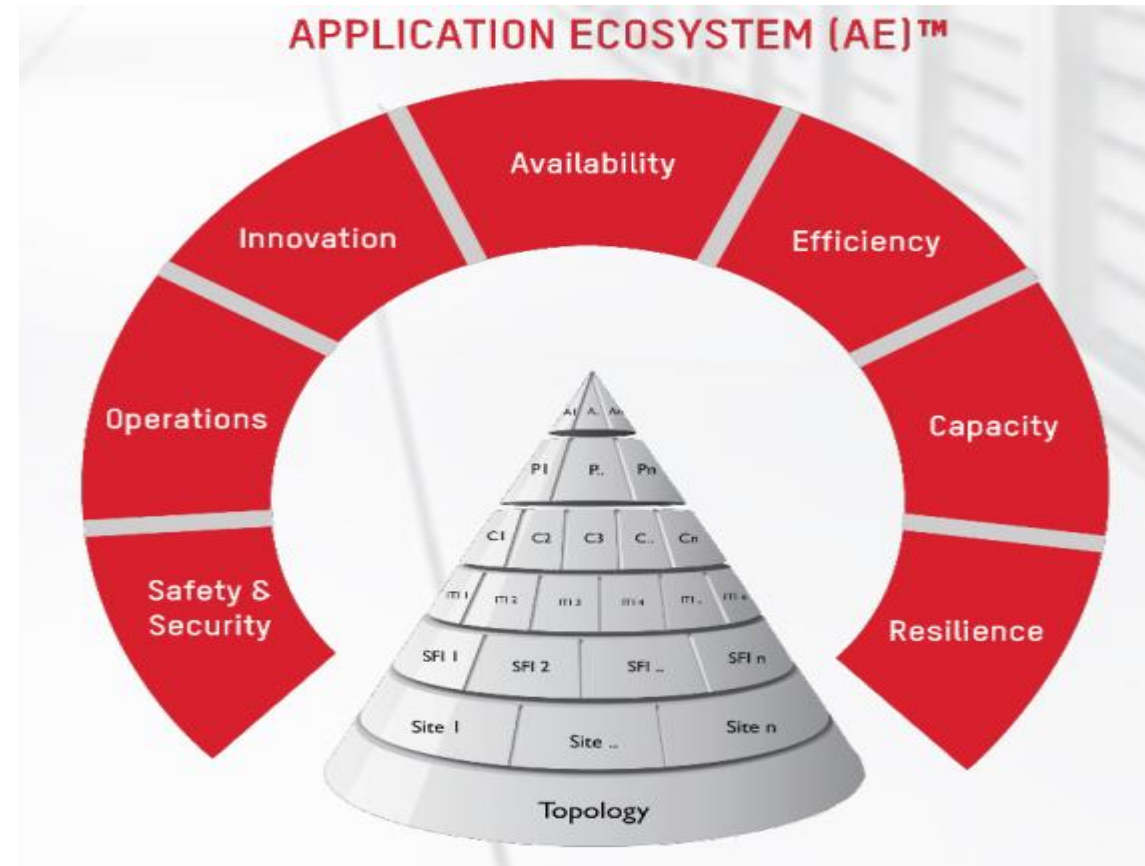
<div>Fire Protection</div> <div>Quality Assurance</div> <div>Information Security</div> <div>Business Continuity</div> <div>Management</div> <div>Disaster Recovery</div> <div>Seismic</div> <div>Service Provider Controls Auditing</div> <div>Miscellaneous</div>	<div><ul style="list-style-type: none">•NFPA-75•NFPA-2001</div> <div><ul style="list-style-type: none">•ISO 9001</div> <div><ul style="list-style-type: none">•ISO/IEC 27001</div> <div><ul style="list-style-type: none">•ISO 22301•BS 25999•ISO/IEC-24762</div> <div><ul style="list-style-type: none">•Telecordia GR-63-CORE</div> <div><ul style="list-style-type: none">•SAS 70(I & II)•SSAE 16•SOC 1 (Type 1& 2)•SOC 2•SOC 3</div> <div><ul style="list-style-type: none">• FISMA• NIST Cloud• HIPAA• PCI – DSS• SOX(Sarbanes-Oxley)• FIPS-140• CIBSE- Chartered Institution of Building Services Engineers• CENELEC-European Committee for Electrotechnical Standardization</div>	<div>Electrical</div> <div>Earthing/Grounding</div> <div>Environmental</div> <div>Networks/Cabli ng</div>	<div><ul style="list-style-type: none">•UNE-EN 50160•UNE-EN 61000-3-2•UNE-EN 61000-3-3•UNE-EN 55011•IEC-61643-1•EN-60730-2-7</div> <div><ul style="list-style-type: none">•MIE-REBT-039•EN-50310•IEC 60364•EMF :•EN 61000-4-8•EN 55022•EN 55011</div> <div><ul style="list-style-type: none">•IEC-61340-5-1•ISO 14001•Raised Floor, Cable Pathways•BS/EN 12825•UK-PSA PF2•TIA 569</div> <div><ul style="list-style-type: none">•ISO-11801•EN-50173•TIA-568A</div>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

I.1. Généralités




La pyramide du framework « Infinity Paradigm » par IDCA

I.1. Généralités



La pyramide du framework « Infinity Paradigm » par IDCA

I.1.Généralités

	Availability	Capacity	Operations	Safety & Security	Resilience	Efficiency	Innovation	Aggregate Grade
Application	Application Availability Controls	Application Capacity Controls	Application Operations Controls	Application Security Controls	Application Resilience Controls	Application Efficiency Controls	Application Innovation Controls	GX
Platform	Platform Availability Controls	Platform Capacity Controls	Platform Operations Controls	Platform Security Controls	Platform Resilience Controls	Platform Efficiency Controls	Platform Innovation Controls	GX
Compute	Compute Availability Controls	Compute Capacity Controls	Compute Operations Controls	Compute Security Controls	Compute Resilience Controls	Compute Efficiency Controls	Compute Innovation Controls	GX
Information Technology Infrastructure	ITI Availability Controls	ITI Capacity Controls	ITI Operations Controls	ITI Security Controls	ITI Resilience Controls	ITI Efficiency Controls	ITI Innovation Controls	GX
Site Facilities Infrastructure	SFI Availability Controls	SFI Capacity Controls	SFI Operations Controls	SFI Security Controls	SFI Resilience Controls	SFI Efficiency Controls	SFI Innovation Controls	GX
Site	Site Availability Controls	Site Capacity Controls	Site Operations Controls	Site Security Controls	Site Resilience Controls	Site Efficiency Controls	Site Innovation Controls	GX
Topology	Topology Availability Controls	Topology Capacity Controls	Topology Operations Controls	Topology Security Controls	Topology Resilience Controls	Topology Efficiency Controls	Topology Innovation Controls	GX
Efficacy Scores	AER™	CER™	OER™	SER™	RER™	EER™	IER™	ESR®

IDCA ESR

I.1. Généralités

- **Attentes fonctionnelles :**
 - Aligner le Datacenter et les applications au business.
 - Identifier le niveau de Criticité du Datacenter.
- **Besoins Organisationnels :**
 - Gestion optimisée des Opérations.
 - Gestion des ressources et définition d'une organisation Datacenter.
 - Conformité aux Normes et réglementations locales.
- **Besoins Techniques :**
 - Disponibilité.
 - Efficacité.
 - Sécurité.
 - Sureté.
 - Résilience.
 - Continuité d'activité.
 - Capacité.

I.1. Généralités

Phases de Développement d'un Datacenter

- **Phase 0** : Initiation et Cadrage du projet
- **Phase 1** : conception et design
- Etape 1 : Analyse et étude des données recueillies
- Etape 2 : Design conceptuel et détaillé des solutions techniques
- **Phase 2** : Elaboration du cahier des charges
- **Phase 3** : Réalisation projet

I.2. Domaines de virtualisation

- La virtualisation de serveurs/desktops.
- La virtualisation de réseaux.
- La virtualisation du stockage.
- La virtualisation d'applications.
- La virtualisation de données.
- Virtualisation de bout en bout.

Cisco book : Data Center Virtualization Fundamentals

I.2. Virtualisation de serveurs/desktops

- Exécuter plusieurs systèmes d'exploitation sur un seul serveur physique sous forme de machines virtuelles.

I.2. Virtualisation de réseaux

- Reproduire un réseau physique et ses différents composants : ports, interrupteurs, routeurs, firewalls, équilibreurs de charges, etc.

I.2. Virtualisation du stockage

- Dans une machine virtuelle, les données sont stockées sur un disque dur virtuel. Ce disque dur se présente sous forme de fichier dans le système de fichiers de l'hôte :
 - VHD chez Microsoft
 - VDI chez Oracle
 - VMDK chez VMware
 - OVF format ouvert
- Consiste à assembler la capacité de stockage de multiples appareils de stockage en réseau sous forme d'un seul appareil de stockage (virtuel).

I.2. Virtualisation d'applications

- Abstraction de la couche application du système d'exploitation.

I.2. Virtualisation de données

- Abstraction des détails techniques traditionnels des données et du Data Management : localisation, performance, format, etc.

I.2. Techniques de virtualisation serveurs/desktop

- **Hyperviseurs**: intermédiaire entre le système hôte et le système invité :
 - Créer des ressources virtuelles propres à chaque VM.
 - Répartir ces ressources.
- Permet l'abstraction la couche matérielle de la machine hôte.
- Les hyperviseurs utilisent plusieurs techniques pour fonctionner plus qu'un SE sur la même machine.
- Fournit l'isolation entre les systèmes invités.
- Type 1 (bare metal) : **grosses architectures.**

Xen, Hyper-V, ESXi, etc.

- Type 2 (host metal) : **une seule machine.**

Oracle VirtualBox, VMWare Workstation (Player et Pro) et VMware Fusion (pour Mac), etc.

I.2. Techniques de virtualisation serveurs/desktop

- **L'hyperviseur**, ou programme de contrôle, est un logiciel constitué d'un ensemble de modules.
 - **Le régulateur (dispatcher)** : il peut être considéré comme le module de contrôle de plus haut niveau de l'hyperviseur.
 - **L'allocateur** : son rôle est de déterminer quelle(s) ressource(s) doivent être allouées aux applications virtualisées.
 - **Des interpréteurs** : à chacune des instructions privilégiées (à l'exception de celles qui sont prises en charge par l'allocateur), on va associer une routine d'interprétation.

I.2. Techniques de virtualisation serveurs/desktop

- Machine virtuelle.
- Virtualisation d'OS.
- Hyperviseur complet.
- Para-virtualisation.
- La virtualisation assistée au niveau matériel.

I.2. Mise en cluster de machines virtuelles

- Hôtes.
- Clusters.
- Pools de ressources.

I.2. Hôtes

Un hôte représente l'ensemble des ressources de calcul et de mémoire d'un serveur physique. Par exemple, si le serveur physique dispose de quatre processeurs double cœur fonctionnant à 4 GHz chacun et de 32 Go de mémoire système, l'hôte disposera alors de 32 GHz de puissance de calcul et de 32 Go de mémoire disponibles pour exécuter des machines virtuelles affectées à ce serveur.

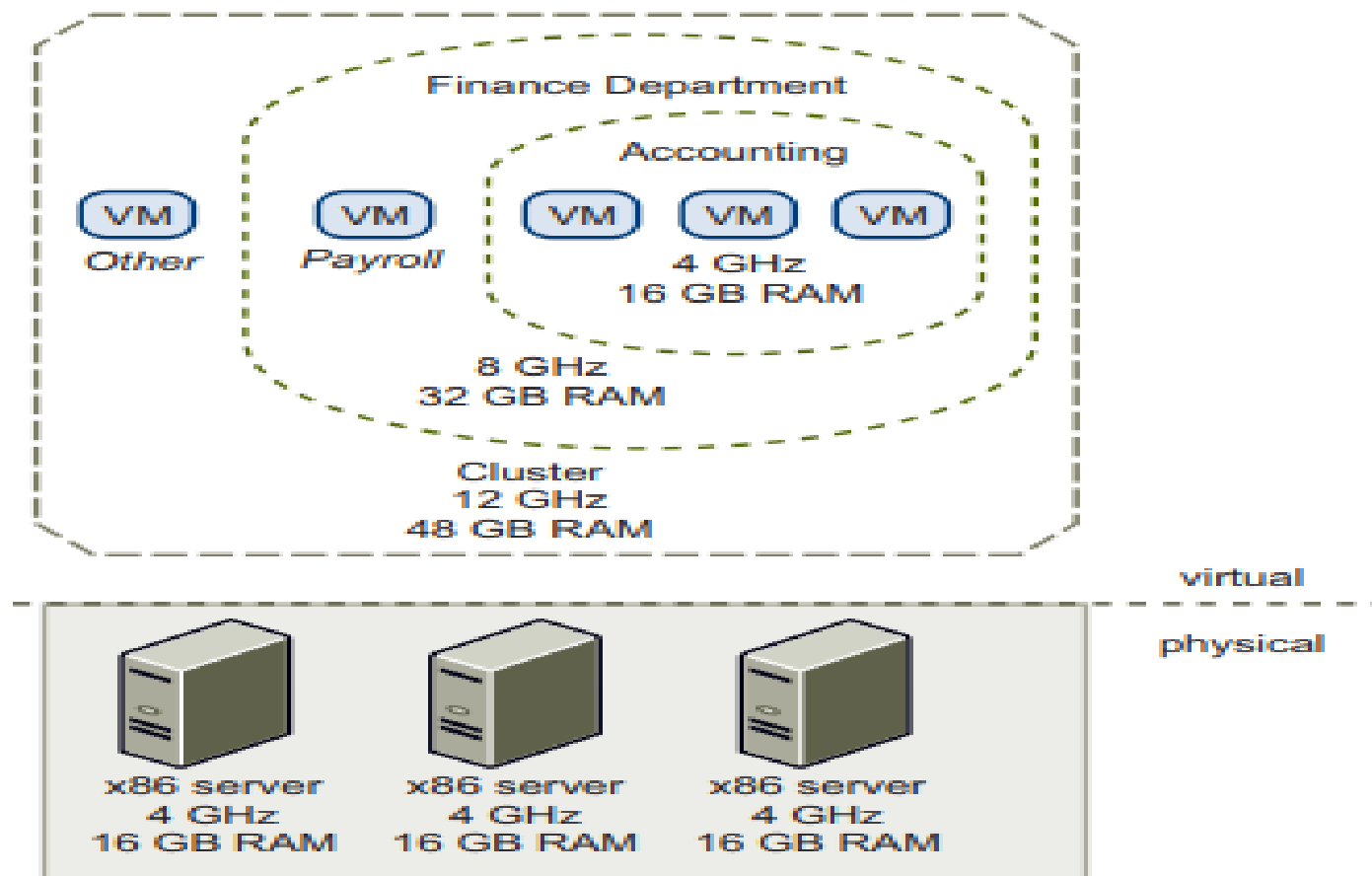
I.2.Clusters

Un cluster représente l'ensemble des ressources de calcul et de mémoire d'un groupe de serveurs physiques partageant le même réseau et les mêmes baies de stockage. Par exemple, si le groupe contient 8 serveurs, chaque serveur dispose de 4 processeurs double cœur fonctionnant à 4 GHz chacun et 32 Go de mémoire. Le cluster disposera alors de 256 GHz de puissance de calcul et de 256 Go de mémoire disponibles pour les machines virtuelles en cours d'exécution qui lui sont affectées.

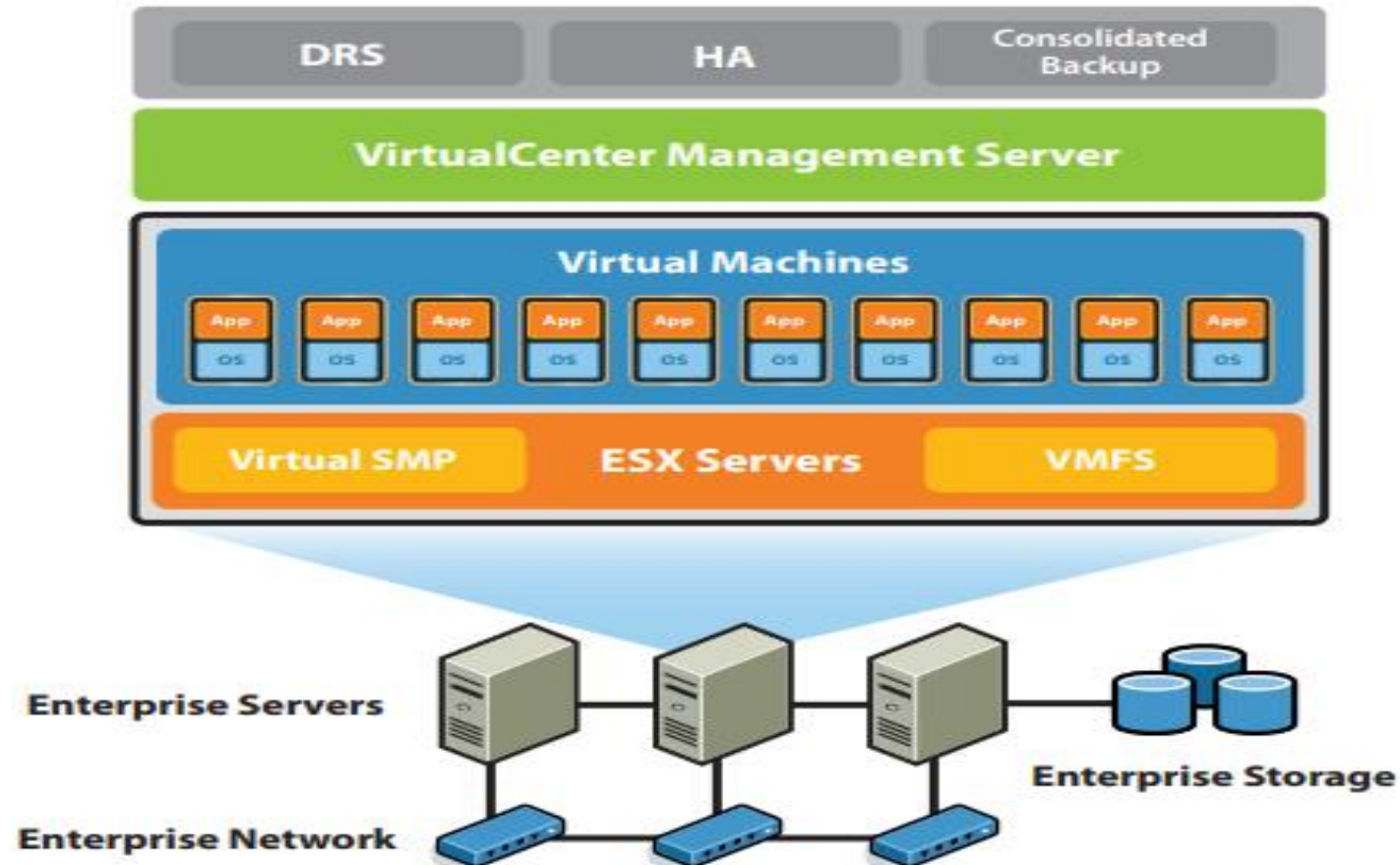
I.2. Pools de ressources

- Les pools de ressources offrent un moyen flexible et dynamique de diviser et d'organiser les ressources de calcul et de mémoire d'un hôte ou d'un cluster.
- Même si les ressources sont réservées à différents pools de ressources, elles ne sont pas gaspillées si elles ne sont pas utilisées par leur propriétaire.

