Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE
MINISTÈRE DE L'INTÉRIEUR

CONCOURS EXTERNE DE TECHNICIEN DE CLASSE NORMALE DES SYSTEMES D'INFORMATION ET DE COMMUNICATION

- SESSION 2019 -

Mercredi 20 mars 2019

Option « Solutions logicielles et systèmes d'information »

Traitement de questions et résolution de cas pratiques, à partir d'un dossier, portant sur l'une des deux options choisies par le candidat le jour de l'épreuve :
- infrastructures et réseaux
- solutions logicielles et systèmes d'information.

Cette épreuve permet d'évaluer le niveau de connaissances du candidat, sa capacité à les ordonner pour proposer des solutions techniques pertinentes et à les argumenter.

Le dossier ne peut excéder 20 pages.

(Durée : 3 heures – Coefficient 2)

L'usage de la calculatrice est strictement interdit

Le dossier documentaire comporte 8 pages.

NOUVEAUTES 2019

1. LES COPIES SERONT RENDUES EN L'ETAT AU SERVICE ORGANISATEUR. A L'ISSUE DE L'EPREUVE, CELUI-CI PROCEDERA A L'ANONYMISATION DE LA COPIE.
2. NE PAS UTILISER DE CORRECTEUR D'ORTHOGRAPHE SUR LES COPIES.
3. ECRIRE EN NOIR OU EN BLEU – PAS D'AUTRE COULEUR.
4. IL EST RAPPELE AUX CANDIDATS QU'AUCUN SIGNE DISTINCTIF NE DOIT APPARAITRE SUR LA COPIE.

SUJET

LES QUESTIONS

Les réponses devront être rédigées. L'ensemble des questions sera noté sur 10 points.

QUESTION 1:

Que signifie l'acronyme LDAP ? Quel en est le principe ?

QUESTION 2 :

Qu'est-ce qu'une authentification forte ? Citer un exemple.

QUESTION 3 :

Quelles sont les différences entre un serveur physique et un serveur virtuel ?

QUESTION 4 :

Quelles différences faites-vous entre machines virtuelles et Cloud ?

QUESTION 5 :

Qu'est-ce qu'un SGBD non R ? Citer un exemple.

QUESTION 6 :

Quelles sont les différences entre XML et XSD ?

QUESTION 7 :

Qu'est-ce qu'un outil collaboratif ? Citer un exemple.

QUESTION 8 :

Qu'est-ce que la cryptomonnaie ? Quel en est le principe ?

QUESTION 9 :

Qu'est-ce qu'une API ?

QUESTION 10 :

Définir les acronymes PCA et PRA ? A quoi servent un PCA et un PRA ?

ETUDE DE CAS

L'étude de cas se subdivise en deux parties distinctes. L'ensemble de ces parties sera noté sur 10 points.

Etude de Cas N°1 (5 points) **:**

Vous êtes technicien(ne) SIC dans une équipe de support aux développeurs. Votre chef de service vous confie l'opération de migration des serveurs de développement actuellement basés sur des composants logiciels propriétaires vers des composants libres «open source».

Un serveur de base de données :
- Système d'exploitation : Microsoft Windows 2008 server 32 bits
- SGBD : Oracle version 9

Un serveur « web »
- Système d'exploitation : Microsoft Windows server 2008 server 32 bits
- Serveur web : Microsoft IIS 7
- Langage de script supporté : PHP

Les deux serveurs sont protégés par une solution de protection contre les virus de l'éditeur McAfee.

Le matériel existant bénéficie encore de trois années de garantie pour les composants physiques, ils sont compatibles avec le système d'exploitation Linux et éligibles pour la virtualisation.

Question 1 :
Proposer un système d'exploitation « open source ». Justifier votre choix.

Question 2 :
Proposer un SGBD « open source ». Justifier votre choix.

Question 3 :
Proposer un serveur web open source. Justifier votre choix.

Question 4 :
Proposer une solution de protection contre les virus compatibles avec des composants « open source ».

