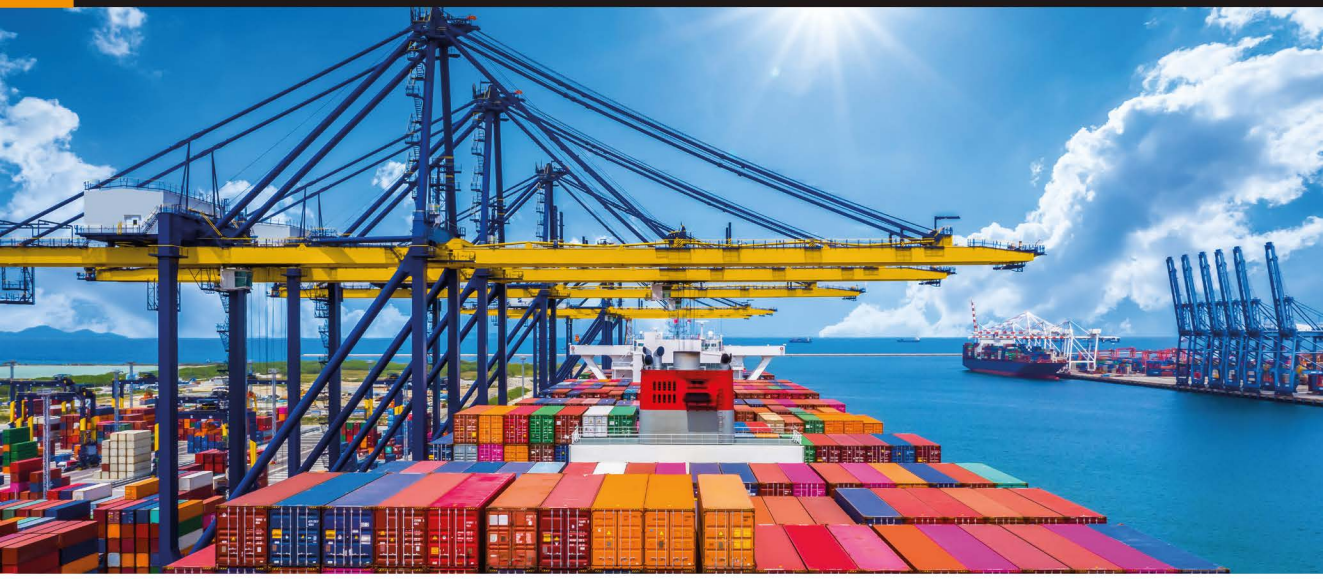
ÉTUDES, DÉVELOPPEMENT, INTÉGRATION

DOCKER ET CONTENEURS

Architectures, développement, usages et outils



Pierre-Yves Cloux
Thomas Garlot
Johann Kohler

LES + EN LIGNE

3e édition

DUNOD

Chez le même éditeur

Kubernetes
K. Hightower, B. Burns et J. Beda
208 pages
2019

SOA, microservices, API management
5e édition
X. Fournier-Morel, P. Grojean, G. Plouin et C. Rognon
528 pages
2020

Programmation Python avancée
X. Olive
368 pages
2021

Scrum
6e édition
C. Aubry
336 pages
2022

Cloud et transformation digitale
6e édition
G. Plouin
272 pages
2022

DOCKER ET CONTENEURS

Architectures, développement, usages et outils

Pierre-Yves Cloux
Head of Architecture & Data
chez Migros Online

Thomas Garlot
Head of Development
chez Migros Online

Johann Kohler
Technical Lead
chez Migros Online

3e édition

DUNOD

Illustration de couverture : © Avigator Fortuner – Shutterstock

Pour aller plus loin et mettre toutes les chances de votre côté, des ressources complémentaires sont disponibles sur le site www.dunod.com.

Connectez-vous à la page de l'ouvrage (grâce aux menus déroulants, ou en saisissant le titre, l'auteur ou l'ISBN dans le champ de recherche de la page d'accueil).
Sur la page de l'ouvrage, sous la couverture, cliquez sur le lien « LES + EN LIGNE ».

11 rue Paul Bert, 92240 Malakoff
www.dunod.com
ISBN 978-2-10-084223-0

Table des matières

TROISIÈME PARTIE
Apprendre Docker

QUATRIÈME PARTIE
Développer, déployer et opérer des conteneurs

CINQUIÈME PARTIE
Orchestration de conteneurs

Avant-propos

Pendant longtemps, déployer du code en production revenait à tenter de transporter de l'eau entre ses mains : c'était fonctionnel, mais pas vraiment optimal. Comme l'eau filant entre les doigts, il manquait presque nécessairement une partie des données de configuration lors du déploiement, ceci en dépit d'efforts méthodologiques, documentaires et humains conséquents.

◆ *L'apport de la virtualisation*

La virtualisation a tenté de répondre à cette problématique (parmi d'autres) sans apporter une réponse complètement satisfaisante. En effet, bien qu'offrant une isolation vis-à-vis de l'architecture matérielle, qui se standardise de plus en plus, la machine virtuelle reste… une machine. Elle exécute un système d'exploitation dont les paramètres peuvent différer d'un environnement à l'autre.

