

INFAL103-Essentiels de l'infrastructure du SI

y.snoussi@outlook.com

<https://github.com/Julia1205/cesi.git>

M. YASSINE SNOUSSI
Lead Architect
Cloud Architect
C# Enthusiast

INFAL103-Essentiels de l'infrastructure du SI

Détail du programme-plan de cours

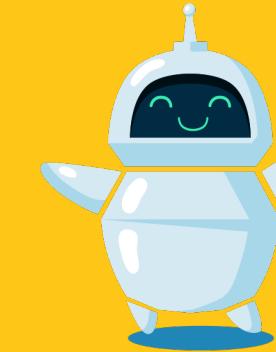
- Rappels sur les infrastructures VM/conteneurs, Réseau, Stockage, Ordonnanceurs
- **Définition et types d'approches**
 - Définition L'approche impérative et L'approche déclarative
 - Liens avec le Devops Avantages et inconvénients
- **Les solutions d'infra as code**
 - Découverte des différents outils d'infra as code Mise en place d'une infra as code (ex : terraform, puppet, ansible,...)

www.wooclap.com

JSGXAC



joystopper.io



BRINGING JOY, STOPPING BOTS

× × × ×



Introduction

Joystopper.io, où la joie rencontre la sécurité !

Chez Joystopper, notre mission est de révolutionner la manière dont nous interactissons avec le monde en ligne.

Notre objectif est de créer une expérience non seulement agréable pour les utilisateurs, mais également impénétrable pour les robots. Nous croyons en la création d'un espace où les utilisateurs authentiques peuvent profiter d'un parcours en ligne fluide sans l'interférence de robots malveillants.

01





Notre Objectif

Expérience Joyeuse : Nous visons à créer un environnement numérique où les utilisateurs peuvent s'engager joyeusement, à l'abri des perturbations causées par les robots automatisés.

Sécurité Robuste : Joystopper s'engage à fournir une sécurité de premier ordre contre les menaces des robots. Grâce à des technologies innovantes et à l'apprentissage automatique, nous détectons et contrecarrons les activités indésirables des robots.

Innovation dans le Captcha : Nos captchas ne sont pas simplement des défis ; ils sont conçus pour être agréables aux utilisateurs tout en constituant une barrière redoutable pour les robots automatisés. Dites adieu aux captchas fastidieux !



Nos user stories sont simples

03

En tant qu'Utilisateur, je veux naviguer sur des sites Web sans interruption de la part des robots pour profiter d'une expérience en ligne fluide et joyeuse.

En tant que Développeur, je veux des outils efficaces et des API de Joystopper pour les intégrer facilement dans mes applications Web et renforcer la sécurité de mes plates-formes en ligne.

En tant que Propriétaire d'Entreprise, je veux une solution fiable de blocage des robots qui garantit l'intégrité des interactions des utilisateurs sur mon site Web, favorisant la confiance et la satisfaction des utilisateurs.



Nous protégeons votre portail et vos APIs Contre :



Account Takeover



Scraping



Déni de Service



Piratage de Carte



Credential Stuffing



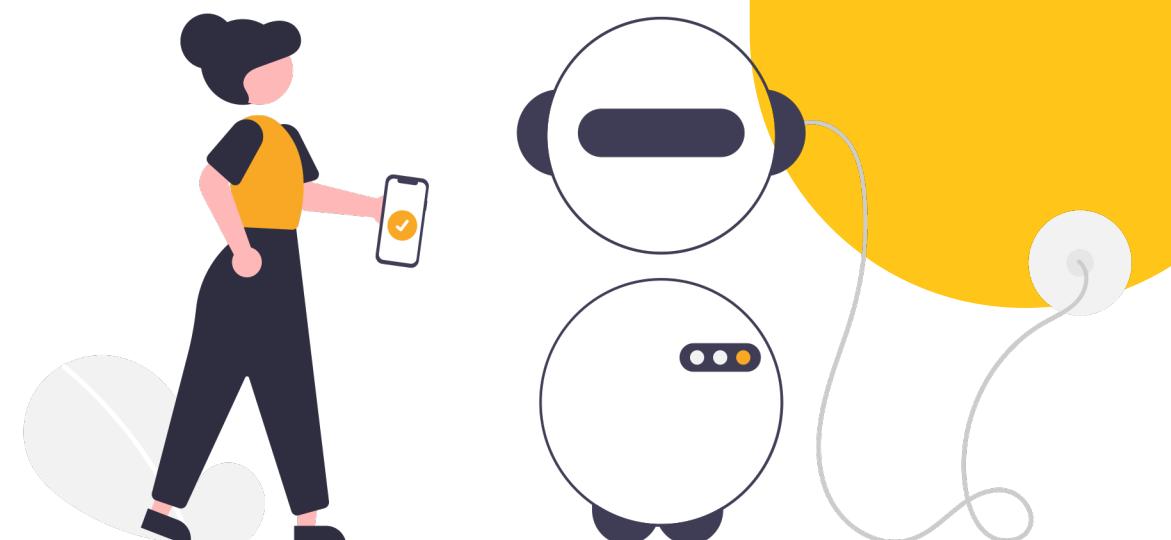
Server Overload



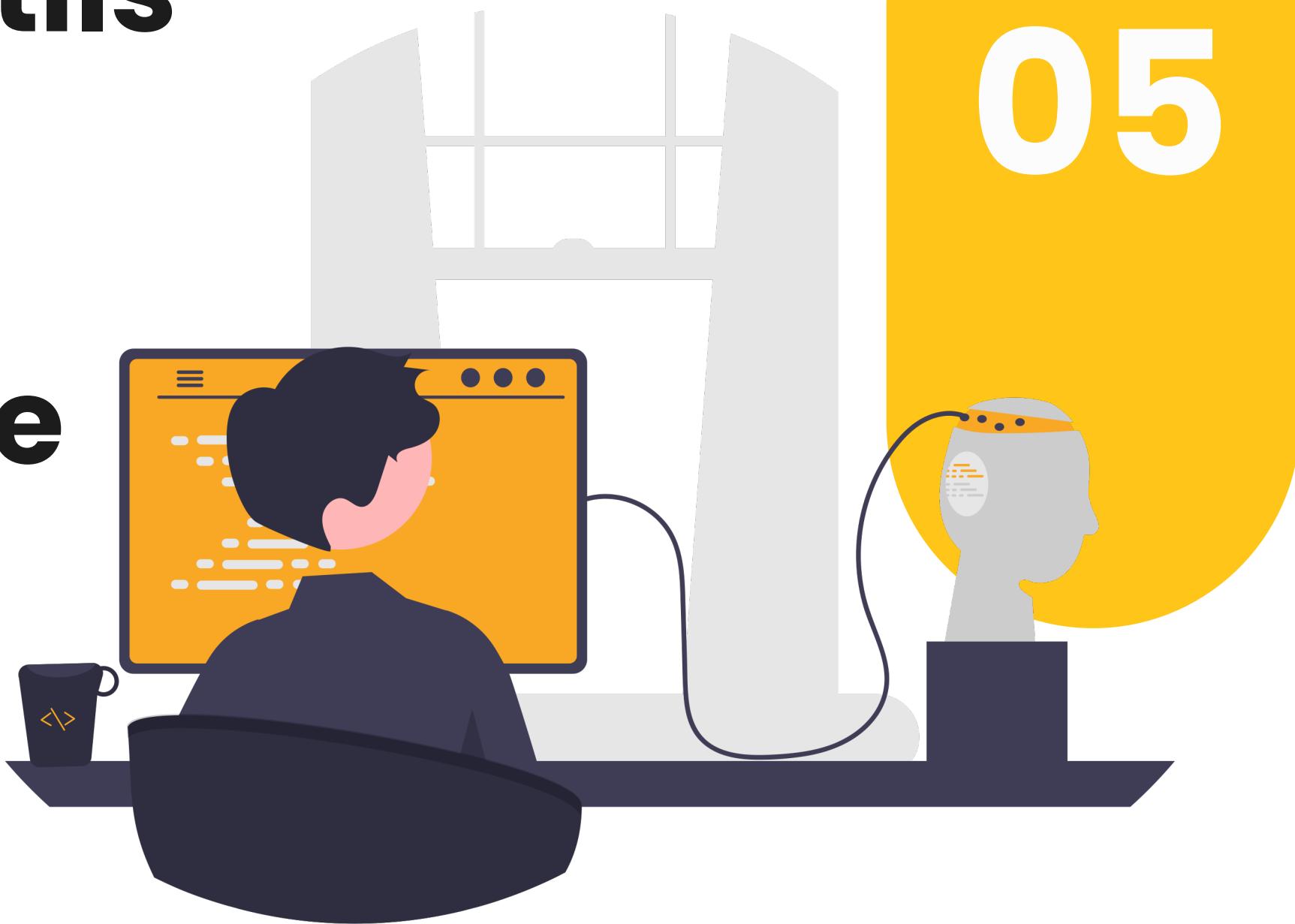
Création de Faux Comptes



Scanning de Vulnérabilités



**Tout ça grâce à nos outils
à la pointe de la
technologie basés sur
l'intelligence Artificielle**



05

Nos principes d'architecture

Séparation des Responsabilités (Separation of Concerns):

Diviser le système en modules ou composants distincts, chacun étant responsable d'une fonctionnalité ou d'un aspect spécifique.

Évolutivité (Scalability): Concevoir l'architecture de manière à ce qu'elle puisse évoluer facilement pour gérer une augmentation du nombre d'utilisateurs, de requêtes, ou de fonctionnalités.



Nos principes d'architecture

07

Sécurité par Conception (Security by Design): Intégrer des mécanismes de sécurité dès la conception pour protéger le système contre les menaces potentielles, en particulier dans le contexte de la détection des bots.

Réactivité (Responsiveness): Assurer une réponse rapide du système aux requêtes des utilisateurs, en minimisant les temps de latence.





08

Nos principes d'architecture

Extensibilité (Extensibility): Concevoir le système de manière à ce qu'il puisse être étendu facilement pour intégrer de nouvelles fonctionnalités sans perturber l'existant.

Modularité: Encourager la construction de modules indépendants et réutilisables, favorisant ainsi la maintenance et l'évolutivité du système.





09

Nos principes d'architecture

Utilisation de Technologies Éprouvées (Use of Proven Technologies): Opter pour des technologies et des bibliothèques bien établies et éprouvées pour assurer la stabilité et la fiabilité du système.





Nos principes d'architecture

10

Collecte Minimale de Données : Limiter la collecte de données aux informations strictement nécessaires pour la détection des bots.