I.2. Pools de ressources



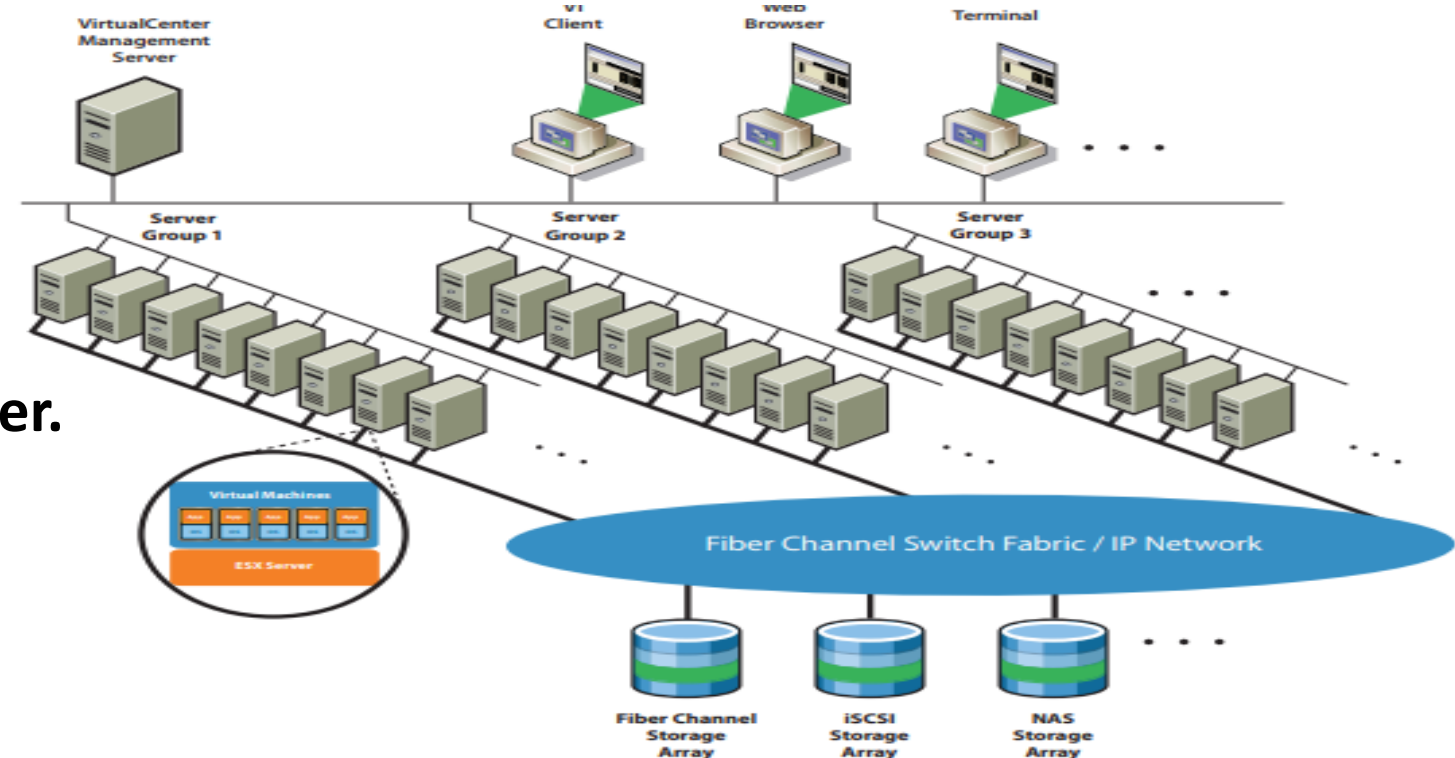
I.2. Virtualisation du centre de données (VMware)



I.2. Topologie physique du centre de données d'infrastructure VMware

Avec VMware Infrastructure, les services informatiques peuvent créer un centre de données virtuel en utilisant leur technologie et leur matériel standard existants.

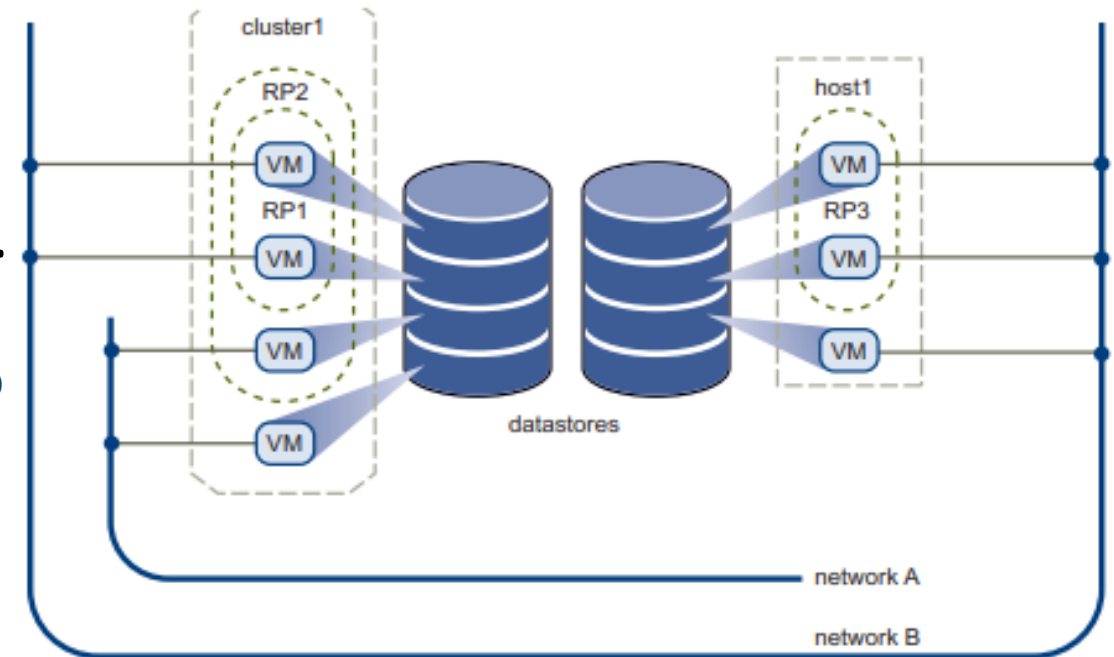
- Computing Servers.
- Storage Networks and Arrays.
- IP Networks.
- VirtualCenter Management Server.
- Desktop Clients.



I.2. Architecture du centre de données virtuel

Avec VMware Infrastructure, les ressources informatiques peuvent être gérées comme un utilitaire partagé et provisionnées dynamiquement vers différentes unités commerciales et projets sans se soucier des différences et limitations matérielles sous-jacentes.

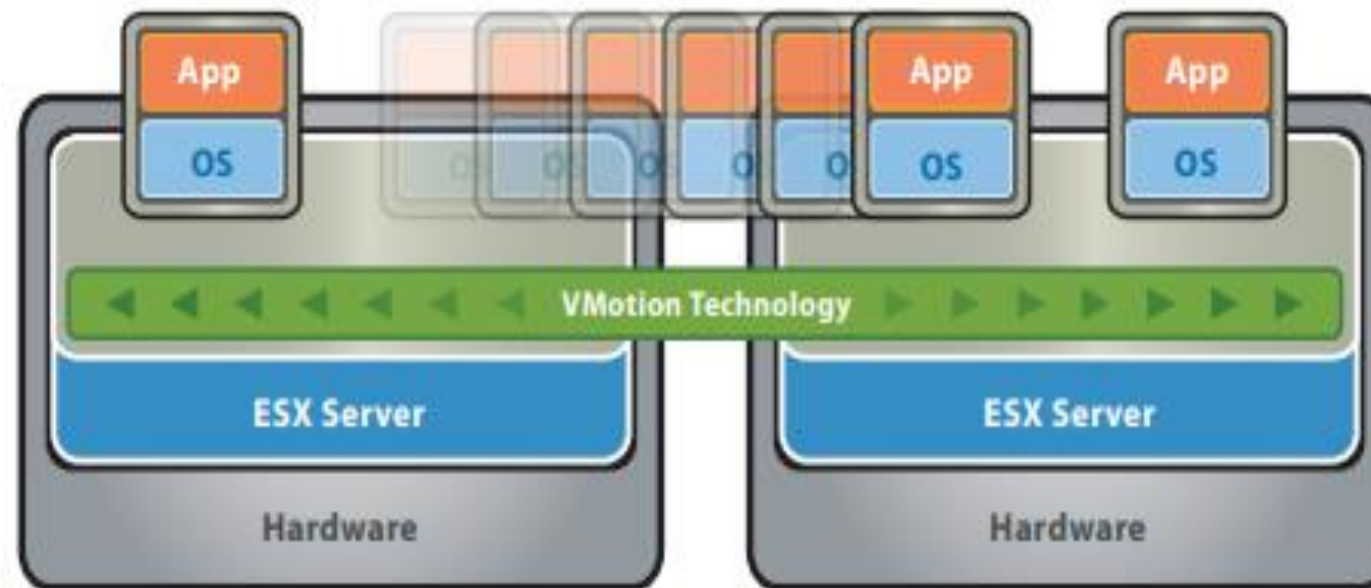
- **Ressources informatiques et mémoire appelées hôtes, clusters et Pools de ressources.**
 - **Ressources de stockage appelées datastores.**
 - **Ressources de mise en réseau appelées réseaux.**
 - **Machines virtuelles.**
- Le provisionnement des machines virtuelles est beaucoup plus rapide et plus facile que les machines physiques.
 - Les ressources sont fournies aux machines virtuelles en fonction de les politiques définies par l'administrateur système qui possède les ressources.



I.2. VMware VMotion, VMware DRS and VMware HA

- VMware Vmotion

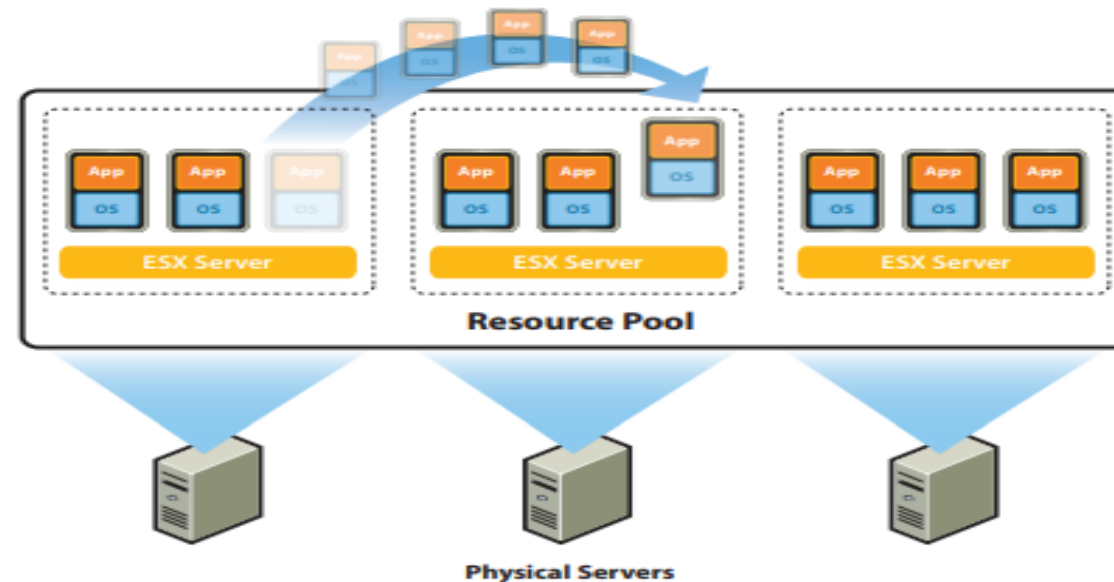
Permet la migration en direct des machines virtuelles en cours d'exécution d'un serveur physique à un autre avec zéro temps d'arrêt, une disponibilité de service continue et une intégrité totale des transactions.



I.2. VMware VMotion, VMware DRS and VMware HA

- VMware DRS

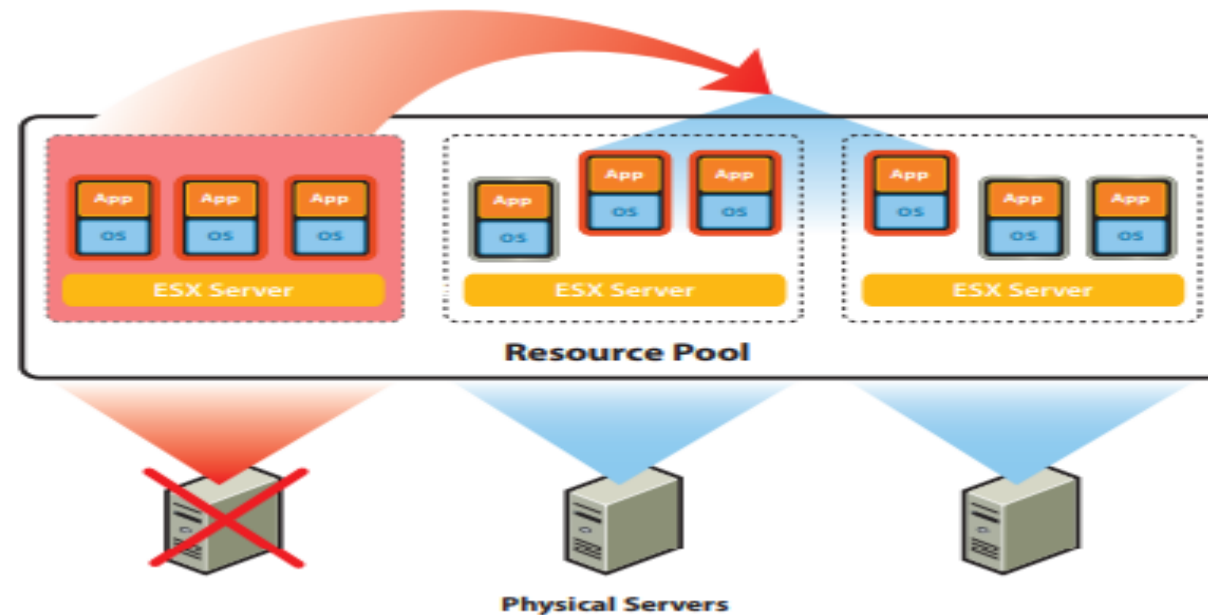
VMware DRS permet à l'administrateur système de définir des stratégies d'attribution de ressources qui reflètent les besoins de l'entreprise et de laisser VMware DRS faire le calcul et gérer automatiquement les affectations de ressources physiques détaillées.



I.2. VMware VMotion, VMware DRS and VMware HA

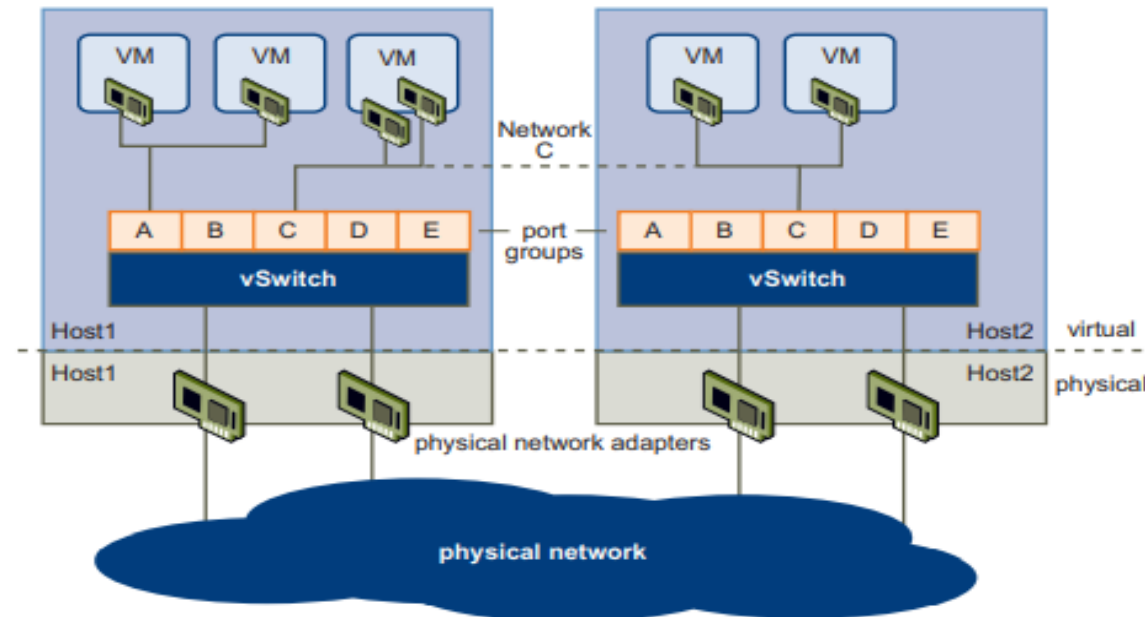
- VMware HA

VMware HA offre une alternative haute disponibilité simple et économique au clustering d'applications.



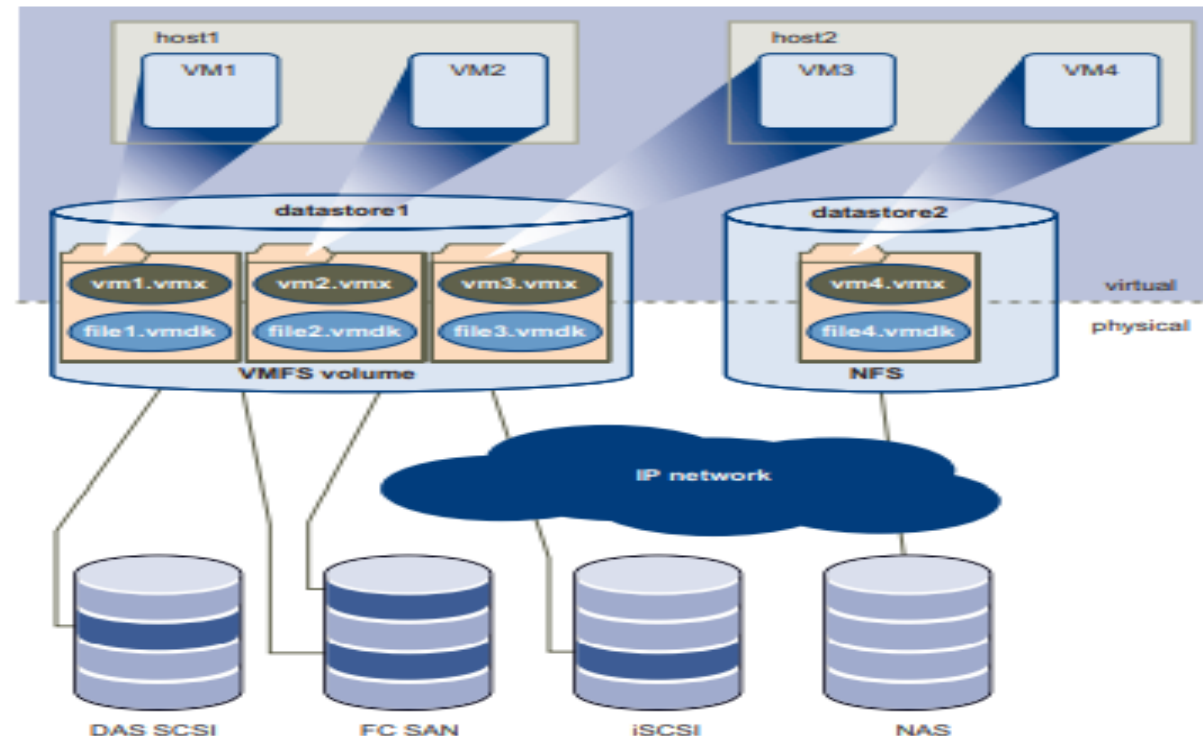
I.2. Architecture réseau

- VMware Infrastructure est la seule solution qui fournit un riche ensemble d'éléments de réseau virtuel.
- NIC associant deux ou plusieurs adaptateurs physiques utilisés pour partager la charge de trafic ou fournir un basculement en cas de panne matérielle de l'adaptateur.
- Port Group est un concept unique dans l'environnement virtuel. Un groupe de ports est un mécanisme permettant de définir des stratégies qui régissent le réseau qui y est connecté.