Des outils comme Chef ou Puppet résolvent une partie du problème, mais n'offrent encore une fois qu'une couche d'abstraction partielle. Enfin, le déploiement d'applications sur la base d'une image de VM (pour *Virtual Machine*) est lourd (plusieurs gigaoctets, y compris pour les OS les plus compacts).

◆ *Les architectures à base de conteneurs*

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).

Nous verrons dans le chapitre 1 que l'implémentation Windows de Docker autorise l'exécution de conteneurs Windows et Linux, ce qui contredit l'affirmation précédente. Néanmoins Microsoft fait usage d'une solution qui a nécessité un travail d'acclimatation de Windows à Unix. Il en est de même pour Apple et l'exécution de conteneurs sous Mac.

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.

Étonnamment, le concept de conteneur et son implémentation ne sont pas nouveaux. La version de OpenVZ pour Linux date, par exemple, de 2005, de même que Solaris Zone ou FreeBSD jail. Docker a changé la donne en simplifiant l'accès à cette technologie par l'introduction d'innovations, comme la mise à disposition d'un langage de domaine (DSL ou *Domain Specific Language*) permettant, au travers du fameux « Dockerfile », de décrire très simplement la configuration et la construction d'un conteneur.

◆ *Le propos de cet ouvrage*

L'objet de cet ouvrage est d'offrir une approche à 360 degrés de l'écosystème « conteneurs ».

Nous aborderons dans le premier chapitre le « cas Docker » et notamment dans quelle mesure l'entreprise Docker Inc. n'est aujourd'hui plus un acteur majeur du monde des conteneurs. Son statut de pionnière ne fait pas moins du terme « Docker » l'équivalent informatique de « Frigidaire » (marque commerciale historique aujourd'hui propriété d'Electrolux). Le lecteur nous pardonnera donc l'usage du terme « Docker » pour évoquer aussi bien l'entreprise que la technologie dont elle a permis la démocratisation et qui est devenue un standard commun à toutes les implémentations.

Les conteneurs, plus qu'une technologie, sont aujourd'hui l'un des éléments d'un écosystème de solutions fourmillantes : Podman, Nomad, Compose, Kubernetes (et ses nombreuses implémentations chez les leaders du cloud public).

Autour du *runtime* (moteur d'exécution) qui exécute les conteneurs (Docker Engine ou Containerd), des outils complémentaires visent à conditionner, assembler, orchestrer et distribuer des applications à base de conteneurs. Des initiatives de standardisation, comme OCI (Open Container Initiative), CRI (Container Runtime Interface) ou CNI (Container Network Interface), témoignent d'une montée en maturité et d'un souci d'interopérabilité entre les solutions commerciales.

Notre objectif dans cet ouvrage est donc multiple :

- ✓ aborder le concept de conteneur et d'architecture à base de conteneurs en décryptant les avantages proposés par cette approche ;
- ✓ apprendre à installer les outils de construction et d'exécution de conteneurs sur un poste de travail ou dans un environnement serveur ;
- ✓ apprendre à utiliser ces outils pour créer des images et manipuler des conteneurs ;
- ✓ étudier des architectures plus complexes au travers d'exemples complets : architectures multi-conteneurs, développement, intégration continue et implémentations multi-hôtes.

◆ *La structure du livre*

Dans une première partie, nous expliquerons ce qu'est un conteneur, sur quels principes et quelles technologies il repose. Nous verrons aussi comment Docker se positionne parmi un nombre croissant d'autres acteurs.

Nous aborderons ensuite le concept de « CaaS », pour « *Container as a Service* », au travers de divers exemples. À cette occasion, nous parlerons d'outils tels que Kubernetes, Nomad, Azure Service Fabric et leurs prédécesseurs Swarm ou Mesos. Nous nous intéresserons aussi aux liens entre les approches conteneurs et celles d'outils de gestion de configuration comme Puppet, Chef ou Ansible.

La deuxième partie de cet ouvrage se focalise sur la prise en main de Docker (ou d'outils fonctionnellement équivalents) en étudiant son installation sur un poste de travail ou dans un environnement virtualisé. Nous commencerons alors à « jouer » avec des conteneurs pour comprendre par la pratique les concepts abordés dans la première partie.

La troisième partie du livre est consacrée à l'apprentissage de Docker. Nous y aborderons les commandes du client et les instructions du Dockerfile. Cette troisième partie peut être lue séquentiellement, mais aussi servir de référence.

La quatrième partie du livre se consacre à la mise en œuvre des concepts appris précédemment autour d'exemples pratiques et didactiques. Nous étudierons ainsi le développement, le conditionnement et le déploiement d'une architecture à base de conteneurs en nous limitant au cas mono-hôte.

La cinquième et dernière partie est consacrée aux solutions d'orchestration de conteneurs : Kubernetes, Nomad et Docker Swarm. Nous montrerons comment l'application étudiée dans la quatrième partie peut être déployée sur plusieurs hôtes sans modification majeure des conteneurs.