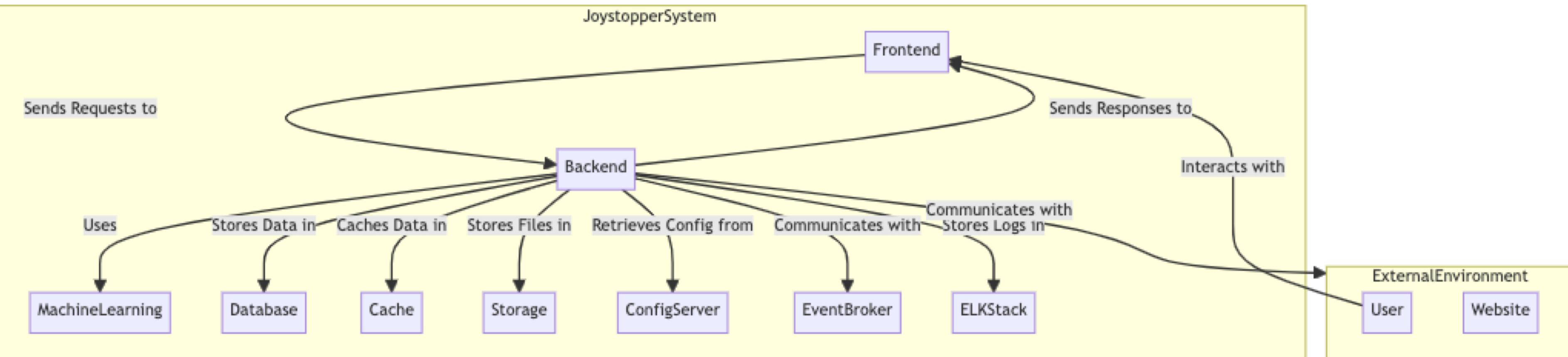


11

**Faites
nous
confiance**



Architecture Fonctionnelle

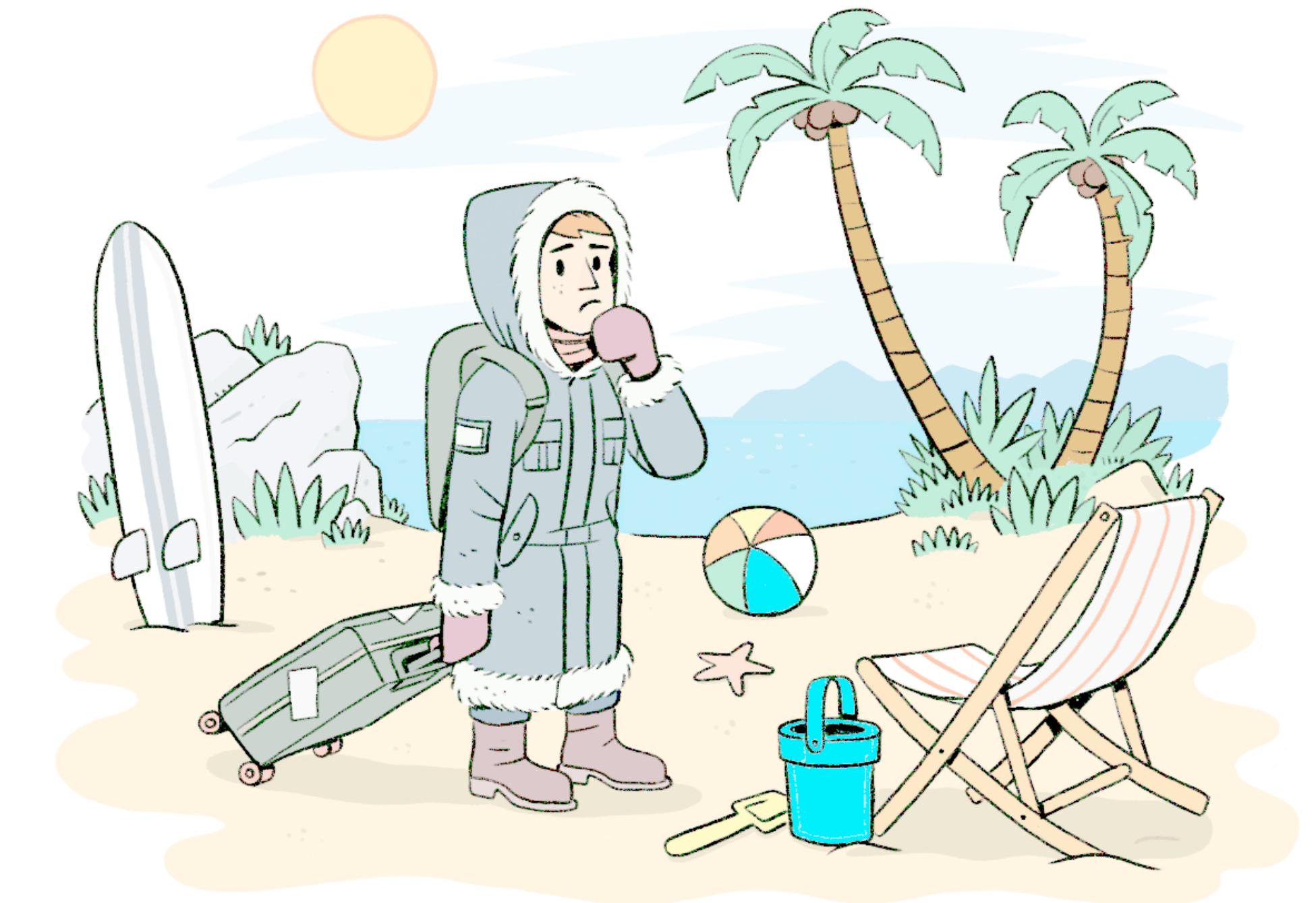


Nous visons 1 million d'utilisateur / jour

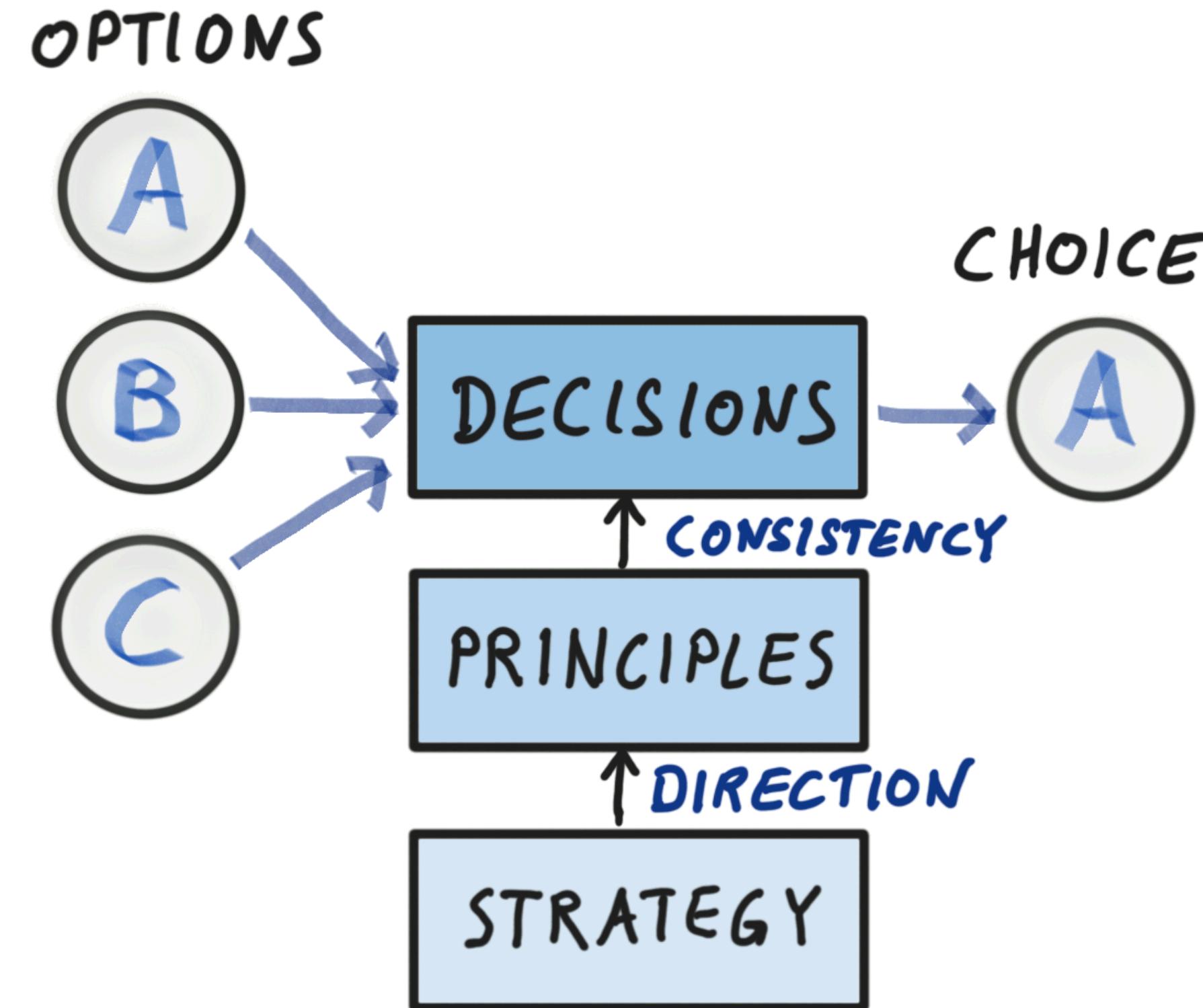
3Mrd Queries à analyser



Comment allons-nous répondre à cette demande?



Strategy Informs Decisions





Parlons infrastructure !!!





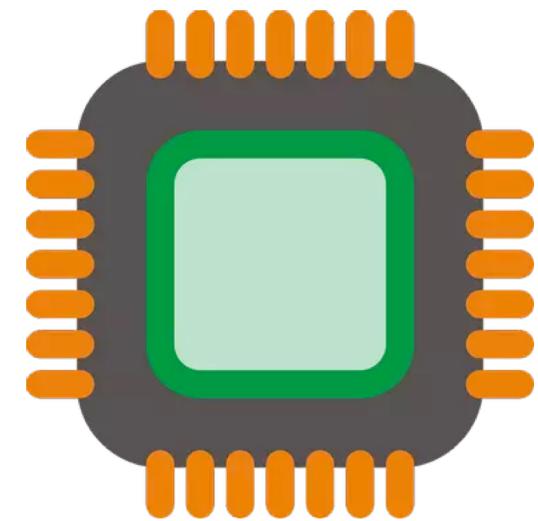
1

Network



2

CPU/ workloads



3

Storage





Network is king





Network

1. Dispositifs Actifs :

- **Routeurs** : Permettent de diriger le trafic entre différents réseaux.
- **Commutateurs (Switches)** : Connectent des appareils au sein d'un réseau local (LAN) en utilisant des adresses MAC.
- **Concentrateurs (Hubs)** : Anciennement utilisés, ils diffusent des données à tous les appareils connectés au réseau.
- **Ponts (Bridges)** : Connectent des segments de réseau et opèrent au niveau de la couche 2 (liaison de données) du modèle OSI.
- **Passerelles (Gateways)** : Connectent des réseaux avec des protocoles différents.





Network

2. Dispositifs Passifs :

Câbles : Les câbles de réseau (Ethernet, fibre optique, etc.) permettent la transmission des données entre les dispositifs.

Connecteurs et Prises : Utilisés pour connecter les câbles aux dispositifs actifs.





Network

3. Protocoles de communication:

Protocoles de Communication : Définissent les règles pour la communication entre les dispositifs (ex : TCP/IP).

Protocoles de Routage : Déterminent comment les données sont dirigées à travers le réseau.

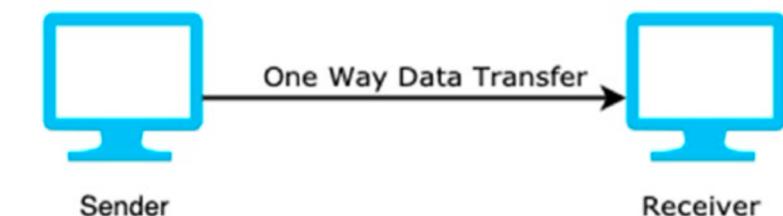
Protocoles de Sécurité : Gèrent la sécurité des données lors de la transmission.