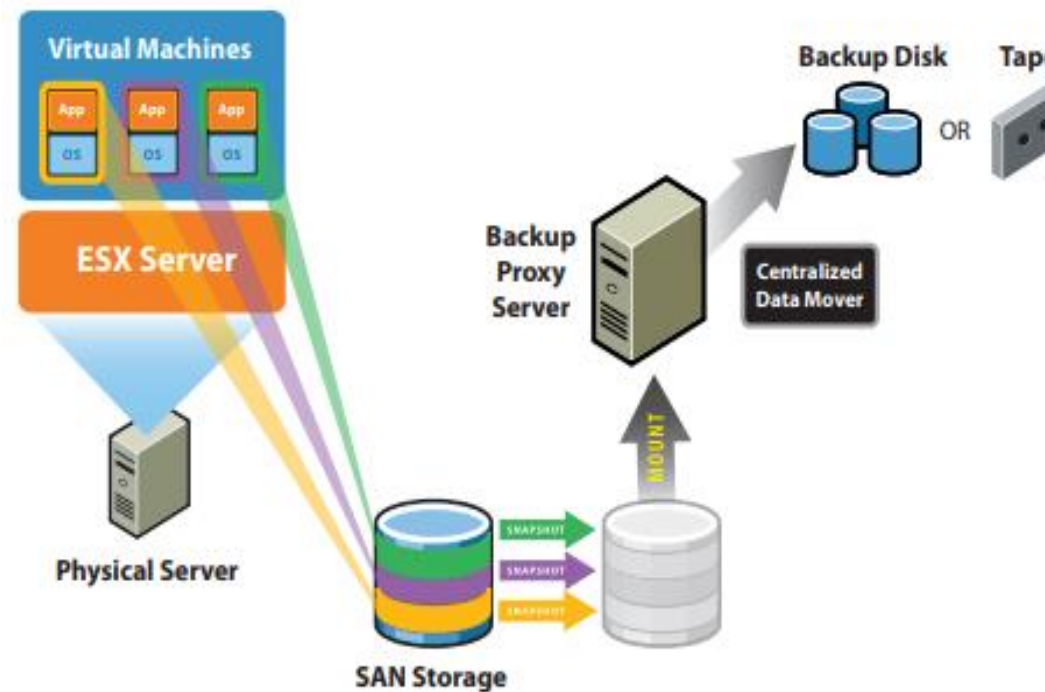
I.2. Architecture du stockage

- L'infrastructure VMware offre des performances, des fonctionnalités et une disponibilité de stockage de classe entreprise sans ajouter de complexité.
- Les disques virtuels peuvent être «ajoutés à chaud» à une machine virtuelle sans la mettre hors tension.



I.2.VMware Consolidated Backup

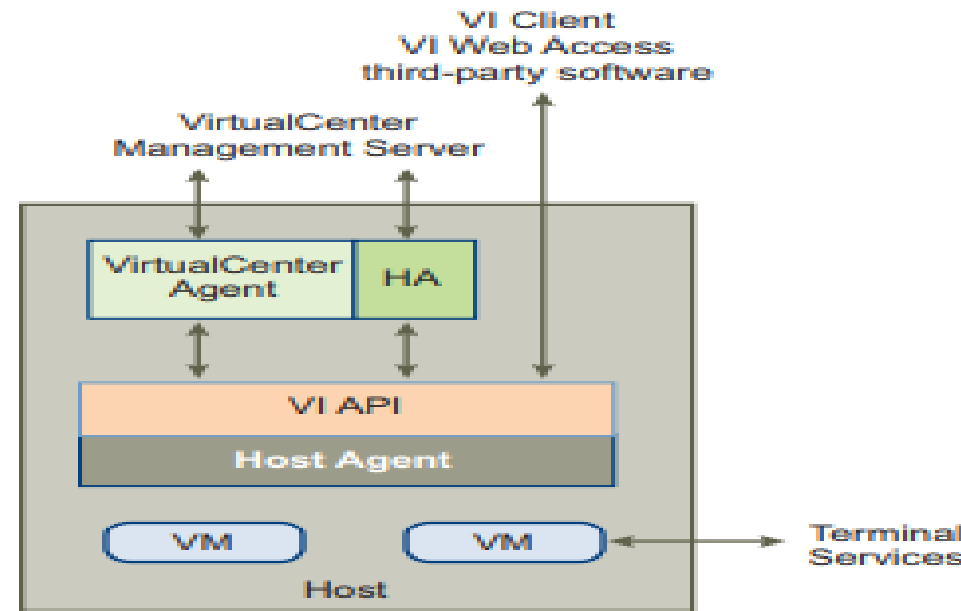
Consolidated Backup offre une fonction centralisée et facile à utiliser pour la sauvegarde sans agent des machines virtuelles.



I.2. ESX Server External Interfacing Components

Certaines des tâches clés effectuées par VC Agent sont:

- Relayer et appliquer les décisions d'allocation de ressources prises dans VirtualCenter Management Server (y compris celles envoyées par le moteur VMware DRS).
- Transmission des commandes de provisionnement de machine virtuelle et de changement de configuration à l'agent hôte.
- Transmission des commandes de modification de la configuration de l'hôte à l'agent hôte.
- Surveillance et collecte de statistiques de performances, d'alarmes et d'événements en communiquant avec l'agent hôte via l'API VI.
- Renvoyer les informations au VirtualCenter Management Server une fois qu'une certaine quantité d'informations est collectée.



I.2. VirtualCenter Management Server Architecture

VirtualCenter Management Server fournit un cockpit de gestion centralisé pratique pour le centre de données.

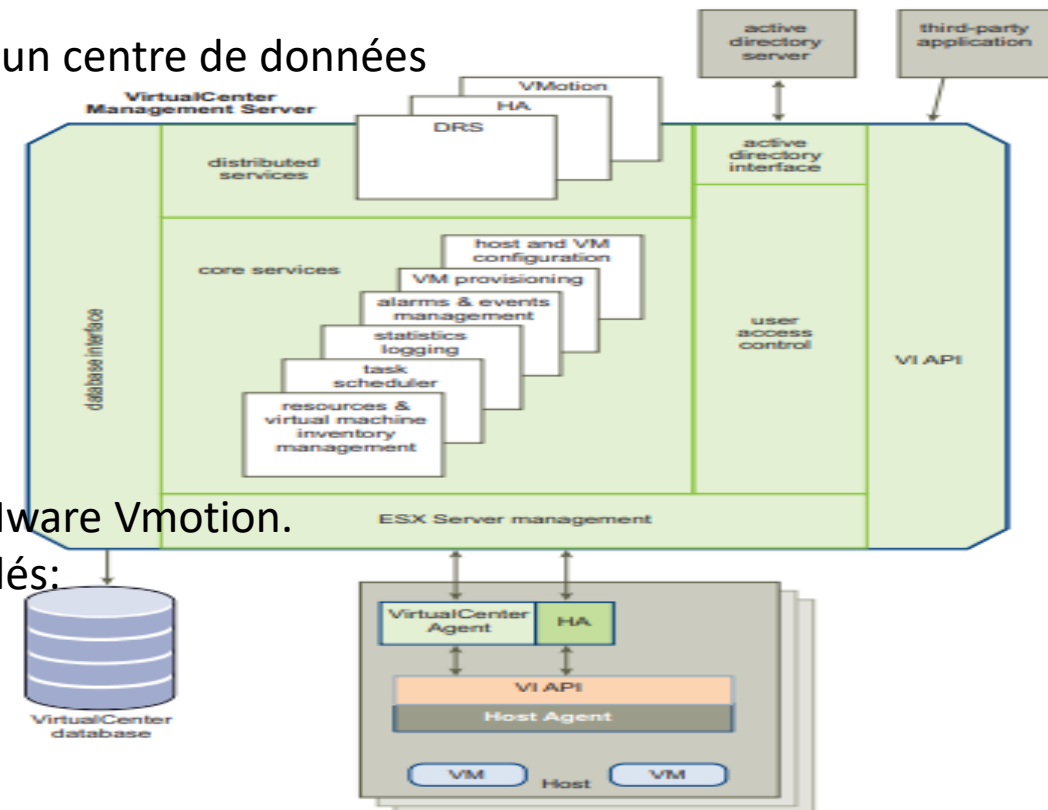
Les services de base sont des services de gestion de base pour un centre de données virtuel. Ils comprennent des services tels que:

- VM Provisioning
- Host and VM Configuration
- Resource and Virtual Machine Inventory Management
- Statistics and Logging
- Alarms and Event Management
- Task Scheduler

Services distribués, comme VMware DRS, VMware HA, and VMware Vmotion.

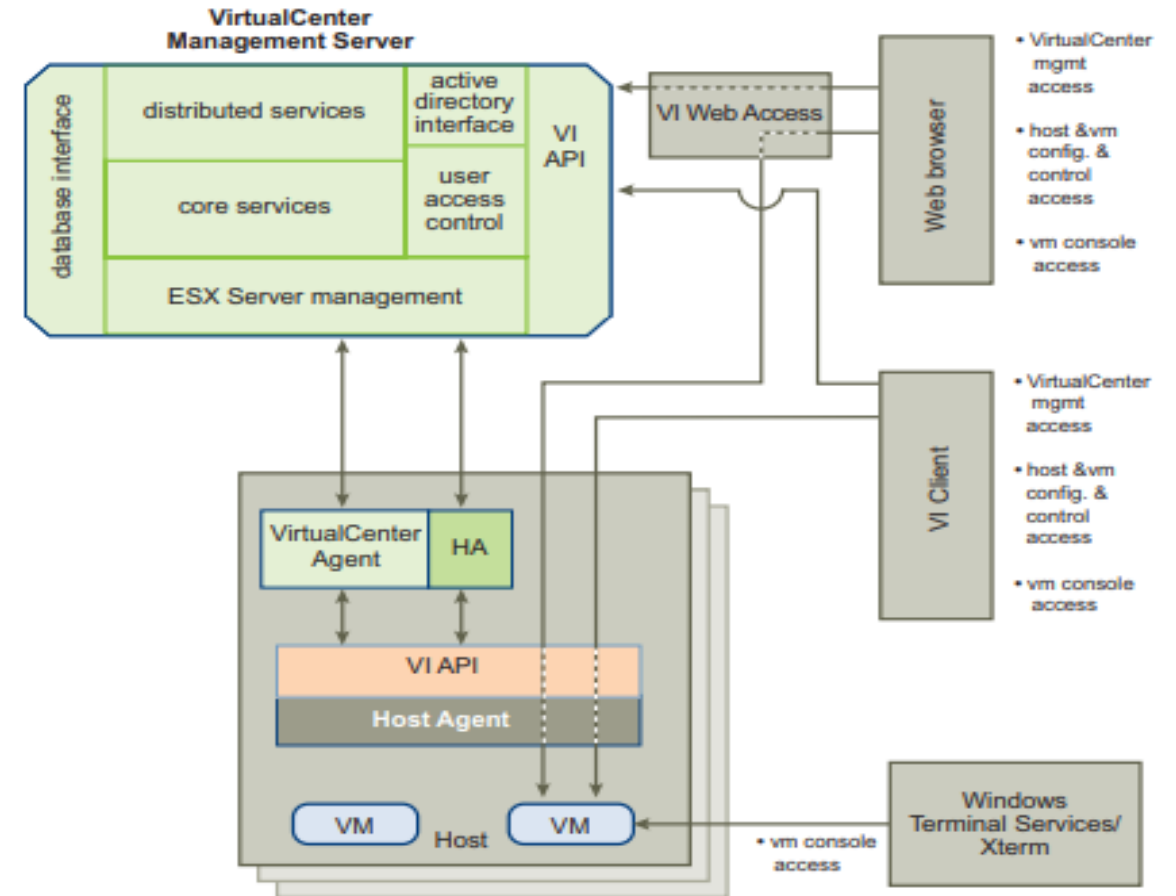
VirtualCenter Management Server possède quatre interfaces clés:

- ESX Server Management
- VMware Infrastructure API
- Database Interface
- Active Directory Interface



I.2. Accessing the Virtual Data Center

- VI Client
- Web Access (Web browser)
- Terminal services (such as Windows Terminal Services or Xterm).



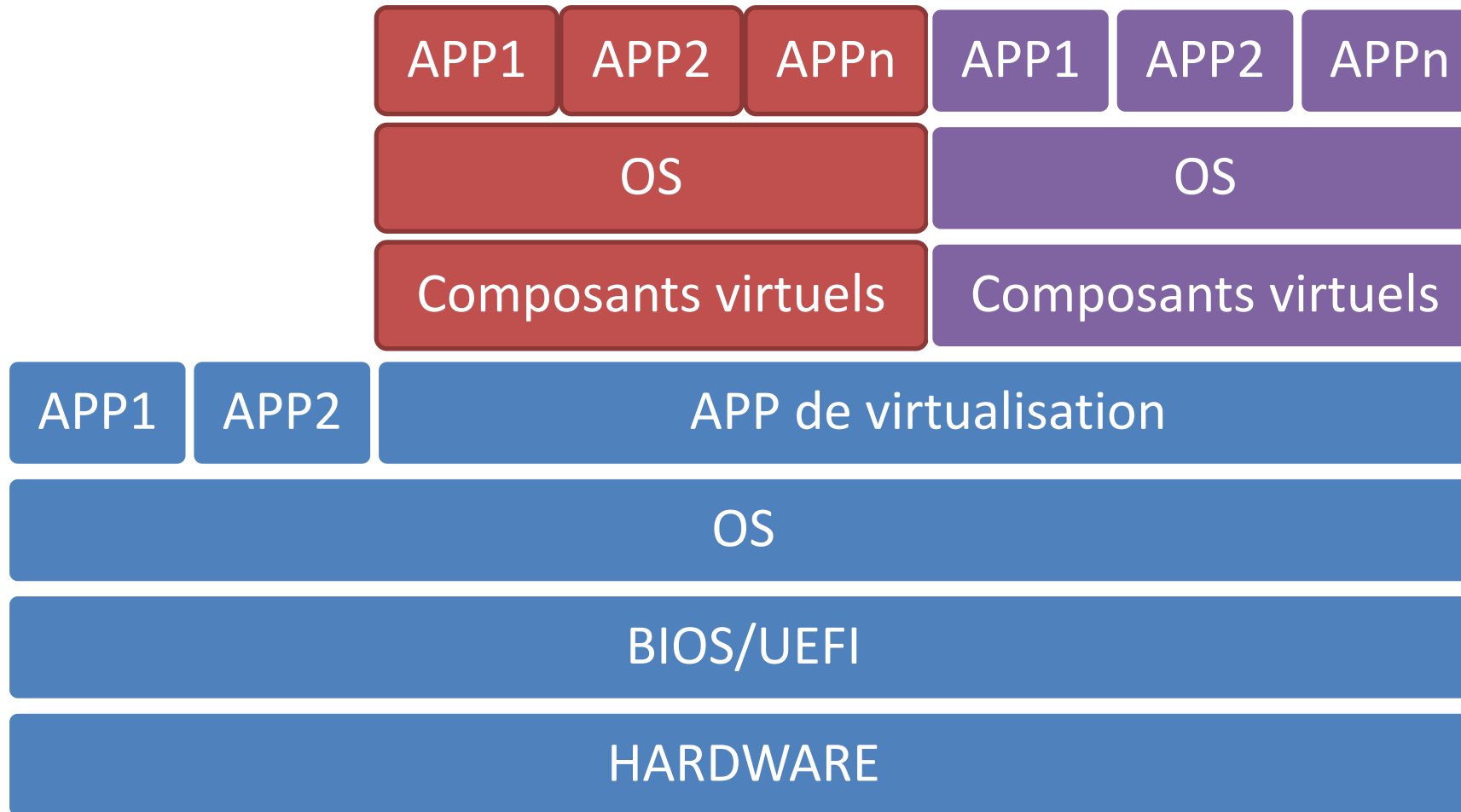
I.2.Virtualisation du centre de données (VMware)

Avec une suite complète de services de virtualisation et de gestion complémentaires tels que VMware VMotion virtuel, VMware DRS, VMware HA et VMware Consolidated Backup, **VMware Infrastructure** est la seule offre qui offre une solution complète plutôt qu'une approche fragmentaire permettant aux clients de construire un centre de données dans un environnement virtuel.

I.2. Virtualisation du centre de données (VMware)

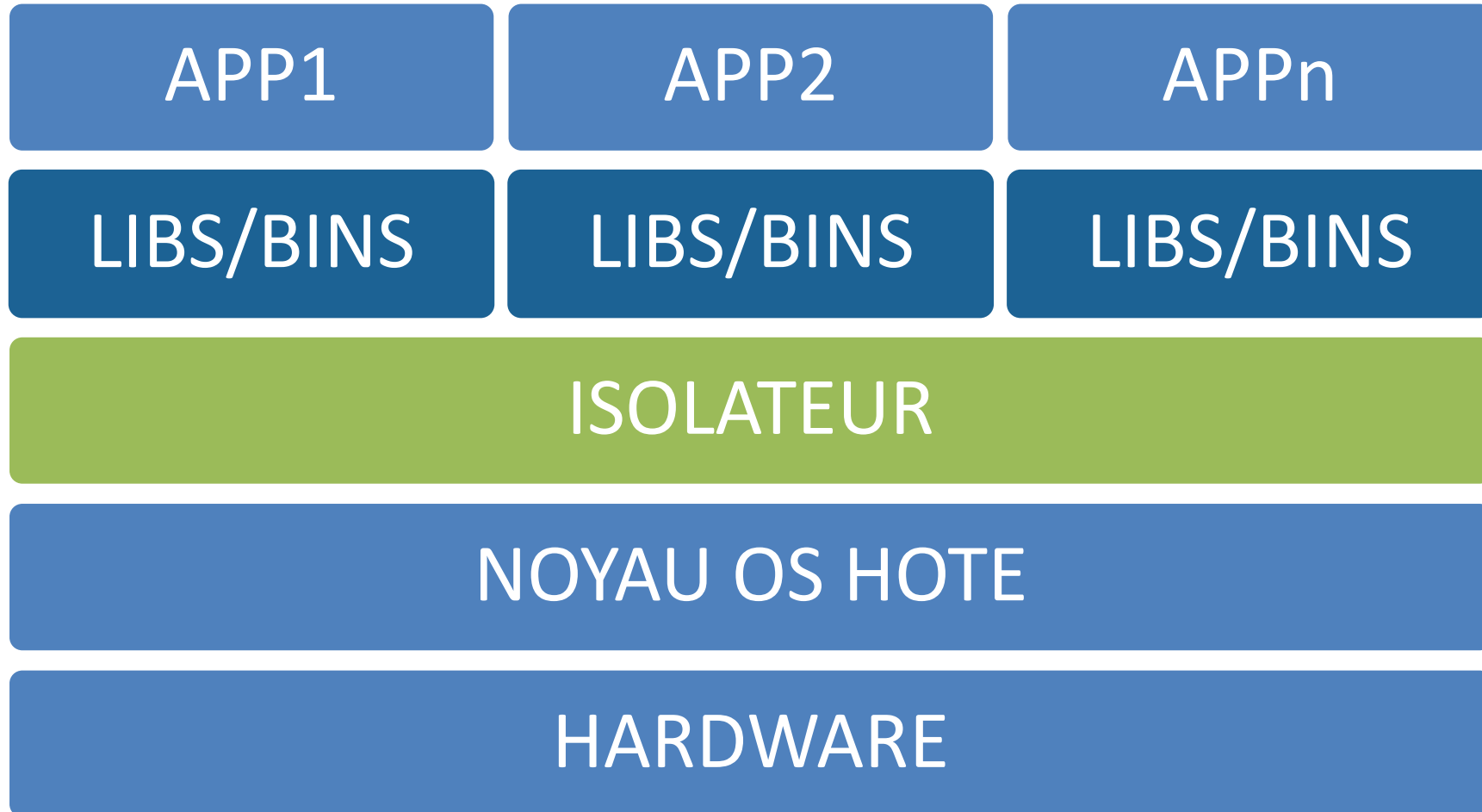
- VMware ESX Server.
- VMware Virtual Machine File System (VMFS).
- VMware Virtual Symmetric Multi-Processing (SMP).
- VirtualCenter Management Server.
- Virtual Infrastructure Client (VI Client).
- Virtual Infrastructure Web Access.
- VMware Vmotion.
- VMware High Availability (HA).
- VMware Distributed Resource Scheduler (DRS).
- VMware Consolidated Backup.
- VMware Infrastructure SDK.

I.3.1. Généralités



Evolution...

I.3.1. Conteneurisation



I.3.1. Conteneurisation

- Regroupe le code et toutes ses dépendances pour que l'application s'exécute rapidement et de manière fiable d'un environnement informatique à un autre.
- Les conteneurs isolent le logiciel de son environnement et s'assurent qu'il fonctionne de manière uniforme malgré les différences, par exemple entre le développement et la mise en scène.
- Les logiciels conteneurisés fonctionneront toujours de la même manière, quelle que soit l'infrastructure.

