

Découverte et Manipulation de Docker pour Débutants

Bienvenue dans cette présentation qui vous guidera pas à pas dans la découverte de Docker. Nous aborderons:

- l'installation,
- la manipulation en ligne de commande et via Portainer,
- ainsi que la création et la gestion de différents types de conteneurs.



Création de la VM et Installation de Docker

Préparation de la VM

Configurez une machine virtuelle Debian en mode console uniquement avec des ressources minimales : 8 Go de disque, 1 Go de RAM et 1 vCPU. Cette configuration légère est parfaitement adaptée pour découvrir Docker.

Installation de Docker

Installez Docker Engine sans interface graphique en utilisant les commandes apt. Cette approche minimalistre vous permettra de comprendre les fondamentaux de Docker via la ligne de commande.

Vérification de l'installation

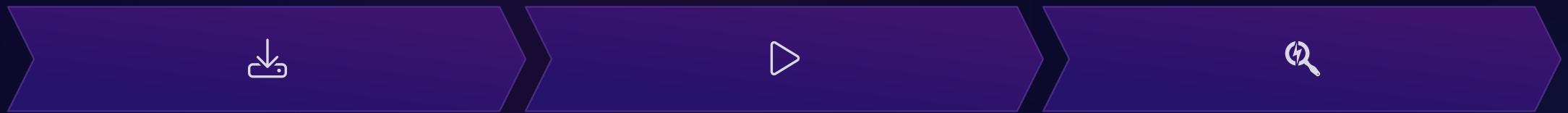
Assurez-vous que le service Docker est correctement démarré et configuré pour se lancer automatiquement au démarrage du système avec systemctl.





hello-world

Test de Docker avec le Conteneur "hello-world"



Téléchargement

Docker télécharge automatiquement l'image hello-world depuis Docker Hub lors de la première exécution

Exécution

Le conteneur s'exécute, affiche son message et se termine

Vérification

Utilisez les commandes "docker ps -a hello-word" pour confirmer l'exécution

Cette étape simple mais essentielle valide votre installation Docker. Le conteneur hello-world représente le test standard pour confirmer le bon fonctionnement de l'environnement Docker. Familiarisez-vous avec les commandes de base comme **docker run**, **docker ps**, et **docker images** qui serviront constamment dans vos futures manipulations.

Recréation du Conteneur "hello-world" avec un Dockerfile

Création du Dockerfile

Créez un fichier nommé "Dockerfile" contenant les instructions pour bâtir votre image personnalisée basée sur Debian slim

Développement du script

Créez un fichier hello.sh qui affiche "Hello World" et rendez-le exécutable avec chmod

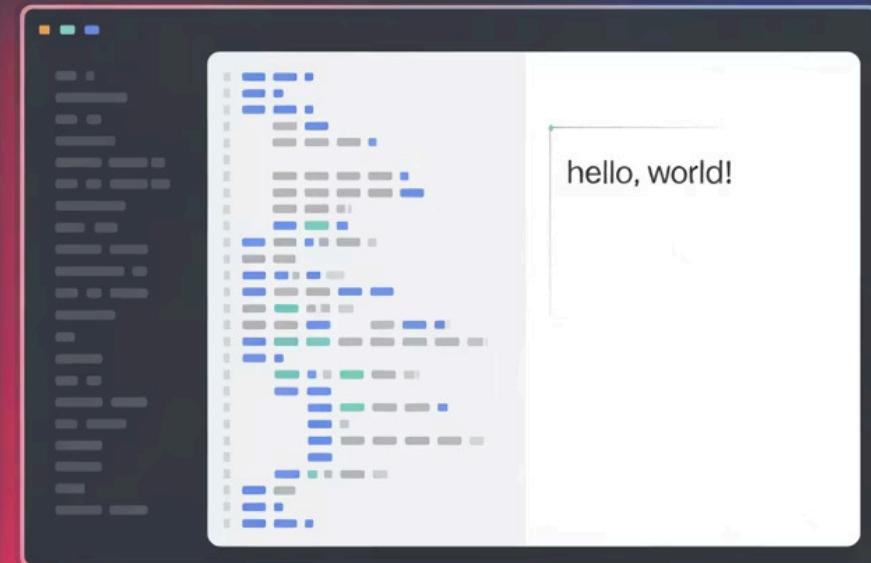
Construction et exécution

Utilisez **docker build** pour créer l'image puis **docker run** pour exécuter votre conteneur personnalisé