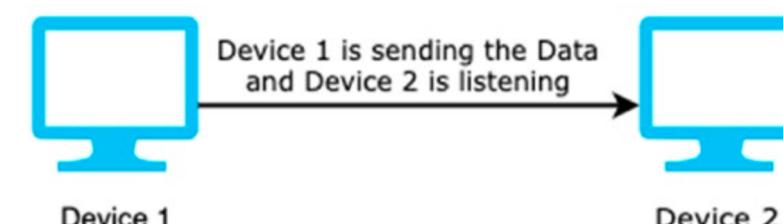
Network – Modes de communication

- Simplex
- Half Duplex
- Full Duplex

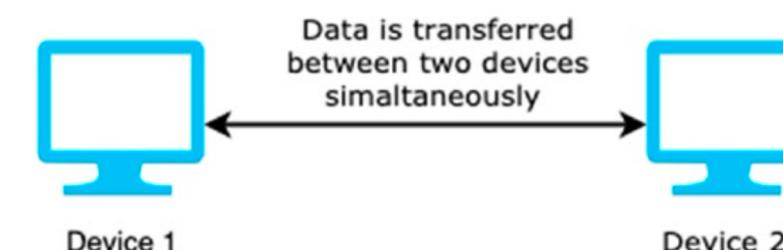
Simplex Mode of Transmission



Half Duplex Mode of Transmission



Full Duplex Mode of Transmission





Network

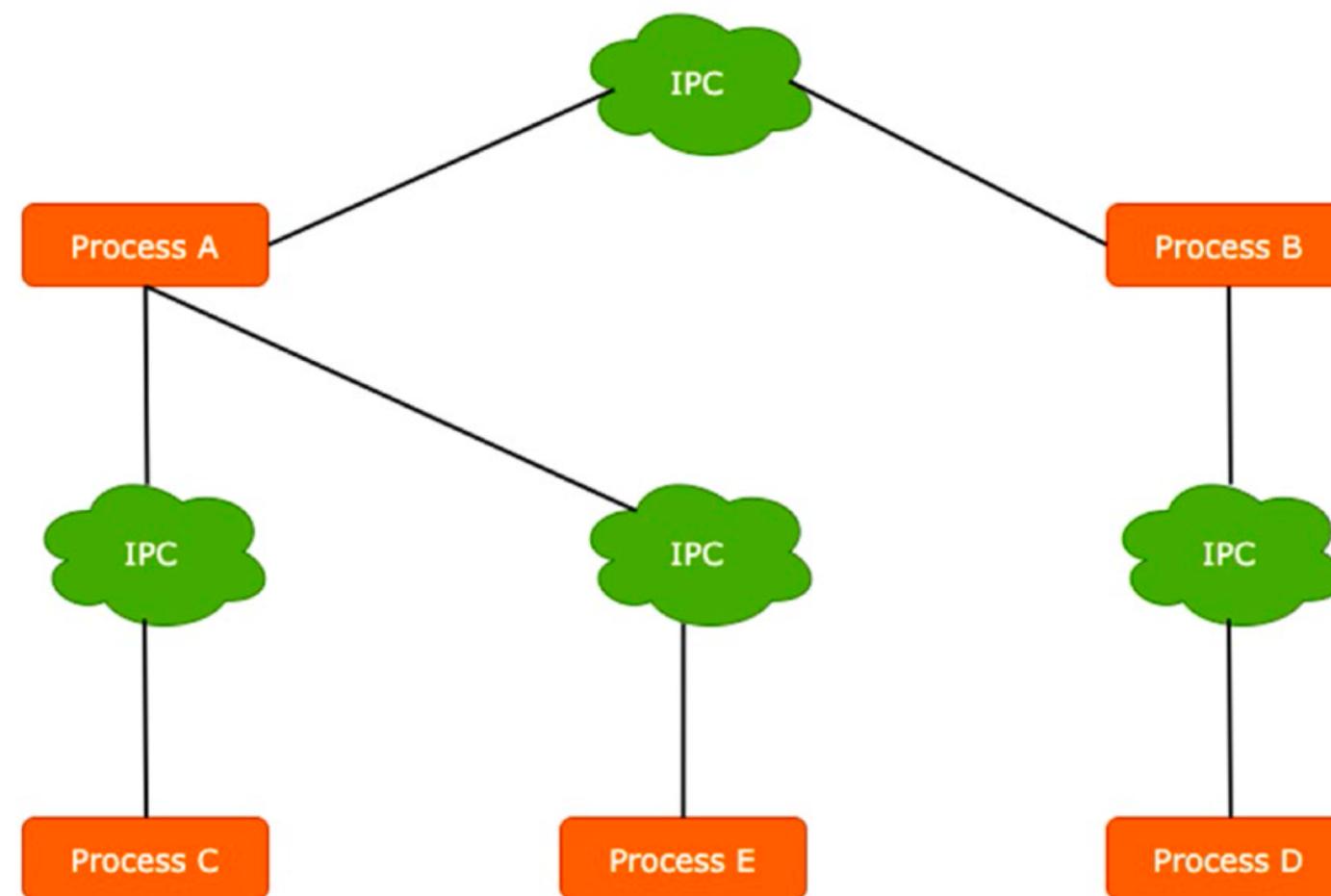
Inter-process communication (IPC)

La communication interprocessus (IPC) est un mécanisme qui permet aux processus de communiquer avec d'autres processus dans un système d'exploitation. Le processus peut être dans le **même système** ou dans un **système différent**.

L'IPC implique également la synchronisation des actions des processus et la gestion de l'activité de **partage de données**.



Network - IPC



Message Passing

Le passage de messages vous permet d'atteindre différents modes de communication.

- Communication entre différents threads au sein d'un processus.
- Communication entre différents processus s'exécutant sur la même machine hôte.
- Communication entre différents processus s'exécutant sur des machines différentes.



Network – Types IPC

- 1. Pipes**
- 2. Message Queue**
- 3. Semaphores**
- 4. Shared Memory**
- 5. Sockets**





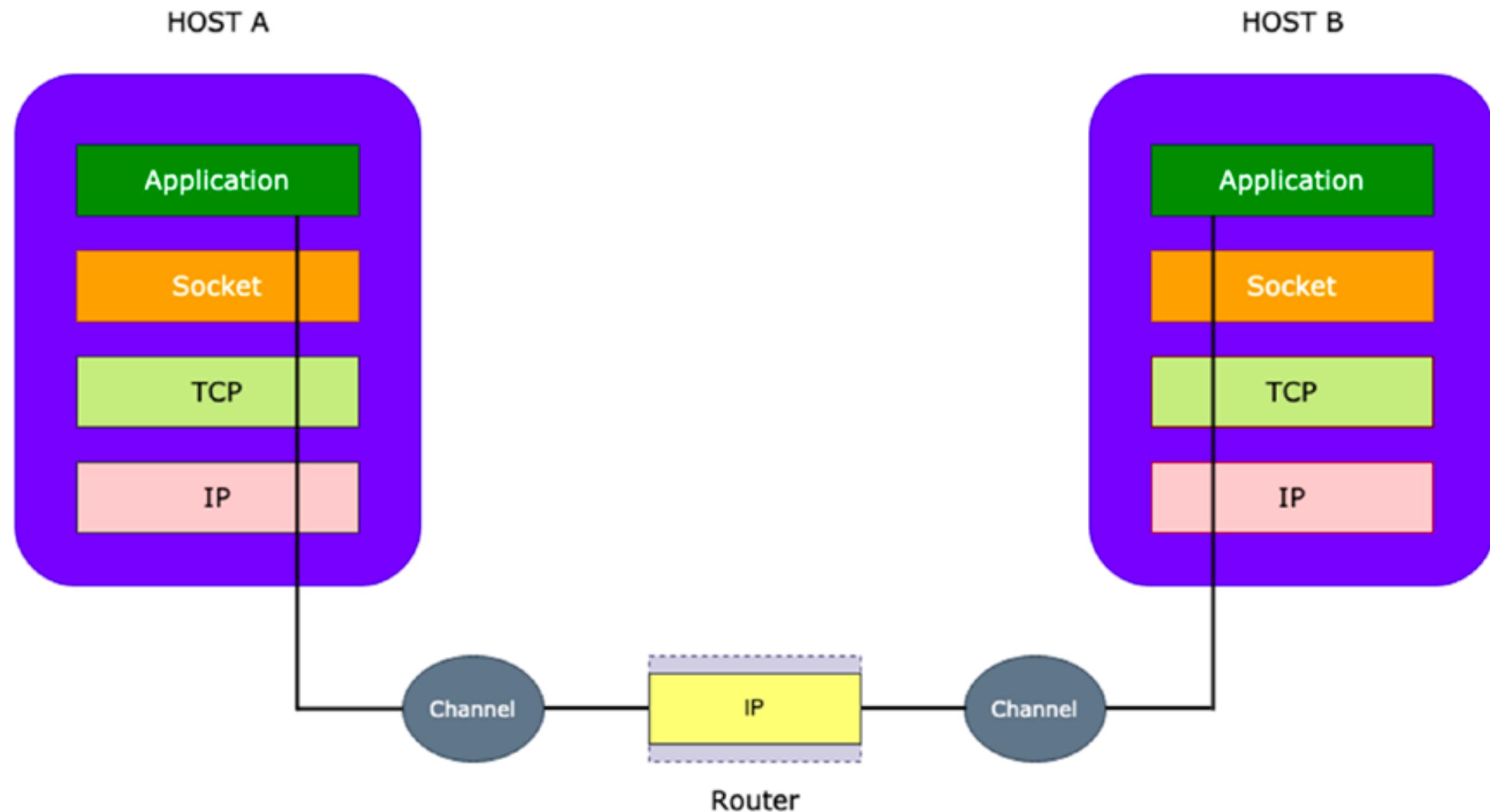
Network – Types IPC

- 1. Pipes**
- 2. Message Queue**
- 3. Semaphores**
- 4. Shared Memory**

5. Sockets



Network – Sockets or IPC over Network

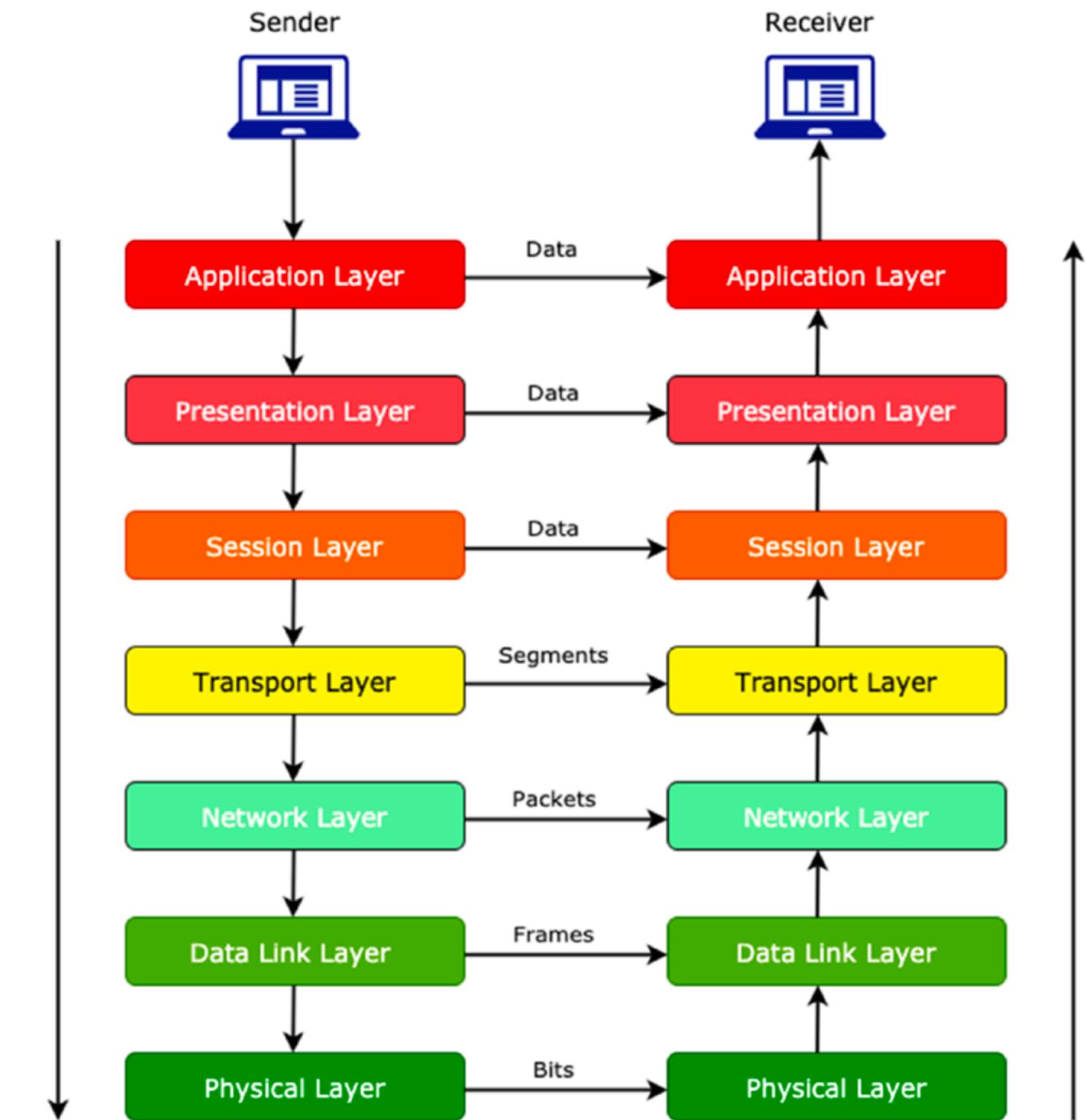




Network – Sockets or IPC over Network

Modèle OSI Open Systems Interconnection

1. C'est un modèle complexe et théorique qui n'a aucune mise en œuvre pratique.
2. L'adressage de la duplication de services (c'est-à-dire le contrôle d'erreur, le contrôle de flux et les données) est effectué à différentes couches.
3. Il est très difficile d'adapter des protocoles à ce modèle. Étant donné que c'est un modèle générique, c'est de votre responsabilité d'adapter le protocole
4. Les couches de session et de présentation ont moins de fonctionnalités que les autres couches.
5. Les couches ne peuvent pas fonctionner en parallèle car chaque couche doit attendre pour obtenir des données d'une autre couche.



