Maîtrisez Docker La Révolution des Conteneurs

Présenté par :
Mr Bonitah RAMBELOSON
Ingénieur Consultant
DevOps et Cloud

Public Concerné

- Développeurs et ingénieurs logiciels
- Administrateurs système et DevOps
- Architectes solutions et cloud
- Étudiants en informatique et technologies de l'information
- Toute personne intéressée par l'optimisation du cycle de développement et de déploiement des applications

Prérequis :

➢ Compétences en Informatique :

- Connaissances de base en ligne de commande (CLI)
- Familiarité avec les concepts de réseaux et de systèmes d'exploitation
- Expérience avec au moins un langage de programmation (facultatif mais recommandé)

Prérequis

➢ Ressources Matérielles :

- Un ordinateur avec accès administrateur pour installer Docker
- Connexion Internet pour télécharger Docker et accéder aux ressources en ligne
- Environnement de développement intégré (IDE) ou éditeur de texte pour écrire des Dockerfiles

Objectifs de la Présentation

COMPRENDRE LES FONDAMENTAUX DE DOCKER ET SON IMPORTANCE DANS LE DÉVELOPPEMENT MODERNE

EXPLORER LES CONCEPTS CLÉS ET LES AVANTAGES DE L'UTILISATION DE DOCKER

APPRENDRE À INSTALLER ET CONFIGURER DOCKER SUR DIFFÉRENTES PLATEFORMES

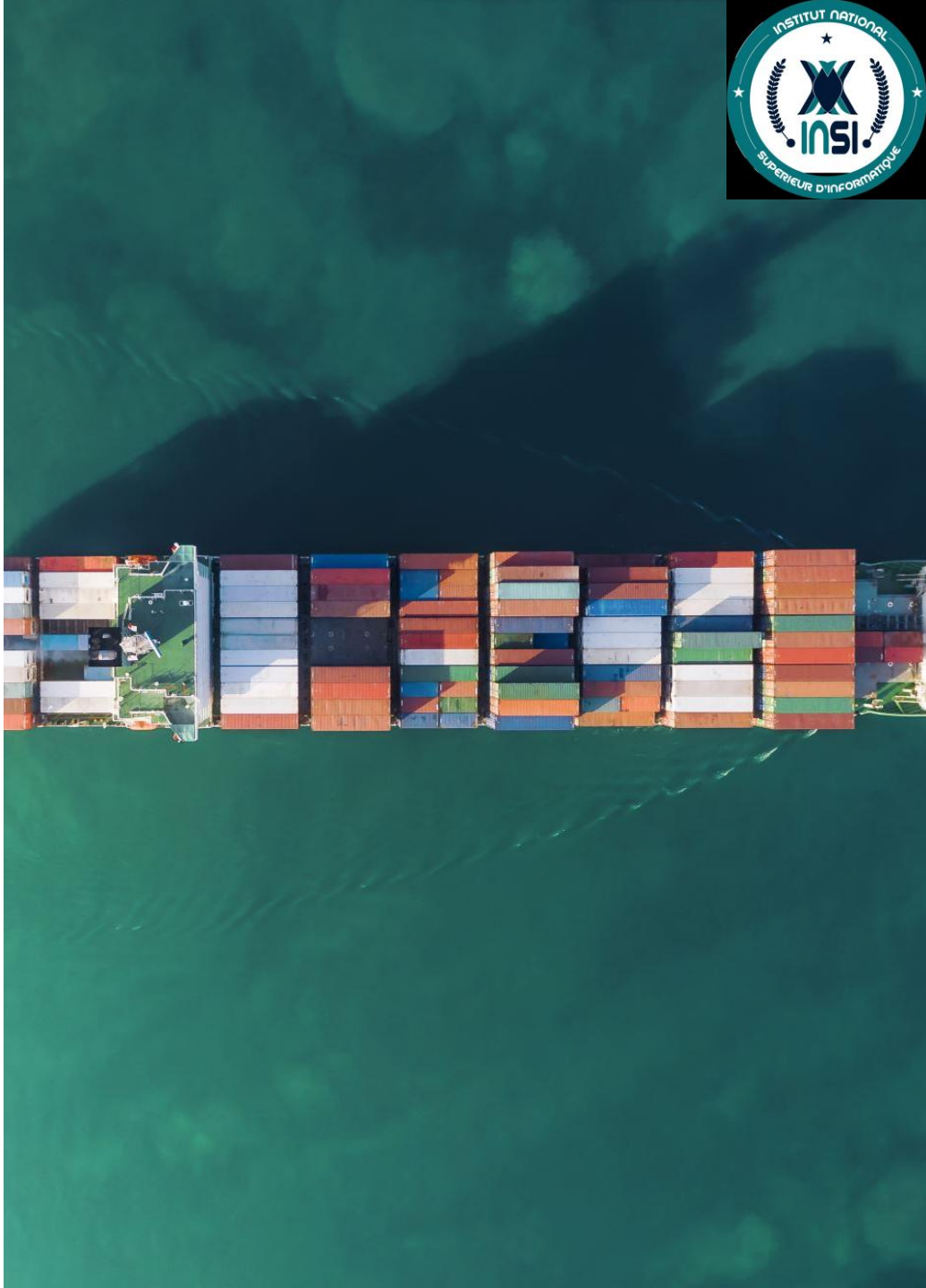
DÉCOUVRIR LES MEILLEURES PRATIQUES POUR CRÉER ET GÉRER DES CONTENEURS DOCKER

ÉTUDIER DES CAS D'UTILISATION RÉELS ET DES EXEMPLES D'IMPLÉMENTATION RÉUSSIE

Formateur Consultant | Ingénieur DevOps & Cloud

Introduction à Docker

Docker est une plateforme open-source qui automatise le déploiement, la mise à l'échelle et l'exécution des applications dans des conteneurs. Les conteneurs permettent aux développeurs de regrouper une application et toutes ses dépendances dans un environnement standardisé, garantissant que l'application fonctionne de manière cohérente sur n'importe quelle infrastructure.



Pourquoi utiliser à Docker ?

Avec Docker vous pouvez :

- Isoler les applications : Chaque conteneur fonctionne comme un environnement léger et isolé, ce qui évite les conflits entre les applications.
