

De la découverte à l'expertise

# Docker

Virtualisation



Martin LEKPA



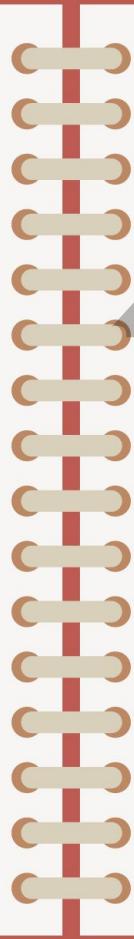
Théorie



Pratique

# Programme

- 1  Virtualisation
- 2  Conteneurisation
- 3  Observabilité





# Virtualisation

Définition et concepts

# Virtualisation

- Serveur hôte
- Serveur privé virtuel
- Environnement isolé
- Système d'exploitation



© Freepik

# Avantages

- Utilisation optimale des ressources
- Installation, déploiement et migration facile
- Sécurisation et/ou Isolation d'un réseau
- Socle de base du Cloud Computing
- ...

# Inconvénients

- Parfois inadapté (Ex : I/O intense).
- Un recours à des machines puissantes
- Une complexité accrue de l'analyse d'erreurs
- ...

# Terminologie

- Le **système hôte (host)** est l'OS principal de l'ordinateur.
- Le **système invité (guest)** est l'OS installé à l'intérieur d'une machine virtuelle.
- Une **machine virtuelle (VM)** est un ordinateur virtuel qui utilise un système invité.
- Un ordinateur virtuel est aussi appelé serveur privé virtuel (**Virtual Private Server ou VPS**) ou environnement virtuel (**Virtual Environment ou VE**)

# Domaines de virtualisation

- Virtualisation de serveurs
- Virtualisation des applications
- Virtualisation du stockage
- Virtualisation du réseau

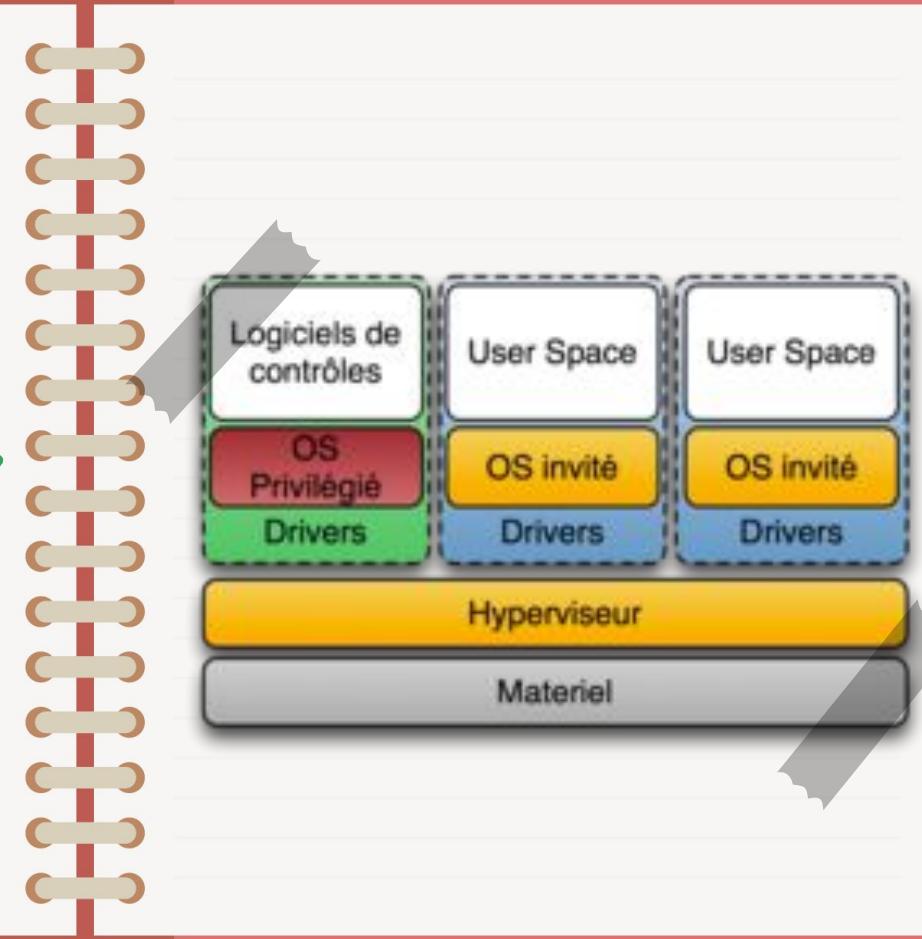


# Virtualisation de serveurs



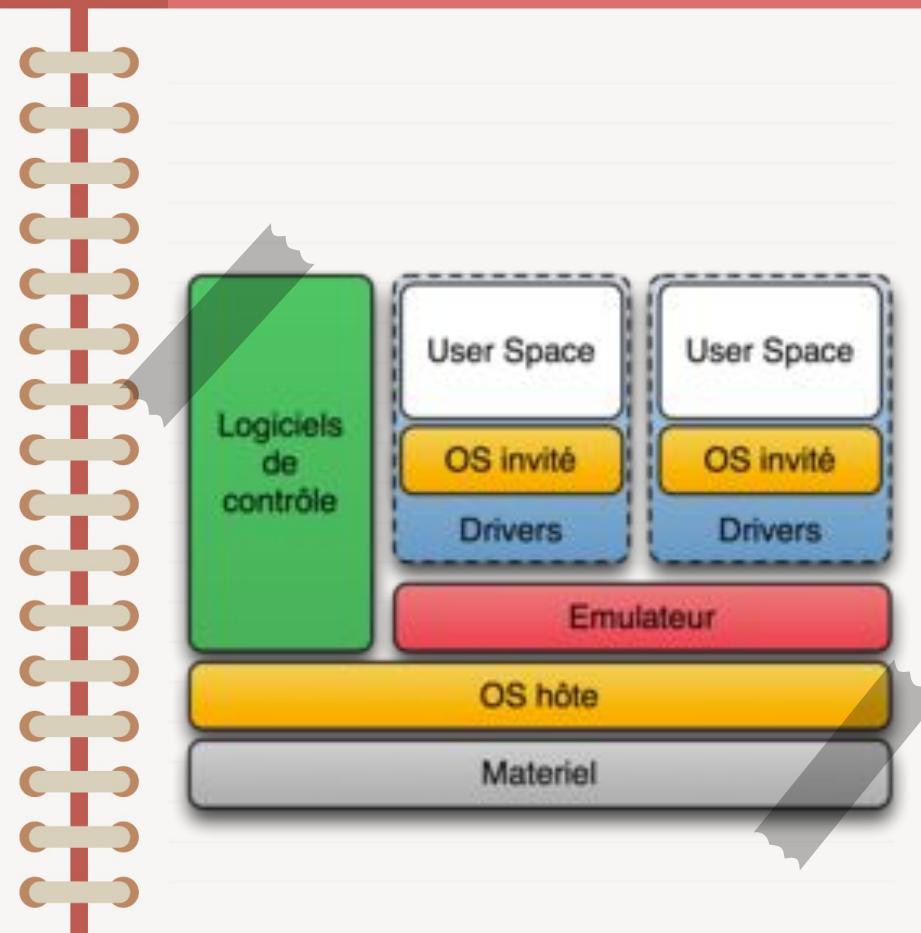
# Hyperviseur de type 1

- Noyau système très léger
- Gestion noyaux OS invités
  - VMware vSphere
  - Microsoft Hyper-V Server
  - Proxmox VE
  - Oracle VM, KVM, ...



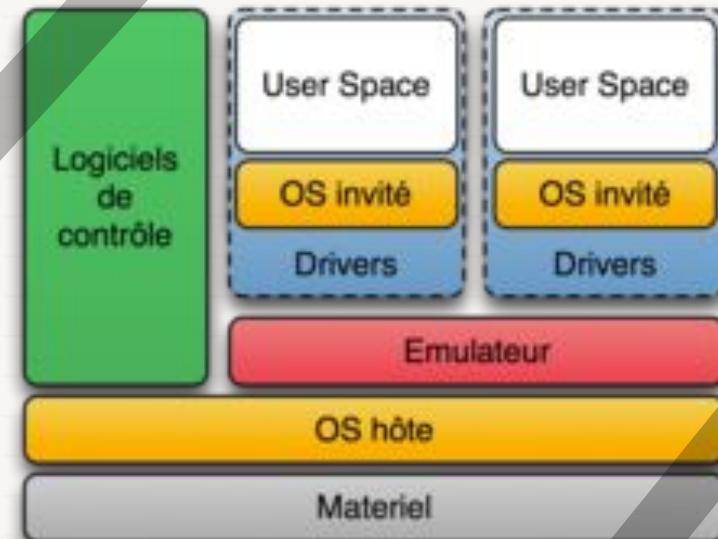
# Hyperviseur de type 2

- Logiciel sur OS hôte
- Gestion OS invités
  - Vmware Fusion/Player
  - Oracle VM Virtualbox
  - QEMU
  - Microsoft VirtualPC/Virtual Server ...