Question 5 :
Afin d'exploiter au mieux les capacités des serveurs à votre disposition et de les mutualiser pour offrir plusieurs environnements aux développeurs, proposer une solution technique de virtualisation souple et adaptée pour le développement. Elle devra être compatible avec vos choix « open source ».

Etude de Cas N°2 (5 points) **:**

Vous êtes technicien(ne) SIC dans un centre d'hébergement du ministère de l'intérieur. Votre chef(fe) de centre vous demande de tester la technologie «docker» afin d'en étudier la faisabilité et le retour sur investissement.

Question 1 :
Présenter la technologie Docker. Quelle est la différence avec de la virtualisation classique ?

Question 2 :
Peut-on facilement mettre en œuvre un serveur «web» avec docker dans un environnement de développement, de qualification et de production ?

Question 3 :
Peut-on mutualiser plusieurs conteneurs sur une même machine ? Quelles sont les principales limites ?

Question 4 :
Existe-t-il des solutions de gestion centralisées de ces conteneurs ? Si oui, citer celle qui est la plus étudiée et mise en œuvre aujourd'hui.

Question 5 :
Quels sont les principaux obstacles à la mise en œuvre de cette technologie pour un système en production ?

Dossier documentaire :

Monter un serveur LAMP grâce à Docker

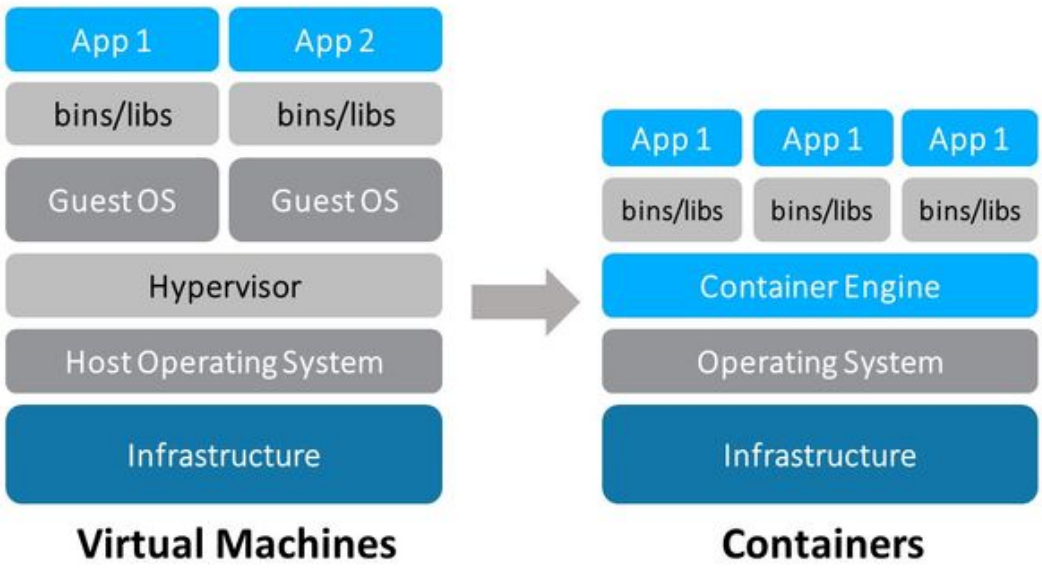
Docker permet d'installer les logiciels de son choix, dans les versions de son choix quelle que soit notre version de Linux. Pour cela il isole les logiciels qu'on souhaite utiliser les uns des autres avec chacun leurs dépendances dans des « containers ». Mais il permet aussi d'éviter les inconvénients de la virtualisation (fichiers lourds, ressources machines divisées, lenteurs, etc.).
C'est donc une solution extrêmement pertinente lorsqu'il s'agit de déployer une plateforme de développement (afin de reproduire n'importe quel environnement de production), ou d'une manière générale de déployer des applications ou des versions de logiciels non supportées par le système courant. De plus cela confère une portabilité très aisée et une grande souplesse à sa configuration.

Nous traiterons ici de la mise en place d'un serveur LAMP (Linux Apache MySQL PHP) au moyen de Docker.
Pour cet exemple nous allons installer PHP 5.6 qui n'est pas disponible sur Xenial.