I.3.1. Conteneurisation

- Il s'agit d'une approche qui s'appuie directement sur les fonctionnalités du noyau (**Kernel**) pour créer des environnements virtuels (Virtual Environment-VE) isolés les uns des autres. Ces VE sont appelés conteneurs, tandis que les fonctionnalités fournies par le noyau du système d'exploitation sont les groupes de contrôles (**cgroups**) et les espaces de noms (**namespaces**).
- Les **namespaces** permettent de contrôler et de limiter la quantité de ressources utilisée pour un processus, tandis que les **cgroups** gèrent les ressources d'un groupe de processus. Un conteneur fournit donc les ressources nécessaires pour exécuter des applications comme si ces dernières étaient les seuls processus en cours d'exécution dans le système d'exploitation de la machine hôte.

I.3.1.Conteneurisation

Paramètres	Machine Virtuelle	Conteneur Docker
Communication	Ethernet	Via les mécanismes standards tels que les sockets, les pipes, la mémoire partagée, etc
Isolation	Le partage de ressources librairies, fichiers, etc. entre les machine virtuelle est impossible	Le partage est possible. Les sous-répertoires peuvent être montés de manière transparente
OS invité	Chaque VM virtualise le matériel hôte et charge son propre OS	Le conteneur utilise directement l'OS hôte
Stockage	Nécessite assez du stockage	Nécessite moins d'espace du stockage
Sécurité	Dépend du paramétrage de l'hyperviseur	Dépend des contrôles des accès
Temps de démarrage	Nécessite quelques minutes pour démarrer	Ne Nécessite que quelques secondes afin de démarrer
Utilisation des ressources (CPU et RAM) de la machine hôte	Forte	Native

I.3.1. Historique

- Chroot sur Unix (1979).
- Jail sur BSD (2000).
- Linux Vserver (2001).
- conteneurs sur Solaris (2004).
- LXC (Linux conteneurs) sur Linux (2008).
- Docker (2013).

I.3.1. Caractéristiques de conteneurisation

- Flexible.
- Léger.
- Interchangeable.
- Portable.
- Évolutif.
- Empilable.
- Agilité.
- Isolation des erreurs.

I.3.1. Inconvénients de conteneurisation

- Difficile de gérer de façon efficiente un grand nombre de conteneurs simultanément.
- Problème en cas de Bugs.
- Difficulté à les configurer et à les gérer.
- Incompatibilité avec certaines tâches.
- Problème des dépendances.
- Faiblesse relative de l'isolement.
- Risque de prolifération.
- Outils de gestion limités.

I.3.1. Solutions

- Docker
- Singularity
- Shifter
- LXC/LXD
- Rocket



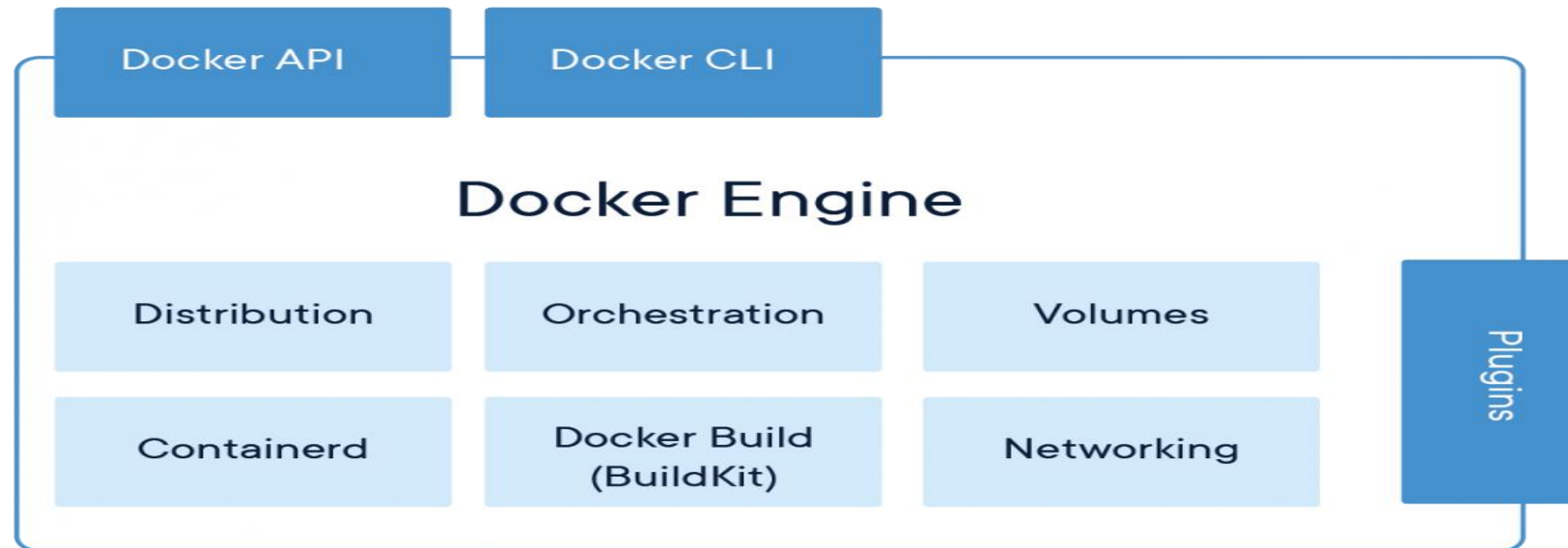
Architecture
Droit/Sécurité
Images
Distribution hôte
Distribution Linux pour les images container
Cadre d'utilisation
Position sur le marché
Documentation/ Complexité d'utilisation
Orchestration

I.3.2. Mise en œuvre des conteneurs avec Docker

- **Docker** est une plateforme open source qui permet à des applications d'être déployées à partir de conteneurs.
- Une **unité standardisée** d'un logiciel.
- **Historique** (dotCloud; Mars 2013 projet open source; en novembre 2019, la société Docker revend son produit « Docker Enterprise » à Mirantis et effectue une levée de fond de près de 35 millions de dollars).
- **Docker Engine** est le moteur d'exécution de conteneur de facto de l'industrie qui s'exécute sur divers systèmes d'exploitation Linux (CentOS, Debian, Fedora, Oracle Linux, RHEL, SUSE et Ubuntu) et Windows.
- **Edition** (community, entreprise)



I.3.2. Mise en œuvre des conteneurs avec Docker

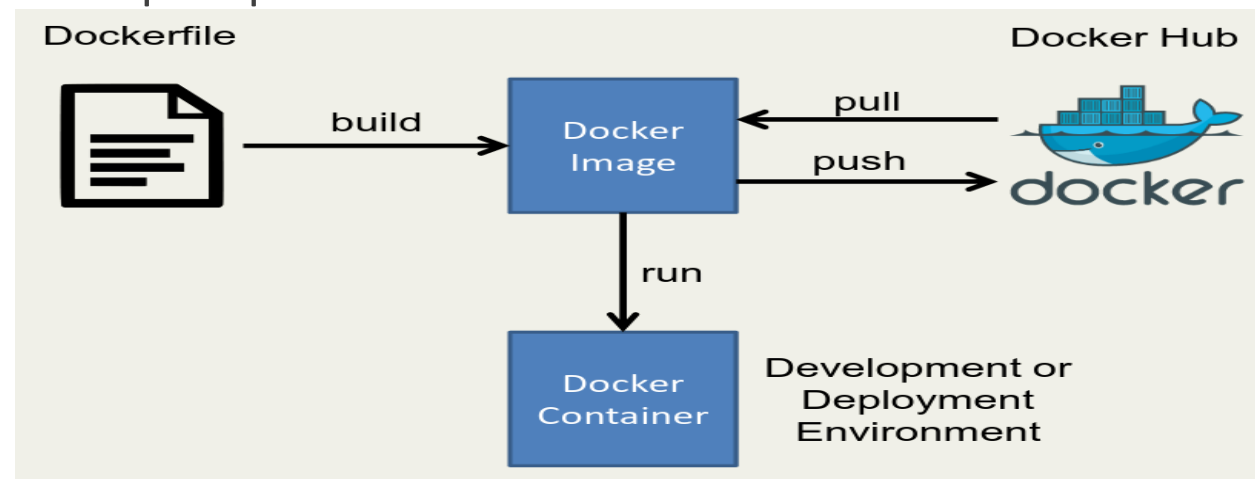


I.3.2. Mise en œuvre des conteneurs avec Docker

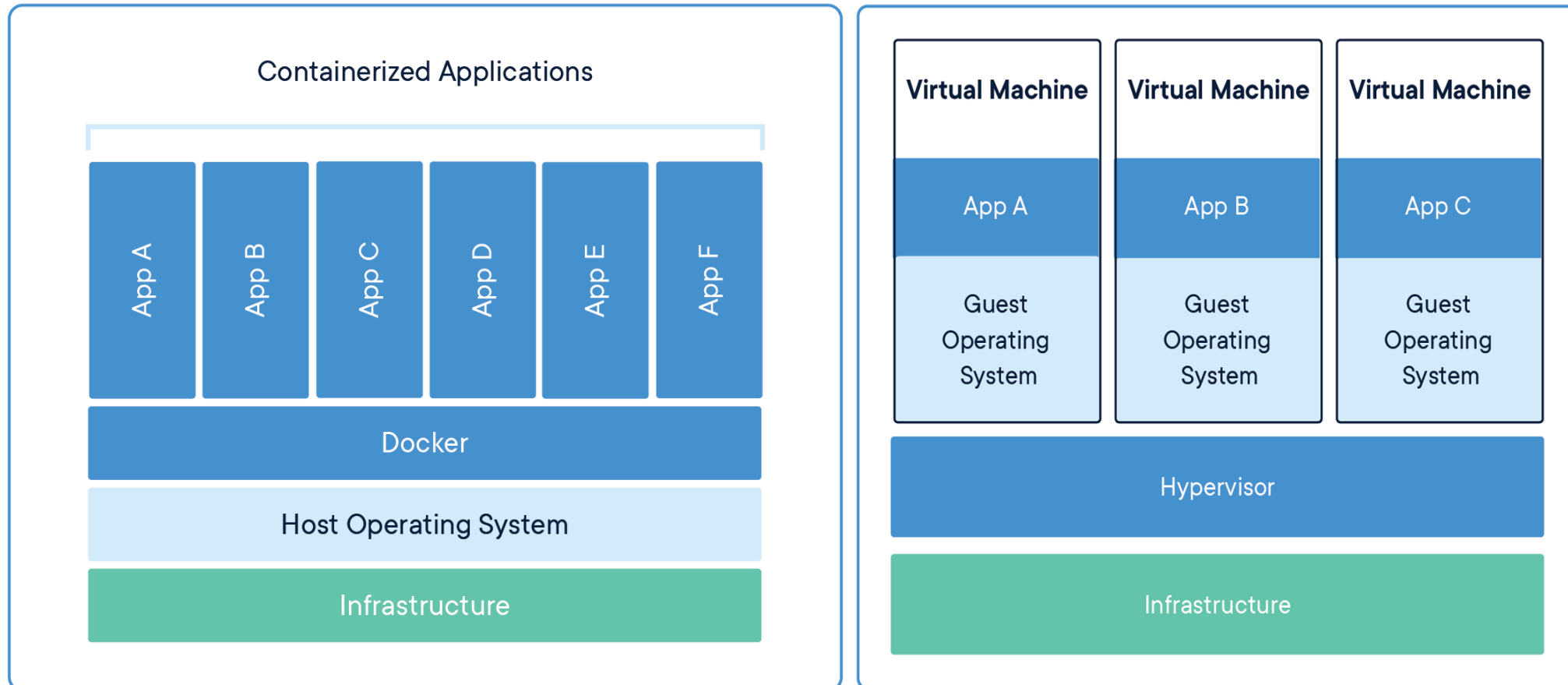
- Un conteneur docker est une enveloppe virtuelle qui permet de packager une application avec tous les éléments dont elle a besoin pour fonctionner : fichiers source, runtime, bibliothèques, outils et fichiers. Ils sont packagés en un ensemble cohérent et prêt à être déployé sur un serveur et son OS.
- Une image docker est un modèle permettant de créer un conteneur. Elle correspond à une manière simple de réutiliser et déplacer des données. Ces données représentées par une image correspondent au contenu d'un conteneur. Il est possible de créer des images ou de télécharger et d'utiliser des images créées et mises à disposition par d'autres personnes.

I.3.2. Mise en œuvre des conteneurs avec Docker

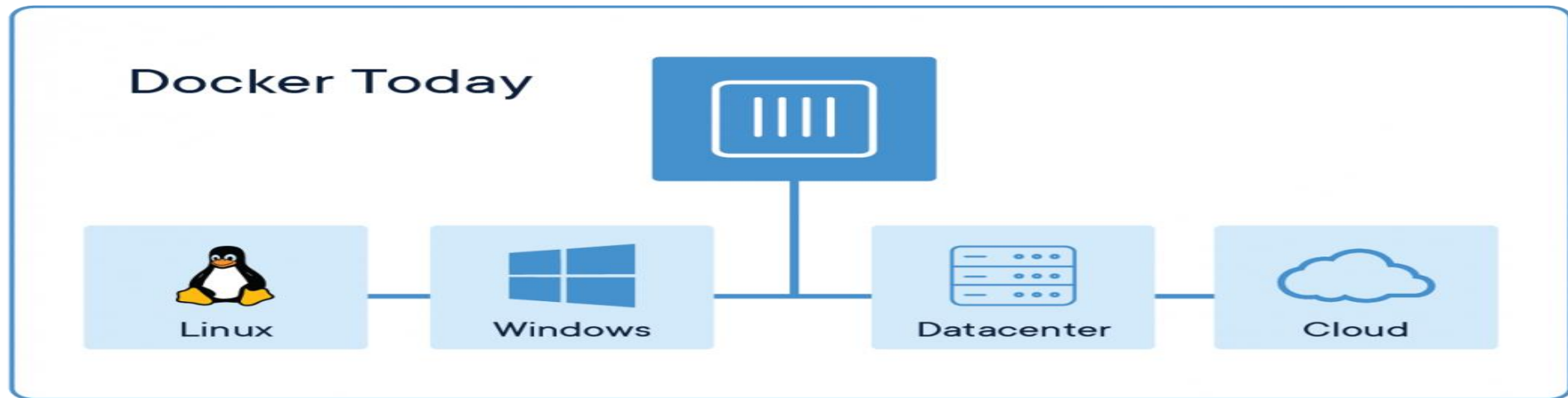
- Rechercher et partager des images de conteneurs avec votre équipe et la communauté Docker.
- Docker Hub est le plus grand référentiel d'images de conteneurs au monde avec un éventail de sources de contenu, notamment des développeurs de communautés de conteneurs, des projets open source et des éditeurs de logiciels indépendants. Les utilisateurs ont accès à des référentiels publics gratuits pour stocker et partager des images ou peuvent choisir un plan d'abonnement pour les dépôts privés.



I.3.2. Mise en œuvre des conteneurs avec Docker



I.3.2. Mise en œuvre des conteneurs avec Docker



I.3.2. Mise en œuvre des conteneurs avec Docker

Docker Desktop

- Mac
- Windows

Docker Engine disponible pour les distributions Linux open source suivantes:

- CentOS
- Debian
- Fedora
- Ubuntu

I.3.2. Mise en œuvre des conteneurs avec Docker Desktop

- Docker Desktop offre la vitesse, le choix et la sécurité dont vous avez besoin pour concevoir et fournir ces applications conteneurisées sur votre bureau.
- Docker Desktop comprend Docker App, les outils de développement, Kubernetes et la synchronisation de version avec les moteurs Docker de production.
- Docker Desktop vous permet d'exploiter des images et des modèles certifiés ainsi que votre choix de langues et d'outils. Les workflows de développement tirent parti de Docker Hub pour étendre votre environnement de développement à un référentiel sécurisé pour une construction automatique rapide, une intégration continue et une collaboration sécurisée.

I.3.2. Installation Docker Desktop

Docker Desktop pour Windows

Installer

Double-cliquez sur Docker pour Windows Installer pour exécuter le programme d'installation.

Exécution

Ouvrez un terminal de ligne de commande comme PowerShell et essayez quelques commandes Docker

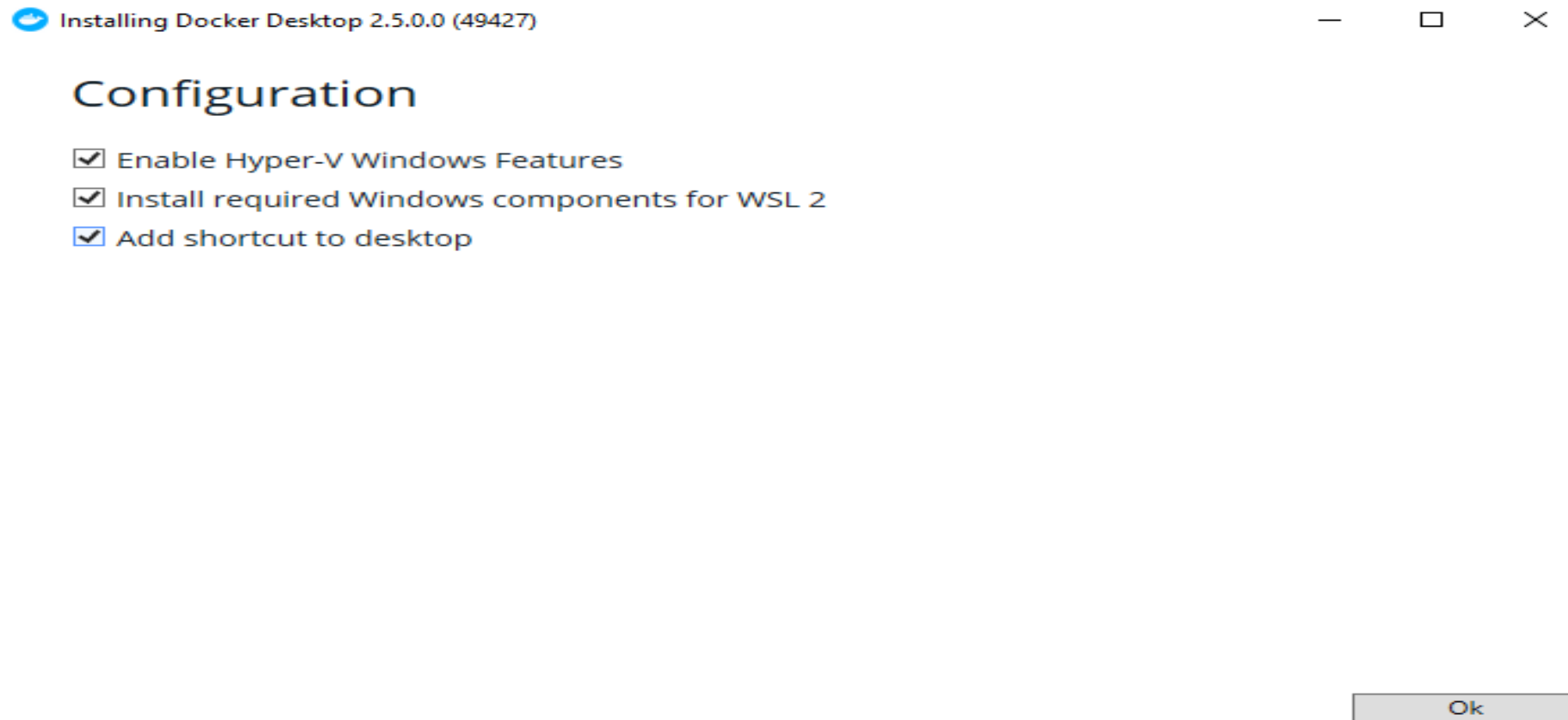
Exécutez ***docker version*** pour vérifier la version.

Exécutez ***docker run hello-world*** pour vérifier que Docker peut extraire et exécuter des images.