Cette étape vous initie à la création d'images Docker personnalisées. Le Dockerfile est le blueprint qui définit comment votre image sera construite. Vous apprendrez à utiliser les instructions FROM, COPY, RUN et CMD pour définir votre environnement conteneurisé.

CODE
YOUR FUTURE
Build, ship, and run with confidence

GET STARTED



Création d'un Conteneur SSH Personnalisé

Dockerfile

Créez un Dockerfile basé sur Debian avec installation d'OpenSSH

Connexion

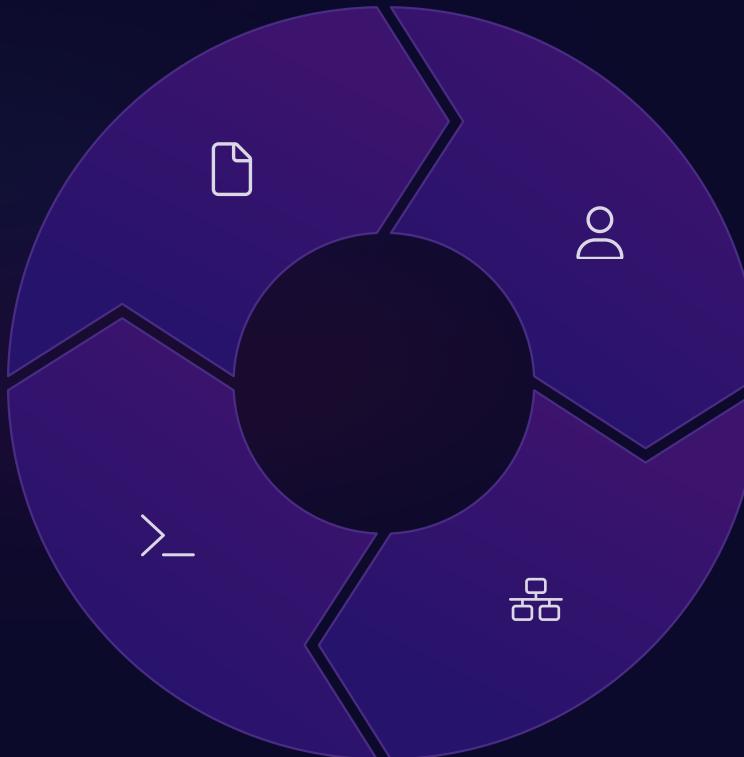
Testez la connexion SSH vers votre conteneur

