**Résumé :**

Dans Le livrable précédent, nous avons traité l’aspect théorique de notre solution exposant les différentes technologies que nous envisageons à utiliser, tout en analysant leurs avantages et les inconvénients de chacune, nous sommes arrivés à la conclusion d'utiliser docker combiné avec docker.

Durant cette phase nous nous intéressons au processus de mise en place et comment Dockeriser une application simple et une autre plus complexe. Docker permet de créer des déploiements à construire une seule fois et à exécuter partout, allant des ordinateurs portables locaux aux serveurs de production et tout ce qui se trouve entre les deux. Malheureusement, tirer parti de cette puissance nécessite de contourner certaines limitations à la fois dans Docker et potentiellement dans l'architecture de l'application choisie.

**Introduction :**

Imaginons une situation où un enseignant qui a passé toute une nuit à préparer un guide d’installation et d’utilisation d’une application, teste le processus sur son ordinateur et fonctionne à merveille ! Le lendemain, les étudiants téléchargent les ressources et tentent d’installer les outils. Mais malheureusement, l’application ne fonctionne pas pour certains ! Une explication à ce problème est que l’environnement d’exécution de ces étudiants est inadéquat. Deux systèmes peuvent avoir des différences de version sur les dépendances ou encore des bibliothèques manquantes.

Avec Docker, nous sommes entrés dans l’ère des architectures à base de « conteneur ». On parle aussi de « virtualisation de niveau système d’exploitation » par opposition aux technologies à base d’hyperviseurs (comme VMWare ou VirtualBox) qui cherchent à émuler un environnement matériel. Contrairement à une VM, un conteneur n’embarque pas un système d’exploitation complet. Il repose pour l’essentiel sur les fonctionnalités offertes par l’OS sur lequel il s’exécute. L’inconvénient de cette approche est qu’elle limite la portabilité du conteneur à des OS de la même famille (Linux dans le cas de Docker).

Cette approche, en revanche, a l’avantage d’être beaucoup plus légère (les conteneurs sont nettement plus petits que les VM et plus rapides au démarrage) tout en offrant une isolation satisfaisante en termes de réseau, de mémoire ou de système de fichiers.

L’objet de cet ouvrage est d’offrir une approche à 360 degrés de l’écosystème Docker. Docker, plus qu’une technologie, est en effet aujourd’hui un écosystème de solutions fourmillantes : Docker Compose, Kubernetes (et ses nombreuses implémentations chez les leaders du cloud public), ou encore Docker Swarm. Autour du runtime (moteur d’exécution) qui exécute les conteneurs (le Docker Engine), des outils complémentaires visent à conditionner, assembler, orchestrer et distribuer des applications à base de conteneurs. Timidement, des initiatives de standardisation voient le jour, laissant espérer une meilleure interopérabilité que celle qui prévaut aujourd’hui dans le domaine de la virtualisation matérielle.

Les dockers viennent proposer une solution à notre problème principal ( et au problème d'hétérogénéité ) en général. Il s’agit d’une plateforme qui permet d'exécuter le code à l'intérieur d'un conteneur indépendamment de la machine sur laquelle on est ! Un conteneur ressemble à une machine virtuelle sauf qu'il n'inclut pas un système d'exploitation ce qui lui permet de s'exécuter en quelque secondes et d'être beaucoup plus léger. Mais, cette plateforme a des insuffisances. En effet, la création des conteneurs nécessite la saisie de plusieurs lignes de commandes.

Durant cette partie du projet, nous exploitons plusieurs aspects techniques notamment la mise en place détaillée des images dockers pour certaines applications utilisées très souvent dans l’apprentissage pratique de plusieurs modules, ensuite expliquer comment le client pourra les utiliser tout en détaillant comment nous avons procédé pour faciliter cette tâche et l’optimiser.

**Conclusion :**

Les expérimentations sont divisées en deux parties, la première a concerné les commandes shell de linux qui représente une application légère et permet après de tester les limites d’une conteneurisation combinée avec orchestration à grande échelle. Ces tests nous ont permis d'avoir une estimation de certains paramètres comme la stabilité, sécurité et scalabilité potentielle. Nous avons aussi présenté les problèmes et défis liés à l’utilisation des images docker ce qui nous pousse par la suite à considérer une solution basée sur l’orchestration des conteneurs. Les expérimentations ont permis d'évaluer la performance des images selon certaines configurations et démontré qu'avec le bon choix de configuration, ces dernières peuvent au moins délivrer la même performance qu'un ordinateur de bureau.

La deuxième partie de ce travail a concerné les résultats obtenus. Nous avons motivé le choix de la technologie qui est Docker, puisqu'elle est la plus populaire et la plus rapide du marché en ce moment et qu'elle offre une facilité de déploiement remarquable, en se basant sur le travail élaboré dans le rapport précédent.