**Etude de l’existant :**

Description du contexte :

L’objectif principal de ce projet est de mettre en place une solution basée kubernetes qui facilitera la gestion des TPs au niveau de l’ESI d’une part, et d’autre part, réduira le temps nécessaire pour préparer les environnements d’exécution par les enseignants. Dans cette phase, on essaiera de faire une étude de l’existant et de critiquer ce dernier.

Description du fonctionnement des salles machines au niveau de l’ESI :

Compte tenu de la diversité de modules enseignés au sein de l’ESI, les séances des travaux pratiques qui nécessitent souvent l’utilisation des salles machines comprennent l’utilisation de plusieurs applications sur les machines soient celles des salles en accédant à travers des terminaux ou les machines personnelles des étudiants (programmation, développement, base de données, conception objet…). Ces derniers sont souvent répartis sur 12 à 15 binômes, chaque binôme travaillant sur la même instance.

Après discussion avec les personnes impliquées dans cette activité pédagogique, notamment les enseignants, il a été apporté à notre attention les différents problèmes qu'ils rencontrent souvent durant ces séances. Parmi les problèmes les plus mentionnés qui perturbent le bon déroulement des TPs :

* Beaucoup de temps pour installer les applications
* Conflits de versions et compatibilité des applications dans les OS des machines
* Démarrage lent des machines
* Utilisation importante de ressources



Solutions existantes :

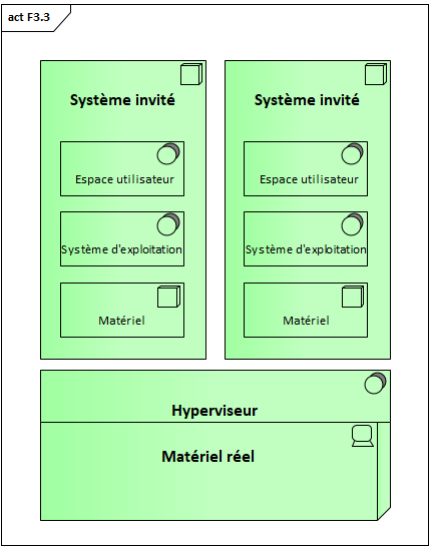
* Technologie de virtualisation : Machine Virtuelle (lien : https://www.cairn.info/revue-distances-et-savoirs-2007-2-page-245.htm)

Une salle virtuelle peut être considérée comme un ensemble de machines virtuelles, agencées en fonction d’objectifs pédagogiques. Le couplage de techniques de virtualisation d’une part, et le déport d’interface d’autre part, permet la mise des machines virtuelles à la disposition des étudiants.

Cette solution permet de faire fonctionner simultanément plusieurs machines virtuelles appelées « ordinateurs invités » sur une seule machine physique appelée « ordinateur hôte ».

Chaque machine virtuelle invitée dispose de son propre système d’exploitation, ses propres applications et son propre environnement de connexion.

Modèle de la virtualisation par hyperviseur :



(Référence : cours VCL)

* Défauts de cette solution :

Bien que l'utilisation des machines virtuelles ait apporté plusieurs bénéfices aux enseignants lors des séances TPs comparé auparavant (plusieurs systèmes différents sur la même machine, réduire les besoins du câblage, tirer un meilleur avantage du processeur...). Cette technologie présente aussi des obstacles intrinsèques à sa raison d’être entre autres : la complexité de gérer les VMs et surtout détecter les erreurs, le nombre limité de VMs par hyperviseurs, consommation importante de ressources et dégradation de performance si le nombre de VMs est important à cause de la communication avec avec le matériel à travers l’hyperviseur, en plus toutes les machines virtuelles invitées dépendent de la machine physique hôte.

**Etude théorique :**

Introduction :

La conteneurisation est devenue une tendance majeure du développement logiciel en tant qu’alternative ou compagnon de la virtualisation.

La technologie évolue rapidement. Cela se traduit par des avantages mesurables pour les développeurs et les équipes d’exploitation ainsi que pour l’infrastructure logicielle globale.

La virtualisation élimine le besoin d’un serveur entier pour une seule application alors que la conteneurisation élimine le besoin d’un système d’exploitation complet pour chaque application.