# Isolateur

- Logiciel sur OS hôte
- Isolation des exécutions



## Les usages

- Hyperviseur de type 1 → Professionnel
- Hyperviseur de type 2 → Professionnel / Particulier
- Isolateur → Professionnel / Particulier
- Noyau en espace utilisateur → Développement de noyaux

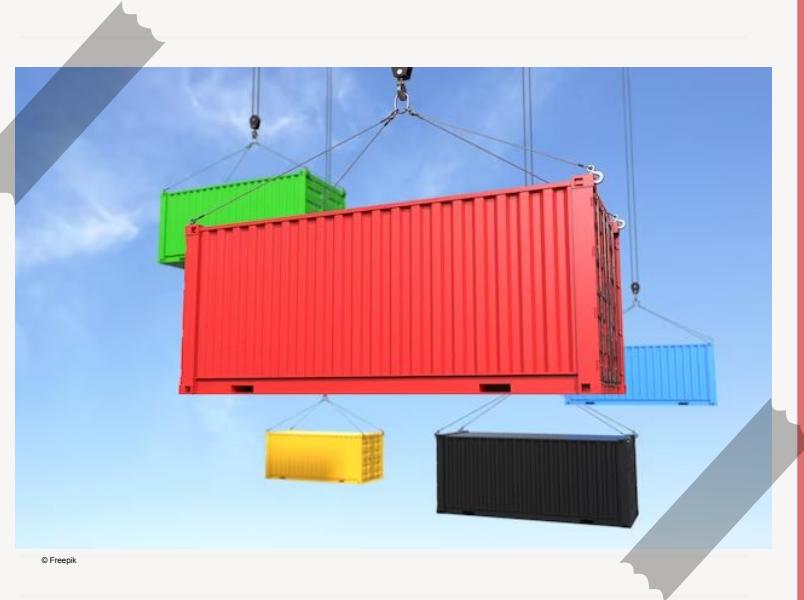
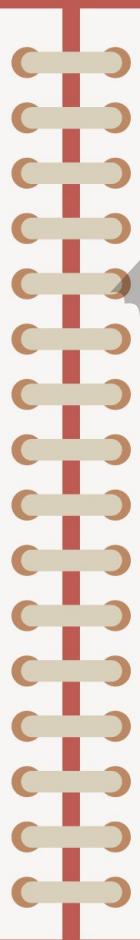


# La Conteneurisation

Cas de Docker

# Conteneurs

- Environnement isolé
- Système hôte
- Fonctions de configuration
- Dépend de l'hôte/kernel

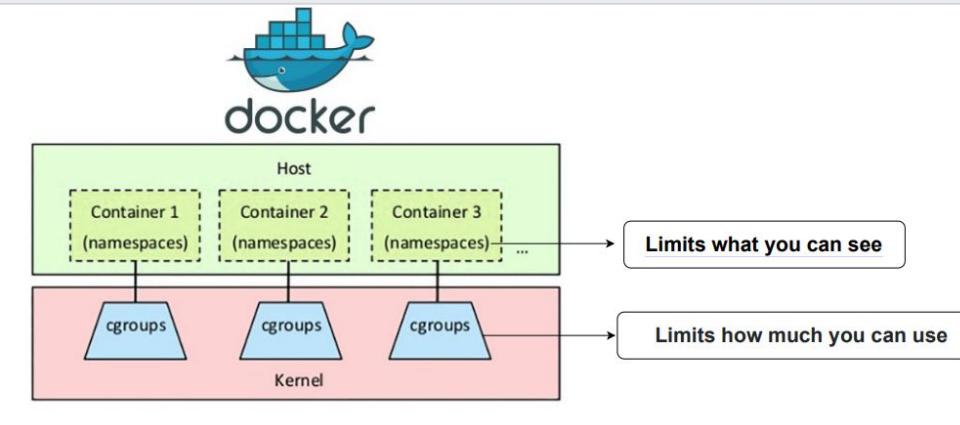


# Les types de conteneurs

- **System** : simule une séquence de boot complète avec un init process ainsi que plusieurs processus (LXC, OpenVZ).
- **Process** : un conteneur exécute un ou plusieurs processus directement, en fonction de l'application conteneurisée (Docker, Rkt).