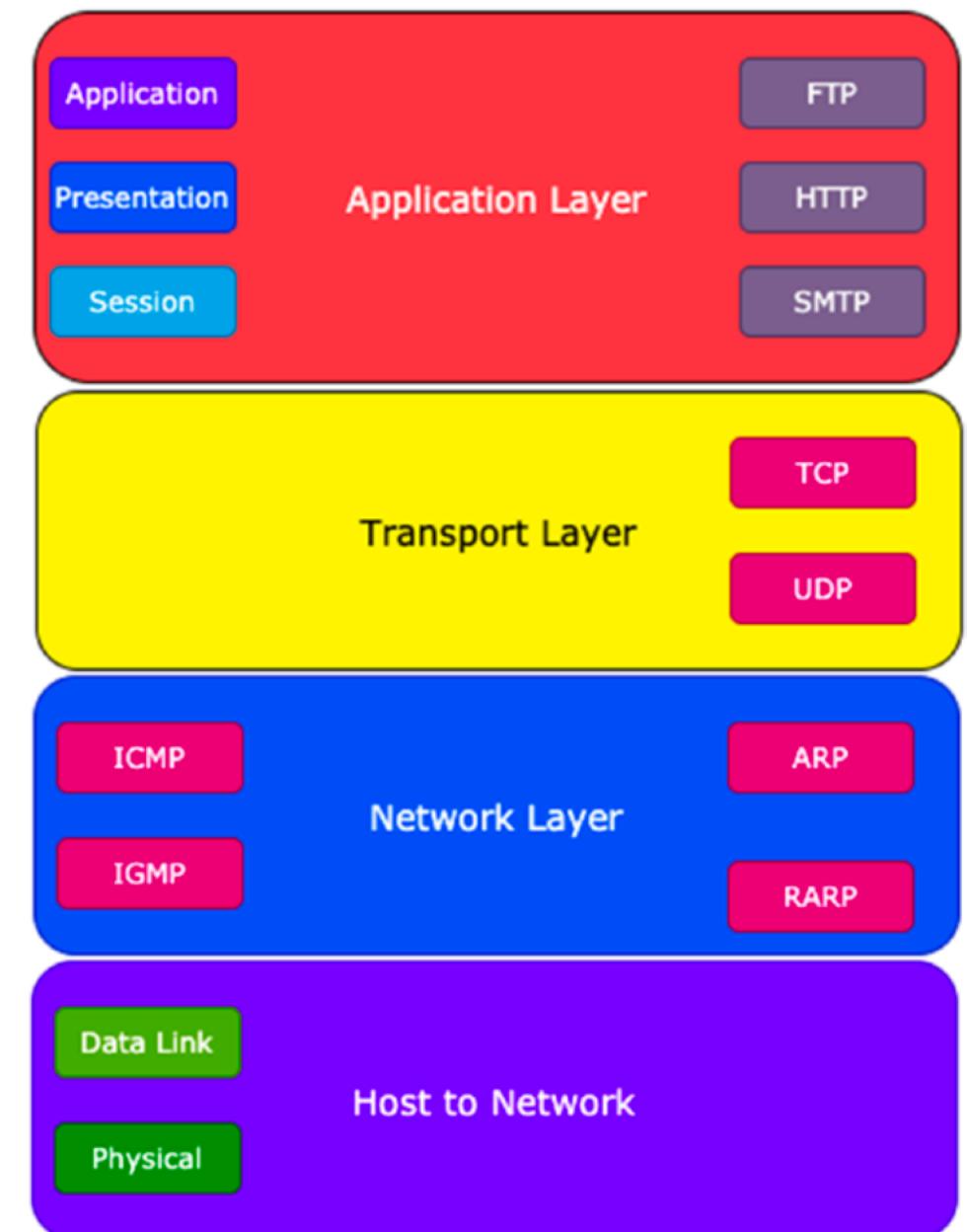


Network – Sockets or IPC over Network

Modèle TCP/IP

Transmission Control Protocol

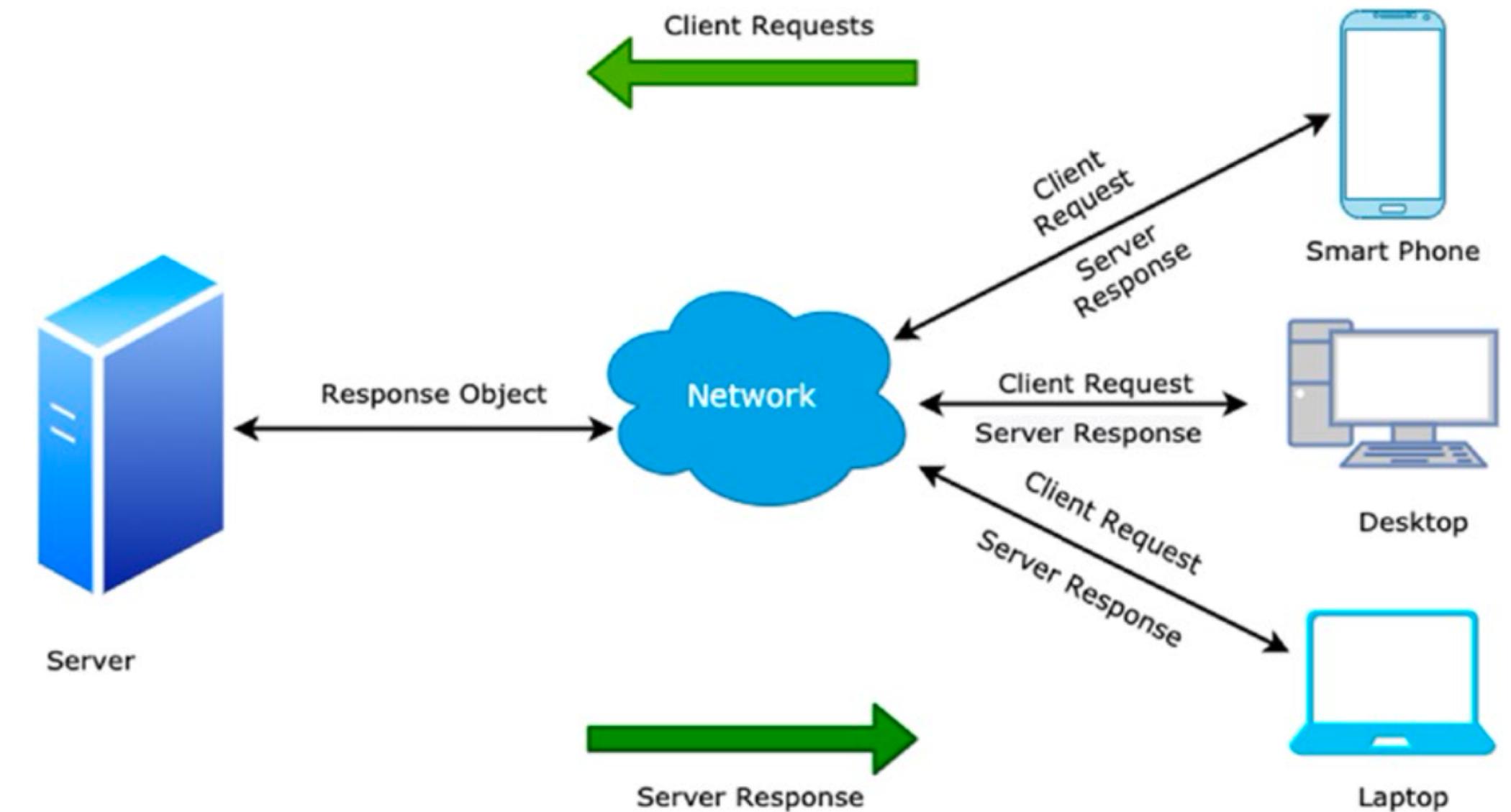
1. Ce n'est pas un protocole générique comme le modèle OSI.
2. Il est complexe à configurer et à utiliser.
3. Adopter de nouveaux protocoles et de nouvelles technologies est très difficile.
4. C'est un modèle de suite de protocoles en source ouverte.
5. Il établit une connexion entre deux systèmes.
6. Il ne dépend pas du système d'exploitation ou du matériel d'un appareil.
7. C'est une architecture évolutive pour la communication. • Il attribue une adresse IP unique à chaque appareil dans le réseau.
8. Il utilise des mécanismes de contrôle de flux, de contrôle d'erreur et de contrôle de congestion pour une meilleure transmission des données.
9. Il garantit la transmission des données sans duplication et offre un meilleur débit.



Network – Sockets or IPC over Network

Architecture client/serveur

À partir des
année 1980





CPU & Workloads





CPU & Workloads

Serveurs - Ordinateurs:

Serveurs de Fichiers : Stockent et fournissent des fichiers aux utilisateurs du réseau.

Serveurs d'Applications : Hébergent des applications qui peuvent être utilisées par les utilisateurs du réseau.

Serveurs de Messagerie : Gèrent les communications électroniques au sein du réseau.