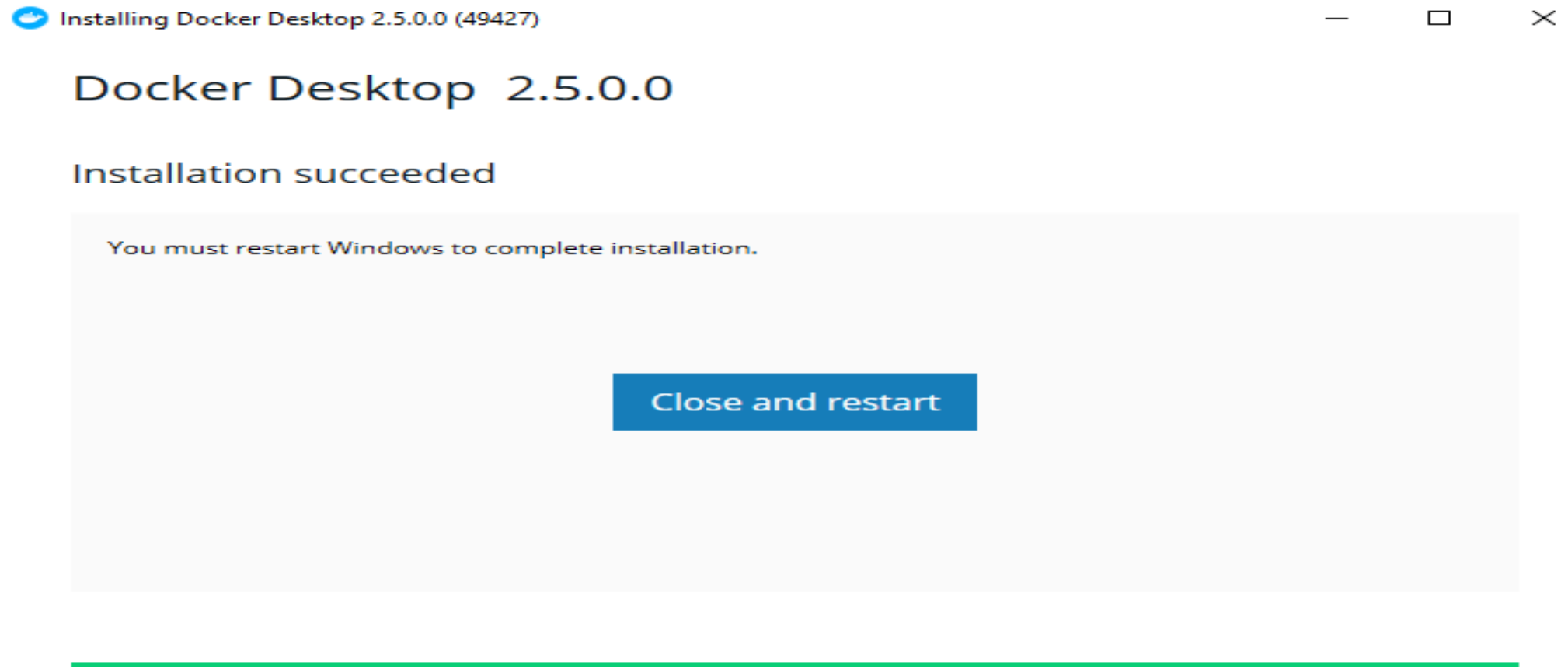
Bénéficier du Docker

Docker est disponible dans n'importe quel terminal tant que l'application Docker Desktop pour Windows est en cours d'exécution. Les paramètres sont disponibles sur l'interface utilisateur, accessible depuis la baleine Docker dans la barre des tâches.

I.3.2. Installation Docker Desktop



I.3.2. Installation Docker Desktop



I.3.2. Installation Docker Desktop



Service is not running



Service is not running

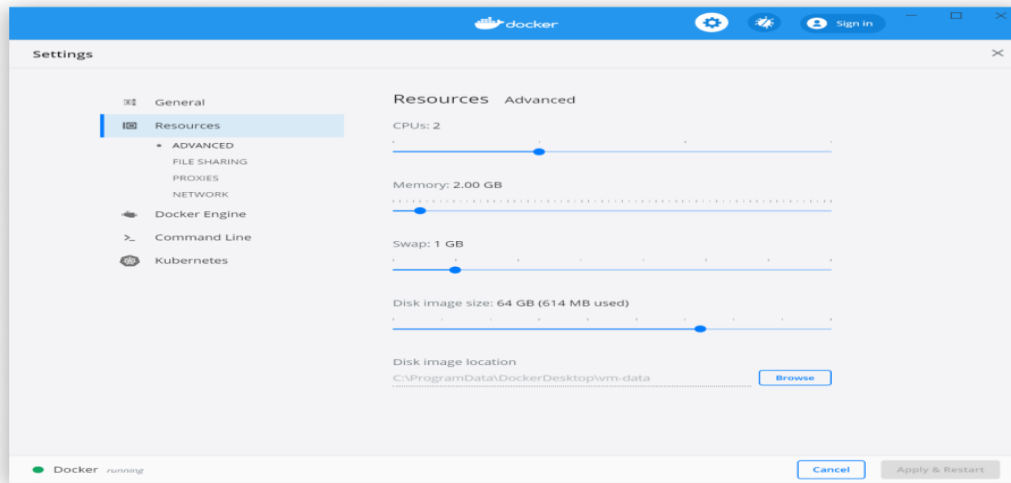
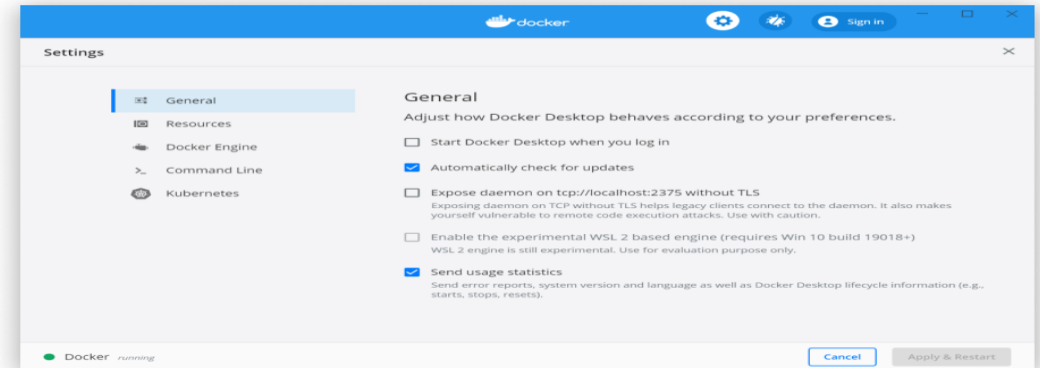
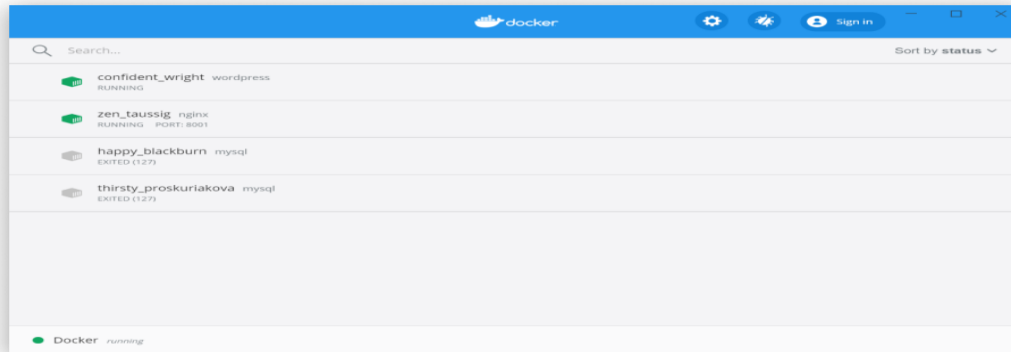


Docker Desktop service is not running, would you like to start it? Windows will ask you for elevated access.

Start

Quit

I.3.2. Installation Docker Desktop



I.3.2. Mise en œuvre des conteneurs avec Docker Engine

Désinstallation

```
$ sudo apt-get remove docker docker-engine docker.io containerd runc
```

Installer à l'aide du script

```
$ sudo apt-get update  
$ sudo curl -fsSL https://get.docker.com -o get-docker.sh  
$ sudo sh get-docker.sh  
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

Lancement des premières images

```
$ sudo docker run hello-world  
$ sudo docker run docker/whalesay cowsay boo
```

I.3.2.Mise en œuvre des conteneurs avec Docker Engine

Commande	Utilisation
Docker version	Afficher les informations de version de Docker
Docker run <image>	Exécuter une commande dans un conteneur en cours d'exécution
Docker run -d <image>	Exécuter le conteneur en arrière-plan
Docker run -d -p <nouveauport:portàmapper image>	Exécuter le conteneur en arrière-plan/Publier le (s) port (s) d'un conteneur sur l'hôte
Docker run --name <name> -e <key=val> -d <image>	Attribuer un nom au conteneur/Définir les variables d'environnement/Exécuter le conteneur en arrière-plan
Docker run <image: tag>	vous pouvez spécifier une version d'une image avec laquelle vous souhaitez exécuter le conteneur
Docker exec -it <image> option <option>	Exécuter une commande dans un conteneur en cours d'exécution
Docker images	Liste des images
Docker ps	Liste des conteneurs
Docker ps -a	Afficher tous les conteneurs (la valeur par défaut montre juste en cours d'exécution)
Docker rm <idconteneur>	Supprimer un ou plusieurs conteneurs
Docker rmi <idimage>	Supprimer un ou plusieurs images
docker stop <idconteneur>	Arrêtez un ou plusieurs conteneurs en cours d'exécution
docker start <idconteneur>	Démarrer un ou plusieurs conteneurs arrêtés
Docker attach <idconteneur>	Attachez des flux d'entrée, de sortie et d'erreurs standard locaux à un conteneur en cours d'exécution
Docker logs <idconteneur>	Récupérer les logs d'un conteneur
Docker inspect <idconteneur>	Renvoyer des informations de bas niveau sur les objets Docker

I.3.2. API REST Docker

- Sudo nano/lib/systemd/system/docker.service
- Execstart=/usr/bin/dockerd -H fd:// -H tcp://0.0.0.0:2375
--containerd=/run/containerd/containerd.sock
- Sauvegarder le fichier modifié (ctrl+o)
- Ctrl+x
- Sudo systemctl daemon-reload
- Sudo service docker restart
- <http://localhost:2375/images/json>

I.3.2. API REST Docker

- <https://jcutrer.com/windows/install-chocolatey-choco-windows10>

```
PS C:\WINDOWS\system32> Set-ExecutionPolicy Bypass -Scope Process -Force;`  
>> iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))
```

- <https://chocolatey.org/packages/docker-cli>

Choco install docker-cli

- Variable d'environnement

DOCKER_HOST=tcp://@ip:port

I.3.2. Créer une image Docker

- **Docker file** est un fichier texte qui contient les commandes nécessaires pour la description d'une image Docker.
- Des configurations et informations pour utiliser les versions des logiciels et dépendances pour assurer un déploiement stable et consistant.
- Docker build & run.

I.3.2.Créer une image Docker

Un [Dockerfile](#) est un fichier de configuration contenant plusieurs instructions permettant de construire une Docker image. Les instructions concernent plusieurs tâches comme la création d'un répertoire, la copie de fichiers, etc.

Instruction	Tâche
ADD	L'instruction ADD copie les nouveaux fichiers, répertoires ou URL de fichiers distants depuis <src> et les ajoute au système de fichiers de l'image au chemin <dest>.
CMD	Spécifie la commande à exécuter dans le conteneur.
COPIE	Copie des fichiers depuis un fichier de l'OS hôte vers l'image
ENTRYPOINT	Commande à effectuer au démarrage du container
ENV	Définit une variable d'environnement dans l'image
EXPOSE	Expose/ouvre un port du container
FROM	Indique l'image sur laquelle se base la nouvelle image
RUN	Effectue une commande, compatible avec l'image d'origine, au moment de la construction de l'image
USER	Si un service peut s'exécuter sans privilèges, utilisez USER pour passer à un utilisateur non root
VOLUME	Définit un dossier comme un point de montage d'un volume externe
WORKDIR	Pour plus de clarté et de fiabilité, vous devez toujours utiliser des chemins absolus pour votre WORKDIR

I.3.2. Créer une image Docker-exemple

Fichier Docker

From nginx

COPY index.html /usr/share/nginx/html

EXPOSE 80

Build and Run

Docker build -t webapp .

Docker run -d -p 8081:80 webapp

I.3.2. Docker Layered Architecture

- Une image Docker se compose de couches en lecture seule dont chacune représente une instruction Dockerfile. Les couches sont empilées et chacune est un delta des modifications par rapport à la couche précédente.

- Considérez ce Dockerfile:

FROM ubuntu:18.04

COPY . /app

RUN make /app

CMD python /app/app.py

- Chaque instruction crée une couche:

FROM crée un calque à partir de l'image Docker ubuntu: 18.04.

COPY ajoute des fichiers à partir du répertoire actuel de votre client Docker.

RUN construit votre application avec make.

CMD spécifie la commande à exécuter dans le conteneur.

- Lorsque vous exécutez une image et générez un conteneur, vous ajoutez une nouvelle couche writable (la «couche du conteneur») au-dessus des couches sous-jacents. Toutes les modifications apportées au conteneur en cours d'exécution, telles que l'écriture de nouveaux fichiers, la modification de fichiers existants et la suppression de fichiers, sont écrites sur cette couche de conteneur writable mince.
- **Best-practices** : https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

Layers
Base ubuntu layer
Changes in apt packages
Changes in pip packages
Source code
Update entrypoint

I.3.2. Plugin Eclipse

Le **plug-in Eclipse Docker Tooling** offre la possibilité de gérer les images et les conteneurs Docker à partir de l'IDE Eclipse.

I.3.2. Docker Networking

- L'une des raisons pour lesquelles les conteneurs et les services Docker sont si puissants est que vous pouvez les connecter entre eux ou les connecter à des charges de travail non Docker.
- Les conteneurs et services Docker n'ont même pas besoin de savoir qu'ils sont déployés sur Docker ou si leurs homologues sont également des charges de travail Docker ou non. Que vos hôtes Docker exécutent Linux, Windows ou un mélange des deux, vous pouvez utiliser Docker pour les gérer d'une manière indépendante de la plate-forme.

I.3.2. Docker Networking

Le sous-système réseau de Docker est enfichable, à l'aide de pilotes. Plusieurs pilotes existent par défaut et fournissent des fonctionnalités réseau de base:

- **Bridge:** le pilote réseau par défaut. Si vous ne spécifiez pas de pilote, c'est le type de réseau que vous créez. Les réseaux de pont sont généralement utilisés lorsque vos applications s'exécutent dans des conteneurs autonomes qui doivent communiquer.
- **Hôte:** pour les conteneurs autonomes, supprimez l'isolation réseau entre le conteneur et l'hôte Docker et utilisez directement la mise en réseau de l'hôte.
- **Overlay:** les réseaux de overlay connectent plusieurs démons Docker ensemble et permettent aux services swarm de communiquer entre eux. Vous pouvez également utiliser des réseaux de overlay pour faciliter la communication entre un service Swarm et un conteneur autonome, ou entre deux conteneurs autonomes sur différents démons Docker. Cette stratégie supprime la nécessité d'effectuer un routage au niveau du système d'exploitation entre ces conteneurs.
- **Macvlan:** les réseaux Macvlan vous permettent d'attribuer une adresse MAC à un conteneur, le faisant apparaître comme un périphérique physique sur votre réseau. Le démon Docker achemine le trafic vers les conteneurs par leurs adresses MAC. L'utilisation du pilote macvlan est parfois le meilleur choix lorsqu'il s'agit d'applications héritées qui s'attendent à être directement connectées au réseau physique, plutôt qu'à être acheminées via la pile réseau de l'hôte Docker.
- **None:** pour ce conteneur, désactivez tous les réseaux. Habituellement utilisé avec un pilote réseau personnalisé. none n'est disponible pour les services swarm.
- **Plugins réseau:** vous pouvez installer et utiliser des plugins réseau tiers avec Docker. Ces plugins sont disponibles auprès de Docker Hub ou auprès de fournisseurs tiers.

I.3.2. Docker Networking

- Chaque installation de Docker Engine comprend automatiquement trois réseaux par défaut. Vous pouvez les lister:

\$ docker network ls

```
$ docker network ls
```

NETWORK ID	NAME	DRIVER
18a2866682b8	none	null
c288470c46f6	host	host
7b369448dccb	bridge	bridge

- Le réseau nommé bridge est un réseau spécial. Sauf indication contraire, Docker lance toujours vos conteneurs dans ce réseau. Essayez ceci maintenant:

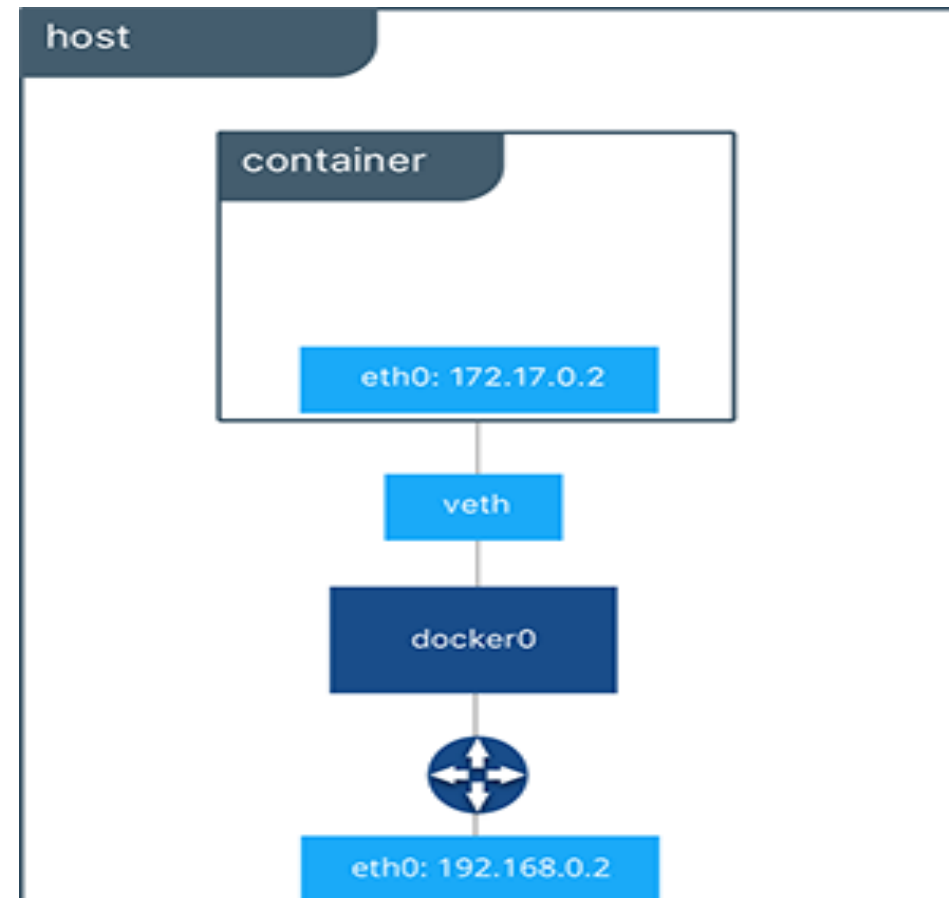
\$ docker run -itd --name = networktest ubuntu

```
$ docker run -itd --name=networktest ubuntu
```



```
74695c9cea6d9810718fddadc01a727a5dd3ce6a69d09752239736c030599741
```

I.3.2. Docker Networking



I.3.2. Docker Networking

L'inspection du réseau est un moyen simple de connaître l'adresse IP du conteneur.

\$ docker network inspect bridge

```
$ docker network inspect bridge
[
  {
    "Name": "bridge",
    "Id": "f7ab26d71dbd6f557852c7156ae0574bbf62c42f539b50c8ebde0f728a253b6f",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": null,
      "Config": [
        {
          "Subnet": "172.17.0.1/16",
          "Gateway": "172.17.0.1"
        }
      ]
    },
    "Internal": false,
    "Containers": {
      "3386a527aa08b37ea9232cbcace2d2458d49f44bb05a6b775fba7ddd40d8f92c": {
        "Name": "networktest",
        "EndpointID": "647c12443e91faf0fd508b6edfe59c30b642abb60dfab890b4bdccee38750bc1",
        "MacAddress": "02:42:ac:11:00:02",
        "IPv4Address": "172.17.0.2/16",
        "IPv6Address": ""
      }
    },
    "Options": {
      "com.docker.network.bridge.default_bridge": "true",
      "com.docker.network.bridge.enable_icc": "true",
      "com.docker.network.bridge.enable_ip_masquerade": "true",
      "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
      "com.docker.network.bridge.name": "docker0",
      "com.docker.network.driver.mtu": "9001"
    },
    "Labels": {}
  }
]
```

I.3.2. Docker Networking

- Vous pouvez supprimer un conteneur d'un réseau en déconnectant le conteneur. Pour ce faire, vous fournissez à la fois le nom du réseau et le nom du conteneur. Vous pouvez également utiliser l'ID de conteneur. Dans cet exemple, cependant, le nom est plus rapide.