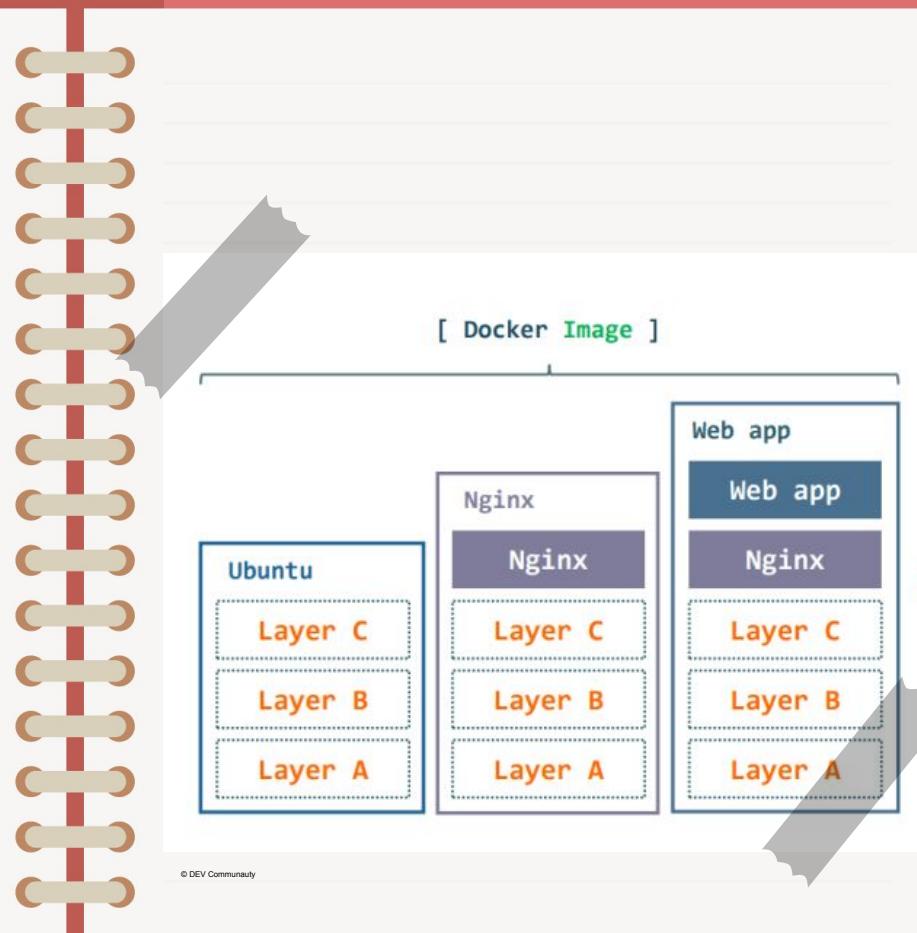
# Notions importantes

- Cgroup (Control Groups)
- Namespaces



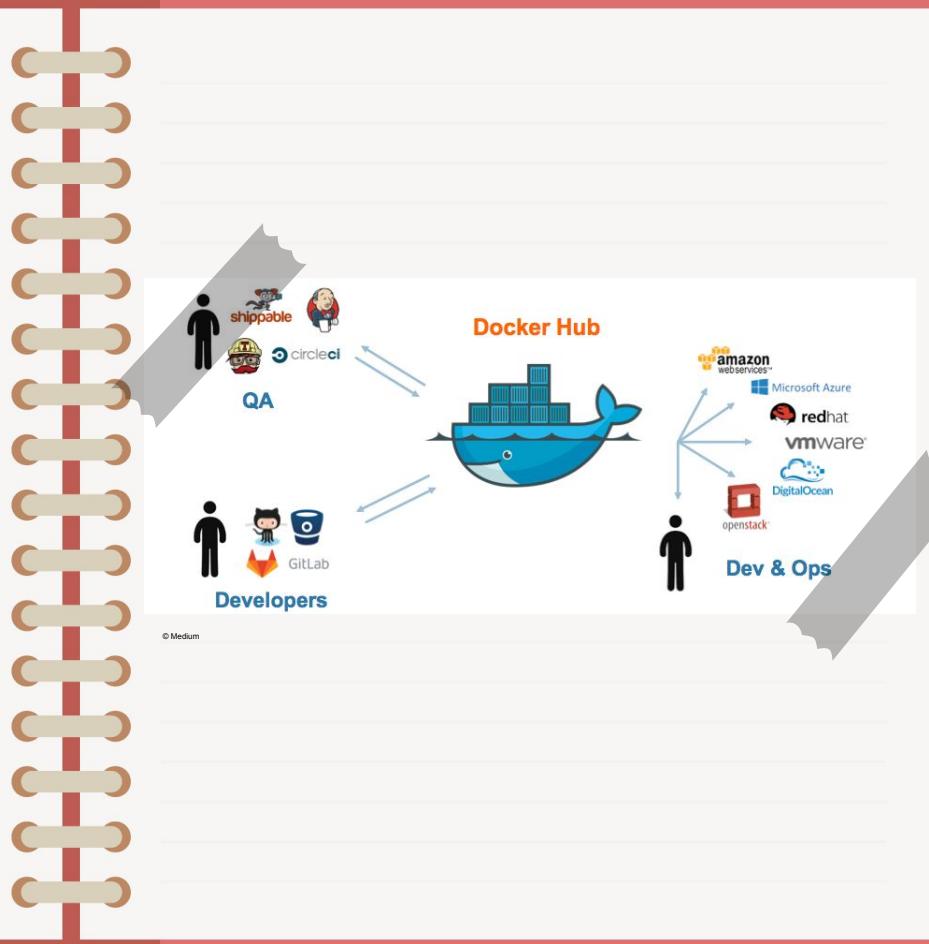
# Les images

- Point de départ
- Archives / Snapshot
- On ne refait pas la roue



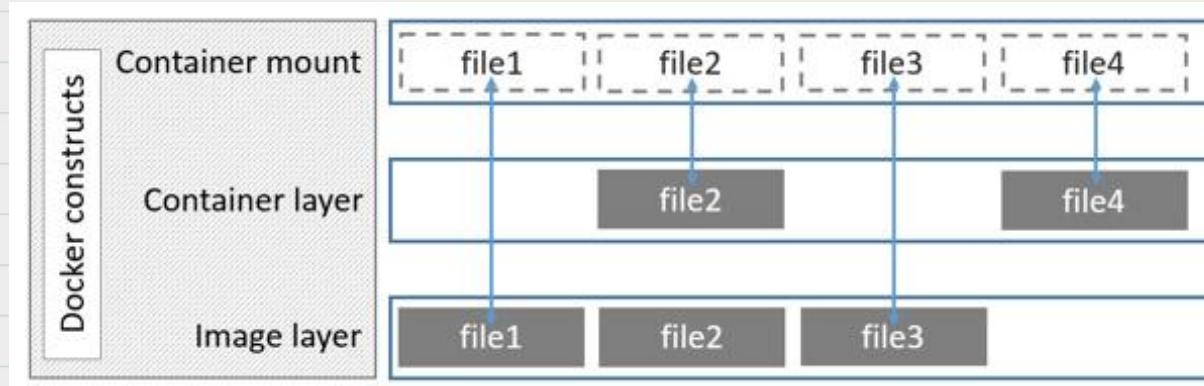
# Le registre

- Distributeur d'images
- Sécurité ?
- Docker Hub



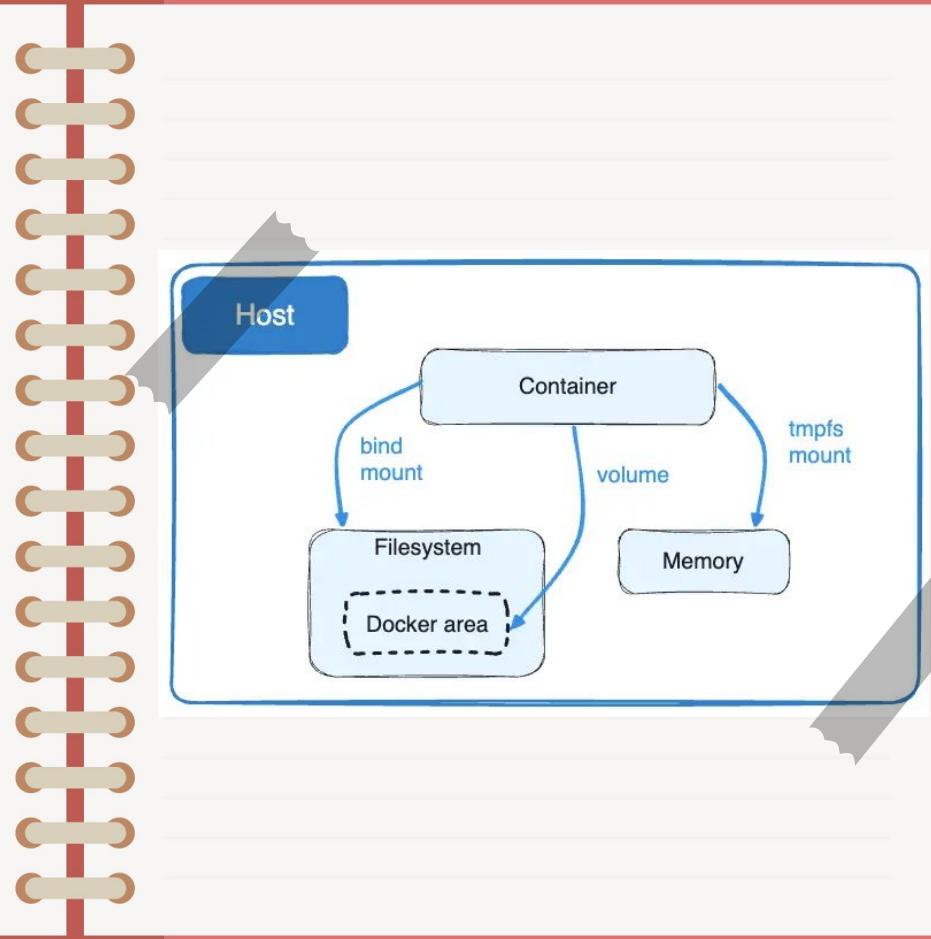
# L'empilement des couches (Layers)

- *Structure en couche*
- *Partage de couches*



# Le stockage

- AUFS
- DeviceMapper
- OverlayFS
- Plugins

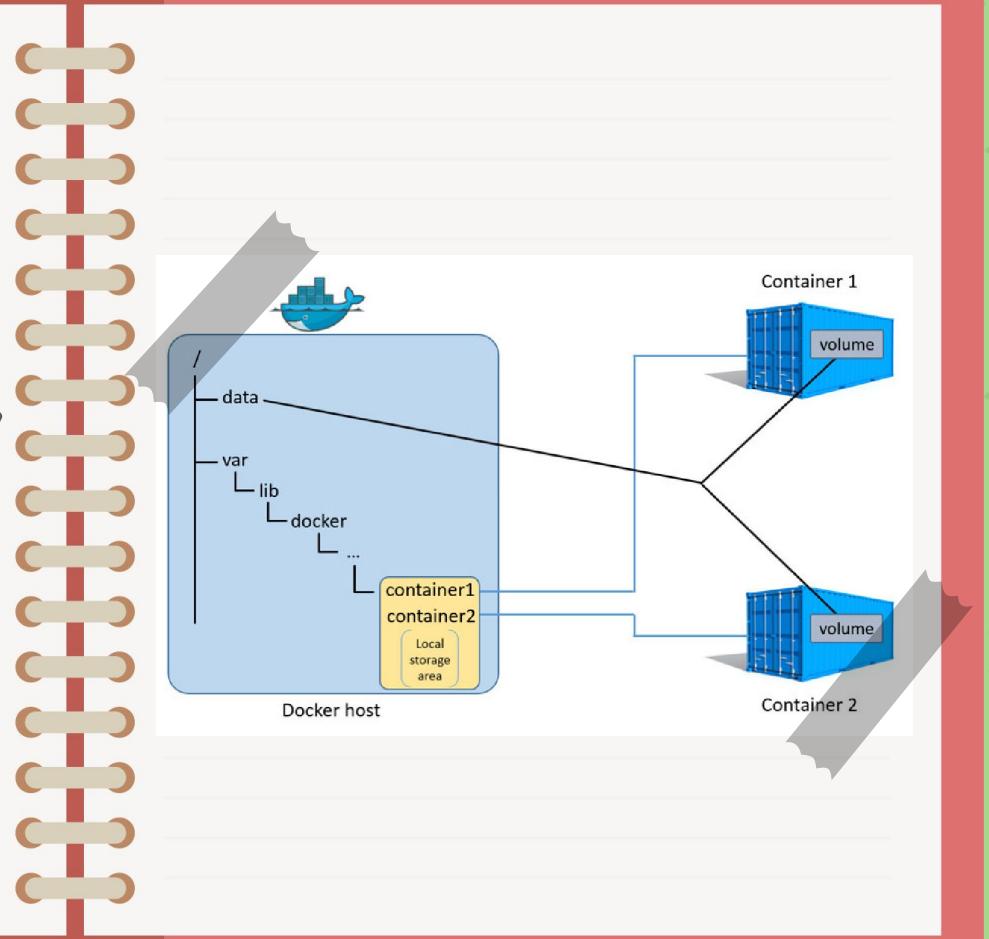


# Les volumes

- Persistance des données
- Indépendance vis à vis des conteneurs/layers
- Deux types de volume:

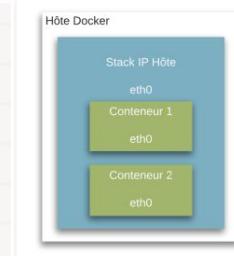
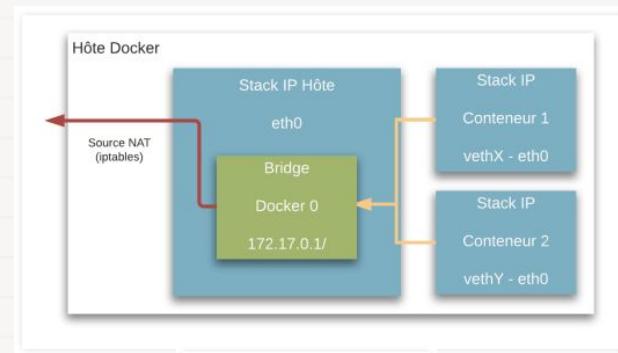
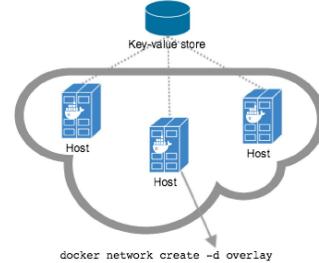
Conteneur : data conteneur

Hôte : dossier



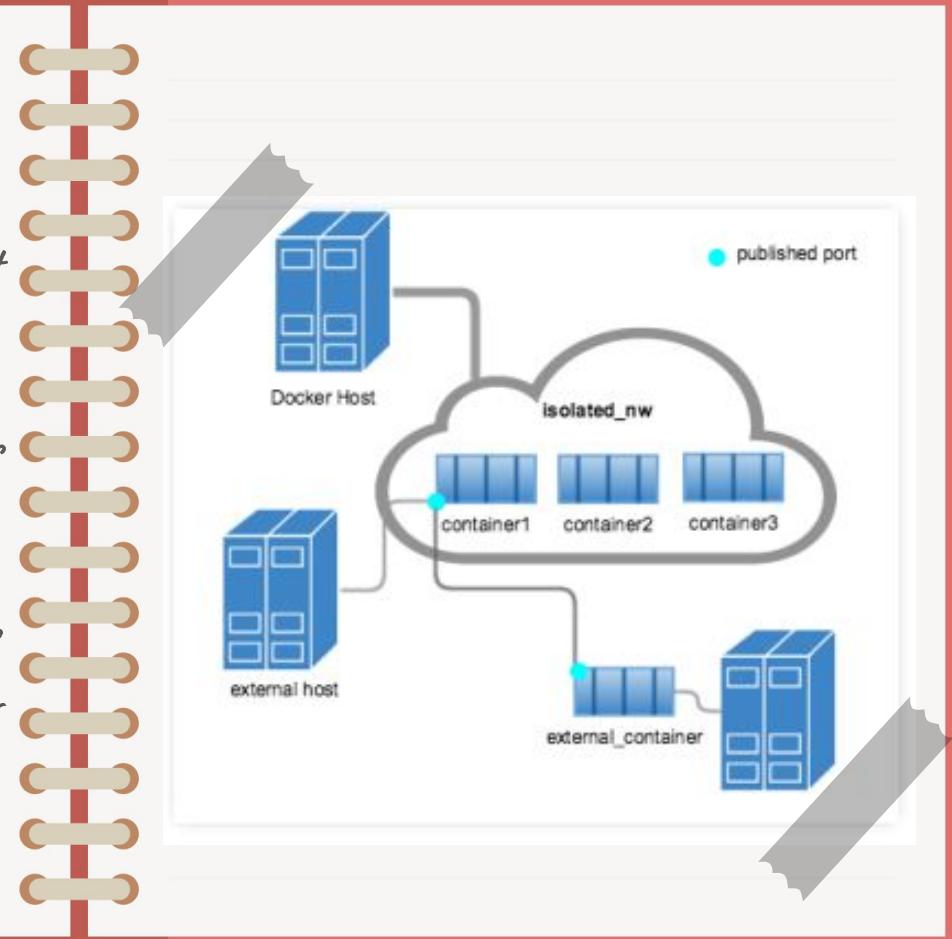
# Réseau (Network) - Les types

- Bridge
- Host
- None
- Overlay



## Réseau (Network) Publication des ports (Publish)

- Dans le cas d'un réseau différent de l'hôte
- Les conteneurs ne sont pas accessible depuis l'extérieur
- Possibilité de publier des ports depuis l'hôte vers le conteneur (iptables)
- L'hôte sert de proxy au service