- Accélérer le développement : Les conteneurs se lancent rapidement et permettent de tester les applications dans des environnements identiques à la production.
- Simplifier le déploiement : Docker facilite le déploiement continu et l'intégration continue (CI/CD), permettant de livrer des applications plus rapidement et plus efficacement.
- Docker est devenu un outil essentiel pour les développeurs et les équipes DevOps, transformant la manière dont les applications sont construites, livrées et exécutées.

Plan du Cours

PARTIE I
- Concepts de Base (Historique et évolution de Docker)
- Conteneurs vs Machines Virtuelles
- Architecture de Docker

PARTIE II
- Installation de Docker
- Images Docker
- Conteneurs docker
- Registre Docker (Docker Hub)

PARTIE III
- Création d'Images Docker avec Dockerfile
- Persistance des données sur Docker (Docker volumes)
- Réseaux Docker (Docker Network)

PARTIE IV
- Docker Compose (application multi-conteneurs)

PARTIE V
- Études de cas réussies (**TD** et **TP**)
- Conclusion et Ressources

Concepts de Base : Historique et Évolution de Docker

- **Qu'est-ce que Docker ?**

• Docker est une plateforme open-source qui automatise le déploiement, la mise à l'échelle et l'exécution des applications dans des conteneurs.

• Les conteneurs permettent d'emballer une application et toutes ses dépendances dans un environnement standardisé, garantissant une exécution cohérente sur n'importe quelle infrastructure.

Concepts de Base : Historique et Évolution de Docker

- **Historique de Docker**

- **2010 :** Lancement de la technologie de conteneurs Linux (LXC) qui a inspiré Docker.
- **2013 :** Docker est lancé par Solomon Hykes lors de la conférence PyCon.
- **2014 :** DockerHub est lancé, offrant un registre public pour partager des images Docker.
- **2015 :** Introduction de Docker Compose pour gérer des applications multi-conteneurs.
- **2016 :** Docker Swarm est intégré pour l'orchestration native des conteneurs.
- **2017 :** Docker adopte Kubernetes comme solution d'orchestration, renforçant l'interopérabilité.