Configuration

Configurez un utilisateur root avec le mot de passe root123

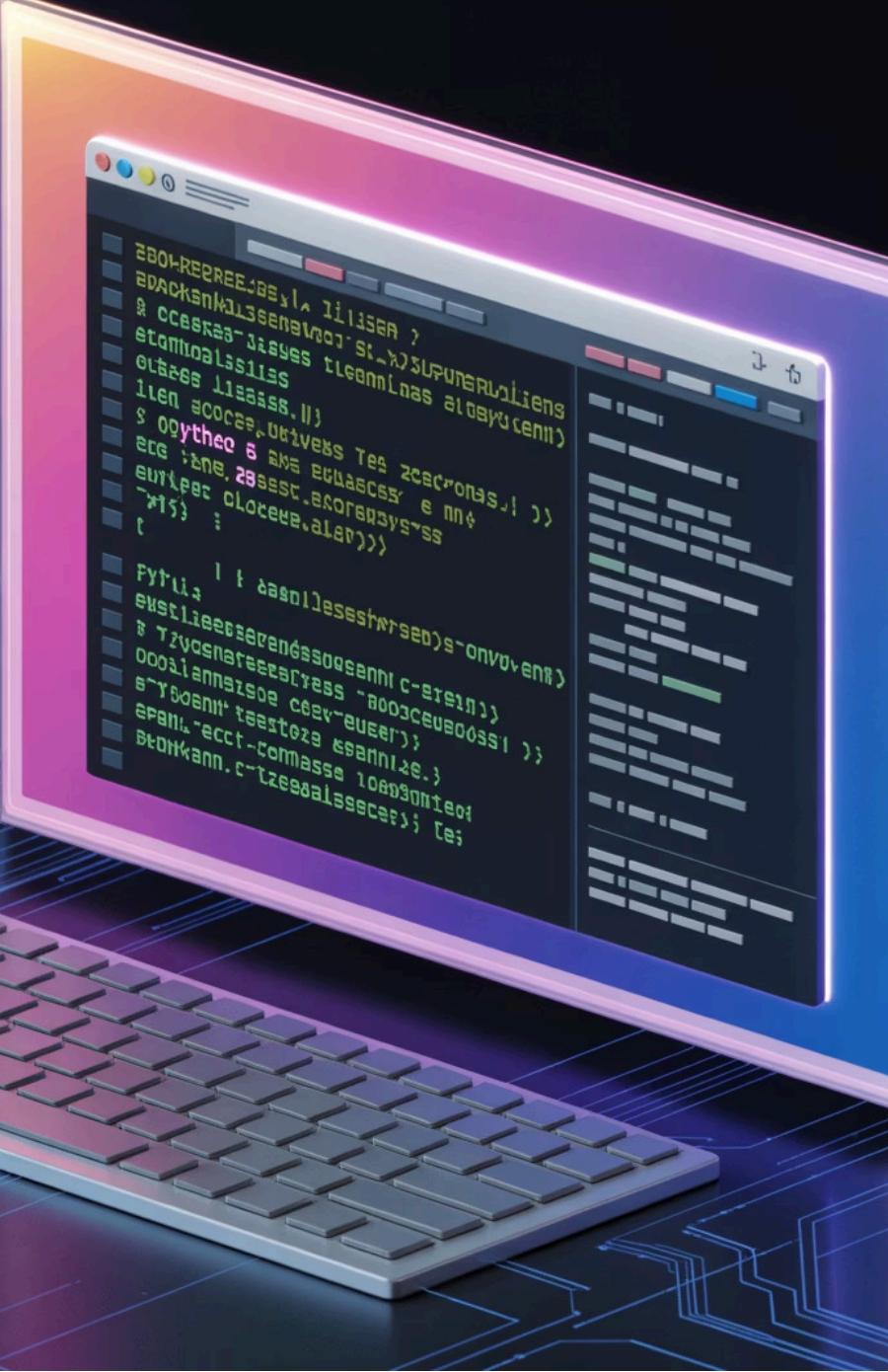
Exposition

Exposez un port non standard pour le service SSH



Ce projet initie aux concepts avancés de Docker, notamment la configuration de services réseau et la personnalisation des conteneurs. Apprendre à installer des paquets dans une image, configurer des services et exposer des ports pour une communication externe.

Création d'Alias Docker



Commande longue	Alias suggéré	Description
docker ps -a	dps	Afficher tous les conteneurs
docker images	dim	Lister toutes les images
docker build -t	dbuild	Construire une image avec tag
docker-compose up -d	dcup	Démarrer les services en arrière-plan

Les alias Docker améliorent considérablement votre efficacité en ligne de commande. En créant ces raccourcis dans votre fichier `~/.bashrc`, vous réduisez la quantité de texte à taper pour les commandes fréquentes. N'oubliez pas d'exécuter **source `~/.bashrc`** après avoir ajouté vos alias pour les rendre disponibles dans votre session courante.