# Le réseau (Network)

- Les conteneurs d'un même réseau peuvent communiquer via IP
- Système de DNS rudimentaire (`/etc/hosts`)

# Sécurité (Security)

- Les conteneurs sont très sûr mais pour cela il faut respecter les bonnes pratiques de sécurité
- La remédiation peut être assez fastidieuse. Il est donc préférable de mettre en place à l'initialisation
- <https://github.com/docker/docker-bench-security>



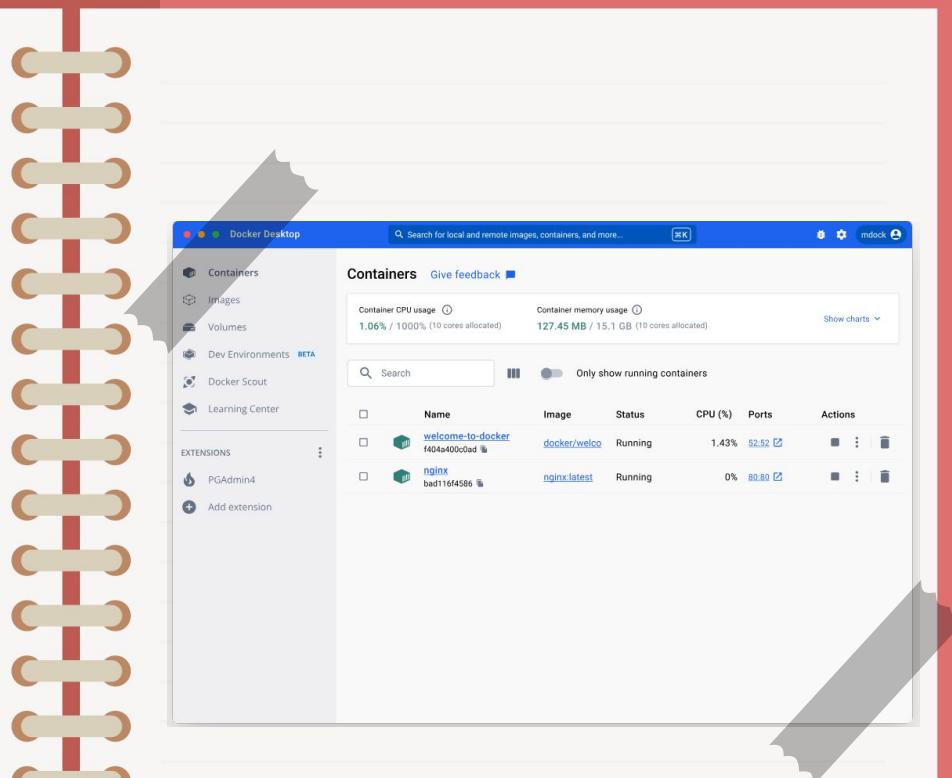
```
# Docker Bench for Security v1.3.6
# Docker, Inc. (c) 2015-2022
#
# Checks for dozens of common best-practices around deploying Docker containers in production.
# Based on the CIS Docker Benchmark 1.4.0.
#
# Initializing 2022-03-07T12:37:08+00:00

Section A - Check results

[INFO] 1 - Host Configuration
[INFO] 1.1 - Linux Hosts Specific Configuration
[WARN] 1.1.1 - Ensure a separate partition for containers has been created (Automated)
[INFO] 1.1.2 - Ensure only trusted users are allowed to control Docker daemon (Automated)
[INFO]           * Users: vagrant
[WARN] 1.1.3 - Ensure auditing is configured for the Docker daemon (Automated)
[WARN] 1.1.4 - Ensure auditing is configured for Docker files and directories - /run/containerd (Automated)
[INFO] 1.1.5 - Ensure auditing is configured for Docker files and directories - /var/lib/docker (Automated)
[WARN] 1.1.6 - Ensure auditing is configured for Docker files and directories - /etc/docker (Automated)
[WARN] 1.1.7 - Ensure auditing is configured for Docker files and directories - docker.service (Automated)
[INFO] 1.1.8 - Ensure auditing is configured for Docker files and directories - containerd.sock (Automated)
[INFO]           * File not found
[WARN] 1.1.9 - Ensure auditing is configured for Docker files and directories - docker.socket (Automated)
[WARN] 1.1.10 - Ensure auditing is configured for Docker files and directories - /etc/default/docker (Automated)
[INFO] 1.1.11 - Ensure auditing is configured for Docker files and directories - /etc/docker/daemon.json (Automated)
```

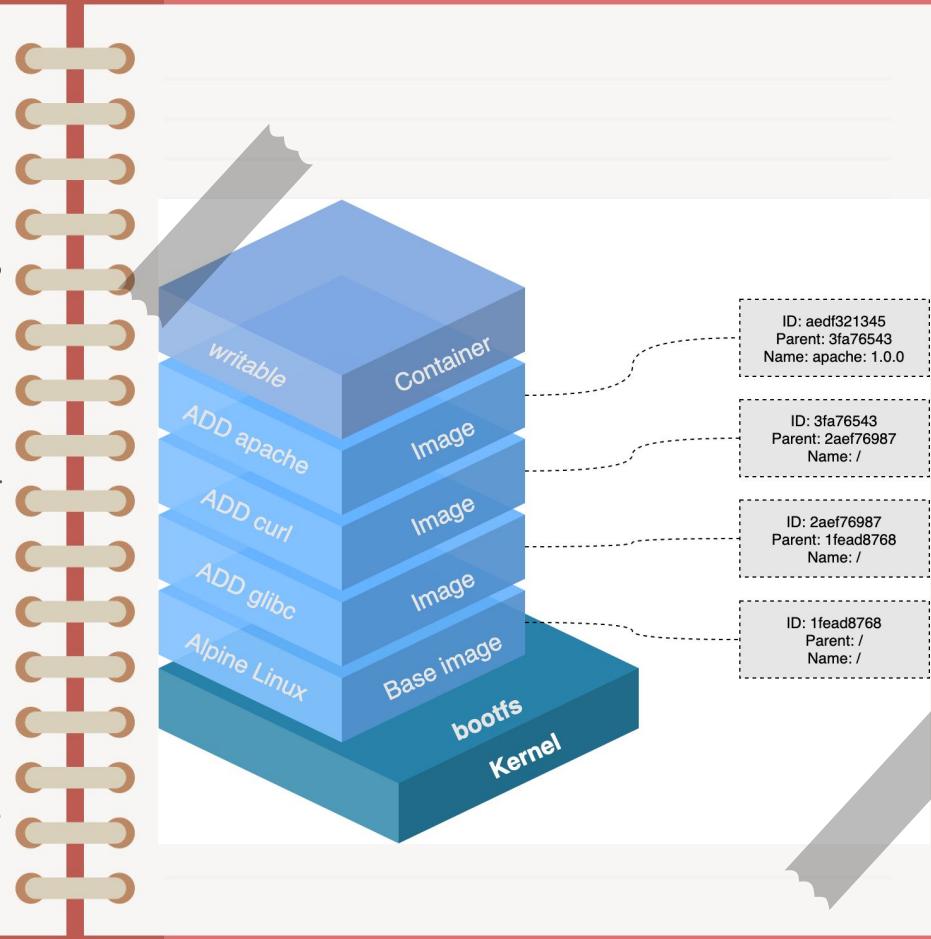
# Docker Desktop

- Docker engine
- Interface graphique
- Et plus



# Dockerfile - Définition

- Fichier de configuration d'une image docker
- Standardisation par rapport à l'export d'un conteneur en image
- Succession d'instructions. Chaque instruction représente une couche (layer)



# Dockerfile - Les instructions 1/3

- **FROM** : socle de l'image.
- **LABEL** : défini des métadonnées de l'image. Auteur, version, ...
- **COPY**: Copie des fichiers dans l'image.
- **ADD**: Copie des fichiers dans l'image (accepte les URL)
- **ENV** : définit des variables d'environnement
- **RUN** : Commandes à exécuter

# Dockerfile - Les instructions 2/3

- **EXPOSE**: déclare les ports d'écoute du conteneur
- **USER**: définit l'utilisateur(groupe) qui exécutera le(s) processus
- **VOLUME**: déclare les points de montage des volumes
- **WORKDIR** : définit le répertoire de travail
- **HEALTHCHECK** : commande de test de bon fonctionnement
- **ARG** :

# Dockerfile - Les instructions 3/3

- CMD, SHELL, ENTRYPOINT, ONBUILD, ...