Nous allons considérer 2 méthodes : une simple, l'autre plus avancée.
La méthode simple consiste à installer un seul container avec tout le stack LAMP.
Avec la méthode avancée on installera le serveur web (Apache et PHP) séparément du serveur de base de données, dans 2 containers différents. C'est la méthode conseillée par Docker, qui recommande de séparer chaque processus / service dans un container différent. Cela rend effectivement notre environnement de travail extrêmement modulaire et permet par ex. de switcher de PHP 5.6 à PHP 7 ou de MySQL à MariaDB en un claquement de doigt !

https://doc.ubuntu-fr.org/docker_lamp

Machine virtuelle / Conteneur



https ://rancher.com/playing-catch-docker-containers/

Document 3

Article 1 : Quelle différence entre conteneurisation et virtualisation ?

Une machine virtuelle(VM – Virtual Machine) « imite » intégralement un serveur. Dans un serveur virtualisé type, chaque VM « invitée » contient un système d'exploitation complet, avec ses pilotes, fichiers binaires ou bibliothèques, ainsi que l'application elle-même. Chaque VM s'exécute alors sur un hyperviseur, qui s'exécute à son tour sur un système d'exploitation hôte, qui lui même fait fonctionner le matériel du serveur physique.

Bien que la méthode ait fait ses preuves, on s'aperçoit aisément que chaque itération du système d'exploitation hôte et des fichiers binaires associés risque d'entraîner des doublons entre les VM ; cela gaspille la mémoire du serveur et limite forcément le nombre de VM prises en charge par chaque serveur. Ce qui pour certains, remet en cause le futur de cette technologie.

A la base, le concept de conteneurisation permet aux instances virtuelles de partager un système d'exploitation hôte unique, avec des fichiers binaires, bibliothèques ou pilotes.

Cette approche réduit le gaspillage des ressources car chaque conteneur ne referme que l'application et les fichiers binaires ou bibliothèques associés. On utilise donc le même système d'exploitation (OS) hôte pour plusieurs conteneurs, au lieu d'installer un OS (et d'en acheter la licence) pour chaque VM invitée. Ce procédé est souvent appelé virtualisation au niveau du système d'exploitation.

Le rôle de l'hyperviseur est alors assuré par un moteur de conteneurisation, tel que Docker, qui s'installe par-dessus le système d'exploitation hôte.

Comme le conteneur de chaque application est libéré de la charge d'un OS, il est nettement plus petit, plus facile à migrer ou à télécharger, plus rapide à sauvegarder ou à restaurer.

Enfin, il exige moins de mémoire. La conteneurisation permet au serveur d'héberger potentiellement beaucoup plus de conteneurs que s'il s'agissait de machines virtuelles. La différence en termes d'occupation peut être considérable, car un serveur donné accueillera de 10 à 100 fois plus d'instances de conteneurs que d'instances d'application sur VM.

Un environnement conteneurisé, comme Docker, fonctionne comme une série de couches, avec tout d'abord une image de base composée d'un OS et d'une application, dont linux, Apache et une application WEB personnalisée. Les mises à jour et modifications s'appliquent par couches supplémentaires pour créer de nouvelles images, puis l'image voulue sert à lancer des conteneurs pouvant également être arrêtés, transférés et supprimés selon les besoins.

En isolant les conteneurs les uns des autres, la conteneurisation assure la sécurité des applications et empêche la prolifération de logiciels malveillants entre les instances… Même si, par définition, l'isolation est plus importante entre VM qu'entre conteneurs.

Stephen J. Bigelow

Article 2 : Cinq inconvénients des conteneurs (et comment y remédier)

Nous allons examiner chacun des 5 inconvénients des conteneurs et essayer d'y apporter des solutions.

1 – Incompatibilité avec certaines tâches

Thomas Bittman fait remarquer que les conteneurs, bien que polyvalents, sont loin de pouvoir remplacer tous les déploiements de machines virtuelles (VM) existants. En effet, tout comme d'anciennes applications se prêtaient mieux à des déploiements physiques aux premiers temps de la virtualisation, certaines applications ne conviennent pas du tout à une virtualisation en conteneurs.