```
$ docker network disconnect bridge networktest
```

- Bien que vous puissiez déconnecter un conteneur d'un réseau, vous ne pouvez pas supprimer le réseau de pont intégré nommé bridge. Les réseaux sont des moyens naturels d'isoler les conteneurs des autres conteneurs ou d'autres réseaux.

I.3.2. Docker Networking

- Docker Engine prend en charge nativement les réseaux de pont et les réseaux de overlay. Un réseau pont est limité à un seul hôte exécutant Docker Engine. Un réseau de overlay peut inclure plusieurs hôtes et constitue un sujet plus avancé. Pour cet exemple, [créez un réseau de pont](#):

```
$ docker network create -d bridge my_bridge
```

- L'indicateur -d indique à Docker d'utiliser le pilote de pont pour le nouveau réseau. Vous auriez pu laisser cet indicateur désactivé car bridge est la valeur par défaut de cet indicateur. Allez-y et [listez les réseaux](#) sur votre machine:

```
$ docker network ls
```

```
$ docker network ls
```

NETWORK ID	NAME	DRIVER
7b369448dccb	bridge	bridge
615d565d498c	my_bridge	bridge
18a2866682b8	none	null
c288470c46f6	host	host

I.3.2. Docker Networking

Si vous inspectez le réseau, il ne contient rien.

```
$ docker network inspect my_bridge

[
  {
    "Name": "my_bridge",
    "Id": "5a8afc6364bccb199540e133e63adb76a557906dd9ff82b94183fc48c40857ac",
    "Scope": "local",
    "Driver": "bridge",
    "IPAM": {
      "Driver": "default",
      "Config": [
        {
          "Subnet": "10.0.0.0/24",
          "Gateway": "10.0.0.1"
        }
      ]
    },
    "Containers": {},
    "Options": {},
    "Labels": {}
  }
]
```

I.3.2. Docker Networking

- Pour créer des applications Web qui agissent ensemble mais le font en toute sécurité, créez un réseau. Les réseaux, par définition, assurent une isolation complète des conteneurs. Vous pouvez ajouter des conteneurs à un réseau lors de la première exécution d'un conteneur.
- Lancez un conteneur exécutant une base de données PostgreSQL et transmettez-lui l'indicateur `--net = my_bridge` pour le connecter à votre nouveau réseau:

```
$ docker run -d --net = my_bridge --name db training / postgres
```

- Si vous inspectez votre `my_bridge`, vous pouvez voir qu'il a un conteneur attaché. Vous pouvez également inspecter votre conteneur pour voir où il est connecté:

```
$ docker inspect --format = '{{json .NetworkSettings.Networks}}' db
```

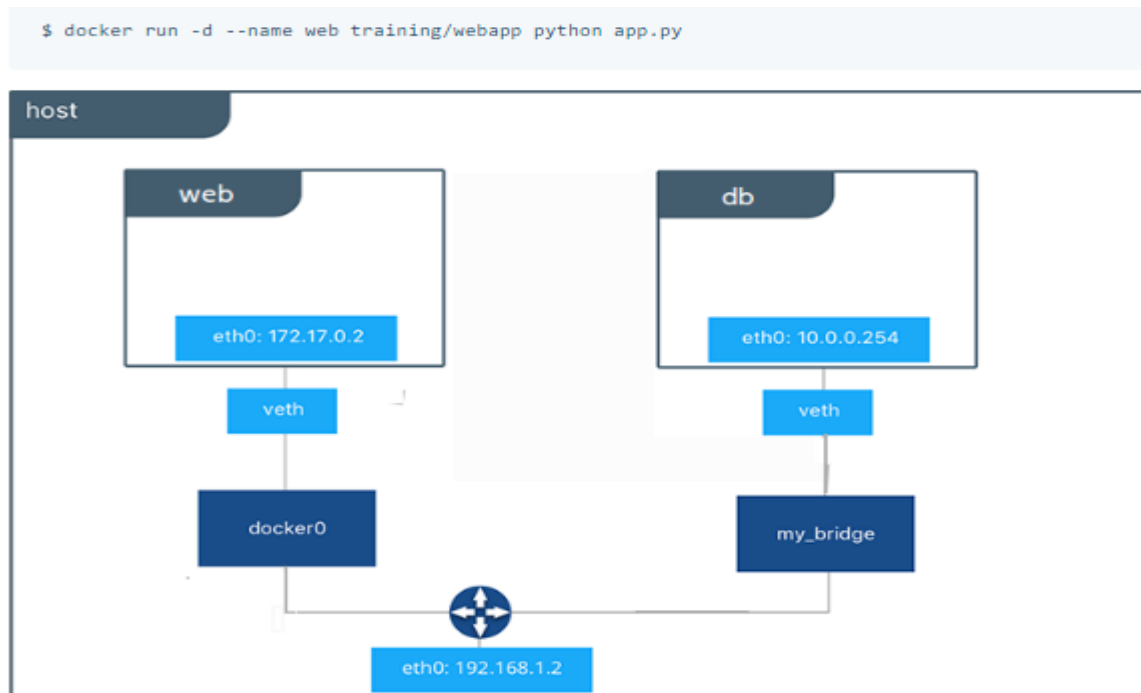
```
$ docker inspect --format='{{json .NetworkSettings.Networks}}' db

{"my_bridge":{"NetworkID":"7d86d31b1478e7cca9ebed7e73aa0fdeec46c5ca29497431d3007d2d9e15ed99",
"EndpointID":"508b170d56b2ac9e4ef86694b0a76a22dd3df1983404f7321da5649645bf7043","Gateway":"10.0.0.1","IPAddress":"10.0.0.254","IPPre
```

I.3.2. Docker Networking

Maintenant, lancez votre application Web désormais familière. Cette fois, ne spécifiez pas de réseau.

```
$ docker run -d --name web training/webapp python app.py
```



I.3.2. Docker Networking

- Sur quel réseau votre application Web fonctionne-t-elle? Inspectez l'application pour vérifier qu'elle s'exécute dans le réseau de pont par défaut.

```
$ docker inspect --format='{{json .NetworkSettings.Networks}}' web

{"bridge":{"NetworkID":"7ea29fc1412292a2d7bba362f9253545fecdfa8ce9a6e37dd10ba8bee7129812",
"EndpointID":"508b170d56b2ac9e4ef86694b0a76a22dd3df1983404f7321da5649645bf7043","Gateway":"172.17.0.1","IPAddress":"10.0.0.2","IPPre
```

- Ensuite, obtenez l'adresse IP de votre site Web

```
$ docker inspect --format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' web

172.17.0.2
```

I.3.2. Docker Networking

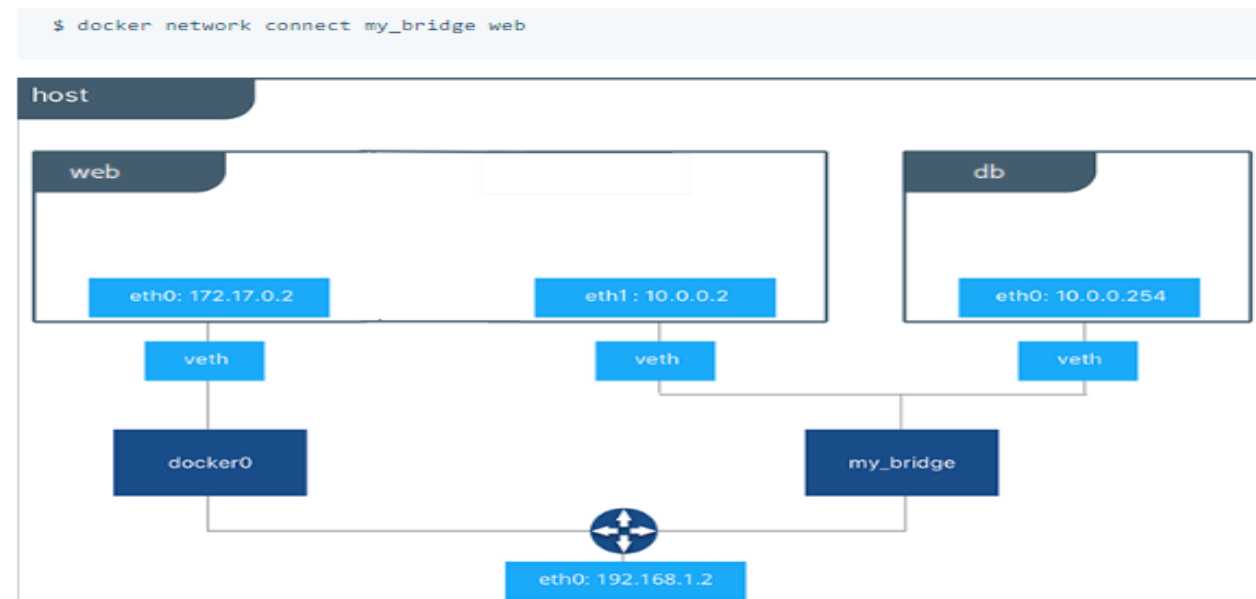
Maintenant, ouvrez un shell dans votre conteneur db en cours d'exécution:

```
$ docker container exec -it db bash

root@a205f0dd33b2:/# ping 172.17.0.2
ping 172.17.0.2
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
^C
--- 172.17.0.2 ping statistics ---
44 packets transmitted, 0 received, 100% packet loss, time 43185ms
```

I.3.2. Docker Networking

- Après un moment, utilisez CTRL-C pour mettre fin au ping et notez que le ping a échoué. C'est parce que les deux conteneurs fonctionnent sur des réseaux différents. Vous pouvez réparer ça. Ensuite, utilisez la commande exit pour fermer le conteneur.
- La mise en réseau Docker vous permet d'attacher un conteneur à autant de réseaux que vous le souhaitez. Vous pouvez également attacher un conteneur déjà en cours d'exécution. Allez-y et associez votre application Web en cours d'exécution à my_bridge.



I.3.2. Docker Networking

- Ouvrez à nouveau un shell dans l'application db et essayez la commande ping. Cette fois, utilisez simplement le nom du conteneur web plutôt que l'adresse IP.

```
$ docker container exec -it db bash

root@a205f0dd33b2:/# ping web
PING web (10.0.0.2) 56(84) bytes of data.
64 bytes from web (10.0.0.2): icmp_seq=1 ttl=64 time=0.095 ms
64 bytes from web (10.0.0.2): icmp_seq=2 ttl=64 time=0.060 ms
64 bytes from web (10.0.0.2): icmp_seq=3 ttl=64 time=0.066 ms
^C
--- web ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.060/0.073/0.095/0.018 ms
```

- Le ping montre qu'il contacte une adresse IP différente, l'adresse sur le my_bridge qui est différente de son adresse sur le réseau de pont.

I.3.2. Docker Storage drivers and file system

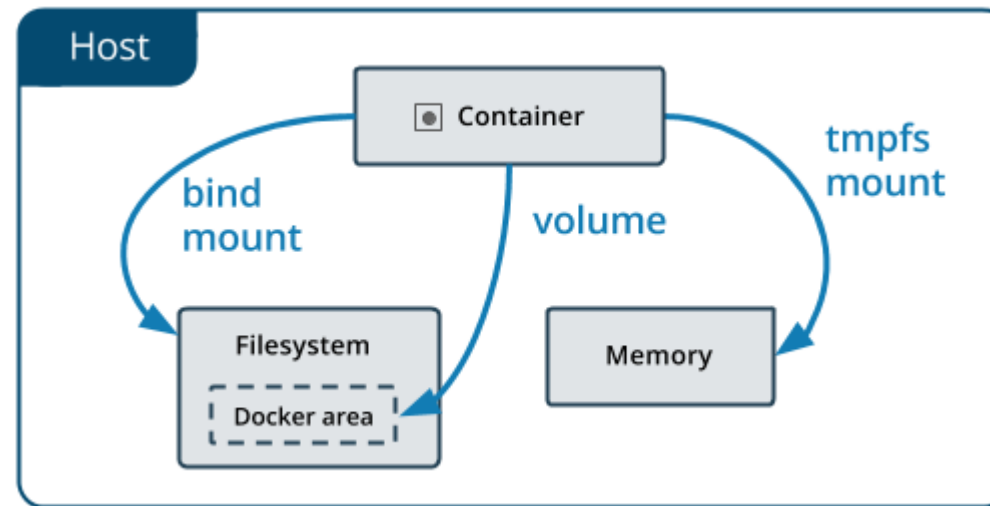
Par défaut, tous les fichiers créés à l'intérieur d'un conteneur sont stockés sur une couche de conteneur écrivable. Cela signifie que:

- Les données ne sont pas conservées lorsque ce conteneur n'existe plus et il peut être difficile d'extraire les données du conteneur si un autre processus en a besoin.
- La couche écrivable d'un conteneur est étroitement couplée à la machine hôte sur laquelle le conteneur s'exécute. Vous ne pouvez pas facilement déplacer les données ailleurs.
- L'écriture dans la couche écrivable d'un conteneur nécessite un pilote de stockage pour gérer le système de fichiers. Le pilote de stockage fournit un système de fichiers union, utilisant le noyau Linux. Cette abstraction supplémentaire réduit les performances par rapport à l'utilisation de volumes de données, qui écrivent directement sur le système de fichiers hôte.

Docker propose deux options permettant aux conteneurs de stocker des fichiers sur la machine hôte, afin que les fichiers soient conservés même après l'arrêt du conteneur: les volumes et les montages de liaison. Si vous exécutez Docker sous Linux, vous pouvez également utiliser un montage tmpfs. Si vous exécutez Docker sous Windows, vous pouvez également utiliser un canal nommé pipe.

I.3.2. Docker Storage drivers and file system

- Quel que soit le type de montage que vous choisissiez d'utiliser, les données se ressemblent depuis l'intérieur du conteneur. Il est exposé sous forme de répertoire ou de fichier individuel dans le système de fichiers du conteneur.
- Un moyen simple de visualiser la différence entre les volumes, les montages de liaison et les montages tmpfs consiste à réfléchir à l'emplacement des données sur l'hôte Docker.



I.3.2. Docker Storage drivers and file system

- Les volumes sont stockés dans une partie du système de fichiers hôte qui est géré par Docker (/ var / lib / docker / volumes / sous Linux). Les processus non-Docker ne doivent pas modifier cette partie du système de fichiers. Les volumes sont le meilleur moyen de conserver les données dans Docker.
- Les montages de liaison peuvent être stockés n'importe où sur le système hôte. Il peut même s'agir de fichiers ou de répertoires système importants. Les processus non-Docker sur l'hôte Docker ou un conteneur Docker peuvent les modifier à tout moment.
- Les montages tmpfs sont stockés uniquement dans la mémoire du système hôte et ne sont jamais écrits sur le système de fichiers du système hôte.

I.3.2. Docker Storage drivers and file system- Exemple montage volume

```
$ docker volume create my-vol
```

```
$ docker volume ls
```

```
$ docker volume inspect my-vol
```

```
$ docker volume rm my-vol
```

```
$ docker run -d \
```

```
--name devtest \
```

```
-v myvol2:/app \
```

```
nginx:latest
```

```
$ docker inspect devtest
```

```
$ docker container stop devtest
```

```
$ docker container rm devtest
```

```
$ docker volume rm myvol2
```

I.3.2. Docker Storage drivers and file system- Exemple Backup/restauration

- Les volumes sont utiles pour les sauvegardes, les restaurations et les migrations. Utilisez l'indicateur `--volumes-from` pour créer un nouveau conteneur qui monte ce volume. Sauvegarder un conteneur : Par exemple, créez un nouveau conteneur nommé `dbstore`:

```
$ docker run -v /dbdata --name dbstore ubuntu /bin/bash
```

- Ensuite, dans la commande suivante, : Lancer un nouveau conteneur et monter le volume à partir du conteneur `dbstore`. Monter un répertoire hôte local en tant que `/ backup`. Passez une commande qui compresse le contenu du volume `dbdata` dans un fichier `backup.tar` dans notre répertoire `/ backup`.

```
$ docker run --rm --volumes-from dbstore -v $(pwd):/backup ubuntu tar cvf /backup/backup.tar /dbdata
```

- Lorsque la commande se termine et que le conteneur s'arrête, il nous reste une sauvegarde de notre volume `dbdata`. Restaurer le conteneur à partir de la sauvegarde : Avec la sauvegarde que vous venez de créer, vous pouvez la restaurer dans le même conteneur, ou dans un autre que vous avez effectué ailleurs. Par exemple, créez un nouveau conteneur nommé `dbstore2`:

```
$ docker run -v /dbdata --name dbstore2 ubuntu /bin/bash
```

- Ensuite, décompressez le fichier de sauvegarde dans le volume de données du nouveau conteneur:

```
$ docker run --rm --volumes-from dbstore2 -v $(pwd):/backup ubuntu bash -c "cd /dbdata && tar xvf /backup/backup.tar --strip 1"
```

I.3.2. Sécurité

Il y a quatre domaines principaux à prendre en compte lors de l'examen de la sécurité Docker:

- La sécurité intrinsèque du noyau et sa prise en charge des espaces de noms et des groupes de contrôle;
- La surface d'attaque du démon Docker lui-même;
- Des failles dans le profil de configuration du conteneur, soit par défaut, soit lors de la personnalisation par les utilisateurs.
- Les fonctionnalités de sécurité «renforcées» du noyau et leur interaction avec les conteneurs.

Kernel namespaces, Control groups, Docker daemon attack surface, Linux kernel capabilities

Docker Content Trust Signature Verification, Other kernel security features

I.4.Docker Compose

Docker Compose est un outil permettant de définir et d'exécuter des applications Docker multi-conteneurs. Avec Compose, vous utilisez un fichier YAML pour configurer les services de votre application. Ensuite, avec une seule commande, vous créez et démarrez tous les services à partir de votre configuration.

Compose fonctionne dans tous les environnements: production, staging, développement, test, ainsi que les workflows CI. Vous pouvez en savoir plus sur chaque cas dans les cas d'utilisation courants.

L'utilisation de Compose est essentiellement un processus en trois étapes:

- Définissez l'environnement de votre application avec un Dockerfile afin qu'il puisse être reproduit n'importe où.
- Définissez les services qui composent votre application dans `docker-compose.yml` afin qu'ils puissent être exécutés ensemble dans un environnement isolé.
- Exécutez `docker-compose up` et Compose starts et exécuter toute votre application.

I.4. Docker Compose

Un docker-compose.yml ressemble à ceci:

```
version: "3.8"
services:
  web:
    build: .
    ports:
      - "5000:5000"
    volumes:
      - ./code
      - logvolume01:/var/log
    links:
      - redis
  redis:
    image: redis
volumes:
  logvolume01: {}
```

I.4. Docker Compose

Compose propose des commandes pour gérer l'ensemble du cycle de vie de votre application:

- Démarrer, arrêter et reconstruire les services
- Afficher l'état des services en cours d'exécution
- Diffuser la sortie du journal des services en cours d'exécution
- Exécuter une commande ponctuelle sur un service

I.4. Docker Compose

Les fonctionnalités de Compose qui le rendent efficace sont:

- Plusieurs environnements isolés sur un seul hôte
- Préserver les données de volume lors de la création des conteneurs
- Recréez uniquement les conteneurs qui ont changé
- Variables et déplacement d'une composition entre environnements

I.4. Docker Compose

Compose peut être utilisé de différentes manières. Certains cas d'utilisation courants sont décrits ci-dessous.

- Environnements de développement
- Environnements de test automatisés
- Déploiements d'hôte unique

I.4.Docker Compose-Etapes de configuration

Installaion Docker compose-Power shell

```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12  
Invoke-WebRequest "https://github.com/docker/compose/releases/download/1.27.4/docker-compose-Windows-x86_64.exe" -UseBasicParsing -OutFile $Env:ProgramFiles\ Docker\docker-compose.exe
```

Installaion Docker compose-Linux ou Chocolatey

```
apt-get install docker-compose ou choco install docker-compose
```

Tester l’installation