Conteneurisation : (lien : https://www.hebergeurcloud.com/conteneurs-avantages-inconvenients/#:~:text=La%20conteneurisation%20%C3%A9limine%20le%20besoin,de%20la%20technologie%20de%20conteneurisation.)

Principe :

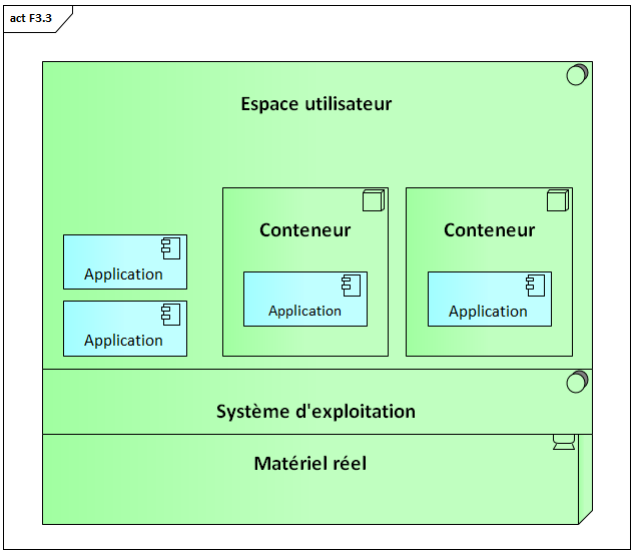
La conteneurisation permet aux développeurs de créer et de déployer des applications plus rapidement et de manière plus sécurisée. Avec les méthodes traditionnelles, le code est développé dans un environnement informatique spécifique. Lorsqu’il est transféré vers un nouvel emplacement, cet environnement entraîne souvent des bogues et des erreurs.

Par exemple, lorsqu’un développeur transfère du code d’un ordinateur de bureau vers une machine virtuelle (VM) ou d’un système d’exploitation Linux vers un système d’exploitation Windows.

La conteneurisation élimine ce problème en regroupant le code de l’application avec les fichiers de configuration, les bibliothèques et les dépendances associés nécessaires à son exécution. Ce package unique de logiciel ou « conteneur » est extrait du système d’exploitation hébergeur.

Par conséquent, il est autonome et devient portable – capable de fonctionner sur n’importe quelle plateforme ou cloud, sans problèmes.

Modèle de la conteneurisation :



(Référence : cours VCL)

Etapes de conteneurisations d’une application : (lien : <https://www.lemagit.fr/conseil/Conteneurisez-vos-applications-en-cinq-etapes>)

*Etape1* : Décomposition

Pour les applications formées de nombreux composants et qui sont complexes à gérer, elle consiste à mettre chaque composant dans un conteneur et assurer la communication entre eux.

*Etape2* : Sélection d’une image de base

Chercher une image existante qui servira de base à l’application

*Etape3* : Prise en compte de la sécurité

*Etape4* : Ajout du code

*Etape5* : Configuration, test, déploiement

Avantages :