# Dockerfile - Les bonnes pratiques 1/2

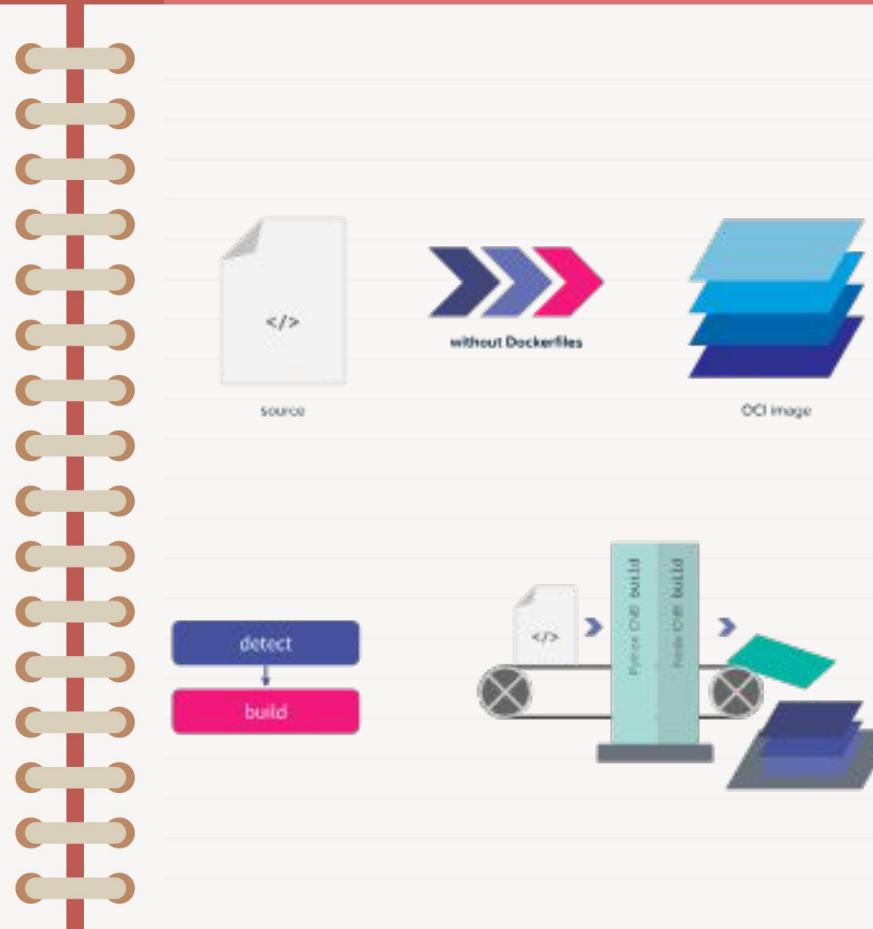
- **FROM**: Utiliser des images minimales et officielles
- **FROM**: Utiliser des images avec des versions spécifiques
- **LABEL**: Indiquer des méta-data sur l'image
- **RUN**: Installer le minimum de dépendances
- **USER**: Appliquer le principe de moindre priviléges (PoLP)
- **EXPOSE**: Exposer les ports

# Dockerfile - Les bonnes pratiques 2/2

- Regrouper au maximum les instructions
- Utiliser les volumes
- Ne pas stocker les informations sensibles et locales
- Utiliser le multi-stage et les cibles pour chaque environnement
- .dockerignore : Contrôler les fichiers copiés dans l'image

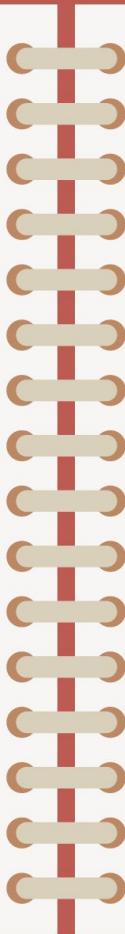
# Buildpacks

- Pas besoin de “Dockerfile”
- Fichiers de dépendance
  - Python : requirements.txt ou setup.py
  - Node: package-lock.json
  - ...
- OCI image : CNCF



## Les autres

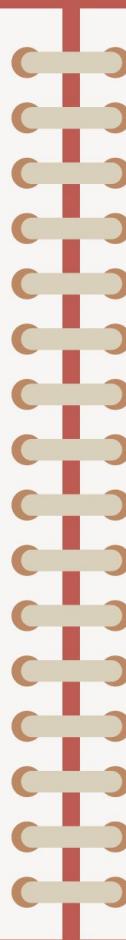
- Buildah
- Buildkit
- IMG
- Kaniko
- Podman
- PouchContainer



OCI

## Docker compose

- Permet d'exécuter plusieurs conteneurs sur Docker
- Simplifie le déploiement de stack : LAMP, supervision...
- docker-compose.yml



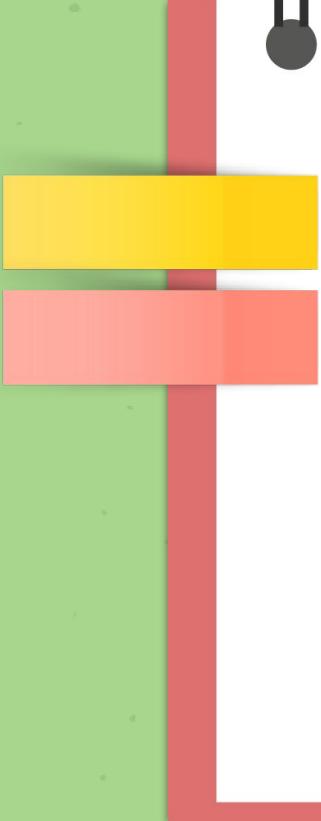
Docker



Compose

# Les langages de données : configuration et échange

- **JSON** : JavaScript Object Notation. Avec séparateur et indentation uniquement visuelle.
- **YAML** : Yet Another Markup Language. Sans séparateur et une indentation (doubles espaces) obligatoire.
- Ils sont facile à lire ou à écrire pour des humains.
- Sont actuellement très utilisés comme langage pour des entrées et sorties



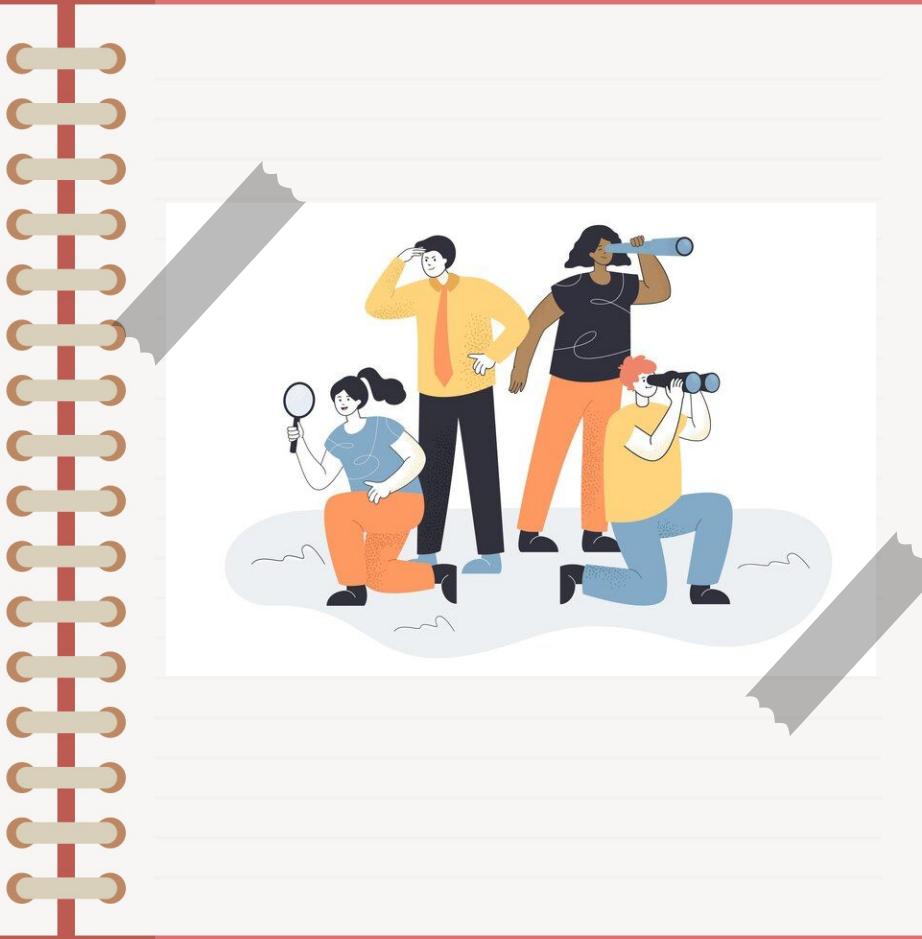
03

# Observabilité

Définition et concepts

# Observabilité

- Etat interne d'un système
- Données émises
- Piliers de l'observabilité
  - Logs
  - Traces
  - Métriques



# Les attentes

- Des rapports sur l'état général des systèmes (Mes systèmes fonctionnent-ils ? Disposent-ils de suffisamment de ressources ?).
- Des rapports sur l'état du système tel qu'il est perçu par les clients (Mes clients savent-ils si mon système est en panne et leur expérience a-t-elle été insatisfaisante ?).
- Une surveillance des métriques clés de l'entreprise et des systèmes.
- Des outils permettant de comprendre et de déboguer vos systèmes en production.
- Des outils permettant de trouver des informations sur des choses que vous ne connaissez pas (c'est-à-dire, vous permettant d'**identifier des inconnus inconnus**).
- L'accès à des outils et à des données permettant de suivre, de comprendre et de diagnostiquer les problèmes d'infrastructure dans votre environnement de production, y compris les interactions entre les services.