Concepts de Base : Historique et Évolution de Docker

- **Évolution de Docker**

- **Adoption Croissante :** Docker est rapidement adopté par les entreprises pour moderniser leurs infrastructures.
- **Écosystème Riche :** Développement d'outils et de services autour de Docker pour améliorer la gestion des conteneurs.
- **Communauté Active :** Une communauté open-source dynamique contribue à l'amélioration continue de Docker.
- **Intégration CI/CD :** Docker devient un élément clé des pipelines d'intégration et de déploiement continus.

Concepts de Base : Historique et Évolution de Docker

- **Pourquoi Docker ?**

- **Portabilité :** Les applications Docker fonctionnent de manière cohérente sur n'importe quel environnement.
- **Efficacité :** Les conteneurs sont légers et se lancent rapidement, optimisant l'utilisation des ressources.
- **Isolation :** Chaque conteneur est isolé, évitant les conflits entre les applications.

Conteneurs vs Machines Virtuelles

- Machines Virtuelles (VM)

- **Définition :** Une machine virtuelle est une émulation d'un système informatique, comprenant un système d'exploitation complet et des ressources matérielles virtuelles.
- **Isolation :** Chaque VM inclut un noyau d'OS complet, offrant une isolation totale entre les applications.
- **Ressources :** Les VM sont souvent lourdes et nécessitent plus de ressources (CPU, mémoire) pour fonctionner.
- **Démarrage :** Le démarrage d'une VM peut prendre plusieurs minutes en raison de l'initialisation du système d'exploitation.
- **Utilisation :** Idéales pour exécuter plusieurs systèmes d'exploitation sur un même matériel ou pour des environnements nécessitant une isolation complète.

Conteneurs vs Machines Virtuelles

- Conteneurs

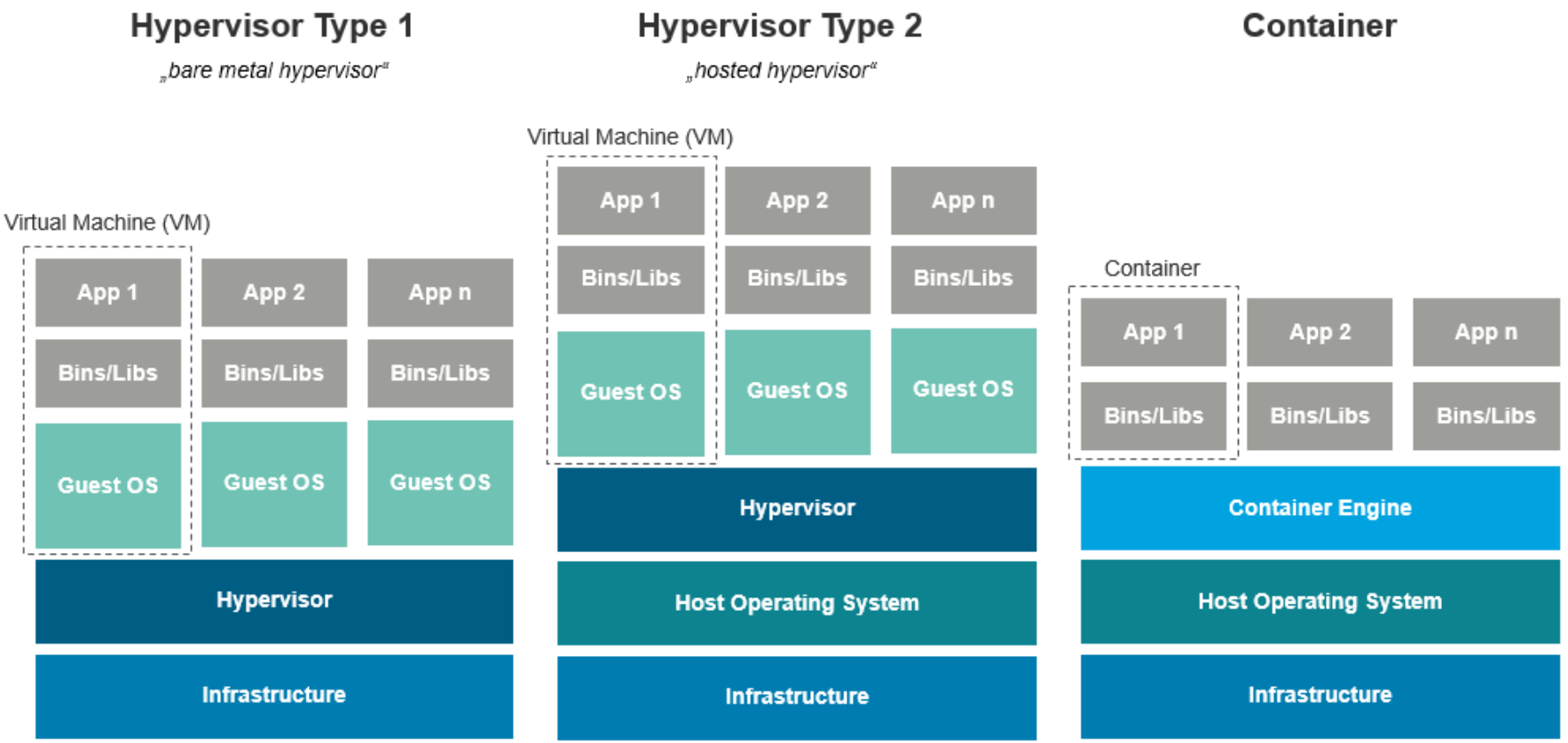
- **Définition :** Un conteneur est un environnement léger et portable qui regroupe une application et toutes ses dépendances.
- **Isolation :** Les conteneurs partagent le noyau du système d'exploitation hôte, offrant une isolation au niveau des processus.
- **Ressources :** Les conteneurs sont légers et utilisent moins de ressources que les VM, car ils ne nécessitent pas un OS complet.
- **Démarrage :** Les conteneurs se lancent en quelques secondes, ce qui accélère le développement et le déploiement.
- **Utilisation :** Parfaits pour le développement et le déploiement d'applications modernes, les microservices, et les environnements CI/CD.