Volumes Docker entre Deux Conteneurs



Les volumes Docker constituent la solution recommandée pour persister et partager des données entre conteneurs. Contrairement aux bind mounts, les volumes sont gérés entièrement par Docker et offrent davantage de fonctionnalités. Dans ce job, vous allez créer un volume, le monter dans deux conteneurs différents, puis vérifier que les modifications apportées dans un conteneur sont visibles dans l'autre.

Cette expérience pratique vous fera comprendre l'importance des volumes pour les applications qui génèrent des données devant survivre au cycle de vie des conteneurs.

Docker Compose - NGINX + FTP + Volume

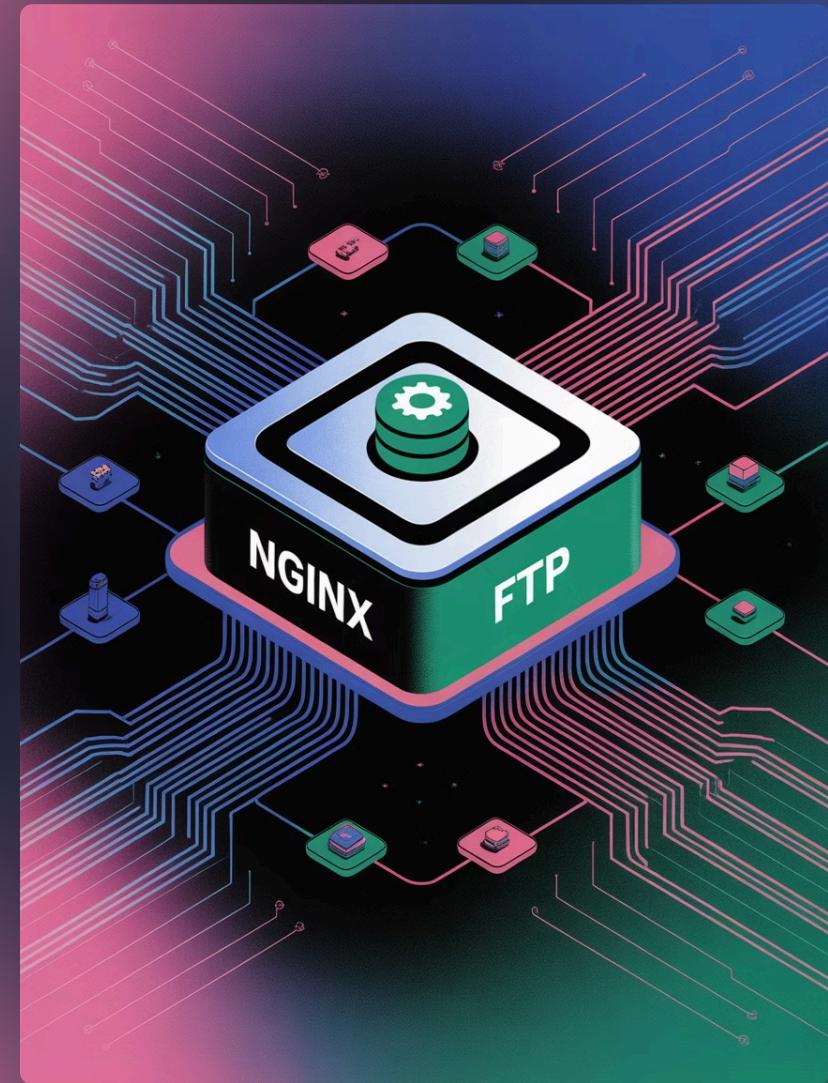
Volume partagé
Stockage commun pour les fichiers web et FTP