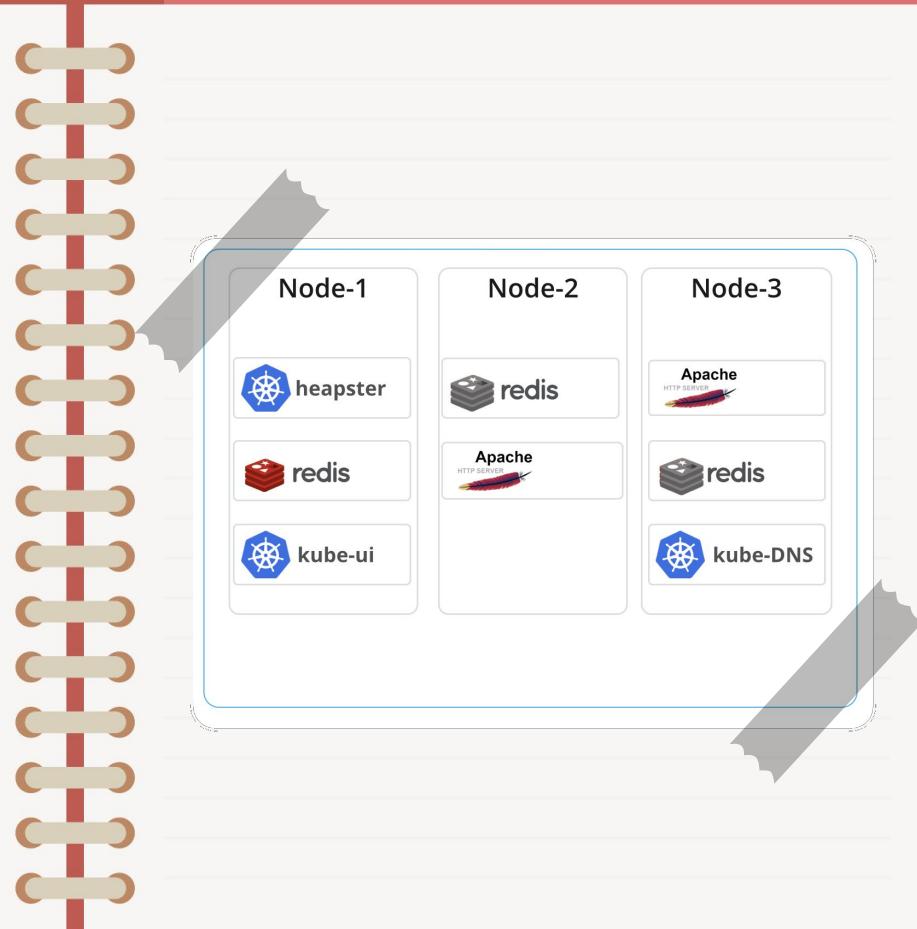
# Monitored son infrastructure

Monitored des serveur est une problématique résolue depuis de nombreuses années par des outils devenus des standards :

- Nagios
- Shinken
- Centreon
- Zabbix ...

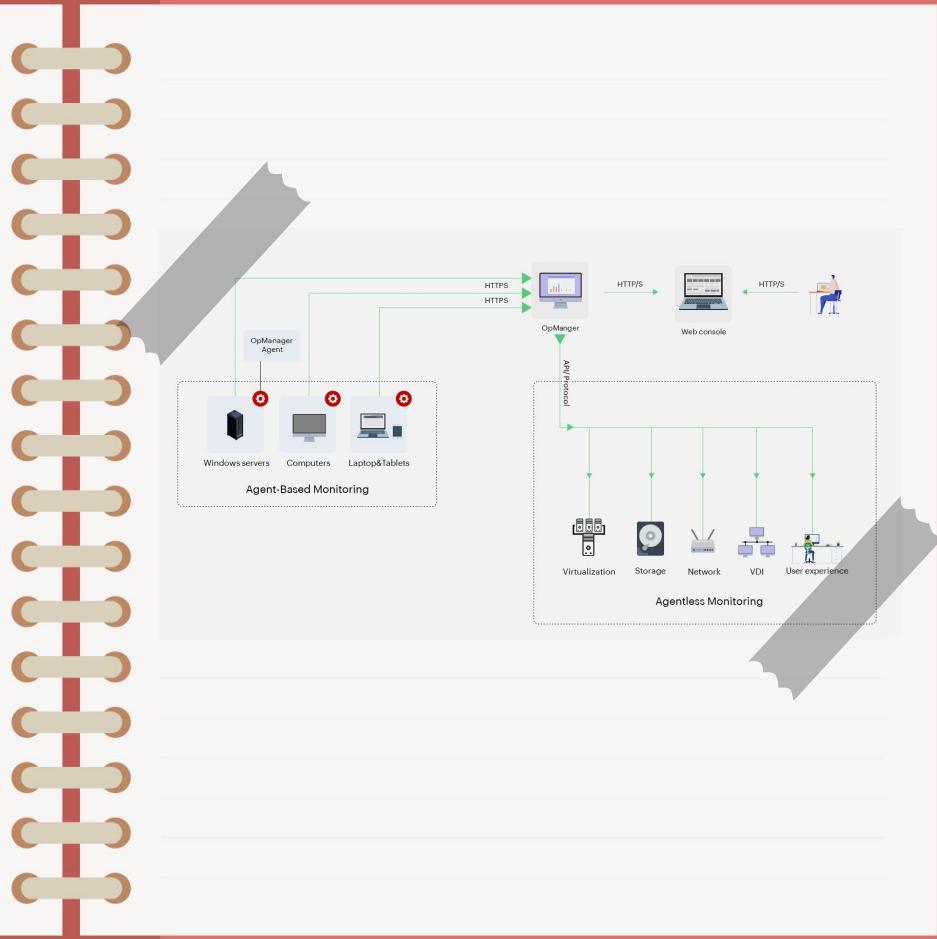
# Problématique des conteneurs

- Les conteneurs sont des boîtes noires
- Les conteneurs sont dynamiques
- La scalabilité induite par les conteneurs devient problématique



# Comment observer

- Commandes exécutées dans le conteneur
- Agents à l'intérieur du conteneur
- Agents à l'extérieur du conteneur



# Les outils de monitoring : Docker stats

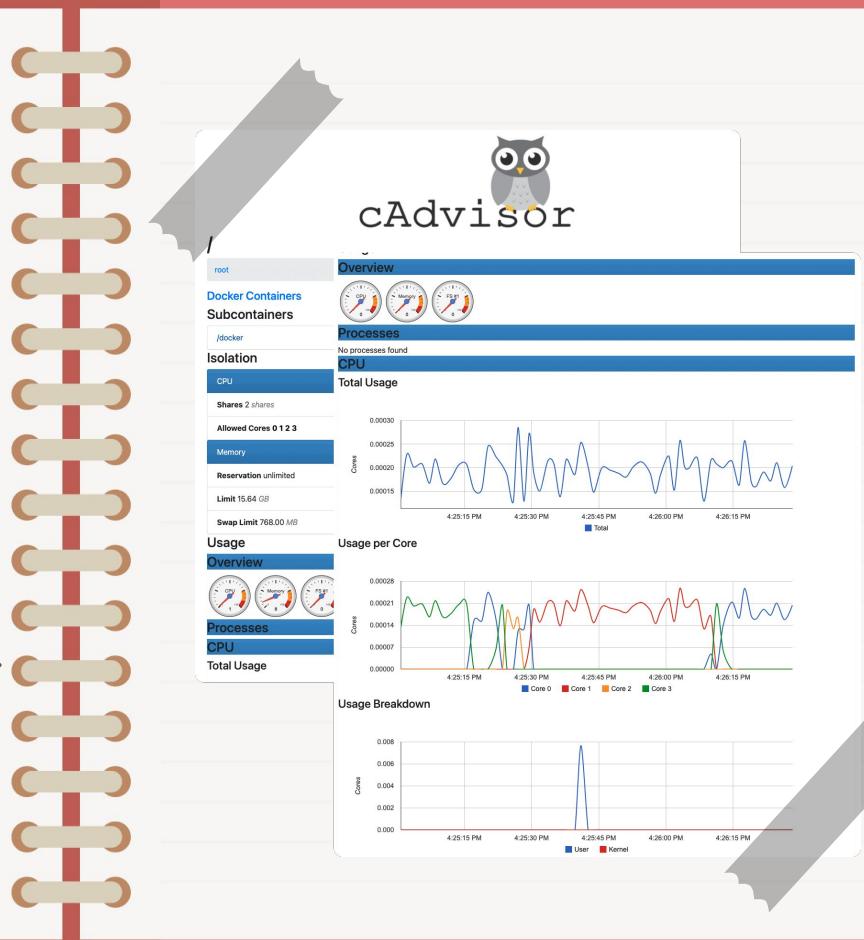
CLI native et très basique qui affiche en continu l'utilisation

- CPU : pourcentage
- MEMOIRE : pourcentage, utilisation et limite
- RESEAU : entrée et sorties
- ...

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
4a5f92e90f36	tp4-php-1	0.01%	4.98MiB / 15.52GiB	0.03%	267kB / 126kB	0B / 0B	3
0224433dd362	tp4-php-2	0.02%	4.926MiB / 15.52GiB	0.03%	148kB / 58.5kB	0B / 0B	3
22ae5042346e	tp4-redis-1	0.24%	101.6MiB / 15.52GiB	0.64%	2.37MB / 81.8kB	0B / 0B	29
c9e2fb06e5e1	tp4-nginx-1	0.00%	5.492MiB / 15.52GiB	0.03%	228kB / 250kB	0B / 0B	5
81f772d4fa5e	tp4-redisinsight-1	0.03%	91.86MiB / 15.52GiB	0.58%	159kB / 3.69kB	0B / 0B	9