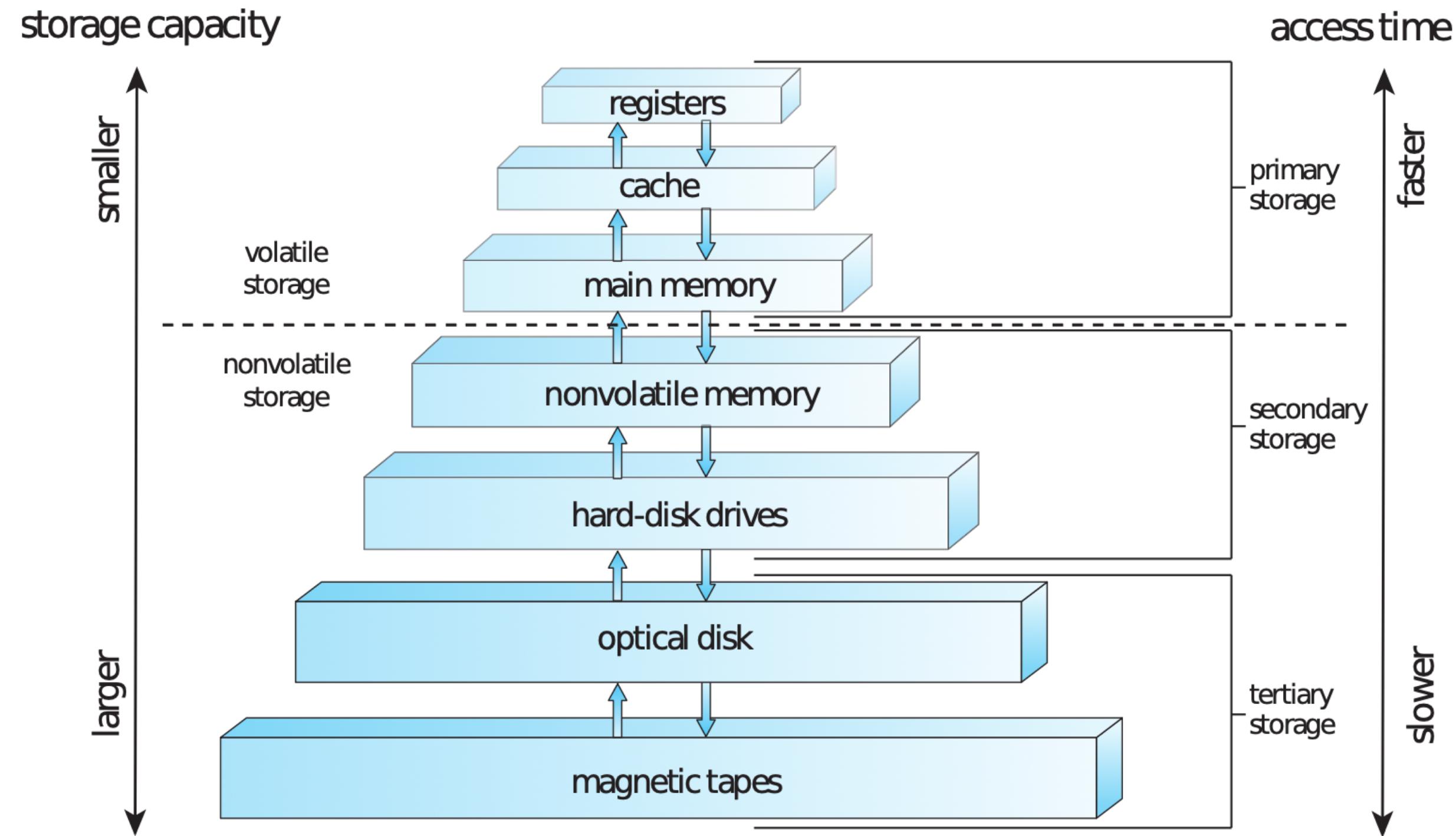


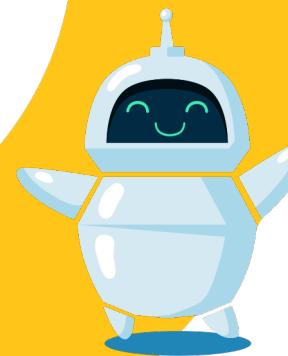
Storage





Storage

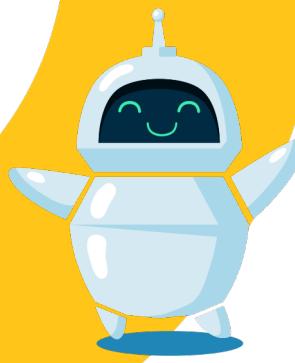




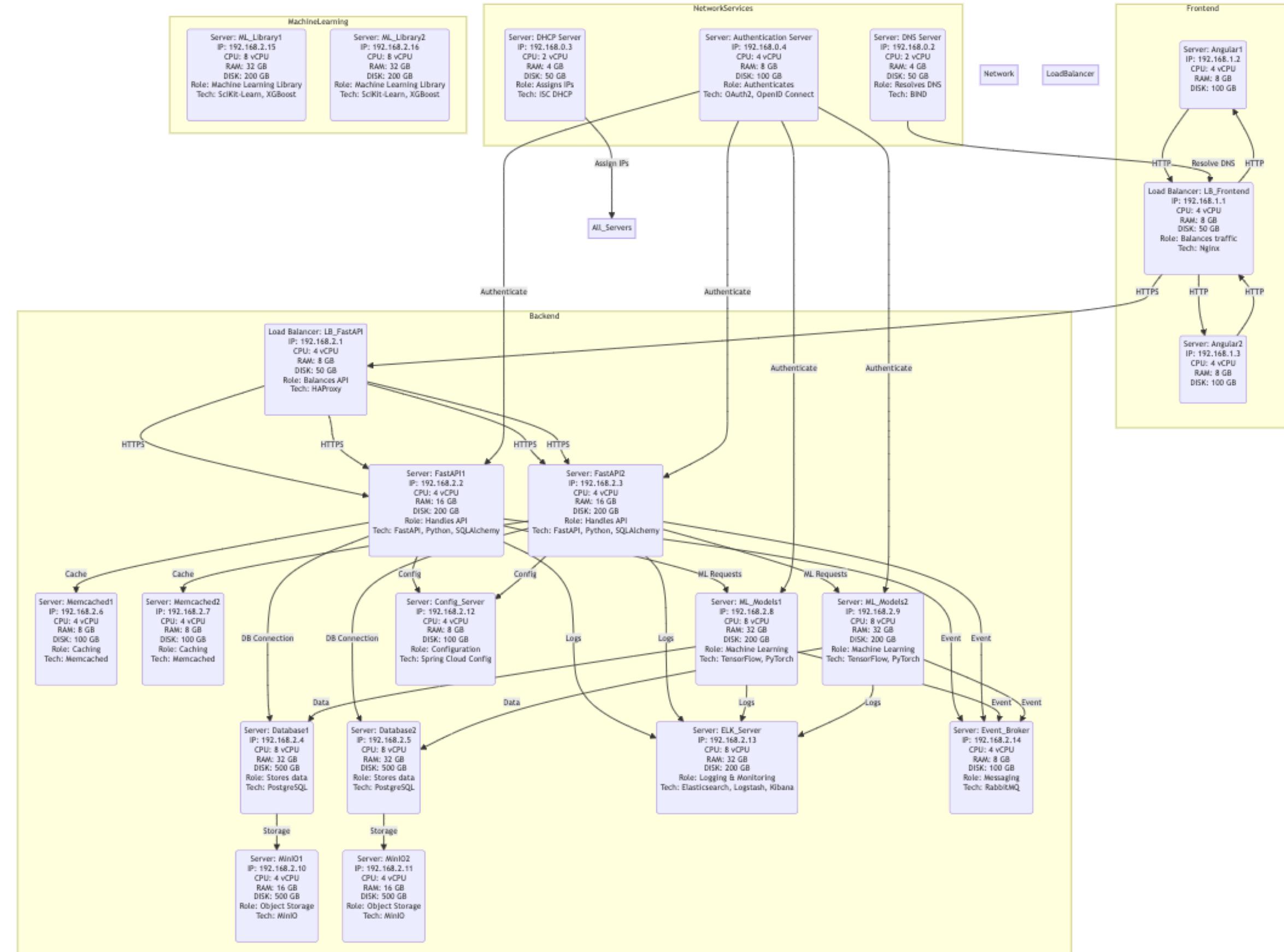
X X X X

Joystopper.io revenons à notre sujet





Joystopper.io revenons à notre sujet



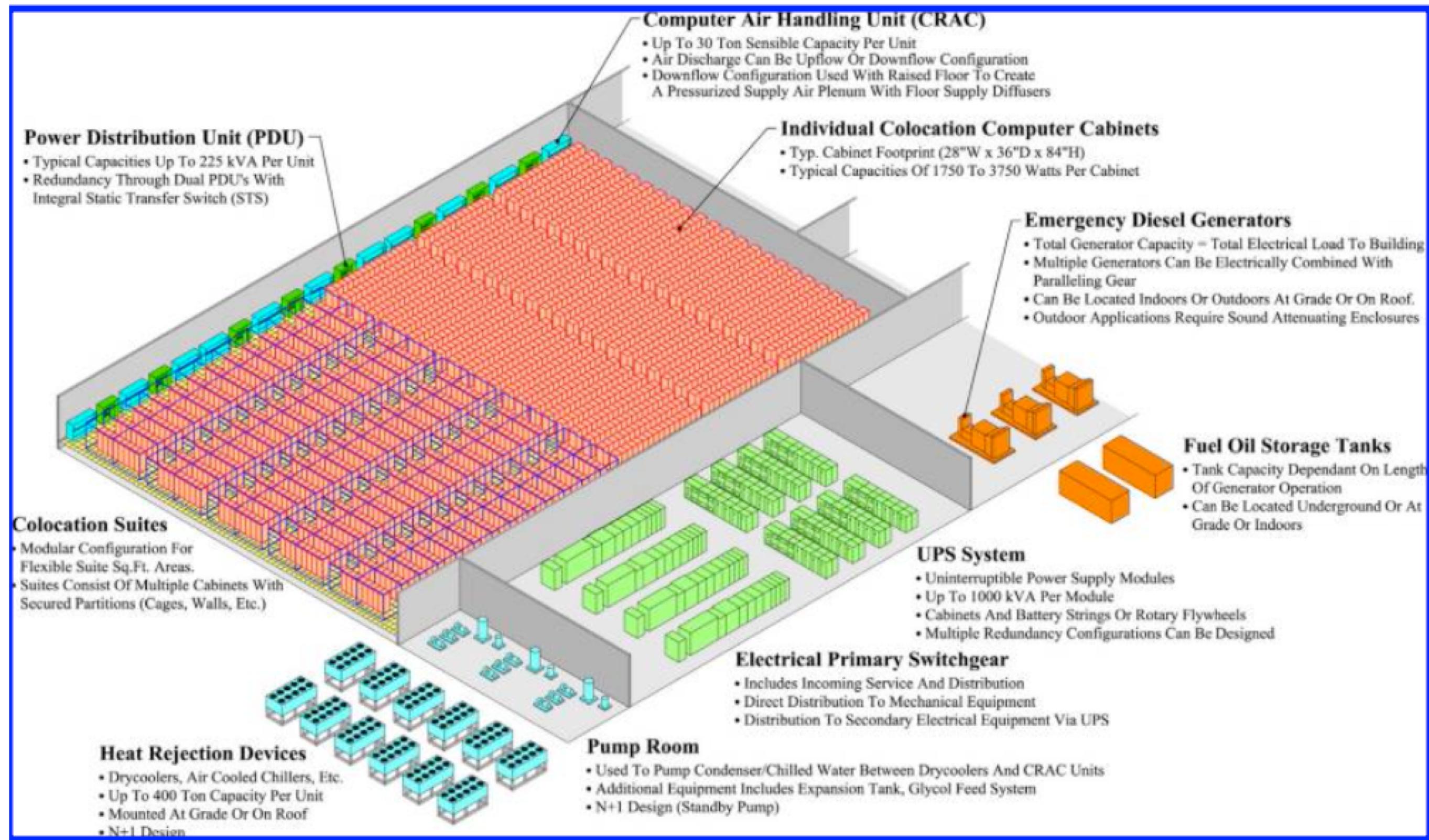
X X X X

Quels sont les options ?





Construire un DC à lingo?





Acheter ou louer des serveurs Physiques ?

- **~ 22 Serveurs**
- **2 Switch**
- **Min 2 Lignes fibre Noire à 10Gb/s**
- **2 routeurs**
- **2 Firewall**





Logiciels ?

- 1. OS Stable**
- 2. Serveur web**
- 3. Database Server**
- 4. Application Server**
- 5. Firewalls**
- 6. Antivirus**
- 7. Monitoring**
- 8. Backup & recovery**
- 9. HA**
- 10. Config Management**
- 11. User Authentication and Authorization**





Option 2: Acheter 5 Serveurs et Utiliser des VMs





Parlons Virtualisation !!





Définition de la Virtualisation

La virtualisation est une technologie qui permet de créer des versions virtuelles d'éléments informatiques tels que des serveurs, des systèmes d'exploitation, des applications, des réseaux ou des ressources de stockage.

Ces environnements virtuels fonctionnent de manière indépendante de leur infrastructure physique sous-jacente, offrant ainsi une **flexibilité**, une **isolation** et une gestion améliorées des ressources informatiques. En d'autres termes, la virtualisation permet de **maximiser** l'utilisation des ressources matérielles, de faciliter la gestion et de créer des environnements informatiques plus **agiles** et **adaptables**.





Histoire de la Virtualisation

1960 – IBM avec sa solution VM/370

1979 – Chroot :

Chroot, introduit dans les systèmes Unix, permet d'altérer la racine du répertoire, fournissant une première forme d'isolation d'environnement. Cependant, cette isolation est limitée et n'offre pas une véritable virtualisation.

1999 – VMware :

VMware, fondée par Diane Greene et Mendel Rosenblum, lance son premier produit de virtualisation, **VMware Workstation**. Cette technologie permettait d'exécuter plusieurs systèmes d'exploitation sur un seul ordinateur physique.

2003 : Acquisition de Connectix par Microsoft

En 2003, Microsoft a acquis Connectix, la société qui développait Virtual PC. Cette acquisition a contribué à l'introduction de la virtualisation dans l'écosystème Microsoft.

2001 – Sortie 2005 Xen Hypervisor :

Xen, un hyperviseur open source, fait son apparition, marquant une étape importante dans la virtualisation. Il permettait l'exécution simultanée de plusieurs machines virtuelles sur un même serveur.





Histoire de la Virtualisation

2006 - Intel VT et AMD-V :

Les extensions de virtualisation matérielles, Intel VT (Virtualization Technology) et AMD-V (AMD Virtualization), sont introduites. Elles améliorent les performances et la sécurité des machines virtuelles

2006 : VMware ESX Server

VMware a publié VMware ESX Server en 2001, une plateforme de virtualisation de serveurs qui a gagné en popularité dans les centres de données.

2007 - Introduction de KVM :

Le Kernel-based Virtual Machine (KVM) est intégré au noyau Linux, fournissant une solution de virtualisation de type hyperviseur pour les processeurs x86.

2010-2013 - Docker et la Virtualisation Légère :

Docker introduit la **virtualisation légère** avec l'utilisation de conteneurs, permettant l'isolation des applications sans le besoin d'un système d'exploitation complet pour chaque application.





Histoire de la Virtualisation

2013 - Introduction de Microsoft Hyper-V pour Windows 8 :

Microsoft Hyper-V devient accessible pour un public plus large avec son inclusion dans les versions grand public de Windows, démocratisant davantage la virtualisation.

2014 - Containers et Docker Boom :

Les conteneurs, popularisés par Docker, gagnent en popularité en raison de leur légèreté, de leur rapidité et de leur efficacité par rapport à la virtualisation traditionnelle.

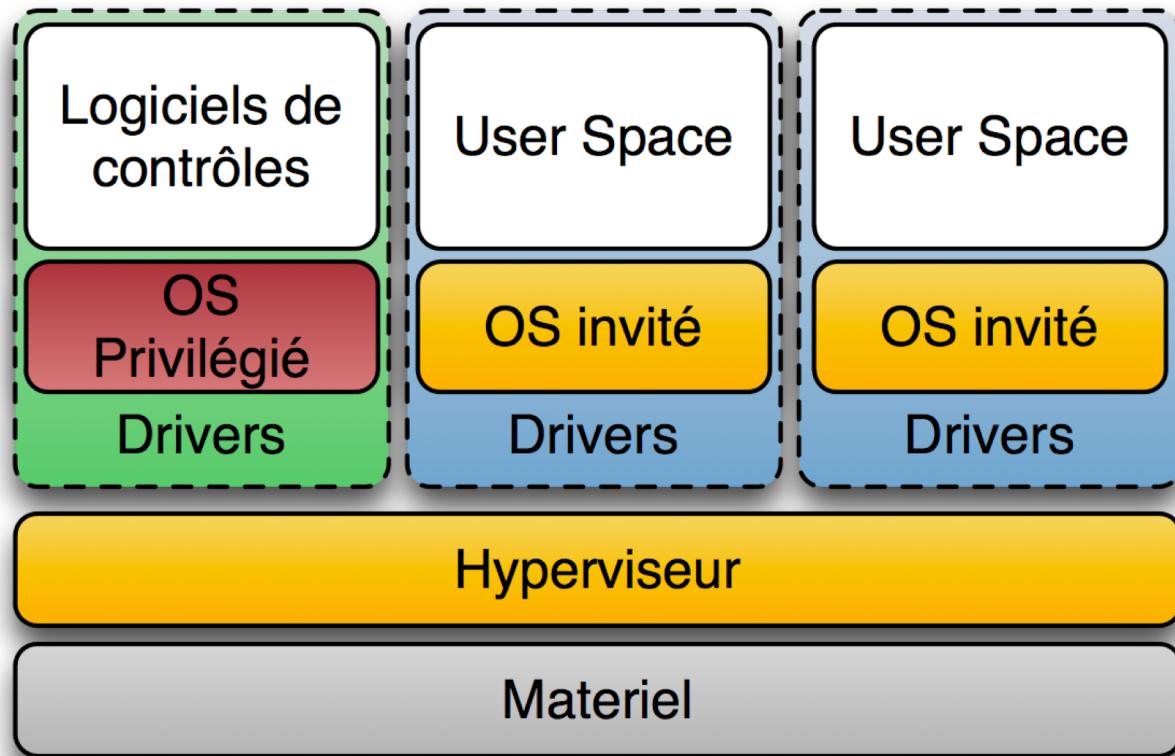
Aujourd'hui - Fusion de Technologies :

Les approches de virtualisation traditionnelles et les technologies de conteneurs se fusionnent dans des solutions comme **Kubernetes**, permettant une gestion harmonieuse d'environnements hétérogènes.