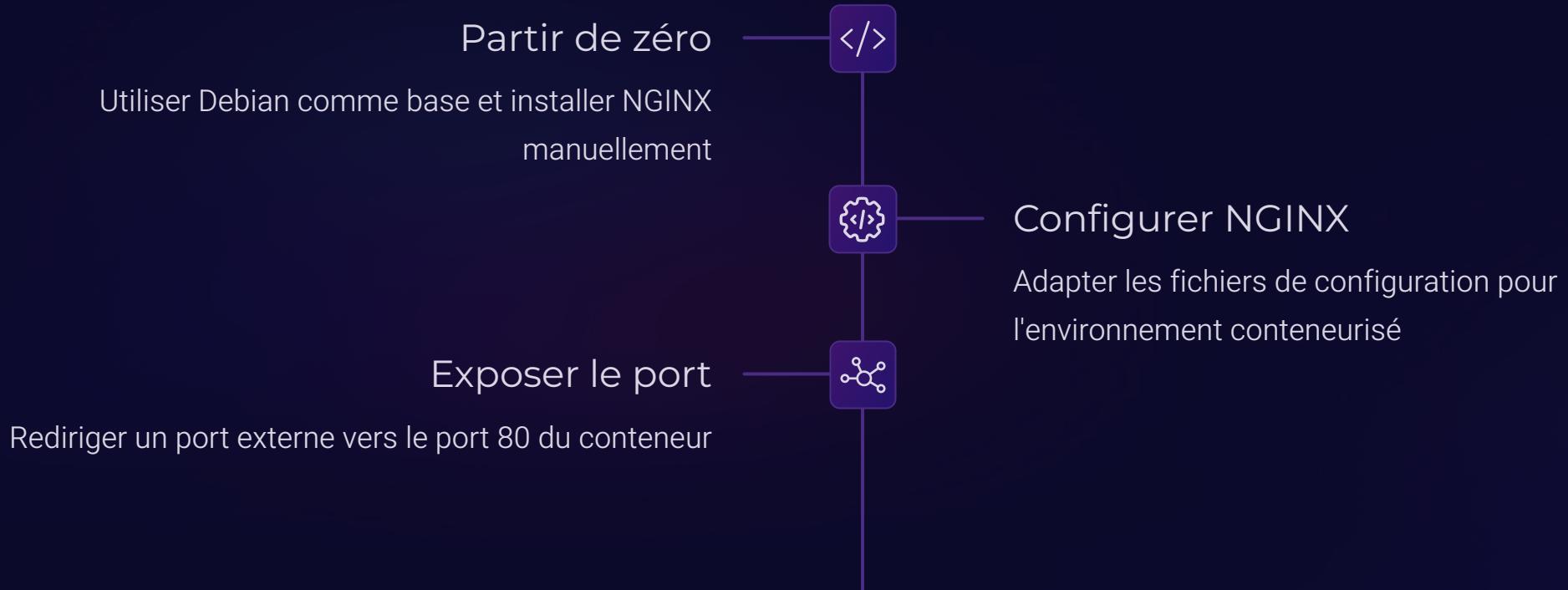
Services
NGINX et serveur FTP configurés automatiquement
Fichiers de configuration
Tout défini dans un seul docker-compose.yml

Docker Compose simplifie considérablement la gestion d'applications multi-conteneurs. Dans cet exercice, ***un fichier docker-compose.yml*** définira deux services interdépendants : ***un serveur web NGINX et un serveur FTP, tous deux partageant un volume commun pour les fichiers web.***

Cette configuration permettra d'uploader des fichiers via FTP depuis la machine locale avec FileZilla, puis de vérifier qu'ils sont immédiatement disponibles via le serveur web NGINX. C'est une excellente introduction aux architectures multi-services dans Docker.



Création d'un Conteneur NGINX sans Image Existante



Cette étape pousse à comprendre comment construire une image complexe depuis une base minimaliste. Au lieu d'utiliser l'image officielle NGINX, une image Debian est utilisée et toutes les dépendances nécessaires sont installées pour faire fonctionner un serveur web.

Ce processus permettra de comprendre les mécanismes sous-jacents de la création d'images Docker et d'acquérir une compréhension approfondie de la configuration de services dans un environnement conteneurisé.