```
docker-compose --version  
docker-compose version 1.27.4, build 01110ad01
```

Premiers pas avec Docker Compose

- Étape 1: Configuration
- Étape 2: Créer Dockerfile
- Étape 3: Définir les services dans Compose file
- Étape 4: Créez et exécutez votre application avec Compose
- Étape 5: Modifiez le fichier de composition pour ajouter un montage de liaison
- Étape 6: Reconstruisez et exécutez l'application avec Compose
- Étape 7: mettre à jour l'application
- Étape 8: Expérimentez avec d'autres commandes

Commandes essentielles
docker-compose up [-d]
docker-compose down
docker-compose ps
docker-compose events
docker-compose logs [-f]

I.4.Docker Compose-Etapes de configuration

Installaion Docker compose-Power shell

```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12  
Invoke-WebRequest "https://github.com/docker/compose/releases/download/1.27.4/docker-compose-Windows-x86_64.exe" -UseBasicParsing -OutFile $Env:ProgramFiles\Docker\docker-compose.exe
```

Installaion Docker compose-Linux ou Chocolatey

```
apt-get install docker-compose ou choco install docker-compose
```

Tester l’installation

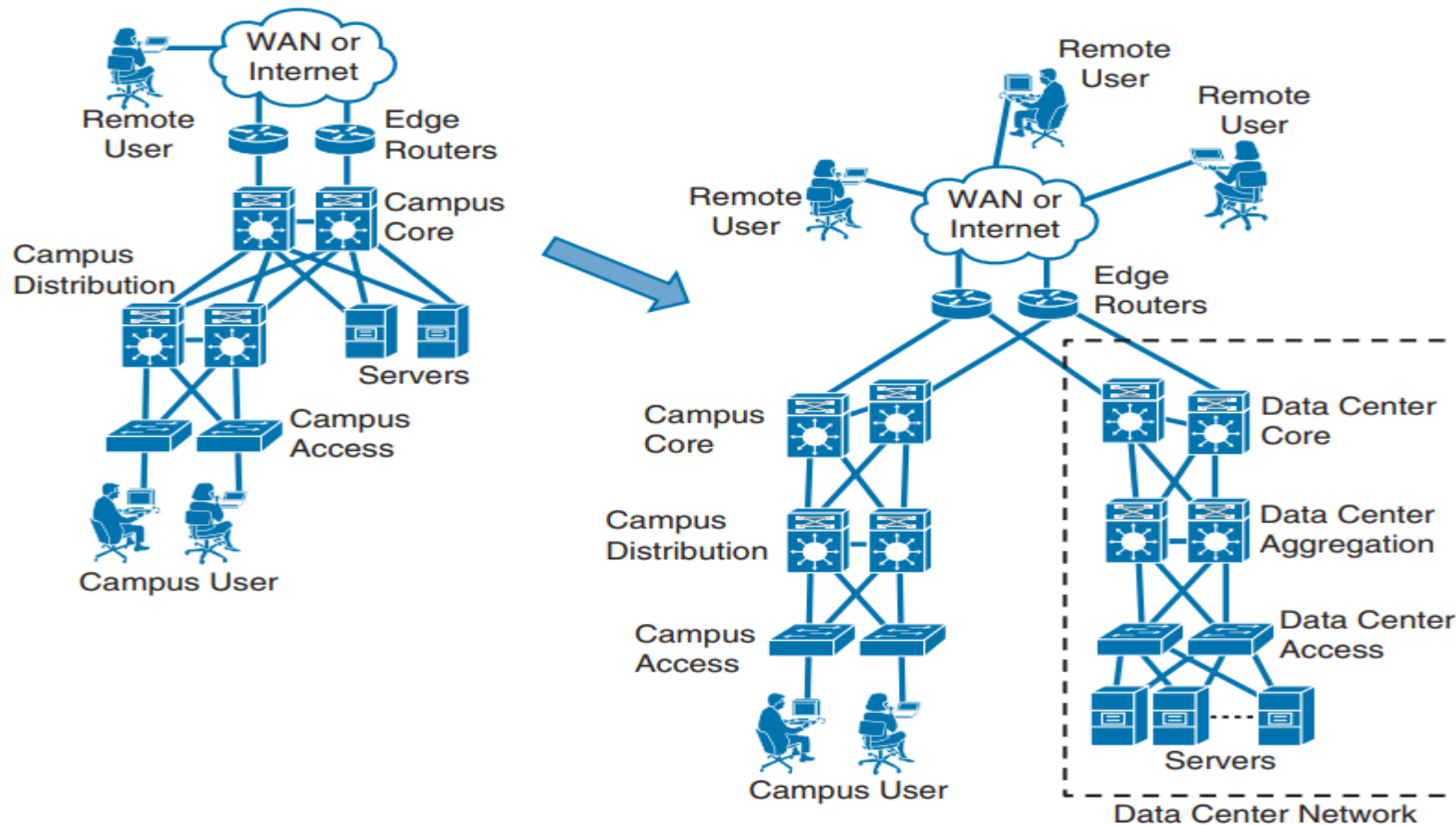
```
docker-compose --version  
docker-compose version 1.27.4, build 01110ad01
```

Premiers pas avec Docker Compose

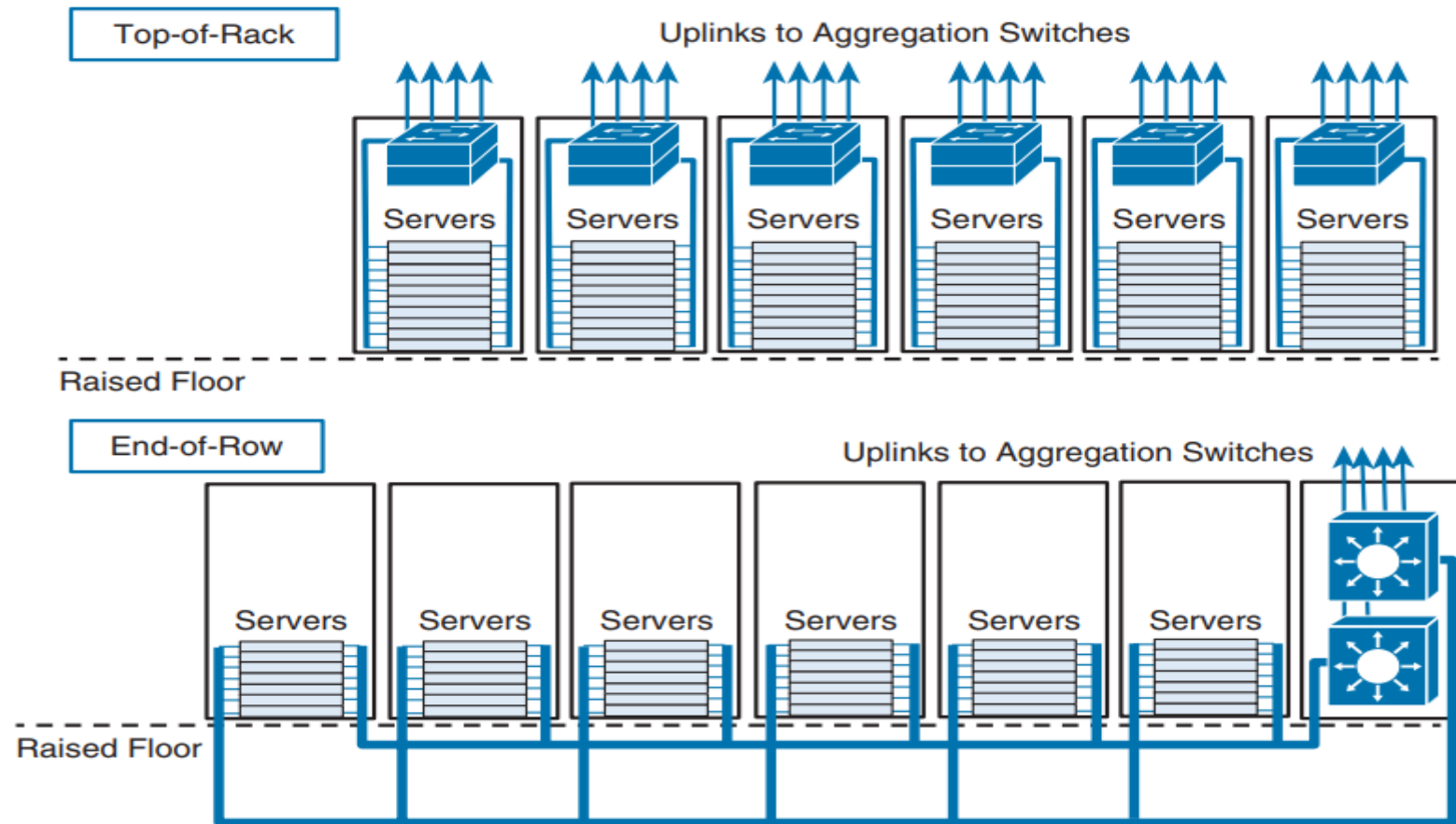
- Étape 1: Configuration
- Étape 2: Créer Dockerfile
- Étape 3: Définir les services dans Compose file
- Étape 4: Créez et exécutez votre application avec Compose
- Étape 5: Modifiez le fichier de composition pour ajouter un montage de liaison
- Étape 6: Reconstruisez et exécutez l'application avec Compose
- Étape 7: mettre à jour l'application
- Étape 8: Expérimentez avec d'autres commandes

Commandes essentielles
docker-compose up [-d]
docker-compose down
docker-compose ps
docker-compose events
docker-compose logs [-f]

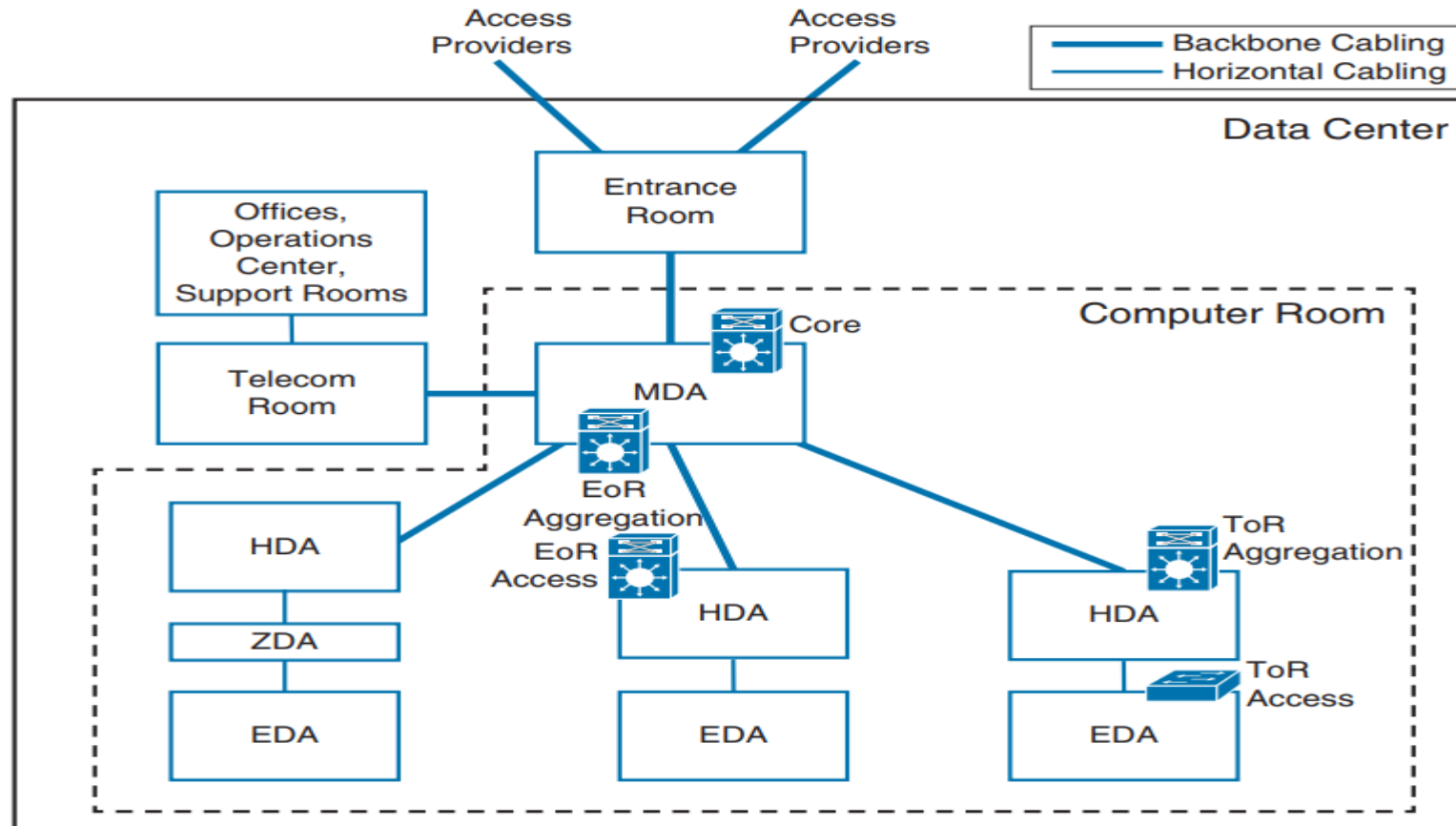
I.5. Evolution réseau des Datacenter



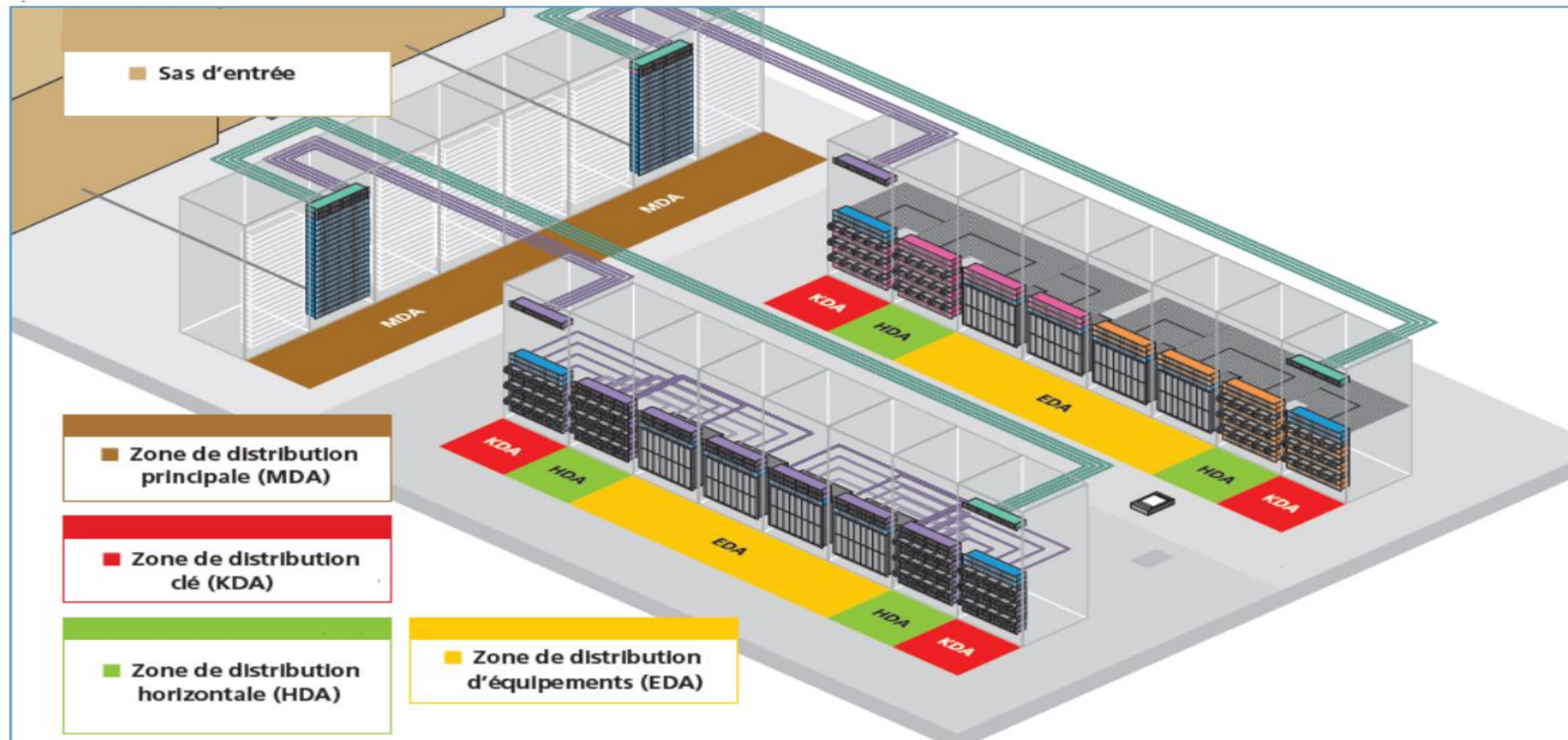
I.5. Evolution réseau des Datacenter



I.5. Evolution réseau des Datacenter



I.5. Evolution réseau des Datacenter



I.5. Bénéfices

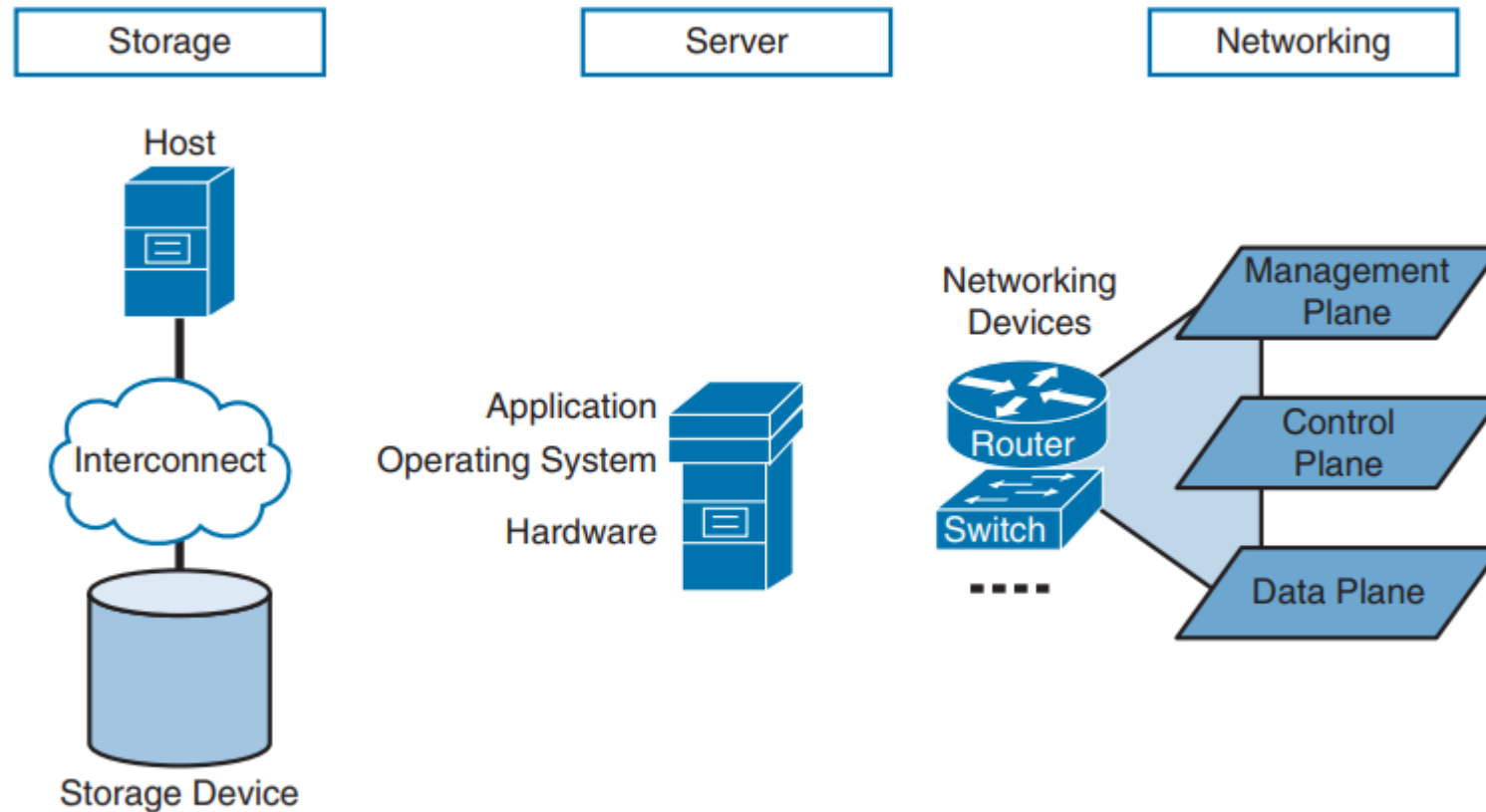
- Les technologies de virtualisation de réseau peuvent regrouper les avantages de deux conceptions contradictoires (top of rack, end of row) tout en minimisant leurs inconvénients.
- Certaines fonctionnalités et techniques qui résolvent différents problèmes avec des adaptations opérationnelles subtiles.

I.5. Virtualisation des réseaux

En général, la virtualisation de réseau est effectuée sur des périphériques. Néanmoins, ces techniques de virtualisation peuvent être réparties entre les plans réseau, qui représentent différents composants fonctionnels des périphériques réseau. Une technologie de virtualisation de réseau peut agréger, créer ou segmenter un (ou plusieurs) des plans suivants :

- Plan de données : gère le trafic qui traverse deux ou plusieurs interfaces d'un périphérique réseau (paquets de transit). Responsable de la majorité des flux de données sur ces appareils, il est également appelé plan de transfert.
- Plan de contrôle : traite le trafic dirigé vers le périphérique réseau lui-même et provenant d'autres périphériques. Il est illustré par les paquets de contrôle des protocoles de routage et contrôle le comportement du plan de données.
- Plan de gestion : exécute les composants destinés à la gestion des périphériques, tels que l'interface de ligne de commande (CLI) et le Simple Network Management Protocol (SNMP). Ce plan interagit généralement avec des logiciels tiers et est capable de modifier le comportement des plans de contrôle et de données.

I.5. Virtualisation des réseaux



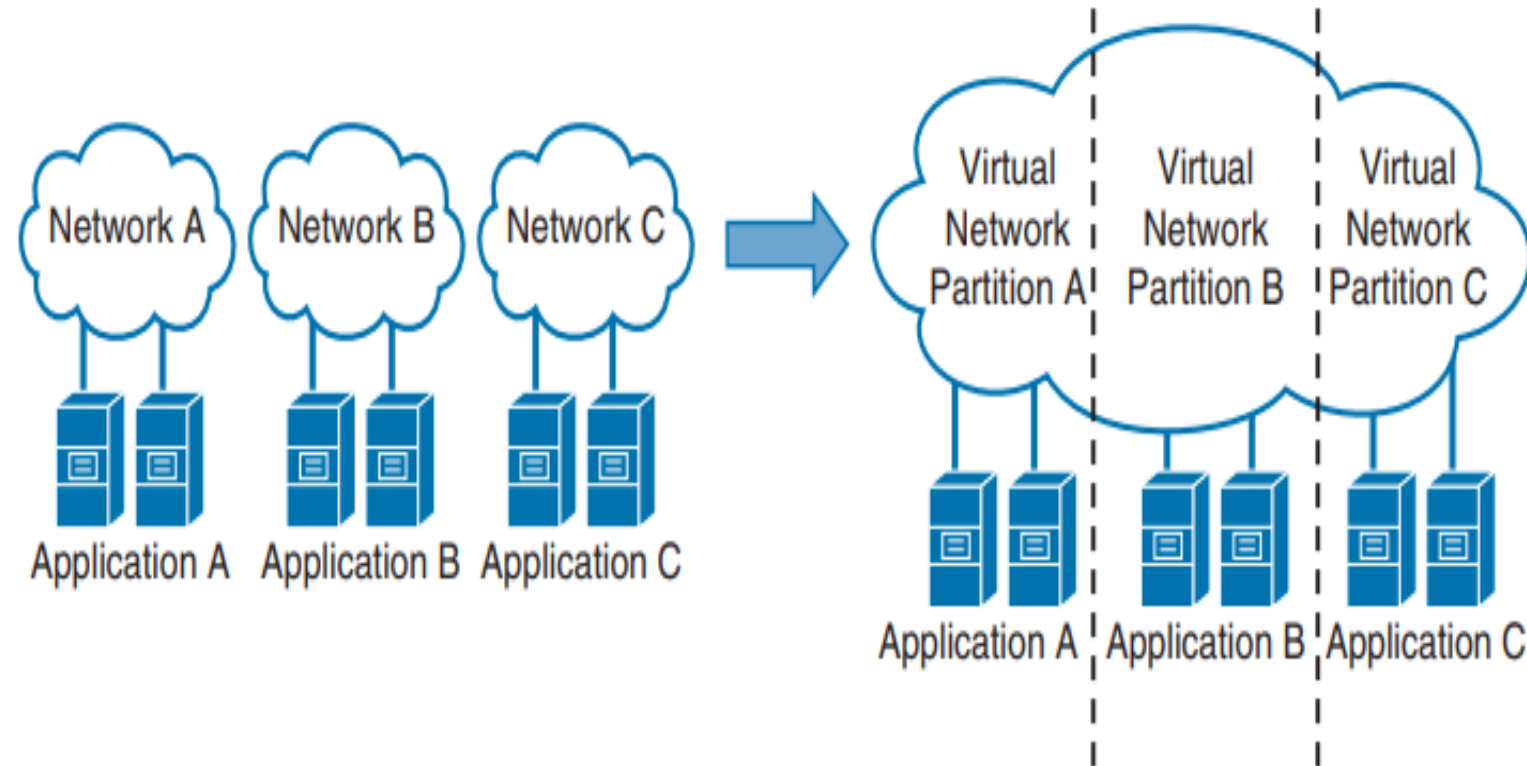