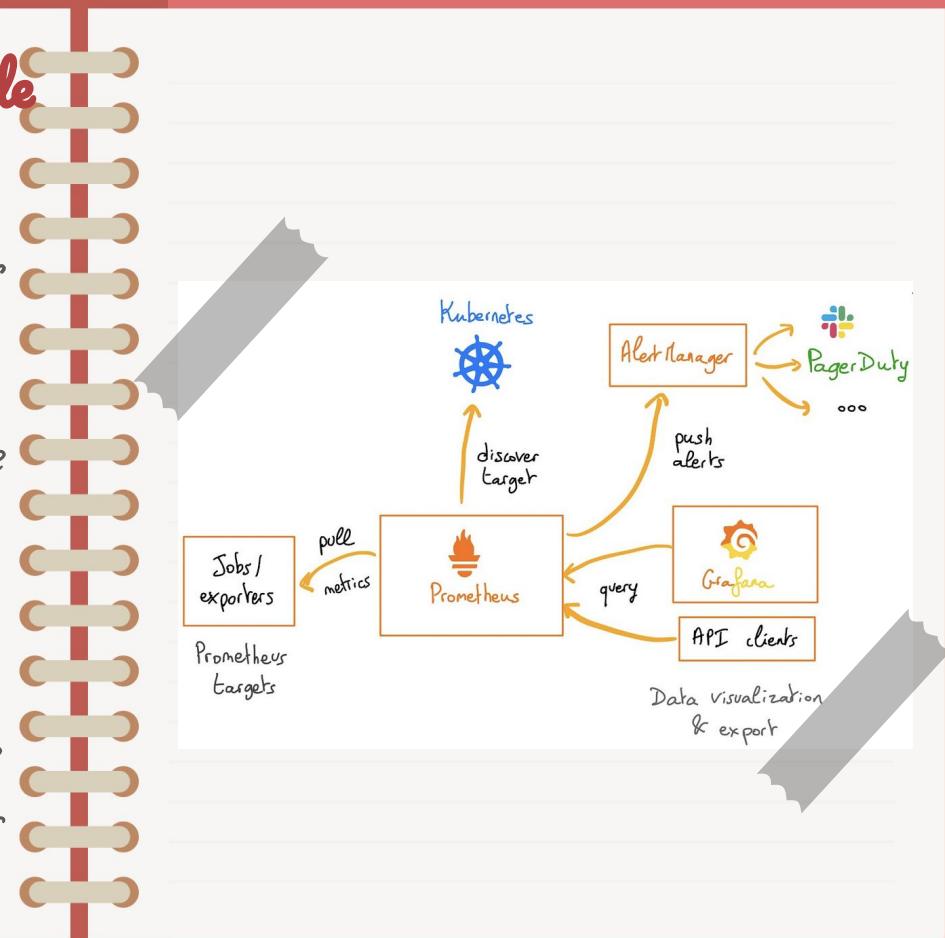
# Les outils de monitoring : cAdvisor

- Conteneur à déployer avec des droits élevés sur chaque hôte
- Dispose d'une interface graphique basique
- En général utilisé conjointement avec d'autres solutions comme Prometheus

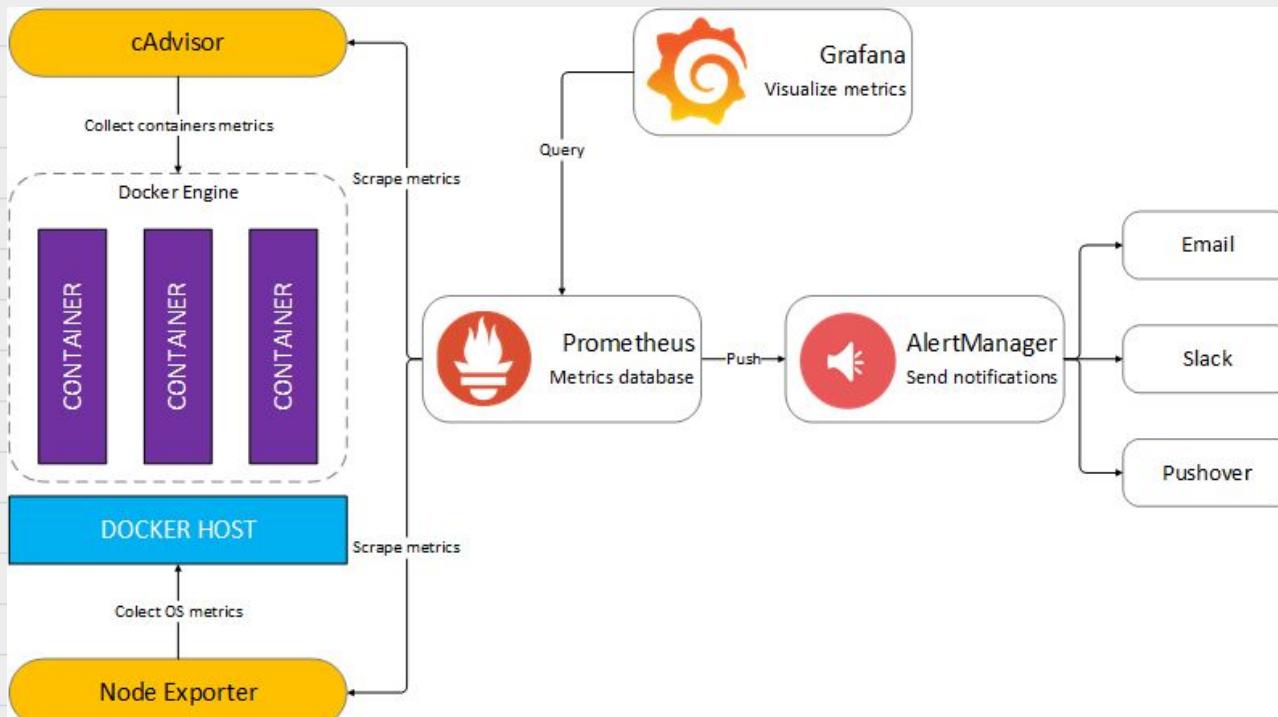


# Les briques principales : autour de Prometheus

- Actuellement la référence pour les métriques
- Problème de scalabilité : montée en puissance de **Victoriameetrics**
- Se base sur le concept d'exporter
- Interface graphique basique /Se repose sur d'autres interfaces comme grafana

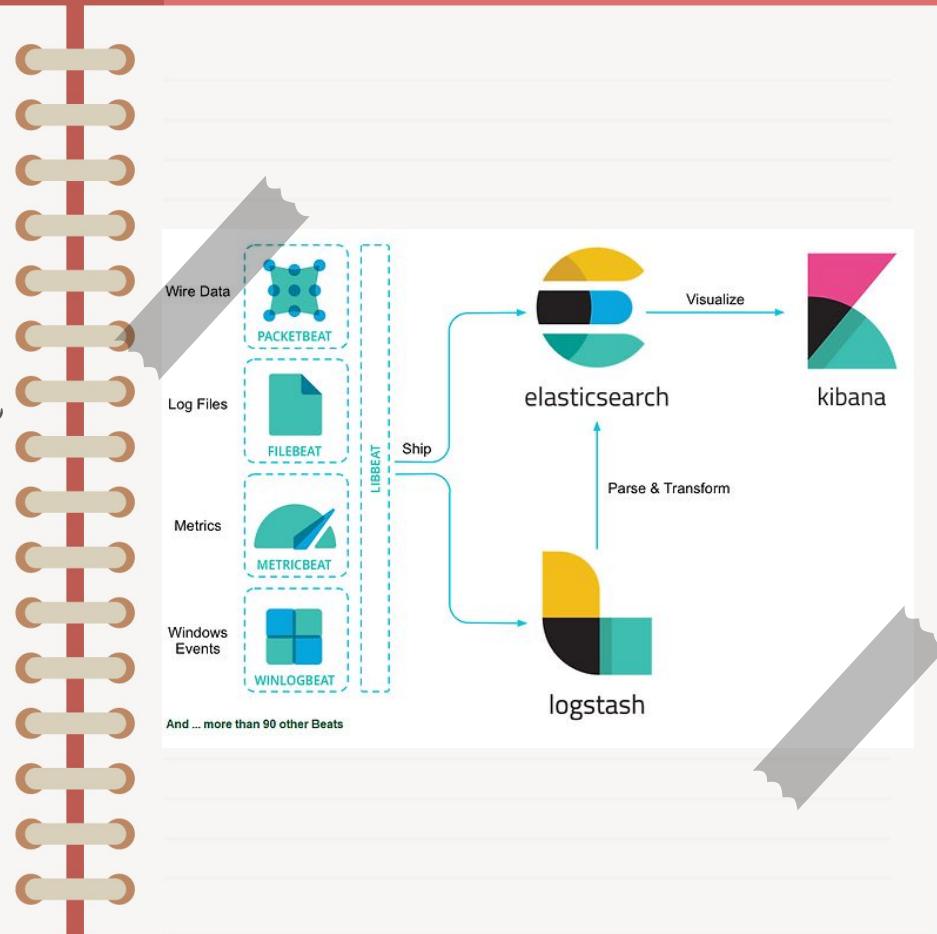


# Les briques principales : autour de Prometheus



## Les outils de collecte de logs : Les Beats

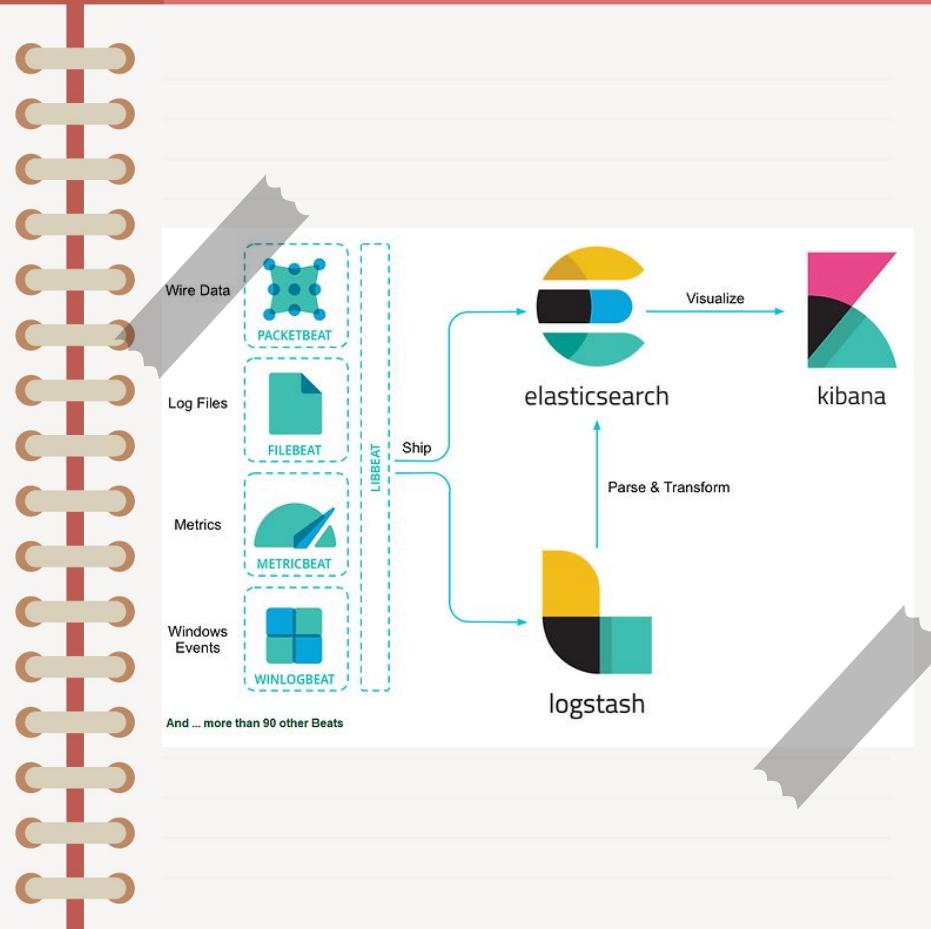
- Collecteurs d'Elastic
- Filebeat, MetricBeat, WinlogBeat, PacketBeat
- Modules et labels (conteneurs)
- Envoi direct vers elastic ou passage par logstash pour transformation / enrichissement