Registry Docker Local + UI

Registry privé

Stockage local pour vos images Docker personnalisées, évitant de les publier sur Docker Hub

Interface web

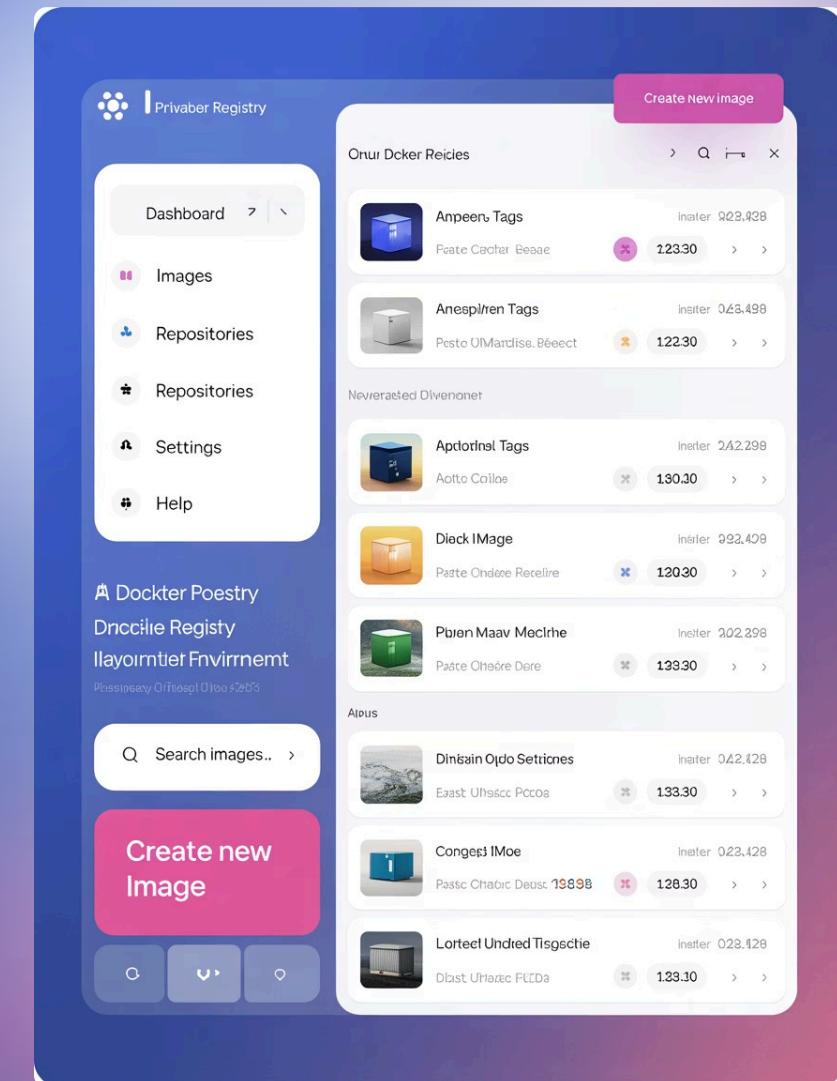
Console graphique pour visualiser et gérer facilement le contenu de votre registry

Sécurité

Contrôle total sur vos images avec possibilité d'ajouter une authentification

Un registry Docker local offre l'indépendance nécessaire pour développer et stocker vos propres images sans dépendre de services externes. Cela est particulièrement utile dans des environnements d'entreprise où la sécurité et la confidentialité sont primordiales.

L'ajout d'une interface utilisateur comme docker-registry-frontend facilite considérablement la gestion de ce registry en fournissant une vue d'ensemble claire de toutes vos images stockées et de leurs versions.



