**DIAPO 3**

## Definir la virtualisation avant de parler conteneurisation

La virtualisation est une technologie qui vous permet de créer plusieurs environnements simulés ou ressources dédiées à partir d'un seul système physique. Son logiciel, appelé hyperviseur, est directement relié au matériel et vous permet de fragmenter ce système unique en plusieurs environnements sécurisés distincts. C'est ce que l'on appelle les machines virtuelles. Ces dernières exploitent la capacité de l'hyperviseur à séparer les ressources du matériel et à les distribuer convenablement. La virtualisation vous aide à tirer le meilleur parti de vos anciens investissements. Le matériel physique doté d'un hyperviseur est appelé « hôte », tandis que toutes les machines virtuelles qui utilisent ses ressources sont appelées « invités ». Ces invités traitent les ressources de calcul (processeur, mémoire, stockage) à la manière d'un pool de ressources qui peut être déplacé sans difficulté. Les opérateurs peuvent contrôler les instances virtuelles du processeur, de la mémoire, du stockage et des autres ressources, de sorte que les invités reçoivent les ressources dont ils ont besoin, lorsqu'ils en ont besoin.

Pour ? Serveur Un seul serveur peut fonctionner comme si vous en aviez deux, ou bien plus.

Réseau Il est possible de créer des réseaux virtuels isolés à partir d'un seul réseau d'origine.

OS Un ordinateur peut exécuter plusieurs systèmes d'exploitation différents.

Avantages : c'est d'optimiser les ressources physiques d'un matériel en les virtualisant.

(Le logiciel de gestion de la virtualisation facilite la gestion de la virtualisation. Vous pouvez parfaitement allouer manuellement des ressources aux machines virtuelles, leur faire de la place sur les serveurs, les tester et installer les correctifs le cas échéant. Mais dès que vous divisez un système unique en plusieurs centaines de systèmes, le volume de travail pour en assurer le bon fonctionnement, la mise à jour et la protection est multiplié d'autant).

On peut migrer facilement la machine sur un autre environnement pour augmenter ses capacités et sa densité.

(Imaginez une ferme de serveurs qui peuvent être réalloués en quelques secondes selon la charge de travail et l'heure. Si une instance invitée se met à consommer plus de ressources que d'habitude, le système de surveillance la déplace sur un autre serveur moins sollicité ou lui alloue plus de ressources à partir d'un pool central).

**DIAPO 4**

Au niveau sécurité : Separation des sytèmes dans un environnement avec système invité isolé sur le système hôte pour la sécurité avec des règles de parefeu

Au niveau programmation : Chaque développeur peut disposer de son propre environnement de test, à l'abri des codes non auorisés ou non contrôlés des autres développeurs

## LIEN ENTRE VIRTUALISATION ET CONTENEURISATION

La virtualisation fournit les ressources que les conteneurs peuvent utiliser. Les machines virtuelles sont des environnements dans lesquels les conteneurs peuvent s'exécuter sans y être forcément liés. Certains logiciels, tels que la solution Red Hat® OpenShift® Virtualization présentée lors du Red Hat Summit 2020, permettent à la fois d'orchestrer des conteneurs et de gérer des machines virtuelles. Toutefois, cela ne signifie pas que ces deux technologies ne font qu'une.

Les machines virtuelles ont des capacités limitées, car les hyperviseurs qui les créent dépendent des ressources finies d'une machine physique. Les conteneurs, quant à eux, partagent le même noyau de système d'exploitation et mettent en paquet chaque application avec son environnement d'exécution de sorte que le paquet final peut être déplacé, ouvert et utilisé aussi bien en développement, qu'en test et en production.
(Conteneur intéressant pour déployer ses applications et les modifier rapidement, clonage très rapide contrairement VM base de données dans des conteneurs directement)

**DIAPO 5**

## LIMITE VIRTUALISATION

Redit du point précédent au niveau des coûts :

Dépendance du système d'exploitation : Les applications déployées sont exécutées uniquement sur des systèmes d'exploitation compatibles

Niveau d'isolement : Incapacité à fournir un sandbox instantané au-dessus du niveau du système d'exploitation (applications)

Granularité de la consommation de calcul : Impossibilité de déployer plusieurs applications répliquées, alors que l'équilibrage de la charge sur la couche applicative ne se produit qu'au sein d'une seule machine et non au niveau de la couche du système d'exploitation.

(charge des ressouces trop importante sur la machine pour répliquer les applications)

Manque de cohérence des environnements : Déploiement d'applications et de packages dans des environnements virtuels

**DIAPO 6 + DIAPO 7**

Redit de ce que j'ai dit précedemment mais qui est important à comprendre :
L'industrie qualifie souvent les conteneurs de légers, ce qui signifie qu'ils partagent le noyau du système d'exploitation de la machine et ne nécessitent pas l'association d'un système d'exploitation à chaque application, comme c'est le cas avec la virtualisation. Par conséquent, les conteneurs ont une capacité intrinsèquement plus petite qu'une machine virtuelle et nécessitent moins de temps de démarrage, ce qui permet l'exécution de plus de conteneurs sur une seule capacité de calcul en tant que machine virtuelle unique. Ainsi, l'efficacité des serveurs est augmentée et les coûts des serveurs et des licences sont réduits.