I.5.Exemple de classification

- Pooling
- Partitioning
- Abstraction

Characteristic	Virtual Memory	Mainframe Virtualization	HSRP
Emulation	Main memory	Mainframe	Router interface IP address
Type	Pooling	Partitioning	Abstraction
Subtype	Heterogeneous	Resource allocation	Address remapping
Scalability	Implementation dependent ¹	Hardware availability and software version dependent	255 groups, one active router, and one standby router per group ²
Technology area	Storage and server	Server	Networking
Subarea	Host (storage) hardware or operating system (server)	Operating system	Data plane
Advantages	Memory expansion and code reusability	Resource optimization and software compatibility	Default gateway high availability

I.5. Partitionnement logique du réseau

- VLAN
- VRF
- LOAD BALANCERS
- VDC
- ETC.



I.5. Simplification du réseau et équilibrage de la charge de trafic

- EtherChannel
- Virtual PortChannel (vPC)
- Layer 2 multipathing with FabricPath
- FHRP
- Etc.

I.5.Extension du réseau

La virtualisation du réseau peut également être utilisée pour connecter des réseaux de couche 2 sur des réseaux MPLS (Multiprotocol Label Switching) ou IP. Renonçant au déploiement de connexions fibre physiques coûteuses, les technologies de virtualisation telles que :

- Ethernet sur MPLS (EoMPLS)
- Virtual Private LAN Service (VPLS)
- Overlay Transport Virtualization (OTV)
- VXLAN, NVGRE, SST, GRE, LISP, NVO3, EVPN, IPSEC
- Etc.

Emulent les composants de réseau Ethernet pour permettre le clustering de serveurs et la migration de charges de travail entre les sites des centres de données. .

I.5.Management Consolidation and Cabling Optimization

- Les **Fabric Extenders** sont des périphériques réseau qui dépendent d'un commutateur parent pour fonctionner.
- Cependant, ils fonctionnent comme une carte de ligne distante d'un châssis distribué virtuel qui peut tirer parti des avantages des conceptions physiques d'accès en fin de rangée et en haut du rack.

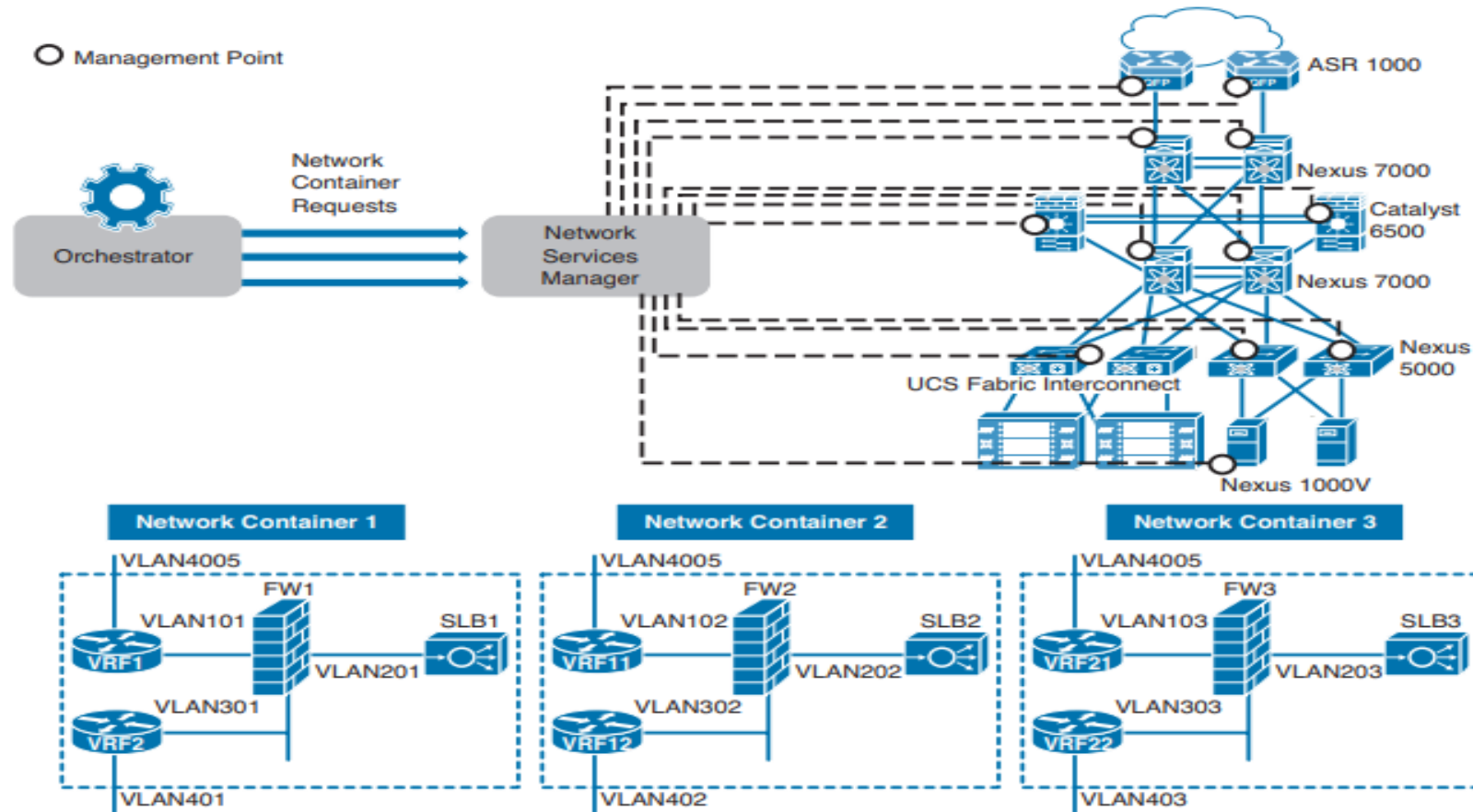
I.5. Virtualisation des fonctions réseaux

- La virtualisation des fonctions réseau (NFV) est un cadre architectural créé par l'Institut européen des normes de télécommunications (ETSI) qui définit des normes pour dissocier les fonctions réseau des appareils matériels propriétaires et les faire fonctionner sous forme de logiciel sur des serveurs x86 standard.
- Il définit également comment gérer et orchestrer les fonctions réseau.
- La fonction réseau (NF) fait référence à la fonction exécutée par un appareil physique, comme un pare-feu ou une fonction de routeur.

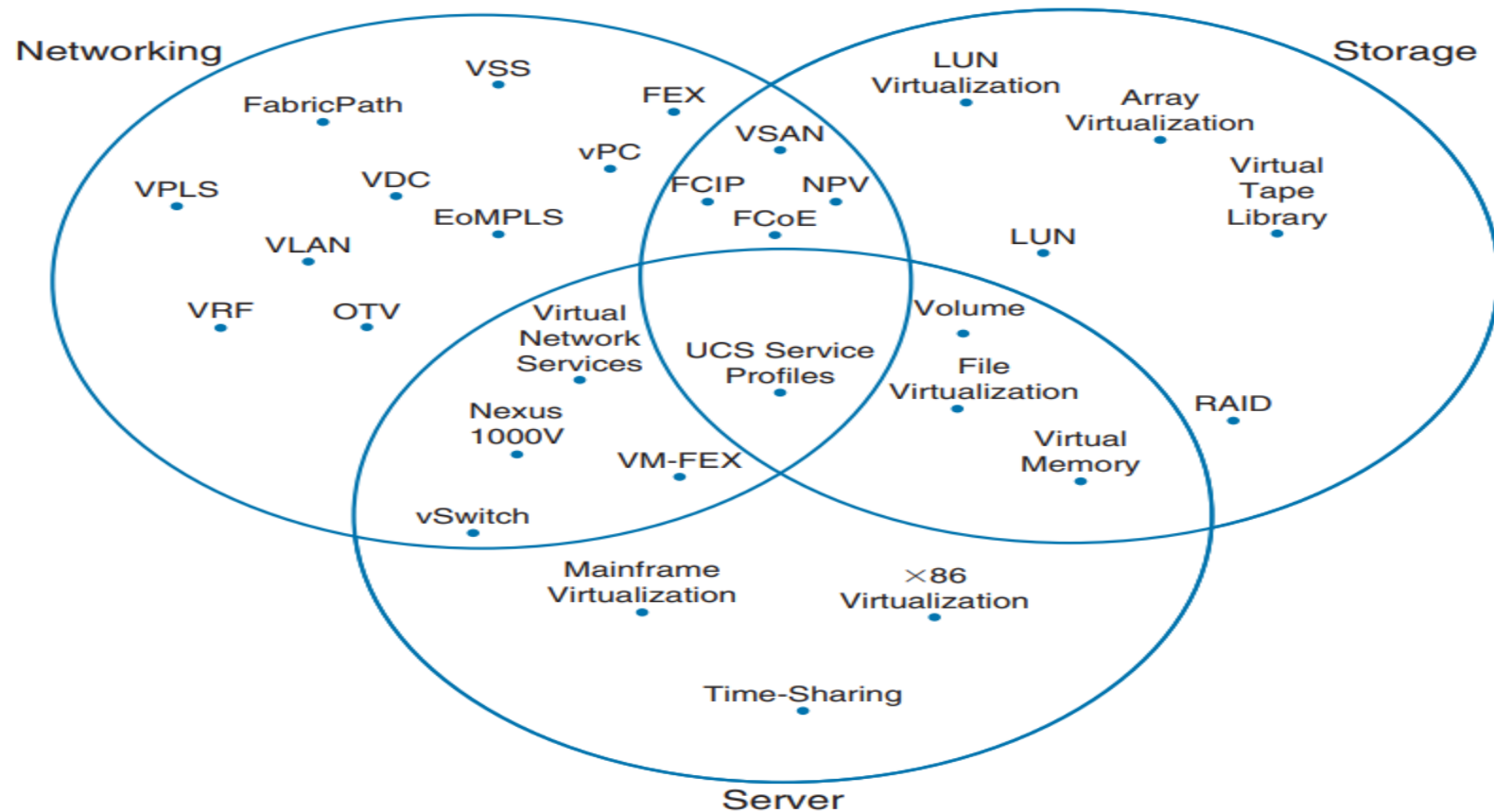
I.5. Virtualisation de bout en bout

- Automatiser pour les administrateurs la gestion et la livraison des services de bout en bout.
- Minimiser les interactions physiques pour gérer le pooling, partitioning, abstraction à partir d'un DC virtualisé comme fournisseur de service interne via des NSM.
- Offrant les éléments virtuels suivants à d'autres domaines de l'organisation, plusieurs clients ou environnements applicatifs (multi tenants).
- Partager une infrastructure physique commune en conservant des caractéristiques.
- Améliorer la gestion selon la notion de POD et les modèles de Datacenter.
- SDN c'est un NSM qui gère la partie réseau et assure aussi la partie DEVNET.
- DevOps assure l'automatisation de la configuration et les modifications du réseau virtuel à l'aide d'outils tels que Puppet, Ansible et Chef.
- SDWAN et SDACCESS.
- Le bout en bout avancé donne des self service sur un Datacenter virtualisé, cloud computing.

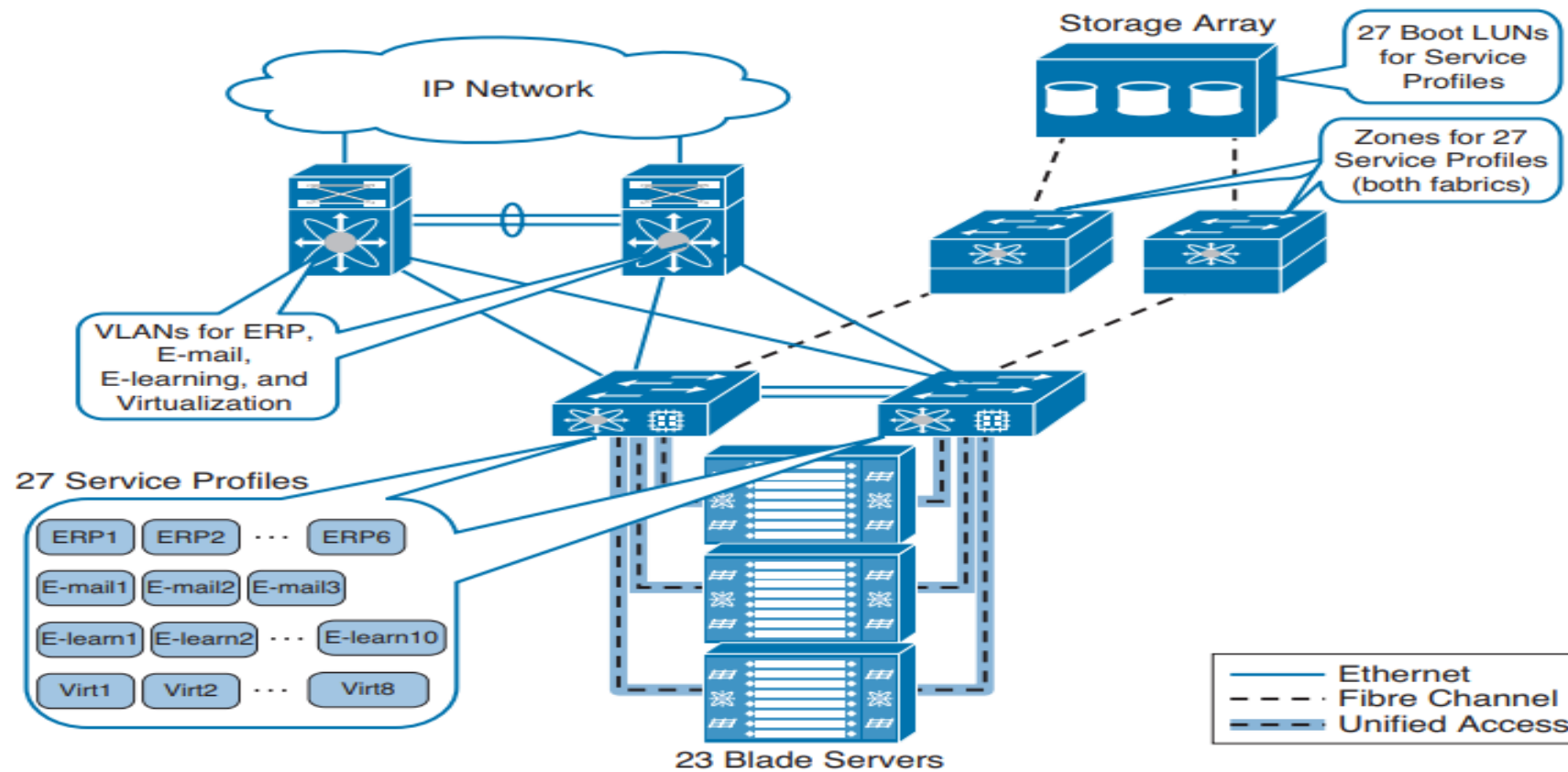
I.5. Virtualisation de bout en bout



I.5. Virtualisation des réseaux/stockage/serveur



I.5.UCS



I.5. Virtualisation Cloud