Scripts d'Automatisation Docker

Script d'installation

Ce script doit automatiser entièrement la mise en place de Docker :

- Mise à jour du système
- Installation des prérequis
- Ajout des clés GPG et du dépôt Docker
- Installation des packages Docker
- Configuration du démarrage automatique
- Ajout de l'utilisateur au groupe docker

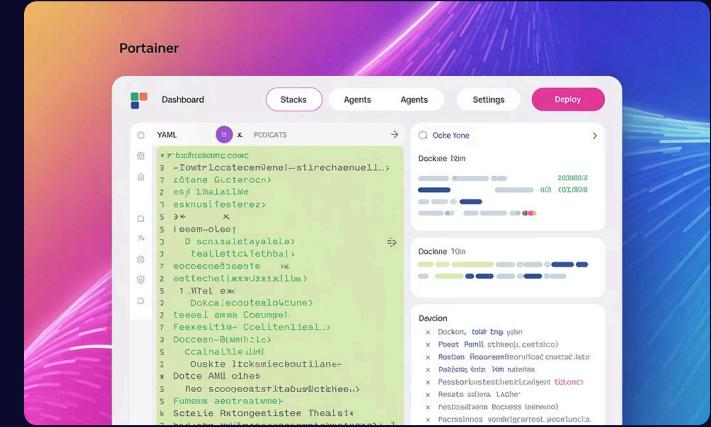
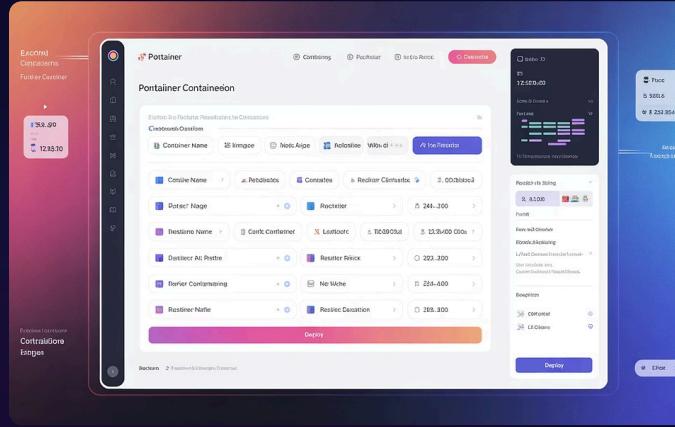
Script de désinstallation

Ce script doit nettoyer complètement l'environnement Docker en supprimant :

- Tous les conteneurs (actifs et arrêtés)
- Toutes les images téléchargées et créées
- Tous les volumes de données
- Les réseaux personnalisés
- Les packages Docker eux-mêmes

L'automatisation est un aspect fondamental de l'administration système moderne. Ces scripts vous permettront de gagner du temps et d'assurer la reproductibilité de vos installations Docker sur différentes machines.

Découverte de Portainer



Interface intuitive

Portainer offre une vue d'ensemble claire de tous vos conteneurs, réseaux et volumes avec des statistiques en temps réel sur l'utilisation des ressources.

Portainer transforme l'expérience d'utilisation de Docker en proposant une interface web complète pour gérer tous les aspects de votre environnement. Il est particulièrement utile pour les débutants mais offre également des fonctionnalités avancées pour les utilisateurs expérimentés.