**DIAPO 8**

**CONTENEURISATION**

Il s'agit d'une forme de virtualisation du système d'exploitation dans laquelle vous exécutez des applications dans des espaces utilisateurs isolés appelés conteneurs qui utilisent le même système d'exploitation partagé. Les conteneurs encapsulent une application en tant que progiciel exécutable qui regroupe le code de l'application et tous les fichiers de configuration, dépendances et bibliothèques nécessaires à son exécution. Les applications conteneurisées sont isolées car elles ne sont pas groupées dans une copie du système d'exploitation. Au lieu de cela, le développeur installe un moteur d'exécution open source (par exemple, le moteur d'exécution Docker) sur le système d'exploitation de l'hôte qui devient l'intermédiaire permettant aux conteneurs de partager un système d'exploitation avec d'autres conteneurs d'applications sur le système informatique.

Un conteneur d'applications est un environnement informatique entièrement regroupé en package et portable :

- Il dispose de tout ce dont une application a besoin pour s'exécuter, y compris ses fichiers binaires, ses bibliothèques, ses dépendances et ses fichiers de configuration, le tout encapsulé et isolé dans un conteneur.

- La conteneurisation d'une application permet d'isoler le conteneur du système d'exploitation hôte, avec un accès limité aux ressources sous-jacentes, à l'instar d'une machine virtuelle légère.

- Vous pouvez exécuter l'application conteneurisée sur différents types d'infrastructure, tels qu'un serveur bare metal, dans le cloud ou sur des VM, sans la remanier pour chaque environnement.

La conteneurisation permet de réduire les charges au démarrage et de supprimer la nécessité de configurer des systèmes d'exploitation invités distincts pour chaque application, car ils partagent tous un seul noyau de système d'exploitation.  En raison de cette efficacité élevée, les développeurs de logiciels utilisent couramment la conteneurisation des applications pour regrouper plusieurs microservices individuels constituant les applications modernes.

La conteneurisation permet aux développeurs de logiciels de créer et de déployer des applications de façon plus rapide et plus sécurisée. Avec les méthodes traditionnelles, vous codez dans un environnement informatique spécifique, ce qui entraîne souvent des erreurs et des bogues lorsque vous transférez ce code dans un nouvel emplacement. Par exemple, lorsque vous transférez du code de votre ordinateur de bureau vers une machine virtuelle, ou d'un système d'exploitation Windows vers Linux.

La conteneurisation élimine ce problème en vous permettant de regrouper le code de la demande avec les fichiers de configuration, les dépendances et les bibliothèques associées. Vous pouvez ensuite isoler ce package de logiciels unique (conteneur) du système d'exploitation hôte, ce qui lui permet d'être autonome et de devenir portable, c'est-à-dire de s'exécuter sur n'importe quelle plate-forme ou n'importe quel cloud sans aucun problème.

Pour faire simple, la conteneurisation permet aux développeurs d'écrire des applications une seule fois et de les exécuter partout. Ce niveau de portabilité est essentiel pour développer la compatibilité des processus et des fournisseurs. Elle présente également d'autres avantages, comme l'isolation des pannes, la sécurité et la facilité de gestion.

**DIAPO 9 + DIAPO 10**

**HISTORIQUE :**

**DIAPO 11 + DIAPO 12**

**DOCKER :**

Fonctionnement de Docker

La technologie Docker utilise le noyau Linux ainsi que ses fonctionnalités (comme les Cgroups et les espaces de noms) pour séparer des processus afin qu'ils s'exécutent de manière indépendante. Cette indépendance reflète l'objectif des conteneurs : exécuter plusieurs processus et applications séparément les uns des autres afin d'optimiser l'utilisation de votre infrastructure tout en bénéficiant du même niveau de sécurité que celui des systèmes distincts.

Les outils de conteneurisation, dont Docker, fournissent un modèle de déploiement basé sur les images pour faciliter le partage d'une application ou d'un ensemble de services ainsi que toutes leurs dépendances dans plusieurs environnements. Docker automatise le déploiement de l'application (ou des ensembles combinés de processus qui la composent) dans l'environnement du conteneur.

Docker repose sur une architecture client-serveur. Un client Docker communique avec le Docker daemon pour gèrer la construction et le fonctionnement des conteneurs Docker ainsi que la distribution des images Docker issues d'un Docker Registry, public ou privé.