Conteneurs vs Machines Virtuelles

- Comparaison

- **Efficacité :** Les conteneurs sont plus efficaces en termes de ressources et de rapidité de démarrage.
- **Portabilité :** Les conteneurs offrent une portabilité accrue, permettant de déployer des applications de manière cohérente sur différentes infrastructures.
- **Gestion :** Les conteneurs sont plus faciles à gérer et à orchestrer, surtout dans des environnements à grande échelle.

Architecture Conteneurs vs Machines Virtuelles



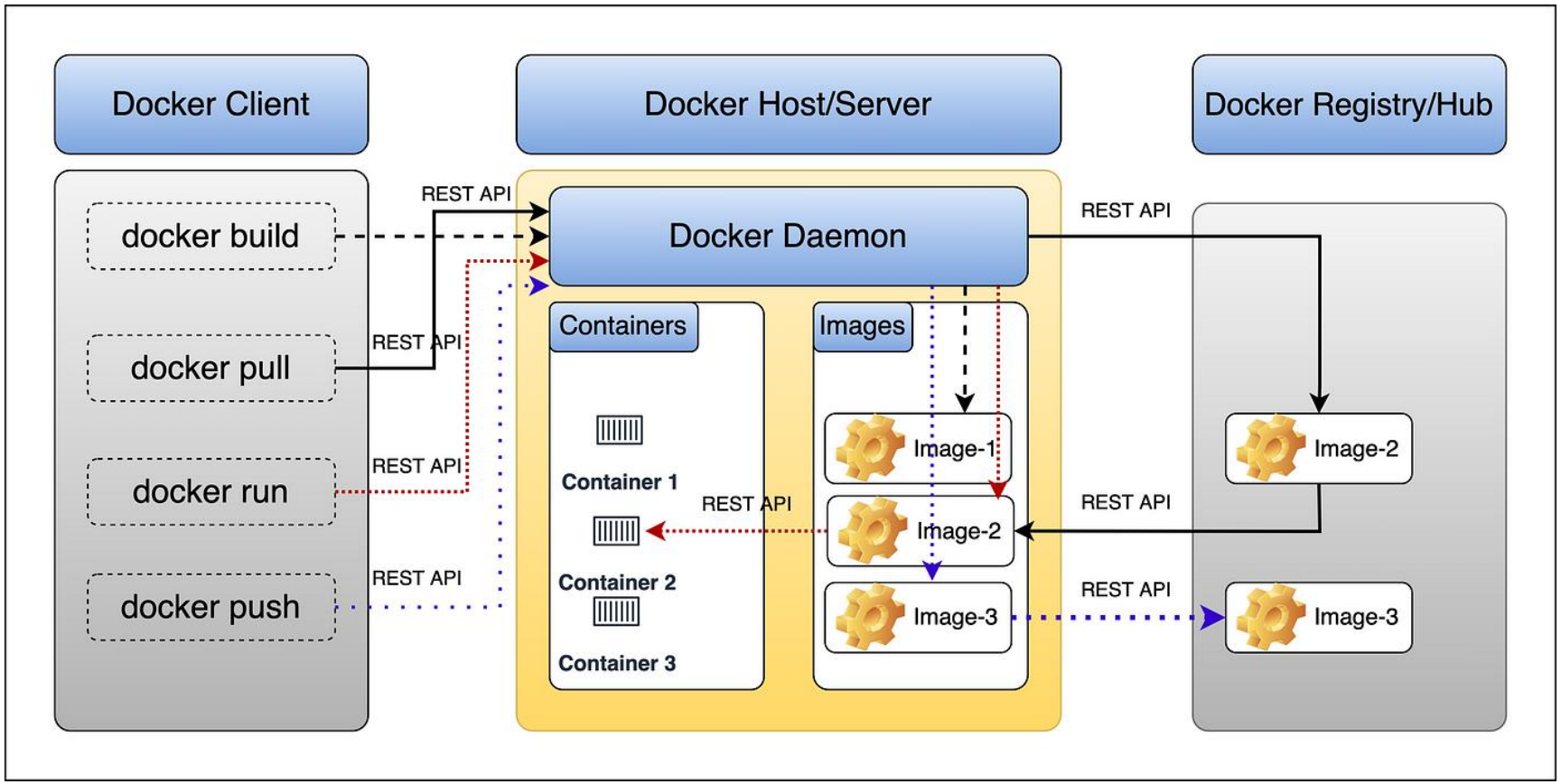
Formateur Consultant | Ingénieur DevOps & Cloud