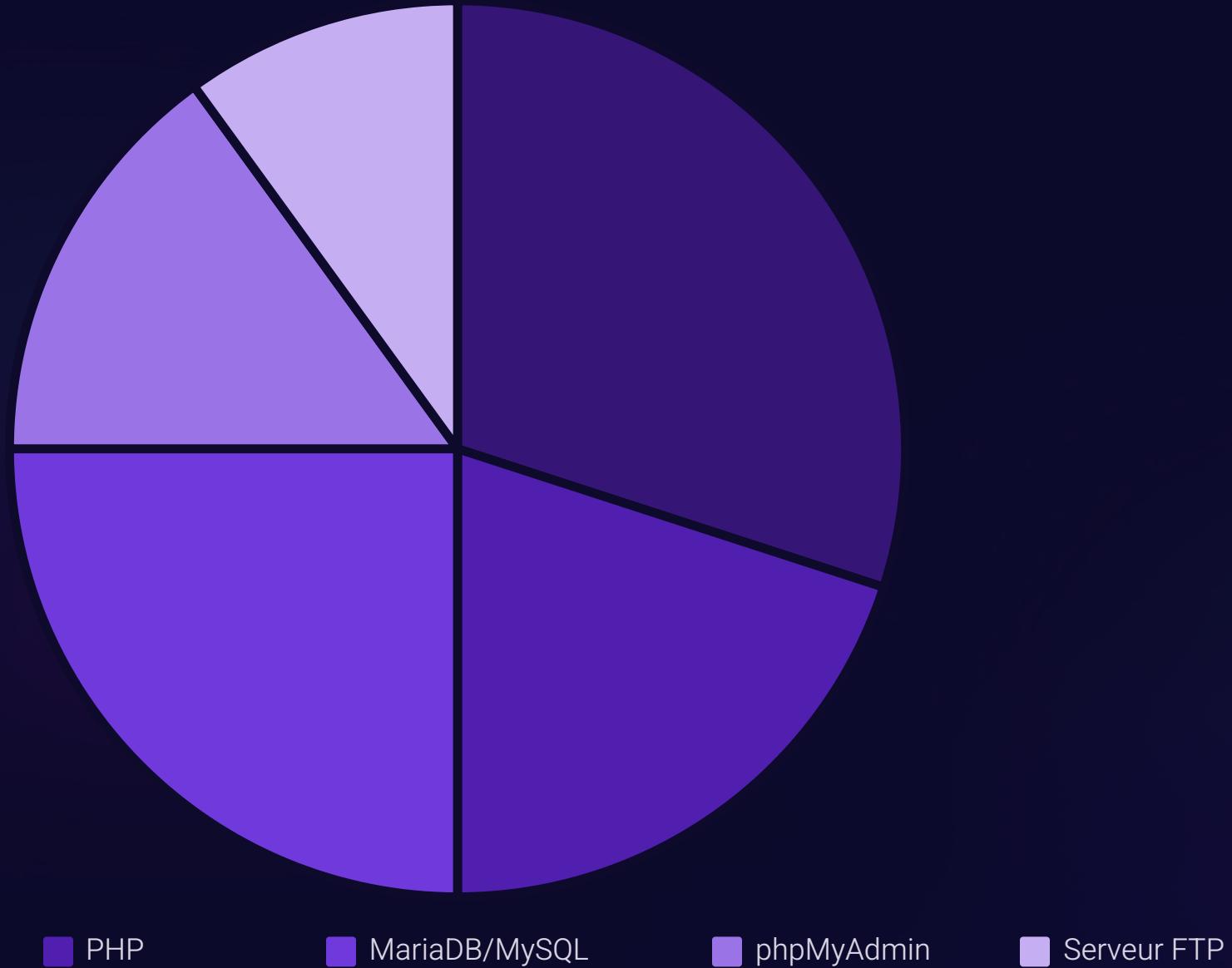
Création simplifiée

Créez et configurez des conteneurs via une interface graphique sans avoir à mémoriser la syntaxe des commandes Docker complexes.

Gestion des stacks

Déployez des applications multi-conteneurs en important directement vos fichiers docker-compose.yml dans l'interface web de Portainer.

Mini-XAMPP Dockerisé : Pour Aller Plus Loin



Ce projet final vous invite à créer un environnement de développement web complet en utilisant Docker. L'objectif est de reproduire les fonctionnalités de XAMPP, la suite populaire pour le développement PHP, mais de manière conteneurisée avec tous les avantages que cela implique : isolation, portabilité et facilité de déploiement.

Vous devrez orchestrer plusieurs services (serveur web, PHP, base de données, phpMyAdmin, FTP) et les faire communiquer entre eux, tout en partageant un volume commun pour les fichiers de développement. Ce projet synthétise toutes les compétences acquises précédemment et vous prépare à développer des architectures microservices complètes.