Parce qu'ils sont créés à partir de conteneurs Linux, ces outils Docker sont faciles à utiliser et uniques. Ils offrent aux utilisateurs un accès sans précédent aux applications, un déploiement rapide et un contrôle des versions et de leur distribution.

Le Docker Engine est devenu une norme de l'industrie en ce qui concerne le processus de conteneurisation grâce à une approche de packaging universelle pour des applications et à des outils simples pour les développeurs.

Lorsqu'un Docker accède à un noyau de système d'exploitation unique, il peut gérer plusieurs applications distribuées exécutées dans leurs conteneurs respectifs. La base de la conteneurisation est le progiciel que les développeurs mettent en œuvre dans un transport virtuel unique.

Les développeurs créent des conteneurs à partir d'images Docker. Malgré leur statut de lecture seule, le Docker crée un conteneur en ajoutant un système de fichiers en lecture/écriture. Il lance une interface réseau pour permettre la communication entre le conteneur et un hôte local. Ensuite, il

ajoute une adresse IP et exécute le processus indiqué. Chaque conteneur contient les éléments nécessaires à l'exécution d'un programme (fichiers, redondances et bibliothèques).

**DIAPO 13 + DIAPO 14**

**KUBERNETES :**

Kubernetes = Kubernetes (« k8s » ou « kube ») est une plateforme Open Source d'orchestration des conteneurs qui automatise de nombreux processus manuels associés au déploiement, à la gestion et à la mise à l'échelle des applications conteneurisées.

Kubernetes vous aide à gérer facilement et efficacement des clusters au sein desquels vous aurez rassemblé des groupes d'hôtes exécutant des conteneurs Linux®.

Système open source offrant une platforme pour automatiser le deploiement, la montée en charge et la mise en oeuvre de conteneurs d'applications sur les grappes de serveur. Développé par Google puis offert à la CNCF (cloud native computing foundation

Avec Kubernetes, vous pouvez réaliser les tâches suivantes :

- Orchestration des conteneurs sur différents hôtes
- Améliorer l'utilisation du matériel de manière à optimiser les ressources nécessaires pour exécuter les applications d'entreprise (scalablité)
- Contrôle et automatisation des déploiements et mises à jour des applications
- Montage et ajout de stockage pour l'exécution d'applications stateful (stockage de données en temps réel)
- Mise à l'échelle des applications conteneurisées et de leurs ressources à la volée
- Gestion déclarative des services pour garantir le bon fonctionnement des applications déployées
- Contrôle de l'intégrité et autoréparation des applications grâce au placement, au redémarrage, à la réplication et à la mise à l'échelle automatiques

Vocabulaire :

- **Plan de contrôle :** ensemble de processus qui contrôle les nœuds Kubernetes et assigne toutes les tâches.

- **Nœuds :** machines qui exécutent les tâches qui leur sont assignées par le plan de contrôle.
- **Pod :** groupe d'un ou de plusieurs conteneurs déployés sur un seul nœud. Tous les conteneurs d'un pod partagent une même adresse IP, un même IPC, un même nom d'hôte et d'autres ressources. Les pods permettent de dissocier le réseau et le stockage du conteneur sous-jacent. Ainsi, vous pouvez déplacer vos conteneurs au sein du cluster très simplement.
- **Contrôleur de réplication :** composant qui vérifie le nombre de copies identiques d'un pod qui doivent s'exécuter quelque part dans le cluster.
- **Service :** élément qui dissocie les définitions de tâche des pods. Les proxies de service Kubernetes reçoivent automatiquement des demandes de service dans le bon pod, même s'il a été déplacé dans le cluster ou s'il a été remplacé.
- **Kubelet :** service exécuté sur des nœuds qui lit les manifestes du conteneur pour s'assurer que les conteneurs définis ont démarré et fonctionnent.

**kubectl :** outil de configuration des lignes de commande pour Kubernetes.

Utilisation possible :

Vous pouvez utiliser Docker en tant qu'environnement d'exécution orchestré par Kubernetes. Lorsque Kubernetes planifie un pod dans un nœud, le kubelet de ce nœud donne l'ordre à Docker de lancer les conteneurs spécifiés.

- Le kubelet collecte ensuite en continu le statut de ces conteneurs via Docker et rassemble ces informations dans le plan de contrôle. Docker transfère ces conteneurs dans ce nœud, les démarre et les arrête.
- Lorsque vous utilisez Kubernetes avec Doker, la différence est l'origine des ordres : ils proviennent d'un système automatisé et non plus d'un administrateur qui assigne manuellement des tâches à tous les nœuds pour chaque conteneur.

**DIAPO 15 + 16**

**AVANTAGES :**

**L’accélération des développements** = le développeur travaille dans un cadre restreint à son strict nécessaire ce qui lui épargne le codage de tests d’interactions notamment. Cela favorise aussi la création de **bacs à sable** et donc une montée en compétence et une **capacité d’innovation** plus rapides.