# Les outils de collecte de logs :

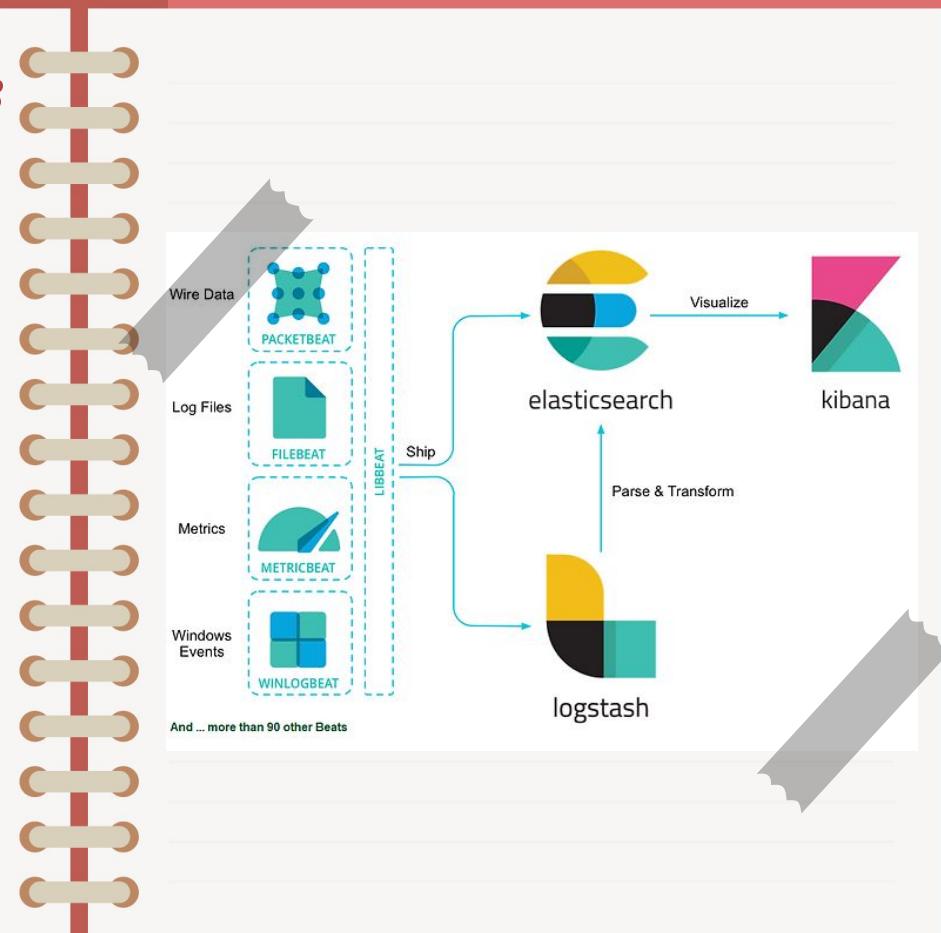
## Les autres

- syslog, rsyslog, syslog-ng
- Fluentbit
- JDK
- ...



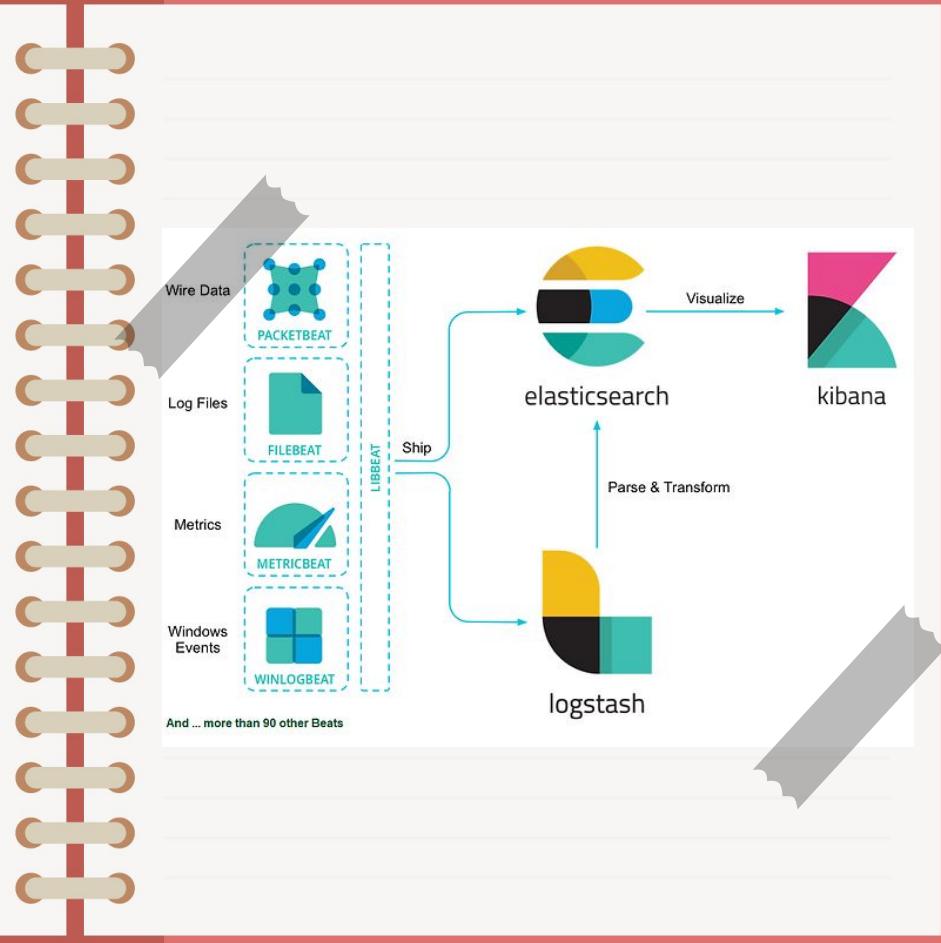
# Stockage et recherche des logs : Elasticsearch

- Cœur d'Elastic
- Se base sur Lucene
- Nécessite une licence pour certaines fonctionnalités : watcher, machine learning
- Dispose d'une API très complète



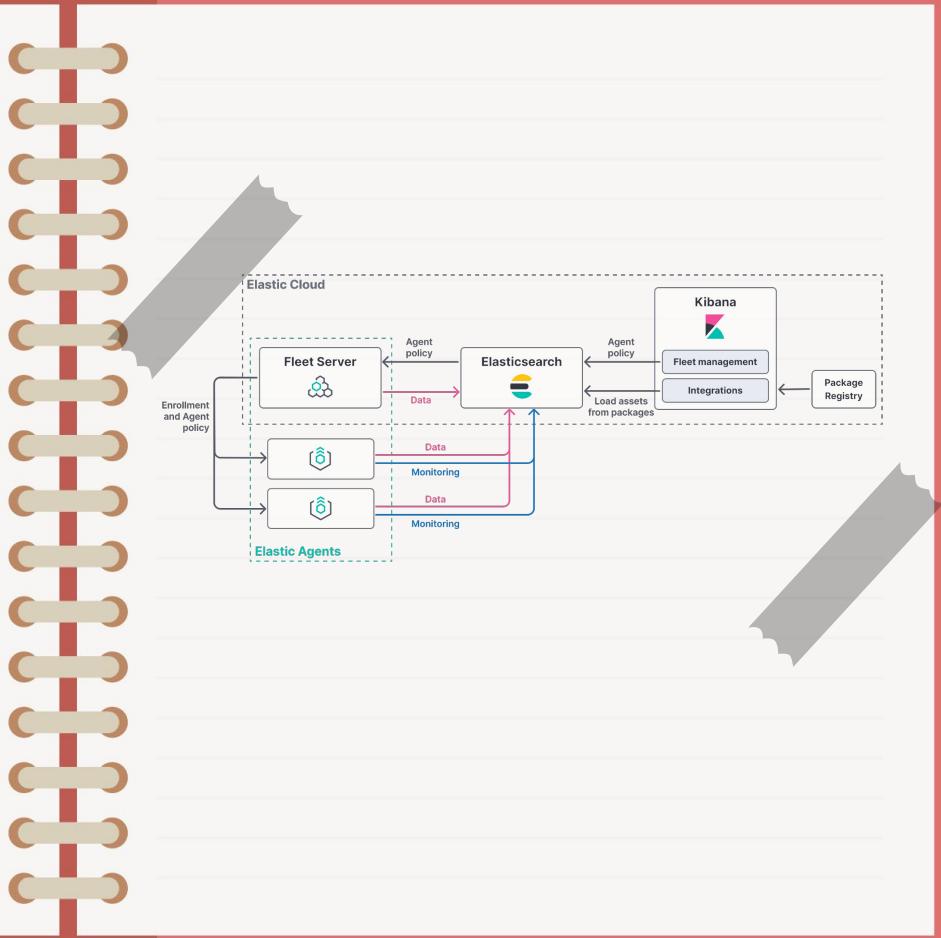
# Visualisation des logs : Kibana

- Console graphique d'Elastic
- Visualisation sous forme de lignes de logs ou graphique
- Rend visuel l'administration du cluster



# Elastic Agent

- Tout dernier venu d'Elastic
- Unifie la collecte avec un seul agent
- Facilite la mise à jour des agents





Du concept à l'expertise

Pour en savoir plus

**<https://elearning.lekpa.fr>**

Martin LEKPA

<https://martin.lekpa.fr>