Types de virtualisation

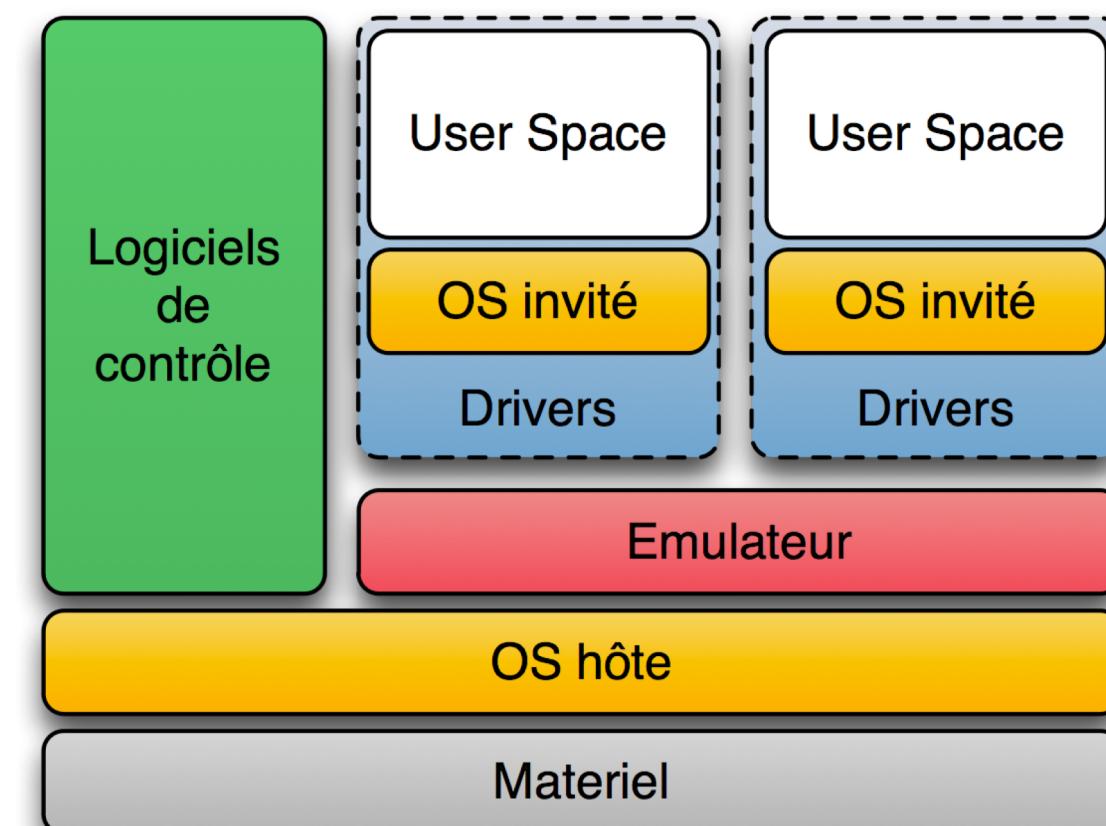
Les hyperviseurs sont principalement catégorisés en **type 1 ou type 2**, en fonction de leur emplacement dans le système, c'est-à-dire de la présence ou non du système d'exploitation sous-jacent



Un hyperviseur de type 1 s'exécute directement sur le matériel, sans nécessiter de système d'exploitation hôte

<<bare-metal, embarqué ou natif>>

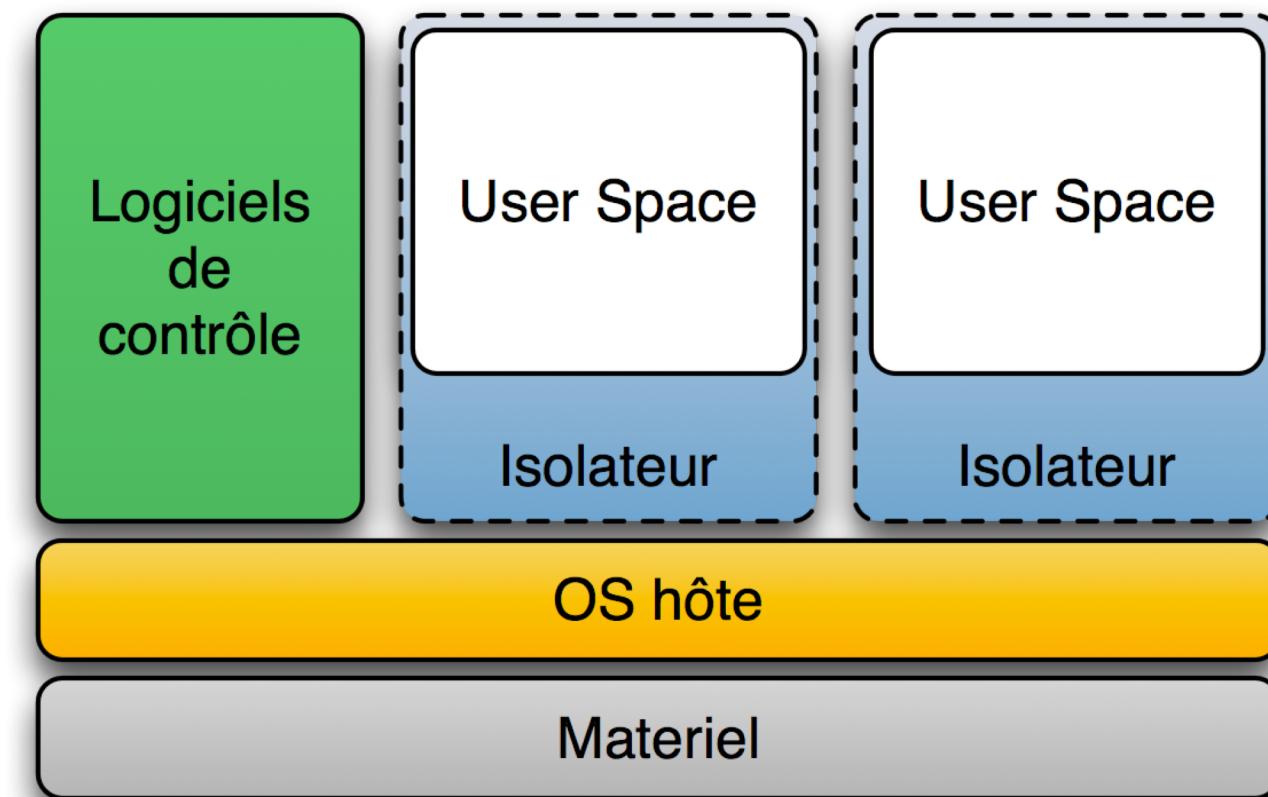
Xen, VMware ESXi/vSphere et Red Hat Enterprise Virtualization Hypervisor (RHEV-H).



Un hyperviseur de type 2 opère en tant que couche distincte au-dessus d'un système d'exploitation existant

Virtualbox, VMWare

Types de virtualisation



Isolation

Chroot, LXC, Docker, Open VZ, BSD Jail



Domaines de virtualisation

Desktop virtualization – Virtual Desktop Infrastructure (VDI) :

Utilisée par de nombreuses entreprises offre une flexibilité exceptionnelle, permettant aux utilisateurs de se connecter à leur bureau virtuel depuis divers appareils. Les avantages clés incluent une gestion centralisée simplifiée, des mises à jour efficaces pour de multiples machines virtuelles, et des processus de déploiement sans installations physiques, ce qui facilite la gestion de la conformité et de la sécurité.





Domaines de virtualisation

La virtualisation de serveurs:

Largement adoptée par la plupart des entreprises, permet une **consolidation** efficace des machines virtuelles par rapport aux serveurs physiques. Elle offre également de nombreux avantages opérationnels, tels que des sauvegardes facilitées, une plus grande **efficacité énergétique**, une flexibilité accrue pour déplacer des charges de travail de serveur à serveur, et bien d'autres.





Domaines de virtualisation

La virtualisation réseau, Réseau Défini par Logiciel (Software Defined Networking (SDN))

Permet de créer des réseaux virtuels indépendants des appareils physiques, tels que les commutateurs.

À une plus grande échelle, le SDN étend cette idée sur plusieurs sites ou centres de données, permettant une configuration réseau basée sur le logiciel sans nécessiter de configurations réseau physiques spécifiques. L'avantage principal réside dans la gestion simplifiée de réseaux complexes couvrant divers emplacements, éliminant le besoin d'une reconfiguration physique extensive des appareils réseau.





Domaines de virtualisation

La virtualisation du stockage (et un concept plus récent, Software-Defined Storage - SDS)

Est une technologie qui crée des dispositifs de stockage virtuels à partir de dispositifs physiques regroupés que nous pouvons gérer de manière centralisée comme un seul dispositif de stockage. Cela signifie que nous créons une sorte de couche d'abstraction qui isole la fonction interne des dispositifs de stockage des ordinateurs, des applications et d'autres types de ressources. Le SDS, en tant qu'extension de cela, dissocie la pile logicielle de stockage du matériel sur lequel elle s'exécute en abstrayant les plans de contrôle et de gestion du matériel sous-jacent, tout en offrant différents types de ressources de stockage aux machines virtuelles et aux applications (ressources basées sur les blocs, les fichiers et les objets).



www.wooclap.com

NPIMAU



www.wooclap.com

XMKPTY





Exercice

Exercice : Installation de 2 Frontends Ubuntu/focal64 avec Load Balancer sur VirtualBox

Objectif : Mettre en place un environnement de serveurs frontend avec équilibrage de charge sur VirtualBox en utilisant Ubuntu/xenial64, Nginx pour les fronts et le load balancer.

Instructions :

1.Configuration du réseau :

1. Assurez-vous que Créez un réseau interne dans VirtualBox pour connecter les machines virtuelles (VM).
2. les VM peuvent communiquer entre elles via ce réseau interne.

2.Installation des Frontends :

1. Créez deux machines virtuelles Ubuntu/focal64 dans VirtualBox.
2. Configurez les paramètres réseau de chaque VM pour les connecter au réseau interne.
3. Installez les services nécessaires sur chaque VM pour les préparer en tant que frontends.

3.Configuration de l'Équilibrage de Charge :

1. Créez une troisième machine virtuelle qui servira de load balancer.
2. Configurez le load balancer pour redistribuer le trafic entre les deux frontends.
3. Assurez-vous que le load balancer peut communiquer avec les frontends via le réseau interne.

4.Validation :

1. Testez l'équilibrage de charge en accédant aux frontends via l'adresse IP du load balancer.
2. Assurez-vous que le trafic est distribué de manière équitable entre les deux frontends.

Remarques :

- Documentez chaque étape de votre configuration.
- Utilisez des outils de gestion à distance (SSH) pour faciliter la configuration.





IaC Infrastructure as Code





IaC Infrastructure as Code c'est Quoi?

GitOps

Configuration

DevOps

Infrastructure as Code

Canary

Deployment

UAT

Containers

Infrastructure Provisioning

CI/CD

DevSecOps

Provisioning

Prod

DEV

Orcherstration

Blue/Green

Automatisation

OPS

Continuous Testing

Continuous Delivery

Cloud

Scalability





IaC Infrastructure as Code c'est Quoi?

Traditionnellement, le **développement** était distinct de l'équipe **opérationnelle** et des serveurs.

Auparavant, lorsqu'un **développeur** travaillait sur une application, il la construisait localement sur sa machine, puis transférait les fichiers via **FTP** sur un serveur en direct (production) pour exécuter le code.

En cas de problèmes ou de bugs, le développeur devait effectuer des modifications sur l'environnement du serveur et déboguer le code.