Facilité d'utilisation pour les développeurs = Les conteneurs sont conviviaux pour les développeurs, car il est possible d'utiliser un seul environnement pour le développement et la production, un obstacle courant dans le développement d'applications web. Votre équipe de développement peut écrire une app sur un

ordinateur portable Windows, mais celle-ci ne s'exécute pas sur un poste de travail Mac.

La **portabilité** et donc **l'accélération des déploiements** = le conteneur créé est cohérent et ne souffrira pas d'être exécuté sur un autre environnement, tout en étant moins gourmand qu'une VM et peut donc être plus **facilement déplacé, copié, relancé**.

**L'impact moindre sur les performances du serveur**, un conteneur pouvant libérer rapidement les ressources (mémoire, stockage) inutilisées.

Fin de dépendance de programme pour les applications = Pour les applications, il n'est plus nécessaire de tenir compte des dépendances d'un programme, simplifiant donc son évolution et la gestion de ses versions. Les applications ne sont également plus directement impactées par les changements d'environnements. Les opérations de maintenance vont aussi bénéficier de la conteneurisation. Le déploiement des mises à jour et des correctifs ne devront être effectués qu'une seule fois puisque le noyau du système d'exploitation est désormais commun.

Gestion de la scalibilité = La scalabilité et la gestion de la montée en charge est le principal avantage de Kubernetes. Pour illustrer avec un exemple : si vous avez un site web reposant sur un serveur d'application avec une architecture classique de VM (machine virtuelle) avec un load balancer, pour gérer un pic d'utilisateurs, vous devez ajouter une nouvelle VM avec ce même serveur d'application installé. Désormais avec Kubernetes, il peut ajouter automatiquement une nouvelle machine sans se soucier de l'OS ou du soft. Cette partie étant gérée au niveau du conteneur.

Création facile d'image via Docker = L'utilisation d'un conteneur Docker vous permet de créer une version principale d'une application (image) et de la déployer rapidement sur demande. Un environnement conteneur garantit une grande flexibilité lorsque vous souhaitez créer plusieurs nouvelles instances conteneurisées d'applications à la demande.

Continuité =

Un environnement conteneur permet l'ajout de nouvelles fonctions, mises à jour et caractéristiques instantanément sans que cela affecte les applications d'origine. Par conséquent, les conteneurs permettent l'évolutivité des applications avec une utilisation des ressources minimale.En utilisant une plate-forme d'orchestration des conteneurs, vous pouvez automatiser l'installation, la gestion et l'évolution des charges de travail et des services conteneurisés. L'orchestration des conteneurs permet de faciliter les tâches de gestion, comme le déploiement de nouvelles versions d'applications, l'évolution d'applications conteneurisées ou la mise à disposition de fonctions de surveillance, de consignation et de débogage.

Isolation d'application pour la sécurité = La conteneurisation d'une application permet de l'isoler et de la faire fonctionner de façon indépendante. Par conséquent, la défaillance d'un conteneur n'affecte pas le fonctionnement des autres. Les équipes de développement peuvent rapidement identifier et corriger les problèmes techniques d'un conteneur défectueux sans provoquer l'arrêt du reste des conteneurs. En outre, le

moteur de conteneur peut tirer parti de techniques d'isolation de la sécurité du système d'exploitation comme le contrôle d'accès SELinux (controle dacces fichiers syteme linux) pour identifier et isoler les pannes au sein des conteneurs.

L'isolation des applications via les conteneurs empêchent le code malveillant d'affecter d'autres applications conteneurisées ou le système hôte. Vous pouvez également définir des autorisations de sécurité pour bloquer automatiquement l'accès aux composants indésirables qui cherchent à s'introduire dans d'autres conteneurs ou à limiter les communications.

L'isolation des applications aide les développeurs à partager des fonctionnalités supplémentaires sans facteur de risque. Par exemple, si vous travaillez avec une équipe de développement à l'extérieur de votre réseau, vous pouvez partager les ressources nécessaires sans que les informations critiques ne se trouvent dans votre réseau.

Facilité de la gestion =

En utilisant une plate-forme d'orchestration des conteneurs, vous pouvez automatiser l'installation, la gestion et l'évolution des charges de travail et des services conteneurisés. L'orchestration des conteneurs permet de faciliter les tâches de gestion, comme le déploiement de nouvelles versions d'applications, l'évolution d'applications conteneurisées ou la mise à disposition de fonctions de surveillance, de consignation et de débogage.

**DIAPO 17**

**INCONVENIENTS :**

Lourd travail à mettre en place = En augmentant la flexibilité des applications, la conteneurisation apporte de la complexité de différentes manières. Cette complexité peut être liée à la sécurité, à l'orchestration, à la surveillance et au stockage des données.