Conclusion

Conteneurs vs Machines Virtuelles

Les conteneurs Les conteneurs, comme ceux utilisés par Docker, offrent une solution légère et efficace pour le déploiement d'applications, tout en maintenant un niveau d'isolation suffisant pour la plupart des cas d'utilisation.

Architecture de Docker



Composants Clés

- Docker Engine

  - **Docker Daemon (dockerd) :** Le cœur de Docker, qui gère les conteneurs, les images, les réseaux et les volumes.

  - **Docker CLI :** Interface en ligne de commande permettant aux utilisateurs d'interagir avec le Docker Daemon.

  - **Docker API :** Interface permettant aux applications d'interagir avec le Docker Daemon.

Composants Clés

- Images Docker

- **Définition :** Une image Docker est un modèle en lecture seule utilisé pour créer des conteneurs.

- **Couches :** Les images sont constituées de couches, chacune représentant une modification (ajout, suppression, modification de fichiers).

- **Dockerfile :** Fichier texte contenant les instructions pour construire une image Docker.

Composants Clés

- Conteneurs

  - **Instance d'une Image :** Un conteneur est une instance exécutable d'une image Docker.

  - **Isolation :** Chaque conteneur fonctionne dans un environnement isolé, partageant le noyau de l'hôte mais avec ses propres ressources.

  - **Portabilité :** Les conteneurs peuvent être déployés sur n'importe quelle infrastructure compatible avec Docker.

Composants Clés

- Registres Docker

  - **Docker Hub :** Registre public où les utilisateurs peuvent stocker et partager des images Docker.

  - **Registres Privés :** Solutions pour stocker des images Docker de manière privée, souvent utilisées dans les entreprises.

Composants Clés

- Réseaux et Volumes

Réseaux : Docker permet de créer des réseaux virtuels pour connecter les conteneurs entre eux.

Volumes : Stockage persistant pour les données générées par les conteneurs, permettant de conserver les données même après l'arrêt du conteneur.

Fonctionnement

- **Création d'un Conteneur**

- Un utilisateur exécute une commande Docker via le CLI.
- Le Docker Daemon reçoit la commande via l'API.
- Le Daemon crée un conteneur à partir d'une image Docker.
- Le conteneur s'exécute en utilisant les ressources de l'hôte