Certaines entreprises continuent d'adopter cette méthode, peut-être en raison de leur environnement ou parce qu'ils n'ont pas d'autres choix.

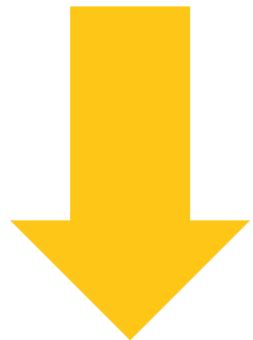


× × × ×

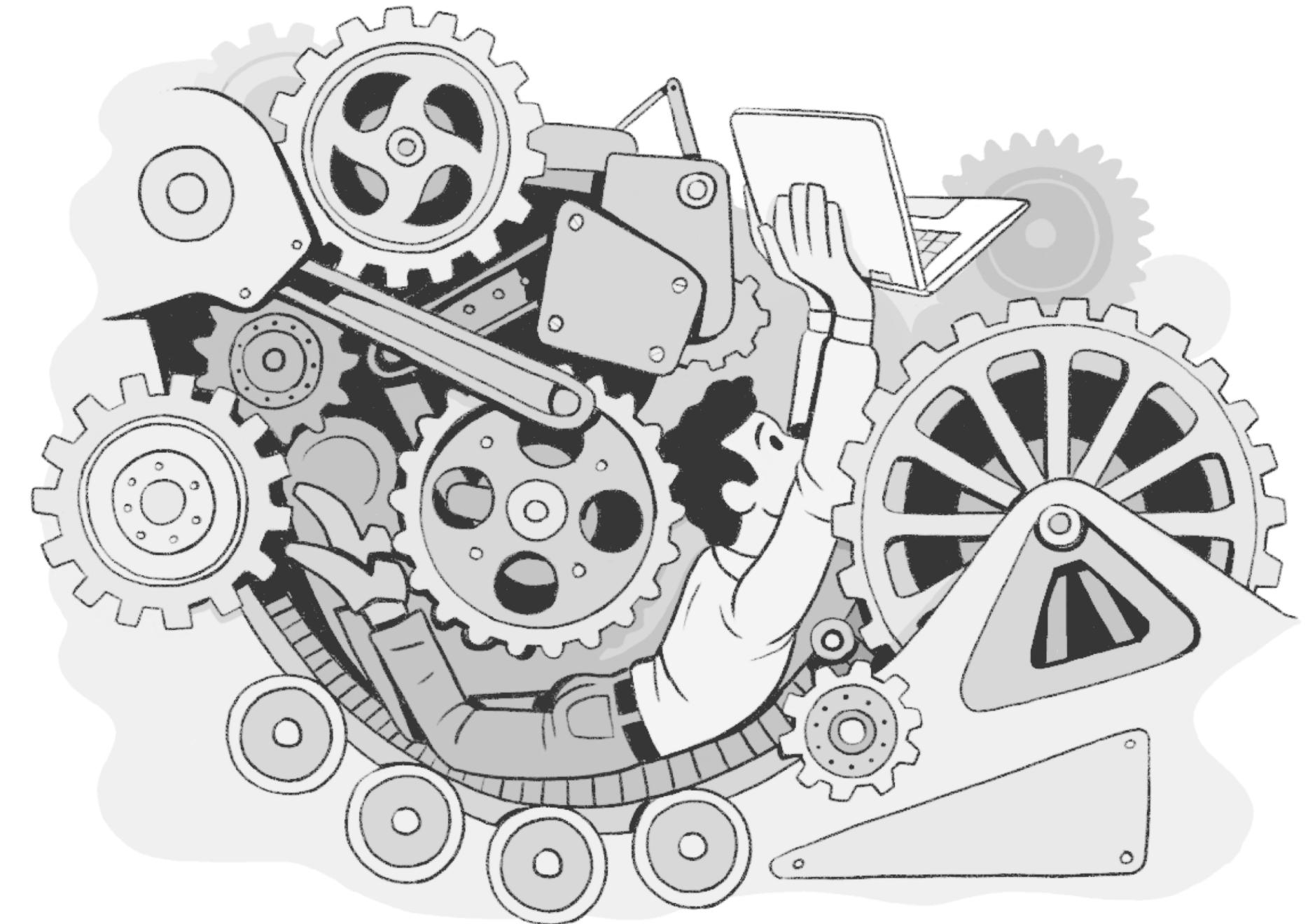
IaC Infrastructure as Code c'est Quoi?



IaC Infrastructure as Code c'est Quoi?



Automatisation





Objectifs de l'automatisation

La vitesse :

La vitesse est la monnaie de l'économie numérique car elle permet une innovation rapide et peu coûteuse. L'automatisation vous rend plus rapide et aide ainsi l'entreprise à rivaliser avec les concurrents. (**TTM Time to Market**)

Répétabilité:

Aller vite, c'est bien, mais pas si cela se fait au détriment de la qualité. L'automatisation accélère non seulement les choses, mais elle élimine également la principale source d'erreur dans le déploiement de logiciels : **les humains**. C'est pourquoi vous ne devriez jamais confier à un humain le travail d'une machine.

L'automatisation élimine la marge d'erreur d'un processus et le rend reproductible.

Confiance :

La répétabilité engendre la confiance. Si l'équipe de développement craint la prochaine version, elle ne sera pas rapide :

la peur entraîne **l'hésitation**, et l'hésitation **ralentit** le processus. Si le déploiement a suivi le même processus automatisé les 100 dernières fois, la **confiance** de l'équipe dans la prochaine version est élevée.





Objectifs de l'automatisation

Résilience

On oublie facilement qu'une nouvelle version n'est pas le seul moment où vous déployez un logiciel.

En cas de défaillance matérielle, vous devez déployer rapidement un logiciel sur un nouvel ensemble de machines. Avec une livraison complètement automatisée, cela se fait en quelques secondes ou minutes, souvent avant que le premier appel d'incident ne puisse être mis en place.

L'automatisation du déploiement augmente le temps de disponibilité et la résilience.

Transparence

Les processus automatisés offrent une meilleure compréhension de ce qui s'est passé, **quand**, pour **quelles raisons**, et combien de temps cela a pris pour être terminé. Ce niveau de transparence vous permet, par exemple, de trouver des points de défaillance courants.





Objectifs de l'automatisation

Amélioration continue/perfectionnement

Avoir un processus reproductible et transparent est la condition de base pour une amélioration continue

Conformité

La gouvernance traditionnelle est basée sur la définition de processus manuels et des vérifications périodiques. Il va sans dire que l'écart entre ce que prescrit la gouvernance et la conformité réelle tend à différer assez largement.

L'automatisation garantit la conformité – une fois que le script d'automatisation est défini conformément à la gouvernance, vous êtes assuré d'avoir une conformité à chaque fois.





IaC Infrastructure as Code c'est Quoi?

Concept



Outils



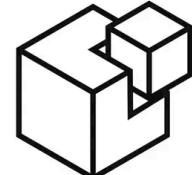
CloudFormation



 **Terraform**

 **puppet**

 **Vagrant**



SALTSTACK



ANSIBLE





IaC Infrastructure as Code c'est Quoi?

Il existe 3 catégories de tâches qu'on souhaite automatiser

- 1. Infrastructure Provisioning**
- 2. Configuration de l'infrastructure installée**
- 3. Déploiement des applications**





IaC Infrastructure as Code c'est Quoi?

Phases

Initialisation

Première initialization de l'infra & config
Installation des application la première fois
Configuration initiale des applications

Maintenance

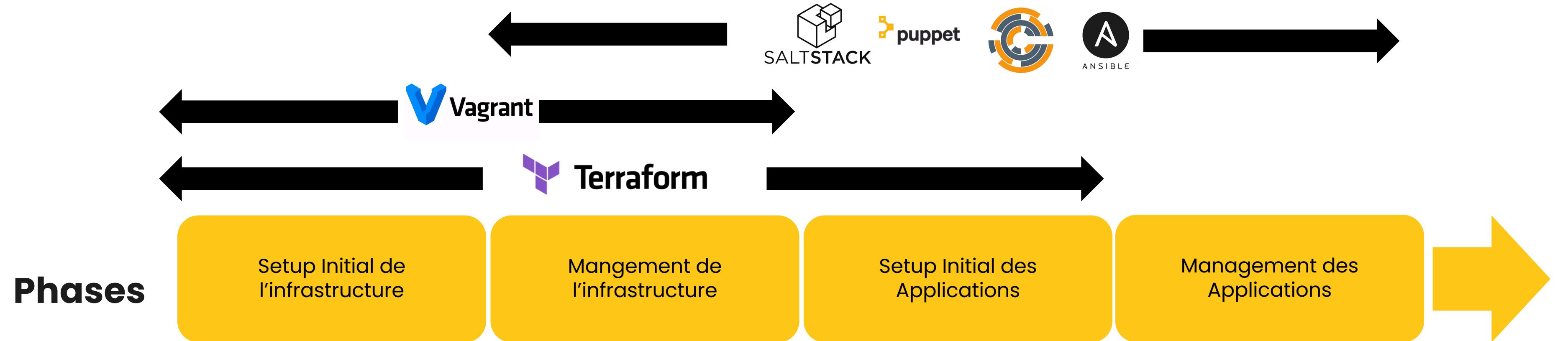
Day-to-day Operations





IaC Infrastructure as Code c'est Quoi?

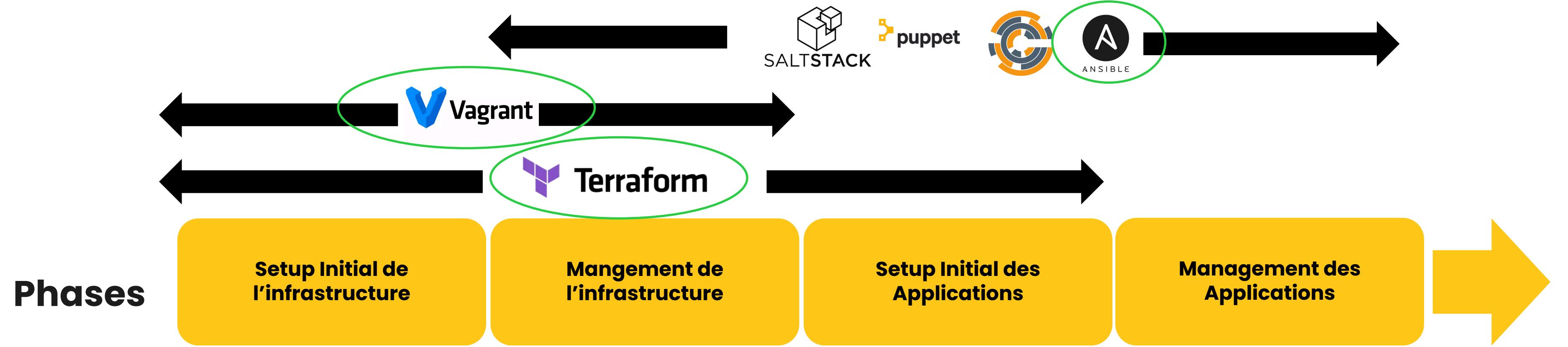
Dans la majorité des cas nous aurons besoin d'une combinaison d'au moins (2) outils afin d'atteindre nos objectifs





IaC Infrastructure as Code c'est Quoi?

La combinaison la plus courante





IaC Infrastructure as Code Les types d'approches?

Impérative

L'approche impérative consiste à décrire **étape par étape comment atteindre un résultat spécifique**. Cela implique de spécifier les actions à effectuer pour réaliser une tâche, souvent dans un ordre séquentiel. Dans le contexte de l'Infrastructure as Code (IaC), cela signifie détailler les étapes pour configurer chaque composant de l'infrastructure.

Exemple:

1. Installez le serveur web Apache.
2. Configurez les règles du pare-feu pour autoriser le trafic sur le port 80.
3. Créez un répertoire pour le site web.
4. Copiez les fichiers du site web dans le répertoire.
5. Redémarrez le service Apache

VS.

Déclarative

L'approche **déclarative** consiste à décrire le résultat souhaité sans spécifier explicitement les étapes pour l'atteindre. Cela permet au système de décider de la meilleure façon de mettre en œuvre la configuration. Dans le contexte de l'IaC, cela signifie spécifier **l'état désiré du système**, et l'outil d'IaC se charge de déterminer comment y parvenir.

Exemple:

```
1  ---
2  - name: Install Docker
3  hosts: docker
4  become: true
5
6  vars:
7  ansible_distribution_release: trusty
8  theuser: vagrant
9
10 tasks:
11 - name: Ensure old versions of Docker are not installed.
12   package:
13     name:
14     - docker
15     - docker-engine
16     state: absent
17
```





IaC Infrastructure as Code

Vagrant est un outil qui simplifie le workflow et réduit la charge de travail nécessaire pour exécuter et opérer des **machines virtuelles (VM)**.