Changement d'OS difficile = Un conteneur Linux ne peut pas être utilisé sur une machine Windows (et inversement) à moins d’une couche intermédiaire de virtualisation/émulation de l’OS correspondant.

Une sécurité pas infaillible = Des cas de failles d’une application conteneurisée ont permis d’atteindre le kernel de la machine (plusieurs cas dans le cloud) et de mettre en danger tous les conteneurs portés par ce serveur (physique ou virtuel).

Par rapport aux VM traditionnelles, les conteneurs présentent un risque de sécurité potentiellement plus important. Ils ont besoin d'une sécurité multiniveau, car ils ont plusieurs couches. Par

conséquent, vous devez sécuriser l'application conteneurisée ainsi que le registre(emplacement de memoire interne differents registres qui font des calculs arithmétiques), le daemon (processus arriere plan) Docker et le système d'exploitation hôte.

- Orchestration des conteneurs limitée: vous pouvez utiliser un seul outil d'orchestration(processus automatique organisation, gestion et coordination de services et middleware = logiciel tiers echange d'infos entre applications) pour les machines virtuelles, fourni avec une solution virtualisée (par exemple, un outil d'orchestration VMware pour VMware). Toutefois, lorsqu'il s'agit de conteneurs, vous devez choisir parmi différents outils d'orchestration comme Kubernetes, Mesos ou Swarm.

- Risque de pert de données = Le stockage des données sur les VM est simple, mais se complexifie lorsqu'il s'agit de conteneurs. Pour les données persistantes du conteneur, vous devez les déplacer hors du conteneur d'application vers le système hôte ou vers un endroit disposant d'un système de fichiers persistant. La conception des conteneurs est à l'origine de la perte de données. En effet, les données du conteneur peuvent disparaître à jamais une fois celui-ci supprimé, à moins d'en faire une sauvegarde ailleurs.

- Supersivison accrue nécessaire : il est également essentiel de surveiller les conteneurs pour détecter les problèmes de performance et de sécurité. Vous pouvez utiliser différents outils de surveillance, services de surveillance externes et analyses pour faire face à ce problème. L'environnement cloud étant complexe, vous devez surveiller de près les problèmes de sécurité.

Néanmoins, les avantages de la conteneurisation l'emportent largement sur les inconvénients. Par conséquent, la décision concernant la nécessité des conteneurs dépendra uniquement de vos besoins spécifiques en matière de stockage dans le cloud.

**DIAPO 18**

**ALTERNATIVES**

D’autres solutions facilitant la conteneurisation existent bien évidemment, sous Linux comme sous Windows, FreeBSD ou Solaris :

- Virtualisation
- LXC (la base historique de la conteneurisation sous Linux),
- Rocket (rkt) de CoreOs,
- Windows Hyper-V Containers (qui s’apparente à des VM légères),
- Oracle Solaris Zones

FreeBSD

Il est à noter que la conteneurisation n’exclut pas la **virtualisation machine** : les deux méthodes peuvent être mixées en fonction des besoins.

Une **machine virtuelle** peut accueillir des conteneurs si cela facilite la gestion et la sécurité de votre organisation par environnement ou domaine.

**DIAPO 19**

**OUTILS**

**Docker est une surcouche qui rend le developpement et le deploiement des conteneurs beaucoup plus simple tout en les standardisant. La 1ere version docker 1.0 date de 2014 mais son adoption par les plus grands compte de l'informatique en fait deja un outil de reference**

Kubernetes est un systeme open source permettant d'automatiser le deploiement, la mise a lechelle et la gestion des applications conteneurisées

Openshift est une plateforme de conteneurs d'applications open source, princialement basée sur docker et orchestrée à l'aide de la gestion des clusters de conteneur Kubernetes qui est chargé du demarrage et du scaling (rajouter des charges sans pour autant nuire aux performances) des conteneurs

Il propose des outils de configuration plus poussés comme construire des images, intégrer un registre privé paramétrable pour séparer les images par projet, proposer une interface utilisable par un non administratueur, proposer des routeurs vers les applications, gerer les droits/restritions/authentification etc..

**DIAPO 20**

**HISTORIQUE**

Tout au long des années 1960, de multiples terminaux informatiques étaient reliés à un seul ordinateur central, ce qui permettait de centraliser l'informatique. Les ordinateurs centralisés permettaient de contrôler tous les traitements à partir d'un seul endroit, de sorte que si un terminal tombait en panne, l'utilisateur pouvait simplement se rendre sur un autre terminal, s'y connecter et avoir encore accès à tous ses fichiers. Par exemple, si l'utilisateur devait planter l'ordinateur central, le système s'arrêterait pour tout le monde. Des problèmes de ce genre ont montré que les ordinateurs

devaient être capables de séparer non seulement les individus mais aussi les processus du système.