Par exemple, les conteneurs sont la solution idéale pour le développement d'applications de type microservice. Cette approche permet de configurer des applications plus complexes à partir de composants élémentaires, chacun de ces composants étant déployé dans un conteneur et les conteneurs consécutifs étant reliés entre eux pour former l'application complète.

Pour faire évoluer les fonctionnalités de l'application, il suffit de déployer de nouveaux conteneurs renfermant les composants appropriés au lieu de créer entièrement de nouvelles itérations de l'application.

Cependant, certaines applications ne peuvent fonctionner que de façon monolithique. Ainsi conçues, elles sont difficilement compatibles avec des avantages tels que l'évolutivité ou le déploiement rapide. Dans ces cas-là, les conteneurs ne font que rigidifier encore la charge de travail.

La meilleure méthode consiste souvent à faire des essais pour voir quelles applications existantes peuvent bénéficier d'une conteneurisation. Les nouveaux modèles de développement d'applications en tireront sans doute parti. En revanche, les applications qui ne se prêtent pas à cette transformation pourront toujours s'exécuter comme des VM entièrement fonctionnelles adossées à un hyperviseur conventionnel.

2 – Problème des dépendances

Les VM classiques sont extrêmement autonomes chacune comprenant un système d'exploitation (OS) unique, des pilote et des composants d'application. Elles peuvent également migrer vers n'importe quel autre système du moment qu'un hyperviseur approprié est disponible.

De leur côté, les conteneurs s'exécutent sur un OS et partagent la majeure partie du noyau sous-jacent ainsi qu'un grand nombre de fichiers binaires et de bibliothèques. D'après Bittman, les dépendances imposées aux conteneurs peuvent limiter la portabilité entre serveurs.

Par exemple, les conteneurs Linux sous Docker ne peuvent pas être exécuter sur les versions actuelles de Windows Server.

La réponse est plus un constat qu'une véritable solution : les conteneurs peuvent être créés et multipliés en quelques secondes, tandis que les systèmes d'exploitation évoluent pour créer des variantes « micro » ou « nano » capables d'offrir une formidable stabilité et des redémarrages extrêmement rapides.

Au niveau natif, les conteneurs sont plus disponibles dans ces environnements et il est toujours possible de les déplacer ou d'en supprimer tant que d'autres serveurs sont disponibles dans le datacenter.

Ces dépendances s'allègent au fur et à mesure que ne nouveaux OS apparaissent. Ainsi Windows Server 2016 prévoit la prise en charge de Docker et des conteneurs Hyper-V natifs. Par ailleurs, outre Docker, de nombreuses plateformes de conteneurs sont disponibles : LXC, Parrallels Virtuozzo, Joyent, Canonical LXD, Spoon et d'autres encore. Même Vmware entre dans la course.

3 – Faiblesse relative de l'isolement

Les VM reposant sur un hyperviseur offrent un degré élevé d'isolement les unes des autres, car les ressources matérielles du système sont toutes virtualisées et présentées aux VM par le biais de l'hyperviseur.

Autrement dit, une bogue, un virus ou une intrusion peut porter atteinte à une VM sans se propager aux autres.

Les conteneurs eux sont plus vulnérables, car ils partagent un noyau et des composants systèmes et leur fonctionnement exige déjà un niveau d'autorisation élevé (généralement l'accès root dans les environnements Linux).

En conséquence, les erreurs et attaques ont bien plus de chances de se répercuter sur un OS sous-jacent et sur d'autres conteneurs, risquant ainsi de propager des activités malveillantes bien au-delà de l'événement d'origine.

Certes, les plateformes de conteneurs évoluent dans le sens d'une séparation des droits des OS et d'une restriction des comportements contraires à la sécurité. Mais comme l'explique Thomas Bittman, les administrateurs peuvent d'ores et déjà renforcer la sécurité en exécutant les conteneurs dans une VM. Par exemple, il est possible de configurer une VM Linux sur Hyper-V et d'y installer des conteneurs Docker.