◆ *À qui s'adresse ce livre*

Cet ouvrage s'adresse à un public mixte « Dev/Ops » :

- ✓ si vous êtes engagé dans une organisation de développement (en tant que développeur, architecte ou manager), vous trouverez les informations nécessaires pour maîtriser rapidement la conception d'images Docker, mais aussi pour la réalisation d'architectures multi-conteneurs ;
- ✓ si votre domaine est plutôt celui de l'exploitation, vous acquerrez les compétences nécessaires au déploiement de Docker sous Linux.

Ce livre a un quadruple objectif :

- ✓ il enseigne les concepts de base nécessaires à la bonne compréhension de la technologie des conteneurs ;
- ✓ il permet d'accélérer la prise en main de cette technologie en présentant des cas d'usages illustrés par des exemples didactiques ;
- ✓ il offre aussi une référence ponctuée d'exemples du langage DSL et des commandes de Docker ;
- ✓ il apporte enfin des éléments d'information sur l'écosystème foisonnant qui s'est développé autour du concept de conteneur en s'appuyant sur des exemples pratiques.

Compléments en ligne

Le code source et les exemples de ce livre sont distribués via GitHub sur le dépôt public :

https://github.com/dunod-docker/docker-exemples-edition3/

Le dépôt contient, outre les exemples de code, toutes les commandes chapitre par chapitre (permettant un copier/coller) de même que l'ensemble des liens vers des sites externes.

La technologie évoluant constamment, nous invitons le lecteur à vérifier régulièrement si des mises à jour sont disponibles et à lever un ticket en cas de problème.

Veuillez vous reporter à la procédure d'installation du chapitre 3 et aux explications disponibles sur GitHub.

◆ *Remerciements*

L'écriture d'un livre est une activité consommatrice de temps qu'il n'est pas possible de mener à son terme sans l'assistance de nombreuses personnes.

Nous voudrions donc adresser nos remerciements à :

- ✓ Patrick Moiroux, pour son expertise de Linux en général, des architectures web distribuées et de Docker en particulier ;
- ✓ Bernard Wittwer, pour avoir attiré notre attention, il y a près de dix ans, sur un petit projet open source sympathique nommé Docker dont il percevait déjà le potentiel ;
- ✓ Marc Schär, stagiaire de compétition il y a bien longtemps, devenu aujourd'hui expert DevOps et Kubernetes ;
- ✓ nos épouses et nos enfants pour leur patience au regard du temps que nous avons passé isolés du monde extérieur à expérimenter des logiciels et à en décrire le fonctionnement.

PREMIÈRE PARTIE

Les conteneurs : principes, objectifs et solutions

Cette première partie vise à présenter :

- ✓ les origines du concept de conteneur ;
- ✓ l'apport de Docker à cette technologie déjà ancienne ;
- ✓ comment les conteneurs autorisent la réalisation de nouvelles architectures informatiques ;
- ✓ comment les conteneurs colonisent les solutions de gestion d'infrastructure.

Cette première partie comprend deux chapitres. Le premier décrit ce qu'est un conteneur d'un point de vue technique et conceptuel. Le second chapitre présente un survol complet des solutions de gestion d'infrastructure à base de conteneurs : des plus modernes aux plus classiques pour lesquelles les conteneurs apportent les avantages décrits dans le chapitre 1.

Les conteneurs et le cas Docker

Objectif

L'objectif de ce chapitre est de décrire les concepts qui ont présidé à l'émergence de la notion de conteneur logiciel. Nous présenterons les briques de base sur lesquelles les conteneurs reposent. Puis nous expliquerons comment sont construits et distribués les conteneurs. Enfin, nous nous intéresserons aux différents éléments des architectures à base de conteneurs : Docker et les autres.

À l'issue de ce chapitre vous comprendrez ce qu'est un conteneur, quels sont les concepts architecturaux et logiciels qu'il implémente et ce qu'est Docker. Vous saurez aussi quels sont les autres acteurs de cet écosystème.

1. 1 LA CONTENEURISATION

Les conteneurs logiciels cherchent à répondre à la même problématique que les conteneurs physiques aussi appelés conteneurs intermodaux, ces grandes boîtes de métal standardisées qui transportent de nos jours l'essentiel des biens matériels par camion, train, avion ou bateau.

Nous allons donc faire un peu d'histoire.

1.1.1 L'histoire des conteneurs intermodaux

Avant l'apparition de ces conteneurs standardisés, les biens étaient manipulés manuellement. Les Anglo-Saxons utilisent le terme de *break bulk cargo*, ce que nous pourrions traduire par « chargement en petits lots ». Plus simplement, les biens étaient chargés individuellement par des armées de travailleurs.

Les biens étaient, par exemple, produits dans une usine, chargés un par un dans un premier moyen de transport jusqu'à un hangar (généralement près d'un port ou d'une gare) où avait lieu un déchargement manuel. Un second chargement survenait alors depuis ce premier lieu de stockage jusque dans le moyen de transport longue distance (par exemple un bateau). Ce second chargement était lui aussi manuel. En fonction des destinations, des contraintes géographiques et légales, cette séquence de chargement et déchargement pouvait avoir lieu plusieurs fois. Le transport de biens était donc coûteux, lent et peu fiable (biens abîmés ou perdus).