En 1979, nous avons fait un pas de plus vers la création d'environnements partagés, mais isolés, avec le développement de la commande chroot (change root). La commande chroot permet de modifier le répertoire racine apparent d'un processus en cours d'exécution, ainsi que tous ses enfants. Cela permettait d'isoler les processus système dans leurs propres systèmes de fichiers distincts afin que les tests puissent avoir lieu sans avoir d'impact sur l'environnement système global. **En mars 1982, Bill Joy a ajouté la commande chroot à la 7e édition d'Unix.**

Pour comprendre les conteneurs, nous pouvons avancer un peu dans le temps, jusqu'aux années **1990, lorsque Bill Cheswick**, un chercheur en sécurité informatique et en réseaux, s'efforçait de comprendre comment un pirate utiliserait son temps s'il avait accès à son système Dans cette recherche, Cheswick a construit un environnement qui lui a permis d'analyser les frappes du pirate afin de le tracer et d'apprendre ses techniques. Sa solution a consisté à utiliser un environnement chrooté et à y apporter des modifications. Le résultat de ses études est ce que nous connaissons aujourd'hui comme **la commande jail de Linux.**

**Le 4 mars 2000, FreeBSD a introduit la commande jail dans son système d'exploitation.** Bien qu'elle soit similaire à la commande chroot, elle comprenait également des fonctionnalités supplémentaires de sandboxing des processus pour isoler les systèmes de fichiers, les utilisateurs, les réseaux, etc. FreeBSD jail nous a donné la possibilité d'attribuer une adresse IP, de configurer des installations logicielles personnalisées, et d'apporter des modifications à chaque jail. Cela n'était pas sans poser quelques problèmes, car les applications à l'intérieur de la prison étaient limitées dans leurs fonctionnalités.

**En 2001, la technologie des conteneurs est passée du côté de Linux. Jacques Gélinas a créé le projet VServer**, qui, selon le journal des modifications de la version 0.0, permettait "d'exécuter plusieurs serveurs Linux d'usage général sur une seule boîte avec un haut degré d'indépendance et de sécurité".

**En 2004,** nous avons assisté à la sortie des conteneurs Solaris, **qui ont permis de créer des environnements d'application complets grâce à l'utilisation de Solaris Zones.** Avec les zones, vous pouvez donner à une application tout l'espace utilisateur, processus et système de fichiers, ainsi que l'accès au matériel du système.  Toutefois, l'application ne peut voir que ce qui se trouve dans sa propre zone. En 2006, les ingénieurs de Google ont annoncé le lancement de conteneurs de

processus conçus pour isoler et limiter l'utilisation des ressources d'un processus. En 2007, ces conteneurs de processus ont été rebaptisés groupes de contrôle (cgro**ups) pour éviter toute confusion avec le mot conteneur.**

C'est l'approche de **Paul Menage en 2006**, **consistant à adapter le mécanisme cpusets déjà présent dans le noyau principal**, qui a réellement fait avancer la conteneurisation sur Linux, en exigeant des changements peu intrusifs avec un faible impact sur les performances, la qualité du code, la complexité et la compatibilité future.

Le résultat était des conteneurs de processus génériques, qui ont été plus tard renommés groupes de contrôle, ou cgroups, pour refléter le fait que "ce code est une partie importante d'une solution de conteneur... il est loin d'être la chose entière." Les cgroups permettent de regrouper des processus et de s'assurer que chaque groupe obtient une part de mémoire, de CPU et d'E/S de disque ; empêchant ainsi tout conteneur de monopoliser l'une de ces ressources.
**En 2008, les cgroups ont été intégrés au noyau Linux 2.6.24**, ce qui a conduit à la création du projet que nous connaissons aujourd'hui sous le nom de LXC. LXC est l'acronyme de Linux Containers et fournit une virtualisation au niveau du système d'exploitation en permettant à plusieurs environnements Linux isolés (conteneurs) de fonctionner sur un noyau Linux partagé. Chacun de ces conteneurs dispose de son propre processus et de son propre espace réseau.

En 2013, Google a changé une fois de plus les conteneurs en mettant en libre accès sa pile de conteneurs sous la forme d'un projet appelé Let Me Contain That For You (LMCTFY). Grâce à LMCTFY, les applications pouvaient être écrites de manière à tenir compte des conteneurs et donc être programmées pour créer et gérer leurs propres sous-conteneurs. **Le travail sur LMCTFY a été arrêté en 2015 lorsque Google a décidé de contribuer les concepts de base derrière LMCTFY au projet Docker libcontainer.**

L'essor de DockerDocker a été publié en tant que projet open-source en 2013. Docker offrait la possibilité d'empaqueter des conteneurs afin de pouvoir les déplacer d'un environnement à un autre. Au départ, Docker s'appuyait sur la technologie LXC, mais en 2014, LXC a été remplacé par libcontainer, qui permettait aux conteneurs de fonctionner avec les espaces de noms Linux, les groupes de contrôle libcontainer, les capacités, les profils de sécurité AppArmor, les interfaces réseau et les règles de pare-feu. Docker a poursuivi ses contributions à la communauté en incluant des registres de conteneurs globaux et locaux, une API reposante et un client CLI. **Plus tard,**