Dans cette configuration, même en cas d'atteinte aux conteneurs, la faille ne s'étendra pas en dehors de la VM, limitant ainsi la portée des dégâts potentiels.

4 – Risque de prolifération

Si la gestion du cycle de vie des VM est importante dans les environnements basés sur un hyperviseur, elle s'avère absolument essentielle pour les conteneurs. En effet, ces derniers offrent l'avantage non négligeable de pouvoir être mis en service dupliqués à la vitesse de l'éclair.

Le revers de la médaille est qu'il est également possible de consommer une grande quantité de ressources informatiques sans vraiment s'en rendre compte.

Ce n'est guère grave si les conteneurs qui composent l'application sont arrêtés ou supprimés dès lors qu'ils ne sont plus nécessaires. Mais en cas d'oubli, la montée en charge d'une application conteneurisée peut se traduire par des coûts de Cloud computing tout aussi importants qu'inutiles pour l'entreprise.

Évidemment, comme le souligne Bittman, les fournisseurs de Cloud se frottent les mains (puisqu'ils font leur beurre en louant de la puissance de traitement). C'est donc aux utilisateurs qu'il revient de surveiller le déploiement des conteneurs.

5 – Outil de gestion limités

Les types d'outils nécessaires pour surveiller et gérer les conteneurs sont encore rares dans le secteur. Le phénomène n'est pas nouveau : déjà, aux premiers temps de la virtualisation sur hyperviseur, on manquait d'outils adaptés.

Et maintenant qu'il en existe pour la surveillance et la gestion des VM, de nouveaux outils commencent à apparaître pour les conteneurs. On peut citer Kubernetes, outil de gestion de Docker open source de Google, DockerUI qui remplace les fonctions de ligne de commande Linux par ne inerface Web, Logspout qui achemine les logs des conteneurs vers un emplacement centralisé, etc.

Thomas Bittman suggère aux administrateurs de pallier la pénurie d'outils en utilisant des conteneurs dans des VM afin de tirer parti des outils des VM pour assurer certaines fonctions de surveillance et de gestion.

En effet puisque les outils de VM sont plus nombreux et plus avancés, ils peuvent servir de substituts en attendant que les outils dédiés aux conteneurs arrivent à maturité.

Pour finir, Bittman ne tarit pas d'éloges sur les conteneurs, mettant l'accent sur leur capacité de dépliement rapide et léger dans des environnements haute densité et évolutifs, sur les E/S natives (non virtualisées) pour de meilleures performances, sur les infrastructures de développement prêtes à l'emploi comme Docker et sur des outils renommés de partage et de collaboration tels que GitHub et autres.

Cependant, les conteneurs ne constituent pas la panacée pour toutes les tâches de virtualisation. Simple article dans la panoplie des outils de virtualisation, ils font souvent bon ménage avec les traditionnelles VM.

Stephen J. Bigelow

www.lemagit.fr, extrait de 2016

Docker : une version Enterprise pour Docker Desktop

Alibaba Cloud, AWS, Google Cloud, Microsoft Azure, OpenStack... Les containers Docker sont pris en charge par les principaux clouds, tant sur le créneau du cloud privé que du cloud public.

[Mis à jour le 17 janvier 2019 à 11h52] *La technologie Docker ne cesse de s'enrichir. Dernier chapitre en date : lors de son événement clients européen, la Dockercon Europe, début décembre, Docker a levé le voile sur une version Enterprise de son application de pilotage de containers logiciels pour poste de travail (Docker Desktop). Cette déclinaison permet d'automatiser le déploiement d'environnements de développement Docker sur des dizaines de milliers de postes, puis d'assurer la gestion de leur maintenance et leur cohérence au regard des applications en production (lire l'article : Docker industrialise les développements Docker sur le poste de travail).*

1- Comment résumer ce qu'est Docker ?

Docker permet d'embarquer une application dans un container virtuel qui pourra s'exécuter sur n'importe quel serveur machine, qu'il soit physique ou virtuel. D'abord optimisé pour Linux, il l'est désormais pour Windows Server. C'est une technologie qui a pour but de faciliter les déploiements d'application, et la gestion du dimensionnement de l'infrastructure sous-jacente. Elle est en partie proposée en open source (sous licence Apache 2.0) par une société américaine, également appelée Docker, qui a été lancée par un Français : Solomon Hykes.