* **Portabilité** : un conteneur crée un package exécutable de logiciels qui est extrait du système d’exploitation hébergeur (non lié ou dépendant de celui-ci). Par conséquent, ce package est portable et capable de fonctionner de manière uniforme et cohérente sur n’importe quelle plateforme ou cloud.
* **Agilité** : Le Docker Engine open source qui exécute les conteneurs a lancé la norme de l’industrie pour les conteneurs avec des outils de développement simples et une approche de packaging universelle qui fonctionne à la fois sur les systèmes d’exploitation Linux et Windows.  
  L’écosystème des conteneurs s’est déplacé vers des moteurs gérés par l’Open Container Initiative (OCI). Les développeurs de logiciels peuvent continuer à utiliser des outils et des processus agiles ou [DevOps](https://www.hebergeurcloud.com/devops-cloud-10-etapes-reussir/)pour le développement et l’amélioration rapide des applications.
* **Vitesse**: les conteneurs sont souvent qualifiés de « légers ». Cela signifie qu’ils partagent le noyau du système d’exploitation (OS) de la machine et ne sont pas embourbés par ces frais supplémentaires.  
  Non seulement cela améliore l’efficacité des serveurs, mais cela réduit également les coûts de serveur et de licence. Puis, les temps de démarrage sont accélérés étant donné qu’il n’y a pas de système d’exploitation à démarrer.
* **Isolation des erreurs** : chaque application conteneurisée est isolée et fonctionne indépendamment des autres. La défaillance d’un conteneur n’affecte pas le fonctionnement continu des autres conteneurs. Les équipes de développement peuvent identifier et corriger tout problème technique dans un conteneur sans aucun temps d’arrêt dans d’autres conteneurs. En outre, le moteur de conteneur peut tirer parti de toutes les techniques d’isolation de sécurité du système d’exploitation, telles que le contrôle d’accès SELinux, pour isoler les erreurs dans les conteneurs.
* **Efficacité** : les logiciels exécutés dans des environnements conteneurisés partagent le noyau du système d’exploitation de la machine. Puis, les couches d’application d’un conteneur peuvent être partagées entre les conteneurs. Ainsi, les conteneurs ont une capacité intrinsèquement plus petite qu’une machine virtuelle et nécessitent moins de temps de démarrage. Cela permet à beaucoup plus de conteneurs de s’exécuter sur la même capacité de calcul qu’une seule machine virtuelle. Ce qui améliore l’efficacité des serveurs, réduisant les coûts de serveur et de licence.
* **Facilité de gestion** : une plateforme d’orchestration de conteneurs automatise l’installation, la mise à l’échelle et la gestion des charges de travail et des services conteneurisés. Les plateformes d’orchestration de conteneurs peuvent faciliter les tâches de gestion telles que la mise à l’échelle des applications conteneurisées, le déploiement de nouvelles versions d’applications et la surveillance, la journalisation et le débogage, entre autres fonctions. Kubernetes est une technologie open source (à l’origine open-source par Google, basée sur leur projet interne appelé Borg). Il est peut-être le système d’orchestration de conteneurs le plus populaire disponible. A l’origine, il automatise les fonctions des conteneurs Linux. Kubernetes fonctionne avec de nombreux moteurs de conteneur, tels que Docker, mais aussi avec tout système de conteneur conforme aux normes Open Container Initiative (OCI) pour les formats d’image de conteneur et les environnements d’exécution.
* **Sécurité**: l’isolement des applications en tant que conteneurs empêche par nature l’invasion de code malveillant d’affecter d’autres conteneurs ou le système hébergeur. De plus, des autorisations de sécurité peuvent être définies pour empêcher automatiquement les composants indésirables d’entrer dans des conteneurs ou limiter les communications avec des ressources inutiles.

Inconvénients :

* **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 constitutifs étant reliés entre eux pour former l'application complète.

Pour faire évoluer les fonctionnalités de l'application, il suffit alors 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](https://www.lemagit.fr/definition/Hyperviseur) conventionnel.

Comme l'indique un architecte informatique d'une grande compagnie d'assurance, « Les conteneurs sont intéressants, mais notre équipe informatique aurait pas mal de retard à rattraper pour apprendre à s'en servir correctement. »

* **Problème des dépendances**

Les VM classiques sont extrêmement autonomes, chacune comprenant un système d'exploitation (OS) unique, des pilotes 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 exécutés 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 de nouveaux OS apparaissent. Ainsi, [Windows Server 2016 prévoit la prise en charge de Docker](https://www.lemagit.fr/conseil/Comment-fonctionne-Docker-sous-Windows) et [des conteneurs Hyper-V natifs](https://www.lemagit.fr/actualites/4500244249/Hyper-V-Containers-Microsoft-veut-un-peu-plus-securiser-Docker). Par ailleurs, outre Docker, de nombreuses plateformes de conteneurs sont disponibles : LXC, Parallels Virtuozzo, Joyent, Canonical LXD, Spoon et d'autres encore. Même [VMware entre dans la course](https://www.lemagit.fr/actualites/4500248915/Bonneville-VMware-adopte-les-conteneurs-Docker).

* **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.

* **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 et 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 pas très 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.

Evidemment, 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.

* **Outils 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](https://www.lemagit.fr/article/Quelle-difference-entre-Docker-et-Kubernetes), outils de gestion Docker open source de Google, [DockerUI](https://github.com/crosbymichael/dockerui) qui remplace les fonctions de ligne de commande Linux par une interface Web, [Logspout](https://github.com/gliderlabs/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 de VM pour assurer certaines fonctions de surveillance et de gestion.