**Docker a mis en œuvre un système de gestion de cluster de conteneurs appelé Docker Swarm.**

SELinux est le système de contrôle d'accès obligatoire (MAC) développé par un consortium d'entreprises et d'agences gouvernementales qui est très efficace pour "étiqueter les processus, les fichiers et les périphériques [et] définir des règles sur la façon dont les processus étiquetés interagissent avec les processus, les fichiers et les périphériques étiquetés". **Les conteneurs Linux, en tant que groupe de processus, se prêtent bien au durcissement avec SELinux.**

Malgré une longue histoire d'innovations progressives, les récentes avancées de Linux autour des conteneurs Linux sont en train de révolutionner la manière dont les entreprises vont développer, consommer et gérer les applications. Comme pour les applications traditionnelles, les applications conteneurisées interagissent avec le système d'exploitation et en dépendent.

Les innovations futures en matière de conteneurisation des applications s'appuieront sur les améliorations progressives du système d'exploitation Linux, comme elles l'ont toujours fait. La stratégie de conteneurs de Red Hat aide l'informatique à fournir des applications pour accélérer l'agilité de l'entreprise, la productivité des développeurs et la flexibilité du déploiement dans les environnements de cloud hybride.

**DIAPO 21**

**EXEMPLES :**

1. La DockerCon 2017 a été l'occasion pour **Paypal** de présenter son cheminement dans la transformation de ses applications pour les rationaliser et améliorer leur disponibilité. C'est ainsi que 150 000 containers ont émergé en 2 ans de travail pour moderniser leurs applications de cœur de métier au profit de leurs clients.

2. **Blablacar** a démontré comment la conteneurisation alliée à une bonne orchestration permettent des gains de plusieurs jours dans le cycle de vie d'une application, là encore dans un contexte de croissance exponentielle. Par ailleurs, un retour d'expérience est très attendu sur les bénéfices de l'élasticité de la combinaison conteneur-cloud dans un contexte de crise

comme celui de la crise sanitaire vécu de plein fouet par le monde du transport.

3. **Docker** a mis en avant les propositions de Netflix pour conforter son usage de ces technologies en apportant des solutions visant à sécuriser ses conteneurs, en rationaliser l'utilisation et assurer leur meilleure interopérabilité.

En 2014, Joe Beda de Google qui a été le pionnier de Kubernetes a annoncé que Google démarrait 2 milliards de conteneurs toutes les 2 semaines. Ce n'est pas surprenant compte tenu de l'échelle à laquelle Google opère. Au fait, Kubernetes est un cadre permettant d'orchestrer des conteneurs lorsque vous devez les faire évoluer à l'échelle de Google ou à une échelle similaire.

**LEXIQUE :**

**- Hyperviseur** = logiciel qui permet à plusieurs OS de travailler sur une même machine physique (partage les ressources du système hote = mémoire, capacités de traitement)

**- Environnement** = Pour une application c'est l'ensemble de logiciels (dont l'OS) et materiels sur lesquels sont éxécutés les programmes = x environnements pour developper, tester, se former ou executer des programmes

**- Système** = ensemble des ressources hardware et software

- **Run time** = ensemble de fichiers qui permet le bon fonctionnement de l'applicaiton en allouant de la mémoire en utilisant le sous systeme dexploitation (systeme fichier, systeme réseau..)

**- Pool central** = ensemble de ressources réutilisables (par exe agregation de disques durs physiques en 1 logique)

- **Sandbox** = effectuer des tests d'execution d'un programme sans risquer le péril de la machine

- **Granularité** = définit la taille du plus petit élément

**- Couche applicative** = types de messages échangés msg demande et réponse, syntaxe des messages et la maniere de champs, la sémantique donc la signification des champs, règles pour un process de quand et comment il envoie un msg et répond.

- **Bare metal** = serveur administré repose directement sur le materiel, applis direct éxécutés sur OS

- **API** = interface de programmation qui permet à des applications de communiquer, d'échanger entre elles des services ou des donner

- **Progiciel** = (ensemble de programme, generalement dédié clientele)

- **Chroot** = permet de changer le repertoire racine d'un processus de la machine hôte (prémices virtualisation).