2- Quelle différence avec la virtualisation traditionnelle ?

La virtualisation traditionnelle permet, via un hyperviseur, de simuler une ou plusieurs machines physiques, et les exécuter sous forme de machines virtuelles (VM) sur un serveur ou un terminal. Ces VM intègrent elles-mêmes un OS sur lequel des applications sont exécutées. Ce n'est pas le cas du container. Le container fait en effet directement appel à l'OS de sa machine hôte pour réaliser ses appels système et exécuter ses applications.

Les containers Docker au format Linux exploitent un composant du noyau Linux baptisé LXC (ou Linux Container). Au format Windows Server, ils s'adossent à une brique équivalente.

https://www.journaldunet.com/solutions:cloud-computing/1146290-docker-une-version-enterprise-pour-docker-desktop/

Kubernetes – Tout savoir sur la plateforme d'orchestration de containers

Kubernetes est une plateforme open source d'orchestration de containers créé par Google. Découvrez son utilité, son fonctionnement, ainsi que ses différences avec Docker.

Les containers sont une méthode de virtualisation de système d'exploitation permettant de lancer une application et ses dépendances à travers un ensemble de processus isolés du reste du système. Cette méthode **permet d'assurer le déploiement rapide et stable des applications** dans n'importe quel environnement informatique.

En plein essor depuis plusieurs années, **les containers ont modifié la façon dont nous développons**, déployons et maintenons des logiciels. De par leur légèreté et leur flexibilité, ils ont permis l'apparition de nouvelles formes d'architectures d'application, consistant à constituer des applications au sein de containers séparés pour ensuite déployer ces containers sur un cluster de machines virtuelles ou physiques. Toutefois, cette nouvelle proche a **créé le besoin de nouveaux outils « d'orchestration de containers »** pour automatiser le déploiement, le management, le networking, le scaling et la disponibilité des applications basées sur container. Tel est le rôle de Kubernetes.

Kubernetes est un projet Open Source créé par Google en 2015. Il permet **d'automatiser le déploiement et la gestion d'applications multi-container à l'échelle**. Il s'agit d'un système permettant d'exécuter et de coordonner des applications containerisées sur un cluster de machines. C'est une plateforme conçue pour gérer entièrement le cycle de vie des applications et services containerisés en utilisant des méthodes de prédictibilité, de scalabilité et de haute disponibilité.

Principalement compatible avec Docker, **Kubernetes peut fonctionner avec n'importe quel système de container** conforme au standard Open Container Initiative en termes de formats d'images et d'environnements d'exécution. De par son caractère open source, Kubernetes peut aussi être utilisé librement par n'importe qui, n'importe où.

Kubernetes : comment ça marche ?

Les architectures **Kubernetes reposent sur plusieurs concepts et abstractions** : certaines existaient déjà auparavant, d'autres lui sont spécifiques. La principale abstraction sur laquelle repose Kubernetes est le cluster, à savoir le groupe de machines exécutant Kubernetes et les containers qu'il gère.

Un cluster Kubernetes doit avoir **un master** : le système qui commande et contrôle toutes les autres machines du cluster. Un cluster Kubernetes hautement disponible réplique les fonctions du master sur les différentes machines, mais seul un master à la fois exécute le controller-manager et le scheduler.

Chaque cluster contient des noeuds Kubernetes. Il peut s'agir de machines physiques ou virtuelles. Les noeuds quant à eux exécutent des pods : les objets Kubernetes les plus basiques pouvant être créés ou gérés. Chaque pod représente une seule instance d'une application ou d'un processus en cours d'exécution sur Kubernetes, et se constitue d'un ou plusieurs containers. Tous les containers sont lancés et répliqués en groupe dans le pod. Ainsi, l'utilisateur peut se concentrer sur l'application plutôt que sur les containers.

https ://www.lebigdata.fr/kubernetes-definition