Il réalise également les actions suivantes :

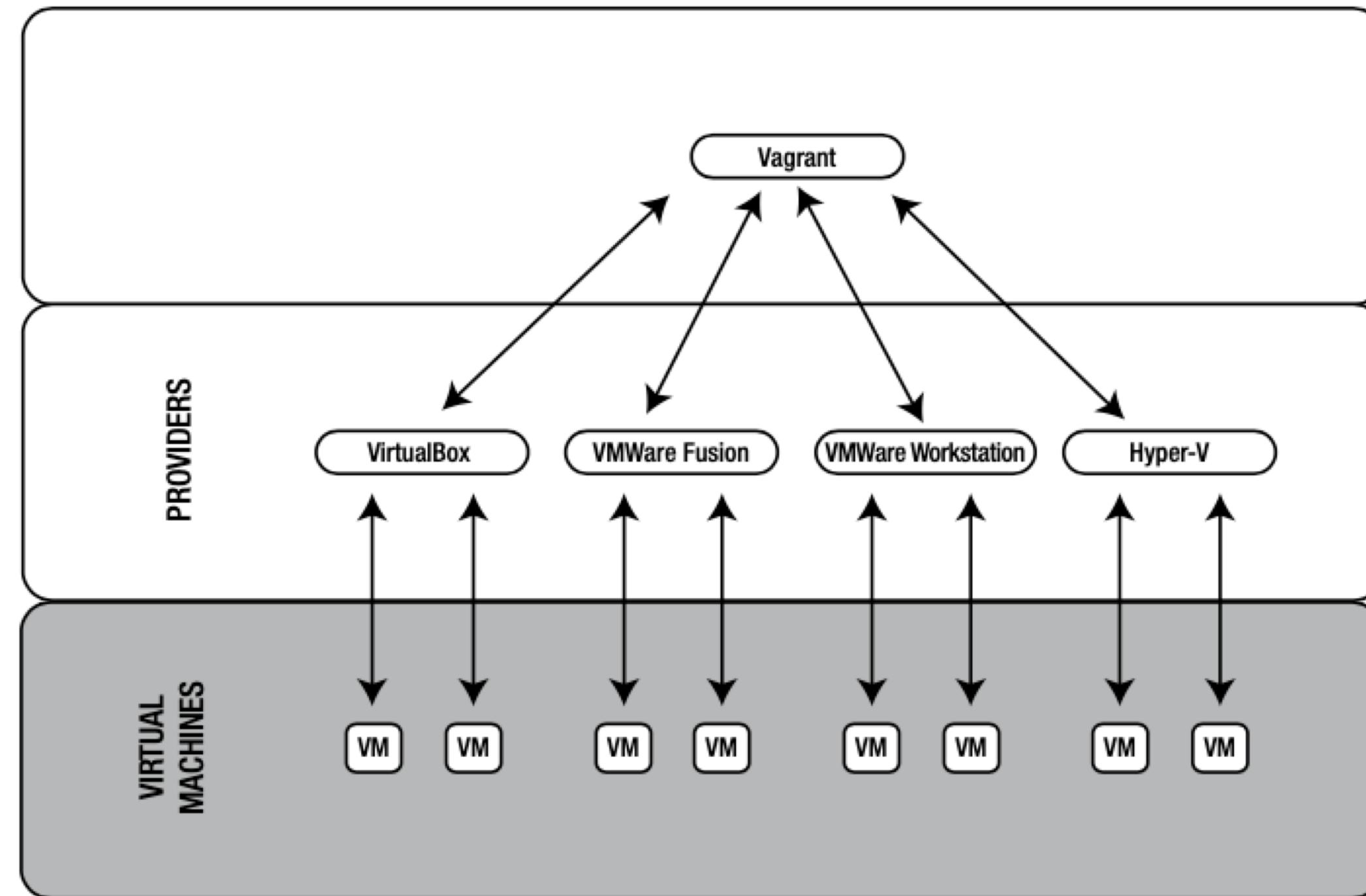


- Offre une interface en ligne de commande très simple pour gérer les VM.
- Prend en charge toutes les solutions virtuelles majeures : VirtualBox, VMWare et Hyper-V.
- Supporte la plupart des outils de configuration logicielle populaires, notamment Ansible, Chef, Puppet et Salt.
- Facilite les procédures de distribution et de partage d'environnements virtuels.



IaC Infrastructure as Code

Développé initialement par **Mitchell Hashimoto en 2010**





IaC Infrastructure as Code



Vagrantfile

Un fichier **Vagrantfile** est un fichier de configuration qui utilise la syntaxe du langage de programmation Ruby.

Il est facile à comprendre et peut être rapidement testé en apportant une modification, puis en exécutant la commande **vagrant up** pour voir si les résultats escomptés se produisent.

Un fichier **Vagrantfile** peut être facilement partagé et ajouté au contrôle de version. Il est léger et contient tout le nécessaire pour qu'un autre utilisateur puisse reproduire l'environnement/application virtuel.





IaC Infrastructure as Code



Boxes

Les boxes Vagrant sont des packages qui, tout comme les fichiers Vagrantfile, peuvent être partagés et utilisés pour reproduire des environnements virtuels.

Les boxes Vagrant peuvent être facilement téléchargées en utilisant la commande `vagrant box add`.

Le cloud Vagrant propose un catalogue facilement consultable de boxes.





IaC Infrastructure as Code



Networking

Vagrant prend en charge trois principaux types de réseaux lors de la création d'environnements virtuels :

les réseaux publics, les réseaux privés et le renvoi de ports (Port-forwarding).

L'option de réseau la plus simple est le renvoi de ports, qui vous permet d'accéder à un port spécifique à travers le système d'exploitation invité dans la machine Vagrant.





IaC Infrastructure as Code



Provisionning

Le provisionnement dans Vagrant offre une manière de configurer la machine Vagrant de manière encore plus approfondie.

Vous pouvez installer des logiciels et des dépendances au fur et à mesure que la machine est créée.

Pour provisionner une machine Vagrant, vous pouvez utiliser des scripts shell, Docker, Chef, Ansible et d'autres logiciels de gestion de configuration, tels que Puppet





IaC Infrastructure as Code



Plugins

Les plugins Vagrant offrent une autre façon de personnaliser et d'étendre les fonctionnalités de Vagrant.

Ils vous permettent d'interagir avec les aspects de bas niveau de Vagrant et fournissent souvent de nouvelles commandes à utiliser



CLPPAY

www.wooclap.com





Exercice

Exercice : En utilisant Vagrant - Installation de 2 Frontends Ubuntu/xenial64 avec Load Balancer sur VirtualBox

Objectif : Mettre en place un environnement de serveurs frontend avec équilibrage de charge sur VirtualBox en utilisant Ubuntu/xenial64, Nginx pour les fronts et le load balancer.

Instructions :

1.Configuration du réseau :

1. Assurez-vous de Créez un réseau interne dans VirtualBox pour connecter les machines virtuelles (VM).
2. les VM peuvent communiquer entre elles via ce réseau interne.

2.Installation des Frontends :

1. Créez deux machines virtuelles Ubuntu/focal64 dans VirtualBox.
2. Configurez les paramètres réseau de chaque VM pour les connecter au réseau interne.
3. Installez les services nécessaires sur chaque VM pour les préparer en tant que frontends.

3.Configuration de l'Équilibrage de Charge :

1. Créez une troisième machine virtuelle qui servira de load balancer.
2. Configurez le load balancer pour redistribuer le trafic entre les deux frontends.
3. Assurez-vous que le load balancer peut communiquer avec les frontends via le réseau interne.

4.Validation :

1. Testez l'équilibrage de charge en accédant aux frontends via l'adresse IP du load balancer.
2. Assurez-vous que le trafic est distribué de manière équitable entre les deux frontends.

Remarques :

- Documentez chaque étape de votre configuration.
- Utilisez des outils de gestion à distance (SSH) pour faciliter la configuration.





IaC Infrastructure as Code



Première version stable date de **2012**
Racheté par RedHat en 2015

«Ansible is Simple»





IaC Infrastructure as Code

Ansible c'est quoi?



ANSIBLE

- Outil d'automatisation open source
- Gestion de configuration et déploiement d'applications
- Sans agent : pas besoin d'installer un logiciel sur les nœuds cibles





IaC Infrastructure as Code



ANSIBLE

- **Agentless:** Utilise SSH pour se connecter aux nœuds
- **Déclaration d'État:** Décrire l'état souhaité du système
- **Playbooks:** Scripts décrivant les tâches à effectuer
- **Modules:** Petites unités de travail exécutées par Ansible



IaC Infrastructure as Code

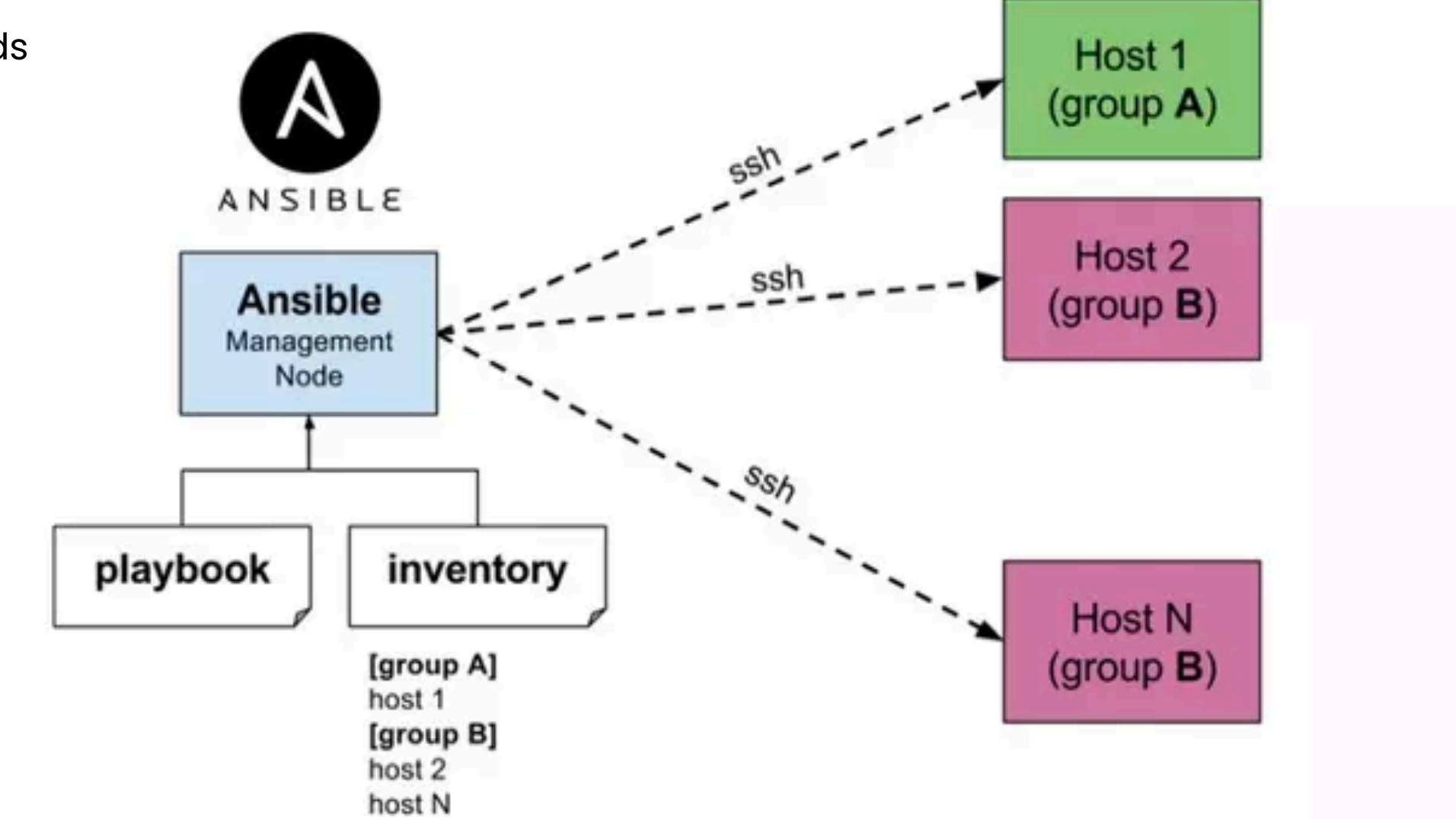
Fonctionnement

Inventaire: Liste des nœuds à gérer

Playbooks: Descriptions d'états souhaités

Modules: Exécution de tâches spécifiques

Connexion SSH: Aucun agent à installer sur les nœuds





IaC Infrastructure as Code

Installation

Il faudra avoir Python & pip

pip install ansible





IaC Infrastructure as Code



Terraform