Chroot permet également de faire tourner plusieurs processus démons sur la mâchine hôte (qui tournent en arriere plan)

- **Jail** = sorte de vm avec une partition du système dédiée, compte utilisateurs, arborescence fichiers.. l'user root à l'intérieur n'a qu'un accès administrateur et ne peut accèder à rien d'autre que ce propre environnement (prémices virtualisation côté sécurité)

- **LXC** = Linux Containers, système de virtualisationutilisant l'isolation comme méthoe de cloisonnement au niveau de l'OS. Le conteneur apporte une virtualisation de l'environnement d'execution (processeur, memoire vive, reseau, système de fichiers..) et non pas de la machine. Pour cette raison on parle de conteneur et non de machine virtuelle.

- **LMCTFY** (let me contain that for you) = logiciel libre license Apache, reprend LXC et début Docker.

- **Application stateful** = stockage de données en temps réel (genre banque ou messagerie si ca bug serveur te remet où c'était)

- **Cgroups** = groupes de contrôle linux pour limiter, compter et isoler l'utilisation des ressources

- **Plan de contrôle =** ensemble de processus qui contrôle les nœuds Kubernetes et assigne toutes les tâches.

- **Nœuds =** machines qui exécutent les tâches qui leur sont assignées par le plan de contrôle.

- **Pod =** groupe d'un ou de plusieurs conteneurs déployés sur un seul nœud. Tous les conteneurs d'un pod partagent une même adresse IP, un même IPC, un même nom d'hôte et d'autres ressources. Les pods permettent de dissocier le réseau et le stockage du conteneur sous-jacent. Ainsi, vous pouvez déplacer vos conteneurs au sein du cluster très simplement.

- **Contrôleur de réplication =** composant qui vérifie le nombre de copies identiques d'un pod qui doivent s'exécuter quelque part dans le cluster.

- **Service =** élément qui dissocie les définitions de tâche des pods. Les proxies de service Kubernetes reçoivent automatiquement des demandes de service dans le bon pod, même s'il a été déplacé dans le cluster ou s'il a été remplacé.

**- Kubelet =** service exécuté sur des nœuds qui lit les manifestes du conteneur pour s'assurer que les conteneurs définis ont démarré et fonctionnent

**- Kubectl =** outil de configuration des lignes de commande pour Kubernetes.

Technologie apparente au PAAS

**- Paas** (platforme as a service) = platforme proposée par fournisseur via internet, permet aux equipes de developper des applis et logiciels sans assurer la maintenance. De grandes variétés de langage de programmation sont mise à disposition

**- Saas** (software as a service) = service cloud le plus répandu. Logiciel fonctionnant sur l'infrastructure d'un prestataire. L'user paie la license mais ne s'ocuppe ni du stockage des données ni de l'entretien du matériel physique

**- Iaas** (infrastructure as a service) = ensemble de ressources informatiques proposées par un hebergeur pour virtualiser : big data, hebergement, machine learning

**- Grappe de serveurs** = ferme, cluster, groupement ou noeud

**- Cloud Native** = gestion automatisée qui permet de mettre en place des applications évolutives dans des environnements cloud hybride/public ou privé.

**- Cloud privé** = dans une entreprise, cloud public prestataire qui offre des hebergements

**- L'intégration** = (un serveur qui cherche à s'adapter à differents OS)

**- Les services** = (qui veillent au bon fonctionnement de la machine, protocoles web, transfert fichiers, performance, acc-s distant, ADirectory)

**- SELinux** = (controle dacces fichiers syteme linux)

**- Workflow** = (suite et modelisation de tâches)

**- Pipelines** = (compilateur pour optimiser les boucles)

**- CI/CD** = (outil pour automatiser le deploiement et les tests)

**- Registre** = (emplacement de memoire interne differents registres qui font des calculs arithmétiques)

**- Daemon** = (processus arriere plan)

**- Middleware** = logiciel tiers echange d'infos entre applications

- **Rocket de CoreOS** = open source alternative direct à Docker, plus sécurisé (conteneurs chiffrés des la creation) et modulable (conteneurs interconnectés mais indépendants).

- **Hyper-v** = création d'environnement informatique virtuel pour gerer et executer plusieurs OS sur un seul serveur physique (crée disque dur virtuel, commutateur virtuel, periph virtuels..)

- **Oracle Solaris Zone** = environnement virtualisé séparé de l'hardware qui permet de gerer, modifier les ressources

- **Free BSD** = OS Unix open source

- **Scalabilité** = adapter les performances du matériel en fonction de la demande

## OUTILS / SOURCES :

- Extension Abonnement RSS (par Google)
- Feedly Rss, Twitter abonnements Docker et Kubernetes

- recherche site [redhat, veritas, wikipedia, alibabacloud, acloudguru, techtarget,section.io,appvizer,sqorus,wiki ubuntu, stackoverflow] (platforme présentations et gudie solutions IT pour entreprise)
- vidéos discussion autour de c'est quoi Docker, c'est quoi Kubernetes (redhat)
- vidéo cookie connecté comprendre l'essentiel de Docker
- ANSSI recommandation sécurité déploiement conteneur Docker (gouv.fr)
- google scholar